Цель задания: Получить практические навыки разработки модулей в Odoo, включая настройку среды, создание моделей, представлений, действий и интеграцию с GitHub.

> ⓘ **Внимание!**
>
> Задание может показаться большим, но только из-за его подробного описания и обилия подсказок.
>
> На решение отводится 1-2 недели.

Шаг 1: Установка Odoo через Docker

Задача:

- Развернуть сервер Odoo на вашем ПК с использованием Docker.
- Создать базу данных через веб-интерфейс.
- В результате должно быть доступно приложение Odoo по адресу `http://localhost:8069`.

Подсказки:

- Используйте официальное руководство по установке Odoo с помощью Docker.
- При создании базы данных не обязательно заполнять все поля. В поле Email можно указать, например, `admin@example.com` или просто `admin`

Шаг 2: Создание и установка пустого модуля

Задача:

- Создать свой собственный пустой модуль с именем `test_module`.
- Разместить модуль в отдельной папке `extra_addons`.
- Установить модуль в Odoo через веб-интерфейс.
- Настроить автоматическое обновление модуля при перезапуске сервера Odoo.

Подсказки:

- Убедитесь, что Odoo знает о директории `extra_addons`. Для этого отредактируйте файл конфигурации Odoo, добавив путь к `extra_addons` в параметр `addons_path`.
- Перед установкой модуля включите режим отладки (Debug mode) в Odoo и обновите список приложений, нажав кнопку "Update Apps List" на странице **Приложения**.

- Для автоматического обновления модуля при перезапуске сервера используйте параметр `-u` при запуске Odoo.

Шаг 3: Создание модели `test.model` с полями

Задача:

- Создать модель с именем `test.model`.
- Добавить в модель примеры основных типов полей Odoo (кроме реляционных): `Char`, `Text`, `Integer`, `Float`, `Boolean`, `Date`, `Datetime`, `Selection`, `Binary`.
- Создать представления **Дерево** (Tree) и **Форма** (Form) для модели, отобразив в них все созданные поля.

Подсказки:

- Используйте документацию Odoo по основным типам полей.
- При создании представлений убедитесь, что все поля правильно отображаются и имеют понятные метки.

Шаг 4: Добавление поля `Создатель документа`

Задача:

- Добавить в модель `test.model` поле `document_creator` типа `Many2one`, связанное с моделью `res.users`.
- Поле должно называться "Создатель документа".
- Установить значение по умолчанию для этого поля равным текущему пользователю.
- Сделать поле обязательным для заполнения.

Подсказки:

- Значение текущего пользователя доступно через `self.env.user`.
- Не стесняйтесь искать подсказки в поиске по коду в других модулях Odoo.

Шаг 5: Добавление поля `Ответственный`

Задача:

- Добавить поле `responsible_partner_id` типа `Many2one`, связанное с моделью `res.partner`.
- Поле должно называться "Ответственный".
- Сделать поле доступным для редактирования только после заполнения поля "Создатель документа".

Подсказки:

- Для применения условий отображения или редактирования используйте атрибут `attrs` в XML-файлах представлений.

Шаг 6: Создание модели `test.model.line` и поля `Клиенты`

Задача:

- Создать новую модель `test.model.line`.
- В этой модели определить следующие поля:
  - `test_model_id` типа `Many2one`, связанное с `test.model`.
  - `partner_id` типа `Many2one`, связанное с `res.partner`.
  - `email` типа `Char`, связанное через `related` с полем `email` модели `res.partner`.
- В модели `test.model` добавить поле `clients` типа `One2many`, связанное с моделью `test.model.line`.

Подсказки:

- Поле `email` должно быть определено с параметром `related='partner_id.email'`.
- Такая структура моделей имитирует связь "многие ко многим" через промежуточную модель.

Шаг 7: Добавление логики для полей выбора

Задача:

- В модели `test.model` добавить три поля типа `Boolean`:
  - `option_one` с меткой "Опция 1".
  - `option_two` с меткой "Опция 2".
  - `select_all` с меткой "Выбрать все".
- Реализовать следующую логику:
  - Если поля `option_one` и `option_two` установлены в `True`, то поле `select_all` автоматически устанавливается в `True`.
  - Если все три поля установлены в `True` и пользователь снимает отметку с `select_all`, то поля `option_one` и `option_two` должны автоматически стать `False`.
  - Если все три поля установлены в `True` и пользователь снимает отметку с `option_one` или `option_two`, то поле `select_all` должно автоматически стать `False`.

Подсказки:

- Для реализации такой логики можно использовать декоратор `@api.onchange` в модели.
- Обратите внимание на возможные циклы при изменении значений полей; аккуратно продумайте условия.

Шаг 8: Скрытие ненужных полей под условием

Задача:

- Перенести все дополнительные поля из Шага 3 (кроме тех, что используются в дальнейших заданиях) в отдельную группу внизу формы.
- Сделать эту группу невидимой по умолчанию.
- Добавить условие отображения группы:
  - Группа становится видимой, если:
    - Поле `option_one` установлено в `True` **и** заполнено поле "Создатель документа".
    - **Или** поле `option_two` установлено в `True` **и** заполнено поле "Ответственный".

Подсказки:

- Используйте атрибут `attrs` с логическими условиями в префиксной нотации (Польская запись).
- Пример использования `attrs` на группе:

```
<group attrs="{'invisible': ['!', '|', '&', ('option_one', '=', True),
('document_creator', '!=', False), '&', ('option_two', '=', True),
('responsible_person', '!=', False)]}">
    <!-- поля -->
</group>
```

- Логические операторы в Odoo:
  - `&` — логическое И.
  - `|` — логическое ИЛИ.
  - `!` — логическое НЕ.

Шаг 9: Добавление кнопки и визарда для создания клиентов

Задача:

- В представление формы `test.model` добавить кнопку "Создать и добавить клиента".
- По нажатию на кнопку должен открываться визард с полем `name` и тремя кнопками:
  - "Создать и добавить".
  - "Создать и изменить".
  - "Отменить".
- Логика кнопок:
  - "Создать и добавить": создаёт контакт `res.partner` с указанным именем и добавляет его в поле `clients`.
  - "Создать и изменить": создаёт контакт `res.partner`, добавляет его в `clients` и открывает форму созданного контакта для редактирования.
  - "Отменить": закрывает визард без сохранения.
- Система должна предотвращать создание контакта с пустым или неуникальным именем, выводя соответствующие сообщения об ошибках.

Подсказки:

- Визард следует реализовать как временную модель (`transient model`).
- Для проверки уникальности имени используйте ограничения SQL или методы `@api.constrains`.
- Для кнопки "Отменить" используйте стандартное действие закрытия окна.
- Для открытия формы созданного контакта верните действие `ir.actions.act_window` с параметрами для открытия формы `res.partner`.

Шаг 10: Загрузка проекта на GitHub

Задача:

- Создать репозиторий на GitHub и загрузить в него папку с модулем `test_module`.
- Оформить код согласно стандартам PEP8.
- Добавить файл `README.md` с описанием модуля и инструкциями по установке.
- Создать новую ветку, внести в неё изменения (например, обновить `README.md`), закоммитить и запушить изменения.
- Создать Pull Request и слить новую ветку с основной (merge).

Подсказки:

- Используйте осмысленные сообщения коммитов.
- Убедитесь, что в репозитории нет лишних файлов (например, файлов с паролями или сгенерированных данных).

Дополнительные инструкции

- **Качество кода:** При оценке задания основное внимание уделяется качеству и аккуратности кода, а также соблюдению сроков выполнения.
- **Отчётность:** Рекомендуется регулярно сообщать о своих достижениях и выполненных этапах в специальном чате или канале коммуникации, чтобы мы могли отслеживать ваш прогресс и степень заинтересованности.
- **Помощь и вопросы:** Если у вас возникнут вопросы или трудности, не стесняйтесь обращаться за помощью в чат. Документация Odoo может быть неполной, поэтому совместное обсуждение может быть очень полезным.
- **Сроки выполнения:** На выполнение задания отводится от **1 до 3 недель**.

Результат выполнения

После завершения задания вы приобретёте необходимые базовые навыки работы с Odoo, которые достаточны для начала выполнения реальных коммерческих проектов.

Удачи в выполнении задания!