Implantació i adaptació digital d'un centre educatiu mitjançant Odoo

Carlos López Arroyo

Resum— Aquest projecte consta de la implantació digital d'un software de gestió (ERP) anomenat Odoo a un Centre educatiu. Tot el cicle vital del mateix ha estat planificat i elaborat en base a la pròpia experiència amb el sector, a més de la presa de requeriments funcionals i no funcionals d'un centre real. Les possibilitats que ofereix el programa donen abast per a totes les classes i departaments amb el que funciona una escola, desde infantil fins a la secundària. Algunes de les funcionalitats principals són: la capacitat de crear usuaris de diferents perfils segons el grau de responsabilitat al centre, la creació d'anys acadèmics, cursos, classes, assignatures, alumnes, etcètera. Tots aquests registres es relacionen mitjançant una estructura de classes documentada i contrastada amb el context real educatiu. Pel que fa a l'estructura interna del programa, aquesta està constituïda per un fitxer docker-compose que crea imatges per a cada una de les parts que es necessiten al desplegar aquest programa al núvol (db-proxy, servidor on establir l'entorn, APIs…). Finalment, penso que podem estar davant d'una iniciativa que faci evolucionar l'operativa interna de les escoles a Catalunya.

Paraules clau—ERP, cicle vital, programa, centre educatiu, anys acadèmics, cursos, classes, assignatures, alumnes, docker-compose, iniciativa.

Abstract—This project consists of the digital implementation of a management software (ERP) called Odoo in an educational center. The entire life cycle of the same has been planned and elaborated on the basis of my own experience with the sector, in addition to the taking of functional and non-functional requirements of a real center. The possibilities offered by the program cover all the classes and departments with which a school operates, from kindergarten to secondary school. Some of the main functionalities are: the ability to create users of different profiles according to the degree of responsibility in the center, the creation of academic years, courses, classes, subjects, students, etc. All these records are related through a documented class structure and contrasted with the real educational context. As for the internal structure of the program, it is made up of a docker-compose file that creates images for each of the parts that are needed when deploying this program in the cloud (db-proxy, server to set the environment , APIs...). Finally, I think we can be faced with an initiative that will evolve the internal operations of schools in Catalonia.

Index Terms—ERP, life cycle, program, educational center, academic years, courses, classes, subjects, students, docker-compose, initiative.

1 Introducció - Context del treball

1.1 Context

Digitalitzar i adaptar els processos del nostre dia a dia és una tònica comú els últims temps. És per això, que la proposta per aquest projecte tracta de la possibilitat d'implantar l'ERP[1] Odoo a un centre educatiu. Aquest és un software de gestió Open Source que crea un entorn digital per poder fer qualsevol tràmit que avui en dia es necessita a les PyMEs (*pequeñas y medianas empresas*).
L'elecció principal d'aquest programa i no un altre es deu a diferents motius: utilització i desenvolupament propis del mateix a la meva empresa actual, molta facilitat d'adaptament i customització, consagrat al país amb una gran diversitat de partners que venen el seu servei i possibilitat d'ajudar-me a millorar com a programador i manager en potència del sector.

1.2 Motivacions

Per trobar motivació a elaborar un Treball de Final de Grau propi, m'he basat en l'experiència passada i la que espero obtenir un cop finalitzat aquest projecte. Tinc clar com a enginyer que vull acabar exercint de Manager d'un grup de treballadors per poder organitzar i gestionar diferents projectes.
És per això que he decidit agafar aquest repte de gestionar un projecte desde el seu inici i passant per totes les fases que té en compte l'enginyeria del Software. A més, espero obtenir, ordenar de manera autònoma i profunditzar sobre el coneixement que dóna aquesta eina de gestió a les empreses.
Si fos possible, ser capaç de desenvolupar un nou flux aprofitant com és l'educació i combinar-ho amb les funcionalitats que porta de sèrie aquest ERP.

1.3 Objectius

L'objectiu que he volgut aconseguir amb aquest projecte és crear un programa funcional per a què un centre educatiu pugui gestionar cursos, classes, assignatures, professors i alumnes.

Algunes de les principals funcionalitats són les següents:
- Registrar informació dels estudiants.
- Registrar informació dels professors.
- Assignar un professor a una classe i/o assignatura.
- Vincular un estudiant a un curs i assignatura.
- Possibilitat de crear usuaris per a cada professor.
- Classificar professors a diferents apartaments.
- Vincular un fil de missatges possibles a diferents classes, cursos, estudiants, etc.
- Exportar qualsevol informació a un excel o csv.

Finalment, aquest document s'organitzarà en:
- Un anàlisi del disseny del programa
- Els requeriments i diagrama de classes
- Les metodologies que he emprat en la seva elaboració.
- Una planificació inicial
- L'explicació general del desenvolupament
- Exemples de validacions i testeig
- Propostes futures de desenvolupament.
- Conclusions.

2 Anàlisi de disseny

2.2 Requeriments

La cerca de requeriments ha estat construida a partir de diverses fonts, amb les quals he creat la següent taula, amb una ID, una descripció i la prioritat del propi requeriment.

Aquesta recerca l'he aconseguit mitjançant una conversa/ entrevista amb una professora de l'escola de primària i ESO on vaig anar. Segons la seva manera de treballar diària, les necessitats que tenen i les mancances que troben, he aconseguit establir una llista de requeriments funcionals i no funcionals.

Alguns dels seus hàbits són els següents:

- ➔ Llista d'assistència de nens i nenes de cada classe.
- ➔ Llista de professorat i horaris del professorat.
- ➔ Llista d'assignatures per classe i horari.
- ➔ Llista de material per cada assignatura.
- ➔ Llista de plantilla per cada any acadèmic.

Algunes de les mancances que trobaven:

- ➔ Falta de comunicació propera amb el professorat intern i amb l'alumnat.
- ➔ Falta de diferència d'accions entre un professor i un altre, de diferents càrrecs.

A partir d'aquest comentaris he elaborat una taula separada per classes / models amb els propis requeriments necessaris per safister aquestes necessitats.

Veure taula de requeriments a l'Annex A1.

2.2 Estructura

Es diferencien dues parts que han donat vida a aquest projecte. La part interna de programari i el conjunt de diagrames del que constarà l'aplicació.

2.2.1 Estructura interna de programari

Per poder iniciar un projecte a Odoo, es necessiten els següents elements:
Comptem amb un fitxer docker-compose.yml per definir i executar aplicacions Docker de diversos contenidors. En el context d'un projecte Odoo, s'utilitza per especificar els diferents serveis que formen l'aplicació Odoo, com ara la base de dades, el servidor Odoo i qualsevol dependència o servei addicional que sigui necessari. Cada servei es defineix com un bloc al fitxer docker-compose.yml i es pot configurar amb diverses opcions, com ara variables d'entorn, volums i ports. L'ordre docker-compose up/ down es pot utilitzar per iniciar, aturar i gestionar tota l'aplicació com a unitat única, en lloc d'haver de gestionar cada contenidor individualment. Concretament, definim dos volums per carregar mòduls al nostre projecte, que es poden veure a la figura inferior d'aquest apartat.
A la primera ruta trobem els mòduls creats de manera personalitzada i que difereixen dels que proporciona Odoo com a base, així com la modificació dels comportaments dels mateixos que ofereix l'ERP.
La segona ruta carrega els mòduls base i OCA (Odoo Community Association), organització explicada a l'apartat 4.

```
volumes:
    - ./odoo/custom:/opt/odoo/custom:ro,z
    - ./odoo/auto/addons:/opt/odoo/auto/addons:rw,z
```

Els mòduls que he programat es troben al primer volum i a la ruta especificada.
També existeix la key command, la qual serveix per configurar com volem que s'inicii el servidor. Alguns de les comandes que més he utilitzat durant el desenvolupament han estat les següents:

- –dev=xml: serveix per renderitzar les vistes sense haver de reiniciar el contenidor.
- –update=odoo_school: serveix per actualitzar el mòdul al reiniciar el docker. Útil i necessari al crear atributs nous d'una classe.
- Moltes altres comandes de configuració de workers i límits de memòria i time out.

2.2.2 Estructura de funcionament dels mòduls School

Per aconseguir una funcionalitat adaptada al dia a dia d'un centre educatiu, he hagut de crear dos diferents mòduls (definició de mòdul a l'apartat 4): School Base i School Classroom.

El primer concentra els models (classes) principals del circuit. El segon crea les relacions entre els models principals, adaptats a una classe (com p.e. 4 primària B). I el tercer involucra una nova funcionalitat extra d'afegir exàmens a les diferents assignatures existents. L'School Base és el *core* de les funcionalitats els dos altres mòduls depenen del mateix.

2.3 Diagrama de classes

Per veure-ho més visual, he creat un diagrama de classes que explica les diferents entitats i les relacions que les vinculen, es pot veure en l'Annex A2.

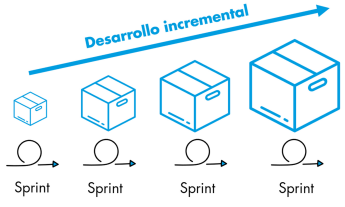
Les classes existents són les següents:

- **Student:** estudiant.
- **StudentCourse:** accés d'estudiant a un curs.
- **Course:** curs.
- **AcademicYear:** any acadèmic.
- **Subject:** assignatura.
- **Department:** departament.
- **Classroom:** classe.
- **Batch:** lot.
- **Faculty:** professorat.
- **Professor:** professor
- **Director:** director.

3 METODOLOGIA

3.1 Metodologia de desenvolupament

La metodologia escollida per portar a terme aquest projecte és la scrum incremental o iterativa. He fet una planificació individual inicial dividida en iteracions, que comença just després de l'entrega de l'informe inicial i acaba uns dies abans de la presentació de la memòria final. Aquest tindrà en compte el desenvolupament i la documentació de tot el projecte.



A la figura inferior es pot veure les iteracions que he formulat amb el seu contingut. Inicialment he seguit bi-setmanalment cada sprint, encara que apropant-se a la data he hagut d'afegir coses que no havia contemplat a les iteracions anteriors.

Sprint 1	Sprint 2	Sprint 3	Sprint 4
Cerca requeriments	Diagrama de classes	Desenvolupament incial mòduls	Desenvolupament de
Creació repositori	Disseny del circuit inicial	Documentació inicial creació mòdul	Formularis, llistes
Creació connexió amb servidor	Analitzar UX	Desenvolupament classes amb atributs	Afegir icones
Connexió amb BD			Documentació de vi

Sprint 5	Sprint 6	Sprint 7
Desenvolupament exportació registres	Creació de fitxers de traduccions	Testeig i validacions funcion
Documentació exportació de registres	Documentació traduccions	Documentació circuit sence
Implementació chatter a models	Valoració de requeriments satisfets	Analitzar desenvolupaments f
Documentació chatter		

3.2 Planificació

3.2.1 Planificació general

Es pot veure el diagrama de gantt creat a l'informe inicial adjuntat a l'Annex A3, on es distribueix cronològicament les diferents etapes del cicle de vida del projecte.

He dividit les tasques a tenir en compte segons grans blocs:

- Configuració del projecte
- Disseny i desenvolupament
  - Disseny de vistes
  - Implementació del front-end
  - Implementació del back-end
  - Proves funcionals
- Documentació i testeig

Aquesta s'ha vist afectada i modificada segons ha avançat el projecte, semblant-se més a la divisió per sprints de les figures de l'apartat 3.1.

3.2.1 Planificació de lliuraments

He creat un repositori amb una branca màster on tindrà el codi un cop validat i implementat. Per crear nous features es crea altres branques filles, on cada una d'aquestes són la implementació d'una nova funcionalitat.

La nomenclatura dels commits ha seguit la següent expressió:

[ADD/FIX/DEL] v.x *description of changes*

4 DESENVOLUPAMENT

4.1 Mòduls

A Odoo, un mòdul és una col·lecció de fitxers que defineixen una funcionalitat empresarial específica i l'implementa a Odoo. Pot contenir models, vistes, controladors, informes, menús i altres components necessaris per implementar la funcionalitat desitjada. Els mòduls es poden instal·lar, configurar i gestionar des de la interfície d'Odoo. Us permeten ampliar i personalitzar la funcionalitat d'Odoo per adaptar-se a les necessitats específiques del vostre negoci.

Es diferencien tres tipus de mòduls a Odoo:

Base: es desenvolupen internament desde l'empresa Odoo i són testejats i mantinguts per la pròpia entitat. Normalment conforme el funcionament estàndard de qualsevol empresa PyME (vendes, compres, comptabilitat, inventari…)
OCA: es desenvolupen per la Odoo Community Association, una empresa espanyola que es dedica a proveïr mòduls de comptabilitat adaptats al sistema del país. Aquests també estan mantinguts i migrats[3] per ells mateixos.
Custom: es desenvolupat per una empresa o individu propi, on la manutenció i la migració dels mateixos estan a càrrec de l'entitat o persona que el crea. És el meu cas.

Els elements bàsics d'un mòdul i que jo he tingut en compte han estat els següents, a l'Annex A4 es poden veure imatges de com es distribueix als directoris interns:

School_base: nom del mòdul i què servirà per distingir entre identificadors externs al codi.
controllers: controlar el canvi de urls segons unes condicions.
Data: informació basada en models, que ajuda a carregar registres creats desde el propi codi. P.e. una acció planificada, la qual executa una funció python es pot crear desde la carpeta data.
Demo: variació de la carpeta data, on, en comptes de crear registres reals que s'aprofiten a producció, es poden crear ficticis i carregar-los, per fer tests i tenir informació al programa.
i18n: carpeta on es troben els arxius .po i .pot, aquests són arxius de text que modulen les traduccions entre els diferents idiomes.

```
#. module: school_base
#: model:ir.ui.menu,name:school_base.menu_school_config_course
msgid "Course Management"
msgstr "Gestión de los Cursos"
```

Menu: carpeta opcional, on aniran els punts de menú principals del mòdul en format .xml, aquest es veuran distribuïts a la part superior de la finestra (UI i UX parlant), podem veure una imatge a l'inferior.
Report: carpeta on es situen els informes. Aquests són imprimibles desde accions als registres (com per exemple es pot imprimir una comanda de venda o un full d'horaris d'una classe). Estan en format .xml.

Static: formen part d'aquesta carpeta tots els arxius binaris multimedia que entren dins del mòdul, desde la seva pròpia imatge dins de Odoo, fins mostres de registres que es pugui pujar una foto per identificar-los (com per exemple un estudiant)
Security: carpeta on es troba un fitxer csv per declarar que grups de permisos tenen accés a la creació, modificació, eliminació, etc. d'un model. Els models inherit hereden també aquest fitxer.
Views: en aquesta carpeta es poden veure les vistes que donaran lloc el back-end creat a la carpeta models. Són en format xml i el seu nom consta del nom del model i acompanyat de view(s).
Wizard: formen part d'aquest carpeta tots els fitxers, tant xml com python, creen i modifiquen finestres emergents que fan alguna acció. Aquestes utilitzen transcient models (explicats més avall). Algun exemple comú podria ser un importador de dades.
Manifest: és el fitxer més important del mòdul, aquest conté tota la informació general del mateix i carrega les vistes, dades, security, etc. És un fitxer json.

4.1.1 Mòduls desenvolupats

Un cop explicada la introducció als tipus de mòduls i els seus directoris principals, el meu desenvolupament ha estat la generació de dos mòduls. L'School Base és el mòdul custom referència desenvolupat desde 0, he creat els directoris de data, i18n, menu, models, report, security, static, views, wizard i el fitxer manifest. El mateix es pot veure a la figura inferior:

```
{
    'name': 'School Base',
    'version': '14.0.1.0',
    'license': 'LGPL-3',
    'category': 'Education',
    "sequence": 1,
    'summary': 'Manage Students, Faculties and Education Institute',
    'depends': ['hr', 'web', 'website'],
    'data': [
        'security/op_security.xml',
        'security/ir.model.access.csv',
        'report/report_menu.xml',
        'report/report_student_bonafide.xml',
        'report/report_student_idcard.xml',
        'wizard/faculty_create_employee_wizard_view.xml',
        'wizard/faculty_create_user_wizard_view.xml',
        'wizard/students_create_user_wizard_view.xml',
        'views/department_view.xml',
        'views/res_company_view.xml',
        'views/student_view.xml',
        'views/hr_view.xml',
        'views/category_view.xml',
        'views/course_view.xml',
        'views/batch_view.xml',
        'views/subject_view.xml',
        'views/faculty_view.xml',
        'views/website_assets.xml',
        'views/subject_registration_view.xml',
        'views/res_config_setting_view.xml',
        'views/student_portal_view.xml',
        'views/student_course_view.xml',
        'views/op_academic_year_view.xml',
        'views/op_academic_term_view.xml',
        # 'data/ir_cron_data.xml',
        'menu/school_base_menu.xml',
        'menu/faculty_menu.xml',
        'menu/student_menu.xml',
```

Dependencies de mòduls als que fa herència

Depends és una llista dels mòduls que han d'estar instal·lats per poder fer servir el mateix. HR per poder donar la possibilitat de, al crear un usuari vincular-ho a un professor o alumne, web i website per poder utilizar funcionalitat de capçaleres al generar un informe i desenvolupat amb llenguatge xml i utilitzar la modalitat Qweb[2].

La key JSON data ha d'incloure security, vistes, wizard i menus creats a les seves respectives carpetes.

- Security: restricció de lectura, escriptura i eliminació dels models creats (models explicats a l'apartat 4.2.1) en base a un grup específic. A més, op_security.xml (situat a la propia carpeta) s'encarrega de crear per codi els grups i les normes de registres. Aquestes s'assignen als diferents usuaris segons el seu grau al centre educatiu.
- He creat un report mostrant la infomació bàsica sobre un estudiant.
- Existeixen vistes per a cada model, explicades a l'apartat 4.3.1.
- Els wizards existents són per afegir un pop-up al clicar el botó de crear usuari juntament al emplenar el formulari d'estudiant o professor, els tres segueixen la mateixa estructura. Com a professor també s'habilita crear un empleat, a part de l'usuari de l'aplicació.

4.2 Models

Carpeta més important del propi mòdul, aquí és on es situa tot el back-end. cada fitxer python compendrà un model diferent, o sino els models agrupats per funcionalitats més atòmiques. els noms de tots estan formats per dos paraules o més en anglès separades per un punt.

Es diferencien varis tipus de models dins de Odoo:

Base / Custom: no deixa de formar part dels mòduls base que s'han comentat anteriorment, a més poden ser creats de 0 sense heredar cap funcionalitat d'un altre. Alguns exemples podrien ser: sale.order, purchase.order, res.company, product.template.
Inherit: model que hereda les característiques d'un altre, ja sigui base o custom. Si es vol afegir una nova funcionalitat o camp a un model ja existent, hem de declarar una classe com a p.e. _inherit='product.template'.
Transcient: models ons els registres es netejen i s'eliminen cada 24h, serveixen per executar un procés secundari dins d'un model principal, un exemple molt clar és un wizard (que es comenta posteriorment).
Aquí es mostra una imatge de la creació d'un model inherit on afegeix camps al model res.users (usuaris de l'applicació).

```
class ResUsers(models.Model):
    _inherit = "res.users"

    def _department_count(self):
        return self.env['op.department'].sudo().search_count([])

    student_line = fields.Many2one('op.student', 'Line')
    user_line = fields.One2many('op.student', 'user_id', 'User Line')
    child_ids = fields.Many2many(
        'res.users', 'res_user_first_rel1',
        'user_id', 'res_user_second_rel1', string='Childs')
    dept_id = fields.Many2one('op.department', string='Department Name')
    department_ids = fields.Many2many('op.department',
                                      string='Allowed Department')
    department_count = fields.Integer(compute='_compute_department_count',
                                      string="Number of Departments",
                                      default=_department_count)
```

4.1.1 Models desenvolupats

En base als requeriments establerts i al diagrama de classes presentat als apartats 2.2 i 2.3, he creat models inherit, transcient i custom i també he afegit atributs a models ja existents. Mostro una llista explicativa exposant les diferents classes i el motiu / explicació de la seva estructura:

Inherit: He creat models heredant funcionalitats profitoses pel comportament que ha de tenir el mateix.

- Student: s'ha heredat camps que té el res.partner (client) de la base de odoo, com pot ser la informació de l'adreça. A més, quan es crea un registre d'estudiant, també es crea un client vinculat, fent-lo potencial comprador de material, quotes o més possibilitats futures.
- Faculty: mostra el mateix comportament que l'estudiant, aquest professorat anirà associat a un client mitjançant aquesta herencia amb el model res.partner, a més de camps informatius que ja conté aquest model i que no cal crear dues vegades.
- Cas 'mail.thread': A odoo, heredar un model no sempre fa que l'aspecte i els atributs siguin els mateixos, hi ha un cas com és el model mail.thead, que s'utilitza per afegir un fil de correus a qualsevol dels registres. Per fer-ho més visual, si es vol fer possible aquesta funcionalitat, heredem el model i afegim atributs a la vista formulari del mateix, fent possible establir una conversa entre els usuaris (res.user) o contactes (res.partner) i la persona que ho envia. Concretament, he heredat aquesta funcionalitat a estudiants, professorat, cursos i assignatures.

Transcient: l'utilitat principal de crear un model d'aquest tipus és l'estalvi de memòria, ja què aquests netegen els seus registres un cop fets servits. El cas més comú és la creació de pop-ups, taules one2many amb registres intermitjos per a fer càlculs, etc. El meu cas he fet servit aquests models per generar *wizard.student*, *wizard.faculty* i *wizard.faculty.employee,* un exemple és el que es veu a la figura següent:

```
from odoo import models, fields


class WizardOpStudent(models.TransientModel):
    _name = 'wizard.student'
    _description = "Create User for selected Student(s)"

    def _get_students(self):
        if self.env.context and self.env.context.get('active_ids'):
            return self.env.context.get('active_ids')
        return []

    student_ids = fields.Many2many(
        'student', default=_get_students, string='Students')

    def create_user(self):
        active_ids = self.env.context.get('active_ids', []) or []
        records = self.env['student'].browse(active_ids)
        records.create_student_user()
```

Custom: he escollit els models custom com aquells que:

- No necessiten heredar cap funcionalitat ja existent. P. e. *academic.year, classroom* (al mòdul school classroom).
- Permeten ser un seleccionable modificable amb atributs propis sobre un model principal. P. e. *category* (categoria escolar és un atribut d'estudiant)
- Servir com a taula intermitja entre dos models principals (normalment many2many). P. e. *student.course, subject.registration.*

Base: quan es vol afegir atributs sobre models ja existents, normalment es pot fer per dues raons. La primera és personalitzar un circuit ja existent i que ofereix Odoo (per exemple, afegir més dades a una comanda de venda, ficar nous atributs a productes d'acord del que es vengui, etc), la segona és afegir dades a models globals dins de la

jerarquia del propi ERP. He valorat els dos casos i ara explico el perquè.

1. He afegit informació amb possibilitat d'ampliació en un futur sobre la companyia, adaptada a l'escola al model *res.company*. Aquest model s'utilitza per configurar variables globals dins d'una mateixa companyia i que, si afegís una altra nova, podria modificar-se i seria diferenciadora entre ambdues.
2. He afegit informació al model de *res.users* (usuari) com el departament i un víncul amb l'estudiant.

4.3 Vistes

A Odoo, una vista fa referència a la manera com es presenta la informació a l'usuari. Defineix la disposició, l'estructura i l'organització d'un formulari, arbre o Kanban. Odoo ofereix diversos tipus de vista, com ara la vista de formulari, la vista en arbre, la vista kanban i la vista de cerca, que es poden utilitzar per representar dades de diferents maneres. Les vistes es defineixen mitjançant XML i es poden personalitzar afegint camps i widgets personalitzats per mostrar la informació en el format desitjat.

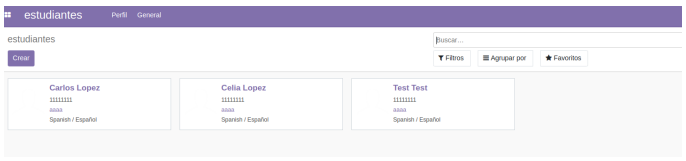
Concretament a aquest projecte, he utilitzat totes les vistes possibles depenent del tipus de classe. A la part inferior, es poden veure com es defineixen tots els tipus pertinents.

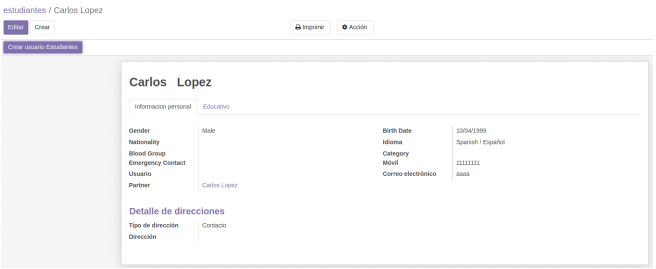
```
<record id="view_op_course_form" model="ir.ui.view">
    <field name="name">course.form</field>
    <field name="model">course</field>
    <field name="priority" eval="8"/>
    <field name="arch" type="xml">
        <form string="Course">
            <sheet>
```

```
<record id="view_op_course_tree" model="ir.ui.view">
    <field name="name">course.tree</field>
    <field name="model">course</field>
    <field name="priority" eval="8"/>
    <field name="arch" type="xml">
        <tree string="Course">
            <field name="name"/>
```

```
<record id="view_op_course_search" model="ir.ui.view">
    <field name="name">course.search</field>
    <field name="model">course</field>
    <field name="priority" eval="8"/>
    <field name="arch" type="xml">
        <search string="Course">
```

Els resultats de les següents vistes també es poden veure a les figures de la part inferior, tree, search i formulari respectivament.





4.1.1 Vistes creades

Com ja sabem, al igual que els models, les vistes també es poden heredar. Aquesta acció pot fer-se servir per diferents motius segons la intenció del programador. El meu cas ha estat divers, és a dir, he heretat vistes formulari i llista si es volgut afegir un camp (o treure, restringir, moure'l de lloc…) com és el cas dels nous camps de la companyia i usuaris.
Un altre cas és crear una vista de 0 per presentar informació sobre un model nou o adaptat al nou flux. Si recuperem la llista de models creats, mostraré les vistes que he creat per a cadascun.
Student: tree, form, kanban, search, menu
Faculty: tree, form, kanban, search, menu
Academic Year: tree, form, menu
Academic Term: tree, form, menu
Courses: tree, form, menu
Departments: tree, form, menu
Subjects: tree, form, menu
Subject Registration: tree, form, menu, states[4]

** Si un tipus de vista necessària no es crea per codi, genera una per defecte.

4.4 Funcions principals bàsiques

En aquest apartat, explicaré uns exemples de funcions significatives i com utilitzar el llenguatge que ofereix odoo per programar el back-end de l'aplicació.

Per cridar un model a la base de dades s'utilitza l'expressió *self.env[nom.model]* seguit de *.search([(field, '=', 'expression')])*, amb això es poden fer consultes, tant simples com compostes, obtenint el(s) registre(s) que necessitem i accedir als seus atributs amb un punt, seguit del propi atribut.
Amb la importació de la llibreria *odoo*, podem inicialitzar camps, utilitzar decoradors a les funcions i crear models. Aquí un exemple:

```
from odoo import models, fields, api

class Student(models.Model)

name = fields.Char()

@api.onchange('name')
def _onchange_name(self):
        #TODO
```

Un altre tip per evitar duplicacions directament a la base de dades, es basa en l'atribut de _sql_constraints, al següent exemple es pot veure com he limitat a que un codi d'uns curs escolar sigui únic per la bd:

```
_sql_constraints = [
    ('unique_course_code',
     'unique(code)', 'Code should be unique per course!')]
```

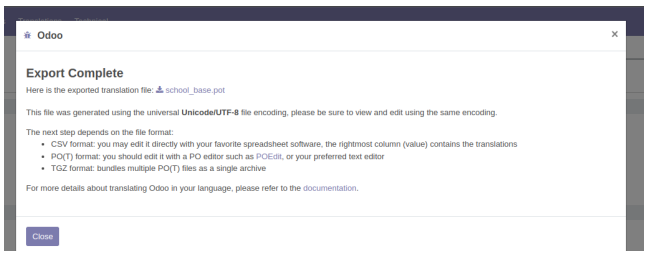
On code és l'atribut a tenir en compte.

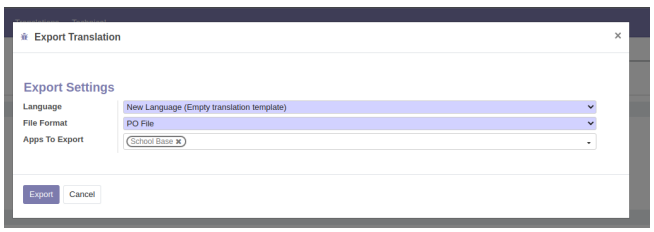
4.5 Traduccions

Comprenem com a traducció a Odoo la possibilitat de d'adecuar l'idioma a l'usuari que veu un text a la interfície. Per poder modificar la traducció d'un camp o text, existeixen els fitxers .po i .pot. Es troba, com he mencionat abans a l'apartat 4.1, a la carpeta i18n. S'ha de crear tants arxius com idiomes es vulgui traduir el mòdul.

4.1.1 Traduccions creades

Jo he creat traduccions al castellà pels dos mòduls i tots els models creats. Funcionalment i per no haver de crear un fitxer de 0, comptem amb una facilitat que ara mostraré a la figura. Accedim a setting i exportem traduccions:





Això genera un fitxer amb tots els camps creats al mòdul amb la traducció buida, simplement emplenem el literal en l'idioma requerit i si modifiquem l'idioma de l'usuari aquell camp es veurà a la vista traduït.

5 Validació i Testeig

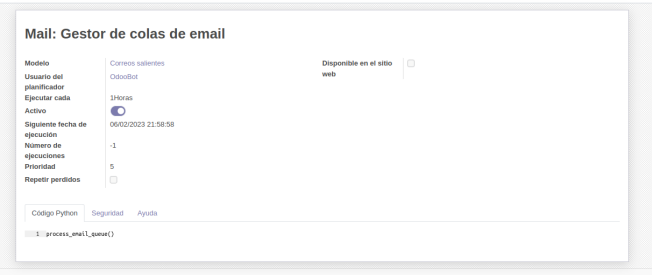
5.1 Proves funcionals

He elaborat proves de UI/UX per validar que el funcionament és correcte i no atenem a bugs descontrolats, ha sigut quan he creat restriccions de duplicitat per registres. Aquestes, com he explicat a l'apartat 4.4, miren que la clau primària de cada model no es pugui repetir.

6 Desenvolupaments Futurs

6.1 Utilitat d'accions planificades

Automatitzar processos sempre és una millora a qualsevol model de negoci, per tant, un següent pas pot ser utilitzar les accions planificades (scheduled actions) per llançar funcions periòdicament. Aquest és el model *ir_cron* i consta de diferents paràmetres de configuració.



Aquest és un exemple d'acció planificada per revisar els correus entrants, es pot executar codi sobre un model i diferents registres periòdicament.

6.2 Nova recerca de requeriments

Aquesta aplicació, ja contant amb un complert circuit per treballar a l'àmbit educatiu, es pot ampliar i obtenir noves funcionalitats. He elaborat una llista de les que penso que serien profitoses i útils segons el que m'he trobat desenvolupant l'existent.

- Gestió d'exàmens i excursions.
- Aplicació de seguretat al servidor (VPN).
- Afegir qualificacions.
- Crear informes sobre autoritzacions d'activitats.
- Traduccions a més idiomes.
- Entorn multi-companyia adaptat a una franquícia.

6.3 Implantació a un servidor real

Per poder utilitzar aquest projecte al núvol cal allotjar el servidor a una de les màquines que pot proporcionar un Amazon, Google, etc. Penso que pot ser útil aquest Treball de Final de Grau segons la menció cursada. A partir d'allotjar al domini tant les dades com el codi de l'aplicació, es podria montar un monitoreig de desplegaments, així com testos automatitzats al servidor real, així com analitzar els logs i l'ús de la CPU. Tot això es pot fer amb programes com el grafana, i ser personalitzable segons el client i el que l'equip desenvolupador necessiti.

7 Conclusió

Després de finalitzar aquest projecte, puc concloure dient que el resultat ha estat un èxit i he aconseguit elaborar una aplicació amb Odoo per a centres educatius.
La part tècnica de desenvolupament ha estat la més amena, ja que compto amb força experiència i coneixement dels llenguatges que utilitza Odoo, així com la versió orm python que implementa.

Estic aprenent també a plasmar el que demana un client i

jo adaptar-ho a codi, per tant, aplicar creativitat i coneixement de l'estructura del codi també m'ajuda a seguir creixent com a desenvolupador i enginyer informàtic, que es del que es tracta aquesta proposta.

A més, penso que, apart d'aportar-me coneixement, m'ha ajudat a tenir experiència en la gestió i avançar amb el camí.

Agraïments

Agraeixo a la meva tutora Helena per ajudar-me a estructurar el treball, així com fer-me un seguiment gairebé setmanal del projecte. Agraeixo també a la visió que m'ha aconseguit donar el meu equip de desenvolupament durant el temps que porto a l'empresa, m'ha ajudat molt a plasmar el que volia a aquest TFG.
Gràcies també a l'universitat per proporcionar-me l'espai per escollir el treball que volia fer i acompanyar-me durant el procés.

Diccionari

3. Enterprise Resource Planning: software de gestió adaptat a empreses
4. Variació del llenguatge XML amb facilitats de classes adaptades a la generació d'informes i impressos.
5. Migració es defineix com el procés de actualització d'un component del software a una versió més recent. P. e. migració de Odoo 12 a Odoo 14, aquest implementen canvis de noms de models, nomenclatures, etc.
6. Els estats a Odoo serveixen per classificar registres. Segons a quin estiguin, poden tenir unes accions o unes altres, ordenar-los a la vista kanban, etc.

Bibliografia

[1] **Pàgina web Odoo** - https://www.odoo.com/es_ES
[2] **Pàgina web gitlab** - https://about.gitlab.com/
[3] **Desenvolupament iteratiu i incremental** - https://proyectosagiles.org/desarrollo-iterativo-incremental/
[4] **Creació diagrama de Gantt** - https://app.teamgantt.com/
[5] **Documentació odoo 15** - https://www.odoo.com/documentation/15.0/
[6] **Pàgina Web OCA** - https://odoo-community.org/
[7] Odoo Models Documentation - https://www.odoo.com/documentation/16.0/es/developer/howtos/rdtraining/04_basicmodel.html
[8] **Odoo Views Documentation** - https://www.odoo.com/documentation/14.0/es/developer/reference/addons/views.html
[9] **What is a wizard?** - https://www.odoo.com/es_ES/forum/ayuda-1/what-is-wizard-145563
[10] **LucidChart Diagram Creator** - https://www.lucidchart.com/pages/es
[11] **Utilitzar sql constraint** - https://www.odoo.com/es_ES/forum/ayuda-1/how-to-use-constraint-and-sql-constraint-64684

APÈNDIX

A1. REQUERIMENTS

CURS:

ID Req.	Descripció	Prioritat
RF-Crear-curs	El sistema ha de ser capaç de crear un curs introduint les seves dades.	A
RF-Editar-curs	El sistema ha de ser capaç de editar un curs introduint les seves dades.	A
RF-Eliminar-curs	El sistema ha de ser capaç de eliminar un curs introduint les seves dades.	A
RF-Assignar-curs-estudiant	El sistema ha de ser capaç d'atribuir un estudiant o més a un curs.	A

CLASSE:

ID Req.	Descripció	Prioritat
RF-Crear-classe	El sistema ha de ser capaç de crear una classe introduint les seves dades.	A
RF-Editar-classe	El sistema ha de ser capaç de editar una classe introduint les seves dades.	A
RF-Eliminar-classe	El sistema ha de ser capaç de eliminar una classe introduint les seves dades.	A
RF-Assignar-curs-estudiant	El sistema ha de ser capaç d'atribuir un estudiant o més a un curs.	A

ASSIGNATURA:

ID Req.	Descripció	Prioritat
RF-Crear-assignatura	El sistema ha de ser capaç de crear una assignatura introduint les seves dades.	A
RF-Editar-assignatura	El sistema ha de ser capaç de editar una assignatura introduint les seves dades.	A
RF-Eliminar-assignatura	El sistema ha de ser capaç de eliminar una assignatura introduint les seves dades.	A
RF-Assignar-material-assignatura	El sistema ha de ser capaç d'assignar material a una assignatura.	M

PROFESSOR:

ID Req.	Descripció	Prioritat
RF-Crear-professor	El sistema ha de ser capaç de crear un usuari (professor) introduint les seves dades.	A
RF-Assignar-horari-professor	El sistema ha de ser capaç d'assignar un horari d'assignatures a un professor.	M
RF-Assignar-classe-professor	El sistema ha de ser capaç d'assignar una classe a un professor	A

DEPARTAMENT:

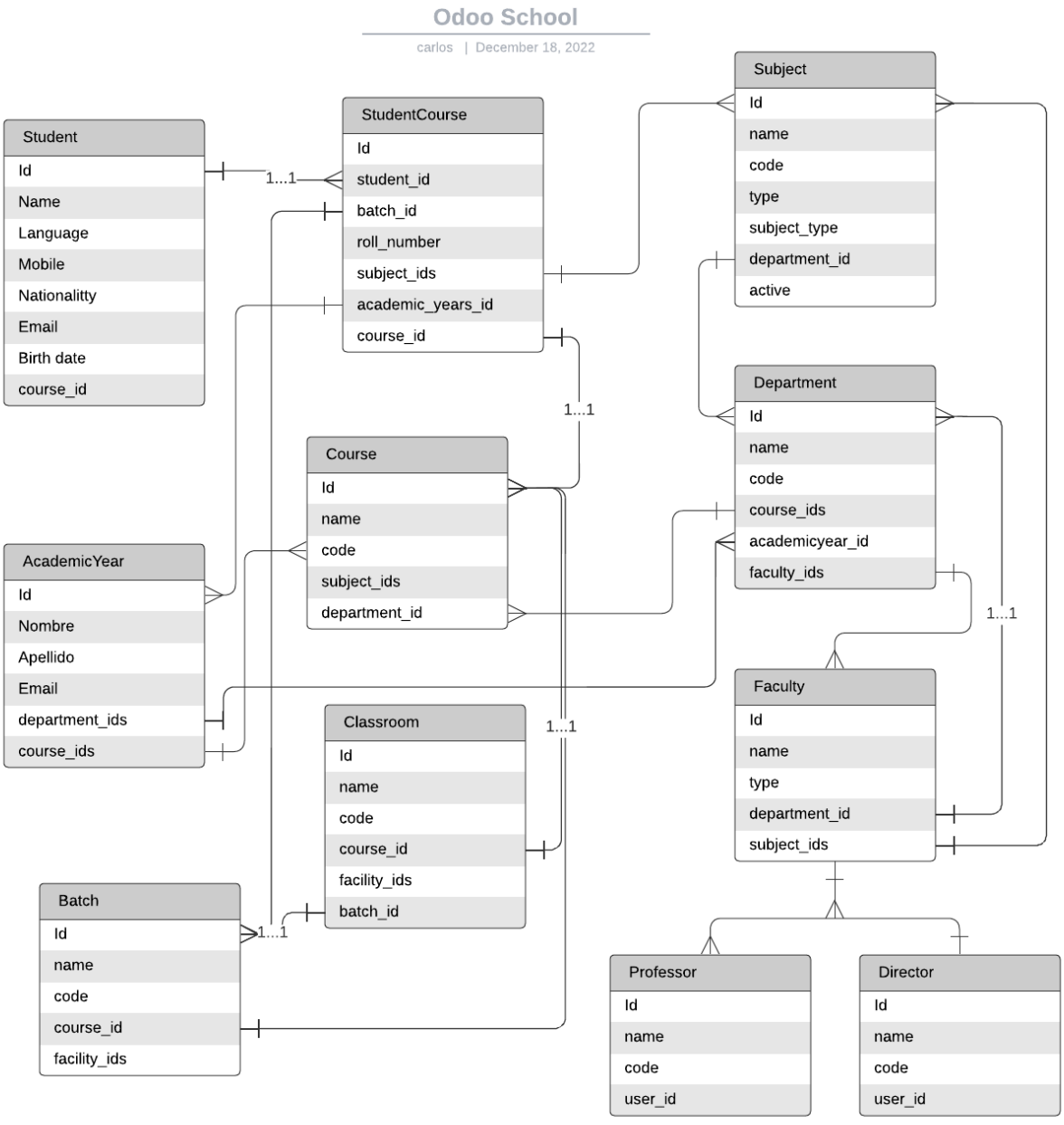
ID Req.	Descripció	Prioritat
RF-Crear-departament	El sistema ha de ser capaç de crear un departament introduint les seves dades.	A
RF-Editar-departament	El sistema ha de ser capaç de editar un departament introduint les seves dades.	A
RF-Eliminar-departament	El sistema ha de ser capaç de eliminar un departament introduint les seves dades.	A
RF-Assignar-departament-professor	El sistema ha de ser capaç de vincular un departament a un o més professors.	A

ESTUDIANT:

ID Req.	Descripció	Prioritat
RF-Crear-estudiant	El sistema ha de ser capaç de crear un estudiant introduint les seves dades.	A
RF-Editar-estudiant	El sistema ha de ser capaç de editar un estudiant introduint les seves dades.	A
RF-Eliminar-estudiant	El sistema ha de ser capaç de eliminar un estudiant introduint les seves dades.	A
RF-Assignar-classe-estudiant	El sistema ha de ser capaç d'atribuir un estudiant o més a una classe.	A
RF-Assignar-curs-estudiant	El sistema ha de ser capaç d'atribuir un estudiant o més a un curs.	A
RF-Assignar-assignatura-estudiant	El sistema ha de ser capaç d'atribuir un estudiant o més a una assignatura.	A

ID Req.	Descripció	Prioritat
RNF-multinavegador	El sistema ha de poder ser iniciat en Google Chrome, Firefox i Opera.	B
RNF-registre-màxim-usuaris	El sistema ha de poder registrar i crear un màxim de 50 usuaris actius.	M
RNF-canvi-idioma	El sistema ha de poder ser traduït en diferents llengües.	B

A2. Diagrama de bases de dades



A3. Planificació per diagrama de Gantt

A4. Estructura mòduls al codi