odoo Development

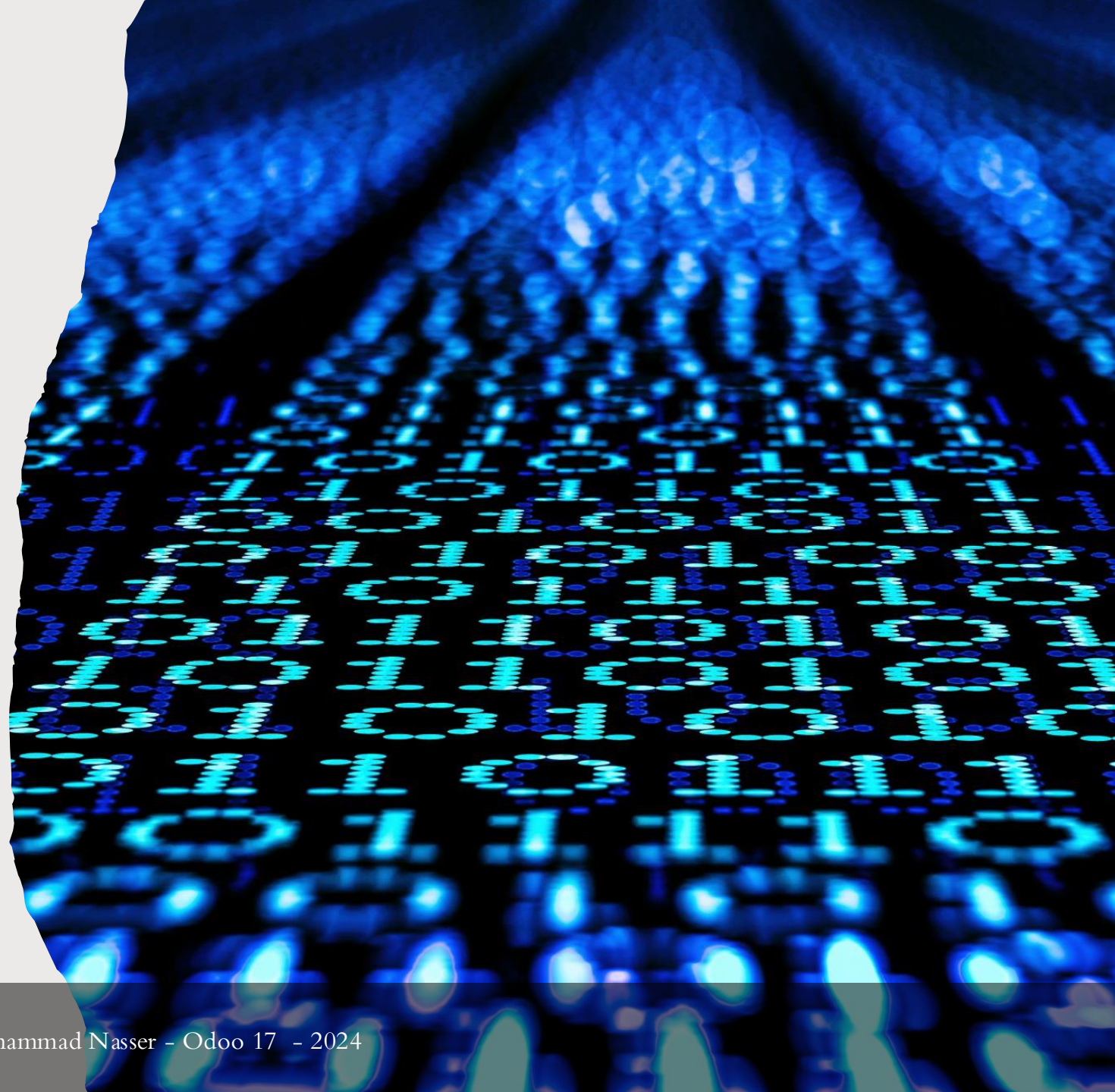
Muhammad Nasser

Information Technology Institute

Day 2

PROJECT:

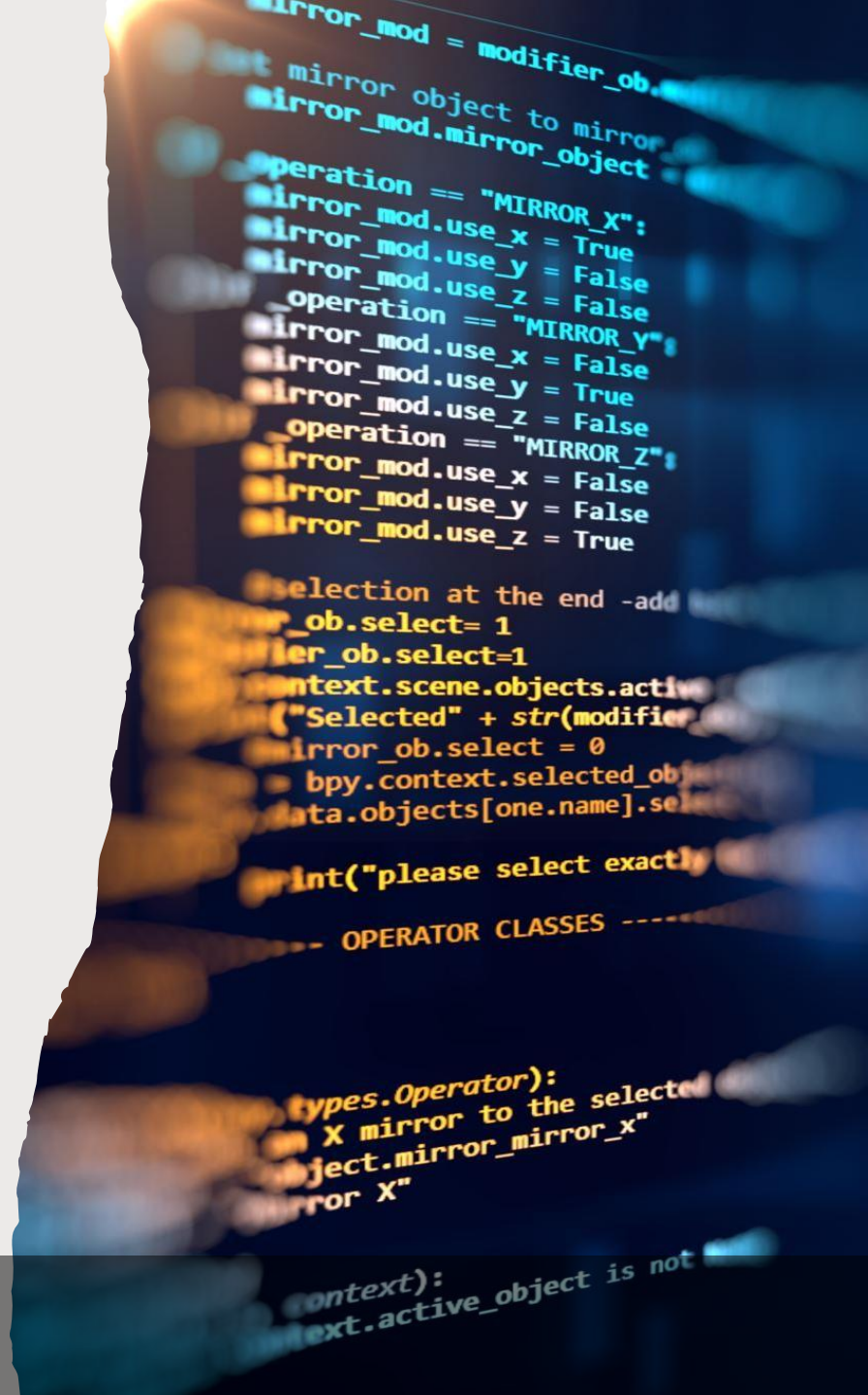
- Basic Structure
- Basic Fields
- DATA FILES



DATA FILES

Odoo is greatly data-driven, and a big part of modules definition is thus the definition of the various records it manages: UI (**menus** and **views**), security (**access rights** and **record rules**), **reports** and **plain data** are all defined via records.

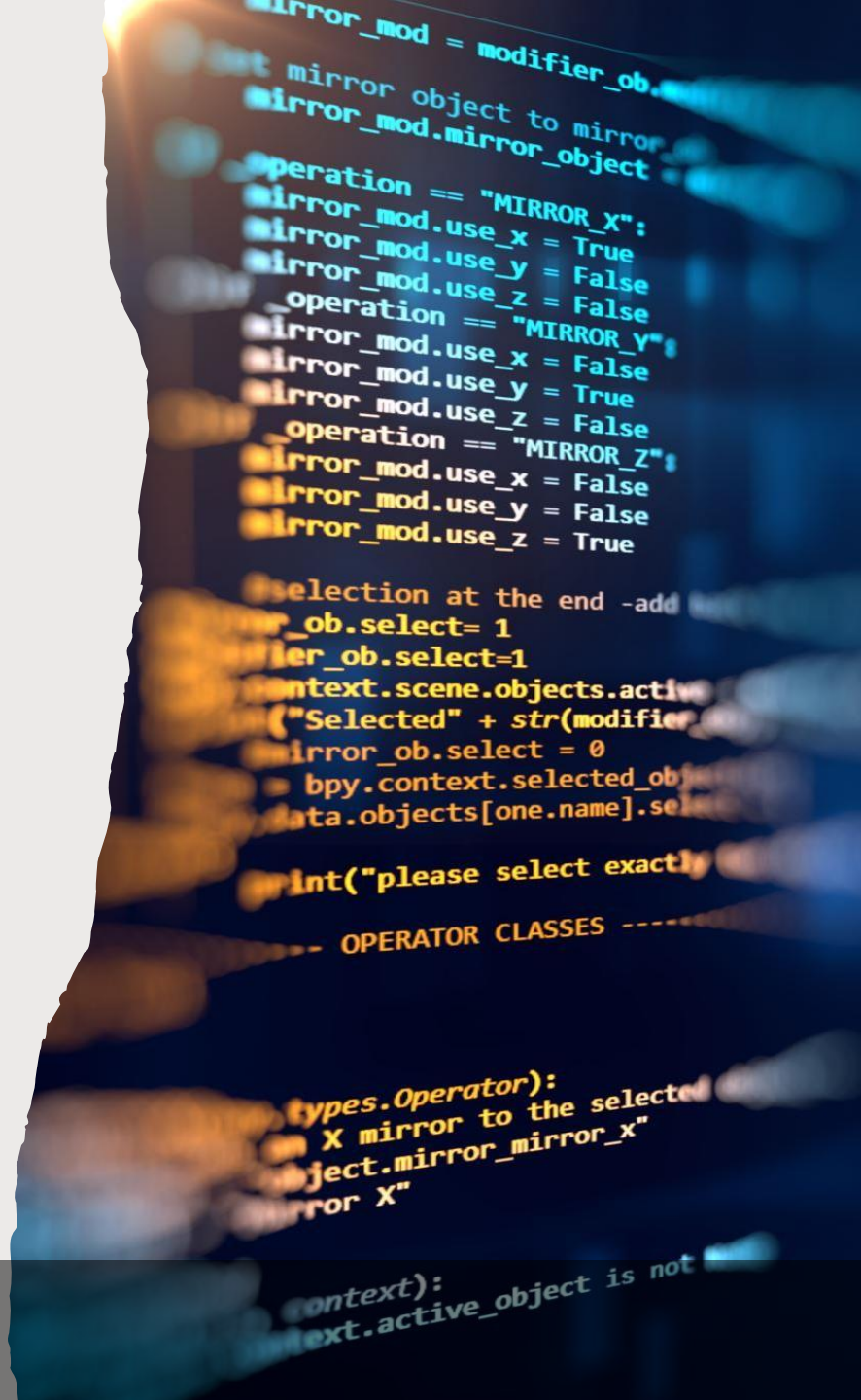
1. **Xml**
2. **Csv**



DATA FILES

Xml

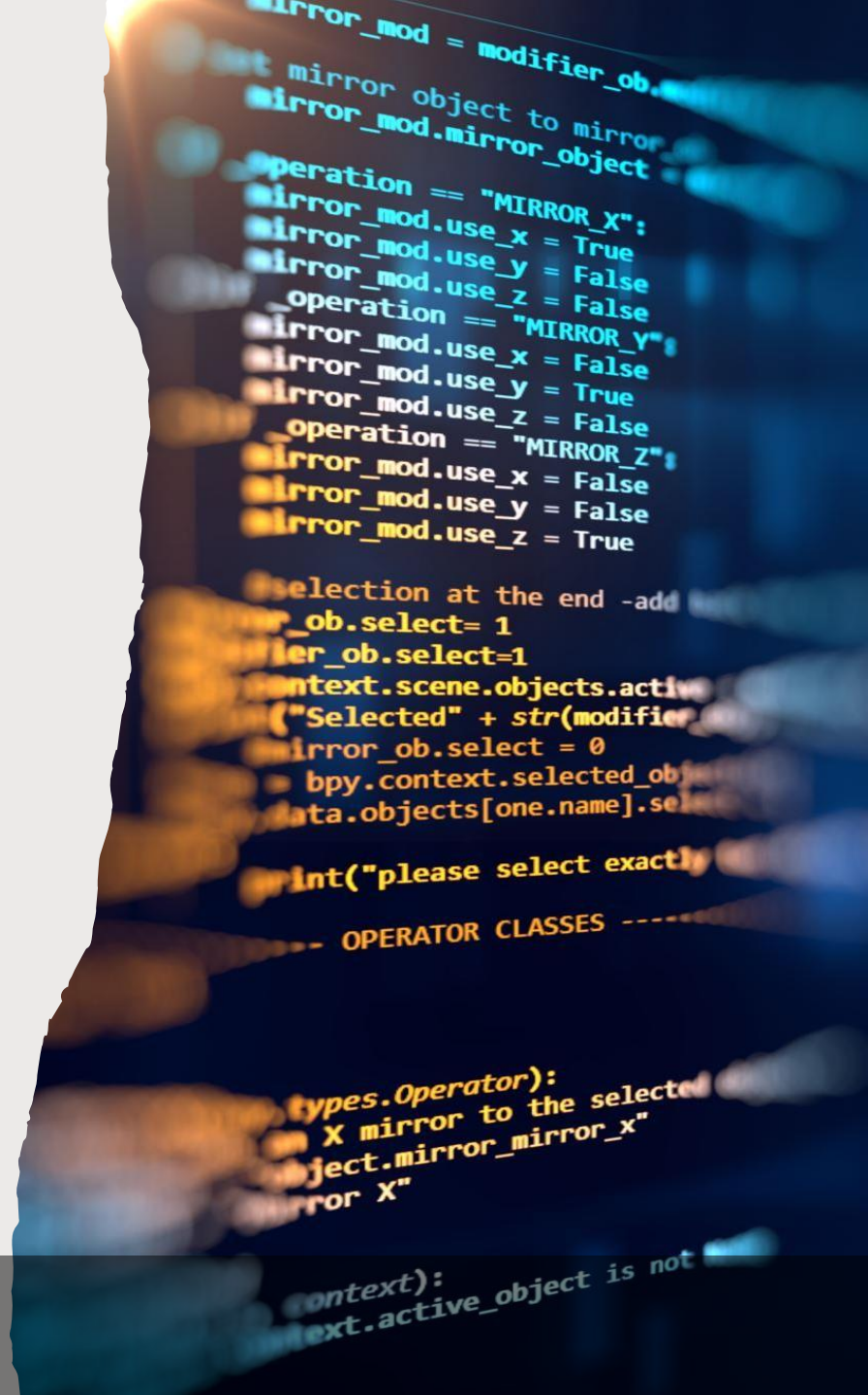
```xml
<odoo>

    <data noupdate="1">
        <!-- Only loaded when installing the module
                (odoo-bin -i module) -->
        <operation/>
    </data>

        <!-- (Re)Loaded at install and update
                (odoo-bin -i/-u) -->
    <operation/>

</odoo>
```



DATA FILES

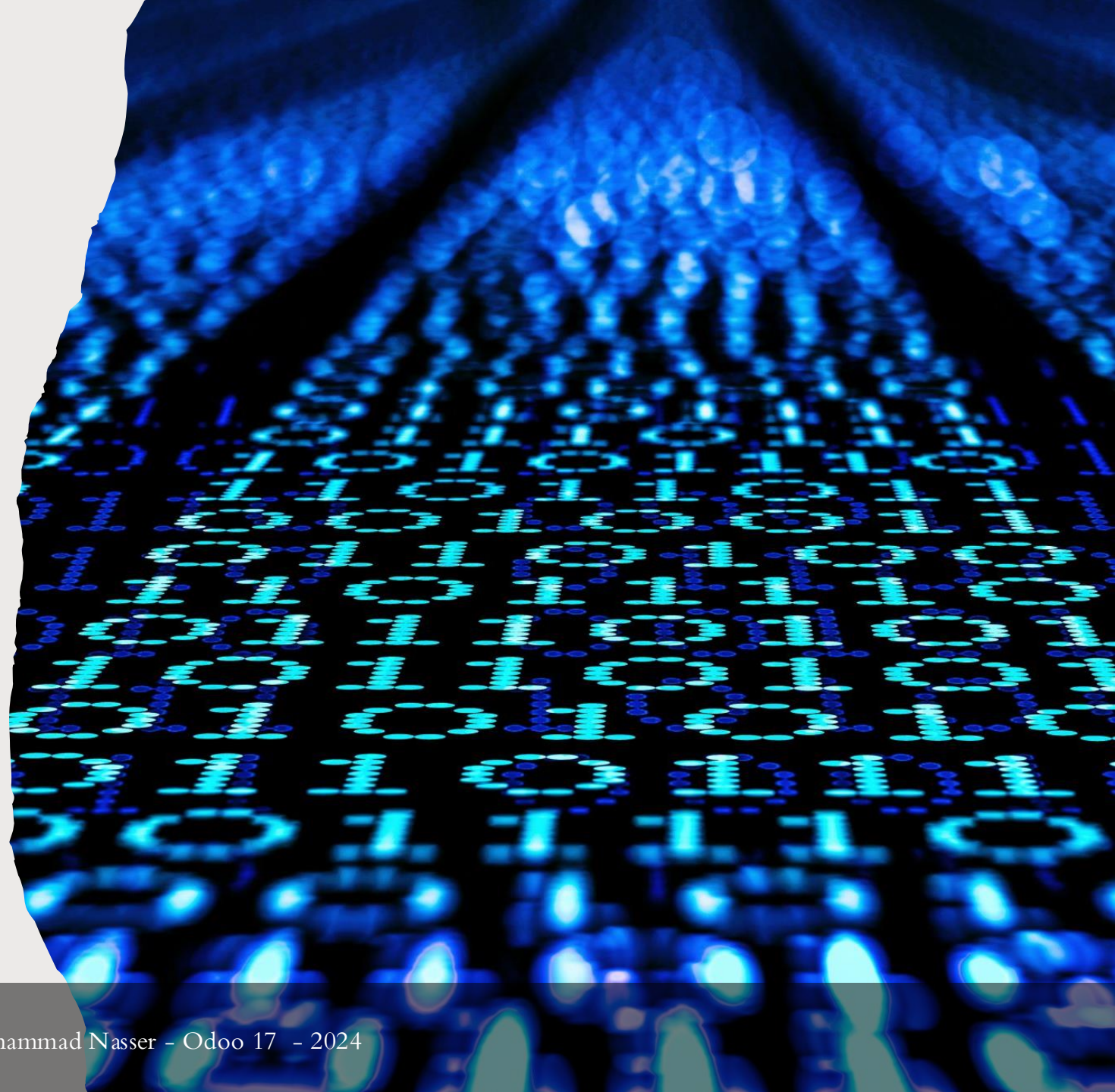
Csv

- The file name is *model_name.csv*
- The first row lists the fields to write, with the special field id for **external identifiers** (used for creation or update)
- Each row thereafter creates a new record



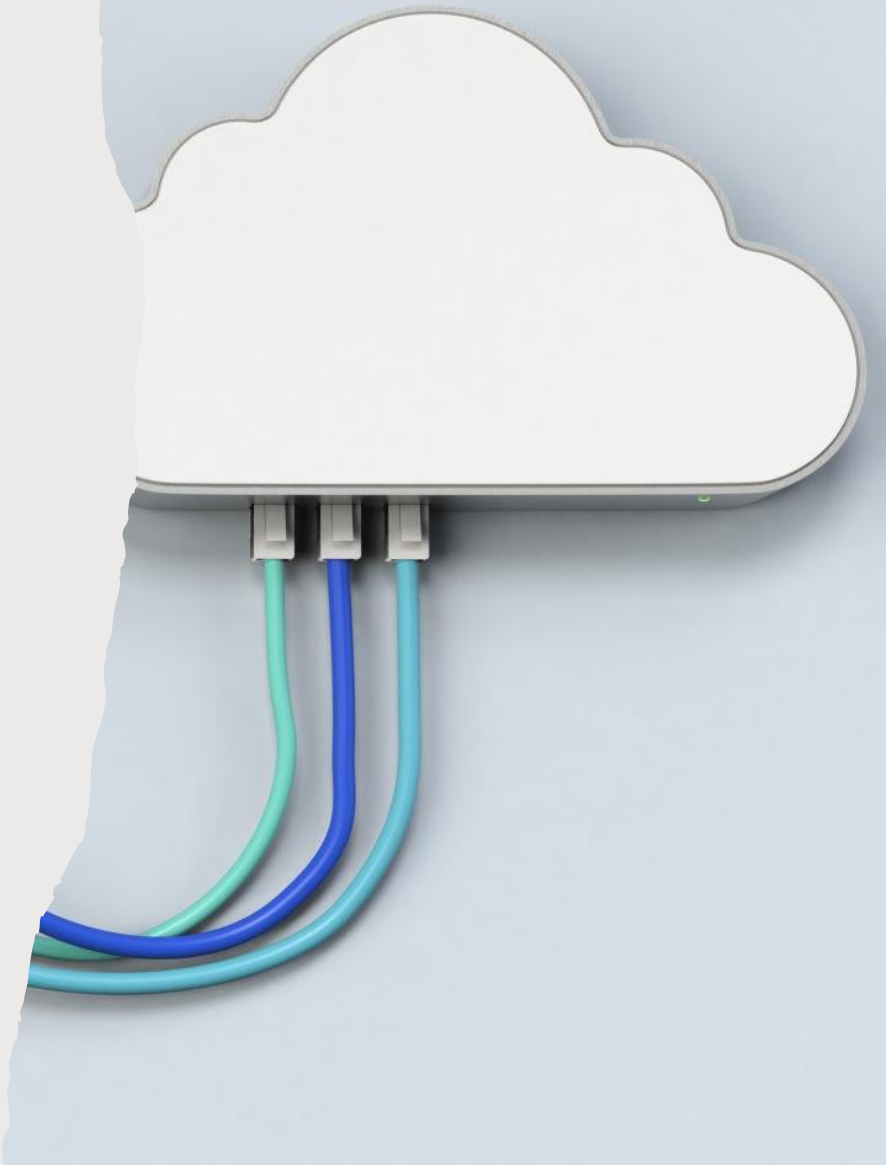
PROJECT:

- Basic Structure
- Basic Fields
- DATA FILES
- BASIC ACTIONS



ACTIONS

- Actions define the behaviour of the system in response to user actions.
- Actions can be stored in the database or returned directly as dictionaries in e.g. button methods.



Window Actions (ir.actions.act_window)

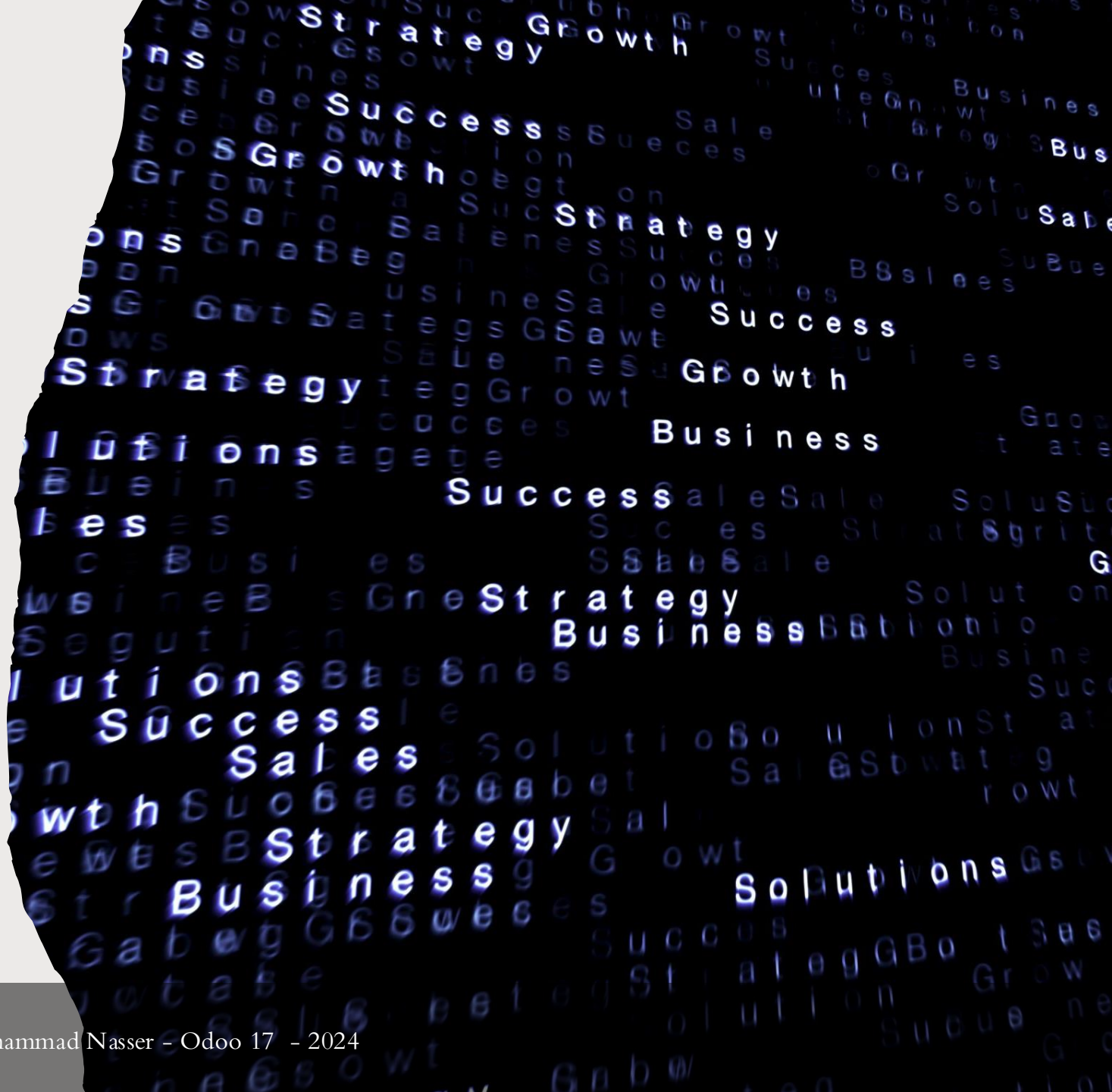
Server Actions (ir.actions.server)

Report Actions (ir.actions.report)

Automated Actions (ir.cron)

Client Actions (ir.actions.client)

Self-study

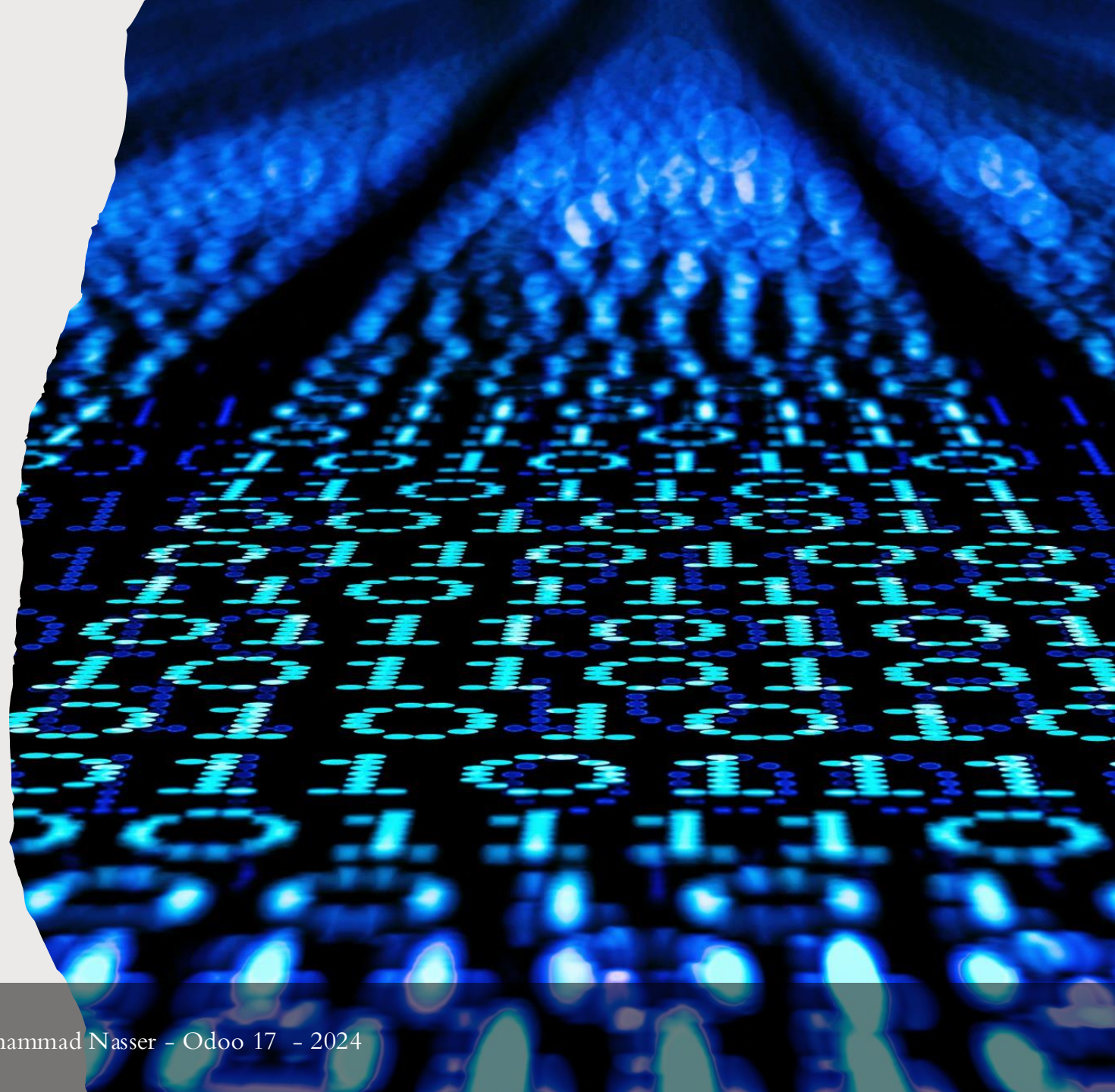


URL Actions (ir.actions.act_url)

Self-study

PROJECT:

- Basic Structure
- Basic Fields
- DATA FILES
- BASIC ACTIONS
- BASIC VIEWS



VIEWS

- Views are what define how records should be displayed to end-users. They are specified in XML which means that they can be edited independently from the models that they represent. They are flexible and allow a high level of customization of the screens that they control. There exist various types of views. Each of them represents a mode of visualization: *form*, *list*, *kanban*, etc.

GENERIC STRUCTURE

Basic views generally share the common structure defined below. Placeholders are denoted in all caps.

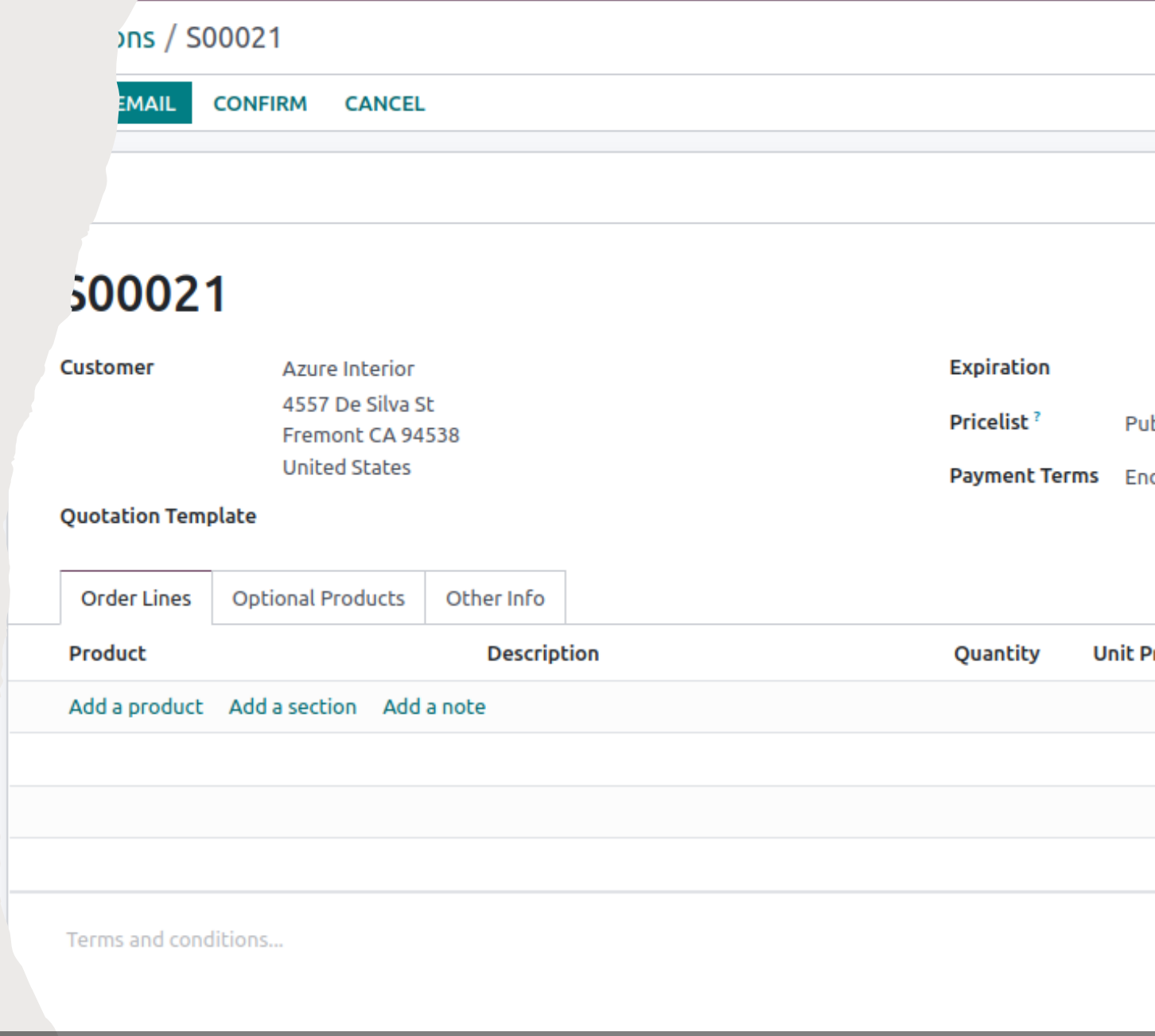
```
<record id="MODEL_view_TYPE" model="ir.ui.view">
  <field name="name">NAME</field>
  <field name="model">MODEL</field>
  <field name="arch" type="xml">
    <VIEW_TYPE>
      <VIEW_SPECIFICATIONS/>
    </VIEW_TYPE>
  </field>
</record>
```

FORM VIEW

Form views are used to display the data from a single record.

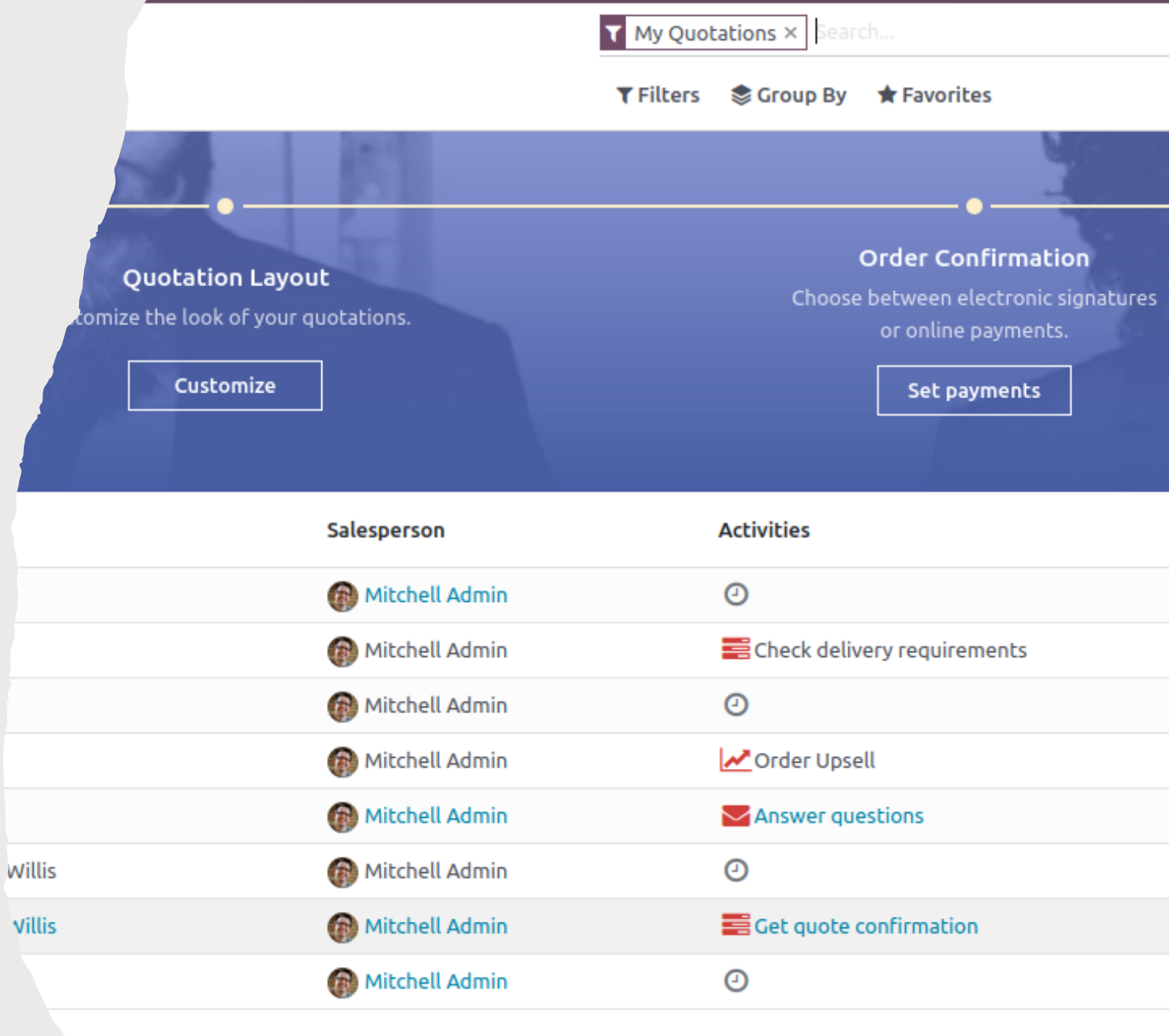
Their root element is **<form>**.

```
<form>
  <header>
    <field name="state" widget="statusbar"/>
  </header>
  <sheet>
    <div class="oe_button_box">
      <BUTTONS/>
    </div>
    <group>
      <group>
        <field name="fname"/>
      </group>
    </group>
    <notebook>
      <page string="Page1">
        <group>
          <CONTENT/>
        </group>
```



LIST VIEW

The root element of list views is **<tree>**



SEARCH VIEW

Search views are a break from previous view types in that they don't display *content*: although they apply to a specific model, they are used to filter other view's content (generally aggregated views e.g. **List** or **Graph**). Beyond that difference in use case, they are defined the same way.

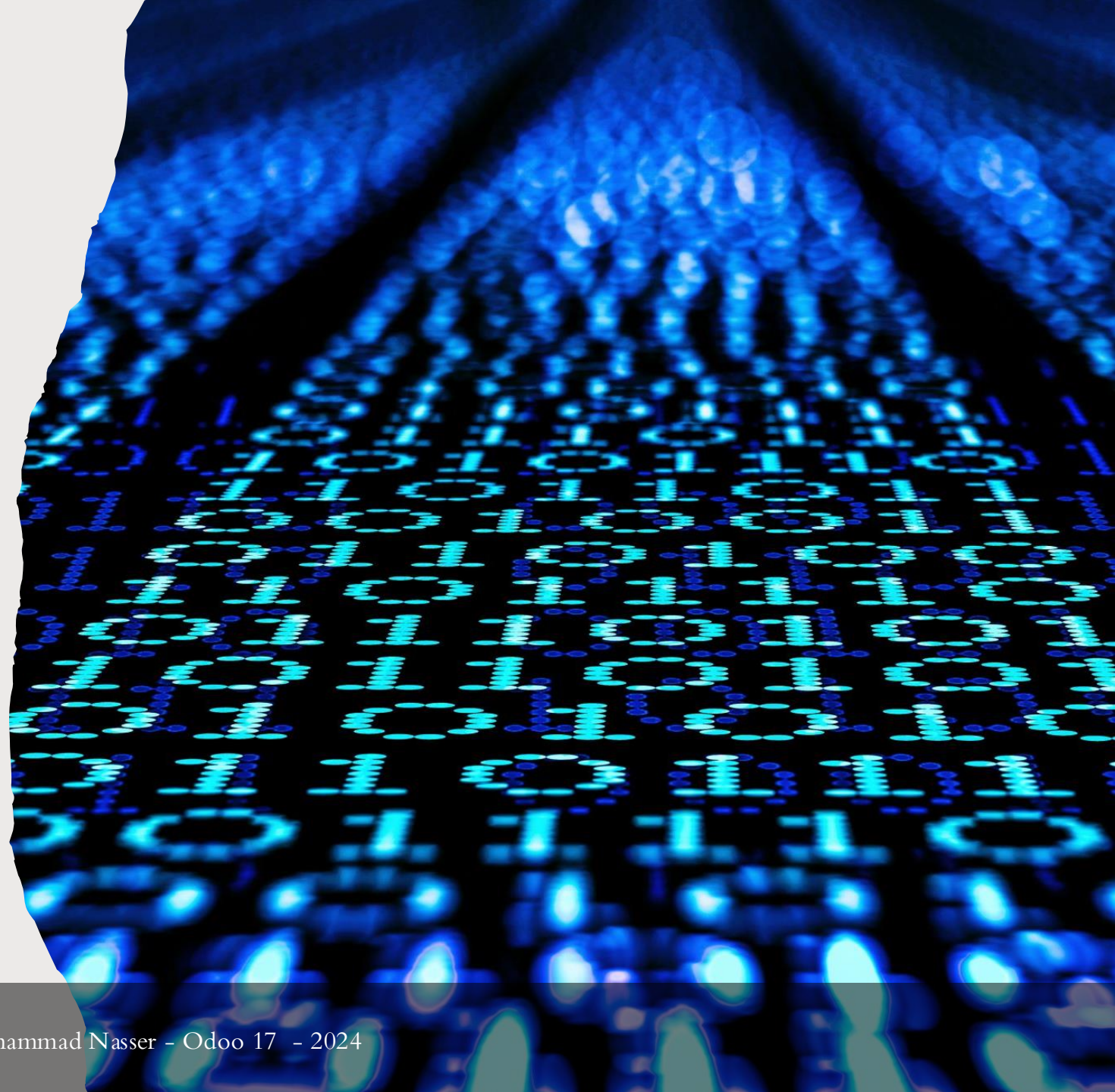
The root element of search views is **<search>**. It takes no attributes.

SELF-STUDY

The rest of views.

PROJECT:

- Basic Structure
- Basic Fields
- DATA FILES
- BASIC ACTIONS
- BASIC VIEWS
- BASIC MODELS



MODEL

***class* odoo.models.Model**

Main super-class for regular database-persisted Odoo models.

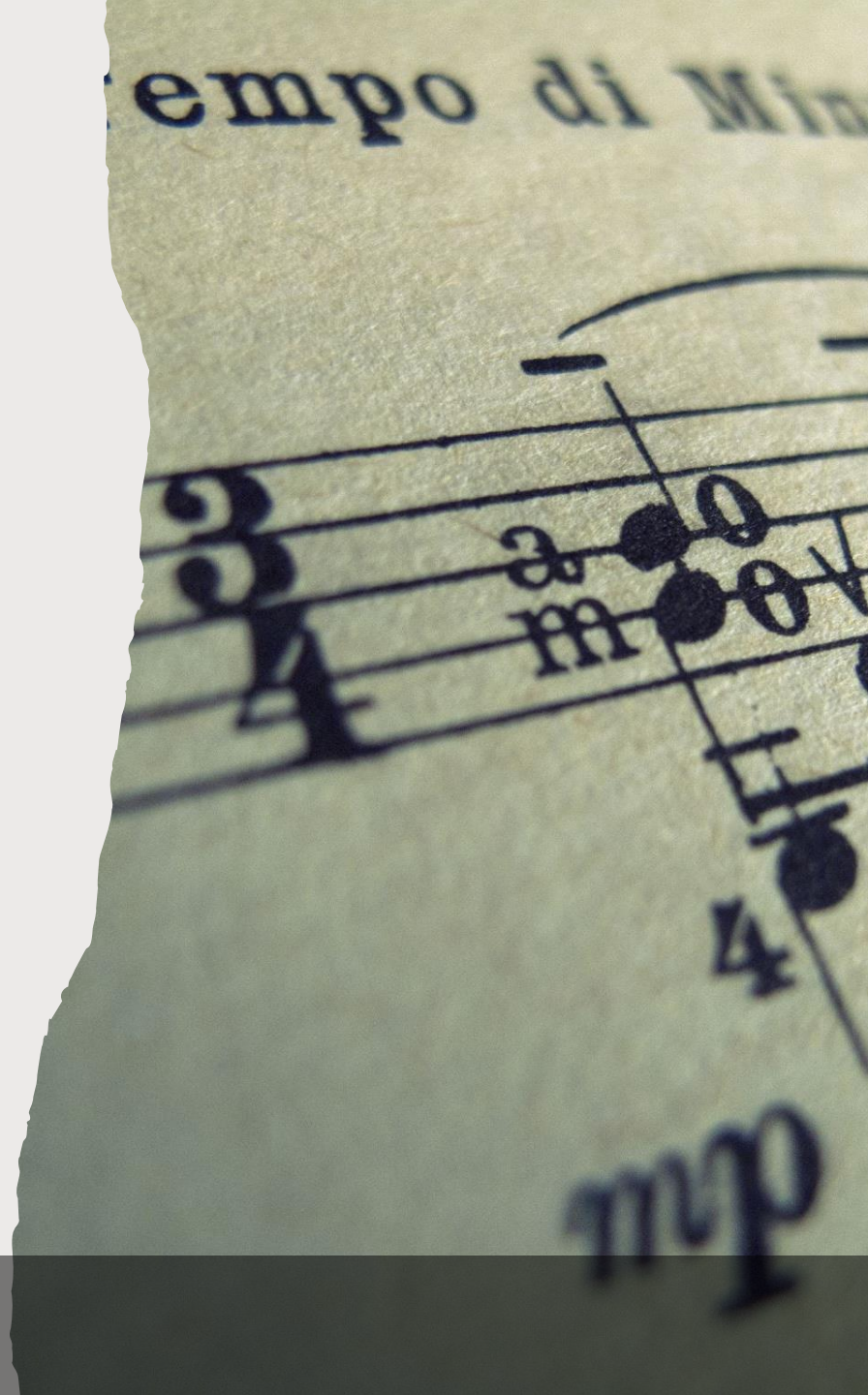
Odoo models are created by inheriting from this class.

TRANSIENT MODEL

***class* odoo.models.TransientModel**

Model super-class for transient records, meant to be temporarily persistent, and regularly vacuum-cleaned.

A TransientModel has a simplified access rights management, all users can create new records, and may only access the records they created. The superuser has unrestricted access to all TransientModel records.

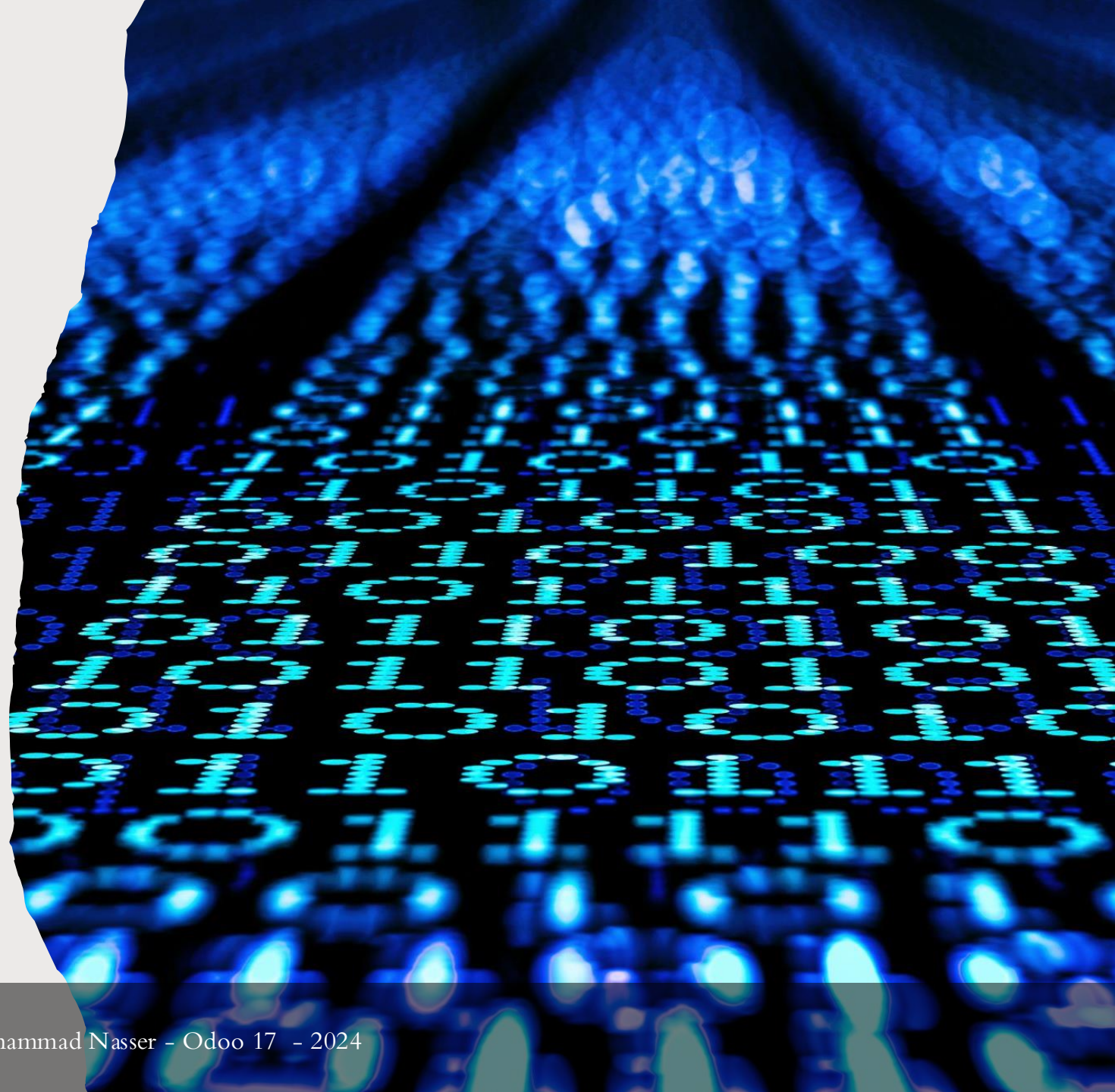


ABSTRACT MODEL

Self-study

PROJECT:

- Basic Structure
- Basic Fields
- DATA FILES
- BASIC ACTIONS
- BASIC VIEWS
- BASIC MODELS
- Method decorators



METHODS DECORATORS

Odoo.api.constrains(*args)

Decorate a constraint checker.

Each argument must be a field name used in the check.

METHODS DECORATORS

odoo.api.depends(args*)**

Return a decorator that specifies the field dependencies of a “compute” method (for new-style function fields).

Each argument must be a string that consists in a dot-separated sequence of field names.

One may also pass a single function as argument. In that case, the dependencies are given by calling the function with the field’s model.

METHODS DECORATORS

odoo.api.onchange(args*)**

In the form views where the field appears, the method will be called when one of the given fields is modified. The method is invoked on a pseudo-record that contains the values present in the form.

Each argument must be a field name.

SELF-STUDY

odoo.api.ondelete(*, *at_uninstall*)

odoo.api.model_create_multi(*method*)

odoo.api.depends_context(args*)**

odoo.api.autovacuum(*method*)

LAP TWO

Hospital Management System (lap 1 and 2)