DIAGRAM SOFTWARE

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

PROGRAMACIÓN EN OPENERP/ODOO

0. ARQUITECTURA Y MÉTODO DE TRABAJO

José Zambudio Bernabeu
jose.zambudio@diagram.es

NubeaERP global cloud solutions

DIAGRAM SOFTWARE

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ÍNDICE

- Arquitectura
  - Multi-tenant
  - Three-tiers architecture
- Método de trabajo
  - Git
    - Flujo de trabajo
    - Documentación
    - Nuestro repositorio.
  - Nuestro entorno de trabajo
    - OpenERP
    - Ejecución + Comandos
    - Nuestro IDE - Sublime Text 2 (3)

ARQUITECTURA

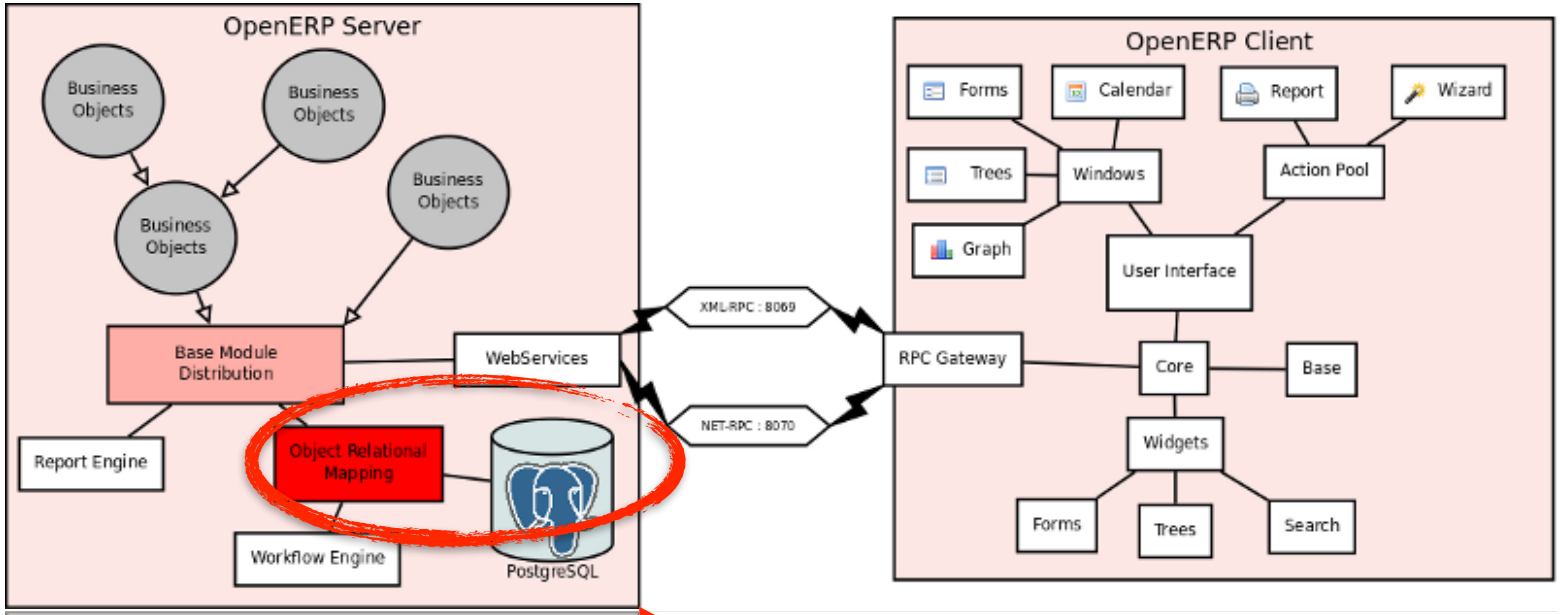
MULTI-TENANT

- Tenencia Múltiple…
- Las arquitecturas Multi-Tenant se refieren a un principio en la que una única instancia del software se ejecuta en un servidor para ser utilizada por diferentes usuarios / clientes / empresas.

THREE-TIERS ARCHITECTURE

- Es una arquitectura cliente-servidor en el que la lógica de proceso funcional, el acceso / almacenamiento de datos y la interfaz del usuario se desarrollan y se mantienen como módulos independientes.
  1. PostgreSQL
  2. OpenERP Server
  3. Clients

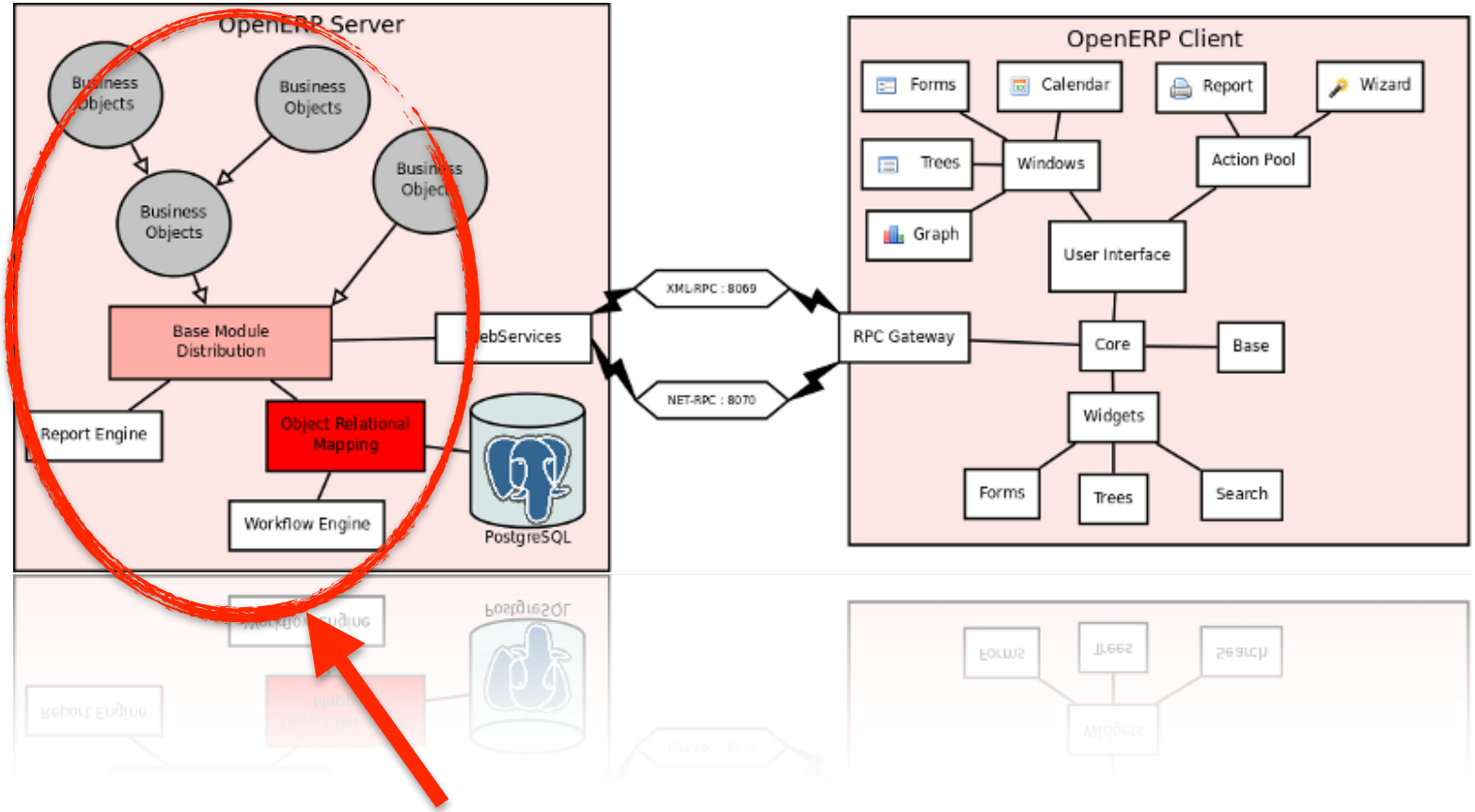
THREE-TIERS ARCHITECTURE



POSTGRESQL

- Nivel de datos.
- Gestor de base de datos
- Contiene toda la información de la aplicación y la mayor parte de configuración de OpenERP.
- Posibilidad de utilizar clusters
- Posibilidad de utilizar queries desde OpenERP…
- … aún así se recomienda siempre utilizar el ORM (Object Relational Mapping) que aporta OpenERP

THREE-TIERS ARCHITECTURE



OpenERP Server

- Nivel de lógica de negocio.
- Completo framework que entre otros, nos proporciona:
  - Server - ORM
  - Server - Web
  - Módulos

OpenERP Server

- Server - ORM
  - En python crearemos clases llamados modelos de los cuales el ORM gestionará las tablas de la base de datos.
  - Potente validación de datos
  - Proporciona una interfaz de objetos (fields, methods, references…) lo que nos facilita la implementación de módulos eficientes.
  - ACL's por usuarios/grupos
  - Varias formas de heredar, facilitando la creación/ modificación de los modelos.

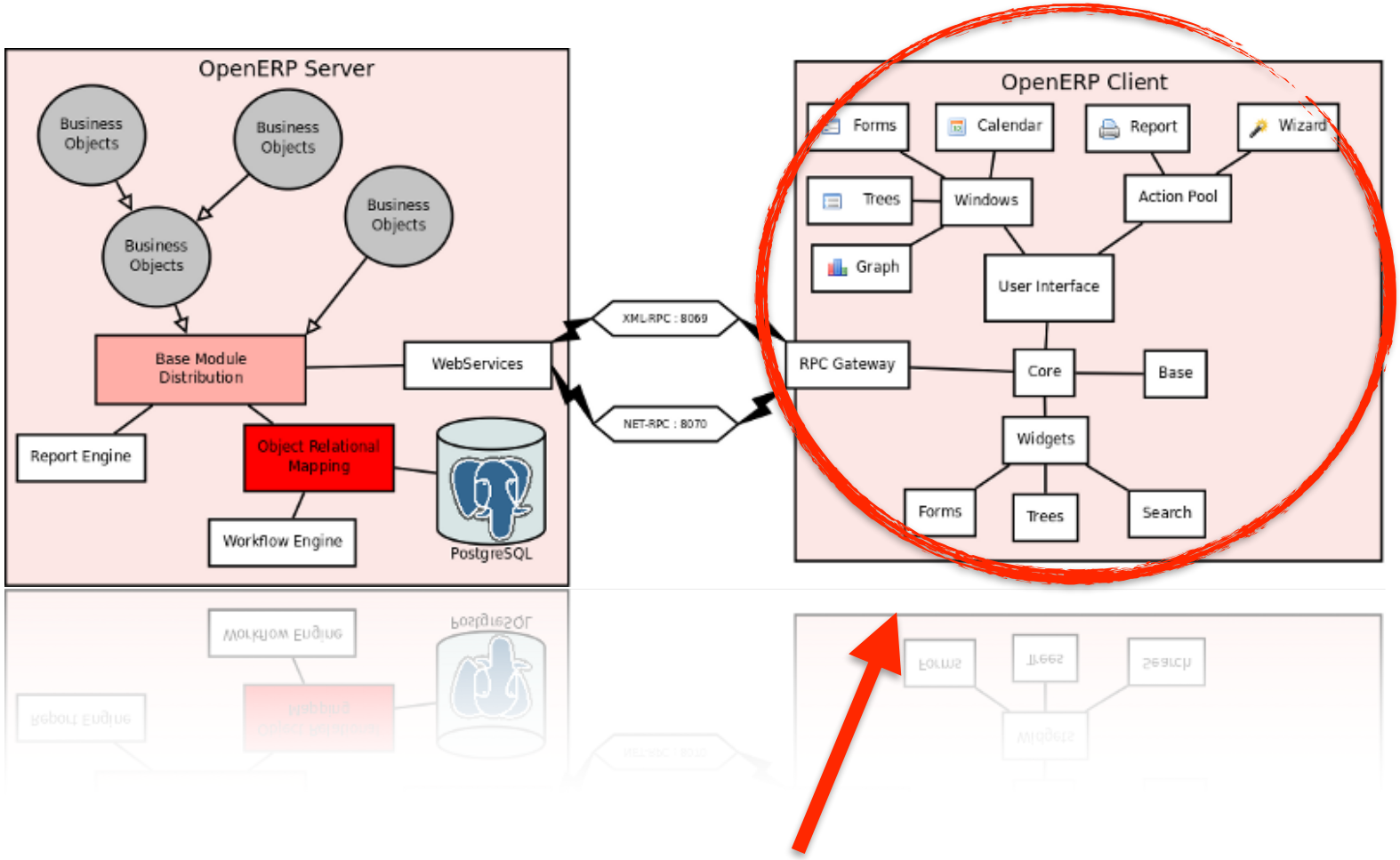
OPENERP SERVER

- Server - Web
  - Esta capa nos proporciona una comunicación con los navegadores.
  - Desde la versión 6.1 esta capa se re-escribió en el nivel de aplicación.
  - Aplicación WSGI basado en Werkzeug.
    - Interfaz para comunicar el servidor web con la aplicación. Trata peticiones HTTP o JSON-RPC queries, del navegador web.
    - En caso de utilizar otro tipo de cliente no web, OpenERP recomienda el uso de XML-RPC ya que mantiene el nivel de seguridad.

OPENERP SERVER

- Modulos
  - OpenERP nos facilita un core, al que podemos añadir cualquier necesidad de negocio mediante la instalación de módulos.
  - Cualquier versión de OpenERP nos facilita un gran listado de módulos, y hay muchos más disponibles creados por la comunidad.

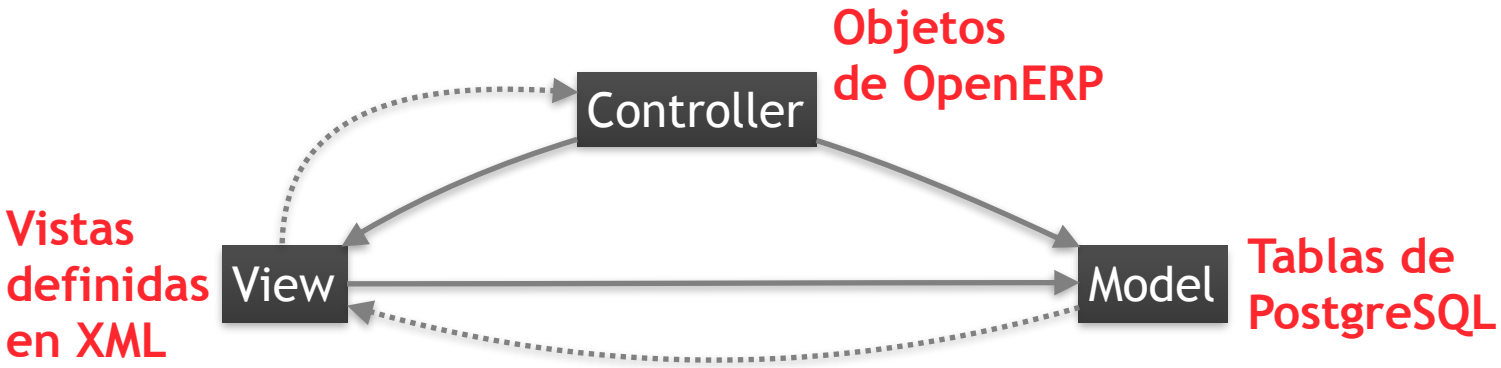
THREE-TIERS ARCHITECTURE



CLIENTE

- Nivel de presentación.
- El cliente ejecuta una aplicación javascript en su navegador web que se comunica con el servidor mediante llamadas JSON-RPC.
- El funcionamiento es simple… el navegador realiza peticiones al servidor, de las que obtiene datos y muestra dichos resultados de diferentes formas (listas, formularios, calendarios…)

MVC



- El controller tiene completo acceso tanto a la vista como al modelo.
- La vista tiene completo acceso al modelo, pero limitado al controlador ya que dichas dependencias deben de ser mínimas, el controller está sujeto a muchos cambios.
- El modelo notifica a la vista que los datos se han modificado para que la vista los actualize.

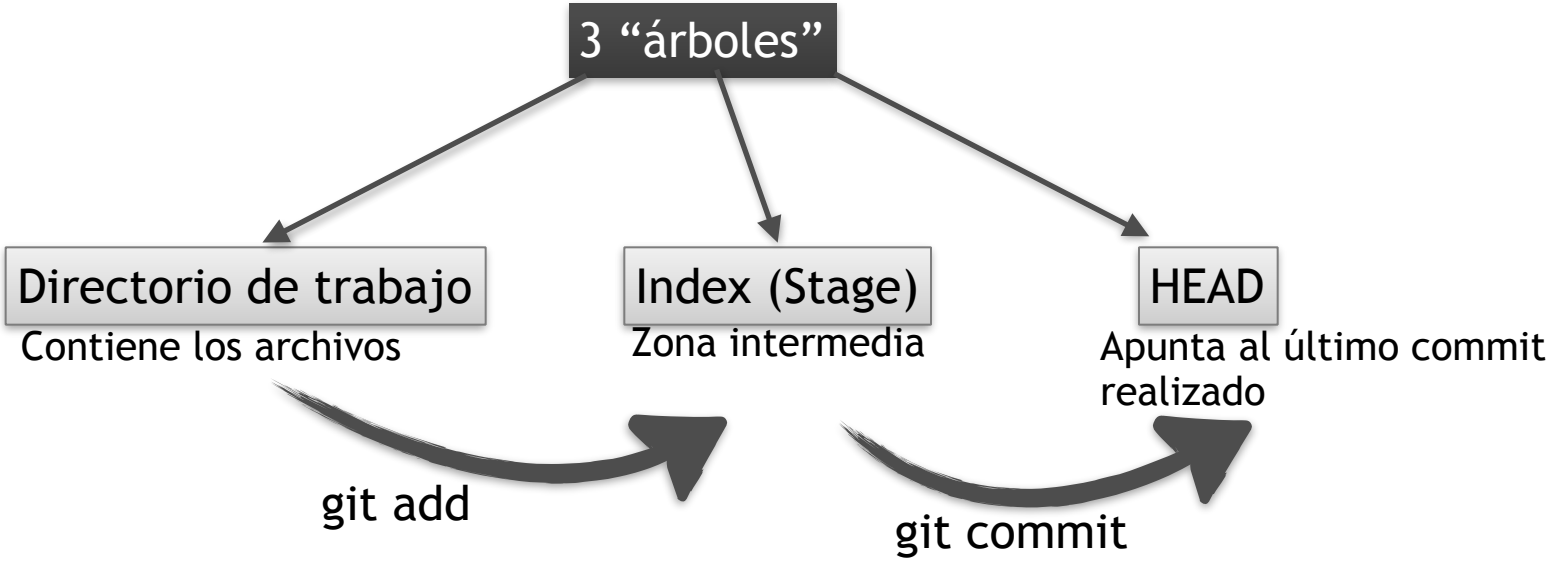
MÉTODO DE TRABAJO

ENTORNO DE TRABAJO

- **Repository**: Git
- **Framework**: OpenERP
  - Ejecución (desarrollo… **NO** para producción)
  - Command line options
  - Configuration file: ~/.openerp_serverrc
- **IDE**: Sublime Text 2 (3)
- **Debugger**: PDB
- **Traducciones**: Poedit
- **Visual diff**: Meld
- **Documentación**: Memento

GIT

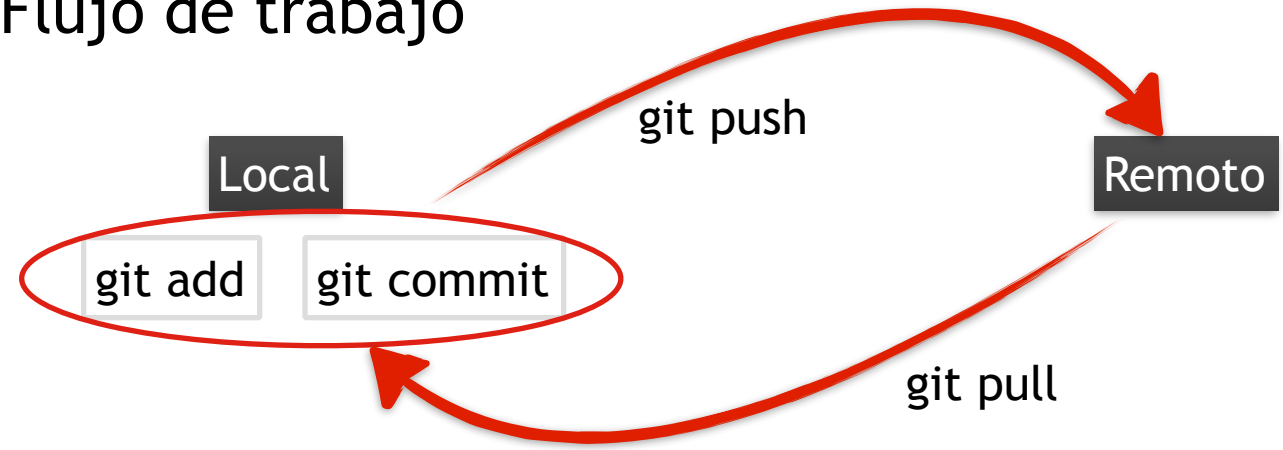
- Flujo de trabajo



- git add <filename> (git add . -> Añadir todo)
  - Registramos cambios realizados.
- git commit -m “Commit message”
  - Añadimos dichos cambios junto con un mensaje.

GIT

- Flujo de trabajo



- git (push | pull) <remote> <branch>
  - remote: Servidor remoto.
  - Concepto branch - Ramas.
- Documentación

GITHUB

- Nuestro repositorio

  `https://github.com/zamberjo/curso_odoo_7`

- Descargarnos contenido
  - git clone https://github.com/zamberjo/curso_odoo_7.git
- Actualizamos contenido
  - git pull origin --all

- Contenido:
  - doc/ : Estas transparencias + documentación a parte.
  - modules/ : Todos los módulos que vamos a ir haciendo.