Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра мультимедійних систем факультету інформатики

ВПРОВАДЖЕННЯ СИСТЕМИ УПРАВЛІННЯ КОНТАКТАМИ ФАКУЛЬТЕТУ ІНФОРМАТИКИ

Текстова частина до курсової роботи за спеціальністю „Інженерія програмного забезпечення” 121

Керівник курсової роботи
ас. Корнійчук М. А.
_____
(підпис)
“____” _____ 2020 р.
Виконала студентка ФІ-4
Козопас В. О.
“____” _____ 2020 р.

Київ 2020

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра мультимедійних систем факультету інформатики

ЗАТВЕРДЖУЮ
Зав.кафедри мультимедійних систем,
проф., д. ф.-м.н.
_____В. В. Бублик
(підпис)
“____”_____ 2020 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
на курсову роботу

студентці 4 курсу факультету інформатики Козопас Вікторії Олександрівні

ТЕМА: Впровадження системи управління контактами факультету інформатики

Вихідні дані:

- Дослідження можливостей Odoo системи
- Реалізація системи управління контактами факультету інформатики на основі Odoo

Зміст ТЧ до курсової роботи:

Індивідуальне завдання
Вступ
1. Збір контактів за допомогою CRM системи
2. Аналіз Odoo системи
3. Створення системи управління контактами на основі Odoo
Висновки
Список використаних джерел

Дата видачі “___”_____ 2020 р.

Керівник _____
(підпис)

Завдання отримав _____
(підпис)

Тема: Впровадження системи управління контактами факультету інформатики

Календарний план виконання роботи

№ п/п	Назва етапу курсової роботи	Термін виконання етапу	Приміт ка
1.	Отримання завдання на курсову роботу.	08.10.2019	
2.	Огляд технічної літератури за темою роботи.	31.10.2019	
3.	Виконати аналіз існуючих досліджень за темою	11.11.2019	
3.	Дослідити можливі рішення для проблеми збору контактів	02.12.2019	
5.	Аналіз можливостей та архітектури Odoo системи	23.12.2019	
6.	Написання першого та другого розділів	13.01.2020	
7.	Розробка системи для управління контактами на основі Odoo	29.02.2020	
8.	Написання третього розділу	09.03.2020	
9.	Створення слайдів для доповіді та написання доповіді.	16.03.2020	
10.	Аналіз виконаної роботи з керівником.	20.03.2020	
11.	Коригування роботи.	25.03.2020	
12.	Остаточне оформлення пояснювальної роботи та слайдів.	30.03.2020	
13.	Захист курсової роботи	14.04.2020	

Студентка Козопас В. О.

Керівник Корнійчук М. А.

“_____”_____

ЗМІСТ

Анотація

В даній роботі розглянуто вирішення проблеми збору та управління контактами факультету інформатики НаУКМА. Крім того висвітлено аналіз можливих рішень та детальний огляд архітектури обраного застосунку. Врешті описано детальні кроки реалізації системи за допомогою Odoo сервера.

ВСТУП

Ні для кого не секрет, що правила ефективності співпраці полягають у даних, що ви збираєте.

Однак, на сьогоднішній день на факультеті інформатики НаУКМА виникла проблема відсутності сукупності стандартизованих даних контактних осіб, що зацікавлені у співпраці. Відтак втрачається зв'язок та можливість подальшої співпраці з гостьовими лекторами, спонсорами, представниками компаній тощо.

Виходячи з наведеної проблеми за мету даної роботи було поставлено створення системи управління контактами факультету інформатики. Зокрема було вирішено за основу скористатися готовим CRM рішенням з відкритим кодом. Використання запропонованого методу дозволяє отримати інструмент з втіленим основним функціоналом для подальшої розробки та внесення кастомних рішень.

Постановка задачі виглядає наступним чином:

1. Виконати аналіз доступних CRM систем з відкритим кодом.
2. Розглянути архітектуру обраної системи та особливості її реалізації.
3. Розгорнути систему та втілити додатковий функціонал відповідно до вимог.

Робота поділена на три розділи.

У першому розділі йдеться про проблему збору та управління контактами та CRM як інструмент для вирішення даної задачі. Крім того розглянуто безкоштовні й з відкритим кодом рішення та проведено їхню порівняльну характеристику відповідно до поставлених вимог до системи.

У другому розділі наведено аналіз Odoo системи, як фінального обраного рішення в результаті оцінок. Також описуються особливості архітектури Odoo сервера, суть модульності та можливість інтеграцій зі сторонніми сервісами.

Третій розділ присвячено власне реалізації системи управління контактами на основі Odoo. Тут йдеться про способи розгортання Odoo сервера, порядок створення власного модуля для кастомізацій та внесення змін, а також наведено приклад використання External API для зовнішній інтеграцій.

В результаті роботи створено програмний продукт, що втілює завдання збору та управління контактами факультету інформатики НаУКМА. Крім того, даний застосунок було інтегровано із сервісом порталу випускників.

РОЗДІЛ 1. ЗБІР КОНТАКТІВ ЗА ДОПОМОГОЮ CRM СИСТЕМИ

1.1 Постановка задачі та вимоги до системи управління контактами

Гостьові лектори, сертифіковані програми, інтернатури, курси, співпраця із закордонними ВНЗ — усе це ланки щоденної співпраці факультету. В такі моменти виникає потреба збереження усіх контактних даних для підтримки комунікації. Саме тут і виникло питання, що наразі таких зібраних та стандартизованих даних немає.

Тому на меті постало створення системи для збору й управління контактами в межах факультету інформатики НаУКМА.

Наступні пункти представляють початкові загальні вимоги до системи:

1. простота та швидкість у розгортанні;
2. безкоштовне рішення;
3. рішення з відкритим кодом для можливості кастомізацій;
4. можливість інтеграцій зі сторонніми сервісами.

Завдання полягає у вивченні питань, що нададуть можливість стандартизації контактної інформації про зацікавлених до співпраці осіб.

Джерелом контактів визначено соціальні мережі та контакти, надані різними відділами університету: деканат, відділ по роботі з випускниками тощо.

Серед основних функціональних вимог до роботи із контактами виділено: створення та видалення контакту, імпорт та експорт контактів, можливість архівації, можливість об'єднання у випадку дублікатів, редагування.

Перевагою даної системи стане надання адміністрації факультету структурованої системи для роботи із контактами.

1.2 CRM система як засіб для збору даних

1.2.1 Визначення поняття CRM

Customer relationship management або CRM — це технологія управління усіма взаємодіями та відносинами з клієнтами та потенційними клієнтами. CRM система допомагає компаніям залишатися на зв'язку з клієнтами, встановлювати довгострокові відносини та впорядковувати внутрішні процеси. Основна задача — зберегти контактну інформацію в один згуртований блок, що покращить видимість та спростить доступ до даних. [1]

1.2.2 Переваги CRM системи в рамках вирішення проблеми

В рамках вирішення даної задачі компанією для CRM слугуватиме факультет інформатики НаУКМА, а клієнтами, в свою чергу, випускники, викладачі, університети, експерти та управлінці IT компаній. Відповідно мета налагодження комунікацій між компанією та клієнтами переходить у площину факультет ↔ спільнота.

Ключові переваги використання CRM системи повністю покривають потреби зі створення системи для збору й управління контактами:

- ➔ управління контактами спільноти
- ➔ уніфікація збору даних
- ➔ категоризація та структуризація контактів
- ➔ відслідковування фінансових взаємодій
- ➔ інтеграція зі сторонніми сервісами

1.2.3 Аналіз існуючих рішень

Вибір та реалізація CRM системи це важливе рішення, що має ґрунтуватися на унікальних потребах залежно від задачі. Основною вимогою до CRM системи в рамках вирішення проблеми управління контактами на факультеті інформатики був вибір Open source рішення. Особливість у тому, що вихідний код застосунку з відкритим кодом (open source) може поширюватися та змінюватися користувачами відповідно до їхніх потреб. Ідея полягає у наданні можливості налаштування роботи додатку та кастомізації, щоб зробити додаток більш корисним.
Загалом було розглянуто більше 15 доступних рішень із відкритим кодом, проте зупинимось на фінальних трьох: VTiger, SuiteCRM та Odoo.

1.2.3.1 VTiger

VTiger — унікальне програмне забезпечення для управління відносинами з клієнтами, яке пропонує як хмарне рішення, так і застосунок з відкритим кодом. VTiger надає основні можливості та може бути ефективно використаний для швидкого розгортання системи.
Основні переваги: вбудована підтримка автоматизації робочих процесів, функції звітування та аналітики, розширення Gmail, 4.5 + млн завантажень на SourceForge, велика спільнота користувачів та розробників
Стек: розроблена за допомогою таких інструментів, як PHP, Apache, MySQL та SugarCRM.
Website: https://www.vtiger.com/open-source-crm/ .

1.2.3.2 SuiteCRM

SuiteCRM — потужна та доступна CRM система з відкритим кодом, модульність налаштувань якої надає гнучкість у кастомізації.
Основні переваги: легка в адаптації завдяки великій документації та спільноті, надає API для інтеграції зі сторонніми сервісами, працює на будь-якій операційній системі.
Стек: PHP.
Website: https://suitecrm.com/ .

1.2.3.3 Odoo

Odoo — надійне програмне забезпечення для управління відносинами з клієнтами з переконливо багатим переліком функцій. Основний функціонал представляється у відкритому доступі, що надає можливість змінити програмне забезпечення відповідно до особливих вимог.
Основні переваги: сучасний та простий у навігації користувацький інтерфейс, широка підтримка інтеграцій зі сторонніми сервісами.
Стек: Python, PostgreSQL, JavaScript, XML.
Website: https://www.odoo.com/page/crm .

Ретельно зваживши всі особливості та переваги кожного з рішень та їхню відповідність до щоденних потреб користувачів системи для розробки було обрано Odoo CRM. [2, 3, 4]

РОЗДІЛ 2. АНАЛІЗ ODOO СИСТЕМИ

2.1 Опис Odoo системи

Odoo — це потужний набір бізнес-додатків з відкритим кодом побудованих на основі OpenObject фреймворку. При першому встановленні Odoo єдиний доступний функціонал — обмежений обмін повідомленнями між користувачами. Уся додаткова функціональність з'являється за допомогою встановлення відповідних до потреб модулів. Така гнучкість робить Odoo набагато доступнішим за інші рішення на ринку.

Налаштування та керування системою Odoo потребує базового розуміння компонентів, що входять до її складу. Кожен модуль як бізнес-система має набір технологій та програмних платформ для забезпечення його коректного функціонування. [6]

2.2 Архітектура Odoo системи

В основі Odoo лежить багатостороння трирівнева архітектура: рівень бази даних для зберігання даних, рівень застосунків та інтерфейсний рівень, що забезпечує користувальницький інтерфейс. Усі вони є окремими шарами в архітектурі.

Рівень застосунків є власне основою, додаткові ж модулі можуть бути встановленні поверх нього. Odoo також наслідує архітектурний паттерн MVC — Model View Controller.

На рис. 1.1 зображено трирівневу архітектуру Odoo.

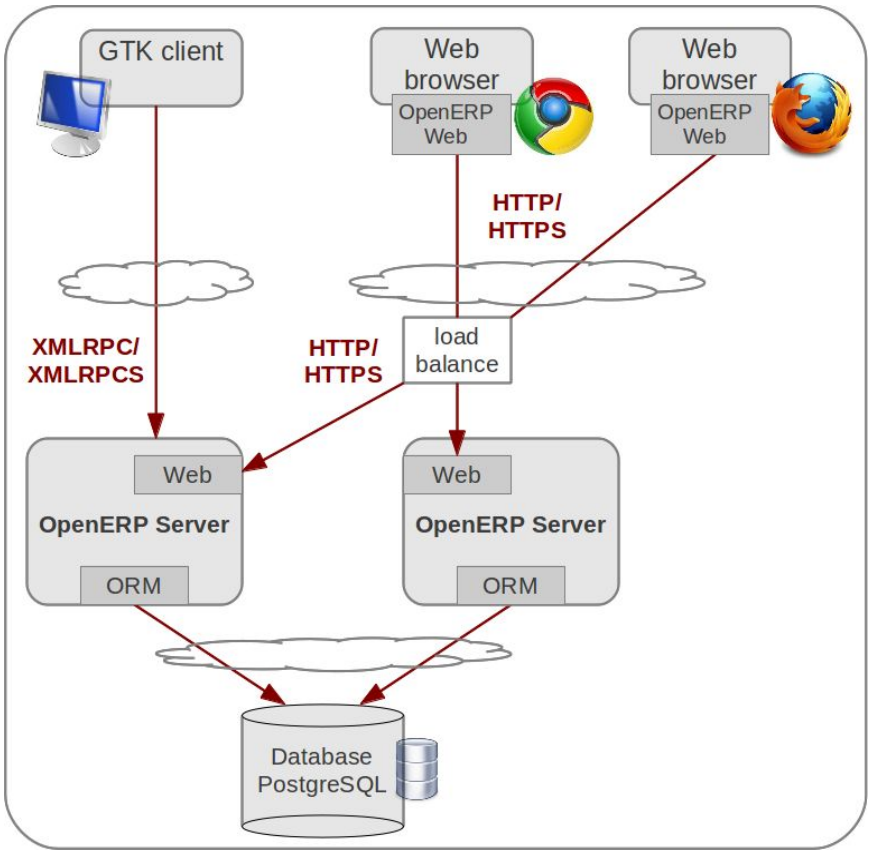

Рис. 1.1 Архітектура системи Odoo. [12]

Як бачимо, система складається із трьох компонентів:

- клієнтська частина як javascript застосунок у браузері;
- Odoo сервер, що містить усю логіку та відповідає за оптимальну роботу застосунку. Одна з частин в Odoo сервері — ORM рушій — призначена для комунікації між сервером та PostgreSQL базою даних, інша — Web — для зв'язку з інтерфейсом;
- PostgreSQL база даних, що містить дані усіх модулів та конфігураційні елементи.

2.2.1 PostgreSQL база даних

Рівень баз даних є найнижчим рівнем та відповідає за збереження даних. Odoo для цього спирається на PostgreSQL сервер баз даних. Більше того, PostgreSQL єдина підтримувана RDBMS — це вибір дизайну, інші бази даних, як MySQL, системою не підтримуються. Бінарні файли, такі як зображення, зазвичай зберігаються напряму в базі даних.

2.2.2 Odoo сервер

Логічний рівень або рівень застосунків відповідає за всі взаємодії із рівнем даних та керується Odoo сервером. Як правило, доступ низького рівня до бази даних може бути здійснений лише через даний рівень, оскільки це єдиний спосіб забезпечити контроль доступу до даних.

2.2.2.1 ORM

В основі сервера Odoo лежить ORM — об'єктно-реляційне відображення, який слугує інтерфейсом поверх сервера PostgreSQL. ORM забезпечує так зване API, який використовують додаткові модулі для зв'язку із даними.
Моделі даних описуються за допомогою Python і Odoo сервер створює відповідні таблиці в базі даних за допомогою ORM. Усі переваги RDBMS такі як обмеження унікальності, цілісність посилань або оптимізація запитів втілюються за допомогою гнучкості Python.
Наприклад, сутність Partner, такі як Customer або Supplier, представлені в ORM як модель. Ця модель є Python класом, що підтримує декілька

методів взаємодій, такі як create() — для створення нового Partner запису, або read() — для запиту усіх існуючих записів в базі. Ці методи імплементують загальну бізнес логіку в конкретній моделі.

2.2.2.2 Web

Web рівень в Odoo сервері відповідає за комунікацію з клієнтом веб-браузера. Цей веб-рівень — це WSGI-сумісна програма базована на werkzeug, що є бібліотекою веб-додатків WSGI. Він обробляє http-запити до сервера та JSON-RPC запити для RPC — remote procedure call — з веб-браузера.

2.2.3 Клієнтська частина

Рівень клієнтської частини відповідає за представлення даних та взаємодію з користувачем. Клієнт взаємодіє із ORM API для читання, запису, підтвердження або виконання будь-яких інших, викликаючи методи ORM API через віддалені виклики процедур — remote procedure call (RPC). Вони надсилаються на Odoo сервер для опрацювання, результати надсилаються ж назад клієнту для подальшої обробки.
Для клієнтського рівня Odoo надає повнофункціональний веб-клієнт “з-під коробки”. Веб-клієнт підтримує всі необхідні для бізнес-програми функції: сеанси логіну, меню навігації, списки даних, форми тощо.
[5, 7]

2.3 Мережеві комунікації та WSGI

Odoo — це HTTP веб-сервер, який також може бути розгорнутий як WSGI-сумісна програма. Клієнти можуть комунікувати з Odoo за допомогою XML-RPC, що є рекомендованим шляхом взаємодії. Веб-клієнти комунікують за допомогою JSON-RPC.

WSGI — це стандартне рішення взаємодії Python-програми та веб-серверу. За допомогою WSGI можна запустити Odoo на будь-якому сервері сумісному з WSGI.

Простори імен HTTP як /openerp/ /object/ /common/ зарезервовані для рівня XML-RPC, кожен модуль також обмежує простір імен на /<name_of_the_module>/. [7]

2.4 MVC архітектура в Odoo сервері

Odoo побудований на основі архітектури Model-View-Controller (MVC). Однією з основних цілей даної архітектури є відокремлення візуального відображення інформації від бізнес-логіки та управління даними. Це означає, що зміни в інтерфейсі користувача не впливають на управління даними, а дані можна реорганізувати без зміни інтерфейсу.

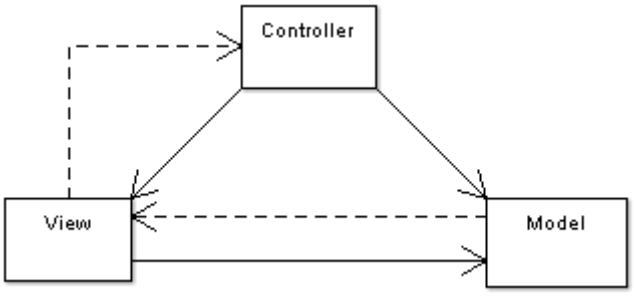

Рис. 1.2. Діаграма архітектури MVC [13]

На діаграмі вище суцільні лінії означають повний доступ, пунктирні — обмежений. Причинами такого дизайну є:

- від View до Model: модель надсилає повідомлення до відображення про зміну, щоб представлення перемалювало свій вміст. Моделі не потрібно здати внутрішню сторону відображення, натомість відображення повинно мати повний доступ до моделі.
- від View до Controller: залежності від представлення до контролера повинні бути мінімальними, щоб контролер можна було змінити у біль-який момент.

Odoo наслідує семантику MVC з:

model: PostgreSQL таблиця в базі даних

view: представлення в XML файлах

controller: Python об'єкти в Odoo. [7]

2.5 Програми та модулі в Odoo

В більшості випадків розробка в Odoo означає створення нових модулів. Модулі або додатки (addons) є будівельними блоками для Odoo застосунку. Модуль може додавати нові функції та можливості, або ж модифікувати існуючі. Власне модуль — це директорія з файлом маніфесту, __manifest__.py, та рештою файлів, що реалізують його особливості.

Програми (applications) — це метод додавання до Odoo основних функцій. Програми забезпечують суцільні функціональні властивості, такі як Бухгалтерський облік або HR. Програми базуються на додаткових модулях та розширюють функціональність. Через це вони виділяють в Odoo Apps меню.

Якщо модуль складний та додає багато нового функціоналу, то є сенс винести його як програму. Якщо модуль лише вносить зміни до існуючого функціоналу, то це радше модуль, а не програма.

Чи є модуль програмою визначається у файлі маніфесту, проте технічно це не впливає на поведінку модуля. [5]

2.6 Кастомізації — внесення змін до модулю

Як правило, вважається поганою практикою модифікувати існуючі модулі шляхом зміни вихідного коду напряму. Особливо це стосується офіційних модулів, наданих Odoo. Така практика не дозволяє чітко розділити різницю між початковим кодом та внесеними змінами, що додає складнощі в оновлення модулів.

Натомість потрібно створити новий модуль — розширення, встановити його поруч із модулем, який хочемо змінити, прописавши в ньому всі необхідні зміни. Кастомізація модуля здійснюється завдяки механізму успадкування, який дозволяє розширювати існуючі модулі.

Для розширення існуючої моделі використовується Python клас з атрибутом _inherit. Це визначає модель, яку потрібно розширити. Новий клас успадковує всі риси батьківського, потрібно лиш дописати бажані модифікації.

Успадкування можливе на всіх рівнях: моделі даних, бізнес-логіка, користувацький інтерфейс. [5]

2.7 CRM модуль

Odoo CRM — це потужний та легкий у використанні модуль для ефективного менеджменту. Odoo CRM — це комбінація багатьох модулів, серед яких CRM, модуль контактів, обговорень, продажів тощо.

Деякі з корисних функцій Odoo CRM включають:

- ➢ пріоритезація завдань — дозволяє виділяти пріоритетні завдання для подальшої діяльності;
- ➢ розклад зустрічей;
- ➢ інформаційні панелі (dashboards);
- ➢ зв'язок із клієнтами — підтримка комунікації електронною поштою, телефоном, чатом;
- ➢ індивідуальні сповіщення тощо. [8]

2.8 External API для інтеграції зі сторонніми сервісами

Odoo сервер надає зовнішній API, який використовується власним веб-клієнтам та також доступний для інших клієнтських додатків. Доступ до Odoo API можна отримати ззовні за допомогою двох протоколів: XML-RPC та JSON-RPC. Будь-яка зовнішня програма, здатна реалізувати клієнта для одного з протоколів може взаємодіяти з Odoo сервером.

Відповідно можна використовувати будь-яку мову програмування, що підтримує XML-RPC та JSON-RPC протоколи. Офіційна документація надає як приклад зразки коду на чотирьох мовах програмування: Python, PHP, Ruby та Java.

Найпростіший методи доступу до Odoo сервера це XML-RPC. У Python можна скористатися стандартною бібліотекою `xmlrpclib`.

Odoo вимагає, щоб користувачі API були автентифіковані перед тим, як зможуть робити запити за даними. `xmlrpc/2/common` ендпоінт забезпечує мета-виклики, які не потребують автентифікації або отримання інформації.

Приклад коду на Python:

```
import xmlrpc.client
common = xmlrpc.client.ServerProxy('{}/xmlrpc/2/common'.format(url))
```

Автентифікація ж проводиться за допомогою функції `authenticate()` та повертає ідентифікатор користувача, що використовується в автентифікованих викликах замість входу.

Приклад коду на Python:

```
import xmlrpc.client
common =
xmlrpc.client.ServerProxy('{}/xmlrpc/2/common'.format(url))
uid = common.authenticate(db, username, password, {})
```

Другий ендпоінт `xmlrpc/2/object` використовується для виклику Odoo моделей через RPC функцію `execute_kw`. Кожен виклик функції має приймає такі параметри:

- база даних для використання
- ідентифікатор користувача (отриманий при автентифікації)
- пароль користувача
- назва моделі
- назва методу

- масив/список параметрів запиту
- словник параметрів за ключем.

Наприклад, щоб побачити чи ми зможемо прочитати модель `res.partner`, ми можемо викликати `check_access_rights` та параметр за ключем `raise_exception`.

Приклад коду на Python:

```
url = <insert server URL>
db = <insert database name>
username = 'admin'
password = <insert password for your admin user (default: admin)>

models = xmlrpc.client.ServerProxy('{}/xmlrpc/2/object'.format(url))
models.execute_kw(db, uid, password,
    'res.partner', 'check_access_rights',
    ['read'], {'raise_exception': False})
```

[5, 9]

РОЗДІЛ 3. СТВОРЕННЯ СИСТЕМИ УПРАВЛІННЯ КОНТАКТАМИ НА ОСНОВІ ODOO

3.1 Розгортання Odoo сервера для розробки

Перш ніж зануритися в розробку з Odoo потрібно відповідно налаштувати середовище.

Odoo побудований з використанням мови програмування Python та використовує PostgreSQL базу даних для сховища даних — це дві основні вимоги для розгортання Odoo.

Загалом є декілька варіантів встановлення Odoo:

1. Online — онлайн версія з 15 днями пробного періоду;
2. Packaged installers — інсталятори для Windows та debian-based дистрибутивів, пакети автоматично встановлюють усі залежності;
3. Source install — забезпечує більшу гнучкість, підходить для розробки модулів;
4. Docker — надає базовий docker image.

Щоб встановити Odoo з вихідного коду (source install), потрібно встановити залежні Python бібліотеки. Сам ж вихідний код можна завантажити з GitHub.

```
$ git clone https://github.com/odoo/odoo.git
```

Odoo використовує PostgreSQL як систему управління базами даних. За замовчуванням єдиним користувачем є postgres, проте Odoo забороняє підєднуватися до даного користувача, тому потрібно створити нового.

Для запуску Odoo потрібен Python версії 3.6 або вище. Усі залежності перераховані у файлі `requirements.txt` , що знаходиться у корені

каталогу. Для встановлення залежностей потрібно ввести: `$ pip install -r requirements.txt` .

Після встановлення всіх залежностей запустити Odoo можна за допомогою інтерфейсу командного рядка odoo-bin, що розташований в корені каталогу. Типовою командою запуску сервера буде:

`python odoo-bin --addons-path=addons` .

За замовчуванням сутності Odoo працюють на порту 8069, тож далі потрібно перейти на `http://<server address>:8069`, щоб отримати доступ до Odoo. При першому запуску ми побачимо помічника для створення нової бази даних для Odoo (див. Рис. 3.1). [5, 10]

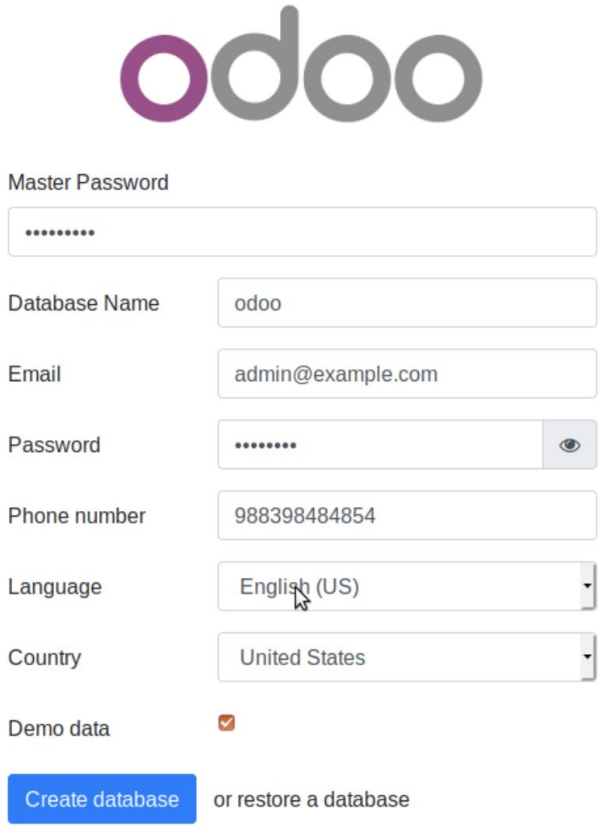

Рис. 3.1 Сторінка створення нової бази даних

Також Odoo пропонує заповнити базу даних тестовими даними.

3.2 Додаткові модулі та програми

Після логіну відкривається сторінка Apps з усіма доступними для встановлення програмами та модулями.

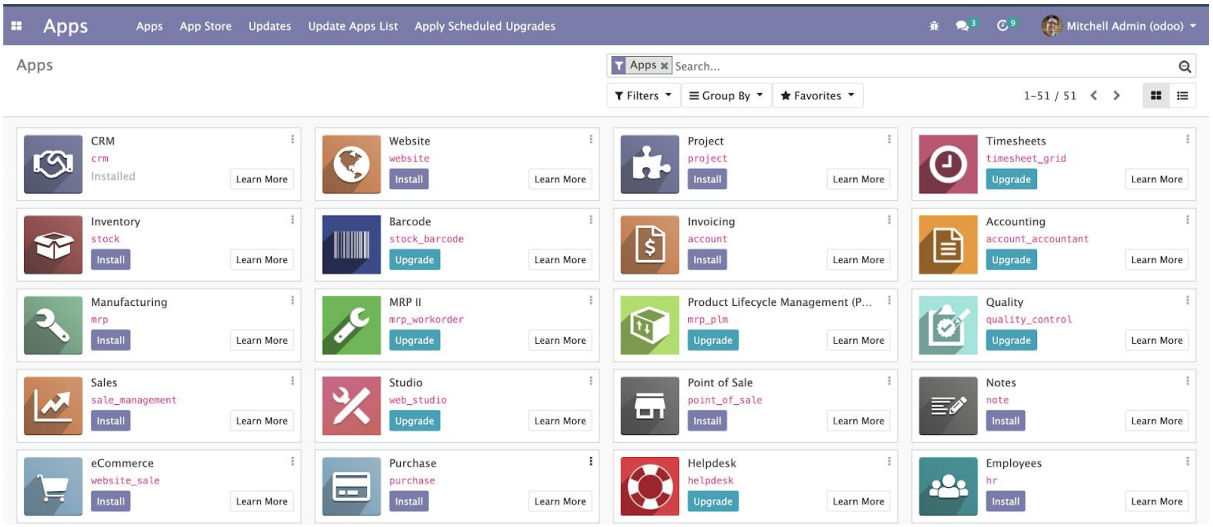

Рис. 3.2 Модулі Odoo

Початково встановлена версія Odoo забезпечує лише обмежену систему обміну повідомленнями. Для поповнення функціоналу потрібно встановити додаткові модулі та програми.

Для нашої задачі — системи управління контактами — найкраще підходить модуль CRM, що має також обов'язковими залежностями модулі Contacts та Calendar.

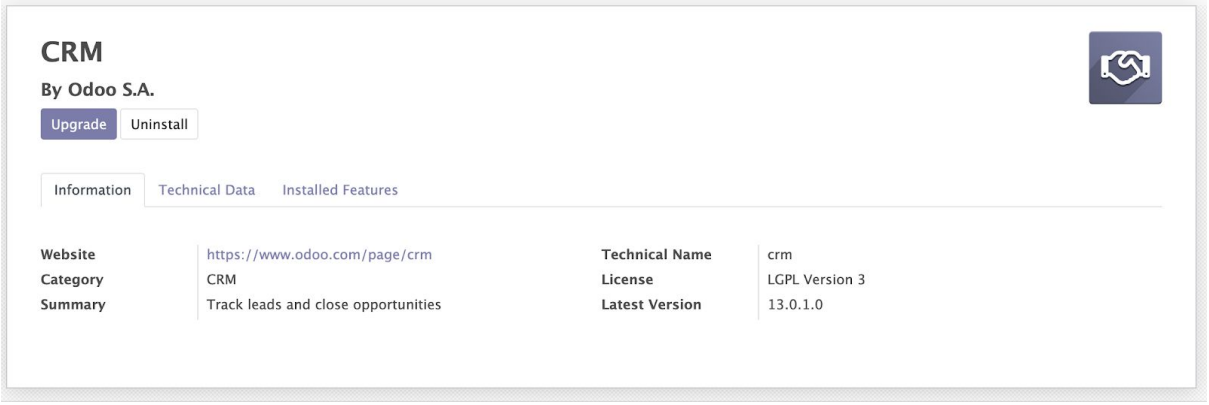

Рис. 3.3 Модуль CRM

За допомогою модуля Contacts можна легко відслідковувати взаємодії з контактами та зручно впорядковувати інформацію.

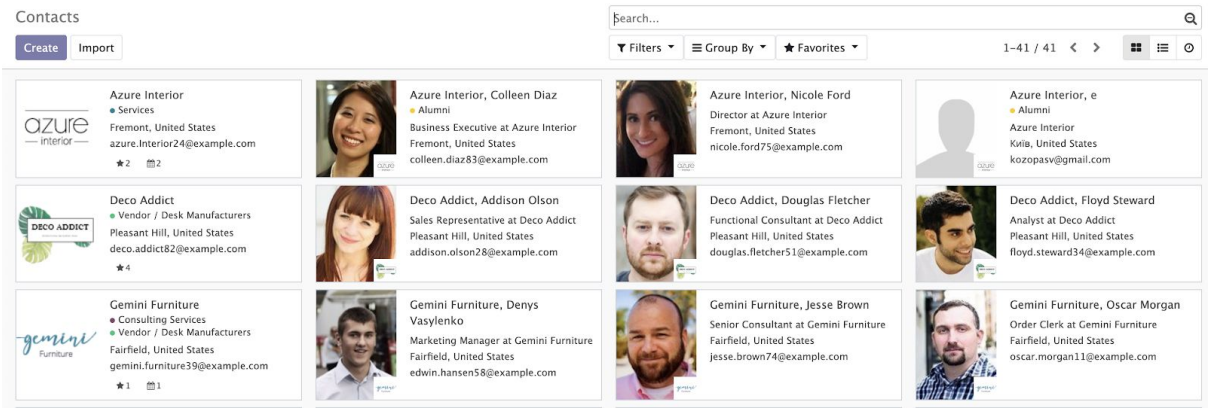

Рис. 3.4 Модуль Contacts

Сутності контактів поділяються на два типи: компанія та особа. В рамках вирішення задачі управління контактами для факультету компаніями є власне IT-компанії, університети, а особами — випускники, викладачі, гостьові лектори, працівники та управлінці компаній.

На кожен із контактів можна навішувати теги, наприклад “Бізнес” чи “Випускник” для зручного групування та фільтрації. До речі, у верхній правій панелі знаходяться опції для фільтрів та групування контактів. Надається можливість створювати також кастомізовані фільтри зі складними умовами.

Крім того, є можливість імпортування контактів за шаблоном (див. Рис. 3.5).

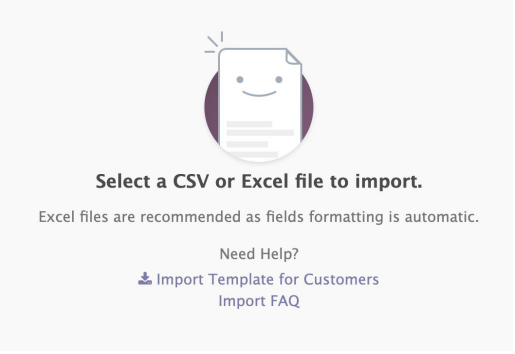

Рис. 3.5 Вікно імпортування контактів

3.3 Створення власного модулю для кастомізацій

Не зважаючи на великий вбудований функціонал, що надає Odoo, все одно виникає потреба додати власні поля відповідно до потреб. На рис. 3.6 виділено нові поля, додані у власному модулі.

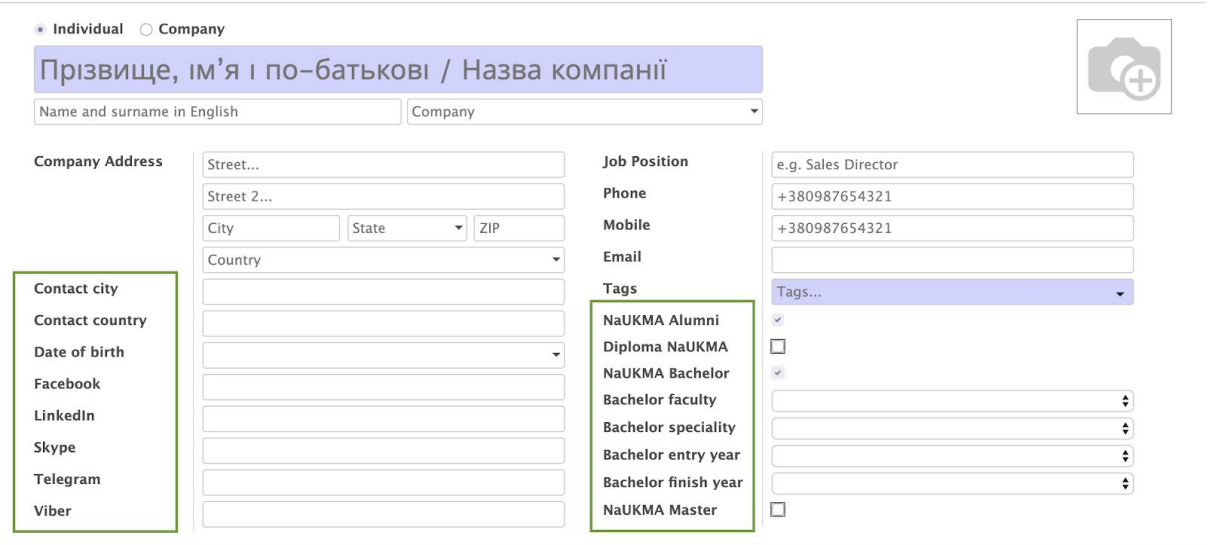

Рис. 3.6 Кастомізовані поля

Для того, щоб розширити модуль, потрібно створити власний модуль, який буде успадковувати базовий, та встановити його поруч.

Щоб автоматично створити структуру нового модулю потрібно ввести команду:

`./odoo-bin scaffold module_name folder_name` ,

де `module_name` — назва модулю, `folder_name` — назва директорії, в якій потрібно створити модуль.

Загальна структура модуля повинна містити основні директорії з моделями, контролерами та відображеннями, а також файли __init__.py та __manifest__.py. У файлі __manifest__.py описуються усі необхідні атрибути модуля для Odoo (назва, опис, версія, залежності тощо). Структуру створеного модуля зображено на рис. 3.7.

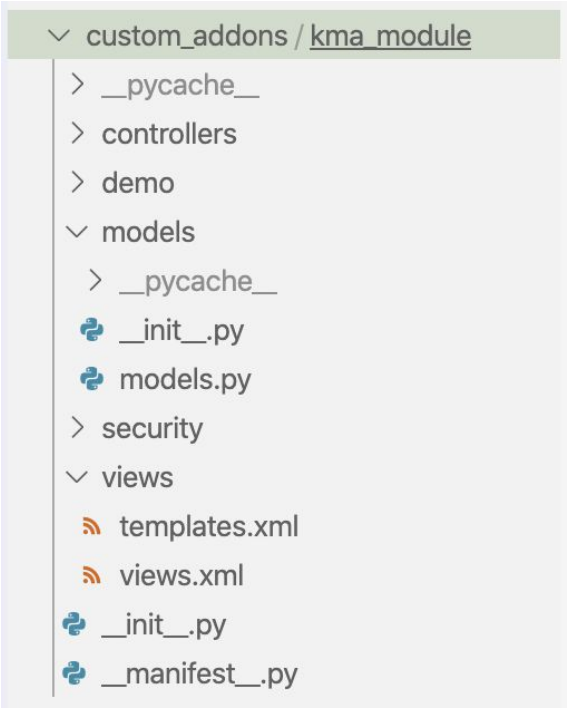

Рис. 3.7 Структура модулю

Для редагування картки контакту потрібно унаслідувати базову модель — `res.partner`, тому у файлі model.py створюємо новий клас, що унаслідуватиме дану модель та додаємо усі потрібні нам поля.

```
class AlumniContact(models.Model):
    _inherit = 'res.partner'

    full_name = fields.Char(string='Full name')
    birth_date = fields.Date(string='Date of birth')

    contact_city = fields.Char(string='Contact city')
    contact_country = fields.Char(string='Contact country')

    facebook_link = fields.Char(string='Facebook')
    linkedin_link = fields.Char(string='LinkedIn')
    skype = fields.Char(string='Skype')
```

Рис. 3.8 Модель

Для відображення нових полів їх потрібно прописати у views.xml файлі, у якому теж потрібно унаслідувати модель `res.partner` (див. рис. 3.9).

```
<record id="view_order_form_inherit" model="ir.ui.view">
    <field name="name">Alumni contact custom</field>
    <field name="model">res.partner</field>
    <field name="inherit_id" ref="base.view_partner_form"/>
    <field name="arch" type="xml">

    <field name="vat" position="after">
        <field name="contact_city" attrs="{'invisible': [('is_company','=', True)]}"/>
        <field name="contact_country" attrs="{'invisible': [('is_company','=', True)]}"/>
        <field name="birth_date" widget="date" attrs="{'invisible': [('is_company','=', True)]}"/>
        <field name="facebook_link" widget="url" attrs="{'invisible': [('is_company','=', True)]}"/>
        <field name="linkedin_link" widget="url" attrs="{'invisible': [('is_company','=', True)]}"/>
        <field name="skype" widget="url" attrs="{'invisible': [('is_company','=', True)]}"/>
```

Рис. 3.9 Відображення

Після внесення усіх змін потрібно підключити новий модуль. Щоб кастомний модуль відображався в Apps потрібно додати шлях до нього при запуску Odoo сервера. Наш модуль знаходиться в директорії custom_addons, тому команда запуску виглядатиме наступним чином: `./odoo-bin --addons-path=addons,custom_addons` .

Далі потрібно оновити список модулів, натиснувши Update Apps, забрати фільтр Apps з пошукової стрічки, щоб модулі також відображались у пошуку. Після цього модуль з'явиться доступним для встановлення:

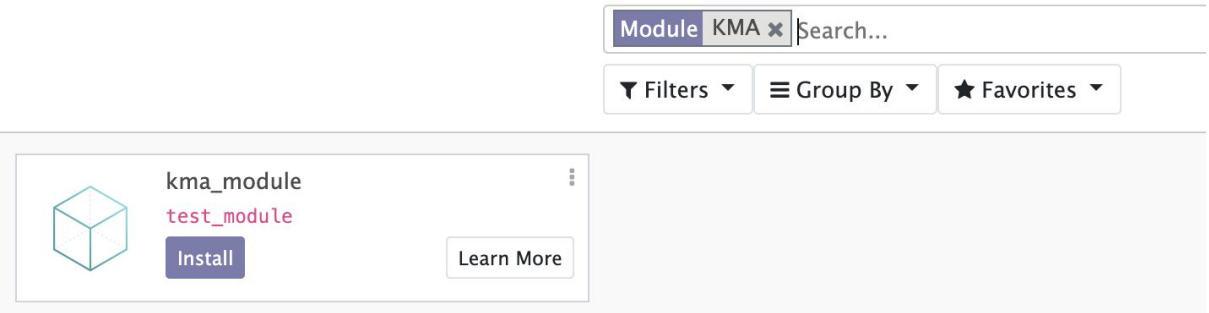

Рис. 3.10 Кастомний модуль

[5, 11]

3.4 Використання External API для інтеграцій

Для інтеграцій зі сторонніми сервісами Odoo надає External API. Наприклад, щоб отримати усі імена та фото осіб, що є випускниками НаУКМА потрібно викликати наступний метод:

```
contacts = models.execute_kw(db, uid, password,
           'res.partner', 'search_read',
           [[['is_company', '=', False], ['is_alumni', '=', True]]],
           {'fields': ['name', 'image_1920'],
            'limit': 0,
            'offset': 0}).
```

У даному випадку ми використовуємо функцію `search_read`, що повертає усі дані з таблиці `res.partner`, що підлягають фільтру `[[['is_company', '=', False], ['is_alumni', '=', True]]]`. Аргументом також можна задати конкретні поля, які потрібно повернути `'fields': ['name', 'image_1920']`. За замовчуванням функція поверне усі елементи, які підлягають умові. Проте можна задати підмножину — пагінацію — за допомогою параметрів `'limit'` та `'offset'`.

Завдяки External API систему управління контактами на базі Odoo вдалося інтегрувати з порталом випускників.

ВИСНОВКИ

В ході роботи було досліджено проблему збору та управління контактами в межах факультету інформатики НаУКМА.

Насамперед, в роботі було розглянуто можливість використання CRM системи для розв'язку поставленої задачі та було проведено порівняльну характеристику наявних рішень відповідно до поставлених вимог.

Як результат дослідження для подальшої роботи було обрано систему Odoo. Також в роботі розглянуто особливості архітектури Odoo, модульність системи, можливості кастомізацій та інтеграцій.

Врешті було реалізовано систему для збору та управління контактами на основі Odoo сервера. В роботі висвітлено детальні кроки розгортання системи, розробка власного модуля для внесення змін відповідно до власних потреб та можливий варіант зв'язку з сервером для інтеграції.

Наразі система в роботі, запущена на сервері факультету та також інтегрована з порталом випускників.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. What is CRM? [Електронний ресурс] – Режим доступу до ресурсу: https://www.salesforce.com/crm/what-is-crm/.
2. 8 Best Open Source CRM Tools [Електронний ресурс] – Режим доступу до ресурсу: https://crm.org/crmland/open-source-crm.
3. The 20 Free and Open Source CRM Solutions for Small Enterprises [Електронний ресурс] – Режим доступу до ресурсу: https://www.ubuntupit.com/free-and-open-source-crm-solutions/.
4. CRM Comparison White Paper: Salesforce, Zoho CRM, SugarCRM, Pipedrive & Odoo [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: https://odoocdn.com/openerp_website/static/src/pdf/crm_comparison.pdf.
5. Reis D. Odoo 10 development essentials / Daniel Reis., 2016. – 273 с.
6. Moss G. Working with Odoo / Greg Moss., 2015. – 399 с.
7. Odoo Architecture [Електронний ресурс] – Режим доступу до ресурсу: https://doc.odoo.com/trunk/server/02_architecture.
8. Odoo CRM Features [Електронний ресурс] – Режим доступу до ресурсу: https://www.cybrosys.com/blog/odoo-crm-features.
9. External API [Електронний ресурс] – Режим доступу до ресурсу: https://www.odoo.com/documentation/13.0/webservices/odoo.html.
10. Installing Odoo [Електронний ресурс] – Режим доступу до ресурсу: https://www.odoo.com/documentation/13.0/setup/install.html#setup-install-source.
11. How to Create a Module in Odoo 12 [Електронний ресурс] – Режим доступу до ресурсу: https://www.cybrosys.com/blog/how-to-create-module-in-odoo12.

12. Architecture [Електронний ресурс] – Режим доступу до ресурсу: https://doc.odoo.com/doc_static/trunk/server/_images/02_openerp_architecture.png.
13. MVC [Електронний ресурс] – Режим доступу до ресурсу: https://doc.odoo.com/doc_static/trunk/server/_images/02_mvc_diagram.png.