**Міністерство освіти і науки України**
**Національний технічний університет**
**«Дніпровська політехніка»**

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

**Кафедра** Програмного забезпечення комп'ютерних систем
(повна назва)

# ПОЯСНЮВАЛЬНА ЗАПИСКА
## кваліфікаційної роботи ступеня

*магістра*
(назва освітньо-кваліфікаційного рівня)

**студента** *Смигунова Іллі Володимировича*
(ПІБ)

**академічної групи** *121м-21-1*
(шифр)

**спеціальності** *121 Інженерія програмного забезпечення*
(код і назва спеціальності)

**освітньої програми** *Інженерія програмного забезпечення*
(назва освітньої програми)

**на тему:** *Розробка та дослідження ефективності впровадження програмного забезпечення для управління персоналом з використанням HRM системи на базі автоматизованих ключових HR-процесів*

| **Керівники** | **Прізвище, ініціали** | **Оцінка за шкалою** | | **Підпис** |
|---|---|---|---|---|
| | | **рейтинговою** | **інституційною** | |
| кваліфікаційної роботи | *проф. Мороз Б.І.* | | | |
| **розділів:** | | | | |
| спеціальний | *проф. Мороз Б.І.* | | | |
| економічний | | | | |
| | | | | |
| | | | | |
| **Рецензент** | *проф. Корнієнко В.І.* | | | |
| **Нормоконтролер** | *проф. Лактіонов І.С.* | | | |

**Дніпро**
**2022**

**Міністерство освіти і науки України**
**Національний технічний університет**
**«Дніпровська політехніка»**

**ЗАТВЕРДЖЕНО:**
Завідувач кафедри
Програмного забезпечення комп'ютерних систем
(повна назва)

_____ І.М. Удовик
(підпис) (прізвище, ініціали)

« » _____ 2022 Року

# ЗАВДАННЯ
## на виконання кваліфікаційної роботи магістра

**спеціальності** *121 Інженерія програмного забезпечення*
*(код і назва спеціальності)*

**студенту** *121м-21-1* *Смигунову Іллі Володимировичу*
(група) (прізвище та ініціали)

**Тема дипломного проекту** *Розробка та дослідження ефективності впровадження програмного забезпечення для управління персоналом з використанням HRM системи на базі автоматизованих ключових HR-процесів*

## 1 ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Наказ ректора НТУ «Дніпровська політехніка» від 31.10.2022 р. № 1200-с

## 2 МЕТА ТА ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

**Об'єкт досліджень** – процес керуванням існуючим людським ресурсом, а також відбору кандидатів на посади.

**Предмет досліджень** – моделі та методи автоматизації ключових HR процесів.

**Мета роботи** – програмна розробка системи, призначеної для підвищення ефективності при керуванні персоналом всередині компанії, а також для забезпечення полегшеної моделі відбору нових кадрів.

## 3 ОЧІКУВАНІ НАУКОВІ РЕЗУЛЬТАТИ

**Наукова новизна** полягає в тому, що було удосконалено існуючі на ринку рішення по керуванню персоналом за допомогою більш актуальних технологій на поточний час.

**Практична цінність** результатів полягає в тому, що запропоноване рішення дозволяє мінімізувати час на менеджмент людського ресурсу, а також значно полегшує процес відбору нових кадрів на ту чи іншу посаду.

## 4 ЕТАПИ ВИКОНАННЯ РОБІТ

| **Найменування етапів робіт** | **Строки виконання робіт(початок – кінець** |
|---|---|
| Аналіз предметної галузі та постановка задачі | 12.09.2022-30.09.2022 |
| Дослідження методів реалізації HRM систем. Функціональні можливості фреймворку Odoo | 01.10.2022-31.10.2022 |
| Проектування та розробка програмного продукту | 01.11.2022-14.12.2022 |

Завдання видав _____ *Мороз Б.І.*
(підпис) (прізвище, ініціали)

Завдання прийняв до виконання _____ *Смигунов І.В.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *12.09.2022*

Термін подання дипломного проекту до ЕК *16.12.2022*

# РЕФЕРАТ

Пояснювальна записка: 109 стор., 29 рис., 14 таблиць, 3 додатка, 17 джерел.

**Об'єкт дослідження:** процес керуванням існуючим людським ресурсом, а також відбору кандидатів на посади.

**Предмет дослідження:** моделі та методи автоматизації ключових HR процесів.

**Мета магістерської роботи:** дослідження та програмна розробка системи, призначеної для підвищення ефективності при керуванні персоналом всередині компанії, а також для забезпечення полегшеної моделі відбору нових кадрів.

**Методи дослідження.** Для вирішення поставлених задач використані методи: аналізу даних, теоретичні основи проектування реляційних баз даних, об'єктно-орієнтоване програмування.

**Наукова новизна** полягає в тому, що було удосконалено існуючі на ринку рішення по керуванню персоналом за допомогою більш актуальних технологій на поточний час.

**Практична цінність** результатів полягає в тому, що запропоноване рішення дозволяє мінімізувати час на менеджмент людського ресурсу, а також значно полегшує процес відбору нових кадрів на ту чи іншу посаду.

**У розділі «Економіка»** проведені розрахунки трудомісткості розробки програмного забезпечення, витрат на створення ПЗ і тривалості його розробки.

**Список ключових слів:** HRMS, Odoo, персонал, XML, автоматизована система, бізнес, база даних, фреймворк, ERP.

## ABSTRACT

**Explanatory note:** 109 pages, 29 figures, 14 tables, 3 applications, 17 sources.

**Object of research:** the process of managing existing human resources, as well as selecting candidates for positions.

**Subject of research:** models and methods of automation of key HR processes.

**Purpose of Master's thesis:** software development of a system designed to increase efficiency in personnel management within the company, as well as to provide a simplified model for the selection of new personnel.

**Research methods.** To solve the problems, the following methods were used: data analysis, theoretical foundations of designing relational databases, object-oriented programming.

**Originality of research is** that the solutions available on the market for personnel management have been improved with the help of more relevant technologies for the current time.

**Practical value of the results lies in** the fact that the proposed solution allows to minimize the time spent on human resource management, and also greatly facilitates the process of selecting new personnel for one or another position.

**In the Economics section we calculated** the complexity of software development, the costs of software creation and the duration of its development were carried out.



**Keywords:** HRMS, Odoo, HR, XML, Automated System, Business, Database, Framework, ERP.

**ЗМІСТ**

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – база даних;

HR – Human Resources;

HRMS – Human Resources Management System;

XML – Extensible Markup Language;

ERP – Enterprise Resource Planning;

ОС – операційна система.

# ВСТУП

Успіх і зростання будь-якої організації залежить від того, наскільки добре керують її співробітниками та ставляться до них. Ось чому деякі організації випробовують різні способи вдосконалення своїх стратегій управління персоналом і досвіду. Саме тут програмне забезпечення для HR справді виявляє свою очевидну користь. Використовуючи програмне забезпечення для управління персоналом, управління співробітниками стає легшим та забезпечується чудовий досвід, який підтримуватиме мотивацію та задоволення співробітників. Кожен аспект роботи з кадрами, можна виконувати з централізованого місця за допомогою програмного забезпечення для кадрів. Він бере на себе всю адміністративну роботу, щоб ви могли піклуватися про своїх співробітників.

Дана магістерська робота поділена на 3 частини.

В першому розділі ми проаналізуємо предметну область, визначимо актуальність поставленої задачі та призначення розробки нашого проекту, ознайомимося з базової термінологією HR, та виокремимо важливість створення HRM систем.

Другий розділ присвячений дослідженню. Буде проаналізовано ринок HRM систем та виокремлено основних розробників даного продукту. Окрім того ми ознайомимося з базовою структурою модулів, архітектурою, методами та інструментами реалізації автоматизованої системи керування персоналом, а також методикою оцінки комплексної ефективності управління персоналом підприємства.

Третій розділ містить інформацію про розробку системи, зокрема, опис структури проекту та бази даних. Також буде розглянуто інтерфейс користувача для взаємодії з системою.

# РОЗДІЛ 1
# АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

## 1.1. Загальні відомості з предметної області

Управління людськими ресурсами (HRM або HR) — це стратегічний і послідовний підхід до ефективного та результативного управління людьми в компанії чи організації, щоб вони допомагали своєму бізнесу отримати конкурентну перевагу. Його створено для максимізації ефективності роботи співробітників на службі стратегічних цілей роботодавця [1]. Управління людськими ресурсами в першу чергу стосується управління людьми в організаціях, зосереджуючись на політиках і системах. Відділи кадрів несуть відповідальність за нагляд за розробкою виплат працівникам, наймом працівників, навчанням і розвитком, оцінкою ефективності та управлінням винагородами, наприклад, управління системами оплати праці та виплат [2]. HR також займається організаційними змінами та виробничими відносинами або балансуванням організаційної практики з вимогами, що випливають із колективних переговорів та державних законів.

Загальна мета HR полягає в тому, щоб гарантувати, що організація здатна досягти успіху завдяки людям. Фахівці з персоналу керують людським капіталом організації та зосереджуються на реалізації політики та процесів. Вони можуть спеціалізуватися на пошуку, наймі, відборі, навчанні та розвитку працівників, а також на підтримці стосунків із працівниками чи пільг. Фахівці з навчання та розвитку гарантують, що співробітники навчаються та постійно розвиваються. Це робиться за допомогою програм навчання, оцінки ефективності та програм винагороди. Відносини з працівниками займаються проблемами працівників, коли вони порушуються, наприклад у випадках переслідування чи дискримінації. Управління виплатами працівникам включає розробку компенсаційних структур, програм батьківських відпусток,

знижок та інших пільг для працівників. На іншій стороні поля знаходяться кадрові спеціалісти широкого профілю або ділові партнери. Ці спеціалісти з кадрів можуть працювати в усіх сферах або бути представниками трудових відносин, які працюють із працівниками, які є членами профспілок.

HR є продуктом руху людських відносин початку 20-го століття, коли дослідники почали документувати шляхи створення цінності бізнесу через стратегічне управління робочою силою[3]. Спочатку в ньому домінувала трансакційна робота, наприклад нарахування заробітної плати та адміністрування пільг, але через глобалізацію, консолідацію компаній, технологічний прогрес і подальші дослідження з 2015 року HR зосереджується на стратегічних ініціативах, таких як злиття та поглинання, управління талантами, планування спадкоємства, промислові і трудові відносини, і різноманітність та інклюзивність. У поточному глобальному робочому середовищі більшість компаній зосереджуються на зниженні плинності кадрів і збереженні талантів і знань, якими володіє їхня робоча сила. Нове наймання не тільки передбачає високі витрати, але й збільшує ризик того, що новий працівник не зможе адекватно замінити посаду попереднього працівника. Відділи кадрів прагнуть запропонувати переваги, які будуть привабливі для працівників, таким чином зменшуючи ризик втрати відданості та психологічної відповідальності співробітників.

HRMS, або система управління людськими ресурсами, — це набір програмних застосунків, які використовуються для управління людськими ресурсами та пов’язаними процесами протягом життєвого циклу співробітника. HRMS дозволяє компанії повністю розуміти свою робочу силу, дотримуючись змін у податковому законодавстві та правилах праці.

Керівники відділу кадрів і персонал є основними користувачами, враховуючи, що вони керують повсякденними процесами, що пов’язані з робочою силою та відповідають за дотримання вимог і звітність про

ефективність. Однак HR — не єдиний відділ, який має привілегії. Компанії можуть надати менеджерам і співробітникам можливість самообслуговування для виконання звичайних завдань, що є важливою перевагою для молодих працівників. Керівники можуть використовувати HRMS для створення даних про тенденції в робочій силі та їхні наслідки для бізнесу.

HRM-системи призначені для управління персоналом, але їхня функціональність ширша, ніж у систем автоматизації кадрових операцій. Продукти цього класу дозволяють працювати не лише з кількісними, а й із якісними показниками персоналу. Основне їх завдання - залучити та утримати цінних для компанії фахівців.

Під HRM-системою розуміється автоматизована комплексна система керування персоналом. У порівнянні з традиційними системами автоматизації кадрового обліку та розрахунку зарплати HRM-системи мають розширену функціональність. Крім облікового (кадровий облік, штатний розпис, документообіг, облік робочого часу та відпусток, пенсійний та військовий облік та ін.) та розрахункового (зарплата, податкові виплати, надбавки та відрахування тощо) контурів, що обробляють кількісні дані, подібні системи також включають як такий HR-контур, призначений для роботи з якісними показниками персоналу.

У 1970-х роках, коли компанії прагнули автоматизувати управління своїми людьми, нарахування заробітної плати стало першою комп’ютеризованою функцією HRMS. Але щоб розрахувати заробіток працівника, утримати відрахування, надрукувати паперовий чек і відстежувати зобов’язання по заробітній платі, потрібна технологія мейнфрейму. Лише на початку 2000-х, із широким впровадженням прямого депозиту та самообслуговування працівників, процес нарахування заробітної плати став повністю електронним.

Наприкінці 1980-х років PeopleSoft була однією з перших, хто розробив повнішу систему HRMS. На додаток до нарахування заробітної плати, він пропонував керування обліковими записами співробітників, підбір персоналу, контроль робочого часу та відвідуваності, адміністрування пільг, компенсацію, звітність про відповідність та інші функції, які допомагають спеціалістам з управління персоналом автоматизувати більшу частину життєвого циклу співробітників і приймати кращі рішення щодо робочої сили.

Розвиток Інтернету наприкінці 1990-х приніс переваги автоматизації ще більшій кількості процесів у сфері управління персоналом. Наприклад, паперові оголошення про пошук допомоги були замінені на електронні дошки вакансій, що дало рекрутерам і кандидатам нові способи зв'язку. У 2010-х роках хмарна технологія була основною — тепер відділи кадрів у компаніях будь-якого розміру могли дозволити собі набір програм, не інвестуючи у дороге комп'ютерне обладнання чи IT-спеціаліст для керування та обслуговування системи.

Незважаючи на те, що витрати на кадри, особливо на офісне приміщення, зараз постійно змінюються враховуючи перехід до моделі роботи з дому, компанії все ще повинні точно розраховувати витрати на оплату праці, щоб підтримувати поточний KPI доходу на одного працівника. Джозеф Хадзіма, старший викладач Школи менеджменту Массачусетського технологічного інституту Слоуна, вважає, що базова заробітна плата плюс податки на працевлаштування та пільги зазвичай додають до 1,25-1,4 річної зарплати. Таким чином, робітник із 50 000 доларів на рік може коштувати від 62 500 до 70 000 доларів, не враховуючи нерухомість і обладнання, як-от комп'ютери та телефони [4].

Крім того, компанії з перенапруженими відділами кадрів повинні розгортати можливості самообслуговування. Немає жодних причин для фахівця з персоналу витрачати час, допомагаючи менеджеру, наприклад, із

регулярними оновленнями відпрацьованих годин або допомагаючи працівникам отримати доступ до таких форм, як W-2.

На щастя, точні фінансові дані та безпечне самообслуговування є лише двома перевагами сучасної системи управління персоналом.

### 1.2. Призначення розробки та область застосування

Розробка системи керуванням персоналом (HRM система) з метою автоматизації ключових HR процесів.

Розроблений продукт має використовуватися користувачами (співробітниками) для отримання досвіду комфортного здійснення повсякденних операцій, що стосується організаційних процесів, а також процесами рекрутинга (найму на нову посаду).

Одним з призначень розробки є створення продукту, який зараз є затребуваним на українському ринку через російське вторгнення на територію держави, адже нині одним з найпопулярніших інструментів для організації системи керуванням персоналу є 1С, що є російським. Також, призначення розробки полягає в наданні користувачам таких переваг автоматизованої системи, як зрозумілий інтерфейс, просте керування співробітниками та кандидатами, створення нових подій в календарі тощо.

Функціонально в структурі нашої автоматизованої системи управління персоналом присутні блоки:

– Employee. Містить весь потрібний функціонал для перегляду, створення, редагування та видалення співробітників;

– Departments. Містить весь потрібний функціонал для перегляду, створення, редагування та видалення департаментів;

– Recruitment. Дає можливість створювати вакансії та кандидатів, зв’язувати їх та проводити повноцінний відбір;

– Contacts. Містить записи щодо контактних даних потрібних осіб в компанії.

– Calendar. Дозволяє брати відгули, а також організовувати нові зустрічі онлайн.

– Configuration. Блок, що містить деякі налаштування системи. Наприклад, створення нових посад або розподіл доступних днів відпустки.

### 1.3. Підстави для розробки

Згідно з навчальним планом та відповідно до навчальної програми та графіка навчального процесу, наприкінці навчання студентами виконується магістерська робота.

Тема проекту є узгодженою з його керівником, кафедрою, а також затверджена наказом ректора

Отже, підставами для виконання кваліфікаційної роботи є:

- навчальна програма спеціальності 121 «Інженерія програмного забезпечення»;
- навчальний план та графік навчального процесу;
- наказ ректора Національного технічного університету «Дніпровська політехніка» № 1200-с від 31.10.2022 р.;
- завдання на магістерську роботу: «Розробка системи керуванням персоналом з метою автоматизації ключових HR процесів».

### 1.4. Постановка завдання

Завданням є розробка автоматизованої системи для управління персоналом та відбору нових співробітників. Обсяг вимог до інформаційного

наповнення даної програми - створити швидку і комфортну для користувача систему з гнучким функціоналом, зрозумілим і доступним інтерфейсом, можливістю подальшого розширення спектру доступних функцій шляхом встановлення додаткових модулів.

Розробка системи керування людським ресурсом відповідно до структури бази даних, який буде володіти наступним функціоналом:

– можливість гнучкого відбору кандидатів та створенням нового співробітника на основі кандидата, що успішно пройшов відбір;

– створення нових співробітників та їх зручне керування, наприклад, зміна департаменту, надання певних прав тощо;

– дозволяє створити базу контактів, за допомогою якої можна буде легко зв’язатися з людиною;

– дає можливість влаштовувати співбесіди та інші події за допомогою зручного блоку з календарем;

– можливість взяти відгул/відпустку потрібного типу або створити новий тип.

## 1.5. Вимоги до програми або програмного виробу

### 1.5.1. Вимоги до функціональних характеристик

Система повинна забезпечувати можливість виконання наступних функцій:

– ініціалізацію системи (введення списків співробітників, переліків департаментів тощо);

– зберігання інформації по всім сутностям (співробітник, кандидат, тощо) в базу даних;

– створення нових записів на основі існуючих за допомогою спеціальних функціональних кнопок;

– отримання відомостей про поточний стан кожного запису.

### 1.5.2. Вимоги до інформаційної безпеки

Основними вимогами до інформаційної безпеки є:

– забезпечення цілісності даних;
– захищеність від втручання в функціонал програмного продукту;
– конфіденційність інформації;
– налаштування доступу до того чи іншого елементу системи за допомогою ролей;
– доступність інформації для усіх користувачів, які пройшли авторизацію.

Для того, щоб програмне забезпечення надійно функціонувало необхідно дотримуватися наступних вимог:

– використання лише ліцензійного програмного забезпечення;
– встановлення блоків безперебійного живлення;
– захищеність від зловмисних програм, які можуть завдати шкоди ПЗ.

### 1.5.3. Вимоги до складу та параметрів технічних засобів

Система запускається на віддаленому сервері, доступ до інформації відбувається через браузер (клієнт), тому даний продукт не є ресурсомістким для користувача, проте для коректного функціонування кінцевого продукту, а саме автоматизованої системи управління ключовими HR процесами необхідними технічними умовами є:

– операційна система Windows, Linux, MacOS, Android або IOS;
– обсяг оперативної пам’яті не менш 4 ГБ;

– наявність будь-якого сучасного браузера, наприклад, Google Chrome або Opera;
– підтримка високошвидкісного Інтернету.

### 1.5.4. Вимоги до інформаційної та програмної сумісності

Для повної сумісності використовуваного для взаємодії з системою девайсу обов’язковими вимогами є встановлення актуальної версії мережевого драйвера, наявність стабільно швидкого Інтернет з’єднання та сучасного браузера, як Google Chrome, Opera, Safari тощо.

## 1.6. Висновки

У першому розділі ми проаналізували предметну область, визначили актуальність поставленої задачі та призначення розробки нашого проекту, ознайомимося з базової термінологією HR, та виокремили важливість створення HRM систем. Також було наведено вимоги щодо функціональних характеристик, інформаційної безпеки, параметрів технічних засобів та сумісності.

# РОЗДІЛ 2
# ДОСЛІДЖЕННЯ МЕТОДІВ РЕАЛІЗАЦІЇ HRM СИСТЕМ. ФУНКЦІОНАЛЬНІ МОЖЛИВОСТІ ФРЕЙМВОРКУ ODOO

## 2.1. Аналіз ринку HRM систем

Обсяг світового ринку систем управління персоналом (Human Resource Management, HRM) зростатиме в середньому на 12,8% щорічно і сягне $56,15 млрд до 2030 року. Такий прогноз аналітики Grand View Research зробили у вересні 2022 року.

За словами експертів, основними драйверами зростання ринку стануть аналітичні інструменти, хмарні та мобільні технології, а також рішення у сфері штучного інтелекту та інтернету речей. Зазначається, що HR-аналітика та її інтеграція з HRM-системами з метою ухвалення обґрунтованих рішень є вкрай важливими для розвитку галузі.

Згідно з очікуваннями експертів, світові витрати на професійні послуги у сфері управління персоналом у 2022-2030 роках. зростатимуть на 12,4% щорічно і становитимуть $7,47 млрд до кінця цього періоду. Зростання попиту на HR-процеси і необхідність керувати віддаленими командами у зв'язку з трансформацією робочих місць, що триває, створюють попит на професійні послуги у сфері HR, зазначено в доповіді.

У період пандемії коронавірусу особливою популярністю у багатьох галузях стали користуватися інструменти автоматизованої оптимізації роботи персоналу. Таке ПЗ, як правило, використовує алгоритми штучного інтелекту, дозволяє складати графіки роботи співробітників, оцінювати зміни попиту та зіставляти роботу персоналу з їхніми робочими процесами.

Найбільшими виробниками систем управління персоналом дослідники назвали такі компанії (перераховані в алфавітному порядку):

- Accenture;
- ADP;
- Cezanne HR;
- Ceridian HCM;
- IBM;
- Kronos;
- Mercer;
- Oracle;
- PwC;
- SAP;
- Cegid;
- UKG;
- Workday.

### 2.2. Методи впровадження системи на український ринок

На поточний час найбільшу долю ринку розробки систем для бізнесу займає 1С. Програмні продукти російської компанії «1С» з початку 2000-х історично є одними з найпопулярніших рішень для автоматизації управління та обліку на підприємствах на пострадянському просторі, зокрема в Україні. Навіть попри те, що на продукти цього розробника з 2017 року накладені санкції, впродовж кількох останніх років аж до лютого 2022-го в українських компаніях був стабільний попит на спеціалістів «1С».

Після початку повномасштабного вторгнення розробники програмного забезпечення для підприємств та представники бізнесів будь-якого рівня

почали шукати альтернативи 1С. Однією з таких альтернатив є продукт бельгійської компанії Odoo S. A.

Серед вже реалізованих модулів системи — бухгалтерський облік, CRM, виробництво, продаж, закупівля, управління складом, управління проектами, управління транспортом, управління претензіями, POS, є модуль інтеграції із соціальними мережами.

Експертами прогнозується ріст долі Odoo на ринку України найближчими роками і це не викликає подиву, адже Odoo має дуже гнучкий та зрозумілий функціонал, а також надає розробниками майже необмежені можливості для створення нового функціоналу та кастомізації вже існуючого.

Odoo вже зарекомендовала себе щодо ефективного управління та реалізації як підприємницької та державної діяльності на усіх етапах розвитку. Використання та впровадження Odoo сприятиме розвитку культури введення бізнесу в Україні та забезпечить підвищення якості знань підприємців у сфері ведення бізнесу.

## 2.3. Методика оцінки комплексної ефективності управління персоналом підприємства

Для більш детального розуміння поставленої задачі нам необхідно ознайомитися з тим, як вимірюється ефективність керування персоналом.

Розрахунок комплексної ефективності керування персоналом підприємства визначається як відношення фактично досягнутих відповідних показників до базисних значень кінцевих результатів діяльності, що зважені ваговими коефіцієнтами значимості функцій керівництва щодо нормативного значення ефективності, що дорівнює 100 балам.

Методика оцінки комплексної ефективності управління персоналом скаладється з методів економічного аналізу, експертних оцінок, бальному методі і теорії класифікації [6].

Метод експертних оцінок та кореляційний аналіз визначаються послідовність економічних, соціальних і організаційних показників, які характеризують кінцеві результати діяльності підприємства.

Список даних показників обумовлений вивченням нормативнозаконодавчих актів, матеріалів підприємства, форм і інструкцій для складання статистичної та оперативної звітності.

Задаються критерії досягнення встановлених кінцевих результатів з щонайменшими витратами ресурсів та високою якістю продукції. Чисельні значення даних критеріїв визначаються з фінансових документів, форм статистичної і оперативної звітності і розраховуються у виді відсоткового відношення фактичного значення кінцевого результату до базисного (2.1):

$$Xi = \frac{Рфi}{Рбi} * 100\%, \qquad (2.1)$$

де

Xi – процентне відношення і-го часткового показника ефективності, %;

Рфі – фактичне значення і-го показника кінцевого результату за звітний період;

Рбі – базисне значення і-го показника кінцевого результату (план, норматив, факт попереднього періоду) за звітний період.

Показник Xi відображає ступінь досягнення фактичного кінцевого результату.

Для того, щоб критерії більш об'єктивно відображали результати керування персоналом та не перекривали один одного, вони попередньо піддаються корегуванню по формулі (2.2).

$$Пі = \int f(Хі), \qquad (2.2)$$

де

Пі – чисельне значення скорегованого і-го показника кінцевого результату, %;

f(Хі) – математична функція корегування і-го показника.

При цьому використовують 4 залежності:

- лінійна висхідна (П = Х), коли заохочується кожен відсоток досягнення кінцевого результату, а при недовиконанні приймається його фактичне значення (має відношення до соціальних показників);
- лінійна низхідна (П = 200 – Х), коли заохочується досягнення результату з найменшими витратами ресурсів, а за перевитрату ресурсів нараховується менша кількість балів. Ця функція застосовується для таких ресурсних показників, як витрати на 1 грн. товарообігу, фонд заробітної плати, плинність робочих кадрів, втрати робочого часу;
- піраміда, коли заохочується тільки 100-відсоткове досягнення кінцевого результату і не заохочується недовиконання або перевиконання. При цьому чисельне значення скорегованого показника до 100% визначається за формулою П = Х, а при Х > 100 % – за формулою П = 200 – Х;
- лінійна зворотна (штрафні санкції) передбачає нарахування негативних відсотків за формулою П = -Х, коли чисельне значення такого показника приводить до негативних явищ у діяльності підприємства (наприклад, розкрадання матеріальних цінностей, виробничий травматизм, порушення трудової дисципліни). Ці показники не плануються, а враховуються у виді штрафних санкцій.

За допомогою зазначених показників розраховується комплексний показник ефективності, у якому порівнюються різні економічні, соціальні і організаційні показники з урахуванням їх важливості.

Комплексний показник ефективності розраховується як сума часткових показників ефективності, що відображають кінцеві результати діяльності підприємства, використання ресурсів, соціальну діяльність і організаційну результативність праці співробітників підприємства.

Часткові показники визначаються за результатами виконання економічних, соціальних і організаційних показників шляхом множення відсотків їхнього виконання на вагові коефіцієнти значимості функцій керівництва (2.3):

$$ЧПі = Пі * Ві, \quad (2.3)$$

де

ЧПі – значення і-го часткового показника ефективності управління персоналом, бали;

Пі – виконання економічних, соціальних і організаційних показників, %;

Ві – ваговий коефіцієнт і-го часткового показника, частка.

Ваговий коефіцієнт показує відносну важливість відповідного показника у загальній сукупності показників комплексної ефективності.

Він також вводиться для усунення розбіжності інтересів підприємства, трудового колективу і кожного працівника. Вагові коефіцієнти прямо пропорційно впливають на величину часткових показників ефективності роботи.

Вони визначаються методом експертних оцінок, шляхом ранжирування показників із присвоєнням їм питомих ваг у частках одиниці. При цьому для окремих показників кінцевих результатів діяльності підприємства доцільно зафіксувати певне значення частки в розмірі не менш 0,5, залишивши на показники результативності праці, організаційної і соціальної ефективності питому вагу 0,5. У цьому випадку буде дотримуватися пріоритет результатів економічного розвитку підприємства.

Комплексний показник ефективності управління персоналом підприємства (е у балах) визначається за формулою (2.4):

$$\text{Еуп} = \sum_{i=1}^{n}(\text{Пі} - \text{Ві}) \Big/ \sum_{i=1}^{n} \text{Ві} * 100, \qquad (2.4)$$

де

n – кількість часткових показників ефективності.

Оцінка підсумкового значення комплексного показника ефективності управління

персоналом підприємства залежить від його чисельного значення:

– якщо воно менш 95 балів, то управління персоналом підприємства є незадовільним;

– якщо воно знаходиться у діапазоні 95-100 балів, то управління персоналом підприємства є задовільним (але не використані усі можливості);

– якщо воно знаходиться у діапазоні 100-105 балів і виконані усі часткові показники, то управління персоналом підприємства є добрим;

– якщо воно більш 105 балів, то загальна оцінка ефективності управління персоналом підприємства є відмінною.

## 2.4. Функціональне призначення системи

Призначення та ціль створеної автоматизованої системи керування ключовими HR-процесами – спрощення взаємодії менеджерів та звичайних співробітників з повсякденними кадровими задачами, а також реалізація вбудованого відбору кандидатів, що дозволяє рекрутерам досягати максимальної ефективності.

У зв'язку з цим наша система повинна задовольнити наступним умовам:

– максимальна швидкість доступу до інформації;

– представлення інформації повинно бути максимально простим і чітким;

– швидкий доступ до всіх функціональних розділів;

– розділення функціоналу системи на ролі для того, щоб користувачі мали доступ лише до тієї інформації, яка відповідає їх посаді, таким чином мінімізується вірогідність порушення конфіденційності та непередбачуваних проблем в системі.

### 2.5. Опис базової архітектури та шаблонів проектування

У програмній інженерії шаблоном проектування є загальне повторюване рішення загальнопоширеної проблеми при розробці програмного забезпечення. Об'єктно-орієнтований шаблон зазвичай є зразком розв’язання проблеми та відображає відношення між класами та об'єктами, без чіткої вказівки на те, яким способом буде зрештою реалізоване дане відношення [5].

Фреймворк Odoo, на якому буде побудовано систему, має багаторівневу архітектуру, що означає, що презентація, бізнес-логіка та зберігання даних розділені. Точніше, він використовує трирівневу архітектуру.

У розробці програмного забезпечення багаторівнева архітектура (часто її називають n-рівневою архітектурою) — це архітектура клієнт-сервер, у якій функції представлення, обробки додатків і керування даними фізично розділені. Найбільш поширеним використанням багаторівневої архітектури є трирівнева архітектура. [7].

Багаторівнева архітектура додатків забезпечує модель, за допомогою якої розробники можуть створювати гнучкі додатки, які можна багаторазово використовувати. Поділяючи програму на рівні, розробники отримують можливість змінювати або додавати певний рівень замість того, щоб

переробляти всю програму. Трирівнева архітектура зазвичай складається з рівня презентації, рівня логіки та рівня даних [8].

**Презентаційний рівень** - найвищий рівень програми. Він спілкується з іншими рівнями, за допомогою яких видає результати на рівень браузера/клієнта та на всі інші рівні в мережі. Простіше кажучи, це рівень, до якого користувачі мають прямий доступ (наприклад, веб-сторінка або графічний інтерфейс операційної системи). В Odoo рівень презентації – це комбінація HTML5, JavaScript і CSS.

**Логічний рівень** витягується з рівня презентації, і, як його рівень, він контролює функціональність програми, виконуючи детальну обробку. Цей рівень координує програму, обробляє команди, приймає логічні рішення та оцінки, а також виконує обчислення. Він також переміщує та обробляє дані між двома навколишніми шарами. В Odoo логічний рівень пишеться виключно мовою Python.

**Рівень даних** включає механізми збереження даних (сервери баз даних, спільні файли тощо) і рівень доступу до даних, який інкапсулює механізми збереження та відкриває дані. Рівень даних в Odoo підтримує лише PostgreSQL як RDBMS.

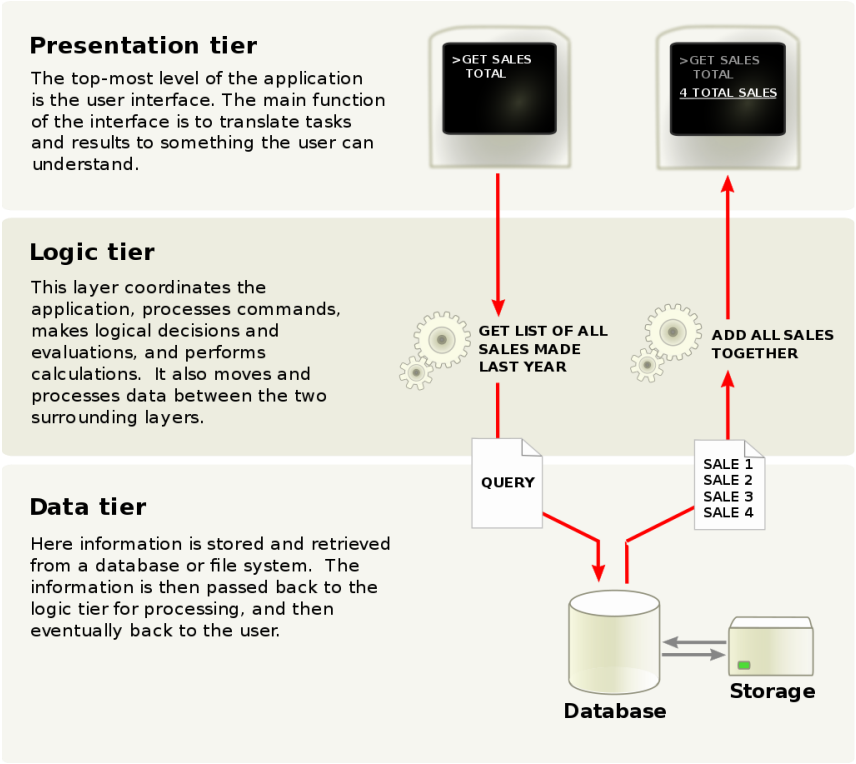



Рис. 2.1. Багаторівнева архітектура схематично

## 2.6. Базова структура модулів

Кожен модуль Odoo може містити такий набір елементів:

**Бізнес-об’єкт** - оголошується як клас Python. Поля, визначені в цих класах, автоматично зіставляються зі стовпцями бази даних завдяки рівні ORM.

**Представлення об'єктів** - визначає відображення інтерфейсу користувача, тобто – як об’єкт буде представлений на боці клієнта. Дані представлення описуються в XML файлах.

**Файли даних** - файли XML або CSV, в яких описуються записи в базі даних, які повинні бути створені при встановленні модуля.

Кожен модуль є каталогом у каталозі модуля. Каталоги модулів вказуються за допомогою параметра --addons-path.

Модуль Odoo оголошується його маніфестом.

Якщо модуль Odoo містить бізнес-об’єкти (тобто файли Python), вони організовані як пакет Python із файлом __init__.py. Цей файл містить інструкції з імпорту для різних файлів Python у модулі.

```
module
├── models
│   ├── *.py
│   └── __init__.py
├── data
│   └── *.xml
├── __init__.py
└── __manifest__.py
```

2.2 Базова структура модуля

## 2.7. Методи реалізації поставленої задачі та обрані технології

Для реалізації серйозних бізнес проектів, які потребують підвищеної надійності і безпеки від несанкціонованого доступу, потрібні правильно підібрані інструменти - мови програмування, фреймворки.

Всі мови WEB-програмування можна класифікувати на клієнтські і серверні. Як випливає з назви, клієнтські мови використовуються для написання програм, які виконуються на стороні клієнта (WEB-браузер), а серверні - для програм, які виконуються на сервері.

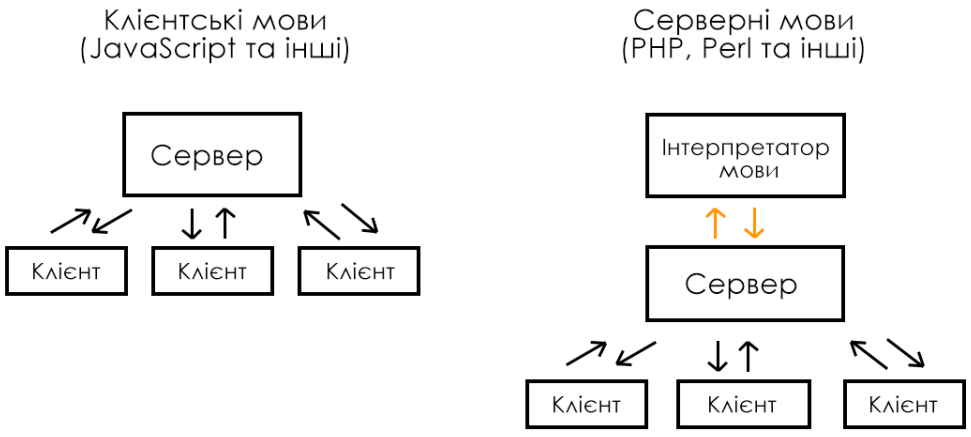


Рис 2.3. Уявлення клієнтських та серверних мов програмування

Правила та архітектура серверних фреймворків не дає можливості розробити веб-додаток з багатим інтерфейсом. Вони обмежені в своїй функціональності, проте ви все одно можете створювати прості сторінки і різні форми. Також вони можуть формувати вихідні дані і відповідати за безпеку в разі атак.

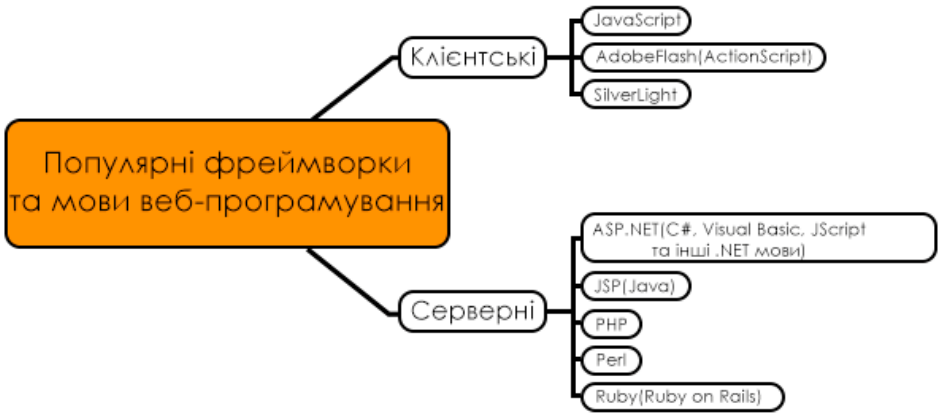



Рис 2.4. Популярні фреймворки та мови веб-програмування

Для виконання даної магістерської роботи були використані такі інструменти, як об'єктно-орієнтована мова програмування Python, фреймворк з відкритим кодом Odoo, система керування реляційними базами даних PostgreSQL, мова розмітки XML для створення представлень та мова розмітки HTML для побудови email шаблонів.

PYTHON

Python — це мова програмування високого рівня загального призначення. Його філософія дизайну наголошує на читабельності коду з використанням значних відступів [9].

Python є динамічно типізованою мовою. Вона підтримує кілька парадигм програмування, включаючи структуроване (зокрема процедурне), об’єктно-орієнтоване та функціональне програмування. Її часто описують як мову з «батарейками в комплекті» через її повну стандартну бібліотеку [10].

Гвідо ван Россум почав працювати над Python наприкінці 1980-х як наступником мови програмування ABC і вперше випустив її в 1991 році як Python 0.9.0 [11]. Python 2.0 був випущений у 2000 році та представив нові функції, такі як розуміння списків, збирання сміття з визначенням циклу, підрахунок посилань і підтримка Unicode. Python 3.0, випущений у 2008 році, був основною версією, яка не повністю сумісна з попередніми версіями. Python 2 було припинено з версією 2.7.18 у 2020 році [12].


Рис 2.5. Логотип Python

## ODOO

Odoo — це набір програмних засобів для управління бізнесом, включаючи, наприклад, CRM, електронну комерцію, виставлення рахунків, бухгалтерський облік, виробництво, склад, управління проектами та управління запасами [13].

За допомогою Odoo Python Framework програмісти можуть створювати нові функціональні модулі або розширювати існуючі.

В Odoo і серверні, і клієнтські розширення упаковані у вигляді модулів, які за бажанням завантажуються в базу даних. Модуль — це набір функцій і даних, призначених для однієї мети.

Модулі Odoo можуть або додати нову бізнес-логіку до системи Odoo, або змінити та розширити існуючу бізнес-логіку. Один модуль можна створити, щоб додати правила бухгалтерського обліку вашої країни до загальної підтримки бухгалтерського обліку Odoo, тоді як інший модуль може додати підтримку візуалізації автобусного парку в реальному часі.

Все в Odoo починається і закінчується модулями.

Розробники групують свої бізнес-функції в модулях Odoo. Основні модулі, призначені для користувача, позначаються як програми, але більшість модулів не є програмами. Модулі також можуть називатися аддонами, а каталоги, де сервер Odoo їх знаходить, утворюють addons_path.

Розширювана архітектура Odoo дозволяє великій кількості фрілансерів і організацій розробляти програми або модулі Odoo і розміщувати їх на ринку для продажу або безкоштовного завантаження. Основними компонентами Odoo є фреймворк, близько 30 основних програм (їх також називають офіційними модулями) і тисячі модулів спільноти.

Рис 2.6. Логотип Odoo

## POSTGRESQL

PostgreSQL, також відомий як Postgres, — це безкоштовна система керування реляційною базою даних (RDBMS) із відкритим вихідним кодом, яка наголошує на розширюваності та сумісності з SQL. Спочатку вона була

названа POSTGRES, посилаючись на її походження як наступницю бази даних Ingres, розробленої в Каліфорнійському університеті в Берклі [14]. У 1996 році проект було перейменовано на PostgreSQL, щоб відобразити підтримку SQL. Після перегляду в 2007 році команда розробників вирішила зберегти назву PostgreSQL і псевдонім Postgres [15].

Використання Postgresql має велику кількість переваг, а саме:

– **Об'єктно-реляційна модель.** Традиційно популярні СУБД – реляційні. Це означає, що дані, які в них зберігаються, подаються у вигляді записів, пов'язаних між собою відносинами, — relations. Виходять пов'язані списки, які можуть мати між собою ті чи інші відносини, - так і утворюється таблиця.

– **Підтримка великої кількості типів даних.** Ще одна особливість PostgreSQL – підтримка великої кількості типів запису інформації. Це не лише стандартні цілочисельні значення, числа з плаваючою точкою, рядки та булеві значення («так/ні»), а й грошовий, геометричний, перерахований, бінарний та інші типи. PostgreSQL «з коробки» підтримує бітові рядки та мережеві адреси, масиви даних, у тому числі багатовимірні, композитні типи та інші складні структури. У ній є підтримка XML, JSON та NoSQL-баз.

– **Робота із великими обсягами.** У більшості СУБД, розрахованих на середні та невеликі проекти, є обмеження за обсягом бази та кількістю записів у ній. У PostgreSQL обмежень немає.

– **Підтримка складних запитів.** PostgreSQL працює зі складними, складовими запитами. Система справляється із завданнями розбору та виконання трудомістких операцій, які мають на увазі і читання, і запис, і валідацію одночасно. Вона повільніша за аналоги, якщо мова заходить тільки про читання, але в інших аспектах перевершує конкурентів.

– **Написання функцій кількома мовами.** У PostgreSQL можна писати власні функції - блоки користувача коду, які виконують ті чи інші дії.

Ця можливість є практично у будь-яких СУБД, але PostgreSQL підтримує більше мов, ніж аналоги. Крім стандартного SQL, у PostgreSQL можна писати на C і C ++, Java, Python, PHP, Lua та Ruby. Він підтримує V8 - один з двигунів JavaScript, тому JS теж можна використовувати разом з PgSQL. Реалізовано підтримку Delphi, Lisp та інших рідкісних мов. За потреби можна розширити систему під інші ЯП.

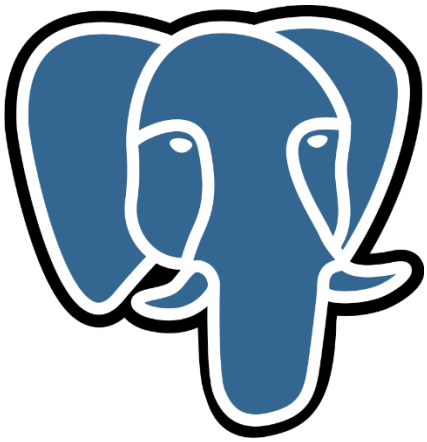


Рис 2.7. Логотип PostgreSQL

XML

Extensible Markup Language (XML) — це мова розмітки та формат файлів для зберігання, передачі та реконструкції довільних даних. Він визначає набір правил для кодування документів у форматі, який одночасно читається людиною та машиною. Специфікація XML 1.0 Консорціуму World Wide Web від 1998 року та кілька інших пов'язаних специфікацій — усі вони є безкоштовними відкритими стандартами — визначають XML [16].

Цілі дизайну XML наголошують на простоті, загальності та зручності використання в Інтернеті. Це формат текстових даних із сильною підтримкою Unicode для різних людських мов. Незважаючи на те, що дизайн XML зосереджений на документах, ця мова широко використовується для представлення довільних структур даних, таких як ті, що використовуються у веб-службах [17].

Рис 2.8. Неофіційний логотип XML

## 2.8. Висновки

У другому розділі ми проаналізували ринок HRM систем та виокремили основних розробників даного продукту, зокрема, це світові компанії, які залучені в дану сферу протягом багатьох років. Оперуючи даними експертів, було виявлено, що обсяг світового ринку систем управління персоналом зростатиме в середньому на 12,8% щорічно. Також було проведено дослідження українського ринку HRMS та зазначено важливість створення таких систем на альтернативних 1С інструментах, таких як Odoo.

Окрім того ми ознайомилися з базовою структурою модулів, архітектурою, методами та інструментами реалізації автоматизованої системи керування персоналом.

# РОЗДІЛ 3
# ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

## 3.1. Опис структури системи керування персоналом

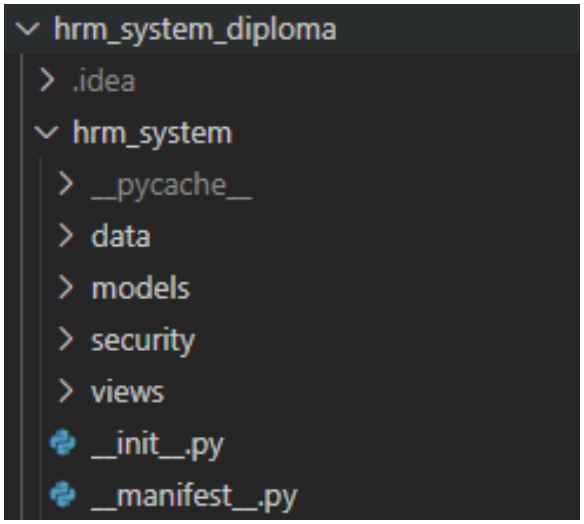
Автоматизована система керування персоналом, розроблювана протягом виконання магістерської роботи представляє собою окремий модуль.

Всі елементи такі як моделі, представлення та файли даних групуються в окремі директорії.

Обов’язковою для будь-якого модуля, в тому числі й нашого, є наявність 2-х .py файлів, а саме __init__.py та __manifest__.py

Файл __manifest__.py — це файл Python, який містить усі описи модуля в одному словнику (ім’я, автор, ліцензія, категорія, тощо). Тут же ми перелічуємо список усіх файлів даних, які повинні бути завантаженні при встановленні/оновленні модуля.

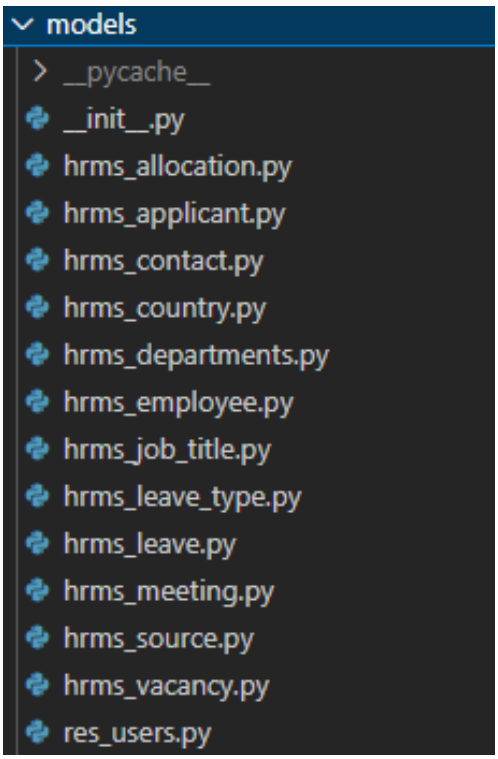
Файл __init__.py допомагає імпортувати пакети/файли Python, які потрібно завантажити в Odoo.




3.9. Структура розроблюваної системи

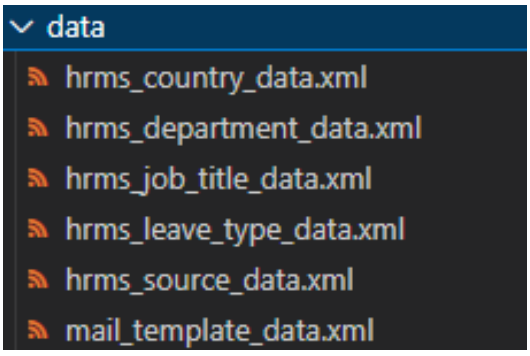

3.10. Файл __manifest__.py

Основою нашої системи є моделі, які описуються в окремих Python файлах як класи та зберігаються у папці models. В кожній моделі описуються дані таблиці, що буде створена в базі даних, мінімально - її ім’я, поля даної таблиці та методи для маніпуляції даними.




3.11. Структура папки models

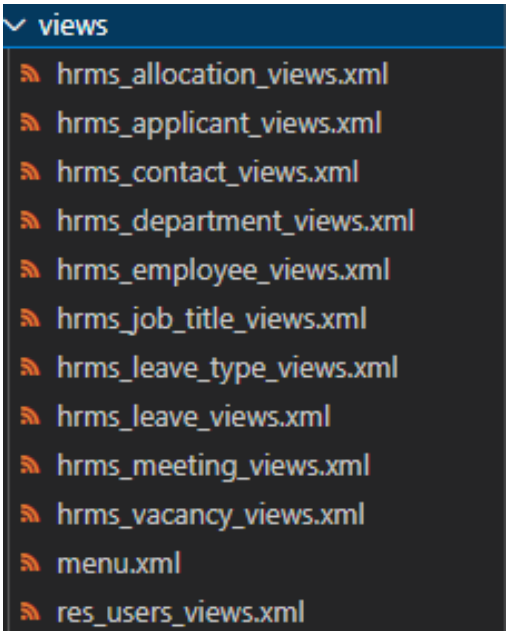
В директорії data зберігаються файли з даними, які повинні бути завантаженні при встановленні модуля. В нашому випадку це файли, що містять дані про країни, департаменти, посади, типи відпусток, джерела пошуку кандидатів та шаблони електронних листів.




3.12. Структура папки data

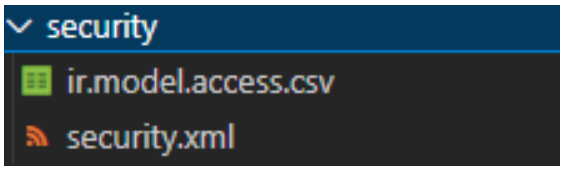
Папка views є відповідальною за зберігання файлів представлень. Представлення визначають спосіб, за допомогою якого моделі/об’єкти відображаються для користувача. Представлення бувають кількох типів,

кожен вид представляє режим візуалізації. Вони роблять модулі зручнішими для користувача та можуть змінюватися відповідно до потреб.




3.13. Структура папки views

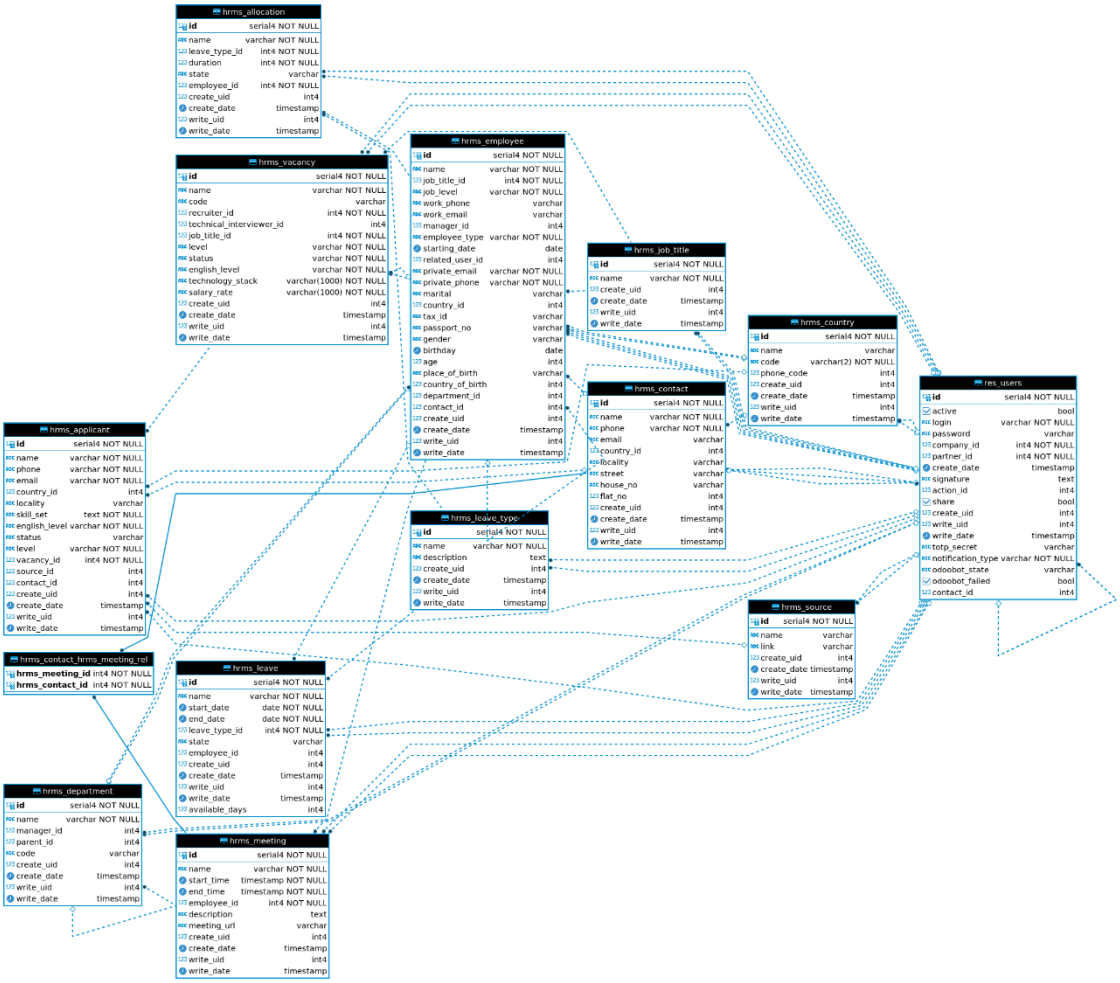
В папочці security зберігаються CSV та XML файли, що є відповідальними за надання доступу тій чи іншій групі користувачів до певних ресурсів. Головним файлом є файл ir.model.access.csv, в якому прописуються, які права доступу до певної моделі буде мати кожна з груп користувачів.




3.14. Структура папки security

### 3.2. Структура та опис бази даних

База даних нашого проекту складається з 14-ти таблиць, зв’язаних між собою.




3.15. Схема бази даних

Таблиця 3.1.

**Сутність «res_users»**

| Назва | Ідентифікатор поля | Тип даних, довжина |
|---|---|---|
| id | id | SERIAL |
| Чи активний запис | active | BOOL |
| Логін | login | VARCHAR |
| Пароль | password | VARCHAR |
| id Компанії | company_id | INT |
| id Партнера | partner_id | INT |
| Дата створення | create_date | TIMESTAMP |
| Підпис | signature | TEXT |
| id Дії | action_id | INT |
| Чи спільний юзер | share | BOOL |

| | | |
|---|---|---|
| Дата останньої модифікації | write_date | TIMESTAMP |
| id Юзера, що створив | create_uid | INT |
| id Юзера, що останню модифікацію здійснив | write_uid | INT |
| id Контакту | contact_id | INT |

Таблиця 3.2.

**Сутність «hrms_employee»**

| Назва | Ідентифікатор поля | Тип даних, довжина |
|---|---|---|
| id | id | SERIAL |
| Ім’я | name | VARCHAR |
| id Посади | job_title_id | INT |
| Кваліфікація | job_level | VARCHAR |
| Робочий номер | work_phone | VARCHAR |
| Робоча пошта | work_email | VARCHAR |
| id Менеджера | manager_id | INT |
| Тип співробітника | employee_type | VARCHAR |
| Дата початку роботи | starting_date | |
| id Пов’язаного користувача | related_user_id | INT |
| Особиста пошта | private_email | VARCHAR |
| Особистий номер | private_phone | VARCHAR |
| Сімейне положення | marital | VARCHAR |
| id Країни | country_id | INT |
| ІПН | tax_id | VARCHAR |
| Номер паспорту | passport_no | VARCHAR |
| Стать | gender | VARCHAR |
| День народження | birthday | DATE |
| Вік | age | INT |
| Місце народження | place_of_birth | VARCHAR |
| id Країни народження | country_of_birth_id | INT |
| id Департаменту | department_id | INT |
| id Контакту | contact_id | INT |
| id Користувача, що створив | create_uid | INT |
| id Користувача, що модифікував останнім | write_uid | INT |
| Дата створення | create_date | TIMESTAMP |

| | | |
|---|---|---|
| Дата останньої модифікації | write_date | TIMESTAMP |

Таблиця 3.3.

**Сутність «hrms_contact»**

| Назва | Ідентифікатор поля | Тип даних, довжина |
|---|---|---|
| id | id | SERIAL |
| Ім’я | name | VARCHAR |
| Номер телефону | phone | VARCHAR |
| Пошта | email | VARCHAR |
| id Країни | country_id | INT |
| Населений пункт | locality | VARCHAR |
| Вулиця | street | VARCHAR |
| Номер дому | house_no | VARCHAR |
| Номер квартири | flat_no | INT |
| id Користувача, що створив | create_uid | INT |
| id Користувача, що модифікував останнім | write_uid | INT |
| Дата створення | create_date | TIMESTAMP |
| Дата останньої модифікації | write_date | TIMESTAMP |

Таблиця 3.4.

**Сутність «hrms_country»**

| Назва | Ідентифікатор поля | Тип даних, довжина |
|---|---|---|
| id | id | SERIAL |
| Назва | name | VARCHAR |
| Код країни | code | VARCHAR |
| Код номеру телефона | phone_code | INT |
| id Користувача, що створив | create_uid | INT |
| id Користувача, що модифікував останнім | write_uid | INT |
| Дата створення | create_date | TIMESTAMP |
| Дата останньої модифікації | write_date | TIMESTAMP |

Таблиця 3.5.

**Сутність «hrms_job_title»**

| Назва | Ідентифікатор поля | Тип даних, довжина |
|---|---|---|
| id | id | SERIAL |
| Назва | name | VARCHAR |
| id Користувача, що створив | create_uid | INT |
| id Користувача, що модифікував останнім | write_uid | INT |
| Дата створення | create_date | TIMESTAMP |
| Дата останньої модифікації | write_date | TIMESTAMP |

Таблиця 3.6.

**Сутність «hrms_source»**

| Назва | Ідентифікатор поля | Тип даних, довжина |
|---|---|---|
| id | id | SERIAL |
| Назва | name | VARCHAR |
| Посилання | link | VARCHAR |
| id Користувача, що створив | create_uid | INT |
| id Користувача, що модифікував останнім | write_uid | INT |
| Дата створення | create_date | TIMESTAMP |
| Дата останньої модифікації | write_date | TIMESTAMP |

Таблиця 3.7.

**Сутність «hrms_department»**

| Назва | Ідентифікатор поля | Тип даних, довжина |
|---|---|---|
| id | id | SERIAL |
| Назва | name | VARCHAR |
| id Менеджера | manager_id | INT |
| id Батьківського департаменту | parent_id | INT |
| Код | code | VARCHAR |
| id Користувача, що створив | create_uid | INT |

| | | |
|---|---|---|
| id Користувача, що модифікував останнім | write_uid | INT |
| Дата створення | create_date | TIMESTAMP |
| Дата останньої модифікації | write_date | TIMESTAMP |

Таблиця 3.8.

**Сутність «hrms_vacancy»**

| Назва | Ідентифікатор поля | Тип даних, довжина |
|---|---|---|
| id | id | SERIAL |
| Назва | name | VARCHAR |
| Код | code | VARCHAR |
| id Рекрутера | recruiter_id | INT |
| id Технічного інтерв'юера | technical_interviewer_id | INT |
| id Посади | job_title_id | VARCHAR |
| Рівень кваліфікації | level | VARCHAR |
| Статус | status | VARCHAR |
| Рівень англійської | english_level | VARCHAR |
| Набір технологій | technology_stack | VARCHAR(1000) |
| Розмір винагороди | salary_rate | VARCHAR(1000) |
| id Користувача, що створив | create_uid | INT |
| id Користувача, що модифікував останнім | write_uid | INT |
| Дата створення | create_date | TIMESTAMP |
| Дата останньої модифікації | write_date | TIMESTAMP |

Таблиця 3.9.

**Сутність «hrms_applicant»**

| Назва | Ідентифікатор поля | Тип даних, довжина |
|---|---|---|
| id | id | SERIAL |
| Ім'я | name | VARCHAR |
| Номер телефону | phone | VARCHAR |
| Пошта | email | VARCHAR |
| id Країни | country_id | INT |
| Населений пункт | locality | VARCHAR |

| | | |
|---|---|---|
| Спектр навичок | skill_set | TEXT |
| Рівень англійської | english_level | VARCHAR |
| Статус | status | VARCHAR |
| Рівень кваліфікації | level | VARCHAR |
| id Вакансії | vacancy_id | INT |
| id Джерела | source_id | INT |
| id Контакту | contact_id | INT |
| id Користувача, що створив | create_uid | INT |
| id Користувача, що модифікував останнім | write_uid | INT |
| Дата створення | create_date | TIMESTAMP |
| Дата останньої модифікації | write_date | TIMESTAMP |

Таблиця 3.10.

**Сутність «hrms_leave»**

| Назва | Ідентифікатор поля | Тип даних, довжина |
|---|---|---|
| id | id | SERIAL |
| Причина | name | VARCHAR |
| Дата початку | start_date | DATE |
| Дата кінця | end_date | DATE |
| id Типу відпустки | leave_type_id | INT |
| Статус | state | VARCHAR |
| id Співробітника | employee_id | INT |
| Кількість доступних днів | available_days | INT |
| id Користувача, що створив | create_uid | INT |
| id Користувача, що модифікував останнім | write_uid | INT |
| Дата створення | create_date | TIMESTAMP |
| Дата останньої модифікації | write_date | TIMESTAMP |

Таблиця 3.11.

**Сутність «hrms_leave_type»**

| Назва | Ідентифікатор поля | Тип даних, довжина |
|---|---|---|
| id | id | SERIAL |
| Назва | name | VARCHAR |
| Опис | description | TEXT |
| id Користувача, що створив | create_uid | INT |
| id Користувача, що модифікував останнім | write_uid | INT |
| Дата створення | create_date | TIMESTAMP |
| Дата останньої модифікації | write_date | TIMESTAMP |

Таблиця 3.12.

**Сутність «hrms_allocation»**

| Назва | Ідентифікатор поля | Тип даних, довжина |
|---|---|---|
| id | id | SERIAL |
| Назва | name | VARCHAR |
| id Типу відпустки | leave_type_id | INT |
| Тривалість | duration | INT |
| Статус | state | VARCHAR |
| id Співробітника | employee_id | INT |
| id Користувача, що створив | create_uid | INT |
| id Користувача, що модифікував останнім | write_uid | INT |
| Дата створення | create_date | TIMESTAMP |
| Дата останньої модифікації | write_date | TIMESTAMP |

Таблиця 3.13.

**Сутність «hrms_meeting»**

| Назва | Ідентифікатор поля | Тип даних, довжина |
|---|---|---|
| id | id | SERIAL |
| Назва | name | VARCHAR |
| Час початку | start_date | TIMESTAMP |
| Час кінця | end_date | TIMESTAMP |
| Опис | description | TEXT |

| | | |
|---|---|---|
| id Співробітника | employee_id | INT |
| Посилання | meeting_url | VARCHAR |
| id Користувача, що створив | create_uid | INT |
| id Користувача, що модифікував останнім | write_uid | INT |
| Дата створення | create_date | TIMESTAMP |
| Дата останньої модифікації | write_date | TIMESTAMP |

Таблиця 3.14.

**Сутність «hrms_contact_hrms_meeting_relation»**

| Назва | Ідентифікатор поля | Тип даних, довжина |
|---|---|---|
| id Події | hrms_meeting_id | SERIAL |
| id Контакту | Hrms_contact_id | SERIAL |

## 3.3. Обґрунтування та організація вхідних та вихідних даних програми

Вхідні дані в системі на базі фреймворку Odoo отримуються шляхом заповнення необхідних полів при створенні або редагуванні запису тієї чи іншої сутності у відповідному представлені, яке описано в XML файлі. Після заповнення даних та натискання на кнопку збереження дані з представлення буде записано в базу даних.

Вихідні дані користувач отримує також всередині даного представлення. Якщо якесь поле залежить від іншого, то при заповненні необхідної інформації значення поля буде розраховано у відповідності до методу, описаного в моделі, та буде виведено на представлення для подальшої роботи з ним.

## 3.4. Опис роботи розробленого програмного продукту

### 3.4.1. Використані технічні засоби

Для розробки кінцевого продукту магістерської роботи був використаний персональний комп’ютер з наступними технічними характеристиками:

- ЦП [CPU]: AMD Ryzen 5 2600 3.4 GHz;
- відеоадаптер [GPU]: nVidia GeForce 1660 ti;
- оперативна пам’ять [RAM] з обсягом: 16 ГБ;
- накопичувач постійної пам’яті з обсягом: 240 ГБ.

Для коректного функціонування даної автоматизованої системи рекомендовано використовувати обчислювальну машину з наступними мінімальними системними параметрами:

- ЦП [CPU]: Intel i3 6XXX / AMD FX 6XXX;
- відеоадаптер [GPU]: nVidia GeForce GT 640 / AMD Radeon HD 7730;
- оперативна пам’ять [RAM] з обсягом: 4 ГБ;
- стабільне Інтернет-з’єднання зі швидкістю 20 мбіт/сек.

### 3.4.2. Використані програмні засоби

В ході виконання кваліфікаційної роботи були використані наступні програмні засоби:

- Visual Studio Code;
- GitHub.
- DBeaver;

Visual Studio Code

Visual Studio Code — це редактор коду, який можна використовувати з різними мовами програмування, включаючи C#, Java, JavaScript, Go, Node.js, Python, C++, C, Rust і Fortran. Він базується на структурі Electron, яка використовується для розробки веб-додатків Node.js, які працюють на механізмі компонування Blink. Visual Studio Code використовує той самий компонент редактора (під кодовою назвою «Monaco»), який використовується

в Azure DevOps (раніше називався Visual Studio Online і Visual Studio Team Services).

З коробки Visual Studio Code містить базову підтримку для більшості поширених мов програмування. Ця базова підтримка включає підсвічування синтаксису, зіставлення дужок, згортання коду та настроювані фрагменти. Visual Studio Code також постачається з IntelliSense для JavaScript, TypeScript, JSON, CSS і HTML, а також підтримує налагодження Node.js. Підтримка додаткових мов може бути забезпечена безкоштовними розширеннями на VS Code Marketplace.

GitHub

GitHub — один з найбільших вебсервісів для спільної розробки програмного забезпечення. Існують безкоштовні та платні тарифні плани користування сайтом. Базується на системі керування версіями Git і розроблений на Ruby on Rails і Erlang компанією GitHub, Inc (раніше Logical Awesome).

Розробники сайту називають GitHub «соціальною мережею для розробників».

Окрім розміщення коду, учасники можуть спілкуватись, коментувати редагування один одного, а також слідкувати за новинами знайомих. За допомогою широких можливостей Git програмісти можуть поєднувати свої репозиторії — GitHub дає зручний інтерфейс для цього і може показувати вклад кожного учасника в вигляді дерева.

Для проєктів є особисті сторінки, невеликі Вікі та система відстеження помилок. Прямо на сайті можна дивитись файли проєктів з підсвічуванням синтаксису для більшості мов програмування.

DBeaver

DBeaver — це SQL клієнт та інструмент управління базами даних. Для реляційних баз даних DBeaver використовує програмний інтерфейс JDBC API, котрий взаємодіє з базами даних через драйвер JDBC. Для інших баз даних (не SQL) він використовує власні драйвери баз даних. DBeaver має функції завершення коду та підсвічування синтаксису, що забезпечує краще сприйняття. Він забезпечує архітектуру плагінів (засновану на архітектурі плагінів (платформі) Eclipse), яка дозволяє змінювати більшу частину роботи програми, для підтримки специфічних для баз даних функцій, незалежних від бази даних.

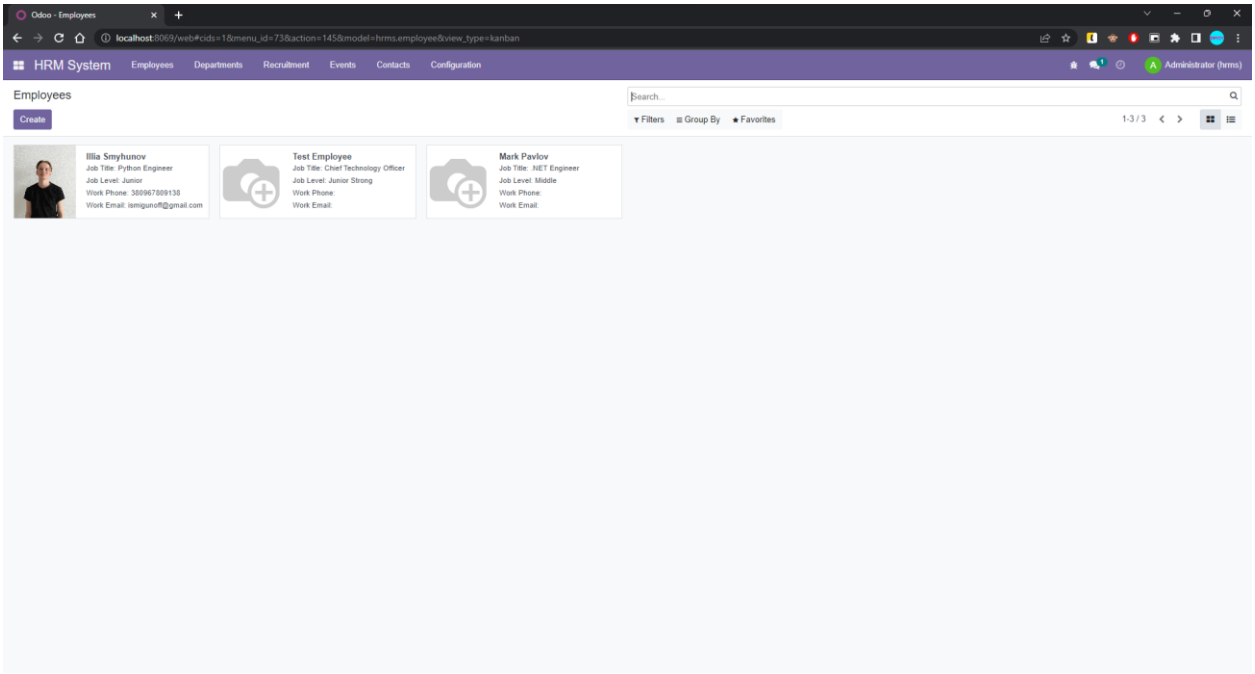
### 3.4.3. Виклик та завантаження програми

Для роботи системи важливо, щоб на комп'ютері був встановлений фреймворк Odoo. Для запуску нашого модулю необхідно завантажити його на обчислювану машину та при запуску серверу Odoo в конфігураційному файлі або в терміналі прописати шлях до нашого модуля під назвою hrm_system. Для того, щоб встановити шлях треба використати параметр addons_path та присвоїти йому значення локації, де лежить на комп'ютері наш модуль. Якщо сервер піднято локально, то треба перейти в браузері за шляхом localhost:8069, створити базу даних, увійти під вказаними даними користувача, перейти в меню Apps та встановити модуль. Після цього в кореневому меню з'явиться підменю HRM System, натиснувши на яке користувач попаде в нашу систему керування персоналом.

### 3.4.4. Опис інтерфейсу користувача

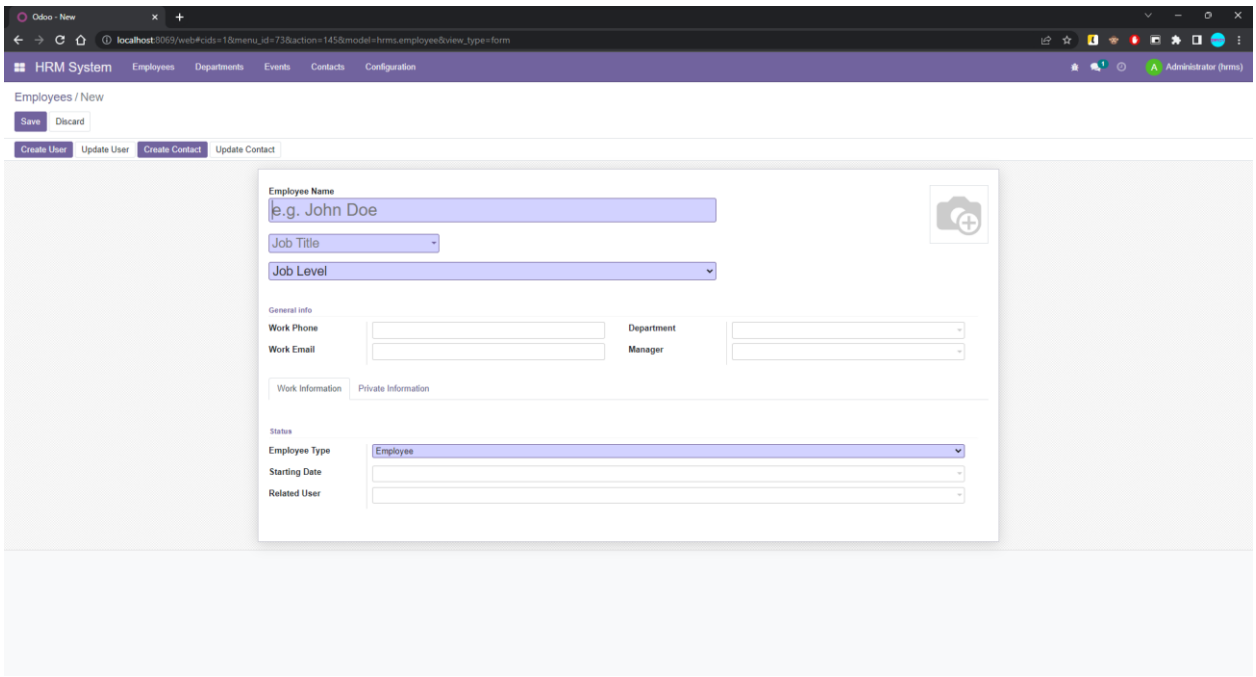
Odoo має дуже зручний та зрозумілий інтерфейс. Після того, як модуль встановлено, щоб потрапити до системи нам достатньо  натиснути на підменю

HRM System. Після цього нас зустріне меню Employees, де ми можемо переглянути список усіх співробітників та створити нових.

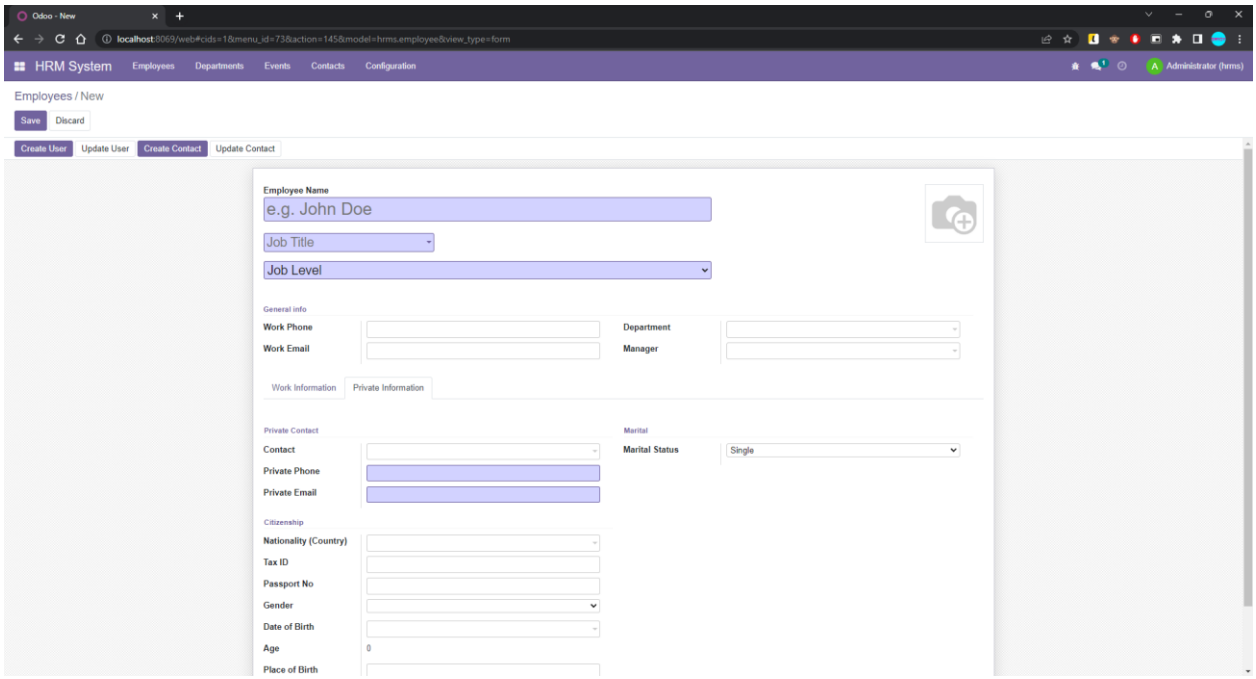



3.16. Меню Employees

Для створення нового співробітника нам достатньо натиснути на кнопку Create, далі заповнити дані нового співробітника. Мінімально необхідно заповнити поля, що підсвічено фіолетовим кольором, це означає, що поле є обов’язковим для заповнення.



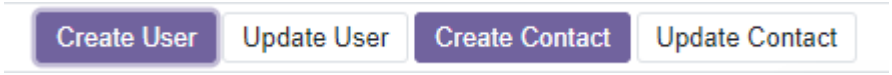

3.17. Створення нового співробітника (1)




3.18. Створення нового співробітника (2)

Вгорі профілю кожного співробітника є спеціальні кнопки, які дозволяють створити нового користувача, оновити існуючого користувача (якщо він є), створити новий контакт або оновити існуючий (якщо такий є).

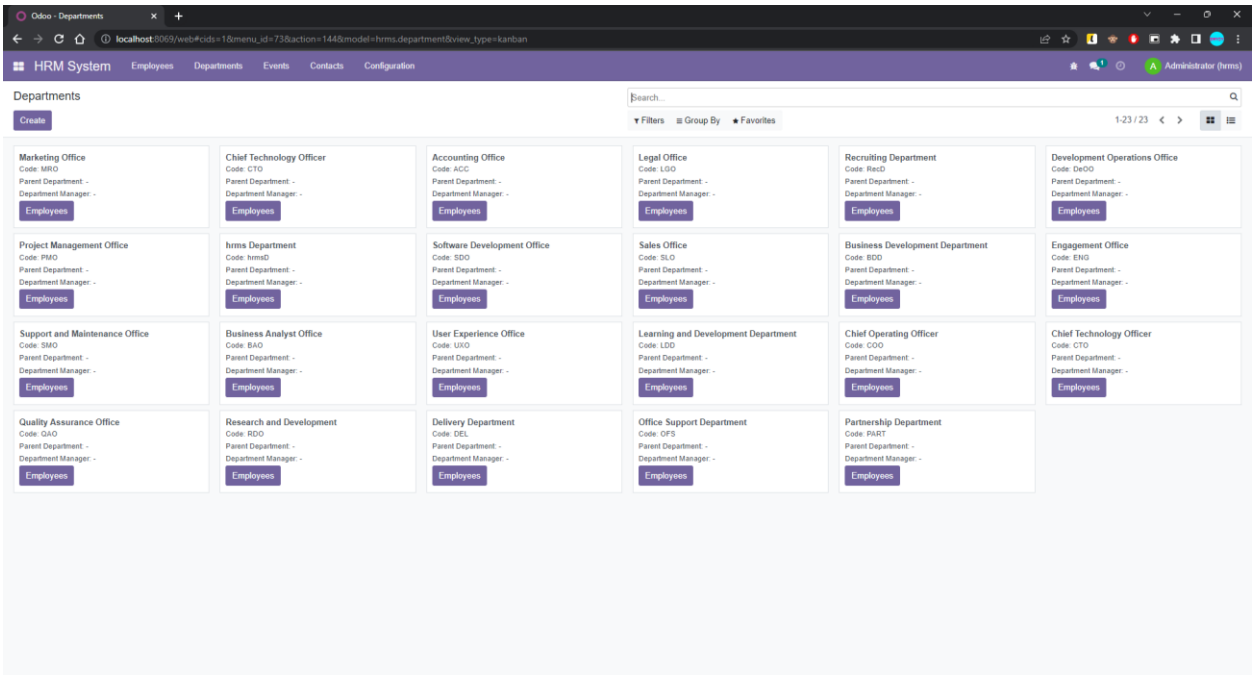
Дані записи будуть створені на основі даних в профілі співробітника, таких як ім’я, пошта та номер телефону.




3.19. Кнопки, за допомогою яких можна створювати записи на основі даних співробітника

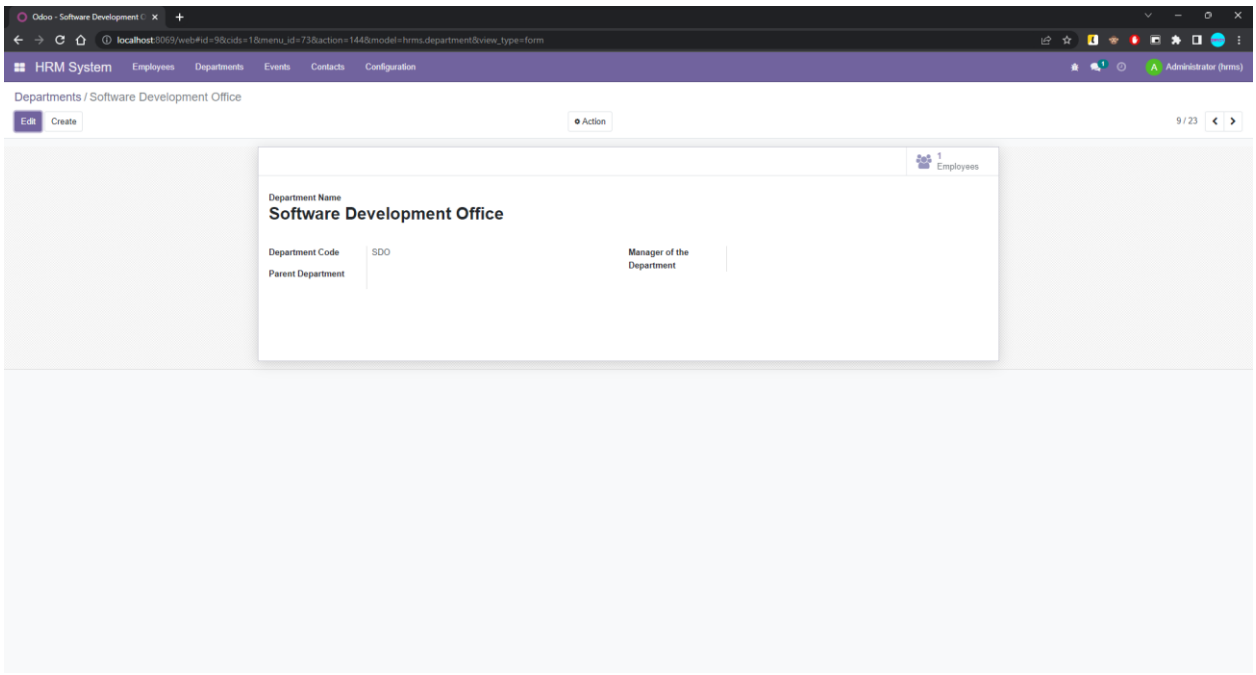
Аналогічний механізм створення, перегляду, редагування та видалення записів є актуальним для кожної з моделей.

Якщо ми перейдемо в меню Departments, то ми побачимо список усіх департаментів.




3.20. Список департаментів

В профілі кожного департаменту є статистична кнопка, яка показує, скільки співробітників налічує даний департамент. При натисканні на дану кнопку ми побачимо список усіх цих людей.

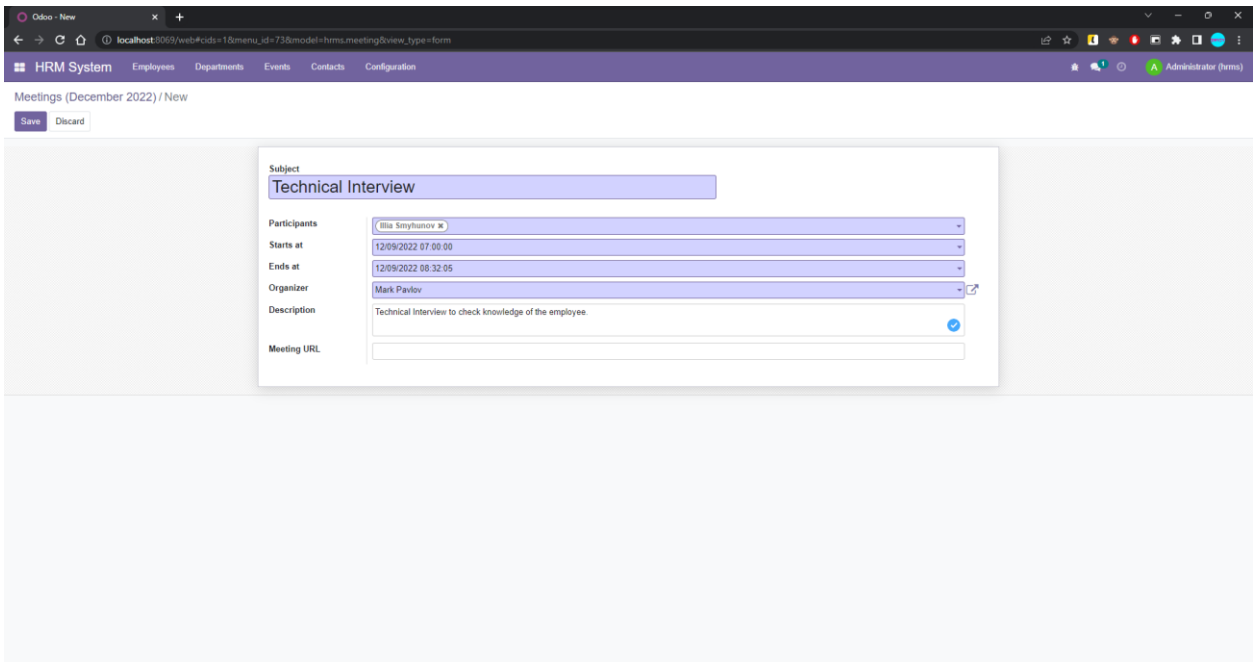



3.21. Приклад департаменту зі статистичною кнопкою

Вкладка Events має 2 підменю, а саме Meetings та Leaves.

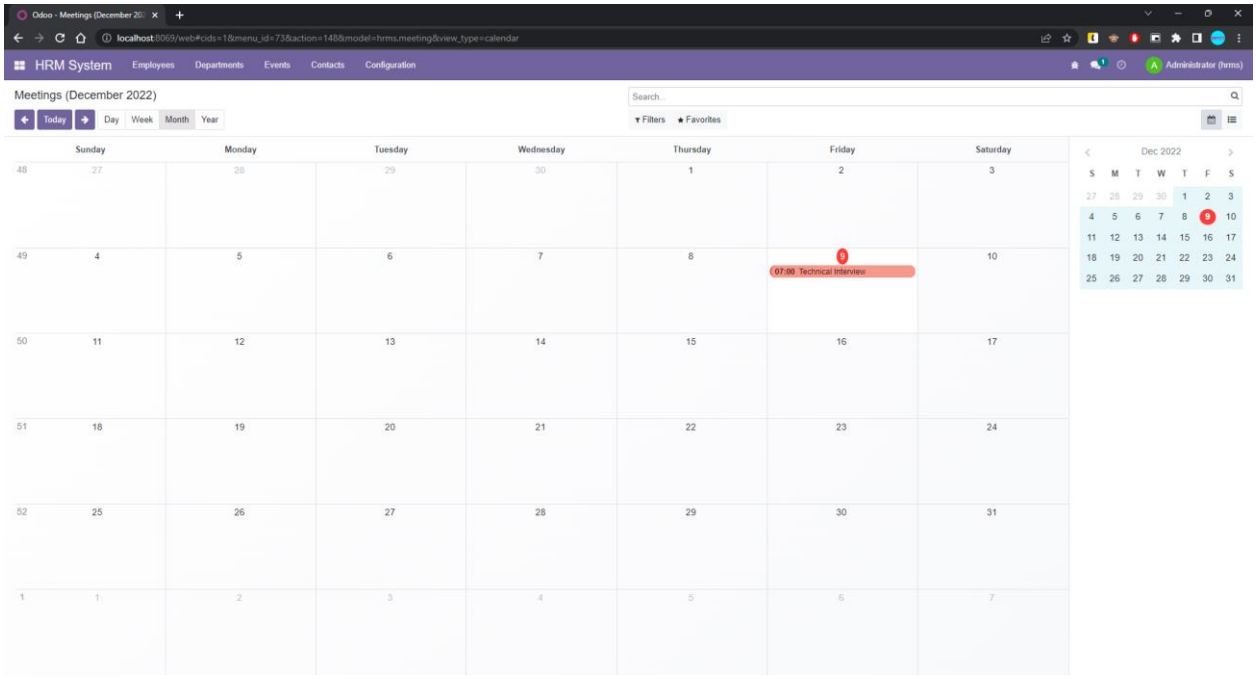
Meetings відповідає за зустрічі, які може назначати менеджер або рекрутер. Це можете бути, наприклад, технічне інтерв'ю або просто банальний дзвінок для перевірки показників співробітника.

Для того, щоб створити нову зустріч достатньо натиснути на потрібну дату в календарі та заповнити дані про зустріч. Створити зустріч може користувач з будь-якою роллю, окрім звичайного співробітника.



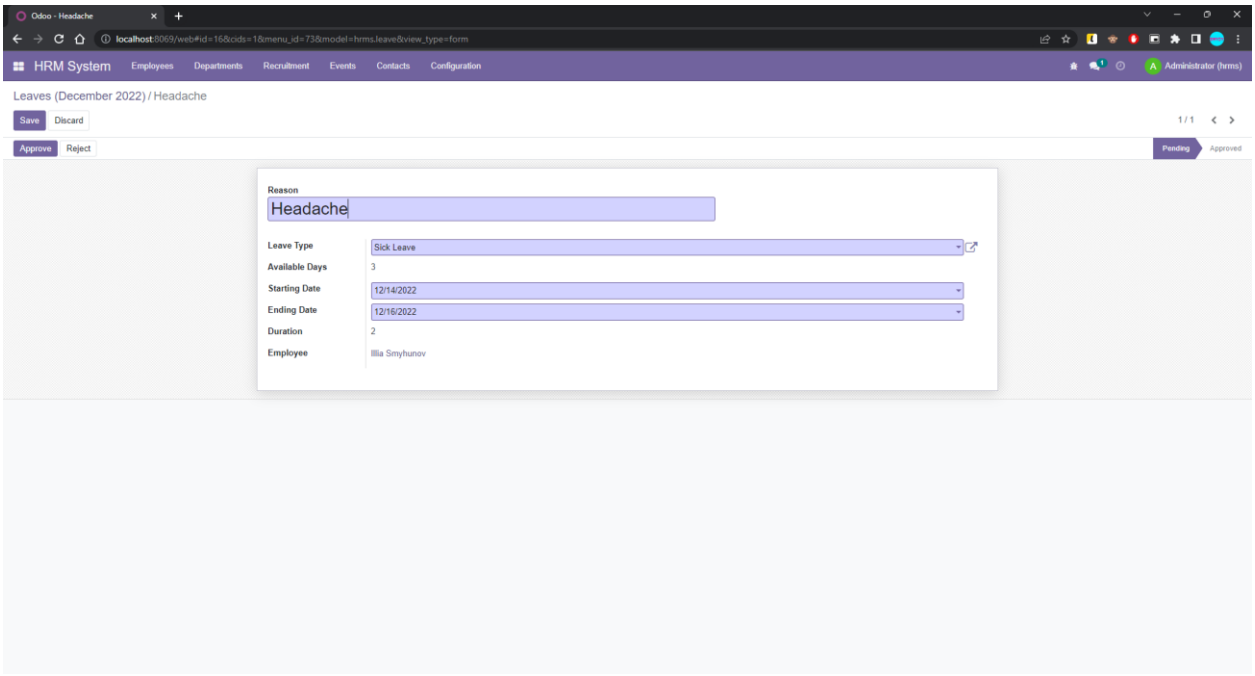

3.22. Створення нової зустрічі

Після того, як зустріч буде створено, вона відобразиться відповідним чином у календарі.



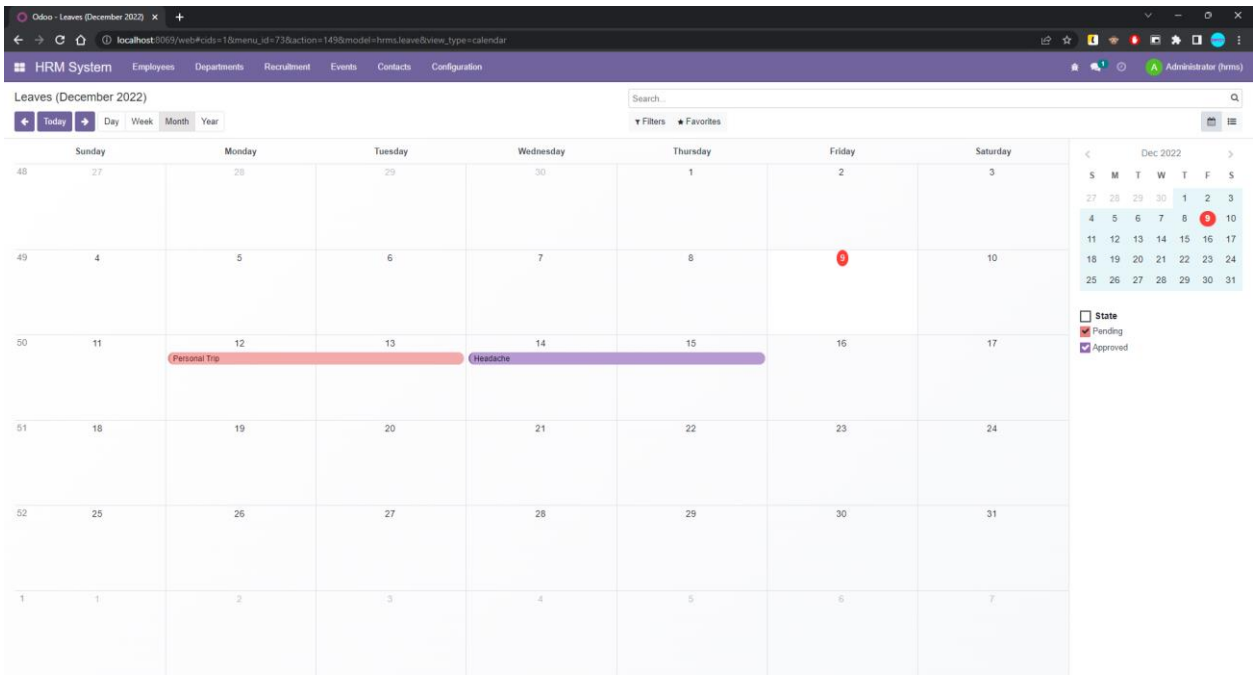

3.23. Відображення створеної зустрічі в календарі

Після переходу в меню Leaves ми також побачимо календарь. Для створення нової відпустки достатньо натиснути на потрібну дату, обрати причину, тип відпустки та дати в рамках доступної кількості.



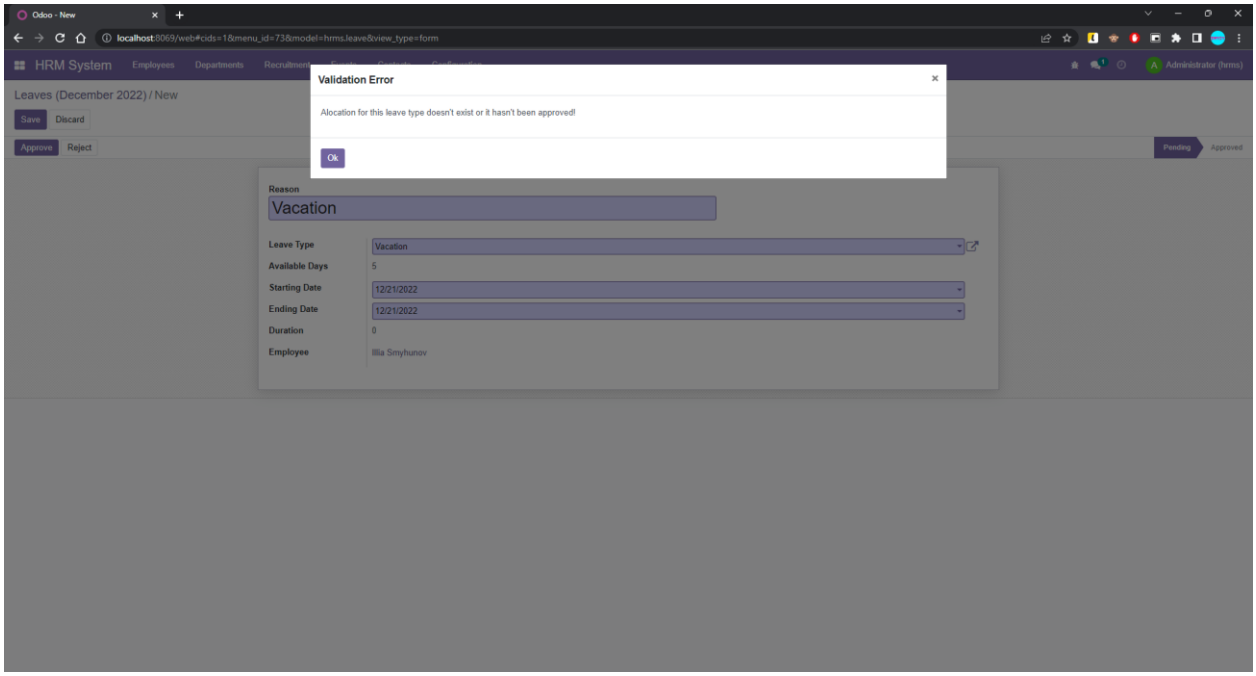

3.24. Заява на відпустку

Після створення нової відпустки вона вопинна бути затверджена менеджером. Для того, щоб затвердити відпусту достатньо обрати її та натиснути кнопку Approve. Затвердженні відпустки в календарі помічені фіолетовим кольором, тоді як незавтверджені – червоним.




3.25. Приклад календаря зі створеними відпустками

Для того, щоб в співробітника була можливість взяти відпустку, менеджер повинен створити для нього розподілення днів (allocation) та затвердити його. В інакшому випадку система видась повідомлення, що в співробітника немає затверджених розподілень на дану кількість днів.



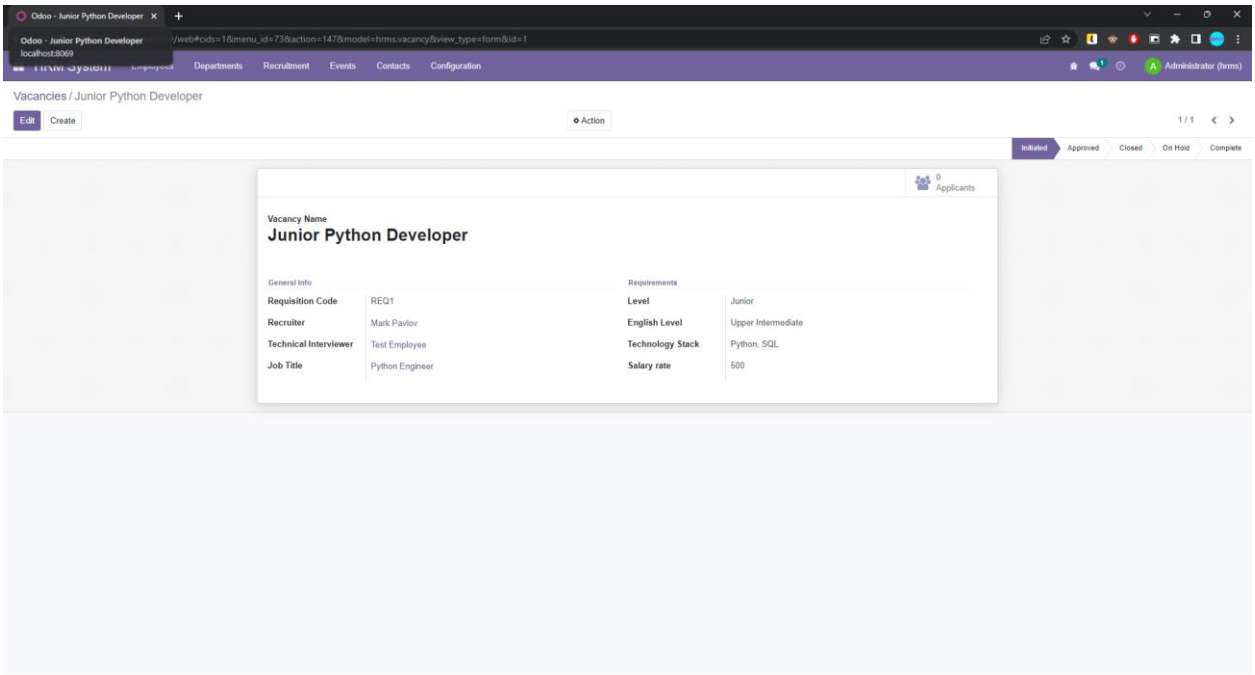

3.26. Повідомлення про неможливість взяти відпустку

Створити новий розподіл днів можна через меню Configuration – Allocation. Це меню бачать лише менеджер, голова HR та адміністратор.

Затвердити відпустку може лише менеджер даного співробітника. При спробі звичайному користувачеві затвердити відпустку, він отримає помилку.

Також наша система має меню Recruitment, яке призначено для зручного відбору кандидатів на нові вакансії. Дане меню бачать лише рекрутер та голова HR.

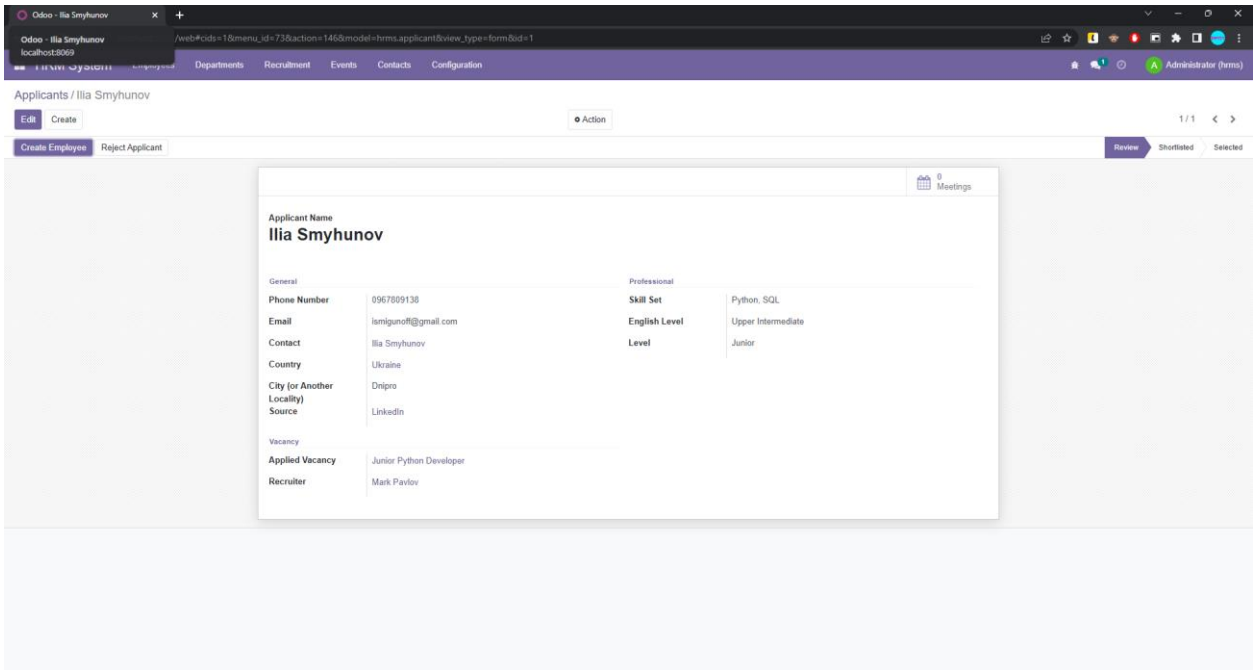
Для створення нової вакансії достатньо перейти в підменю Vacancies та натиснути Create. Після заповнення всіх необхідних даних буде створено нову вакансію, яка також буде містити статистичну кнопку, що відповідає за відображеня усіх кандидатів, яких розглядають на цю посаду. Також, в правому верхньому кутку є статусбар, за допомогою якого можна керувати статусом вакансії.




3.27. Приклад створеної вакансії

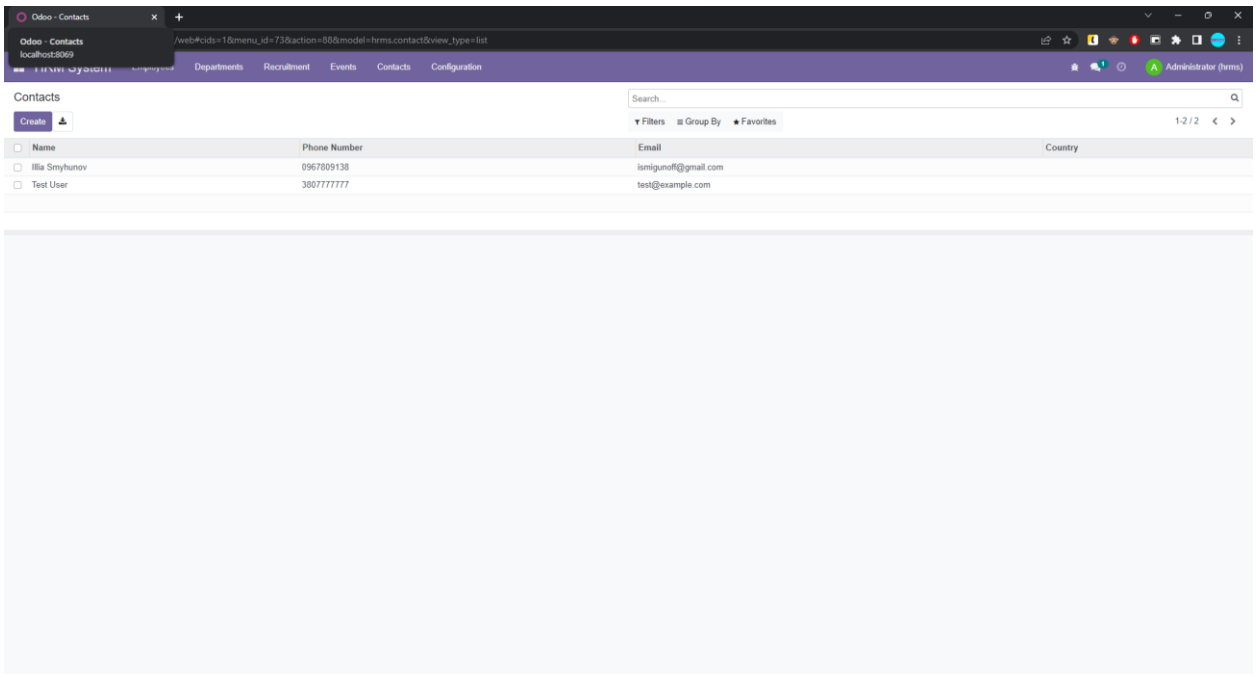
Для того, щоб створити нового кандидата на вакансію потрібно перейти впідменю Applicants та натиснути кнопку Create. В профілі кандидата, окрім його даних будуть такі елементи:

– статусбар, за допомогою якого можна керувати станом кандидата;
– статистична кнопка, що відображає, які зустрічі призначені даному кандидату;
– кнопка Create Employee, яка довзоляє після успішного відбору одразу створити співробітника на основі даних кандидата;
– кнопка Reject, що дозволяє відхили кандидатуру.




3.28. Приклад створеного кандидата

Всі створені контакти зберігаються в меню Contacts




3.29. Приклад списку контактів

## 3.5. Висновки

У третьому розділі було розглянуто структуру проекту магістерської роботи та пояснено, для чого використовується та чи інша директорія. Також було проілюстровано структуру нашої бази даних за допомогою ER-діаграми та описано кожну сутність окремо. Наша база даних складається з 14-ти таблиць.

Окрім того, було проведено опис роботи розробленого програмного продукту, зокрема, використані технічні та програмні засоби, виклик та завантаження системи, а також інтерфейс користувача.

# ВИСНОВКИ

Метою магістерської роботи є розробка та дослідження ефективності впровадження програмного забезпечення для управління персоналом

Система являє собою модуль, розроблений на базі фреймворку Odoo, дозволяє керувати персоналом в компанії, а також відбирати нових кандидатів на вканасії.

Система надає можливість виконувати наступні дії:

– можливість гнучкого відбору кандидатів та створенням нового співробітника на основі кандидата, що успішно пройшов відбір;

– створення нових співробітників та їх зручне керування, наприклад, зміна департаменту, надання певних прав тощо;

– дозволяє створити базу контактів, за допомогою якої можна буде легко зв’язатися з людиною;

– дає можливість влаштовувати співбесіди та інші події за допомогою зручного блоку з календарем;

– можливість взяти відгул/відпустку потрібного типу або створити новий тип.

Актуальність поставленої задачі обумовлюється широким попитом на такі продукти, бо кожен бізнес в нас час прагне максимально автоматизувати всі внутрішні процеси та створити комфортні умови для виконання повсякденних задач. Основний ресурс будь-якої компанії – її працівники, тому необхідно забезпечити зручний механізм для відбору нових кадрів та для керування процесами, пов’язаними напряму із кожним співробітником. Саме за це й є відповідальною наша система. Також важливо використовувати новітні технології, адже це сприяє створенню дійсно гнучкого функціоналу.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Johnason, P. (2009). HRM in changing organizational contexts. In D. G.Collings & G. Wood (Eds.), Human resource management: A critical approach (pp. 19-37). London: Routledge.

2. Paauwe, J., & Boon, C. (2009). Strategic HRM: A critical review. In D. G. Collings, G. Wood (Eds.) & M.A. Reid, Human resource management: A critical approach (pp. 38-54). London: Routledge.

3. Armstrong, Michael (2009). Armstrong's handbook of human resource management practice. Armstrong, Michael, 1928- (Eleventh ed.). London: Kogan Page. ISBN 9780749457389. OCLC 435643771.

4. Marc Holliday. What is a HRMS? [Електронний ресурс]: Oracle Netsuite. – 2020.

5. Алан Шаллоуей, Джеймс Р. Тротт. Шаблоны проектирования. Новый подход к объектно-ориентированному анализу и проектированию = Design Patterns Explained: A New Perspective on Object-Oriented Design. – М.: «Вильямс». – 2002. – 288 с.

6. Романенков Ю., Зєйнієв Т. Завдання контуру стратегічного управління ефективністю бізнес-процесів в організації. Системні дослідження та інформаційні технології №3, м. Київ, Україна, 2015, С.43-47

7. Clean Architecture: A Craftsman's Guide to Software Structure and Design (Robert C. Martin Series) 1st Edition. – 2018. - 450 p.

8. Fowler, Martin "Patterns of Enterprise Application Architecture" (2002). Addison Wesley.

9. Dave Kuhlman. A Python Book: Beginning Python, Advanced Python, and Python Exercises. – 2012.

10. "About Python". Python Software Foundation. Archived from the original on 20 April 2012. Retrieved 24 April 2012., second section "Fans of Python use the

phrase "batteries included" to describe the standard library, which covers everything from asynchronous processing to zip files."

11. Guido van Rossum, Python Reference Manual, release 2.4.4, 18 October 2006.

12. The Making of Python. Архів оригіналу за 1 вересня 2016. Процитовано 4 липня 2010.

13. Holger Brunn, Alexandre Fayolle, Daniel Reis. Odoo Development Cookbook. – 2016. – 400 p.

14. Stonebraker, M.; Rowe, L. A. (May 1986). The design of POSTGRES (PDF). Proc. 1986 ACM SIGMOD Conference on Management of Data. Washington, DC. Retrieved December 17, 2011.

15. "Project name – statement from the core team". archives.postgresql.org. November 16, 2007. Retrieved November 16, 2007.

16. "XML Media Types, RFC 7303". Internet Engineering Task Force. July 2014.

17. Fennell, Philip (June 2013). "Extremes of XML". XML London 2013: 80–86. doi:10.14337/XMLLondon13.Fennell01. ISBN 978-0-9926471-0-0.

**ДОДАТОК А**

# КОД ПРОГРАМИ

## __manifest__.py

```
{
   "name": "HRM System",
   "version": "7.0",
   "description": "HRMS that allows easy HR process management",
   "author": "Illia Smyhunov",
   "depends": [
      "base", "mail"
   ],
   "data": [
      "data/hrms_country_data.xml",
      "data/hrms_department_data.xml",
      "data/hrms_job_title_data.xml",
      "data/mail_template_data.xml",
      "data/hrms_source_data.xml",
      "data/hrms_leave_type_data.xml",

      "security/ir.model.access.csv",
      "security/security.xml",

      "views/hrms_contact_views.xml",
      "views/hrms_department_views.xml",
      "views/hrms_employee_views.xml",
      "views/hrms_applicant_views.xml",
      "views/hrms_vacancy_views.xml",
      "views/hrms_meeting_views.xml",
      "views/hrms_leave_views.xml",
      "views/hrms_leave_type_views.xml",
      "views/hrms_allocation_views.xml",
      "views/hrms_job_title_views.xml",
      "views/res_users_views.xml",
      "views/menu.xml",
   ],
   "application": True,
}
```

## hrms_allocation.py

```
from odoo import fields, models, api, exceptions



class Allocation(models.Model):
   _name = "hrms.allocation"
```

```
name = fields.Char(
    string="Name",
    required=True,
)
leave_type_id = fields.Many2one(
    comodel_name="hrms.leave.type",
    string="Leave Type",
    required=True,
)
duration = fields.Integer(
    string="Number of Days",
    required=True,
)
state = fields.Selection(
    selection=[
        ("pending", "Pending"),
        ("approved", "Approved"),
        ("rejected", "Rejected")
    ],
    string="State",
    default="pending",
)
employee_id = fields.Many2one(
    comodel_name="hrms.employee",
    string="Employee",
    default=lambda self: self.env["hrms.employee"].search(
        [("related_user_id", "=", self.env.uid)]),
    required=True,
)

def action_approve_allocation(self):
    for rec in self:
```

```
            rec.state = "approved"

    def action_reject_allocation(self):
        for rec in self:
            rec.state = "rejected"

    @api.model_create_multi
    def create(self, vals_list):
        for val in vals_list:
            leave_type_id = val.get("leave_type_id")
            employee_id = val.get("employee_id")
            allocation = self.env["hrms.allocation"].search([("leave_type_id", "=", leave_type_id), (
                "employee_id", "=", employee_id)])
            if allocation:
                raise exceptions.ValidationError("Allocation with this type for a current user already
exists!")
        return super().create(vals_list)
```

## hrms_applicant.py

```
from odoo import fields, models, api, exceptions



class Applicant(models.Model):
    _name = "hrms.applicant"

    name = fields.Char(
        string="Name",
        required=True,
    )
    phone = fields.Char(
        string="Phone Number",
        required=True,
    )
```

```
email = fields.Char(
    string="Email",
    required=True,
)
country_id = fields.Many2one(
    comodel_name="hrms.country",
    string="Country",
)
locality = fields.Char(
    string="City (or Another Locality)",
)
skill_set = fields.Text(
    string="Skill Set",
    required=True,
)
english_level = fields.Selection(
    selection=[
        ("beginner", "Beginner"),
        ("elementary", "Elementary"),
        ("intermediate", "Intermediate"),
        ("upper_intermediate", "Upper Intermediate"),
        ("advanced", "Advanced"),
        ("proficient", "Proficient"),
    ],
    string="English Level",
    required=True,
)
status = fields.Selection(
    selection=[
        ("review", "Review"),
        ("shortlisted", "Shortlisted"),
        ("selected", "Selected"),
        ("rejected", "Rejected"),
```

```
    ],
    default="review",
    string="Status",
)
level = fields.Selection(
    selection=[
        ("trainee", "Trainee"),
        ("junior", "Junior"),
        ("middle", "Middle"),
        ("senior", "Senior"),
        ("team_lead", "Team Lead")
    ],
    string="Level",
    required=True,
)
vacancy_id = fields.Many2one(
    comodel_name="hrms.vacancy",
    string="Applied Vacancy",
    required=True,
)
recruiter_id = fields.Many2one(
    comodel_name="hrms.employee",
    compute="_compute_recruiter_id",
    string="Recruiter",
)
source_id = fields.Many2one(
    comodel_name="hrms.source",
    string="Source",
)
contact_id = fields.Many2one(
    comodel_name="hrms.contact",
    string="Contact",
)
```

```
    meeting_count = fields.Integer(
        string="Meetings",
        compute="_compute_meeting_count",
    )

    def _compute_meeting_count(self):
        for rec in self:
            rec.meeting_count = self.env["hrms.meeting"].search_count([("contact_ids", "=",
rec.contact_id.id)])

    @api.depends("vacancy_id")
    def _compute_recruiter_id(self):
        for rec in self:
            rec.recruiter_id = rec.vacancy_id.recruiter_id if rec.vacancy_id else None

    @api.model_create_multi
    def create(self, vals_list):
        res = super().create(vals_list)
        for r in res:
            contact_exists = self.env["hrms.contact"].search(
                [("name", "=", r.name)])
            if not contact_exists:
                contact = self.env["hrms.contact"].create({
                    "name": r.name,
                    "phone": r.phone,
                    "email": r.email,
                })
                r.contact_id = contact
            else:
                r.contact_id = contact_exists
        return res

    def action_show_meetings(self):
```

```
        for rec in self:
            action = self.env["ir.actions.act_window"]._for_xml_id(
                "hrm_system.hrms_meeting_action")
            action.update({
                "domain": [("contact_ids", "=", rec.contact_id.id)]
            })
            return action

    def action_create_employee(self):
        for rec in self:
            self.env["hrms.employee"].create({
                "name": rec.name,
                "job_title_id": rec.vacancy_id.job_title_id.id,
                "job_level": rec.level,
                "contact_id": rec.contact_id.id,
                "private_email": rec.email,
                "private_phone": rec.phone,
            })

    def action_reject(self):
        for rec in self:
            rec.status = "rejected"
```

## hrms_contact.py

```
from odoo import fields, models

class Contact(models.Model):
    _name = "hrms.contact"

    name = fields.Char(
        string="Name",
        required=True,
    )
    phone = fields.Char(
        string="Phone Number",
        required=True,
    )
    email = fields.Char(
```

```
        string="Email",
    )
    country_id = fields.Many2one(
        comodel_name="hrms.country",
        string="Country",
    )
    locality = fields.Char(
        string="City (or Another Locality)",
    )
    street = fields.Char(
        string="Street",
    )
    house_no = fields.Char(
        string="House №",
        help="House №. Can contain numbers and letters",
    )
    flat_no = fields.Integer(
        "Flat №",
    )
    image = fields.Binary(
        string="Avatar",
    )
```

## hrms_country.py

```
from odoo import fields, models


class Country(models.Model):
    _name = "hrms.country"

    name = fields.Char(
        string="Name"
    )
    code = fields.Char(
        string="Name",
        size=2,
        required=True,
    )
    phone_code = fields.Integer(
        string="Calling Code",
    )
```

## hrms_departments.py

```
from odoo import fields, models


class Deparment(models.Model):
    _name = "hrms.department"
```

```
    name = fields.Char(
        string="Department Name",
        required=True,
    )
    manager_id = fields.Many2one(
        comodel_name="hrms.employee",
        string="Manager of the Department",
    )
    parent_id = fields.Many2one(
        comodel_name="hrms.department",
        string="Parent Department",
    )
    code = fields.Char(
        string="Department Code",
    )
    employee_count = fields.Integer(
        string="Employees",
        compute="_compute_employee_count",
    )

    def _compute_employee_count(self):
        for rec in self:
            rec.employee_count = self.env["hrms.employee"].search_count([("department_id", "=",
rec.id)])

    def action_show_employees(self):
        for rec in self:
            action = self.env["ir.actions.act_window"]._for_xml_id(
                "hrm_system.hrms_employee_action")
            action.update(
                {
                    "domain": [("department_id", "=", rec.id)],
                }
            )
            return action
```

## hrms_employee.py

```
from datetime import date

from odoo import fields, models, api, exceptions


class Employee(models.Model):
    _name = "hrms.employee"

    name = fields.Char(
        string="Name",
        required=True,
    )
```

```
job_title_id = fields.Many2one(
    comodel_name="hrms.job.title",
    required=True,
)
job_level = fields.Selection(
    selection=[
        ("trainee", "Trainee"),
        ("junior_low", "Junior Low"),
        ("junior", "Junior"),
        ("junior_strong", "Junior Strong"),
        ("middle_low", "Middle Low"),
        ("middle", "Middle"),
        ("middle_strong", "Middle Strong"),
        ("senior", "Senior"),
        ("team_lead", "Team Lead"),
        ("tech_lead", "Tech Lead"),
        ("head", "Head"),
        ("director", "Director"),
        ("architect", "Architect"),
    ],
    string="Job Level",
    required=True,
)
work_phone = fields.Char(
    string="Work Phone",
)
work_email = fields.Char(
    string="Work Email",
)
manager_id = fields.Many2one(
    comodel_name="hrms.employee",
)
employee_type = fields.Selection([
    ("employee", "Employee"),
    ("student", "Student"),
    ("trainee", "Trainee"),
    ("contractor", "Contractor"),
    ("freelance", "Freelancer"),
],
    string="Employee Type",
    default="employee",
    required=True,
)
starting_date = fields.Date(
    string="Starting Date",
)
related_user_id = fields.Many2one(
    comodel_name="res.users",
    string="Related User",
```

```
)
private_email = fields.Char(
    string="Private Email",
    required=True,
)
private_phone = fields.Char(
    string="Private Phone",
    required=True,
)
marital = fields.Selection([
    ("single", "Single"),
    ("married", "Married"),
    ("cohabitant", "Legal Cohabitant"),
    ("widower", "Widower"),
    ("divorced", "Divorced")
],
    string="Marital Status",
    default="single",
)
country_id = fields.Many2one(
    comodel_name="hrms.country",
    string="Nationality (Country)"
)
tax_id = fields.Char(
    string="Tax ID",
)
passport_no = fields.Char(
    string="Passport No",
)
gender = fields.Selection([
    ("male", "Male"),
    ("female", "Female"),
    ("other", "Other")
],
)
birthday = fields.Date(
    string="Date of Birth",
)
age = fields.Integer(
    string="Age",
    compute="_compute_age",
    store=True,
)
place_of_birth = fields.Char(
    string="Place of Birth",
)
country_of_birth = fields.Many2one(
    comodel_name="hrms.country",
    string="Country of Birth",
```

```
    )
    image = fields.Binary(
        string="Avatar",
    )
    department_id = fields.Many2one(
        comodel_name="hrms.department",
        string="Department",
    )
    contact_id = fields.Many2one(
        comodel_name="hrms.contact",
        string="Contact",
    )

    @api.depends("birthday")
    def _compute_age(self):
        today = date.today()
        for rec in self:
            if rec.birthday:
                rec.age = today.year - rec.birthday.year - \
                    ((today.month, today.day) <
                     (rec.birthday.month, rec.birthday.day))
            else:
                rec.age = 0

    def action_create_contact(self):
        for rec in self:
            contact_exists = self.env["hrms.contact"].search(
                [("name", "=", rec.name)])
            if not contact_exists:
                contact = self.env["hrms.contact"].create({
                    "name": rec.name,
                    "phone": rec.private_phone,
                    "email": rec.private_email,
                })
                rec.contact_id = contact
            else:
                raise exceptions.ValidationError("Contact already exists!")

    def action_create_user(self):
        for rec in self:
            welcome_template = self.env.ref(
                "hrm_system.email_template_employee_created")
            user_exists = self.env["res.users"].search(
                [("name", "=", rec.name)])
            if not user_exists:
                if not (rec.work_phone and rec.work_email):
                        raise exceptions.ValidationError("Both Work Email and Work Phone shoud be
fiiled!")
```

```
            user = self.env["res.users"].create({
                "name": rec.name,
                "phone": rec.work_phone,
                "login": rec.work_email,
                "contact_id": rec.contact_id.id,
            })
            rec.related_user_id = user
            welcome_template.send_mail(rec.id, force_send=True)
        else:
            raise exceptions.ValidationError("User already exists!")

    def action_update_contact(self):
        for rec in self:
            rec.contact_id.write({
                "name": rec.name,
                "phone": rec.private_phone,
                "email": rec.private_email,
            })

    def action_update_user(self):
        for rec in self:
            rec.related_user_id.write({
                "name": rec.name,
                "phone": rec.work_phone,
                "email": rec.work_email,
            })

    @api.onchange("contact_id")
    def _onchange_contact_id(self):
        self.private_email = self.contact_id.email
        self.private_phone = self.contact_id.phone

    def write(self, vals):
        if                                    self.user_has_groups("hrm_system.group_hrms_user,
hrm_system.group_hrms_recruiter") and self.related_user_id.id != self.env.uid:
            raise exceptions.AccessDenied("You have no rights to edit other employees!")
        return super(Employee, self).write(vals)
```

## hrms_job_title.py

```
from odoo import fields, models


class JobTitle(models.Model):
    _name = "hrms.job.title"
    _description = "Job Title"

    name = fields.Char(
        string="Name",
        required=True,
```

```
    )
```

## hrms_leave_type.py

```
from odoo import fields, models


class LeaveType(models.Model):
    _name = "hrms.leave.type"

    name = fields.Char(
        string="Name",
        required=True,
    )
    description = fields.Text(
        string="Description",
    )
```

## hrms_leave.py

```
from odoo import fields, models, api, exceptions


class Leave(models.Model):
    _name = "hrms.leave"

    name = fields.Char(
        string="Reason",
        required=True,
    )
    start_date = fields.Date(
        string="Starting Date",
        required=True,
    )
    end_date = fields.Date(
        string="Ending Date",
        required=True,
    )
    leave_type_id = fields.Many2one(
        comodel_name="hrms.leave.type",
        string="Leave Type",
        required=True,
    )
    duration = fields.Integer(
        compute="_compute_duration",
        string="Duration",
        readonly=True,
    )
    state = fields.Selection(
        selection=[
            ("pending", "Pending"),
```

```
            ("approved", "Approved"),
            ("rejected", "Rejected")
        ],
        string="State",
        default="pending",
    )
    employee_id = fields.Many2one(
        comodel_name="hrms.employee",
        string="Employee",
        default=lambda self: self.env["hrms.employee"].search(
            [("related_user_id", "=", self.env.uid)]),
        readonly=True,
    )
    available_days = fields.Integer(
        compute="_compute_available_days",
        string="Available Days",
    )

    @api.depends("start_date", "end_date")
    def _compute_duration(self):
        for rec in self:
            if rec.start_date and rec.end_date:
                rec.duration = (rec.end_date - rec.start_date).days
            else:
                rec.duration = 0

    def action_approve_leave(self):
        for rec in self:
            rec.state = "approved"

    def action_reject_leave(self):
        for rec in self:
            rec.state = "rejected"

    @api.depends("leave_type_id")
    def _compute_available_days(self):
        for rec in self:
            allocation = self.env["hrms.allocation"].search([("leave_type_id", "=",
rec.leave_type_id.id), (
                "employee_id", "=", rec.employee_id.id)])
            if allocation:
                rec.available_days += allocation.duration
            else:
                rec.available_days = 0

    @api.model_create_multi
    def create(self, vals_list):
        res = super().create(vals_list)
        for r in res:
```

```
            allocation = self.env["hrms.allocation"].sudo().search([("leave_type_id", "=",
r.leave_type_id.id), (
                "employee_id", "=", r.employee_id.id)])
            if allocation and allocation.state == "approved":
                allocation.duration -= r.duration
            else:
                raise exceptions.ValidationError("Alocation for this leave type doesn't exist or it hasn't
been approved!")
        return res

    @api.constrains("start_date", "end_date", "duration", "available_days")
    def _check_vals(self):
        for rec in self:
            if rec.start_date > rec.end_date:
                raise exceptions.ValidationError("Start Date cannot be greater than End Date!")
            elif (rec.available_days - rec.duration) < 0:
                raise exceptions.ValidationError("You don't have enough available days!")
```

## hrms_meeting.py

```
from odoo import fields, models


class Meeting(models.Model):
    _name = "hrms.meeting"

    name = fields.Char(
        string="Subject",
        required=True,
    )
    contact_ids = fields.Many2many(
        comodel_name="hrms.contact",
        string="Participants",
        required=True,
    )
    start_time = fields.Datetime(
        string="Starts at",
        required=True,
    )
    end_time = fields.Datetime(
        string="Ends at",
        required=True,
    )
    employee_id = fields.Many2one(
        comodel_name="hrms.employee",
        string="Organizer",
        required=True,
    )
    description = fields.Text(
        string="Description",
```

```
    )
    meeting_url = fields.Char(
        string="Meeting URL",
    )
```

## hrms_source.py

```
from odoo import fields, models

class Source(models.Model):
    _name = "hrms.source"

    name = fields.Char(
        string="Name",
    )
    link = fields.Char(
        string="Link"
    )
```

## hrms_vacancy.py

```
from odoo import fields, models, api


class Vacancy(models.Model):
    _name = "hrms.vacancy"

    name = fields.Char(
        string="Name of Vacancy",
        required=True,
    )
    code = fields.Char(
        string="Requisition Code",
        readonly=True,
    )
    recruiter_id = fields.Many2one(
        comodel_name="hrms.employee",
        string="Recruiter",
        required=True,
    )
    technical_interviewer_id = fields.Many2one(
        comodel_name="hrms.employee",
        string="Technical Interviewer",
    )
    job_title_id = fields.Many2one(
        comodel_name="hrms.job.title",
        string="Job Title",
        required=True,
    )
    level = fields.Selection(
        selection=[
            ("trainee", "Trainee"),
```

```
        ("junior", "Junior"),
        ("middle", "Middle"),
        ("senior", "Senior"),
        ("team_lead", "Team Lead")
    ],
    string="Level",
    required=True,
)
status = fields.Selection(
    selection=[
        ("initiated", "Initiated"),
        ("approved", "Approved"),
        ("closed", "Closed"),
        ("on_hold", "On Hold"),
        ("complete", "Complete"),
    ],
    string="Status",
    required=True,
    default="initiated",
)
english_level = fields.Selection(
    selection=[
        ("beginner", "Beginner"),
        ("elementary", "Elementary"),
        ("intermediate", "Intermediate"),
        ("upper_intermediate", "Upper Intermediate"),
        ("advanced", "Advanced"),
        ("proficient", "Proficient"),
    ],
    string="English Level",
    required=True,
)
technology_stack = fields.Char(
    string="Technology Stack",
    size=1000,
    required=True,
)
salary_rate = fields.Char(
    string="Salary rate",
    size=1000,
    required=True,
)
applicant_count = fields.Integer(
    string="Applicants",
    compute="_compute_applicant_count",
)

@api.model_create_multi
def create(self, vals_list):
```

```
        res = super().create(vals_list)
        for r in res:
            r.code = f"REQ{r.id}"
        return res

    def _compute_applicant_count(self):
        for rec in self:
            rec.applicant_count = self.env["hrms.applicant"].search_count([("vacancy_id", "=",
rec.id)])

    def action_show_applicants(self):
        for rec in self:
            action = self.env["ir.actions.act_window"]._for_xml_id(
                "hrms_recruitment.hrms_applicant_action")
            action.update(
                {
                    "domain": [("vacancy_id", "=", rec.id)],
                }
            )
            return action
```

## ir.model.access.csv

```
id,name,model_id:id,group_id:id,perm_read,perm_write,perm_create,perm_unlink
access_hrms_country,access_hrms_country,model_hrms_country,base.group_user,1,1,1,1

access_hrms_contact_user,access_hrms_contact_user,model_hrms_contact,hrm_system.group_h
rms_user,1,1,0,0
access_hrms_contact_manager,access_hrms_contact_manager,model_hrms_contact,hrm_system.
group_hrms_manager,1,1,0,0
access_hrms_contact_recruiter,access_hrms_contact_recruiter,model_hrms_contact,hrm_system.
group_hrms_recruiter,1,0,1,0
access_hrms_contact_hr_head,access_hrms_contact_hr_head,model_hrms_contact,hrm_system.
group_hrms_hr_head,1,1,1,1
access_hrms_contact_admin,access_hrms_contact_admin,model_hrms_contact,hrm_system.gro
up_hrms_admin,1,1,1,1

access_hrms_employee_user,access_hrms_employee_user,model_hrms_employee,hrm_system.g
roup_hrms_user,1,1,0,0
access_hrms_employee_manager,access_hrms_employee_manager,model_hrms_employee,hrm_
system.group_hrms_manager,1,1,1,1
access_hrms_employee_recruiter,access_hrms_employee_recruiter,model_hrms_employee,hrm_
system.group_hrms_recruiter,1,1,0,0
access_hrms_employee_hr_head,access_hrms_employee_hr_head,model_hrms_employee,hrm_s
ystem.group_hrms_hr_head,1,1,1,1
access_hrms_employee_admin,access_hrms_employee_admin,model_hrms_employee,hrm_syst
em.group_hrms_admin,1,1,1,1

access_hrms_job_title_user,access_hrms_job_title_user,model_hrms_job_title,hrm_system.grou
p_hrms_user,1,0,0,0
```

access_hrms_job_title_manager,access_hrms_job_title_manager,model_hrms_job_title,hrm_system.group_hrms_manager,1,0,0,0
access_hrms_job_title_recruiter,access_hrms_job_title_recruiter,model_hrms_job_title,hrm_system.group_hrms_recruiter,1,0,0,0
access_hrms_job_title_hr_head,access_hrms_job_title_hr_head,model_hrms_job_title,hrm_system.group_hrms_hr_head,1,1,1,1
access_hrms_job_title_admin,access_hrms_job_title_admin,model_hrms_job_title,hrm_system.group_hrms_admin,1,1,1,1

access_hrms_department_user,access_hrms_department_user,model_hrms_department,hrm_system.group_hrms_user,1,0,0,0
access_hrms_department_manager,access_hrms_department_manager,model_hrms_department,hrm_system.group_hrms_manager,1,1,1,1
access_hrms_department_recruiter,access_hrms_department_recruiter,model_hrms_department,hrm_system.group_hrms_recruiter,1,0,0,0
access_hrms_department_hr_head,access_hrms_department_hr_head,model_hrms_department,hrm_system.group_hrms_hr_head,1,1,1,1
access_hrms_department_admin,access_hrms_department_admin,model_hrms_department,hrm_system.group_hrms_admin,1,1,1,1

access_hrms_vacancy_user,access_hrms_vacancy_user,model_hrms_vacancy,hrm_system.group_hrms_user,1,0,0,0
access_hrms_vacancy_manager,access_hrms_vacancy_manager,model_hrms_vacancy,hrm_system.group_hrms_manager,1,1,1,1
access_hrms_vacancy_recruiter,access_hrms_vacancy_recruiter,model_hrms_vacancy,hrm_system.group_hrms_recruiter,1,1,1,1
access_hrms_vacancy_hr_head,access_hrms_vacancy_hr_head,model_hrms_vacancy,hrm_system.group_hrms_hr_head,1,1,1,1
access_hrms_vacancy_admin,access_hrms_vacancy_admin,model_hrms_vacancy,hrm_system.group_hrms_admin,1,1,1,1

access_hrms_applicant_user,access_hrms_applicant_user,model_hrms_applicant,hrm_system.group_hrms_user,1,0,0,0
access_hrms_applicant_manager,access_hrms_applicant_manager,model_hrms_applicant,hrm_system.group_hrms_manager,1,0,0,0
access_hrms_applicant_recruiter,access_hrms_applicant_recruiter,model_hrms_applicant,hrm_system.group_hrms_recruiter,1,1,1,1
access_hrms_applicant_hr_head,access_hrms_applicant_hr_head,model_hrms_applicant,hrm_system.group_hrms_hr_head,1,1,1,1
access_hrms_applicant_admin,access_hrms_applicant_admin,model_hrms_applicant,hrm_system.group_hrms_admin,1,1,1,1

access_hrms_source_user,access_hrms_source_user,model_hrms_source,hrm_system.group_hrms_user,1,0,0,0
access_hrms_source_manager,access_hrms_source_manager,model_hrms_source,hrm_system.group_hrms_manager,1,1,1,1
access_hrms_source_recruiter,access_hrms_source_recruiter,model_hrms_source,hrm_system.group_hrms_recruiter,1,1,1,1

access_hrms_source_hr_head,access_hrms_source_hr_head,model_hrms_source,hrm_system.group_hrms_hr_head,1,1,1,1
access_hrms_source_admin,access_hrms_source_admin,model_hrms_source,hrm_system.group_hrms_admin,1,1,1,1

access_hrms_meeting_user,access_hrms_meeting_user,model_hrms_meeting,hrm_system.group_hrms_user,1,0,0,0
access_hrms_meeting_manager,access_hrms_meeting_manager,model_hrms_meeting,hrm_system.group_hrms_manager,1,1,1,1
access_hrms_meeting_recruiter,access_hrms_meeting_recruiter,model_hrms_meeting,hrm_system.group_hrms_recruiter,1,1,1,1
access_hrms_meeting_hr_head,access_hrms_meeting_hr_head,model_hrms_meeting,hrm_system.group_hrms_hr_head,1,1,1,1
access_hrms_meeting_admin,access_hrms_meeting_admin,model_hrms_meeting,hrm_system.group_hrms_admin,1,1,1,1

access_hrms_leave_user,access_hrms_leave_user,model_hrms_leave,hrm_system.group_hrms_user,1,1,1,0
access_hrms_leave_manager,access_hrms_leave_manager,model_hrms_leave,hrm_system.group_hrms_manager,1,1,1,1
access_hrms_leave_recruiter,access_hrms_leave_recruiter,model_hrms_leave,hrm_system.group_hrms_recruiter,1,1,1,0
access_hrms_leave_hr_head,access_hrms_leave_hr_head,model_hrms_leave,hrm_system.group_hrms_hr_head,1,1,1,1
access_hrms_leave_admin,access_hrms_leave_admin,model_hrms_leave,hrm_system.group_hrms_admin,1,1,1,1

access_hrms_leave_type_user,access_hrms_leave_type_user,model_hrms_leave_type,hrm_system.group_hrms_user,1,0,0,0
access_hrms_leave_type_manager,access_hrms_leave_type_manager,model_hrms_leave_type,hrm_system.group_hrms_manager,1,1,1,1
access_hrms_leave_type_recruiter,access_hrms_leave_type_recruiter,model_hrms_leave_type,hrm_system.group_hrms_recruiter,1,0,0,0
access_hrms_leave_type_hr_head,access_hrms_leave_type_hr_head,model_hrms_leave_type,hrm_system.group_hrms_hr_head,1,1,1,1
access_hrms_leave_type_admin,access_hrms_leave_type_admin,model_hrms_leave_type,hrm_system.group_hrms_admin,1,1,1,1

access_hrms_allocation_user,access_hrms_allocation_user,model_hrms_allocation,hrm_system.group_hrms_user,1,0,0,0
access_hrms_allocation_manager,access_hrms_allocation_manager,model_hrms_allocation,hrm_system.group_hrms_manager,1,1,1,1
access_hrms_allocation_recruiter,access_hrms_allocation_recruiter,model_hrms_allocation,hrm_system.group_hrms_recruiter,1,0,0,0
access_hrms_allocation_hr_head,access_hrms_allocation_hr_head,model_hrms_allocation,hrm_system.group_hrms_hr_head,1,1,1,1
access_hrms_allocation_admin,access_hrms_allocation_admin,model_hrms_allocation,hrm_system.group_hrms_admin,1,1,1,1

## security.xml

```
<odoo>
  <record id="module_category_hrms" model="ir.module.category">
    <field name="name">HRM System</field>
  </record>
  <record id="group_hrms_user" model="res.groups">
    <field name="name">User</field>
    <field name="category_id" ref="module_category_hrms"/>
  </record>
  <record id="group_hrms_manager" model="res.groups">
    <field name="name">Manager</field>
    <field name="category_id" ref="module_category_hrms"/>
  </record>
  <record id="group_hrms_recruiter" model="res.groups">
    <field name="name">Recruiter</field>
    <field name="category_id" ref="module_category_hrms"/>
  </record>
  <record id="group_hrms_hr_head" model="res.groups">
    <field name="name">Head of HR</field>
    <field name="category_id" ref="module_category_hrms"/>
  </record>
  <record id="group_hrms_admin" model="res.groups">
    <field name="name">Administrator</field>
    <field name="category_id" ref="module_category_hrms"/>
  </record>
  <record id="meeting_record_rule" model="ir.rule">
    <field name="name">Users see only their meetings</field>
    <field name="model_id" ref="model_hrms_meeting"/>
    <field name="domain_force">[('contact_ids', '=', user.contact_id.id)]</field>
    <field name="groups" eval="[
        Command.link(ref('hrm_system.group_hrms_user')),
        Command.link(ref('hrm_system.group_hrms_recruiter')),
      ]"/>
  </record>
  <record id="leave_record_rule_user" model="ir.rule">
    <field name="name">Users see only their leaves</field>
    <field name="model_id" ref="model_hrms_leave"/>
    <field name="domain_force">[('employee_id.contact_id', '=', user.contact_id.id)]</field>
    <field name="groups" eval="[
        Command.link(ref('hrm_system.group_hrms_user')),
        Command.link(ref('hrm_system.group_hrms_recruiter')),
      ]"/>
  </record>
  <record id="leave_record_rule_manager" model="ir.rule">
    <field name="name">Managers see only their leaves and leaves of their teammates</field>
    <field name="model_id" ref="model_hrms_leave"/>
    <field name="domain_force">["|", ('employee_id.contact_id', '=', user.contact_id.id),
('employee_id.manager_id.contact_id', '=', user.contact_id.id)]</field>
    <field name="groups" eval="[
```

```
            Command.link(ref('hrm_system.group_hrms_manager'))
        ]"/>
    </record>
</odoo>
```

## hrms_allocation_views.xml

```
<odoo>
    <record id="hrms_allocation_view_tree" model="ir.ui.view">
        <field name="name">hrms.allocation.view.tree</field>
        <field name="model">hrms.allocation</field>
        <field name="arch" type="xml">
            <tree>
                <field name="name"/>
                <field name="leave_type_id"/>
                <field name="duration"/>
                <field name="employee_id"/>
                <field name="state"/>
            </tree>
        </field>
    </record>
    <record id="hrms_allocation_view_form" model="ir.ui.view">
        <field name="name">hrms.allocation.view.form</field>
        <field name="model">hrms.allocation</field>
        <field name="arch" type="xml">
            <form>
                <header>
                    <button name="action_approve_allocation" type="object" string="Approve"
class="oe_highlight"/>
                    <button name="action_reject_allocation" type="object" string="Reject"/>
                    <field name="state" widget="statusbar" statusbar_visible="pending,approved"/>
                </header>
                <sheet>
                    <div class="oe_title">
                        <label for="name" string="Name"/>
                        <h1>
                            <field name="name" placeholder="e.g. Sick Leave Allocation"/>
                        </h1>
                    </div>
                    <group>
                        <field name="leave_type_id"/>
                        <field name="duration"/>
                        <field name="employee_id"/>
                    </group>
                </sheet>
            </form>
        </field>
    </record>
    <record id="hrms_allocation_action" model="ir.actions.act_window">
        <field name="name">Allocations</field>
```

```
    <field name="res_model">hrms.allocation</field>
    <field name="view_mode">tree,form</field>
  </record>
</odoo>
```

## hrms_applicant_views.xml

```
<odoo>
  <record id="hrms_applicant_view_tree" model="ir.ui.view">
    <field name="name">hrms.applicant.view.tree</field>
    <field name="model">hrms.applicant</field>
    <field name="arch" type="xml">
      <tree string="">
        <field name="name"/>
        <field name="phone"/>
        <field name="email"/>
        <field name="status"/>
        <field name="level"/>
        <field name="vacancy_id"/>
      </tree>
    </field>
  </record>
  <record id="hrms_applicant_view_form" model="ir.ui.view">
    <field name="name">hrms.applicant.view.form</field>
    <field name="model">hrms.applicant</field>
    <field name="arch" type="xml">
      <form>
        <header>
          <button name="action_create_employee" type="object" string="Create Employee"
class="oe_highlight"/>
          <button name="action_reject" type="object" string="Reject Applicant"/>
          <field name="status" widget="statusbar" options="{'clickable': '1'}"
statusbar_visible="review,shortlisted,selected"/>
        </header>
        <sheet>
          <div class="oe_button_box" name="button_box">
            <button name="action_show_meetings" type="object" class="oe_stat_button"
icon="fa-calendar">
              <field name="meeting_count" widget="statinfo"/>
            </button>
          </div>
          <div class="oe_title">
            <label for="name" string="Applicant Name"/>
            <h1>
              <field name="name" placeholder="e.g. Illia Smyhunov"/>
            </h1>
          </div>
          <group>
            <group name="general" string="General">
              <field name="phone"/>
```

```
                    <field name="email"/>
                    <field name="contact_id"/>
                    <field name="country_id"/>
                    <field name="locality"/>
                    <field name="source_id"/>
                </group>
                <group name="professional" string="Professional">
                    <field name="skill_set"/>
                    <field name="english_level"/>
                    <field name="level"/>
                </group>
                <group name="vacancy" string="Vacancy">
                    <field name="vacancy_id"/>
                    <field name="recruiter_id"/>
                </group>
            </group>
        </sheet>
    </form>
  </field>
</record>
<record id="hrms_applicant_view_kanban" model="ir.ui.view">
  <field name="name">hrms.applicant.view.kanban</field>
  <field name="model">hrms.applicant</field>
  <field name="arch" type="xml">
    <kanban>
      <templates>
        <t t-name="kanban-box">
          <div t-attf-class="oe_kanban_global_click">
            <div class="oe_kanban_details">
              <strong>
                <field name="name"/>
              </strong>
              <ul>
                <li>
                  Phone:
                  <field name="phone"/>
                </li>
                <li>
                  Email:
                  <field name="email"/>
                </li>
                <li>
                  Status:
                  <field name="status"/>
                </li>
                <li>
                  Level:
                  <field name="level"/>
                </li>
```

```
                        <li>
                            Vacancy:
                            <field name="vacancy_id"/>
                        </li>
                    </ul>
                </div>
            </div>
        </t>
    </templates>
</kanban>
</field>
</record>
<record id="hrms_applicant_action" model="ir.actions.act_window">
    <field name="name">Applicants</field>
    <field name="res_model">hrms.applicant</field>
    <field name="view_mode">tree,kanban,form</field>
</record>
</odoo>
```

## hrms_contact_views.xml

```
<?xml version="1.0" encoding="utf-8"?>
<odoo>
    <record id="hrms_contact_view_tree" model="ir.ui.view">
        <field name="name">hrms.contact.view.tree</field>
        <field name="model">hrms.contact</field>
        <field name="arch" type="xml">
            <tree string="">
                <field name="name"/>
                <field name="phone"/>
                <field name="email"/>
                <field name="country_id"/>
            </tree>
        </field>
    </record>
    <record id="hrms_contact_view_form" model="ir.ui.view">
        <field name="name">hrms.contact.view.form</field>
        <field name="model">hrms.contact</field>
        <field name="arch" type="xml">
            <form string="Contact Info">
                <sheet>
                    <field name="image" widget="image" class="oe_avatar"/>
                    <div class="oe_title">
                        <label for="name" string="Name"/>
                        <h1>
                            <field name="name" placeholder="e.g. John Doe"/>
                        </h1>
                    </div>
                    <group>
                        <group name="contact_data" string="Contact Data">
```

```
                    <field name="phone"/>
                    <field name="email"/>
                </group>
                <group name="address" string="Address">
                    <field name="country_id"/>
                    <field name="locality"/>
                    <field name="street"/>
                    <field name="house_no" style="width:16%"/>
                    <field name="flat_no" style="width:16%"/>
                </group>
            </group>
        </sheet>
    </form>
  </field>
</record>

<record id="hrms_contact_action" model="ir.actions.act_window">
  <field name="name">Contacts</field>
  <field name="res_model">hrms.contact</field>
  <field name="view_mode">tree,form</field>
</record>

</odoo>
```

## hrms_department_views.xml

```
<odoo>
  <record id="hrms_department_view_tree" model="ir.ui.view">
    <field name="name">hrms.department.view.tree</field>
    <field name="model">hrms.department</field>
    <field name="arch" type="xml">
      <tree string="Departments">
        <field name="name"/>
        <field name="manager_id"/>
        <field name="parent_id"/>
      </tree>
    </field>
  </record>
  <record id="hrms_department_view_form" model="ir.ui.view">
    <field name="name">hrms.department.view.form</field>
    <field name="model">hrms.department</field>
    <field name="arch" type="xml">
      <form string="Department">
        <sheet>
          <div class="oe_button_box" name="button_box">
            <button  name="action_show_employees"  type="object"  class="oe_stat_button"
icon="fa-users">
              <field name="employee_count" widget="statinfo"/>
            </button>
          </div>
```

```
            <div class="oe_title">
               <label for="name" string="Department Name"/>
               <h1>
                  <field name="name"/>
               </h1>
            </div>
            <group>
               <group name="general">
                  <field name="code"/>
                  <field name="parent_id"/>
               </group>
               <group name="managers">
                  <field name="manager_id"/>
               </group>
            </group>
         </sheet>
      </form>
   </field>
</record>
<record id="hrms_department_view_kanban" model="ir.ui.view">
   <field name="name">hrms.department.view.kanban</field>
   <field name="model">hrms.department</field>
   <field name="arch" type="xml">
      <kanban>
         <templates>
            <t t-name="kanban-box">
               <div t-attf-class="oe_kanban_global_click">
                  <div class="oe_kanban_details">
                     <strong>
                        <field name="name"/>
                     </strong>
                     <ul>
                        <li>
                           Code:
                           <field name="code"/>
                        </li>
                        <t t-if="record.parent_id.raw_value">
                           <li>
                           Parent Department:
                              <field name="parent_id"/>
                           </li>
                        </t>
                        <t t-else="">
                           <li>
                           Parent Department: -
                           </li>
                        </t>
                        <t t-if="record.manager_id.raw_value">
                           <li>
```

```
                        Department Manager:
                            <field name="manager_id"/>
                        </li>
                    </t>
                    <t t-else="">
                        <li>
                         Department Manager: -
                        </li>
                    </t>
                    <button class="btn btn-primary" name="action_show_employees"
type="object">Employees</button>
                </ul>
            </div>
        </div>
    </t>
</templates>
</kanban>
</field>
</record>
<record id="hrms_department_action" model="ir.actions.act_window">
<field name="name">Departments</field>
<field name="res_model">hrms.department</field>
<field name="view_mode">kanban,tree,form</field>
</record>
</odoo>
```

## hrms_employee_views.xml

```
<odoo>
<record id="hrms_employee_tree_view" model="ir.ui.view">
<field name="name">hrms.employee.tree.view</field>
<field name="model">hrms.employee</field>
<field name="arch" type="xml">
<tree string="Employees">
<field name="name"/>
<field name="job_title_id"/>
<field name="job_level"/>
<field name="work_phone"/>
<field name="work_email"/>
<field name="manager_id"/>
</tree>
</field>
</record>
<record id="hrms_employee_form_view" model="ir.ui.view">
<field name="name">hrms.employee.form.view</field>
<field name="model">hrms.employee</field>
<field name="arch" type="xml">
<form string="Employees">
<header>
```

```
                <button name="action_create_user" type="object" string="Create User" class="oe_highlight" groups="hrm_system.group_hrms_manager,hrm_system.group_hrms_hr_head,hrm_system.group_hrms_admin"/>
                <button name="action_update_user" type="object" string="Update User"/>
                <button name="action_create_contact" type="object" string="Create Contact" class="oe_highlight" groups="hrm_system.group_hrms_manager,hrm_system.group_hrms_hr_head,hrm_system.group_hrms_admin"/>
                <button name="action_update_contact" type="object" string="Update Contact"/>

            </header>
            <sheet>
                <field name="image" widget="image" class="oe_avatar"/>
                <div class="oe_title">
                    <label for="name" string="Employee Name"/>
                    <h1>
                        <field name="name" placeholder="e.g. John Doe" required="True"/>
                    </h1>
                    <h2>
                        <field name="job_title_id" placeholder="Job Title" />
                    </h2>
                    <h2>
                        <field name="job_level" placeholder="Job Level" />
                    </h2>
                </div>
                <group name="general" string="General info">
                    <group name="work_contacts">
                        <field name="work_phone"/>
                        <field name="work_email"/>
                    </group>
                    <group name="managers">
                        <field name="department_id"/>
                        <field name="manager_id"/>
                    </group>
                </group>
                <notebook>
                    <page name="work_info" string="Work Information">
                        <group name="status" string="Status">
                            <field name="employee_type"/>
                            <field name="starting_date"/>
                            <field name="related_user_id"/>
                        </group>
                    </page>
                    <page name="private_info" string="Private Information">
                        <group>
                            <group name="private_contact" string="Private Contact">
                                <field name="contact_id"/>
                                <field name="private_phone"/>
```

```
                        <field name="private_email"/>
                    </group>
                    <group name="marital" string="Marital">
                        <field name="marital"/>
                    </group>
                    <group name="citizenship" string="Citizenship">
                        <field name="country_id"/>
                        <field name="tax_id"/>
                        <field name="passport_no"/>
                        <field name="gender"/>
                        <field name="birthday"/>
                        <field name="age"/>
                        <field name="place_of_birth"/>
                        <field name="country_of_birth"/>
                    </group>
                </group>
            </page>
        </notebook>
    </sheet>
</form>
</field>
</record>
<record id="hrms_employee_kanban_view" model="ir.ui.view">
    <field name="name">hrms.employee.kanban.view</field>
    <field name="model">hrms.employee</field>
    <field name="arch" type="xml">
        <kanban>
            <field name="id"/>
            <field name="name"/>
            <field name="job_title_id"/>
            <field name="job_level"/>
            <field name="work_phone"/>
            <field name="work_email"/>
            <field name="manager_id"/>
            <field name="image"/>
            <templates>
                <t t-name="kanban-box">
                    <div class="oe_kanban_global_click o_kanban_record_has_image_fill">
                        <div class="o_kanban_image_fill_left d-none d-md-block" t-attf-
style="background-image:url('#{kanban_image('hrms.employee', 'image',
record.id.raw_value)}')"/>
                        <div class="oe_kanban_details">
                            <strong>
                                <field name="name"/>
                            </strong>
                            <ul>
                                <li>
                                    Job Title:
                                    <field name="job_title_id"/>
```

```
                        </li>
                        <li>
                            Job Level:
                            <field name="job_level"/>
                        </li>
                        <li>
                            Work Phone:
                            <field name="work_phone"/>
                        </li>
                        <li>
                            Work Email:
                            <field name="work_email"/>
                        </li>
                    </ul>
                </div>
            </div>
        </t>
    </templates>
</kanban>
</field>
</record>
<record id="hrms_employee_action" model="ir.actions.act_window">
    <field name="name">Employees</field>
    <field name="res_model">hrms.employee</field>
    <field name="view_mode">kanban,tree,form</field>
</record>
</odoo>
```

## hrms_job_title_views.xml

```
<odoo>
<record id="hrms_job_title_view_form" model="ir.ui.view">
    <field name="name">hrms.job.title.view.form</field>
    <field name="model">hrms.job.title</field>
    <field name="arch" type="xml">
        <form>
            <sheet>
                <div class="oe_title">
                    <label for="name" string="Name"/>
                    <h1>
                        <field name="name" placeholder="e.g. Python Engineer"/>
                    </h1>
                </div>
            </sheet>
        </form>
    </field>
</record>
<record id="hrms_job_title_action" model="ir.actions.act_window">
    <field name="name">Job Titles</field>
    <field name="res_model">hrms.job.title</field>
```

```
    <field name="view_mode">tree,form</field>
  </record>
</odoo>
```

## hrms_leave_type_views.xml

```
<odoo>
  <record id="hrms_leave_type_view_form" model="ir.ui.view">
    <field name="name">hrms.leave.type.view.form</field>
    <field name="model">hrms.leave.type</field>
    <field name="arch" type="xml">
      <form>
        <sheet>
          <div class="oe_title">
            <label for="name" string="Name"/>
            <h1>
              <field name="name" placeholder="e.g. Sick Leave"/>
            </h1>
          </div>
          <group>
            <field name="description"/>
          </group>
        </sheet>
      </form>
    </field>
  </record>
  <record id="hrms_leave_type_action" model="ir.actions.act_window">
    <field name="name">Leave Types</field>
    <field name="res_model">hrms.leave.type</field>
    <field name="view_mode">tree,form</field>
  </record>
</odoo>
```

## hrms_leave_views.xml

```
<odoo>
  <record id="hrms_leave_view_tree" model="ir.ui.view">
    <field name="name">hrms.leave.view.tree</field>
    <field name="model">hrms.leave</field>
    <field name="arch" type="xml">
      <tree string="Leaves">
        <field name="name"/>
        <field name="leave_type_id"/>
        <field name="start_date"/>
        <field name="end_date"/>
        <field name="employee_id"/>
        <field name="state"/>
      </tree>
    </field>
  </record>
  <record id="hrms_leave_view_form" model="ir.ui.view">
```

```
      <field name="name">hrms.leave.view.form</field>
      <field name="model">hrms.leave</field>
      <field name="arch" type="xml">
        <form>
          <header>
            <button name="action_approve_leave" type="object" string="Approve" class="oe_highlight"
groups="hrm_system.group_hrms_manager,hrm_system.group_hrms_hr_head,hrm_system.group_hrms_admin"/>
            <button name="action_reject_leave" type="object" string="Reject"
groups="hrm_system.group_hrms_manager,hrm_system.group_hrms_hr_head,hrm_system.group_hrms_admin"/>
            <field name="state" widget="statusbar" statusbar_visible="pending,approved"/>
          </header>
          <sheet>
            <div class="oe_title">
              <label for="name" string="Reason"/>
              <h1>
                <field name="name" placeholder="e.g. Personal Trip"/>
              </h1>
            </div>
            <group>
              <field name="leave_type_id"/>
              <field name="available_days"/>
              <field name="start_date"/>
              <field name="end_date"/>
              <field name="duration"/>
              <field name="employee_id" attrs="{'invisible': [('employee_id', '=', False)]}"/>
            </group>
          </sheet>
        </form>
      </field>
    </record>
    <record id="hrms_leave_view_calendar" model="ir.ui.view">
      <field name="name">hrms.leave.view.calendar</field>
      <field name="model">hrms.leave</field>
      <field name="arch" type="xml">
        <calendar string="Leaves" date_start="start_date" date_stop="end_date" color="state"
mode="month" quick_add="False">
          <field name="name"/>
          <field name="employee_id"/>
          <field name="state" invisible="1" filters="1"/>
        </calendar>
      </field>
    </record>
    <record id="hrms_leave_action" model="ir.actions.act_window">
      <field name="name">Leaves</field>
      <field name="res_model">hrms.leave</field>
      <field name="view_mode">calendar,tree,form</field>
```

```
  </record>
</odoo>
```

## hrms_meeting_views.xml

```
<odoo>
  <record id="hrms_meeting_view_calendar" model="ir.ui.view">
    <field name="name">hrms.meeting.view.calendar</field>
    <field name="model">hrms.meeting</field>
    <field name="arch" type="xml">
      <calendar string="Meetings" date_start="start_time" mode="month">
        <field name="name"/>
      </calendar>
    </field>
  </record>
  <record id="hrms_meeting_view_tree" model="ir.ui.view">
    <field name="name">hrms.meeting.view.tree</field>
    <field name="model">hrms.meeting</field>
    <field name="arch" type="xml">
      <tree string="Meetings">
        <field name="name"/>
        <field name="start_time"/>
        <field name="end_time"/>
        <field name="employee_id"/>
      </tree>
    </field>
  </record>
  <record id="hrms_meeting_view_form" model="ir.ui.view">
    <field name="name">hrms.meeting.view.form</field>
    <field name="model">hrms.meeting</field>
    <field name="arch" type="xml">
      <form>
        <sheet>
          <div class="oe_title">
            <label for="name" string="Subject"/>
            <h1>
              <field name="name" placeholder="e.g. Technical Interview"/>
            </h1>
          </div>
          <group>
            <field name="contact_ids" widget="many2many_tags"/>
            <field name="start_time"/>
            <field name="end_time"/>
            <field name="employee_id"/>
            <field name="description"/>
            <field name="meeting_url"/>
          </group>
        </sheet>
      </form>
    </field>
```

```
    </record>
    <record id="hrms_meeting_action" model="ir.actions.act_window">
        <field name="name">Meetings</field>
        <field name="res_model">hrms.meeting</field>
        <field name="view_mode">calendar,tree,form</field>
    </record>
</odoo>
```

## hrms_vacancy_views.xml

```
<odoo>
    <record id="hrms_vacancy_view_tree" model="ir.ui.view">
        <field name="name">hrms.vacancy.view.tree</field>
        <field name="model">hrms.vacancy</field>
        <field name="arch" type="xml">
            <tree string="Vacancies">
                <field name="name"/>
                <field name="code"/>
                <field name="job_title_id"/>
                <field name="level"/>
                <field name="status"/>
            </tree>
        </field>
    </record>
    <record id="hrms_vacancy_view_form" model="ir.ui.view">
        <field name="name">hrms.vacancy.view.form</field>
        <field name="model">hrms.vacancy</field>
        <field name="arch" type="xml">
            <form>
                <header>
                    <field name="status" widget="statusbar" options="{'clickable': '1'}"/>
                </header>
                <sheet>
                    <div class="oe_button_box" name="button_box">
                        <button  name="action_show_applicants"  type="object"  class="oe_stat_button"
icon="fa-users">
                            <field name="applicant_count" widget="statinfo"/>
                        </button>
                    </div>
                    <div class="oe_title">
                        <label for="name" string="Vacancy Name"/>
                        <h1>
                            <field name="name" placeholder="e.g. Junior Python Developer"/>
                        </h1>
                    </div>
                    <group>
                        <group name="general" string="General Info">
                            <field name="code"/>
                            <field name="recruiter_id"/>
                            <field name="technical_interviewer_id"/>
```

```
                    <field name="job_title_id"/>
                </group>
                <group name="requirments" string="Requirements">
                    <field name="level"/>
                    <field name="english_level"/>
                    <field name="technology_stack"/>
                    <field name="salary_rate"/>
                </group>
            </group>
        </sheet>
    </form>
  </field>
 </record>
 <record id="hrms_vacancy_action" model="ir.actions.act_window">
  <field name="name">Vacancies</field>
  <field name="res_model">hrms.vacancy</field>
  <field name="view_mode">tree,form</field>
 </record>
</odoo>
```

## menu.xml

```
<odoo>
<?xml version="1.0" encoding="utf-8"?>
<odoo>
 <menuitem
  id="hrms_root_menu"
  name="HRM System"/>

 <menuitem
  id="hrms_employee_submenu"
  name="Employees"
  action="hrms_employee_action"
  parent="hrms_root_menu"
  sequence="1"/>

 <menuitem
  id="hrms_department_submenu"
  name="Departments"
  action="hrms_department_action"
  parent="hrms_root_menu"
  sequence="2"/>

 <menuitem
  id="hrms_recruitment_submenu"
  name="Recruitment"
  parent="hrms_root_menu"

groups="hrm_system.group_hrms_recruiter,hrm_system.group_hrms_hr_head,hrm_system.grou
p_hrms_admin"
  sequence="3"/>
```

```
        <menuitem
            id="hrms_vacancy_submenu"
            name="Vacancies"
            action="hrms_vacancy_action"
            parent="hrms_recruitment_submenu"/>

        <menuitem
            id="hrms_applicant_submenu"
            name="Applicants"
            action="hrms_applicant_action"
            parent="hrms_recruitment_submenu"/>

    <menuitem
        id="hrms_contact_submenu"
        name="Contacts"
        action="hrms_contact_action"
        parent="hrms_root_menu"
        sequence="5"/>

    <menuitem
        id="hrms_event_submenu"
        name="Events"
        parent="hrms_root_menu"
        sequence="4"/>

        <menuitem
            id="hrms_meeting_submenu"
            name="Meetings"
            action="hrms_meeting_action"
            parent="hrms_event_submenu"/>

        <menuitem
            id="hrms_leave_submenu"
            name="Leaves"
            action="hrms_leave_action"
            parent="hrms_event_submenu"/>

    <menuitem
        id="hrms_configuration_submenu"
        name="Configuration"
        parent="hrms_root_menu"

groups="hrm_system.group_hrms_manager,hrm_system.group_hrms_hr_head,hrm_system.grou
p_hrms_admin"/>

        <menuitem
            id="hrms_work_category"
            name="Work Settings"
```

```
        parent="hrms_configuration_submenu"/>

        <menuitem
            id="hrms_job_title_submenu"
            name="Job Titles"
            action="hrms_job_title_action"
            parent="hrms_work_category"/>

    <menuitem
        id="hrms_leaves_category"
        name="Leaves"
        parent="hrms_configuration_submenu"/>

        <menuitem
            id="hrms_leave_type_submenu"
            name="Leave Types"
            action="hrms_leave_type_action"
            parent="hrms_leaves_category"/>

        <menuitem
            id="hrms_allocation_submenu"
            name="Allocations"
            action="hrms_allocation_action"
            parent="hrms_leaves_category"/>
</odoo>
```

## res_users_views.xml

```
<odoo>
  <record id="view_users_simple_form_inherit_hrm_system" model="ir.ui.view">
    <field name="name">res.users.view.form.inherit.hrm.system</field>
    <field name="model">res.users</field>
    <field name="inherit_id" ref="base.view_users_simple_form"/>
    <field name="arch" type="xml">
      <field name="mobile" position="attributes">
        <attribute name="invisible">1</attribute>
      </field>
    </field>
  </record>
  <record id="view_users_form_inherit_hrm_system" model="ir.ui.view">
    <field name="name">view.users.form.inherit.hrm.system</field>
    <field name="model">res.users</field>
    <field name="inherit_id" ref="base.view_users_form"/>
    <field name="arch" type="xml">
      <field name="login" position="after">
        <group>
          <field name="contact_id"/>
        </group>
      </field>
    </field>
```

```
    </record>
</odoo>
```

## hrms_department_data.xml

```
<odoo>
    <record id="marketing_office" model="hrms.department">
        <field name="name">Marketing Office</field>
        <field name="code">MRO</field>
    </record>
    <record id="finance_office" model="hrms.department">
        <field name="name">Finance Office</field>
        <field name="code">FIN</field>
    </record>
    <record id="accounting_office" model="hrms.department">
        <field name="name">Accounting Office</field>
        <field name="code">ACC</field>
    </record>
    <record id="legal_office" model="hrms.department">
        <field name="name">Legal Office</field>
        <field name="code">LGO</field>
    </record>
    <record id="recruiting_department" model="hrms.department">
        <field name="name">Recruiting Department</field>
        <field name="code">RecD</field>
    </record>
    <record id="development_operations_office" model="hrms.department">
        <field name="name">Development Operations Office</field>
        <field name="code">DeOO</field>
    </record>
    <record id="project_management_office" model="hrms.department">
        <field name="name">Project Management Office</field>
        <field name="code">PMO</field>
    </record>
    <record id="hrms_department" model="hrms.department">
        <field name="name">hrms Department</field>
        <field name="code">hrmsD</field>
    </record>
    <record id="software_development_office" model="hrms.department">
        <field name="name">Software Development Office</field>
        <field name="code">SDO</field>
    </record>
    <record id="sales_office" model="hrms.department">
        <field name="name">Sales Office</field>
        <field name="code">SLO</field>
    </record>
    <record id="business_development_department" model="hrms.department">
        <field name="name">Business Development Department</field>
        <field name="code">BDD</field>
    </record>
```

```
    <record id="engagement_office" model="hrms.department">
        <field name="name">Engagement Office</field>
        <field name="code">ENG</field>
    </record>
    <record id="support_and_maintenance_office" model="hrms.department">
        <field name="name">Support and Maintenance Office</field>
        <field name="code">SMO</field>
    </record>
    <record id="business_analyst_office" model="hrms.department">
        <field name="name">Business Analyst Office</field>
        <field name="code">BAO</field>
    </record>
    <record id="user_experience_office" model="hrms.department">
        <field name="name">User Experience Office</field>
        <field name="code">UXO</field>
    </record>
    <record id="learning_and_development_department" model="hrms.department">
        <field name="name">Learning and Development Department</field>
        <field name="code">LDD</field>
    </record>
    <record id="chief_operating_officer" model="hrms.department">
        <field name="name">Chief Operating Officer</field>
        <field name="code">COO</field>
    </record>
    <record id="chief_technology_officer" model="hrms.department">
        <field name="name">Chief Technology Officer</field>
        <field name="code">CTO</field>
    </record>
    <record id="quality_assurance_office" model="hrms.department">
        <field name="name">Quality Assurance Office</field>
        <field name="code">QAO</field>
    </record>
    <record id="research_and_development" model="hrms.department">
        <field name="name">Research and Development</field>
        <field name="code">RDO</field>
    </record>
    <record id="delivery_department" model="hrms.department">
        <field name="name">Delivery Department</field>
        <field name="code">DEL</field>
    </record>
    <record id="office_support_department" model="hrms.department">
        <field name="name">Office Support Department</field>
        <field name="code">OFS</field>
    </record>
     <record id="partnership_department" model="hrms.department">
        <field name="name">Partnership Department</field>
        <field name="code">PART</field>
    </record>
</odoo>
```

## mail_template_data.xml

```
<odoo>
  <record id="email_template_employee_created" model="mail.template">
    <field name="name">HRM System: Employee has been created</field>
    <field name="model_id" ref="hrm_system.model_hrms_employee"/>
    <field name="email_to">{{ object.private_email }}</field>
    <field name="subject">Welcome to our company!</field>
    <field name="body_html" type="html">
      <div style="margin: 0px; padding: 0px;">
        <p>
          Hello,
          <t t-out="object.name">{Employee's Name}</t>!
          <br/>
          <br/>
          Thank you for joining us!
          <br/>
          <br/>
          Your working email is  <t t-out="object.work_email">{Working Email}</t>.
          To get your temporary password please contact your system administrator.
        </p>
      </div>
    </field>
  </record>
</odoo>
```

**ДОДАТОК Б**

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ**
**«ДНІПРОВСЬКА ПОЛІТЕХНІКА»**

**Факультет інформаційних технологій**
**Кафедра програмного забезпечення комп'ютерних систем**

# ВІДГУК

Наукового керівника *Мороза Бориса Івановича, доктора технічних наук, професора, професора кафедри програмного забезпечення комп'ютерних систем*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання, посада, місце роботи)

**на магістерську роботу**

студента *Смигунова Іллі Володимировича*
(прізвище, ім'я, по батькові)

курсу II групи *121м-21-1*

спеціальності *Інженерія програмного забезпечення*

освітньої програми *Інженерія програмного забезпечення*

на тему *Розробка та дослідження ефективності впровадження програмного забезпечення для управління персоналом з використанням HRM системи на базі автоматизованих ключових HR-процесів*

Актуальність теми *Тема магістерською роботи є актуальною, бо в наш час бізнес потребує автоматизації внутрішніх процесів для підвищення ефективності праці співробітників та HR-відділ не є виключенням*

Мета досліджень *Метою є розробка автоматизованої системи для керування персоналом та дослідження ефективності впровадження даного програмного продукту*

Коротка характеристика розділів роботи *У першому розділі було проаналізовано предметну галузь, визначено актуальність поставленої задачі та призначення розробки проекту, наведено базову термінологією HR. У другому розділі було проаналізовано ринок HRM систем, а також наведено базову структуру модулів, архітектуру, методи та інструменти*

*реалізації автоматизованої системи керування персоналом. У третьому розділі було розглянуто структуру проекту магістерської роботи та пояснено, для чого використовується та чи інша директорія. Також було проілюстровано структуру бази даних та інтерфейс.*

Практичне значення роботи *Практичне призначення роботи полягає в тому, що запропоноване рішення дозволяє мінімізувати час на менеджмент людського ресурсу, а також полегшує процес відбору нових кадрів.*

Зауваження та недоліки *Оформлення пояснювальної записки магістерської роботи відповідає стандартам. Суттєвих зауважень не маю, окрім недоліку математичних методів та формул.*

Висновки та оцінка *Магістерська робота виконана в повному обсязі та відповідно до вимог. Магістерська робота заслуговує оцінки 90 «відмінно».*

Науковий керівник *Мороз Борис Іванович, професор кафедри програмного забезпечення комп'ютерних систем*
(прізвище, ім’я, по батькові, посада, місце роботи)

«___»_____2022 р. _____
(підпис)

**ДОДАТОК В**

**РЕЦЕНЗІЯ**
**на кваліфікаційну роботу**

студента *Смигунова Іллі Володимировича*
(прізвище, ім'я, по батькові)

курсу ІІ групи *121м-21-1*

кафедри програмного забезпечення комп'ютерних систем

спеціальності *Інженерія програмного забезпечення*

освітньої програми *Інженерія програмного забезпечення*

Тема роботи *Розробка та дослідження ефективності впровадження програмного забезпечення для управління персоналом з використанням HRM системи на базі автоматизованих ключових HR-процесів*

*Стисла характеристика розділів роботи* *Перший розділ містить аналіз предметної галузі, описане завдання і вимоги до програмного продукту. В другому розділі студент аналізує ринок систем керування персоналом, а також знайомить з базовою архітектурою, структурою модулів та методами реалізації. Третій розділ присвячений розробці системи, опису бази даних, структурі модуля та інтерфейсу користувача.*

Пропозиції, внесені студентом, рівень їх наукового обґрунтування *Було запропоновано новий підхід до дослідження та розробки систем керування людським ресурсом, озираючись на новітні технології. Важливість такого рішення обґрунтована статистичними даними та ситуацією на українському ринку.*

Практичне значення роботи *Практична мета цієї роботи полягає в тому, що запропоноване рішення мінімізує час, витрачений на управління людськими ресурсами , а також полегшує процес відбору нового персоналу.*

Якість оформлення роботи *Якість оформлення магістерської роботи відповідає стандартам, описаним в методичних вказівках.*

Недоліки в роботі *В цілому, робота виконана гарно та об'ємно. До*

*зауважень можна віднести недолік математичних методів.*

Загальний висновок *В процесі виконання магістерської роботи студентом була створена автоматизована система керування персоналом, що дійсно має цінність для ведення бізнесу. Праця поділена на три розділи, що відповідають методичним вказівкам. Студента можна вважати готовим до самостійної роботи як спеціаліста.*

(підготовленість студента до самостійної роботи як спеціаліста)

Оцінка магістерської роботи *Вважаю що магістерська робота заслуговує оцінки 90 «відмінно».*

Рецензент *Корнієнко Валерій Іванович, доктор технічних наук, професор, завідувач кафедри безпеки інформації та телекомунікацій.*

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання, посада, місце роботи)

«___»_____2022 р. _____

(підпис)

**ДОДАТОК Г**

## Перелік файлів на диску

| Ім’я файлу | Опис |
|---|---|
| Пояснювальні документи | |
| Диплом Смигунов І.В. 121м-21-1.docx | Пояснювальна записка до кваліфікаційної роботи. Документ Word. |
| Диплом Смигунов І.В. 121м-21-1.pdf | Пояснювальна записка до кваліфікаційної роботи в форматі PDF |
| Програма | |
| Смигунов.rar | Архів. Містить коди програми і скомпільовану програму |
| Презентація | |
| Смигунов 121м-21-1.ppt | Презентація кваліфікаційної роботи |