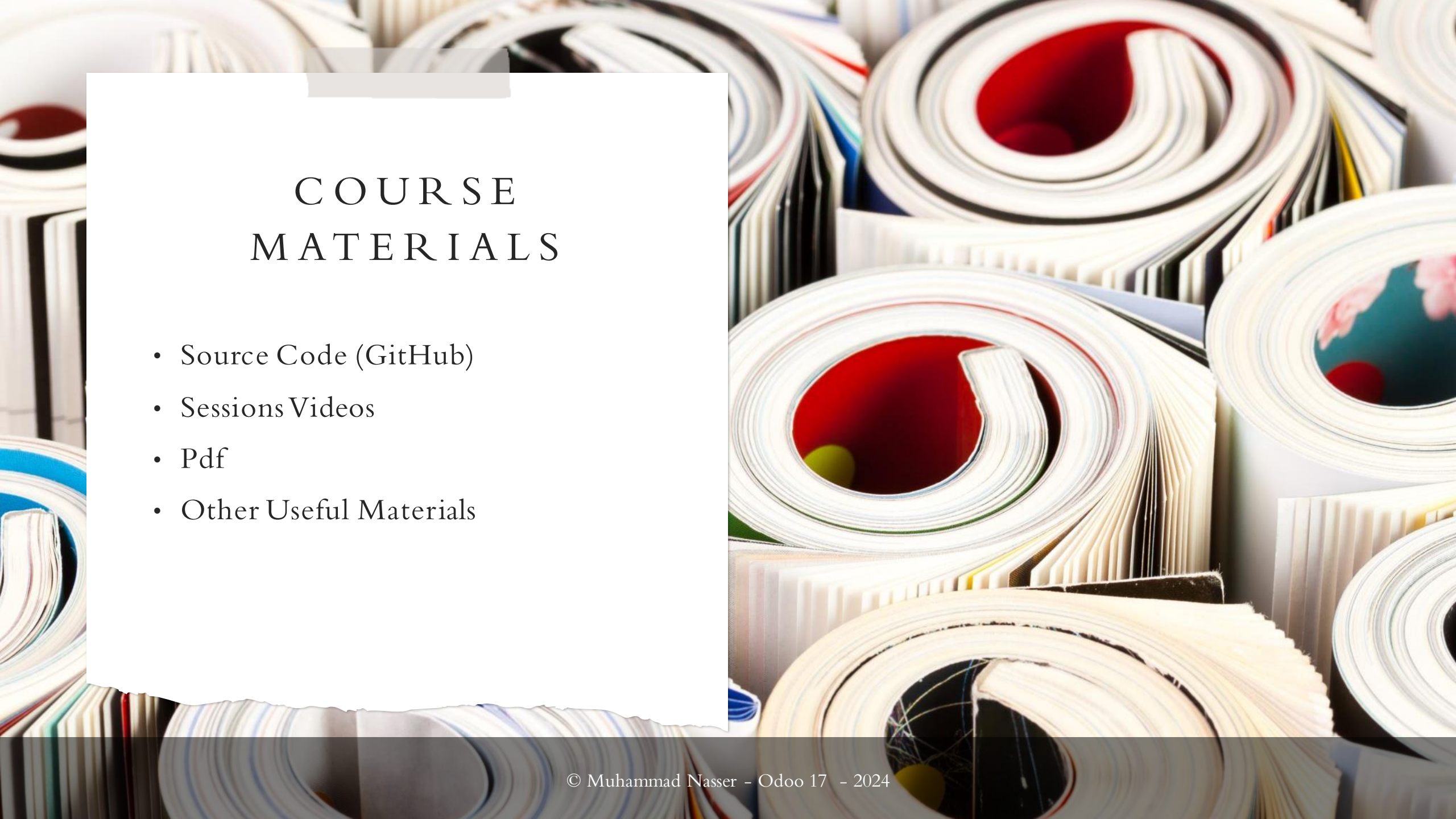
odoo Development

Muhammad Nasser

Information Technology Institute

Day 1

WHAT YOU'LL LEARN?



COURSE MATERIALS

- Source Code (GitHub)
- Sessions Videos
- Pdf
- Other Useful Materials

SKILLS:

- Problem Solving
- Debugging
- Self-Study
- Work with documentation
- Tasks & Projects

COURSE DURATION

- **4 Days** 24 Hours
- **12** Hours Sessions
- **12** Hours Laps

RULES:

1. Questions?
2. Attendance?
3. Breaks?

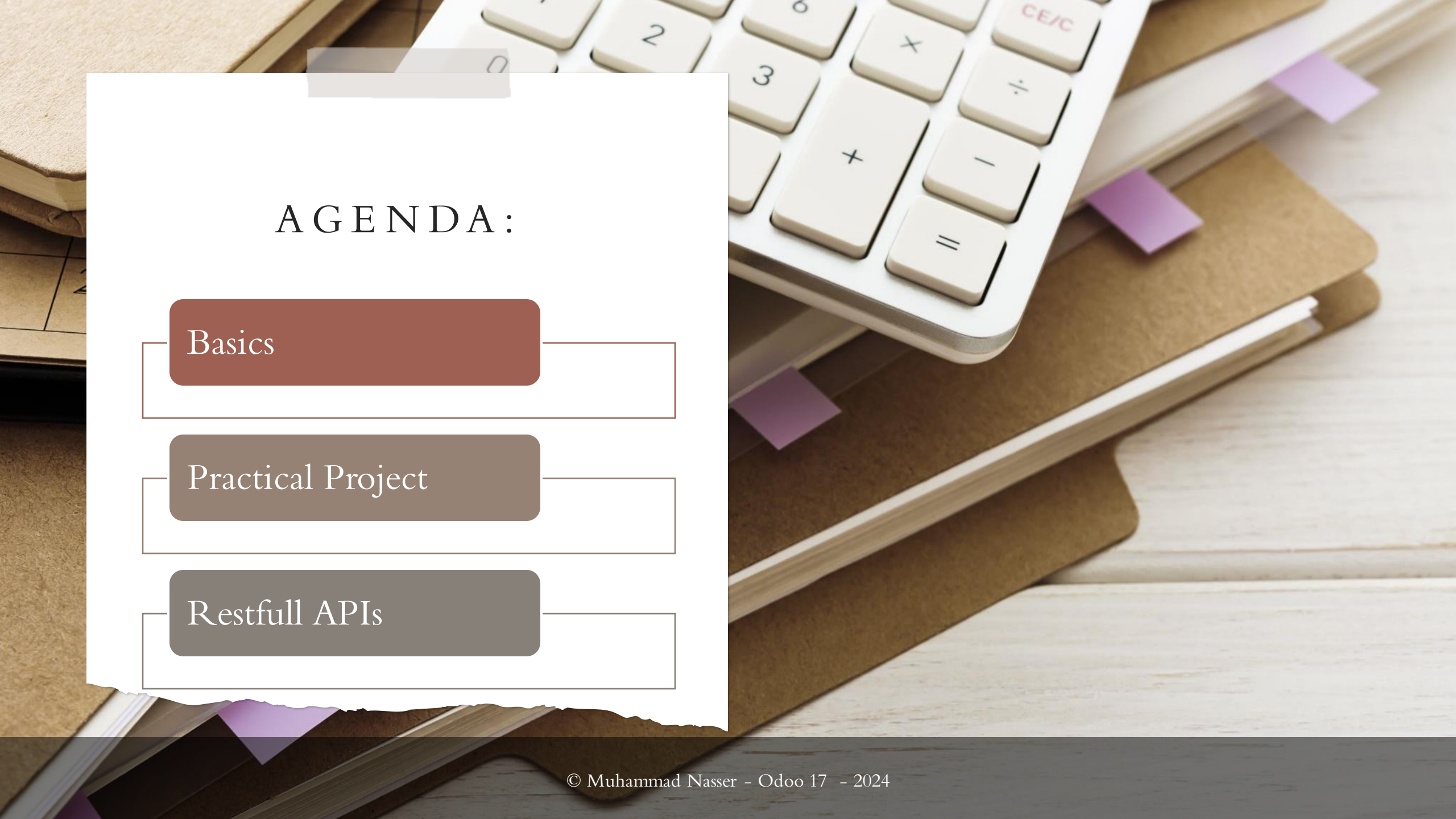
Prerequisites?

Basic of Python

Basic of Linux

Basic of Postgres

Content:

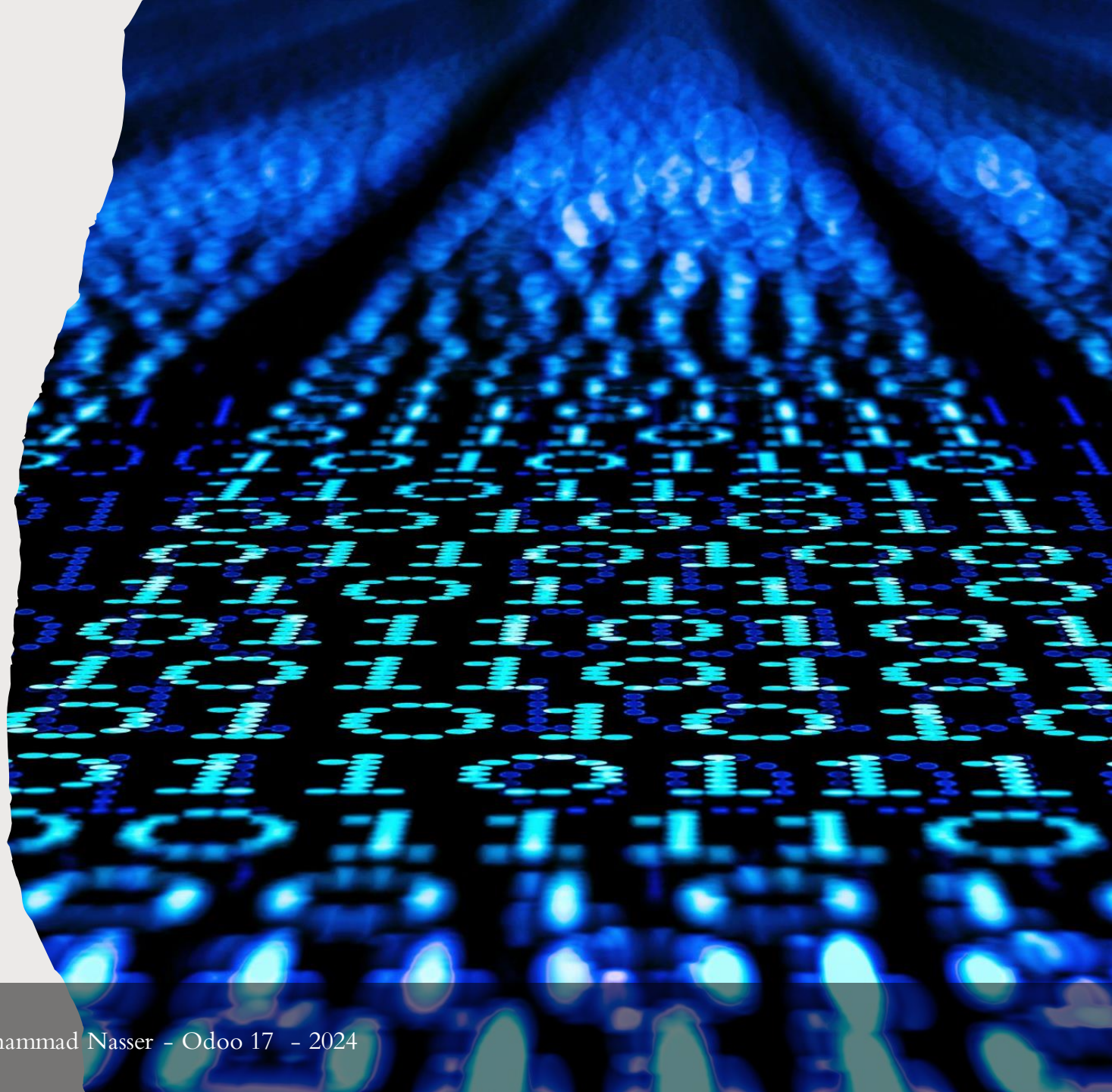


AGENDA:

- Basics
- Practical Project
- Restfull APIs

BASICS:

INTRODUCTION



ERP

ERP (or **Enterprise Resource Planning**) is a category of software that manages the various functions across your business, in one system.

Odoo can be considered both a software product and a framework. published by **Odoo SA**, a **software company** based in Belgium founded by Fabien Pinckaers.

The Odoo software is company-driven, meaning that its roadmap and development are both tightly controlled by Odoo SA. However, it still follows **open source.**

The Odoo software follows the **open core business model**, meaning that some parts of the software are **open source** and some parts are **proprietary**. As a result of this model, Odoo publishes two editions:

The **Community Edition** (CE) is publicly available, open source, and licensed under LGPL.

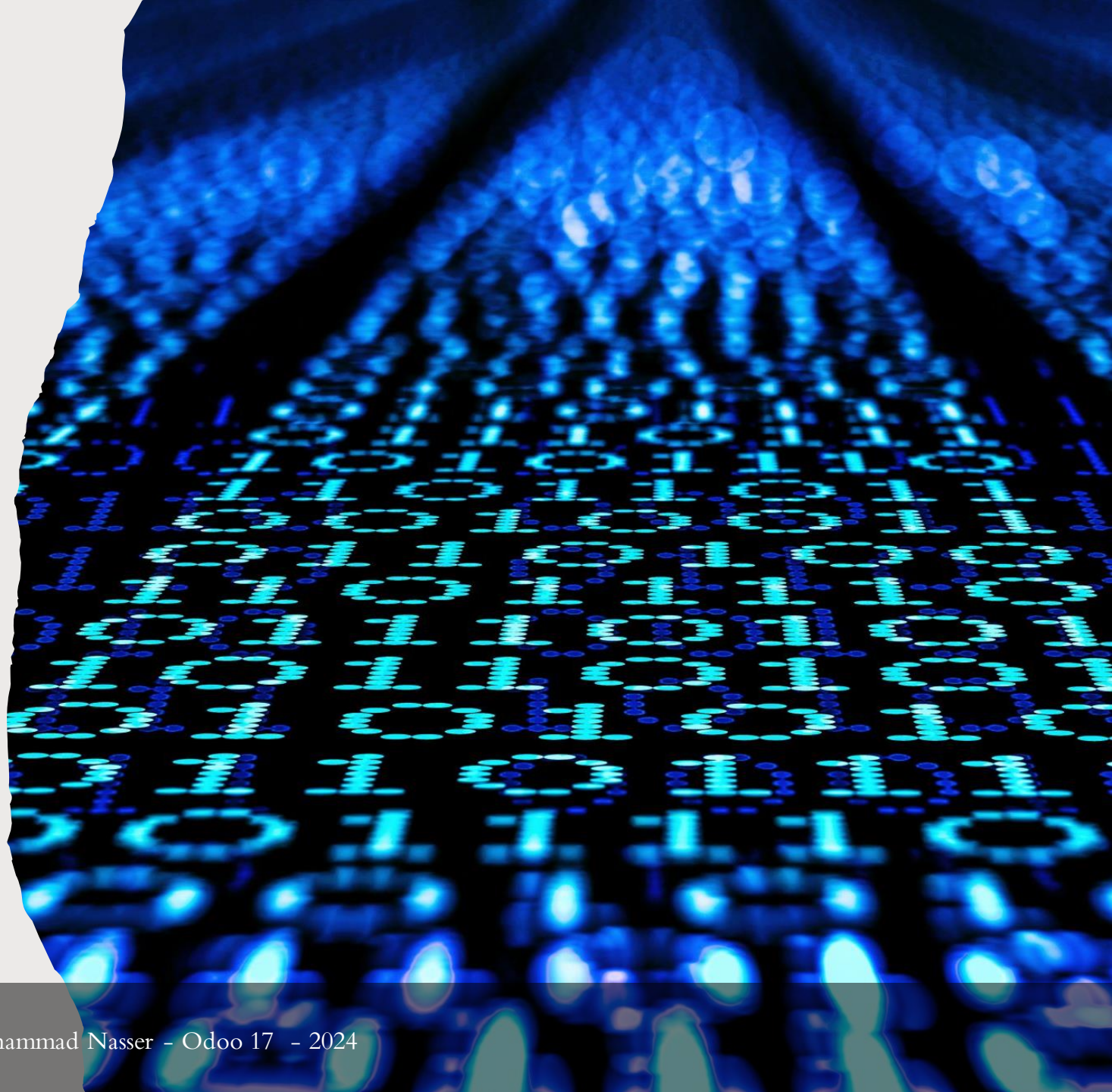
The **Enterprise Edition** (EE) is available only to official partners and customers and has a proprietary license requiring non-disclosure of the code.

Odoo is also a **framework** that allows developers to build custom applications and modules on top of the existing platform. It provides a robust development framework with a modular architecture, allowing developers to extend the core functionality, create new applications, integrate with external systems, and customize the user interface.

The Odoo EE works as a layer of additional modules on top of the Odoo CE core, offering premium features that are expected to provide enough value to motivate users to upgrade. The revenue from the Odoo EE funds the development for both the Odoo CE and EE. The Odoo founder and CEO Fabien Pinckaers has repeatedly pledged a commitment to keeping **80%** of the code as open source in the Odoo CE and **20%** in the proprietary Odoo EE.

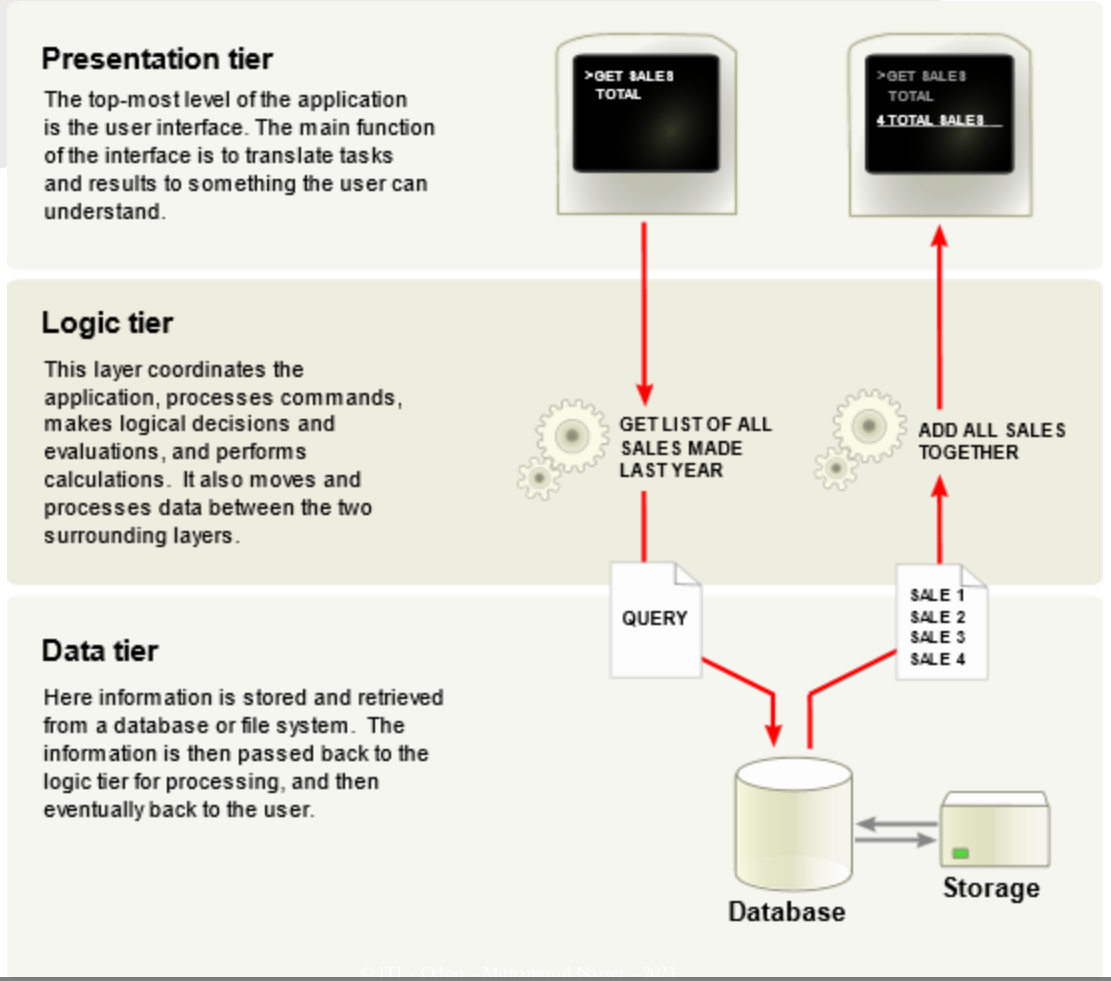
BASICS:

- INTRODUCTION
- ODOO ARCHITECTURE



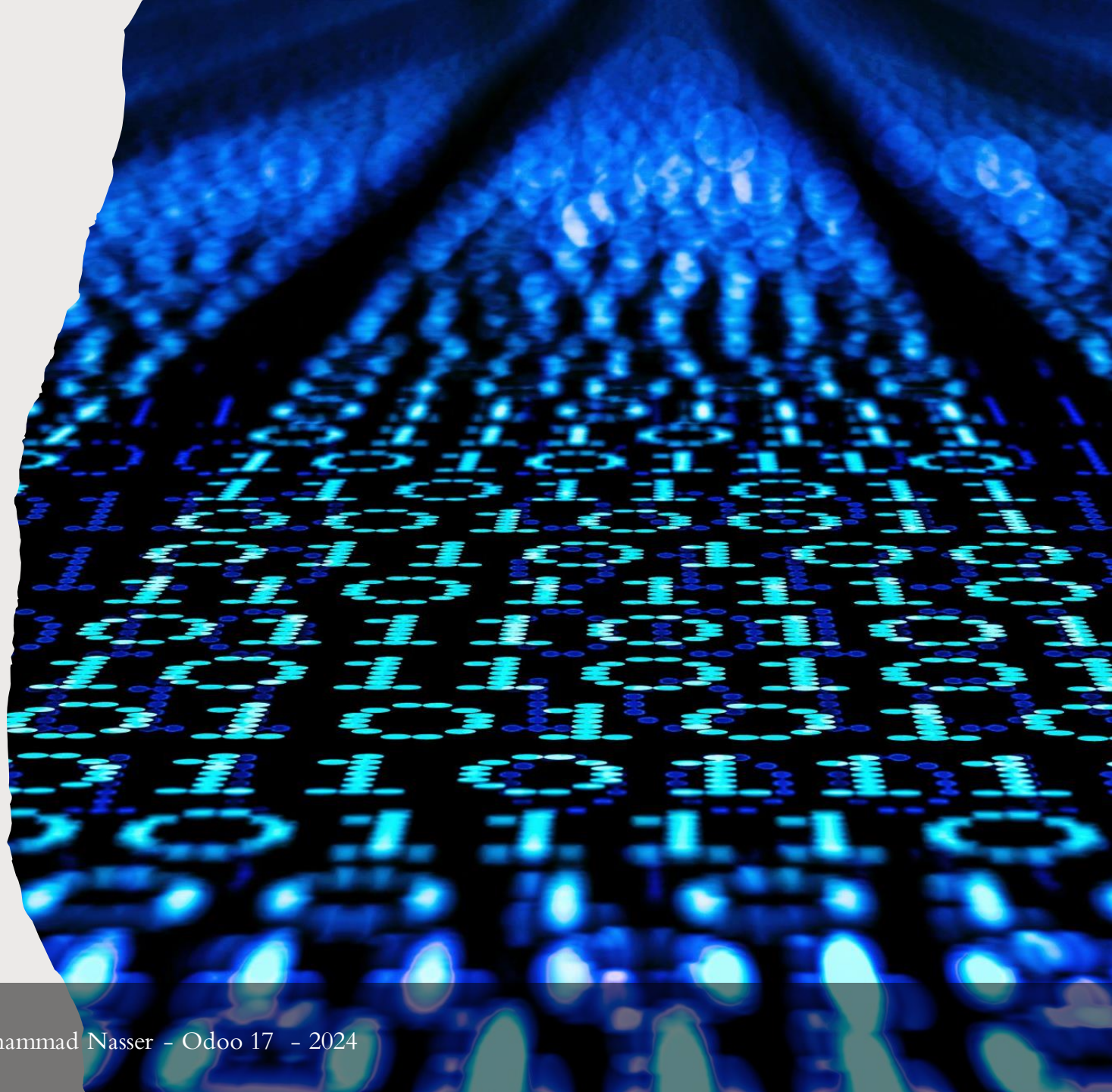
ODOO ARCHITECTURE:

Odoo follows a **multitier architecture**, meaning that the **presentation**, the business **logic** and the **data storage** are separated. More specifically, it uses a three-tier architecture.



BASICS:

- INTRODUCTION
- THE ODOO ARCHITECTURE
- COMPOSITION OF A MODULE



COMPOSITION OF A MODULE

An Odoo module **can** contain a number of elements:

- **Business objects**

  A business object (e.g. an invoice) is declared as a Python class. The fields defined in these classes are automatically mapped to database columns thanks to the ORM layer.

- **Object views**

  Define UI display.

- **Data files**

  XML or CSV files declaring the model data:

  (**views** or **reports**, configuration data (modules parametrization, **security rules**),demonstration data and more.)

- **Web controllers**

  Handle requests from web browsers.

- **Static web data**

  Images, CSS or JavaScript files used by the web interface or website.

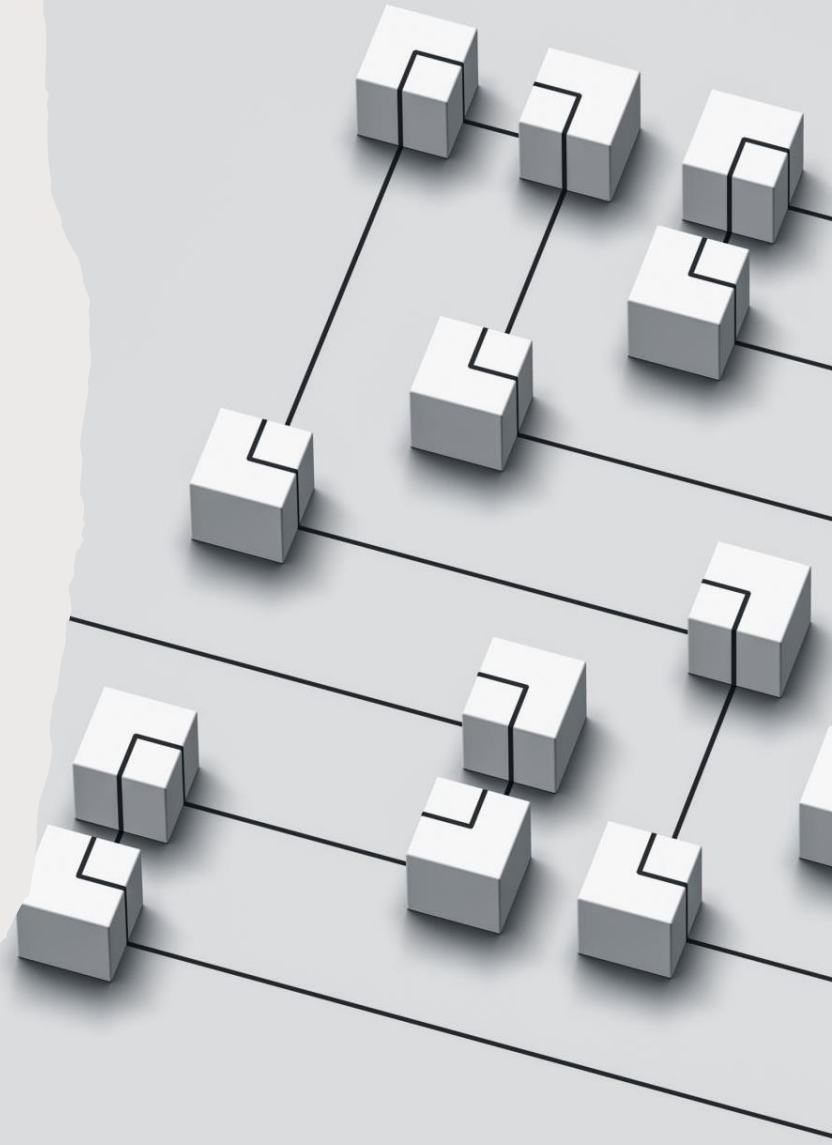
COMPOSITION OF A MODULE

Each module is a **directory** within a ***module directory***.

Module directories are specified by using the **--addons-path** option.

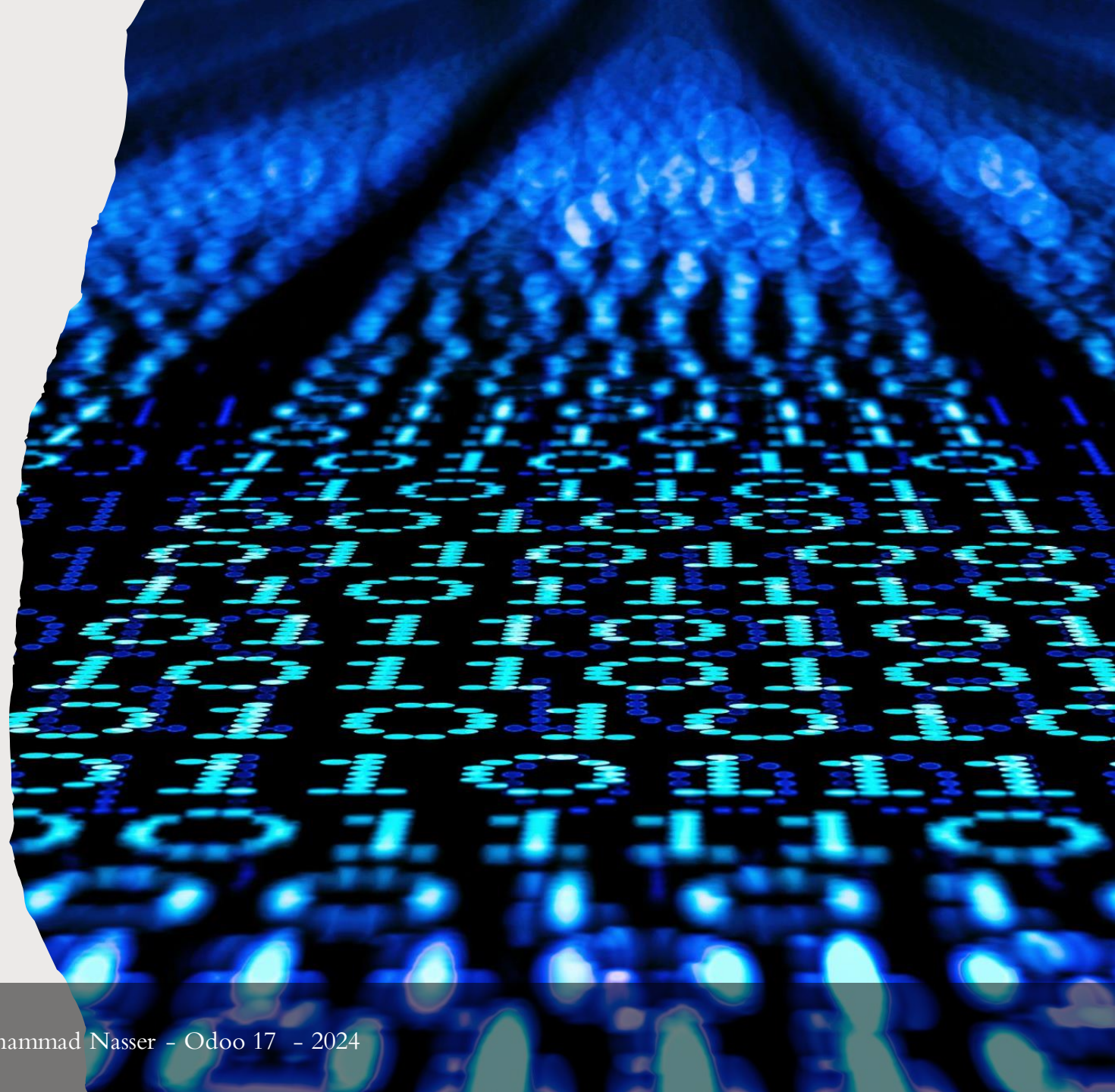
An Odoo module is declared by its **manifest**.

When an Odoo module includes business objects (i.e. Python files), they are organized as a **Python package** with a __init__.py file. This file contains import instructions for various Python files in the module.



BASICS:

- INTRODUCTION
- THE ODOO ARCHITECTURE
- COMPOSITION OF A MODULE
- MODULE STRUCTURE

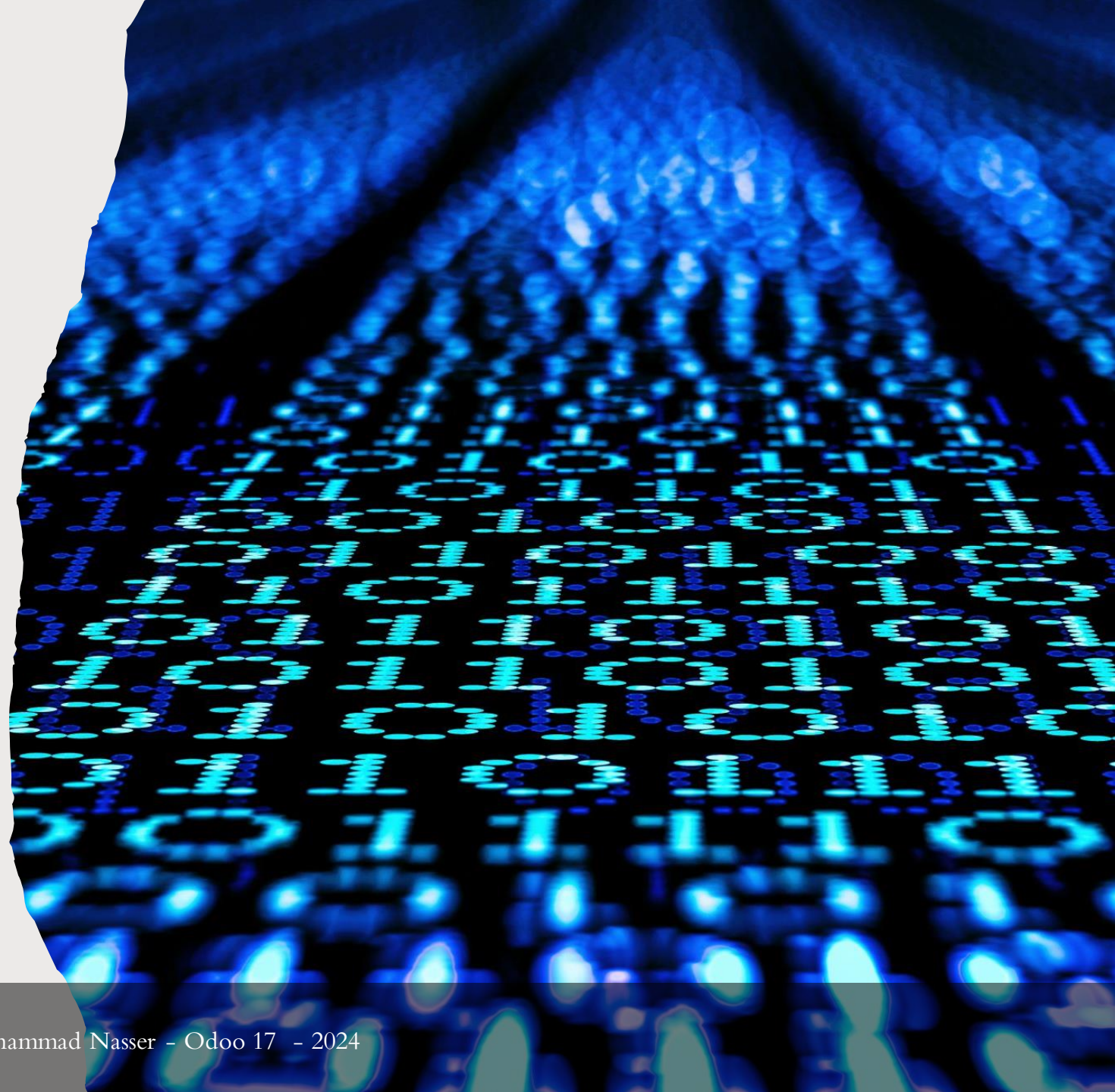


MODULE STRUCTURE

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

BASICS:

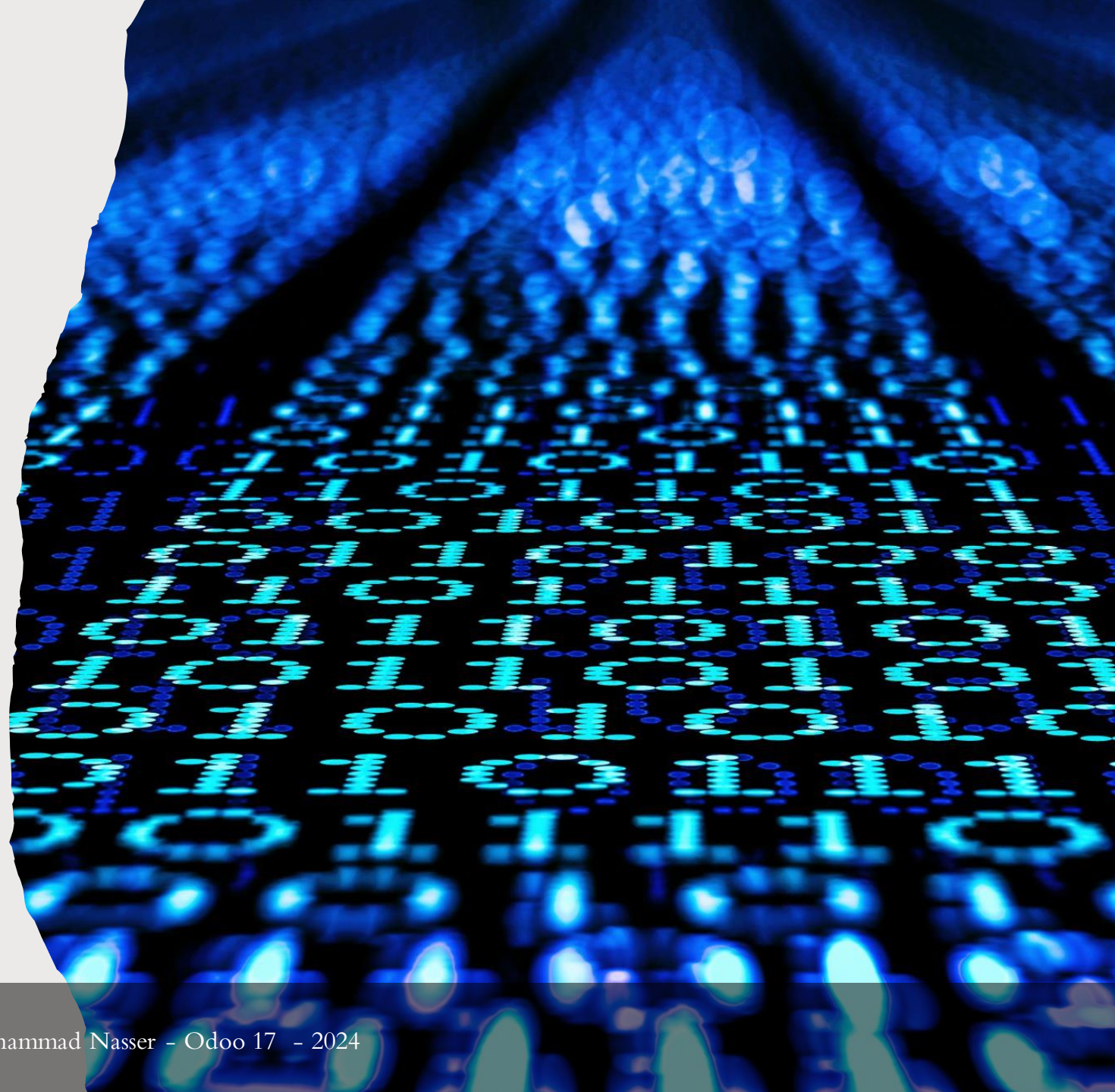
- INTRODUCTION
- THE ODOO ARCHITECTURE
- COMPOSITION OF A MODULE
- MODULE STRUCTURE
- INSTALL ODOO



1. git clone https://www.github.com/odoo/odoo --depth 1 --branch 17.0 odoo17
2. pip3 install wheel
3. pip3 install -r odoo17/requirements.txt

PROJECT:

- Basic Structure
-
-
-
-
-
-
-
-
-
-
-
-

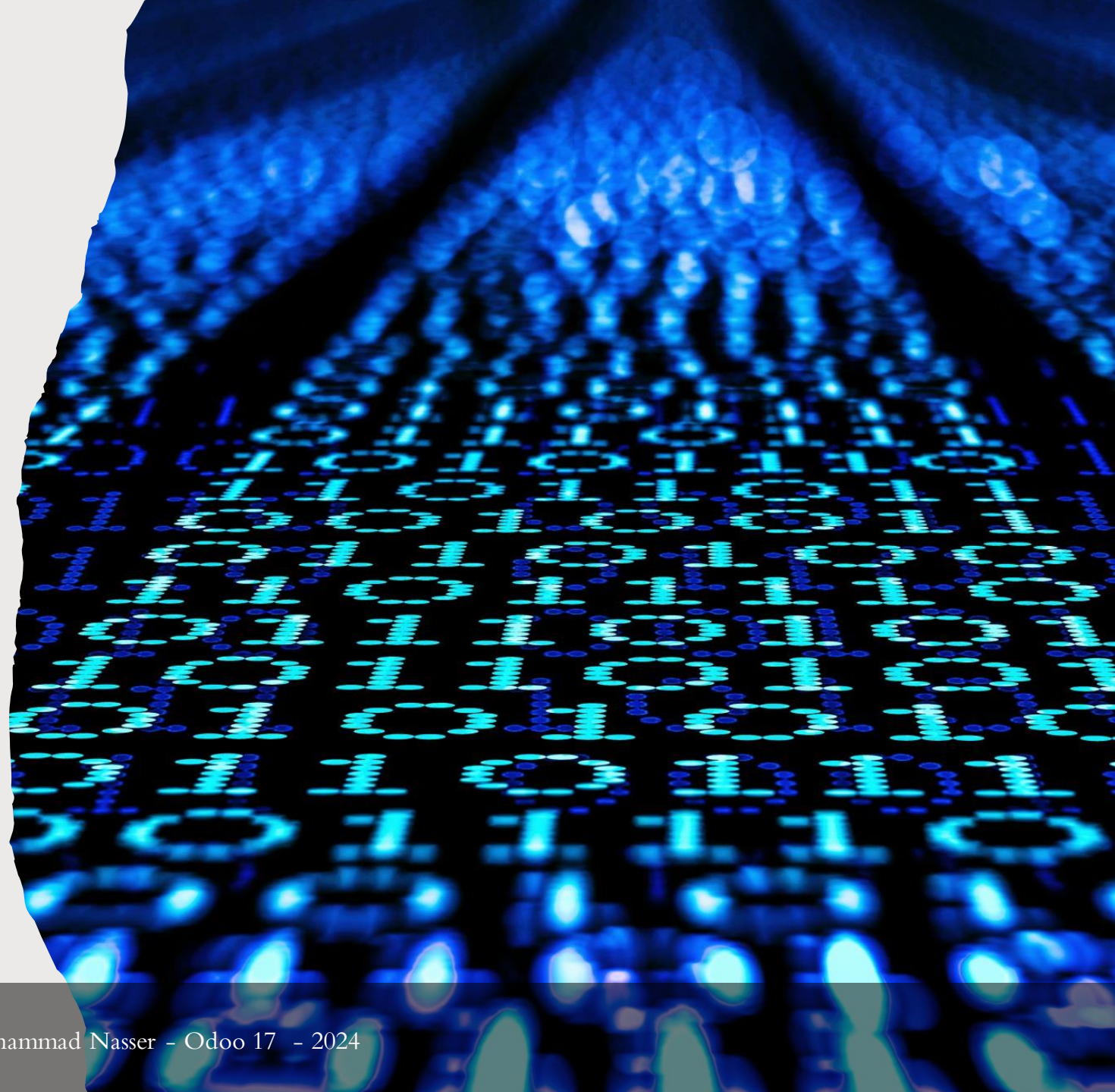


MAIN STRUCTURE

```
library_app/
├── __init__.py
├── __manifest__.py
├── controllers
│   ├── __init__.py
│   └── controllers.py
├── demo
│   └── demo.xml
├── models
│   ├── __init__.py
│   └── models.py
├── security
│   └── ir.model.access.csv
└── views
    ├── templates.xml
    └── views.xml
```

PROJECT:

- Basic Structure
- Basic Fields



BASIC FIELDS

Boolean[source]

Float[source]

Char[source]

Integer[source]

ADVANCED FIELDS

- Binary[source]
- Html[source]
- Image[source]
- Monetary[source]
- Selection[source]
- Text[source]

DATE(TIME) FIELDS

Date[source]

Datetime[source]

RELATIONAL FIELDS

Many2one[source]

One2many[source]

Many2many[source]

Command[source] (self-study)

PSEUDO-RELATIONAL FIELDS

Self-study

AUTOMATIC FIELDS

Id - Identifier **field**

ACCESS LOG FIELDS

These fields are automatically set and updated if **_log_access** is enabled. It can be disabled to avoid creating or updating those fields on tables for which they are not useful.

1.create_date

Stores when the record was created, **Datetime**

2.create_uid

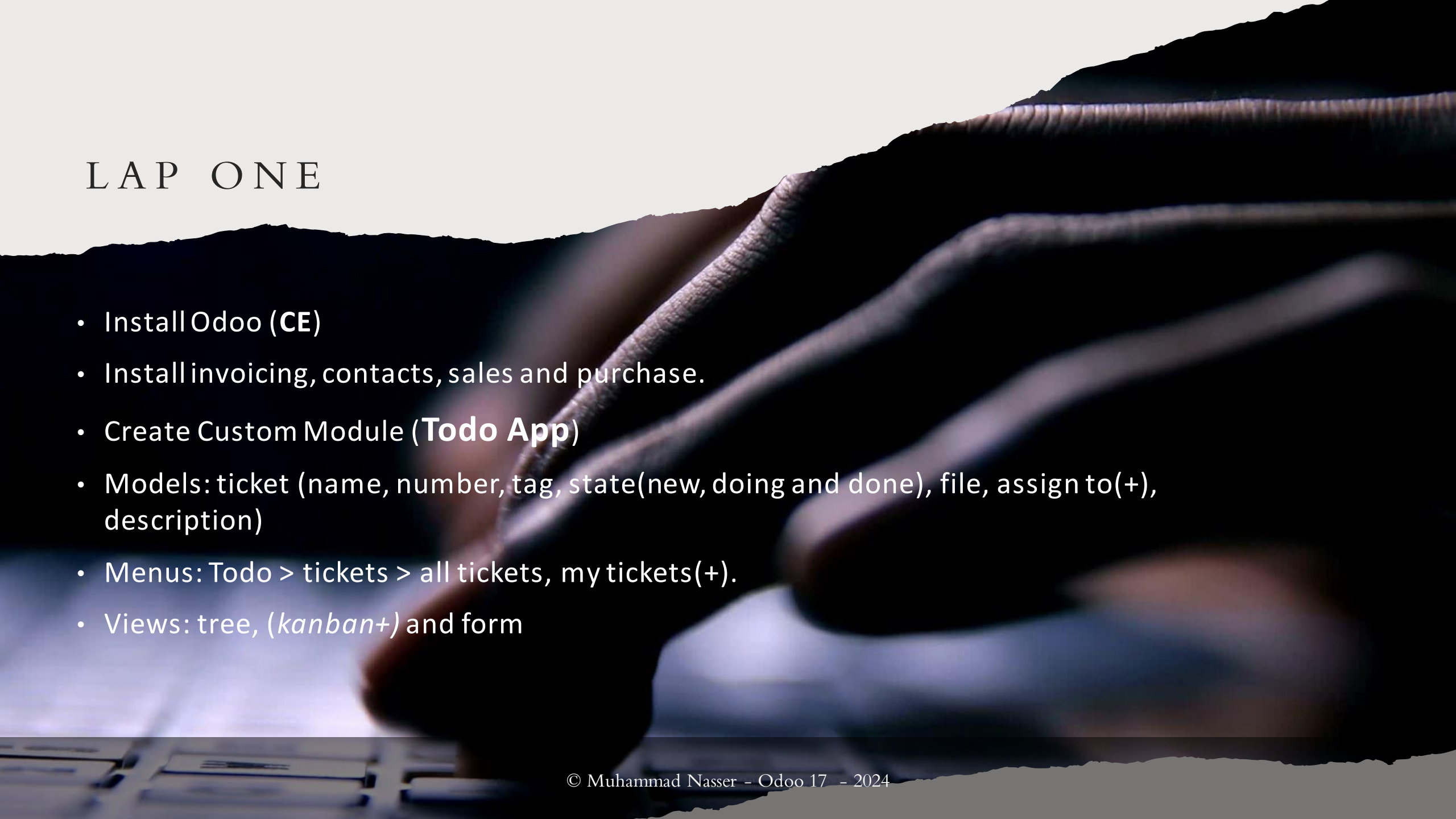
Stores *who* created the record, **Many2one** to a res.users.

3.write_date

Stores when the record was last updated, **Datetime**

4.write_uid

Stores who last updated the record, **Many2one** to a res.users.



LAP ONE

- Install Odoo (**CE**)
- Install invoicing, contacts, sales and purchase.
- Create Custom Module (**Todo App**)
- Models: ticket (name, number, tag, state(new, doing and done), file, assign to(+), description)
- Menus: Todo > tickets > all tickets, my tickets(+).
- Views: tree, (*kanban+)* and form