МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

ЦЕНТР ЗАОЧНОЇ, ДИСТАНЦІЙНОЇ ТА ВЕЧІРНЬОЇ ФОРМ НАВЧАННЯ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

на тему:

«Інформаційна система планування ресурсів підприємства на прикладі компоненти "виставлення рахунків"»

Завідувач
випускаючої кафедри **Довбиш А.С.**

Керівник роботи **Берест О.Б.**

Студента групи ІН.мз–91с **Чайка Н.І.**

СУМИ 2020

Сумський державний університет
(назва вузу)

Факультет *ЦЗДВФН* Кафедра *Комп'ютерних наук*
Спеціальність *«Інформатика»*

Затверджую:
зав.кафедрою ____________________________
____
"________" ______________________ 20___р.

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЕКТ СТУДЕНТОВІ

Чайці Наталії Іванівні
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) *«Інформаційна система планування ресурсів підприємства на прикладі компоненти "виставлення рахунків"»*

затверджую наказом по інституту від "____" ________________ 20 ___ р. № _____________

2. Термін здачі студентом закінченого проекту (роботи) ________________________________

3. Вхідні данні до проекту (роботи) *технічне завдання на розробку програмного модуля*

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) *1) Аналіз області систем ERP. 2) Постановка задачі розробки програмного модуля. 3) Моделювання програмного модуля звітності. 4) Практична реалізація фінансового модуля звітності ERP системи Odoo.*

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

Модулі систем планування ресурсів підпреємства, блок-схеми роботи компонент, діаграми моделювання, структура модулів, логіка роботи модуля, демонастрація роботи розробленого програмного рішення

6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання ____

Керівник ______________________
(підпис)

Завдання прийняв до виконання ______________________
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів дипломного проекту (роботи)	Термін виконання проекту (роботи)	Примітка
1.	*Дослідження предметної області*	*10.09.20 – 15.09.20*	
2.	*Вибір функціоналу програмного модуля*	*15.09.20 – 20.09.20*	
3.	*Визначення технічних засобів реалізації*	*20.09.20 – 25.09.20*	
4.	*Дослідження існуючих модулів*	*25.09.20 – 05.10.20*	
5.	*Розробка макету програмного модуля*	*05.10.20 – 12.10.20*	
6.	*Практична реалізація проекту*	*12.10.19 – 15.11.20*	
7.	*Тестування розробленого модуля*	*15.11.20 – 20.11.20*	
8.	*Оформлення дипломної документації*	*20.11.20 – 30.11.20*	

Студент – дипломник ______________________
(підпис)

Керівник проекту ______________________
(підпис)

РЕФЕРАТ

Записка: 49 стор., 28 рис., 1 додаток, 21 джерела.

Об'єкт дослідження — інформаційні системи планування ресурсів підприємства.

Мета роботи — розробка програмного модуля, який дозволяє автоматизувати процес отримання звітності з виставлених рахунків у системі ERP Odoo.

Методи дослідження — методи програмної автоматизації систем планування ресурсів підприємств.

Результати — розроблений програмний модуль, що дозволяє фінансовим спеціалістам швидко та легко отримати дані з виставлених рахунків та дивитися срок їх оплати замовниками.

СИСТЕМА ПЛАНУВАННЯ РЕСУРСІВ ПІДПРИЄМСТВА,
РАХУНКИ, МОДУЛЬ, ЗВІТНІСТЬ,
ERP, SAP, NETCRACKER, ODOO, DOCKER, XML, PYTHON.

ЗМІСТ

ВСТУП

ERP-системи дуже стрімко розвиваються та є невід’ємним інструментом сучасної IT-компанії. Навіть такі ключові корпорації-гіганти як Amazon, Apple, Google, Samsung використовують ERP. Але в чому ж унікальність ERP?

Абревіатура ERP походить від англійського (Enterprise Resource Planning System), що перекладається як система планування ресурсів підприємства. Продукт являє собою цілісну систему управління компанії, яка об'єднує ресурси підприємства по різних підрозділах. Тобто всі необхідні ресурси, підрозділи, функції та інший інструментарій, необхідний для ефективної роботи, знаходиться в одній комп'ютерній системі. Доступ до інформації отримують всі підрозділи на підприємстві, що істотно спрощує роботу і забезпечує обмін інформацією для більш точного управління.

Існуючі програмні рішенні застосовуються не лище в IT-індустрії, а й широко використовуються у медицині, банківській справі, транспортній сфері, освіті, науці та страхуванні [4].

Розробка модуля звітності для фінансового компонента є досить актуальною задачею, адже виставлення рахунків за виконані роботи – важлива складова ведення бізнесу для кожної компанії внезалежності від сфери застосування, а відстеження їх своєчасної сплати замовником є гарантом стабільної фінансової ситуації компанії.

Метою даного дипломної роботи є аналіз існуючих фінансових модулів та розробка модулю звітності для компоненти «виставлення рахунків» клієнтам.

Для досягнення поставленої мети необхідно виконати наступні задачі:

- проаналізувати предметну область ERP, обрати методи та засоби реалізації задачі;
- створити функціонал звітності для фінансової компоненти «виставлення рахунків»;
- створити демо-презентацію розробленого функціоналу.

Дипломний проект орієнтований на малий та середній бізнеси. Призначений для використання як менеджерами проектів так і персоналом для перевірки сплати виставлених рахунків по виконаним роботам.

Практичне значення дипломного проекту – розробити функціональний модуль інформаційної системи ERP.

1 АНАЛІЗ ОБЛАСТІ ERP СИСТЕМ

Що таке ERP і навіщо це потрібно?

ERP (скор. Від Enterprise Resource Planning) означає "планування ресурсів підприємства". Це програмне забезпечення для управління бізнес-процесами, яке об'єднує фінанси, ланцюжки поставок, операції, звітність, виробництво, кадри і дозволяє управляти ними. Більшість компаній мають у своєму розпорядженні системами управління фінансами і бізнес-процесами, але можливості цього ПЗ, як правило, обмежуються повсякденною рутиною або планами майбутнього розвитку.

Коли потреби компанії розширюються, системи управління повинні розвиватися разом з ними. Ми визначимо вимоги до ERP-системі і з'ясуємо, чому корисно мати ПЗ, що відповідає потребам бізнесу.

Рисунок 1 – Схема модулів класичної ERP-системи

Чим ERP система відрізняється від інших?

Нерідко ERP плутають з іншими прикладними рішеннями і програмними продуктами, які допомагають у вирішенні тих чи інших бізнес-завдань. Наприклад, через незнання ERP вважають аналогом CRM системи або альтернативою програмам бухгалтерського і податкового обліку. Основна відмінність ERP полягає в тому, що вона дозволяє управляти всіма ресурсами підприємства, а не окремими його частинами. Відрізнити її від інших прикладних рішень просто, адже система:

• інтегрує завдання і бази даних всіх відділів компанії;

• забезпечує створення єдиного інформаційного середовища;

• допомагає у вирішенні будь-яких завдань на підприємстві.

Як ERP система може поліпшити роботу компанії?

Що таке ERP? Це оптимальне рішення для підприємств, незалежно від їх масштабів, наявності філій і їх віддаленості. Система дозволяє:

• значно прискорити документообіг між відділами;

• отримати швидкий доступ до інформації;

• ефективно управляти роботою віддалених відділів, філій, співробітників.

Компанії вибирають ERP, виходячи з основних переваг рішення і практичних міркувань.

З'єднання всіх фінансових даних воєдино

Коли керуючому компанією потрібно отримати фінансові дані, він може вивчити безліч звітів, де показники бувають різними, але всі вони виявляться вірними. Адже кожен підрозділ веде свій облік і вносить певний внесок у загальну справу. І не завжди зрозуміло, наскільки ефективні витрати в тому чи іншому підрозділі, а за допомогою ERP це буде набагато простіше визначити. База даних єдина, тому маніпуляції з інформацією виконати практично неможливо.

Встановлення стандартів для виробничих процесів

Це особливо актуально для великих підприємств, у яких філії віддалені один від одного (знаходяться в різних містах або навіть країнах і континентах). Так, в таких підрозділах можуть використовуватися різні методи обліку та комп'ютерні системи, і щоб не заплутатися у всьому цьому якраз і використовується ERP. За допомогою мережі інтернет легко отримати доступ до необхідної інформації, а об'єднана багатовалютна система допомагає скоротити кількість працюючого персоналу і спростити облікові процеси.

Встановлення стандартів в кадровій системі

У ERP є можливість об'єднати всі завдання, пов'язані з персоналом - підбір, перепідготовка, перспективи розвитку та інше. За допомогою системи стає простіше стежити за роботою кожного співробітника і відслідковувати ефективність і його користь для компанії.

Коли необхідно впровадження ERP системи?

На перших етапах існування підприємства необхідності у впровадженні ERP системи немає. Всі необхідні документи працівники розробляють в офісних програмах, а обмін інформацією проходить швидко (достатньо зв'язатися з потрібним співробітником). Але в міру зростання підприємства, коли утворюються різні відділи (кадровий, бухгалтерський та інші), в кожному з них формується власна база даних. Це згодом ускладнює обмін інформацією між відділами, що гальмує роботу.

В таких умовах високої ефективності управління, оптимізації ресурсів і збільшення продуктивності буде домогтися дуже складно. На допомогу прийде ERP система, яка формує єдину базу даних для всіх підрозділів компанії.

1.1 Загальний огляд існуючих ERP систем

Сучасний IT-світ налічує безліч компаній від малого бізнесу (стартапів) до величезний корпорацій з великою кількістю філій по всьому світу.

Для прикладу пропонується розглянути такі ERP-системи як SAP та ERP.Netcracker.com.

SAP

SAP розпочав свою діяльність як постачальник програмного забезпечення для ERP і сьогодні є провідним гравцем на ринку ERP.

SAP розвинув тісні стосунки з різними партнерами по альянсу, що сприяло його зростанню до 1990-х та 2000-х років. Існує велика кількість сторонніх розробників, які постачають численні додаткові програми, які працюють разом із продуктами SAP. SAP також пропонує ERP рішення, що підходять для всіх розмірів організацій.

На основі технологічної платформи SAP NetWeaver, SAP Business Suite – це набір інтегрованих бізнес-додатків, що забезпечують специфічну для галузі функціональність та масштабованість. Хоча дуже потужний, SAP може бути важче змінити в міру розвитку бізнесу. Це одночасно сила та слабкість: з одного боку, вона тісно інтегрована та допомагає забезпечити стандартизовані бізнес-процеси на підприємстві, але це може бути і складніше модифікувати програмне забезпечення для адаптації до нових основних процесів та вимог. Основні пропозиції SAP включають SAP Business All-in-One та SAP Business One [21].

SAP Business All-in-One – це комплексне, інтегроване корпоративне програмне забезпечення, яке пропонує галузево орієнтовані рішення. Все-в-одному фокусується на малих та середніх організаціях, які мають до 2500 співробітників. SAP Business All-in-One базується на шаблонах і налаштовується похідною SAP Business Suite. Він пропонує понад 700 галузевих рішень, впроваджуючи їх "Кращі практики."

SAP Business One – це єдиний інтегрований додаток, розроблений для невеликих організацій з менше 100 працівників. В основному він підтримує роздрібну торгівлю, оптову торгівлю, послуги та виробництво. За допомогою сторонніх доповнень SAP Business One здатний підтримувати різні галузі та функції.

Для задоволення потреб малого та середнього бізнесу SAP пропонує SAP ByDesign. Доступно в США, Німеччині, Франції, Великобританії, Індії та Китаї, SAP ByDesign підтримує організації із 100 - 500 співробітниками. Як SaaS-тип на замовлення система SAP ByDesign має низькі початкові витрати і може зажадати менше IT-ресурсів, ніж традиційне програмне забезпечення ERP [21].

Грунтуючись на якісному та кількісному вкладі наших клієнтів, а також на власному впровадженні досвід, деякі функціональні переваги SAP включають:

- Потужна функціональність розробки продукту;
- Простота в підтримці обробки замовлення;
- Вбудований роздрібний модуль;
- Чітка видимість замовлень на транзит;
- Хороший контроль якості та функціональність забезпечення якості;
- Належне дотримання вимог SOX та податкових норм;
- Потужна функціональність управління готівкою.

На рисунку нижче представлені стандартні модулі ERP-системи SAP.

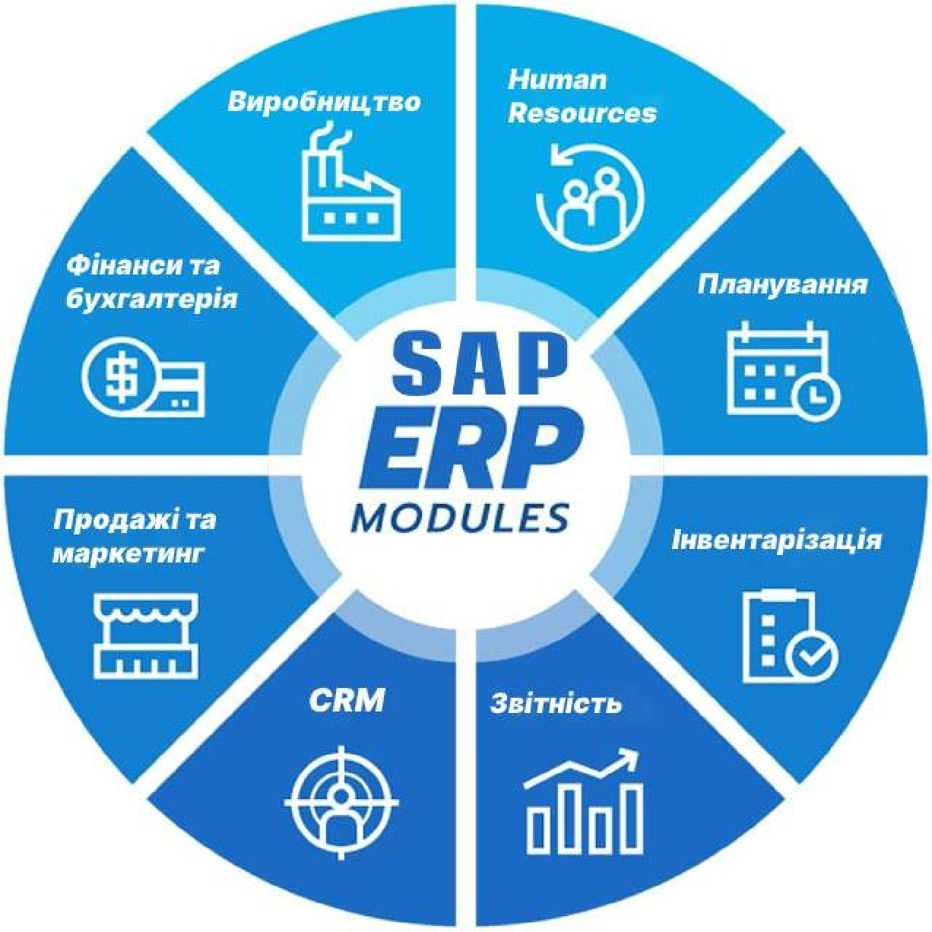

Рисунок 1.1 – Програмні Модулі SAP

ERP.Netcracker.com

Система ERP була вперше представлена у 2004 році у компанії Netcracker. Основна мета використання ERP - об'єднати інформацію та процеси з усіх підрозділів організації та об'єднати їх для створення структурованої робочої системи.

- E (Enterprise) Підприємство - це група людей із спільною метою, яка має певні ресурси у своєму розпорядженні для досягнення цієї мети.
- R (Resource) Ресурси - MMM (Man, Money, Material – Людина, Гроші, Матеріали), необхідні для управління підприємством.
- P (Planning) Планування проводиться для того, щоб нічого не пішло не так.

ERP.netcracker.com - це внутрішня система управління проектами Компанії Netcracker. Кожен працівник має доступ до системних послуг відповідно до своїх посадових обов’язків та посади.

ERP.Netcracker.com включає всі функції, пов'язані з корпоративними системами планування ресурсів:

• Запис TER (внесення робочого часу та витрат);
• Відстеження проектів;
• Доставка ліцензії;
• Інвентар;
• Система виставлення рахунків замовникам;
• Система відстеження замовлень на розгортання IT;
• Візова та туристична підтримка;
• Управління ресурсами.

Зовні система доступна, якщо встановлено з’єднання VPN.

На рисунку нижче представлені модулі ERP.Netcracker.com

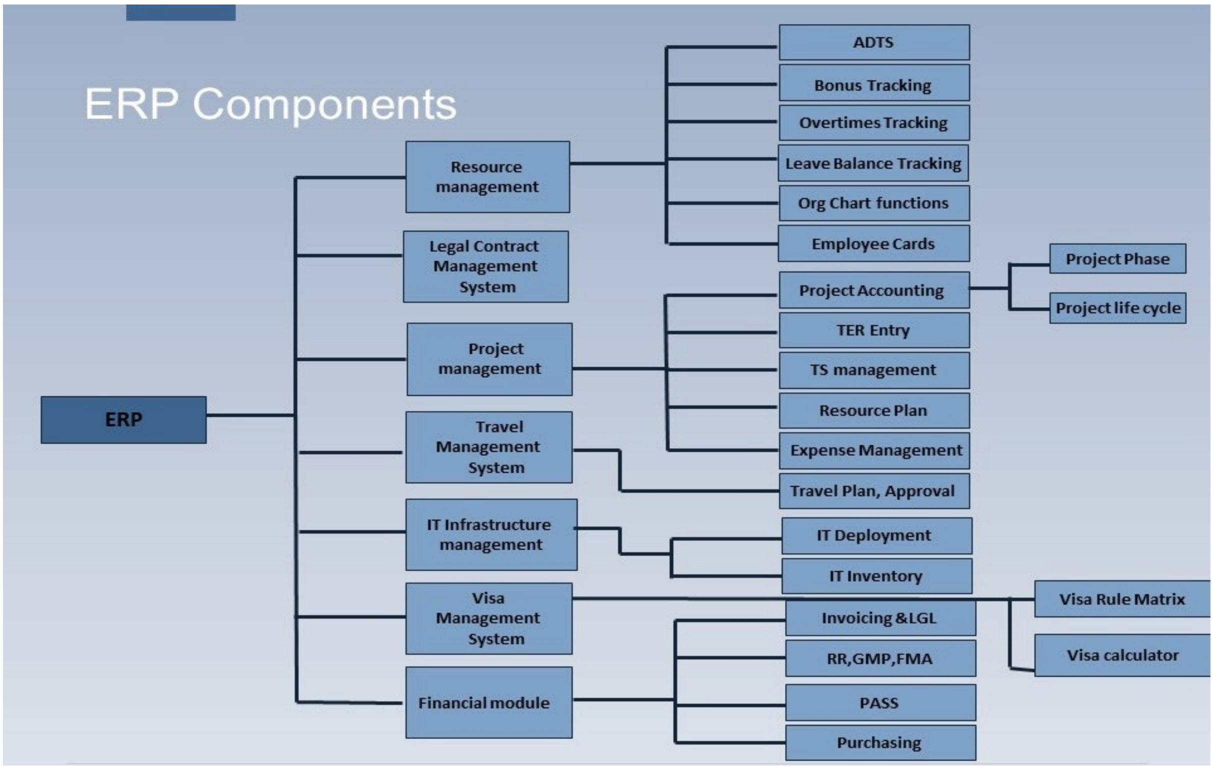

Рисунок 1.1.1 – Модулі ERP.Netcracker.com

1.2 Вибір об'єкту проектування

Чому ERP-система критично важлива для бізнесу

Поки що не існує ідеального програмного рішення для всіх бізнес-процесів, але технології ERP все краще об'єднують їх, удосконалюючи спільну роботу, допомагаючи приймати зважені рішення на основі даних і підвищуючи ефективність бізнесу.

ERP охоплює безліч функцій компанії. Наприклад фінанси:

Сучасні ERP-системи пропонують огляд фінансових показників в режимі реального часу з будь-якого пристрою. Це також допоможе відмовитися від введення даних вручну: автоматизувати щоденні завдання і відстежувати відповідність бізнесу нормативним вимогам.

Фінансова компонента має тісний зв'язок з багатьма іншими компонентами системи, такими як:

- репозитарій контрактів;
- сторінка проектного коду;
- модуль внесення робочого часу та витрат у відрядженнях;

Умовно процес виставлення рахунків можно розділити на такі кроки:

- Внесення підписаного контракту про проведення певних робіт у систему «репозитарій контрактів», оформлення картки документу та створення майлстоунів для подальшого виставлення рахунків;
- Створення сторінки проектного коду у компоненті «проекти». Проектні коди бувають TM (time and material), що базується на внесенні часу співробітників у систему та Fx – узгоджений певний бюджет на розробку, які у свою чергу формуються щомісячно (Date Driven) або за умов надання письмової згоди замовника (Based on Acceptance);
- Безпосередньо сам процес оформлення рахунку за проведення певних виконаних робіт (згідно з умовами контракту).

Нижче зображена блок-схема функціонування компоненти виставлення рахунків у системі ERP.Netcracker.com, що є основним орієнтиром для розробки майбутнього об’єкту дипломної роботи – модуля фінансової звітності.

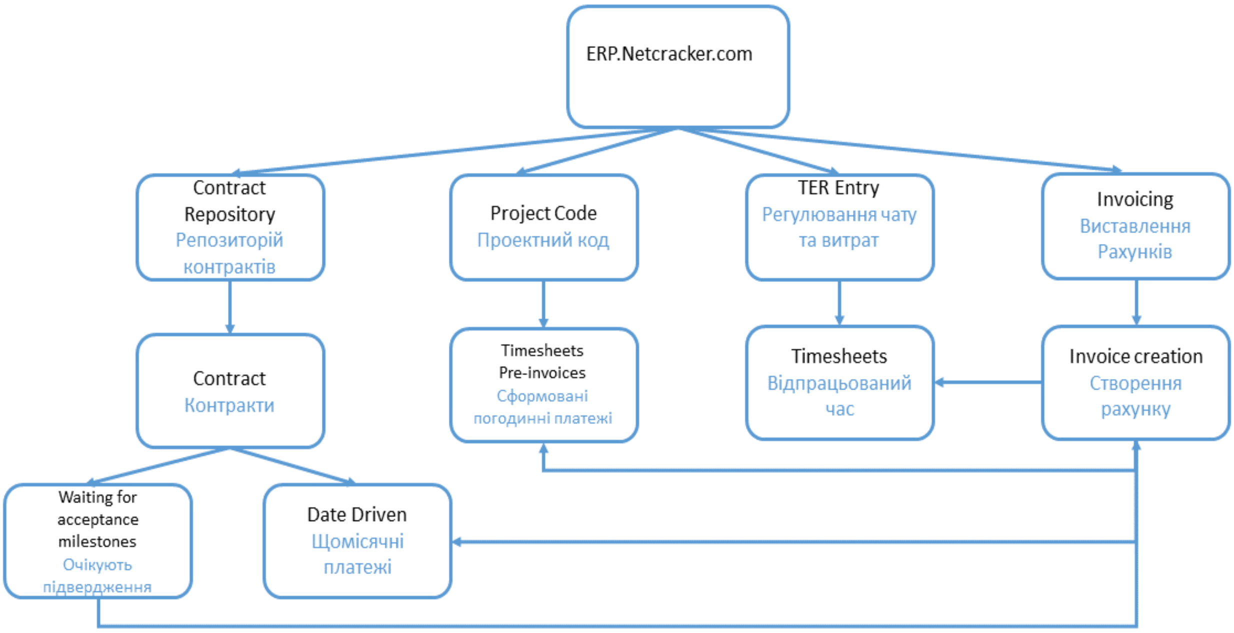

Рисунок 1.2 – Блок-схема процесу виставлення рахунку у системі ERP.Netcracer.com

Після формування рахунка, важливо мати змогу відстежувати статус їх оплати. Даний звіт можна отримати через інший модуль «Звітності» у системі ERP.Netcracer.com.

Тому роблячи висновки, що звітність досить важлива для кожної компанії, а розроблення зручного та швидкого модуля для є актуальним об’єктом проектування дипломної роботи.

2 ПОСТАНОВКА ЗАДАЧІ РОЗРОБКИ ПРОГРАМНОГО МОДУЛЯ

2.1 Мета та задачі

Метою роботи є розробка програмного модуля для автоматизації створення звітності з виставлених рахунків у відкритій ERP-системі, що базується на Odoo [19]. Даний модуль з легкістю можна налаштувати та використовувати, зручність інтерфейсу із можливістю мануально додавати необхідних конфігурацій ERP-системи із шаблонів або налаштувань з можливістю редагування на будь-яких етапах встановлення чи видалення пакетів програмного забезпечення.

Використання даного модуля може скоротити витрати часу та підвищити якість роботи за рахунок мінімізації ймовірності помилок фінансових фахівців.

Розроблений модуль матиме наступні функції:

- Групування даних щодо виставлених рахунків у тимчасовий xls файл за допомогою моделі TransientModel [20] та бібліотек Python IO, що включає дві вкладки: перша – усі рахунки замовникам, друга – сумарні рахунки по кожному замовнику окремо;
- Можливість обирати період для звітності (за замовчуванням встановленого початок року до теперішньої дати);
- Обирати статус рахунків («всі», «сплачені», «не сплачені»);
- Підтримує мультивалютність для одного замовника (додає нові строки для інших валют та сумує дані по різним валютам окремо).

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

Задля реалізації проекту треба вирішити такі задачі:

- окреслити функціонал програмного модуля;

- вибрати технічні засоби реалізації;
- розробити код модуля;
- сформувати шаблони налаштувань проекту;
- організувати тестування розробленого модуля.

Схема роботи модуля є такою:

1. Після встановлення модулю в меню встановленого модуля «Виставлення рахунків» з’являється додатковий пункт «Звітність», яка включає аналітику під розділом «Рахунки» та власне модуль звітності «Роздрукувати звіт про рахунки».
2. Далі модуль «Звітності» дозволяє обрати необхідний період виставлених рахунків та їх статус.
3. Встановивши необхідні критерії система дозволяє вигрузити звіт за допомогою бібліотек Python «IO» не обтяжуючи базу даних.

2.2 Вибір засобів реалізації

Для реалізації програмного модуля було обрано наступні засоби: JavaScript, HTML5, CSS3 (візуальна частина), інтерпритовану об’єктно-орієнтовану мову програмування – Python (функціональная частина), PostgreSQL (база даних), середовище розробки Docker та стандарт XML для зберігання налаштувань проекту.

Архітектура модуля складається з 3 рівнів:

1. Клієнтська/візуальна частина (HTML5, CSS3, JavaScript)
2. Функціонально-логічна частина модуля (Python)
3. База даних (PostgreSQL)

Нижче представлені засоби реалізації цих рівнів:

Презентаційний рівень (клієнт) - HTML5, CSS3, JavaScript.

Прикладний рівень (серверна частина) - Python3.

MVC (Model, View, Controller) framework - один із найбільш часто використовуваних у світі веб-розробок для створення найбільш масштабованих та розширюваних проектів [9].

Модель (Model) - логіка даних, з якою працює користувач.

Клієнтський об'єкт отримує інформацію про клієнта з бази даних, маніпулює нею та оновлює ці дані назад до бази даних або навіть використовує їх для візуалізації даних на інтерфазі.

Перегляд (View) - використовується для всієї логіки інтерфейсу інтерфейсу. Подання клієнта включатиме всі компоненти інтерфейсу користувача, такі як текстові поля, випадаючі меню тощо, з якими взаємодіє кінцевий користувач.

Контролер (Controller) виступає в ролі інтерфейсу між моделлю та видом, обробляє всю бізнес-логіку та запити. Клієнтський контролер буде обробляти всі взаємодії та входи з подання форми клієнта та оновлювати базу даних за допомогою модуля клієнта, і той самий контролер буде використовуватися для перегляду даних клієнта.

Рівень даних (база даних) – PostgreSQL.

Серверні модулі розробляються на Python.

ORM (Object, Relational, Mapping) для взаємодії з базою даних є центральною частиною Odoo [11].

Модуль даних описується та маніпулюється за допомогою класів та об’єктів python. Завданням ORM є максимально прозоре подолання розриву для розробника між Python та базою даних PostgreSQL, що забезпечує постійність, необхідну для наших об'єктів.

JavaScript (JS) — динамічна, об'єктно-орієнтована прототипна мова програмування. Реалізація стандарту ECMAScript. Найчастіше використовується для створення сценаріїв веб-сторінок, що надає можливість на боці клієнта (пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд веб-сторінки [13].

JavaScript класифікують як прототипну (підмножина об'єктно-орієнтованої), скриптову мову програмування з динамічною типізацією. Окрім прототипної, JavaScript також частково підтримує інші парадигми програмування (імперативну та частково функціональну) і деякі відповідні архітектурні властивості, зокрема: динамічна та слабка типізація, автоматичне керування пам'яттю, прототипне наслідування, функції як об'єкти першого класу.

Мова JavaScript використовується для:

- написання сценаріїв веб-сторінок для надання їм інтерактивності;
- створення односторінкових веб-застосунків (React, AngularJS, Vue.js);
- програмування на боці сервера (Node.js(Express.js));
- стаціонарних застосунків (Electron, NW.js);
- мобільних застосунків (React Native, Cordova);
- сценаріїв в прикладних програмах (наприклад, в програмах зі складу Adobe Creative Suite чи Apache JMeter);
- всередині PDF-документів тощо.

Мова розмітки HyperText (HTML) - це сучасна стандартна мова розмітки, яка використовує загальні скорочення, що називаються "теги", щоб вказати веб-браузеру, як автори хотіли б, щоб були викладені розділи веб-сторінки.

Каскадні таблиці стилів - коротше CSS - дають вам творчий контроль над макет та дизайном ваших веб-сторінок. Використовуючи CSS, ви можете вбрати свій текст сайту із привабливими заголовками, крапками та рамками. Ви також можете впорядковувати зображення з точністю, створювати стовпці та банери та виділіть свої посилання за допомогою динамічних ефектів перекидання. Можна навіть зробити елементи зникають або виходять з поля зору, переміщують об’єкти по сторінці або повільно змінюють кольори кнопки, коли відвідувач наводить курсор миші над нею [8].

Вся ідея CSS полягає в спрощенні процесу стилізації веб-сторінок.

В якості середовища розробки було обрано Docker — інструментарій для управління ізольованими Linux-контейнерами. Docker доповнює інструментарій LXC більш високорівневим API, що дозволяє керувати контейнерами на рівні ізоляції окремих процесів. Зокрема, Docker дозволяє не переймаючись вмістом контейнера запускати довільні процеси в режимі ізоляції і потім переносити і клонувати сформовані для даних процесів контейнери на інші сервери, беручи на себе всю роботу зі створення, обслуговування і підтримки контейнерів [14].

За зберігання даних, необхідних для роботи програмного модулю обрано стандарт XML через простоту редагування та інтеграцію з середовищем розробки. XML файли із загальною конфігурацією та окремими налаштуваннями конкретних проектів зберігаються на мережевому диску, до якого є доступ як із хостових систем, так і з віртуальних машин на підприємстві. Взаємодія з форматом xml та файловою системою в середовищі Docker здійснюється за допомогою простору імен print_invoice_summary_view.xml та io.BytesIO відповідно [3].

2.3 Порівняльний аналіз існуючих модулів

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

- час отримання сформованого звіту в системах;
- кількість виконаних кроків для отримання звіту.

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

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

Усі звіти>Фінанси>Архів (GAAP)>Рахунки та виставлення рахунків (GAAP)> Звіт про стан виставлених рахунків>Обрати період та статус

Після отримання звіту, потрібно за допомогою зведених таблиць створити необхідний звіт.

Витрати часу на підготовку звіту становлять 30 хвилин.

На рисунку нижче схематично зображено кроки виконання для отримання звіту в системі ERP.Netcracker.com

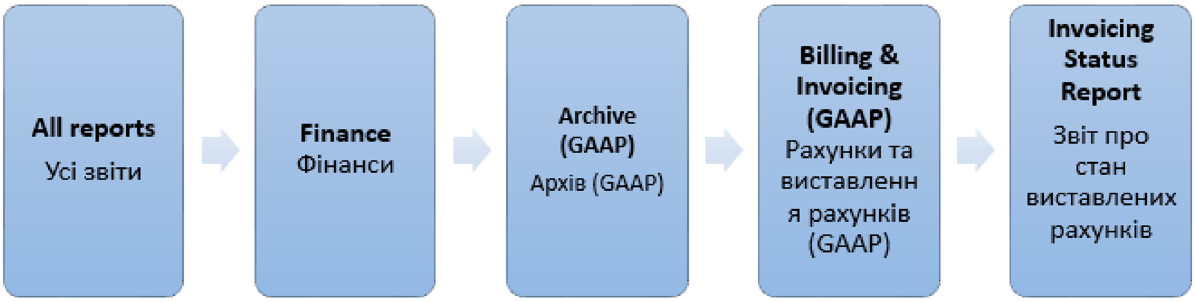

Рисунок 2.3 – Схема отримання фінансового звіту з рахунків в системі ERP.Netcracker.com

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

Invoicing>Reports>Invoice Summary Report

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

Приблизний час отримання звіту – 3 хвилини.

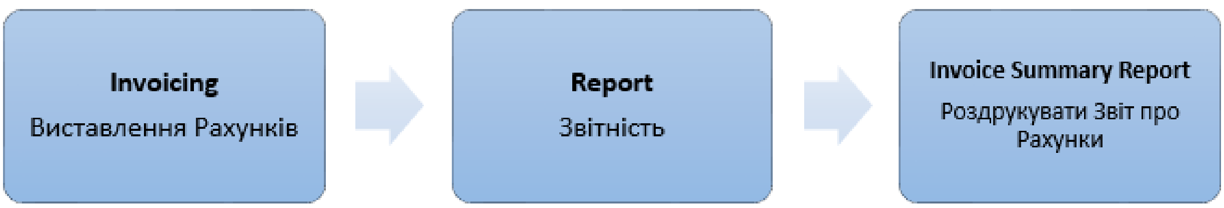

Рисунок 2.3.1 - Схема отримання фінансового звіту з рахунків в системі Odoo

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

3 МОДЕЛЮВАННЯ ПРОГРАМНОГО МОДУЛЯ ЗВІТНОСТІ

Програмний модуль – функціонально завершений фрагмент програми, оформлений у вигляді окремого файлу з сирцевим кодом або його іменованої частини, призначений для використання в інших програмах. Модулі дозволяють розбивати складні задачі на менші відповідно до принципу модульності. Зазвичай проектується таким чином, щоб надати програмістам зручну для багаторазового використання функціональність (інтерфейс) [18].

3.1 Структурно-функціональне моделювання

Для моделювання бізнес-процесів використовується кілька різних методів, основою яких є як структурний, так і об’єктно-орієнтований підходи до моделювання. Одним з найпоширеніших методів моделювання бізнес-процесів є метод функціонального моделювання SADT (IDEF0).

SADT-метод (Structured Analysis and Design Technique) вважається класичним методом процесного підходу до управління. Основний принцип даного підходу полягає у структуризації діяльності організації відповідно до її бізнес-процесів, який представляє собою сукупність правил і процедур, що призначені для побудови функціональної моделі об’єкту будь-якої предметної області. Функціональна модель SADT відображає функціональну структуру об’єкту, тобто дії, які він здійснює та зв’язки між цими діями [2]. IDEF0 – модель дає можливість отримати точну специфікацію усіх операцій та дій, які відбуваються у бізнес-процесі, а також характер взаємозв’язку між ними [2].

Найбільш важлива функція розташована у верхньому лівому кутку. А з'єднуються функції між собою за допомогою стрілок і описів функціональних блоків. При цьому кожен вид стрілки або активності має власне значення. Дана модель дозволяє описати всі основні види процесів, як адміністративні, так і організаційні.

Стрілки можуть бути:

- Вхідні – вступні, які ставлять певне завдання.
- Вихідні – виводять результат діяльності.
- Керуючі (зверху вниз) – механізми управління (положення, інструкції тощо).
- Механізми (знизу вверх) – що використовується для того, щоб виконати необхідну роботу.

В результаті використання IDEF0-моделювання створену діаграму дипломного проекту з подальшою декомпозицією.

Нижче представлені діаграми рівнів A-0, A0 та A1.

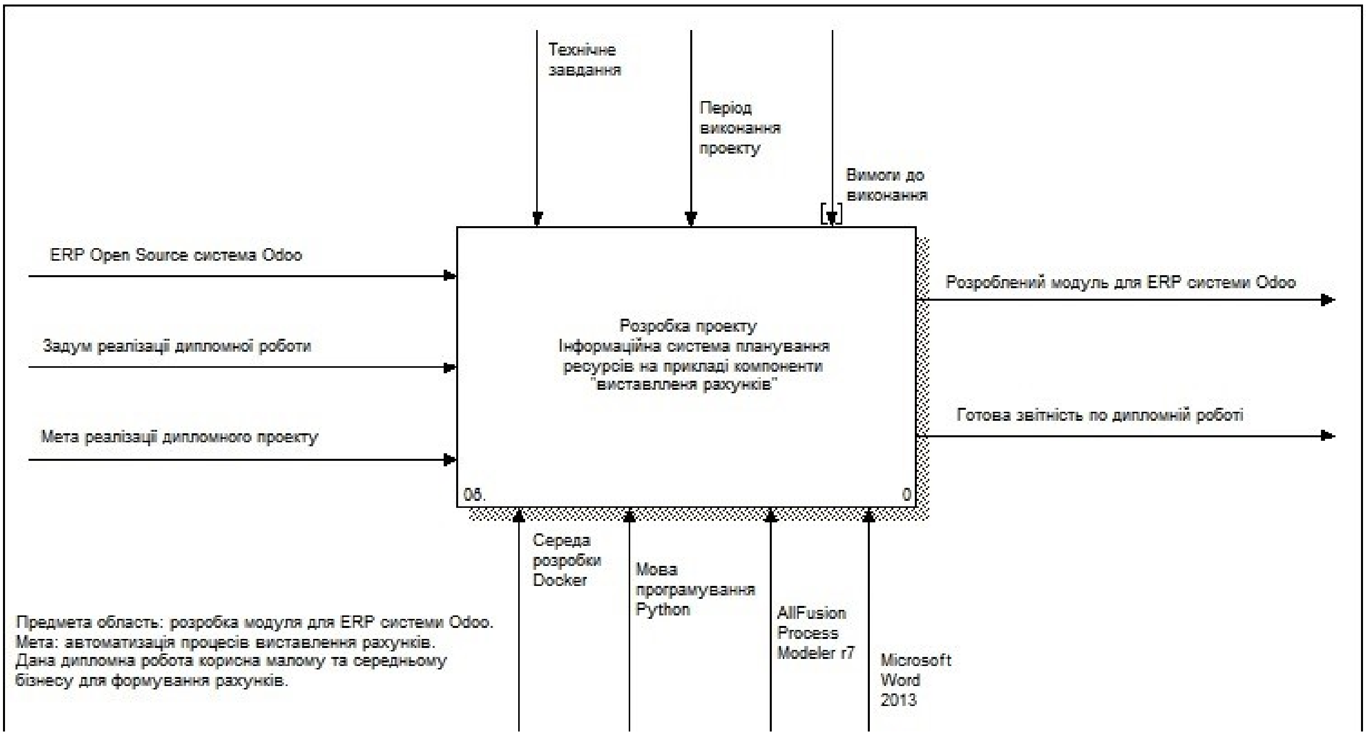

Рисунок 3.1 – Діаграма IDEF0 рівня A-0

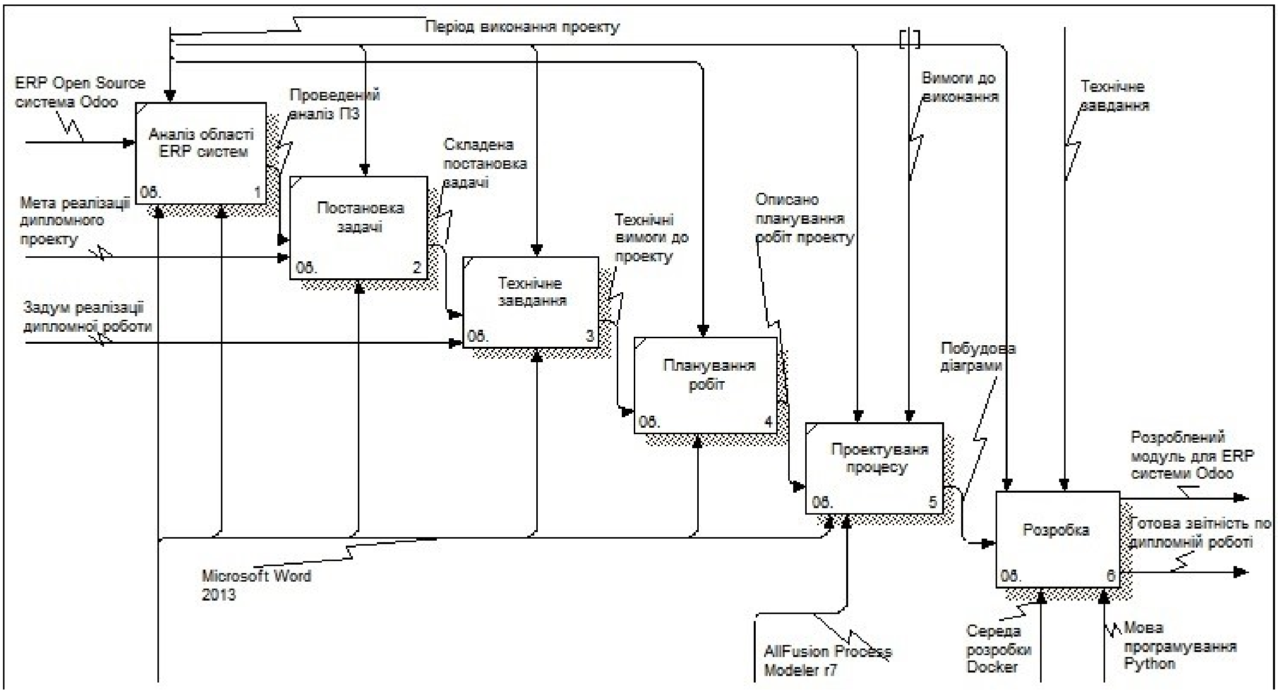

Рисунок 3.2 - Діаграма IDEF0 рівня A0

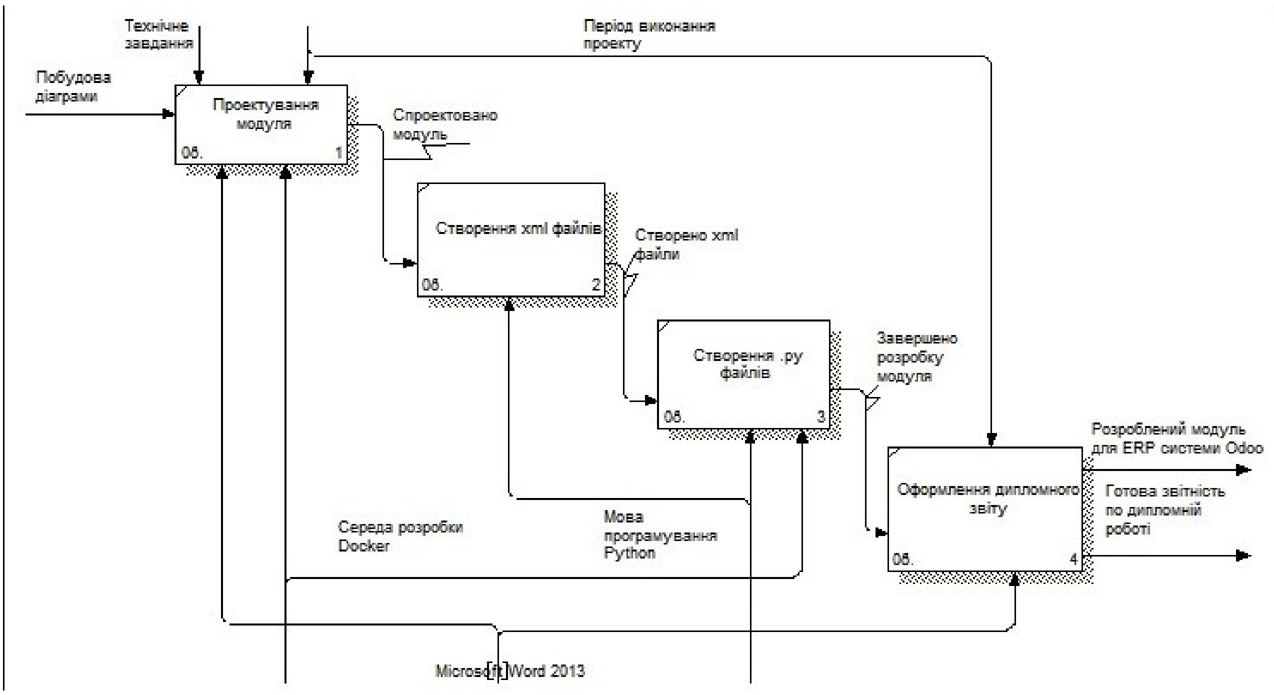

Рисунок 3.2 - Діаграма IDEF0 рівня A1

3.2 Структура інформаційної моделі

Odoo модулі можуть, як додавати абсолютно нову бізнес-логіку, так і розширювати існуючу: модуль може бути створений для того, щоб додати план рахунків відповідної країни таким чином, щоб Odoo міг підтримувати саме ваш тип бухгалтерського обліку. У той же час, інший модуль додає підтримку відображення стану в режимі реального часу [19].

Таким чином, все в Odoo починається і закінчується модулями.

Склад модуля

Модуль Odoo може містити наступні елементи:

1. Бізнес-об'єкти:

Оголошені як класи Python, Odoo автоматично зберігає ці ресурси на основі їх конфігурації

2. Дані

Файли XML або CSV, що декларують метадані (подання або робочі процеси), дані конфігурації (параметризація модулів), демонстраційні дані тощо.

3. Веб-контролери (Web controllers)

Обробляти запити від веб-браузерів

4. Статичні ресурси

Зображення, CSS або javascript-файли, які використовуються веб-інтерфейсом або веб-сайтом.

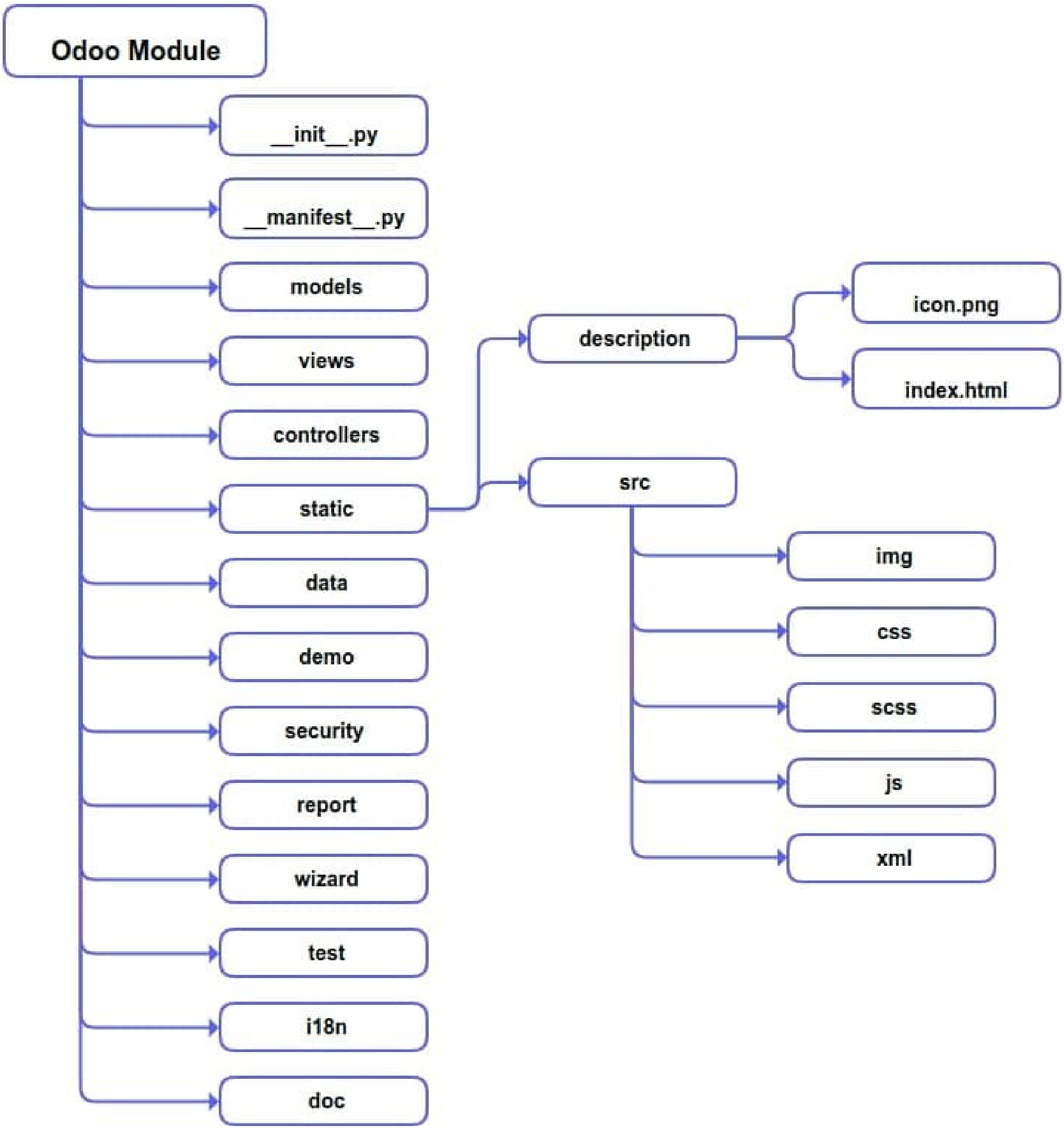

Рисунок 3.2 – Стандартна структура модулів ERP-системи Odoo

Структура каталогів модулів Odoo

__init__.py

У цьому файлі вам доведеться імпортувати або зареєструвати або навіть сказати ініціалізувати два елементи. По-перше, ви можете вказати каталоги, які містять файли python, які ми використовували в модулі Odoo. По-друге, визначає файли python, які безпосередньо існують у модулі Odoo (крім папок). Але вам слід уникати розміщення файлу python у кореневому шляху модуля.

Але файл __manifest__.py у файлі __init__.py. Оскільки це провідний файл в модулі Odoo, то він буде безпосередньо отримувати доступ із фреймворку Odoo для реєстрації модуля.

__manifest__.py

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

<u>У файлі маніфесту ви можете оголосити такі поля:</u>

- name – Назва модуля
- description – Короткий опис модуля
- version – Вкажіть версію модуля
- license – Вкажіть ліцензію на розповсюдження модуля
- author – ім'я автора модуля
- website – URL-адреса веб-сайту автора модуля
- category – Вкажіть назву категорії. Рекомендується використовувати існуючі категорії.
- depends – Вказує список модулів, які встановлюються першими перед установкою модуля.
- data – Список файлів даних, які завжди встановлюються або оновлюються разом із модулем.
- demo – Список файлів даних, які встановлюються або оновлюються в активному демонстраційному режимі.

Моделі (Models)

Каталог моделей містить усі ваші файли python (.py). Ці файли python містять моделі, які ви створюєте або успадковуєте існуючі моделі та основну бізнес-логіку.

Усі файли python (.py) у будь-яких каталогах, потрібно імпортувати / зареєструвати у файл __init__.py у цьому відповідному каталозі.

Погляди (Views)

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

Контролери (Controllers)

Цей каталог містить файли .py. Контролер - це клас, який походить від базового класу. Усі запити обробляються з браузерів і передаються на сервер. Це як середній міст для спілкування між моделями та поглядами.

Усі файли python (.py) у будь-яких каталогах, потрібно імпортувати/зареєструвати у файл __init__.py у цьому шанованому каталозі.

Статичний (Static)

Цей каталог містить опис і src два підкаталоги.

Опис: Цей каталог містить один файл icon.png, який представляє піктограму модуля у списку програм. А інший файл index.html містить документацію щодо модуля. Він надасть інформацію про функціональність модуля разом із скріншотом.

src: Ця папка містить багато підкаталогів.

• img - ця папка зберігає зображення, які ми використовуємо в модулі

• css - зберігає один або кілька файлів CSS для застосування стилів

• scss - зберігає один або кілька файлів SCSS для застосування стилів

• js - містить один або кілька файлів JavaScript

• xml - ця папка містить XML-файл, що містить збережений код QWeb. За допомогою JS він відобразиться.

Дані (Data)

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

Демо (Demo)

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

Безпека (Security)

Ця папка містить два файли. Файл контролю доступу та файл правил запису.

Контроль доступу: визначає права доступу моделей. Отже, вам потрібно створити файл ir.model.access.csv, який містить права доступу до моделей.

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

Доповідь (Report)

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

Модальне вікно (Wizard)

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

Тест (Test)

Файли для тестування зберігатимуться в цій папці. Усі тестові приклади - це файли .py та .yml із записом.

Локалізація (i18n)

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

Документація (Doc)

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

4 ПРАКТИЧНА РЕАЛІЗАЦІЯ ФІНАНСОВОГО МОДУЛЯ ЗВІТНОСТІ ERP СИСТЕМИ ODOO

4.1 Розробка конфігурації програмного модуля

View представлена xml (Розширюючі мову розмітки). Специфікація цієї мови дозволяє описувати поведінку xml програм, що зчитують дані фронт-енд і обробляють їх на бек-енді. Даний View містить записи з полями для вибору даних, які будуть опрацьовані в моделі.

```
<record id="view_print_invoice_summary_form" model="ir.ui.view">
  <field name="name">print.invoice.summary.form</field>
  <field name="model">print.invoice.summary</field>
  <field name="type">form</field>
  <field name="arch" type="xml">
    <form string="Роздрукувати звітний звіт про рахунок">
          <field name="invoice_report_printed" invisible="1"/>
            <group  attrs="{'invisible':[('invoice_report_printed','=',True)]}">
              <group>
                <field name="from_date" required="1" style="width:250px;"/>
                <field name="to_date" required="1" style="width:250px;"/>
                </group>
                <group>
                  <field name="invoice_status" required="1" style="width:250px;"/>
               </group>
             </group>
          <group attrs="{'invisible':[('invoice_report_printed','=',False)]}">
              <field name="file_name" invisible="1"/>
            <field name="invoice_summary_file" readonly="1"  filename="file_name"/>
          </group>
        <footer>
        <button string='Роздрукувати'  name="action_print_invoice_summary"
          type="object" class="btn-primary" attrs="{'invisible':[('invoice_report_printed','=',True)]}"/>
          <button string="Скасувати" class="oe_link" special="cancel"/>
        </footer>
    </form>
  </field>
</record>
```

Рисунок 4.1 – Зміст файлу print_invoice_summary_view.xml

Також представлені інтерактивні кнопки для підтвердження виклику обробки або скасування рендеру xml форми.

```
<footer>
<button string='Роздрукувати'  name="action_print_invoice_summary"
  type="object" class="btn-primary" attrs="{'invisible':[('invoice_report_printed','=',True)]}"/>
  <button string="Скасувати" class="oe_link" special="cancel"/>
</footer>
```

Рисунок 4.1.1 – Відправка або скасування запросу на формування звіту

Представлено розділ в меню, який додається в існуючий модуль Invoicing («Виставлення Рахунків»).

```
<menuitem  id="menu_print_invoice_summary_report" action="action_print_invoice_summary"
 sequence="205" parent="account.menu_finance_reports" />
```

Рисунок 4.1.2 – Додавання розділу «Звітність» в меню модуля «Виставлення Рахунків»

4.2 Розробка логіки роботи програмного модуля

Спочатку відбувається імпорт пакетів Python з ERP-системи Odoo.

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

Рисунок 4.2 – імпорт пакетів Python з ERP-системи Odoo

Даний модуль включає клас PrintInvoiceSummary, що посиалається на TransientModel [20].

```
class PrintInvoiceSummary(models.TransientModel):
    _name = "print.invoice.summary"
```

Рисунок 4.2.1 – клас PrintInvoiceSummary

TransientModel - це клас, який тимчасово зберігає дані в базі даних і автоматично очищається протягом певного періоду часу.

Повний опис цього класу, доданий Odoo в коді:

Модель супер-класу для перехідних записів, призначена для тимчасового зберігання та регулярної очистки.

TransientModel має спрощене управління правами доступу, всі користувачі можуть створювати нові записи та мати доступ лише до записів, які вони створили. Суперкористувач має необмежений доступ до всіх записів TransientModel.

Крім того, коли ви працюєте з цим типом класів, враховуйте, що відносини Many2one заборонені з неперехідною моделлю.

Цей тип моделі в основному використовується для створення Wizards в Odoo.

У даному випадку ця функція бере дані про компанії з бази даних Odoo. За замовчуванням визначає поточну дату фінансового року компанії, або можна вибрати будь-яку необхідну дату в формі вручну. Функція нижче дозволяє виконати підстановку необхідних даних і вибрати статус рахунків.

```
@api.model
def _get_from_date(self):
    company = self.env.user.company_id
    current_date = datetime.date.today()
    from_date = company.compute_fiscalyear_dates(current_date)['date_from']
    return from_date

from_date = fields.Date(string='З',default=_get_from_date)
to_date = fields.Date(string='До',default=datetime.date.today())
invoice_summary_file = fields.Binary('Звітність з рахунків')
file_name = fields.Char('File Name')
invoice_report_printed = fields.Boolean('Звіт про рахунок-фактуру надруковано')
invoice_status = fields.Selection([('all', 'Всі'), ('paid', 'Оплачені'),('un_paid', 'Невиплачені')], string='Статус рахунку')
```

Рисунок 4.2.2 – встановлення періоду та статусу для формування звіту

Далі в залежності від обраних параметрів формується звіт у форматі xls, який генерується за допомогою бібліотек Пітона:
xlwt (створення і заповнення файлів Excel);
easyxf (вирівнювання тексту) [16].

Файл містить 2 листа: 1 - дані всіх інвойсів, 2 - дані всіх інвойсів по замовниках.

```python
    @api.multi
    def action_print_invoice_summary(self):
        new_from_date = self.from_date.strftime('%Y-%m-%d')
        new_to_date = self.to_date.strftime('%Y-%m-%d')


        workbook = xlwt.Workbook()
        amount_tot = 0
        column_heading_style = easyxf('font:height 250;font:bold True;')
        worksheet = workbook.add_sheet('Сумарна звітність')
        worksheet.write(2, 3, self.env.user.company_id.name, easyxf('font:height 250;font:bold True;align: horiz center;'))
        worksheet.write(4, 2, new_from_date, easyxf('font:height 250;font:bold True;align: horiz center;'))
        worksheet.write(4, 3, 'до',easyxf('font:height 250;font:bold True;align: horiz center;'))
        worksheet.write(4, 4, new_to_date,easyxf('font:height 250;font:bold True;align: horiz center;'))
        worksheet.write(6, 0, _('Номер рахунку'), column_heading_style)
        worksheet.write(6, 1, _('Клієнт'), column_heading_style)
        worksheet.write(6, 2, _('Дата рахунку'), column_heading_style)
        worksheet.write(6, 3, _('Сума рахунку'), column_heading_style)
        worksheet.write(6, 4, _('Валюта рахунку'), column_heading_style)
        worksheet.write(6, 5, _('Сума у валюті компанії (' + str(self.env.user.company_id.currency_id.name) + ')'), column_headin

        worksheet.col(0).width = 5000
        worksheet.col(1).width = 5000
        worksheet.col(2).width = 5000
        worksheet.col(3).width = 5000
        worksheet.col(4).width = 5000
        worksheet.col(5).width = 8000
```

Рисунок 4.2.3 – формування листа 1.

```python
worksheet2 = workbook.add_sheet('Підсумок рахунків-фактур для клієнтів')
worksheet2.write(1, 0, _('Клієнт'), column_heading_style)
worksheet2.write(1, 1, _('Сплачена сума'), column_heading_style)
worksheet2.write(1, 2, _('Валюта рахунку'), easyxf('font:height 250;font:bold True;align: horiz left;'))
worksheet2.write(1, 3, _('Сума у валюті компанії (' + str(self.env.user.company_id.currency_id.name) + ')'), easyxf('font
worksheet2.col(0).width = 5000
worksheet2.col(1).width = 5000
worksheet2.col(2).width = 4000
worksheet2.col(3).width = 8000
```

Рисунок 4.2.4 – формування листа 2.

Нижче описано, як виходячи з обраних даних формується звіт.

```python
        row = 7
        customer_row = 2
        for wizard in self:
            customer_payment_data = {}
            heading =  'Зведений звіт про рахунок'
            worksheet.write_merge(0, 0, 0, 5, heading, easyxf('font:height 210; align: horiz center;pattern: pattern solid, fore_
            heading =  'Підсумок рахунків для клієнтів'
            worksheet2.write_merge(0, 0, 0, 3, heading, easyxf('font:height 250; align: horiz center;pattern: pattern solid, fore
            if wizard.invoice_status == 'all':
                invoice_objs = self.env['account.invoice'].search([('date_invoice','>=',wizard.from_date),
                                                                   ('date_invoice','<=',wizard.to_date),
                                                                   ('type','=','out_invoice'),
                                                                   ('state','not in',['draft','cancel'])])

            elif wizard.invoice_status == 'paid':
                invoice_objs = self.env['account.invoice'].search([('date_invoice','>=',wizard.from_date),
                                                                   ('date_invoice','<=',wizard.to_date),
                                                                   ('state','=','paid'),('type','=','out_invoice')])
            else:
                invoice_objs = self.env['account.invoice'].search([('date_invoice','>=',wizard.from_date),
                                                                   ('date_invoice','<=',wizard.to_date),
                                                                   ('state','=','open'),('type','=','out_invoice')])
```

Рисунок 4.2.5 – формування звіту

У разі мультивалютності рахунків по одному замовнику, додається ще одна строка іншої валюти.

```
for customer in customer_payment_data:
    worksheet2.write(customer_row, 0, customer.split('_')[0])
    worksheet2.write(customer_row, 1, customer_payment_data[customer]['amount_total'])
    worksheet2.write(customer_row, 2, customer.split('_')[1])
    worksheet2.write(customer_row, 3, customer_payment_data[customer]['amount_company_currency'])
    customer_row += 1
```

Рисунок 4.2.6 – додається ще одна строка у випадку мультивалютності

На рисунку нижче описано як згренеровані дані з потоку іо зберігаються у файл формату xls.

```
fp = io.BytesIO()
workbook.save(fp)
excel_file = base64.encodestring(fp.getvalue())
wizard.invoice_summary_file = excel_file
wizard.file_name = 'Зведений звіт про рахунки.xls'
wizard.invoice_report_printed = True
fp.close()
return {
        'view_mode': 'form',
        'res_id': wizard.id,
        'res_model': 'print.invoice.summary',
        'view_type': 'form',
        'type': 'ir.actions.act_window',
        'context': self.env.context,
        'target': 'new',
          }
```

Рисунок 4.2.7 – формування звіту у форматі xls

4.3 Опис демо-версії програмного модуля

Після встановлення модуля «Invoice Summary Report», у меню з'являється додатковий пункт «Звітність».

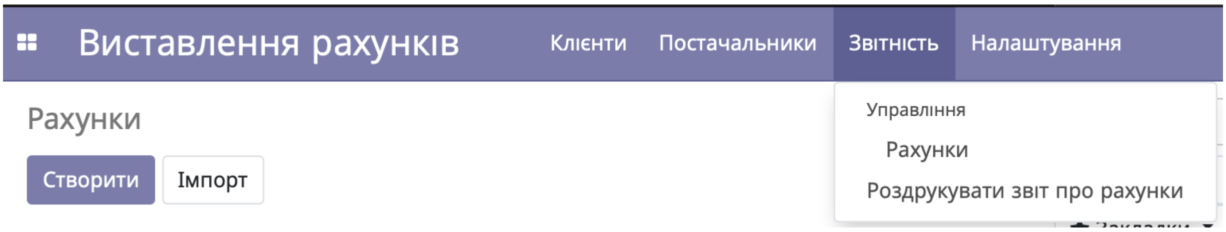

Рисунок 4.3.1 – Пункт «Звітність» у меню модуля «Виставлення рахунків»

Щоб отримати звіт потрібно натиснути на кнопку «Роздрукувати звіт про рахунки».

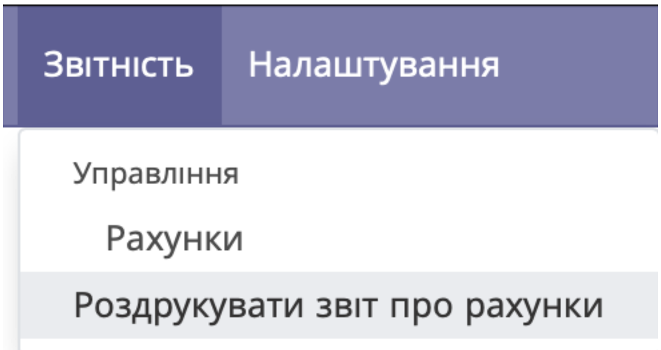

Рисунок 4.3.2 – зображення кнопки «Роздрукувати звіт про рахунки»

За замовчуванням період встановлений з початку фінансового року (або необхідну дату), дату «до» потрібно обрати необхідну дату.

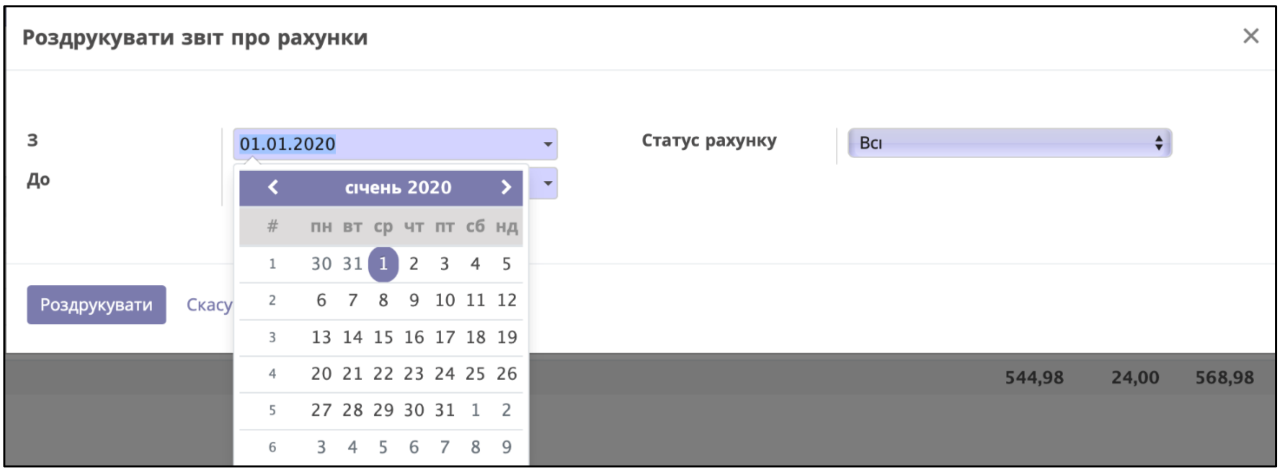

Рисунок 4.3.3 – Вибір періоду звітності

Також потрібно обрати статус рахунку (за замовчуванням установлено «Всі»).

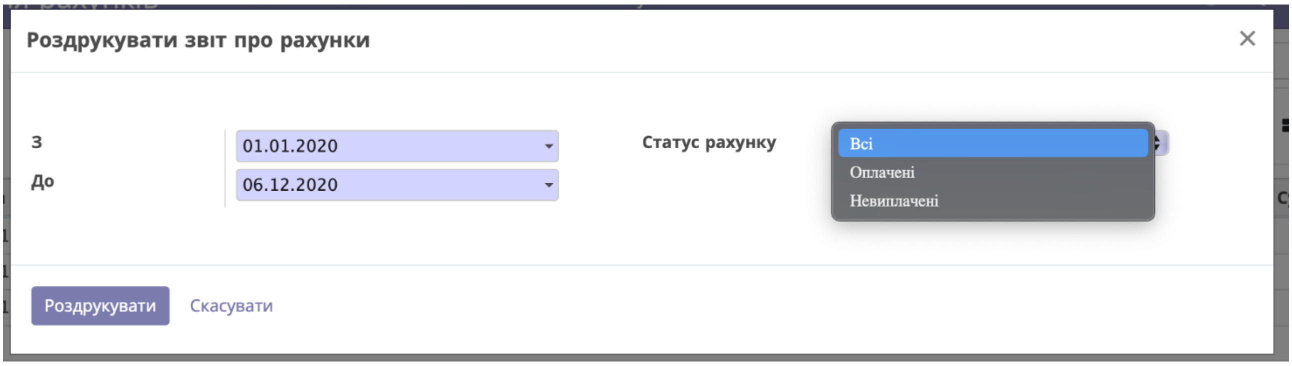

Рисунок 4.3.4 – Вибір статусу рахунків

Після того як виставлені усі поля, можна отримати звіт, якщо натиснути на кнопку «Роздрукувати».

Потім з'являється вікно, яке дозволяє скачати сформований файл звіту.

Рисунок 4.3.5 – Скачування файлу звіту

Як і описувалось, файл містить 2 вкладки:

1) Сумарна звітність усіх рахунків;
2) Підсумок рахунків-фактур для окремих клієнтів.

На рисунках нижче показано, як формується звіт з виставлених рахунків (див. Рисунок 4.3.6 та Рисунок 4.3.7).

Зведений звіт про рахунок-фактуру					
			Компанія Mega		
		2020-01-01	To	2020-12-06	
Номер рахунку	**Клієнт**	**Дата рахунку**	**Сума рахунку**	**Валюта рахунку**	**Сума у валюті компанії (UAH)**
INV/2020/0002	customer1	2020-11-30	120	₴	120
INV/2020/0003	Administrator	2020-11-22	425	₴	425
INV/2020/0001	customer1	2020-11-22	24	₴	24
					569

Рисунок 4.3.6 – Сумарна звітність усіх рахунків

Підсумок рахунків-фактур для клієнтів			
Клієнт	**Сплачена сума**	**Валюта рахунк**	**Сума у валюті компанії (UAH)**
customer1	144	UAH	144
Administrator	425	UAH	425

Рисунок 4.3.7 – Підсумок рахунків-фактур для окремих клієнтів

ВИСНОВКИ

Під час виконання дипломної роботи був розроблений програмний модуль автоматизованої звітності для виставлених рахунків «Invoice Summary Report» для полегшення отримання та обробки звітності фінансового відділу. Рішення було розроблено для застосування працівниками відділу фінансів програмного забезпечення компанії Netcracker та будь-якими користувачами систем ERP.

У процесі виконання проекту було обґрунтовано актуальність роботи, проаналізовано існуючі аналоги та їх переваги і недоліки, визначено функціональні вимоги до програмного модуля та визначено засоби реалізації.

Було виконано планування робіт, за результатами якого розроблено календарний план, а також визначено перелік ризиків під час виконання проекту.

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

- автоматичний збір даних, за певний період та стаус з можливістю їх подальшої модифікації користувачем;
- зберігання отриманого звіту за вказаним шляхом на локальному диску.

Програмний модуль було розроблено засобами об'єктно-орієнтованої мови Python у середовищі розробки Docker. Під час розробки було враховано виключні ситуації та відповідно їх оброблено. Для роботи програмного модуля було створено шаблони структури інструментарію розгортання пакетів програмного забезпечення та файли з конфігураціями проектів за стандартом XML.

Після завершення розробки було проведено тестування продукту.

Отже, виконавши всі етапи дипломного проекту, було розроблено програмний модуль звітності для підтримки діяльності фінансових спеціалістів, що дозволить покращити показники роботи та автоматизувати процес перевірки статусу сплати створених рахунків замовникам.

Використання розробленого програмного забезпечення дозволить зекономити час на виконання фінансових звітів.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Данжу Джульен. Путь Python. Черный пояс по разработке, масштабированию, тестированию и развертыванию СПб.: Питер, 2020. — 258 с.: ил. — (Серия «Библиотека программиста»).
2. Методология функционального моделирования IDEF0. Руководящий документ IDEF0 – 2000. – М.: Госстандарт России, 2000. – 115 с.
3. Одиночкина С.В. Основы технологий XML. Учебное пособие. — СПб: НИУ ИТМО, — 2013. — 56 с.
4. Плескач В. Л., Затонацька Т. Г. Інформаційні системи і технології на підприємствах: підручник. Київ: Знання, 2011. 718 с.
5. О'Лири Дэниел. ERP системы. Современное планирование и управление ресурсами предприятия. Выбор, внедрение, эксплуатация – М.: ООО «Вершина», 2004г. , 272 с.
6. Чёрненькая Л.В. Корпоративные информационные системы. Baan ERP: учеб. пособие / Л. В. Черненькая. – СПб.: Изд-во Политехн. ун-та, 2008 – 331с.
7. SAP ERP. Построение эффективной системы управления / Пер. с англ. — М.: Альпина Бизнес Букс, 2008. — 346 с.
8. David Sawyer McFarland. CSS3 The Missing Manual. O’Reilly Media, Inc., 2013. - 638 p. - ISBN-13 : 978-1449325947
9. Freeman Adam. Pro Entity Framework Core 2 for ASP.NET Core MVC / Apress, 2018. — 650 p. — ISBN-13: 978-1-4842-3434-1.
10. Gronwald Klaus-Dieter. Integrated Business Information Systems: A Holistic View of the Linked Business Process Chain ERP-SCM-CRM-BI-Big Data / Springer, 2017. — 206 p. — ISBN 978-3-662-53290-4.
11. Halpin T. Object-Role Modeling Fundamentals: A Practical Guide to Data Modeling with ORM / Basking Ridge: Technics Publications, 2015. - 151 p. – ISBN-13 : 978-1634620741

12. Head First Design Pattern. Eric Freeman, Elisabeth Robson, Bert Bates, Kathy Sierra – O'Reilly Media, 2004. – 694 p. – ISBN-13: 978-0596007126.
13. Mike McGrath. HTML5 in easy steps. In Easy Steps, 2011 – 240 p. – ISBN-13: 978-1840787542
14. Schenker Gabriel N. Learn Docker - Fundamentals of Docker 19.x: Build, test, ship, and run containers with Docker and Kubernetes / 2nd Edition. — Packt Publishing Ltd., 2020. — 574 p. — ISBN 978-1-83882-747-2.
15. Wallace T.F., Kremzar M.H. ERP: Making It Happen: The Implementers' Guide To Success With Enterprise Resource Planning / John Wiley & Sons, Inc., 2001. – 385 pages. – ISBN: 978-0-471-39201-9
16. Выравнивание текста в xlwt с помощью easyxf // Вопросы - CodeRoad: [Електронний ресурс]. – Режим доступу: https://coderoad.ru/19900972/%D0%B2%D1%8B%D1%80%D0%B0%D0%B2%D0%BD%D0%B8%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5-%D1%82%D0%B5%D0%BA%D1%81%D1%82%D0%B0-%D0%B2-xlwt-%D1%81-%D0%BF%D0%BE%D0%BC%D0%BE%D1%89%D1%8C%D1%8E-easyxf (дата звернення: 08.11.2020).
17. Знакомство с нотацией IDEF0 и пример использования // Habr: [Електронний ресурс]. – Режим доступу: https://habr.com/ru/company/trinion/blog/322832/ (дата звернення: 24.10.2020).
18. Програмний модуль // Вікіпедія: [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%B8%D0%B9_%D0%BC%D0%BE%D0%B4%D1%83%D0%BB%D1%8C#:~:text=%D0%9C%D0%BE%D0%B4%D1%83%D0%BB%D1%8C%20%E2%80%94%20%D1%84%D1%83%D0%BD%D0%BA%D1%86%D1%96%D0%BE%D0%BD%D0%B0%D0%BB%D1

%8C%D0%BD%D0%BE%20%D0%B7%D0%B0%D0%B2%D0%B5%D1%80%D1%88%D0%B5%D0%BD%D0%B8%D0%B9%20%D1%84%D1%80%D0%B0%D0%B3%D0%BC%D0%B5%D0%BD%D1%82%20% (дата звернення: 05.11.2020).

19. Module Structure // Open Source ERP and CRM | Odoo: [Електронний ресурс]. – Режим доступу: https://www.odoo.com/documentation/14.0/howtos/backend.html#module-structure (дата звернення: 05.10.2020).
20. What is TransientModel class // ODOO NINJA: [Електронний ресурс]. – Режим доступу: http://www.odooninja.com/what-is-transientmodel-classes/ (дата звернення: 12.11.2020).
21. 2020 Clash Of The Titans: SAP Vs. Oracle Vs. Microsoft Vs. Infor: [Електронний ресурс]. – Режим доступу: https://www.panorama-consulting.com/resource-center/clash-of-the-titans-2020-sap-vs-oracle-vs-microsoft-dynamics-vs-infor/ (дата звернення: 03.11.2020).

ДОДАТОК А. Лістинг програмного коду

__manifest.py__

```
{
    'name': 'Invoice Summary Report',
    'version': '0.1',
    'category': 'Accounting',
    'summary': 'Invoice Analysis Report',
    'license':'AGPL-3',
    'description': """
    Invoice Summary Report
""",
    'author' : 'Nataliia Chaika',
    'depends': ['account'],
    'images': ['static/description/banner.jpg'],
    'data': [
        'wizard/print_invoice_summary_view.xml'
    ],
    'installable': True,
    'auto_install': False,
    'application': True
}
```

print_invoice_summary_view.xml

```
<?xml version="1.0" encoding="utf-8"?>
<odoo>

        <record id="view_print_invoice_summary_form" model="ir.ui.view">
            <field name="name">print.invoice.summary.form</field>
            <field name="model">print.invoice.summary</field>
            <field name="type">form</field>
            <field name="arch" type="xml">
                    <form string="Роздрукувати звітний звіт про рахунок">
                            <field name="invoice_report_printed" invisible="1"/>
                                    <group  attrs="{'invisible':[('invoice_report_printed','=',True)]}">
                                            <group>
                                                    <field name="from_date" required="1"
style="width:250px;"/>
                                                    <field name="to_date" required="1" style="width:250px;"/>
                                            </group>
                                            <group>
                                                    <field name="invoice_status" required="1"
style="width:250px;"/>
                                    </group>
                            </group>
                            <group attrs="{'invisible':[('invoice_report_printed','=',False)]}">
                                <field name="file_name" invisible="1"/>
                                    <field name="invoice_summary_file" readonly="1"
filename="file_name"/>
                            </group>
                    <footer>
                    <button string='Роздрукувати'  name="action_print_invoice_summary"
```

```
                              type="object" class="btn-primary"
attrs="{'invisible':[('invoice_report_printed','=',True)]}"/>
                    <button string="Скасувати" class="oe_link" special="cancel"/>
                         </footer>
                      </form>
            </field>
        </record>

        <record id="action_print_invoice_summary" model="ir.actions.act_window">
            <field name="name">Роздрукувати звіт про рахунки</field>
            <field name="type">ir.actions.act_window</field>
            <field name="res_model">print.invoice.summary</field>
            <field name="view_type">form</field>
            <field name="view_mode">form</field>
            <field name="target">new</field>
        </record>

       <menuitem  id="menu_print_invoice_summary_report" action="action_print_invoice_summary"
            sequence="205" parent="account.menu_finance_reports" />

</odoo>
```

print_invoice_summary.py

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

class PrintInvoiceSummary(models.TransientModel):
    _name = "print.invoice.summary"

    @api.model
    def _get_from_date(self):
        company = self.env.user.company_id
        current_date = datetime.date.today()
        from_date = company.compute_fiscalyear_dates(current_date)['date_from']
        return from_date

    from_date = fields.Date(string='З',default=_get_from_date)
    to_date = fields.Date(string='До',default=datetime.date.today())
    invoice_summary_file = fields.Binary('Звітність з рахунків')
    file_name = fields.Char('File Name')
    invoice_report_printed = fields.Boolean('Звіт про рахунок-фактуру надруковано')
    invoice_status = fields.Selection([('all', 'Всі'), ('paid', 'Оплачені'),('un_paid', 'Невиплачені')],
string='Статус рахунку')


    @api.multi
    def action_print_invoice_summary(self):
        new_from_date = self.from_date.strftime('%Y-%m-%d')
        new_to_date = self.to_date.strftime('%Y-%m-%d')
```

```
        workbook = xlwt.Workbook()
        amount_tot = 0
        column_heading_style = easyxf('font:height 250;font:bold True;')
        worksheet = workbook.add_sheet('Сумарна звітність')
        worksheet.write(2, 3, self.env.user.company_id.name, easyxf('font:height 250;font:bold True;align:
horiz center;'))
        worksheet.write(4, 2, new_from_date, easyxf('font:height 250;font:bold True;align: horiz center;'))
        worksheet.write(4, 3, 'до',easyxf('font:height 250;font:bold True;align: horiz center;'))
        worksheet.write(4, 4, new_to_date,easyxf('font:height 250;font:bold True;align: horiz center;'))
        worksheet.write(6, 0, _('Номер рахунку'), column_heading_style)
        worksheet.write(6, 1, _('Клієнт'), column_heading_style)
        worksheet.write(6, 2, _('Дата рахунку'), column_heading_style)
        worksheet.write(6, 3, _('Сума рахунку'), column_heading_style)
        worksheet.write(6, 4, _('Валюта рахунку'), column_heading_style)
        worksheet.write(6, 5, _('Сума у валюті компанії (' + str(self.env.user.company_id.currency_id.name)
+ ')'), column_heading_style)

        worksheet.col(0).width = 5000
        worksheet.col(1).width = 5000
        worksheet.col(2).width = 5000
        worksheet.col(3).width = 5000
        worksheet.col(4).width = 5000
        worksheet.col(5).width = 8000



        worksheet2 = workbook.add_sheet('Підсумок рахунків-фактур для клієнтів')
        worksheet2.write(1, 0, _('Клієнт'), column_heading_style)
        worksheet2.write(1, 1, _('Сплачена сума'), column_heading_style)
        worksheet2.write(1, 2, _('Валюта рахунку'), easyxf('font:height 250;font:bold True;align: horiz left;'))
        worksheet2.write(1, 3, _('Сума у валюті компанії (' + str(self.env.user.company_id.currency_id.name)
+ ')'), easyxf('font:height 250;font:bold True;align: horiz left;'))
        worksheet2.col(0).width = 5000
        worksheet2.col(1).width = 5000
        worksheet2.col(2).width = 4000
        worksheet2.col(3).width = 8000

        row = 7
        customer_row = 2
        for wizard in self:
            customer_payment_data = {}
            heading =  'Зведений звіт про рахунок'
            worksheet.write_merge(0, 0, 0, 5, heading, easyxf('font:height 210; align: horiz center;pattern:
pattern solid, fore_color black; font: color white; font:bold True;' "borders: top thin,bottom thin"))
            heading =  'Підсумок рахунків для клієнтів'
            worksheet2.write_merge(0, 0, 0, 3, heading, easyxf('font:height 250; align: horiz center;pattern:
pattern solid, fore_color black; font: color white; font:bold True;' "borders: top thin,bottom thin"))
            if wizard.invoice_status == 'all':
                invoice_objs = self.env['account.invoice'].search([('date_invoice','>=',wizard.from_date),
                                                ('date_invoice','<=',wizard.to_date),
                                                ('type','=','out_invoice'),
                                                ('state','not in',['draft','cancel'])])
            elif wizard.invoice_status == 'paid':
                invoice_objs = self.env['account.invoice'].search([('date_invoice','>=',wizard.from_date),
                                                ('date_invoice','<=',wizard.to_date),
                                                ('state','=','paid'),('type','=','out_invoice')])
```

```
        else:
            invoice_objs = self.env['account.invoice'].search([('date_invoice','>=',wizard.from_date),
                                                ('date_invoice','<=',wizard.to_date),
                                                ('state','=','open'),('type','=','out_invoice')])
        for invoice in invoice_objs:
            invoice_date = invoice.date_invoice.strftime('%Y-%m-%d')
            amount = 0
            for journal_item in invoice.move_id.line_ids:
                amount += journal_item.debit
            worksheet.write(row, 0, invoice.number)
            worksheet.write(row, 1, invoice.partner_id.name)
            worksheet.write(row, 2, invoice_date)
            worksheet.write(row, 3, invoice.amount_total)
            worksheet.write(row, 4, invoice.currency_id.symbol)
            worksheet.write(row, 5, amount)
            amount_tot += amount
            row += 1
            key = u'_'.join((invoice.partner_id.name, invoice.currency_id.name)).encode('utf-8')
            key =  str(key,'utf-8')
            if key not in customer_payment_data:
                customer_payment_data.update({key: {'amount_total': invoice.amount_total,
'amount_company_currency': amount}})
            else:
                paid_amount_data = customer_payment_data[key]['amount_total'] + invoice.amount_total
                amount_currency = customer_payment_data[key]['amount_company_currency'] + amount
                customer_payment_data.update({key: {'amount_total': paid_amount_data,
'amount_company_currency': amount_currency}})
        worksheet.write(row+2, 5, amount_tot, column_heading_style)

        for customer in customer_payment_data:
            worksheet2.write(customer_row, 0, customer.split('_')[0])
            worksheet2.write(customer_row, 1, customer_payment_data[customer]['amount_total'])
            worksheet2.write(customer_row, 2, customer.split('_')[1])
            worksheet2.write(customer_row, 3,
customer_payment_data[customer]['amount_company_currency'])
            customer_row += 1


        fp = io.BytesIO()
        workbook.save(fp)
        excel_file = base64.encodestring(fp.getvalue())
        wizard.invoice_summary_file = excel_file
        wizard.file_name = 'Зведений звіт про рахунки.xls'
        wizard.invoice_report_printed = True
        fp.close()
        return {
                'view_mode': 'form',
                'res_id': wizard.id,
                'res_model': 'print.invoice.summary',
                'view_type': 'form',
                'type': 'ir.actions.act_window',
                'context': self.env.context,
                'target': 'new',
                  }
```