Universitat de Girona
Escola Politècnica Superior

Treball final de grau

Estudi: Grau en Enginyeria Informàtica

Títol:

Automatització de la interacció amb portals de distribuïdores elèctriques

Document: Memòria

Alumne: David Augusto Suárez

Tutor: Dr. Martin Campos, Ignacio Clemente
Departament: Informàtica, Matemàtica Aplicada i Estadística
Àrea: Llenguatges i Sistemes Informàtics

Convocatòria (mes/any) Setembre 2020

Automatització de la interacció amb portals de distribuïdores elèctriques
GEINF - TFG

David Suárez

September 2020

Contents

1 Introducció

1.1 Context, origen i motivació.

Per entendre el sector de l'energía elèctrica tal i com el coneixem actualment a Espanya cal fer un salt enrere fins al 28 de novembre 1997, quan es publica la llei 54/1997[6] que dóna inici al que anomenem la *liberalització del mercat elèctric* amb l'objectiu de fer el sector més competitiu en termes de preu i servei.

Abans d'això, els grans grups empresarials feien de forma íntegra la generació, distribució i comercialització de l'energía elèctrica. A partir de la liberalització, el mercat elèctric es va dividir en quatre activitats:

- Producció: Lloc de generació de l'energia (centrals de gas natural, nuclears, planters solars, hidroelèctriques, etc.)
- Transport: Xarxa de alta tensió. Aquesta activitat la de desenvolupa exclusivament Red Eléctrica Española (REE), qui es responsabilitza del desenvolupament i manteniment d'aquesta xarxa.
- Distribució: Aquesta activitat està regulada per l'estat i és aquest qui assigna a cada zona geogràfica una única empresa distribuïdora. És la responsable de fer arribar l'energia fins als nuclis urbans. Tanmateix, son les responsables dels punts de subministre **CUPS** del que parlem més endavant amb més detall.
- Comercialització: Les empreses comercialitzadores s'encarreguen de comprar energia elèctrica per els seus consumidors. La compra d'aquesta es fa al *Mercat Elèctric* que és gestionat per un operador independent: **OMIE**. Aquesta activitat és de lliure competència, i existeixen múltiples empreses acreditades per la *Comissió Nacional dels Mercats i la Competència* més coneguda per les sigles **CNMC**.

Som Energia

Dins aquest marc trobem la cooperativa *Som Energia* que desenvolupa dues activitats: producció i comercialització d'energia de fonts 100% renovables. Degut a un creixement molt accelerat i a les dificultats tant pels costos elevats com altres factors (localització, burocràcia, etc.) per a poder fer viable un projecte de generació elèctrica, Som Energia genera una petita part de l'energía que comercialitza. No obstant, poder generar la totalitat de l'energia que es comercialitza, és el que a llarg termini es pretén.

Actualment compta amb un equip de prop de 90 persones. Les oficines de la cooperativa es troben al Parc Científic i Tecnològic de la UdG.

Per a la correcta gestió de l'activitat cooperativa, Som Energia comprèn entre altres equips, l'equip IT: que és l'encarregat de donar suport tant a la infraestructura informàtica, com al desenvolupament i manteniment d'eines específiques.

L'equip IT, segueix una línia de software de codi obert, tant pel que fa a les eines de tercers com al desenvolupament que duu a terme per a la cooperativa.

Punt de subministrament CUPS: Importància i problema

Una de les feines que es duen a terme internament a *Som Energia*, és la sincronització de la informació referent a altes, baixes, canvis de potències, modificacions contractuals, incidències, etc. de les diferents distribuïdores elèctriques de l'estat Espanyol encarregades dels punts de subministrament que han contractat a Som Energia com a comercialitzadora elèctrica.

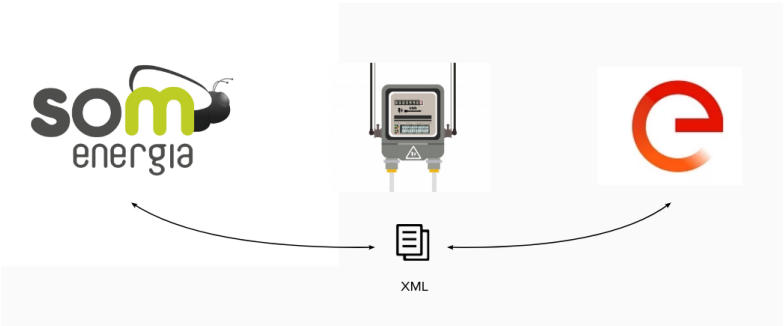


Figure 1: Comunicació entre comercialitzadora i distribuïdora.

Aquesta interacció no es realitza mitjançant un portal comú. Tot el que es regula és el format d'aquestes comunicacions, determinat per la CNMC.

Dit d'una forma molt senzilla, tota la comunicació que ha de tenir una comercialitzadora, posem pel cas *Som Energia*, amb les diferents distribuïdores és per posar-se d'acord amb quines condicions van associades a un punt de subministrament i els seus respectius consums.

Així doncs, cada distribuïdora té el seu propi portal d'accés a aquestes comunicacions, que mitjançant unes credencials, podrem accedir als arxius corresponents, en format XML.

Actualment és un procés manual que requereix entre 10 i 15 hores de feina setmanal repartit entre diverses persones de l'equip tècnic.

És en aquests context que neix el projecte d'Automatització de la interacció amb portals, d'ara en endavant *massive_importer*.

1.2 Definicions, acrònims i abreviatures

ERP: Programa que integra totes les dades i els processos d'una organització en un sistema unificat.
TI o IT: Equip amb coneixements en tecnologies de la informació
ATR: Accés de Tercers a la Xarxa. És el permís que la empresa distribuïdora dona per que el client pugui tenir accés a la xarxa en un mercat lliure.
CNMC: Comissió Nacional dels Mercats i la Competència
ORM: Object-relational mapping
API: Application Programming Interface

FUSE: Filesystem in Userspace (sistema de d'arxius)
XML: eXtensible Markup Language, metallenguatge d'etiquetes

1.3 Definició i propòsits del projecte.

Com qualsevol organització amb una dimensió similar, *Som Energia* treballa amb un ERP el qual en aquest cas s'ha optat per un conegut software de codi obert anomenat *OpenERP*[1] fins 2012, actualment conegut com *Odoo*.

Aquesta eina és una part essencial per a la gestió dels contractes, tarifes, modificacions contractuals, reclamacions, etc.

La solució amb la que treballa la cooperativa, ha estat adoptada de Gisce-TI[2], com a proveïdor d'aquest programari *customitzat* per a la comercialització elèctrica. Partint d'una branca d'OpenERP, Gisce-TI inicia un continu desenvolupament per tal de ajustar-lo cada cop més al context de la comercialització d'energia, el qual va fluctuant amb els Butlletins oficials de l'Estat o directrius de la CNMC, com a principals agents de canvi.

Actualment una de les tasques que s'ha de dur a terme manualment, és la de anar a buscar totes les comunicacions amb la distribuïdora per posar-les en comú al nostre sistema. Aquestes comunicacions en diem casos ATR ("Acceso de Terceros a la Red"). Aquest procés de sincronització, a falta d'un sistema centralitzat, passa a ser totalment disgregat i repartit en tantes plataformes com distribuïdores que tinguin relació amb *Som Energia*.

El procés de importació dels casos ATR (fitxers en format XML, sovint ajuntats dins un mateix fitxer comprimit) ja ha estat desenvolupat de forma que és molt eficient i capaç de trobar errors en el format, així com incoherències en el tractamet de les dades. Per poder accedir a aquest procés, OpenERP ens dona una interfície o API a la que ens podem connectar, per tal d'executar-ne els seus mòduls des d'un programa extern.

En aquest cas concret, el que volem cridar és l'assistent d'importació de casos ATR, que trobem annexat (Annex D, figures 51 en endavant). Internament, els assistens els coneixem com a *Wizard*, i aquest concretament ens permet importar al ERP tota aquesta informació en forma de fitxers *xml*.

Aquesta la tasca doncs, tant l'accés als portals i la descàrrega de fitxers, com accionar l'assistent d'importació del ERP, tot seleccionant els arxius descarregats es realitza **diariament**, a primera hora del matí, entre les 8:00 i 11:00 de dilluns a divendres. L'automatització d'aquest procés és un projecte proposat al llarg dels darrers anys a *Som Energia*, que mai ha conclòs per falta de prioritat davant altres tasques que han anat sorgint eventualment així com per la dimensió de l'equip de IT.

El que es vol aconseguir amb aquest treball és desenvolupar l'**anàlisi**, **disseny** i **implementació** d'un sistema que progressivament porti a terme aquest procés de forma automàtica i massiva.

1.4 Objectius

La intenció final és la d'estalviar temps d'una feina mecànica i tediosa a l'equip tècnic de *Som Energia* a través d'un sistema que automatitzi aquest procés, diferenciat en els següents objectius:

- La descàrrega de fitxers de les diferents plataformes.

- La importació dels nous casos al ERP.
- Seguiment de l'estat d'aquests processos

Descàrrega de fitxers

El procés ha de permetre que, determinat un portal o un llistat de portals, iniciar un procés de navegació pel qual s'accedeixi amb els credencials corresponents a la secció de descàrregues de casos ATR, i efectuï la descàrrega dels casos pendents des de l'ultima execució del mateix.

Importació de nous casos al ERP

El procés ha de permetre que a partir d'un nou fitxer amb extensió *xml* o *zip*, gestionar aquest nou fitxer per incorporar els casos continguts al ERP.
Es vol mantenir l'estat d'aquest procés, per poder identificar en quin moment s'ha efectuat correctament la importació, o bé si ha succeït algun error durant el mateix.

Generació d'un resum

S'ha de poder conèixer si ha correctament tant el procés de descàrrega com el procés d'importació. Aquesta supervisió serà obligatòria, perquè no importar correctament algun cas en la automatització, requerirà la intervenció per part de l'equip de Gestió ATR per incorporar els fitxers que faltin.

1.5 Necessitats

Els integrants de l'equip tècnic, als quals anomenarem *usuaris*, fins ara tenen control sobre tots els aspectes relacionats amb aquesta tasca. No obstant, en tractar-se d'una automatització, les necessitats seran més reduïdes, éssent estrictament de control. D'altra banda tenim les necessitats de l' *administrador* el qual voldrà poder configurar certs aspectes de l'automatització.

Necessitats de l'*usuari*

- Consultar l'estat dels casos importats.
- Consultar el moment de la darrera importació.
- Consultar l'existència de casos pendents.

Necessitats de l'*administrador*

- Gestionar els portals que es vol incloure al procés
- Gestionar els credencials per als diferents portals
- Gestionar la programació de les tasques
- Gestionar les connexions al ERP, BD, etc.
- Consultar un registre d'esdeveniments (log)

1.6 Estructura de la memòria

Capítol 1. Introducció. Contextualització de l'empresa on es realtiza el projecte, així com les raons, motivacions i necessitats per a dur a terme aquest treball.

Capítol 2. Estudi de viabilitat. Precedents del projecte i raons per les qual es considera viable.

Capítol 3. Metodologia. Es justifica la metodología empreada, així com es detalla la seva implementació.

Capítol 4. Planificació. Es defineixen les parts del proejcte, així com l'estrategia seguida per a dur a terme el projecte.

Capítol 5. Marc de treball i conceptes previs. Mencionem breument tecnologíes relacionades directament amb el treball, així com alguna de les desicions preses a l'hora descollir-les.

Capítol 6. Requisits del sistema. Requeriments funcionals i no funcionals del nostre programa.

Capítol 7. Estudis i decisions. Les decisions preses, eines descartades i altres aspectes en relacio a la presa de decisions.

Capítol 8. Estudis i decisions. S'hi descriuen el hardware i les llibreries de software utilitzats durant el desenvolupament del projecte.

Capítol 9. Implementacio i proves. S'hi explica la implementacio i proves de les diferents parts del projecte.

Capítol 10. Implantanció i resultats.

Capítol 11. Conclusions i treball futur.

2 Estudi de viabilitat

La automatització de processos és un objectiu actualment amb molt de protagonisme a *Som Energia*. Es tracta d'una cooperativa molt jove, que ha patit un creixement molt important en un període de temps reduït. Això ha fet que tasques que inicialment, per simplicitat i viabilitat havien estat manuals o bé dutes a terme amb eines ofimàtiques o equivalents, ara passin a ocupar un temps significatiu de l'equip tècnic (sovint tasques directament inviables) el qual no s'està destinant a funcions que requereixen un coneixement de domini.

Actualment la cooperativa ha manifestat la intenció de donar certa prioritat a l'automatització de tasques. És per això, que des de l'equip TI s'ha oferit la possibilitat de transformar un d'aquests automatismes en un Projecte Final de Grau.

Abast del projecte

Es pretén estalviar la tasca tediosa i susceptible a errors humans: L'accés a les múltiples plataformes webs, cadascuna amb les respectives credencials, el filtrat de les cerques i la descarrega de fitxers. Posteriorment accedir a la importació de fitxers des del propi ERP, i iniciar el procés de importació manualment.

Anàlisi de situació

Es disposen d'eines i recursos hardware suficients per tal de no haver de contemplar l'ampliació o adquisició de cap llicència ni cap element de hardware de forma addicional per a implantació d'un sistema que automatitzi aquest procés.

Definició de requisits

- S'haurà de comptar amb algun sistema de base de dades, ja sigui Redis, PostgreSQL, o d'altres per a la persistència.
- Per a l'emmagatzematge dels fitxers descarregats, anàlisi de sistemes de base de dades relacionals o bé els *buckets* d'amazon S3, amb el seu equivalent open source MinIO.
- Anàlisi i desenvolupament d'una èina d'execució i control de la tasca, potencialment ampliar el projecte *massive_importer*.
- Anàlisi i iniciació en tecnologíes d'*aranyes web* o robots web, potencialment Scrapy o Selenium (open source).

Enfoc

Donat que el projecte està obert a l'anàlisi de diverses eines i procurar que conjuntament donin el resultat esperat, ens trobem amb que sovint les eines que explorarem siguin aquelles que ja s'estiguin fent servir a l'empresa, per conveniència a l'hora de mantenir-ho. Aleshores, com ja s'ha comentat als requisits, podem afirmar a hores d'ara, que l'enfoc serà molt pràctic, en el sentit que si existeix una alternativa coneguda i utilitzada actualment a la cooperativa, prevaldrà sobre d'altres.

Avaluació de viabilitat

En termes de viabilitat, com ja s'ha comentat, donada la infraestructura de servidors actual de *Som Energia*, es pot assumir els tant els recursos necessaris d'aquesta tasca com totes les eines utilitzades (seran de software lliure) no suposaran cap cost significativament addicional. Així doncs passem a analitzar el temps de dedicació en l'automatització contra el temps que suposa actualment aquesta tasca a l'equip de *Som Energia*.

	Temps de dedicació (minuts)	Numero de portals	Total diari (hores)	Total setmanal (hores)
Descarrega casos ATR d'un portal web	5	25	2.1	10.5
Importació ERP casos ATR	1	25	0.5	2.5
Temps en rectificacions	-	-	0.5	2.5
Temps total invertit	**-**	**-**	**3.1**	**15.5**

Fent una aproximació, es descarreguen fitxers sobre 25 portals diàriament: ens trobem amb la xifra de 15.5 hores setmanals d'inversió en aquesta tasca.

Una estimació prudent és pensar que un cop desenvolupada l'eina que ens permeti gestionar els diferents robots webs o aranyes (objectiu principal), suposarà un gran percentatge de l'automatització, podem estimar un 70% del total. Suposant que aquesta eina es dugui a terme en el temps estimat per al PFG (aproximadament 350h), parlem que la totalitat de l'automatització es podria trobar al voltant de 450h.

Amb aquests números, 30 setmanes dedicant 15.5h per setmana equivalen al desenvolupament d'aquesta automatització. O el que és el mateix, 2.5 mesos d'un treballador a jornada de 8h.

Si tenim en compte que els 15 crèdits (ECTS) del PFG es troben dins un marc acadèmic, amb més raó podem veure que es tracta d'una inversió en temps que de seguida passarà a ser beneficiosa a raó de 15.5h setmanals per a la cooperativa.

3 Metodologia

A *Som Energia* es fa servir com a metodologia de desenvolupament una simplificació d'*Scrum*: Es tracta d'una metodologia iterativa i incremental on cada iteració ens dona la possibilitat de reconsiderar requisits, disseny i implementacions.

A més, per a aquest projecte en concret, de la mà del tutor a l'empresa, hem fet una adaptació a un equip reduït de dues persones:

No comptem amb amb rols específics, únicament el del *Mentor* que fa el paper de "Scrum Master" i el "desenvolupador" que en en aquest cas soc jo, l'estudiant. Així doncs hem adaptat una metodologia incremental i iterativa, en "Sprints" que hem anomenat "Rondes", setmanals. El començament d'aquesta implica una reunió o planificació de la ronda. Aquestes rondes compten amb "targetes" referents a funcionalitats o tasques a desenvolupar.

Els Daily *StandUps* podem considerar que els hem fet, d'una forma flexible, donat que hem estat en continu contacte. També hem fet ús de l'eina *Trello* per a la visualització de les diferents "rondes".

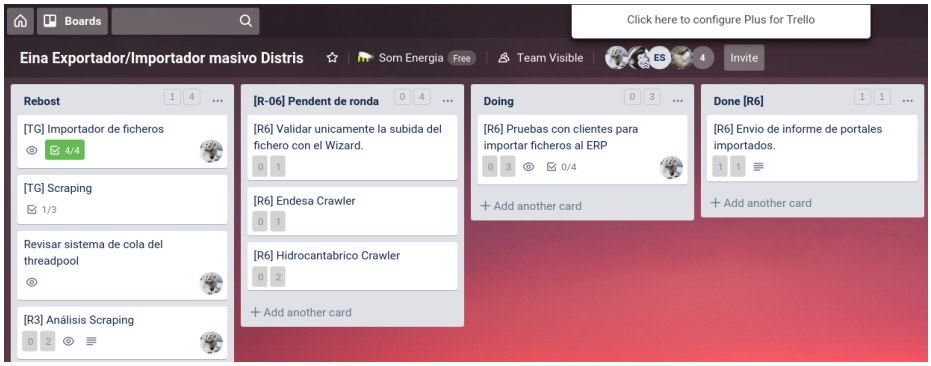


Figure 2: Dashboard de Trello.

L'annex D mostra les diferents "rondes" que es van plantejar, amb les tasques assignades per a cada una d'elles.

4 Planificació

La proposta de realitzar aquesta automatització surt després d'una reunió amb l'equip d'IT de *Som Energia* on es proposen diversos projectes enmarcats dins un propi cotext que impliquen un coneixement acotat i delimitat del funcionament de la cooperativa. D'aquestes necessitats de l'empresa, l'automatització que dona titol a aquest document és de molta prioritat, així com atractiu per a mi, personalment.

Aquest treball és pot pensar en dues parts: El procés de descarrega de fitxers automàticament i el procés de importació (mitjançant l'assistent d'importació) de fitxers al ERP. Si bé al final les dues parts convergiran i acabaran essent un sol projecte, la part de l'automatització de la importació al ERP per si sola, és una tasca que aporta valor amb absència de l'altra. És a dir que es podrà fer una implantació als equips del personal tècnic prèvia a la finalització del projecte, i començar a treure'n rendiment.

Si busquem l'inici del projecte en el qual ens basem, ens remuntem al Gener de 2018, amb massive_importer[4]. No obstant, aquest queda paralitzat en un estat inicial.
El projecte, de la meva mà, es reanuda a mitjans de Març del 2019. No obstant és setmanes abans, cap a finals de Febrer, on comença l'estudi i adaptació a l'entorn de la cooperativa així com al llenguatge de programació utilitzat majoritàriament, Python.

Més enllà del llenguatge, l'unica certesa de la que partíem del projecte *massive_importer* és de l'ús de la llibreria *APScheduler*[8] com a eina de programació de les tasques futures.

Durant el procés d'implementació, van anar sorgint nous elements a tenir en compte, que han estat afegits, per tal de adaptar-se a possibles millores futures. Així doncs, el projecte compta amb una dedicació en gran part del temps per a la modularització.

Per a cada mòdul o funcionalitat s'han fet proves exhaustives del seu funcionament, corregint així els errors trobats, fins a tenir una versió que permeti ser implantada.

Per tal d'ajudar-nos en la coordinació de tasques futures, realitzades i actuals, així com per estimar el temps dedicat a cada feina, hem utilitzat la plataforma de forma gratuïta *Trello* (Als Annexos he adjuntat les 7 "rondes" que hem dut a terme amb els títols de les tasques.).

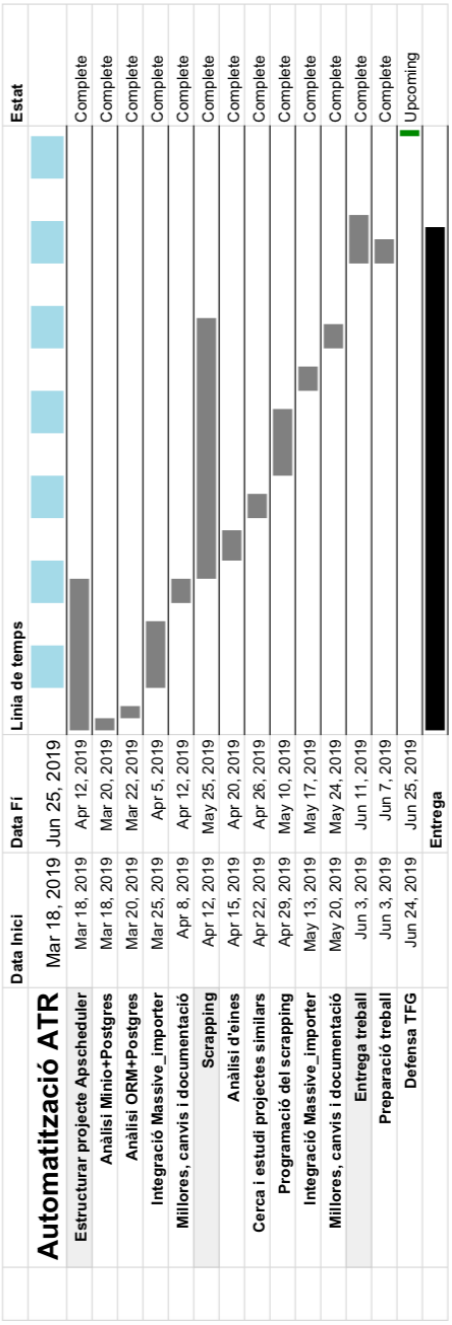


Figure 3: Temporalització inicial del projecte.

Si bé el diagrama de la Figura 2 ens va posar sobre la taula el marge de temps que disposàvem, la planificació o estratègia seguida la podriem representar amb els següents punts:

1. Adquirir el coneixement del llenguatge de programació així com de l'entorn de desenvolupament de *Som Energia*.
2. Diferenciar el projecte en dues parts: descàrrega i importació.
3. Estudi del projecte massive_importer[4] com a punt de partida.
4. Desenvolupament de la primera part: importació.
5. Fer proves de funcionament de la importació.
6. Comprobar la correctesa, i si fos necessari reiterar punts 5 i 6.
7. Desenvolupament de la segona part: descàrrega.
8. Fer proves de funcionament del sistema de descàrrega.
9. Comprobar la correctesa, i si fos necessari reiterar punts 7 i 8.
10. Integració del procés de descàrrega al projecte massive_importer[4].

En aquest treball es realitzarà la descàrrega de almeys un portal web, però no de tots ells, per raons de temps. No obstant, el sistema quedará integrat al projecte de forma que els seguents portals a descarregar no han de seguir la metodología plantejada, sino que unicament s'haura de fer el *crawler*[7] i quedarà integrat de forma molt simple, afegint-lo al directori dels crawlers.

Aquesta estratègia o planificació queda representada amb el següent diagrama:

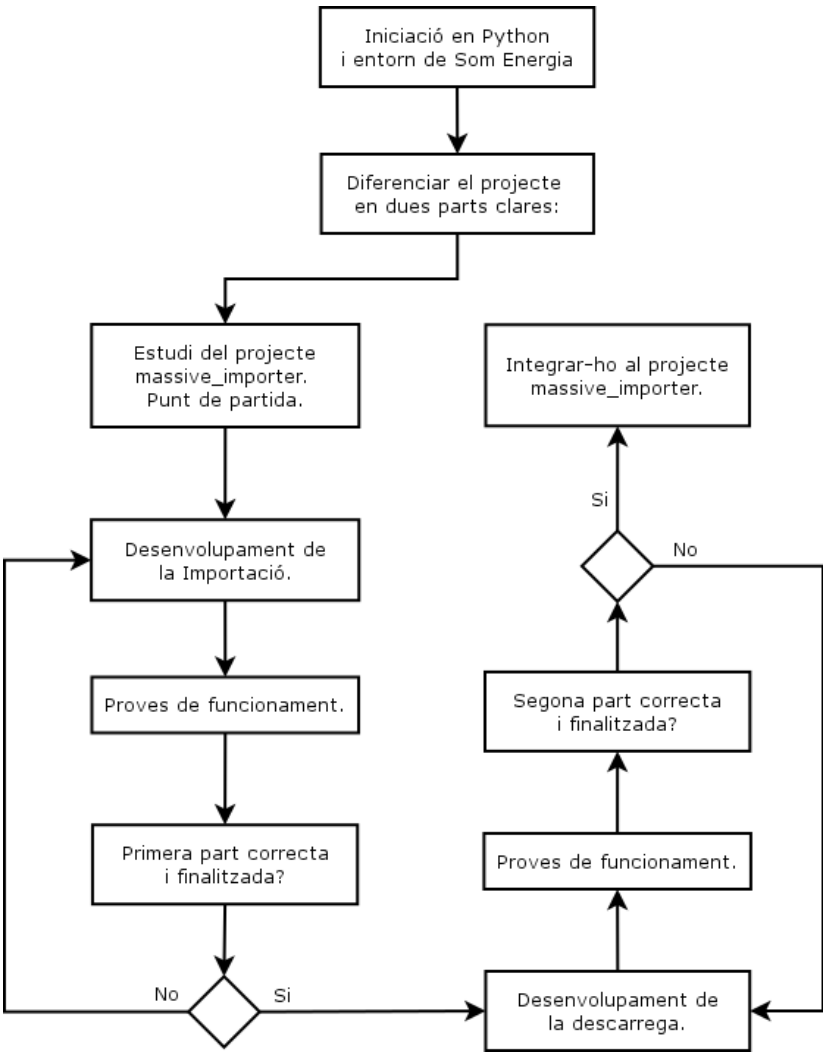


Figure 4: Gràfic planificació.

5 Marc de treball i conceptes previs

Malgrat no és la part més costosa en temps, ni és el més complex, el que dóna títol aquest treball és la interacció amb portals web. Aquesta interacció a efectes pràctics es portarà a terme a través d'eines de **web crawling** o **web scraping**.

5.1 Web Crawling

Un web crawler, també anomenats "aranyes" son programes o bots que tenen com a objectiu escanejar, indexar, descarregar, o qualsevol altre acció d'interés dins la web. Plataformes com Google o Yahoo fan us d'eines d'aquest estil per indexar les seves cerques.

Com veurem més endevant, existeixen formes de crear aquests processos, utilitzant llibreries documentades i àmpliament utilitzades, per moure's dins la web tant a través de comunicacions HTTP com interaccionant amb JS.

Per tal de guardar de forma persistent tant arxius com estats, dates de modificació i altres conceptes, haurem de fer servir un sistema de base de dades. Sense entrar a especificar si farem servir una tecnologia o una altra, parlarem de l'ús d'un **ORM** per modelar els objectes.

5.2 ORM - Mapatge d'objectes relacional

És una tècnica de programació per a convertir dades entre el sistema de base de dades, utilitzant un llenguatge de programació orientat a ovjectes. Utilitza una base de dades relacional com a motor de persistència. A la pràctica, un ORM crea una base de dades virtual, sobre la base de dades relacional. Això possibilitat l'us de característiques pròpies de la orientació a objectes (bàsicament herència i polimorfisme). Hi ha paquets d'us lliure disponibles malgrat alguns programadors prefereixen crear les seves pròpies eines de ORM.

Partint de la idea de notificar en un moment donat l'aparició de nous per descarregar, o altres events, que poden trobar-se en un mateix moment de temps, podem sospitar que ens trobarem amb potencials problemes de concurrència. És per això que parlarem de **thread pooling**.

5.3 Thread Pooling

En programació informàtica, un "pool de subprocesos" és un patró de disseny de programari que té com objectiu aconseguir la concurrència d'execució en un programa. Sovint, també anomenat "model de treballadors" o "tripulants replicats", on un programa de supervisió manté múltiples fils esperant que les tasques s'assignessin a l'execució concurrent del programa de supervisió. Mitjançant el manteniment d'un conjunt de fils, el model augmenta el rendiment i evita la latència en l'execució a causa de la creació i destrucció freqüents de fils per a tasques de curta durada.

El nombre de fils disponibles es sintonitza amb els recursos de computació disponibles per al programa, com ara processadors paral·lels, nuclis, memòria i sòcols de xarxa.
Un mètode comú de planificació de tasques per a l'execució de subprocessos és una cua sincronitzada, coneguda com a cua de tasques. Els fils de la xarxa eliminen les tasques d'espera de la cua i les col·loquen en una cua de tasques completa després d'haver finalitzat l'execució

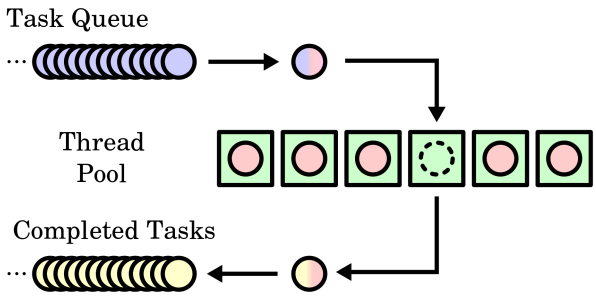


Figure 5: A sample thread pool (green boxes) with waiting tasks (blue) and completed tasks (yellow). https://en.wikipedia.org/wiki/Thread_pool

---

Tanmateix, l'**emmagatzematge basat en objectes** ja s'utilitza a la cooperativa per altres solucions, i és candidat a ser solució de part d'aquesta automatització.

5.4 Emmagatzematge d'objectes

L'emmagatzematge d'objecte (també conegut com emmagatzematge basat en objectes) és un arquitectura qd'emmagatzematge de dades que manega les dades com objectes. Al contrari dels sistemes de fitxers que ho fan com una jerarquía d'arxius i com l'emmagatzematge en bloc que ho fa com blocs dins de sectors i pistes.

Normalment cada objecte inclou les pròpies dades, una quantitat variable de metadades i un identificador global i unic. L'emmagatzematge d'objectes pot ser implementat en multiples nivells, incloent a nivell de dispositiu, de sistema i de interfície.

Escalabilitat, accessibilitat i simplicitat son, tres elements que fan d'aquesta arquitectura un gran atractiu per a solucions d'emmagatzemage.

Partim d'un projecte [4] amb llicència oberta, que podem trobar al repositori **git** *github*. Partirem doncs de git com a **sistema de control de versions**. També cal afegir que dintre del possible (donat que tindrem forces serveis que ho dificultaran) es desenvoluparà amb **TDD**.

5.5 Sistema de Control de Versions

Es diu Control de Versions a la gestió dels diferents canvis que s'han realitzat en els elements d'algun producte o una configuració del mateix. Una versió, revisió o edició d'un producte, és l'estat en que es troba el mateix en un moment concret del seu desenvolupament o modificació.

El Control de Versions es pot realitzar manualment, però és més adequat utilitzar una de les moltes eines disponibles que faciliten aquesta gestió. Aquestes eines s'anomenen Sistemes de Control de Versions, o SVC (System Version Control).

Aquests sistemes faciliten l'administració de les diferents versions de cada producte desenvolupat, així com les possibles especialitzacions realitzades.
La cooperativa *Som Energia* utilitza Git com a sistema de control de versions, a més allibera tot el codi al seu perfil de *github* Som-Energia.

5.6 Desenvolupament guiat per proves (TDD)

Desenvolupament guiat per proves de programari, en anglès: Test-driven development (TDD), és una pràctica d'enginyeria de programari que involucra unes altres dues pràctiques: Escriure les proves primer (Test First Development) i Refacció (Refactoring). Per escriure les proves generalment s'utilitzen les proves unitàries (unit test en anglès). En primer lloc, s'escriu una prova i es verifica que les proves fallen.

A continuació, s'implementa el codi que fa que la prova passi satisfactòriament i seguidament es refacciona el codi escrit. El propòsit del desenvolupament guiat per proves és aconseguir un codi net que funcioni.
La idea és que els requisits siguin traduïts a proves, d'aquesta manera, quan les proves passin es garantirà que el programari compleix amb els requisits que s'han establert.

Com a pauta, *Som Energia* incorpora aquest model de treball. Pel que fal al desenvolupament d'aquest projecte, no s'ha seguit de forma estricta en alguns mòduls.

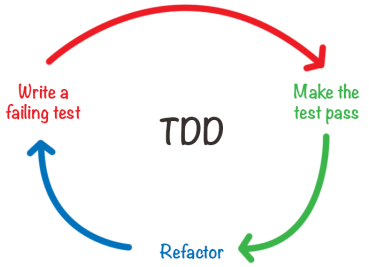


Figure 6: Cicle del TDD. https://github.com/foundersandcoders/testing-tdd-intro

6 Requisits del sistema

6.1 Requeriments funcionals

A la introducció d'aquest document hem mencionat els dos actors que considerarem. Per una banda tenim els anomenats *usuaris* que son el conjunt de l'equip tècnic, i d'altra banda l'*administrador* que serà l'actor encarregat de supervisar, executar i configurar el programa. L'actor *temps*, donat que parlem de tasques programades al llarg del temps, és el responsable d'accionar els diferents processos del programa. A partir d'aquests tres actors, tot seguit es descriuran els requisits funcionals del nostre programa:

6.1.1 Iniciar procés de descarrega dels portals

En funció de com s'hagi programat la tasca, el *temps* engega un procés que ha d'activar el procés de descarrega per a les webs seleccionades a la configuració.

6.1.2 Iniciar procés d'importació al ERP

En funció de com s'hagi programat la tasca, el *temps* engega per a cada element nou descarregat (que encara no ha estat importat), un procés d'importació cap al ERP.

6.1.3 Generació de resum d'importacions i pendents

En funció de com s'hagi programat la tasca, el *temps* engega un procés que genera un resum de totes les descarregues efectuades i de tots els casos importats amb èxit, error, o pendents.

6.1.4 Visualitzar dels resum d'importacions

Un *usuari* ha de tenir accés a un resum dels casos que s'han importat durant el darrer període de temps (període concretat per l'*administrador*), per coneixer l'estat actual de la base de dades del ERP. En cas d'haver surgit algun error en la importació durant aquest periode, de la mateixa forma s'ha de tenir constància.
En cas d'existir processos que encara no s'han iniciat, és a dir, fitxers que encara es troben exclusivament al *bucket* pero encara no s'han importat al ERP, els *usuaris* han de tenir constància.

6.1.5 Configuració dels portals a incloure en el procés

El programa ha de permetre a l'*l'administrador* establir una configuració on es pugui afegir i treure els portals dels quals es vulgui efectuar les descarregues massives.

6.1.6 Gestió de credencials

El programa ha de permetre una gestió de credencials, donat que sovint aquests son modificats per raons de seguretat. Així doncs, l'*administrador* ha de poder accedir a aquesta configuració.

6.1.7 Gestió de la programació de tasques

L'*administrador* ha de poder canviar, ja sigui en forma de interval o bé de hores concretes durant el dia, dies durant el mes, etc. De forma que la programació de les tasques no depengui del programa sinó d'una configuració que pugui ser determinada per aquest actor.

6.1.8 Configuració de les connexions

Es vol disposar d'un sistema que permeti configurar fàcilment les diferents connexions, tant a la BD, com la connexió al ERP, aixi com a qualsevol altre servei que s'utilitzi. L'*administrador* ha de tenir accés a la configuració de les diverses connexions, independentment del codi del programa.

6.1.9 Supervisió dels events del programa

L'*administrador* ha de tenir accés a un fitxer de Log on es reflexin els esdeveniments com excepcions, errors, etc.

6.2 Requeriments no funcionals

- Disposar d'un procediment que automatitzi la descarrega de fitxers dels casos ATR i posteriorment els importi al ERP.
- Es precisa un sistema d'emmagatzematge escalable.
- Mantenir de forma persistent i segura la informació dels estats d'importació dels diferents fitxers importats al ERP.
- Es vol aconseguir un programa modular, que de forma separada es pugui aprofitar per a futures ampliacions.

7 Estudis i decisions

En un primer moment, partim de la idea que s'ha de realitzar unes tasques que seran executades amb un cert interval o bé amb un *cron* fent servir APScheduler. Com ja s'ha comentat anteriorment, el projecte *massive_importer* inicialment l'únic que té definit és el sistema que permet programar les tasques.

Més enllà d'això, s'havien de prendre decisions de com estructurar uns sistema que permetés mantenir d'una forma persistent els arxius es descarreguen, i a l'hora de pujar-los al ERP, una gestió dels diversos estats en cas d'estar en procés, estat d'error, etc.

La decisió presa després d'una reunió entre el meu tutor en pràctiques, altres membres de l'equip d'IT: es va optar per una estructura que comptaria amb els següents components:

7.1 PostgreSQL

PostgreSQL és un programari lliure que implementa un sistema de gestió de bases de dades relacional. És una alternativa a altres sistemes de gestió de base de dades i de sistemes de programari propietari com Oracle, Microsoft SQL Server, etc.

La raó per la que es tria PostgreSQL és per no fer aumentar l'*stack* de coneixement de *Som Energia*. En altres paraules, per un manteniment més viable, fem servir una eina que ja es troba present en la major part de processos de la cooperativa.

Figure 7: Logotip de PostgreSQL.

Sovint ens anem trobant en situacions on el fet de prendre una alternativa vers una altra, no depèn tant el valor que aporta al projecte en sí actualment, sinó de les possibilitats futures i al fet de fer més didàctic l'exercici d'aquest PFG. Dit això, podíem haver modelat el sistema prescindint d'un ORM, però vam decidir incorporar-lo per les dues raons mencionades. En aquest cas però, s'utilitzarà la coneguda llibreria PonyORM.

7.2 PonyORM [13]

Pony és un 'object-relational mapper' avançat. Un ORM permet als desenvolupadors treballar amb el contingut d'una base de dades en forma d'objectes. Si bé hem triat aquest i no un altre, és per:

- Sintaxis simple i entenedora.
- Optimització de queris automàtiques.
- Eina online d'edició d'esquemes.

Figure 8: Logotip de PonyORM.

7.3 Minio [9]

Es tracta d'un sistema d'emmagatzematge de fitxers escalable. Ens permet tractar qualsevol fitxer com si d'un objecte es tractés, a través de la seva API, adaptada entre d'altres a Python.

Minio és una eina equivalent a Amazon S3. La raó de Minio és el software lliure, donat que és en tots els aspectes un *clon* de Amazon S3, tant pel que fa a la gestió com a la compatibilitat amb d'altres serveis. Tanmateix, és una eina eficient, pràctica i preparada per a llençar events.

Un altre punt a favor és que aquesta tecnologia ja existeix dins la cooperativa, utilitzada en diversos projectes.

Figure 9: Logotip de MinIO.

Pel que fa a la segona part del projecte, la descàrrega massiva de fitxers dels diferents portals, teniem sobre la taula diverses opcions. No obstant, finalment es van descartar vàries per raons multiples, deixant les d'Scrapy i Selenium. Aquesta segona amés, tenia l'atractiu d'un projecte començat [5] que ens podia haver servit de pauta per als diferents crawlers.

En una primera instància es decideix, però, optar per Scrapy.

7.4 Scrapy[10]

Scrapy és probablement la llibreria de Python més utilitzada per al web crawling. Es tracta d'una eina molt potent, que inicialment ens permet fer tot tipus de peticions Web.

Si bé Scrapy per si sol, unicament treballa amb peticions HTTP, pertant no executa javascript ni renderitza *a posteriori*, no és cap limitació si tenim en compte la quantitat de *middlewares* (exetensions) que existeixen. Aquestes extensions poden fer multiples funcions, com és el cas d'*Splash*[11] per tal de poder executar JS.

Figure 10: Logotip de Scrapy.

8 Anàlisi i disseny del sistema

8.1 Anàlisi

El nostre programa té un cicle de vida que comença en la descarrega d'un fitxer nou (ja sigui manualment en una primera instància o automatitzat al final del projecte). Aquest nou fitxer es guarda en un *bucket* de MinIO, el qual genera un Event que serà emmagatzemat en una taula de la nostra BD PostgreSQL. Moments després s'inicia el procediment que verifica si existeixen nous Events, no tractats, i en cas de ser així, iniciar el procés d'importació cap al ERP.

Si bé al final del projecte no faria falta cap altre element, la primera implantació, i mentre no es compti amb tots els *crawlers* de tots els portals webs, amés comptaria amb el muntatge d'aquesta unitat de MinIO als equips del personal tècnic, fent servir el sistema de fitxers *FUSE* i la compatibilitat de MinIO amb aquest.

D'aquesta manera, encara es faria de forma manual la descarrega de certs fitxers, però l'exercici d'importació al ERP passaria a ser únicament desar la descarrega a la unitat muntada.

Un esquema d'aquesta idea sorgida de les primeres reunions:

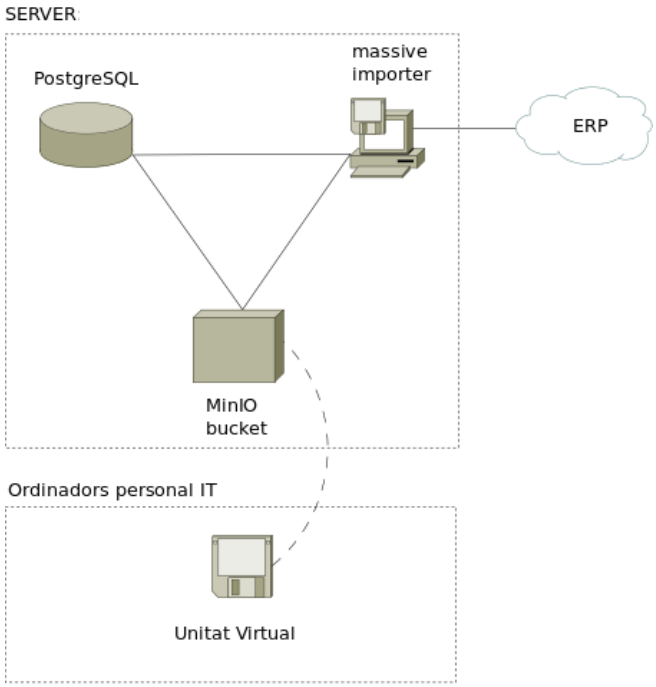


Figure 11: Arquitectura del projecte.

Per fer-nos una idea del flux de les dades, podem entendre-ho de la següent forma:

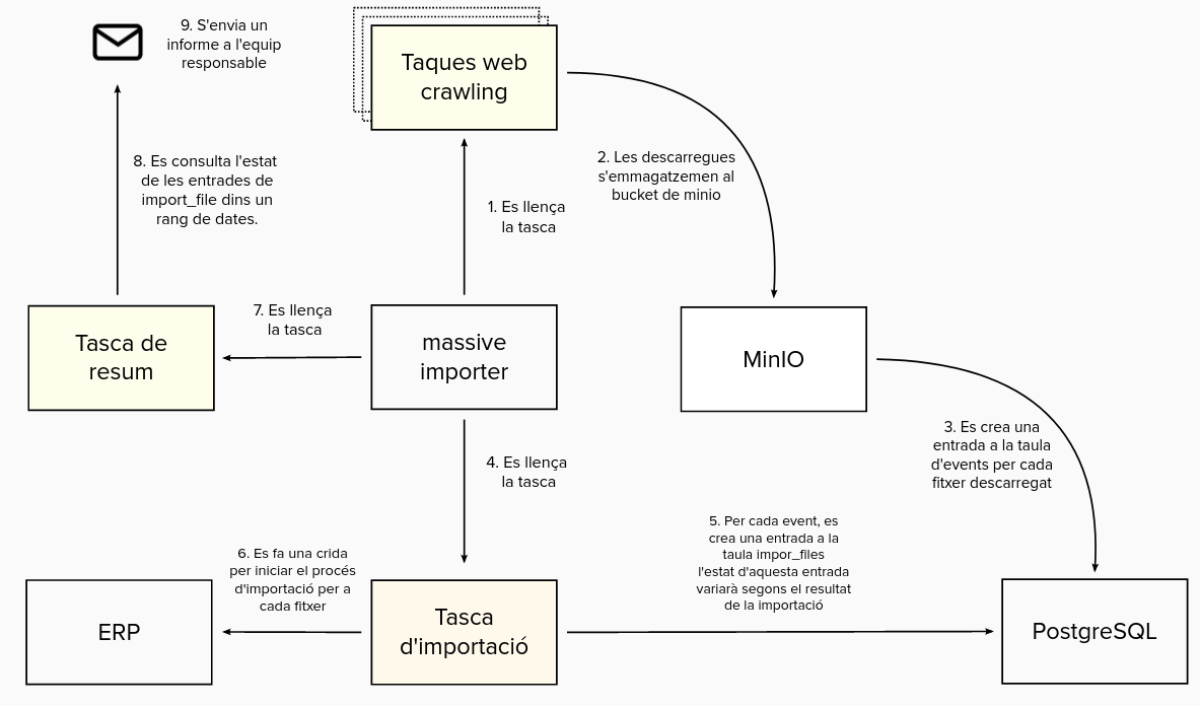


Figure 12: Flux de dades.

Hem de tenir en compte que el Servidor de MinIO, les tasques del *massive_importer* i el servidor PostgreSQL, seran 3 màquines diferents un cop a producció.

El que fins ara haviem considerat com necessitats dels *usuaris*, realment no impliquen una interacció directa amb el sistema, tret de la visualització dels resums de les importacions. Donat que es tracta d'un procés programat en el temps, l'actor que efectua o que inicia les diferents accions és el *temps*. L'*adminsitrador* en canvi, ha de poder realitzar diverses accions directament. No obstant, aquestes no es trobarán disponibles a través d'una interfície sinó que a través de fitxers específics de configuració.

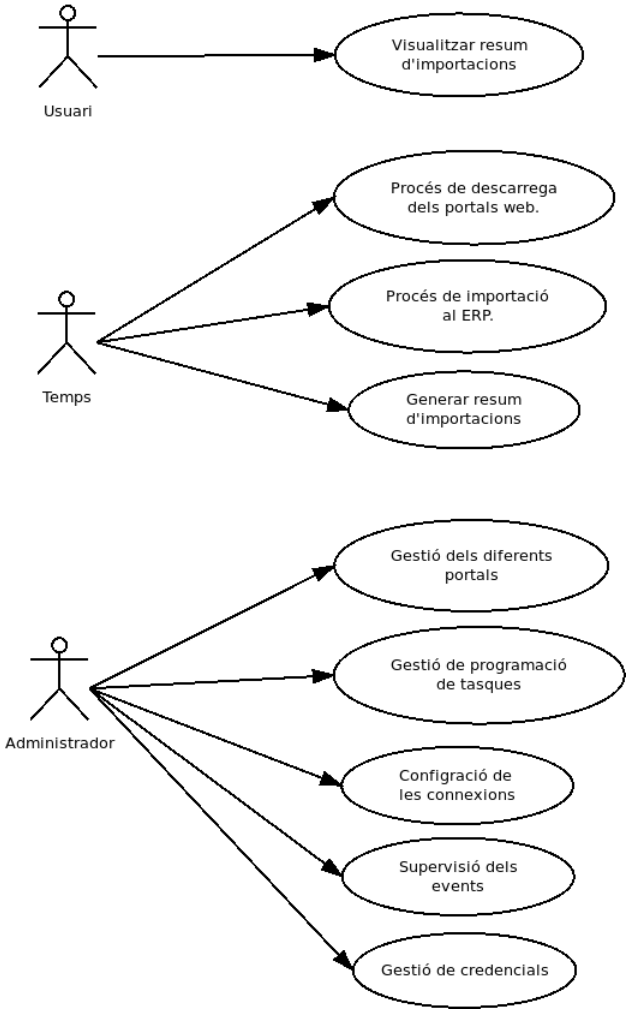


Figure 13: Diagrama de casos d'ús

Cas d'us: Procés d'importació al ERP

El seguiment d'un fitxer descarregat d'un portal web el podem fer inicialment quan es crea a la taula de *Events*. Tot seguit, quan comença el procés de importació, passarà a tenir una entrada en una segona taula *ImportFiles* a on es modificarà l'estat. D'aquesta darrera taula no s'esborrarà l'entrada, sino que mantindrem aquesta informació persistentment.

Descripció	Procés d'importació al ERP.
Actor	Temps
Precondició	El procés ha estat programat en el temps amb anterioritat.
Flux principal	1. Per a cada Event a la taula de Events de la base de dades 1.1. Afegir entrada a la taula de ImportFiles amb estat READY 1.2. Eliminar entrada de la taula de Events. 1.3. Selecciona el fitxer corresponent al bucket de MinIO 1.4. Inicia importació al ERP 1.5. Modificar entrada a la taula de ImportFIles amb estat PROCESS 1.6. Si finalitza correctament, estat passa a ser OK 1.7. Altrament estat passa a ser ERROR.
Postcondició	

8.2 Disseny proposat

8.2.1 Entitat *Event*

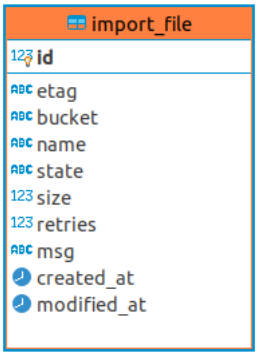
Per a l'entitat Event, els unics camps que voldrem son: *key* que es tractarà del nom del objecte del bucket (fitxer .zip) i *value* que contindrà tots els valors en detall d'aquest. PostgreSQL ens permet utilitzar el format de dades *Json*.

Listing 1: Enitat Event modelada amb PonyORM

```
class Event(db.Entity):
    _table_ = 'events'
    key = PrimaryKey(str, auto=False)
    value = Required(Json)
```

8.2.2 Entitat *ImportFile*

Per a l'entitat *ImportFile* bàsicament despleguem els valors que ens interessen continguts al Json que ens ha generat l'event de MinIO, de la següent forma:



Listing 2: Enitat ImportFile modelada amb PonyORM

```
class ImportFile(db.Entity):
    _table_ = 'import_file'
    id = PrimaryKey(int, auto=True)
    etag = Required(str, unique=True)
    bucket = Required(str)
```

```
    name = Required(str, index='index_importfile_name')
    state = Required(
        str,
        default=UpdateStatus.RDY_TO_PROCESS,
        index='index_importfile_state'
    )
    size = Required(float)
    retries = Required(int, default=0)
    msg = Optional(str)
    created_at = Required(
        datetime,
        index='index_importfile_created_at',
        sql_default='CURRENT_TIMESTAMP'
    )
    modified_at = Required(
        datetime,
        index='index_importfile_modified_at',
        default=datetime.now
    )
```

8.2.3 Entorns de desenvolupament

En aquest treball i des dels origens del projecte *massive_importer*[4] s'ha fet molta èmfasi en la modularització. A més, en tractar-se d'un projecte de codi obert, com tots els projectes de *Som Energia* (els quals es troben a `https://github.com/Som-Energia`) ens obliga a utilitzar un sistema que incorpori de forma independent credencials i dades restringides, amb la resta de codi.

És aqui on neix doncs la idea de separar els fitxers de configuració dins el directori **massive_importer/conf/**. El sistema proposat és la utilització d'un fitxer *yaml* anomenat *config.yaml* que conté els paràmetres de configuració per a la connexió amb la BD, MinIO, ERP, mail així com el llistat de portals web pendents de descarregar, entre d'altres.

```
conf/
├── config.yaml.example
├── envs
│   ├── base.py
│   ├── devel.py
│   ├── __init__.py
│   ├── prod.py
│   └── test.py
├── __init__.py
└── startup_configuration.py
```

Figure 14: Directori d'entorns i configuracions.

Els fitxers específics *test.py, devel.py, prod.py* d'entrada carreguen la configuració establerta al fitxer *base.py*, que a l'hora carrega el contingt del fitxer *.yaml*. No obstant, podem modificar aquells paràmetres que siguin diferents del nostre entorn. Per exemple, per l'entorn de testing, voldrem connexions en *localhost*.

9 Implementació i proves

9.1 Preparació de l'entorn

En primer lloc, com hem parlat a la secció 5. Marc de treball i conceptes previs, hem hagut de configurar a la meva estació de treball les diferents eines per al correcte desenvolupament del projecte. Si bé no hi ha pràcticament cap directriu establerta, el meu entorn, entre consell per part de l'equip d'IT, com per pròpia cerca, compto amb els següents elements, entre d'altres:

- *Python 3*. Així com *pip* i l'eina d'entorn virtual *pipenv*.
- *VSCode*. Com a eina de codi obert, i per simplicitat davant la proposta de l'us de *vim* com a editor standard a *Som Energia*.
- *Gitkraken* [14]. Per a facilitar la gestió del sistema de control de versions.
- *Postman* [15]. Per a les proves a l'hora de realitzar programació de Crawlers.

9.2 Implementació del massive_importer

Per tal de configurar el Sistema d'una forma modular i reaprofitable, s'han dividit les diferents utilitats en forma de fitxers o mòduls separats, que es van incorporant a les tasques a mida que es necessiti. De tots ells s'ha desenvolupat testos de forma que es comprova el seu correcte funcionament.

9.3 Mòdul del MinIO

Aquest mòdul utilitza la llibreria *Minio* que disposa de tots els mètodes necessaris per a llistar objectes, descarregar, pujar, i eliminar objectes donat un bucket i nom de l'objecte (ruta sencera en cas de trobar-se en un directori, donat que *nomcarpeta/fitxer.zip* és un objecte i *fitxer.zip* n'és un altre, al mateix nivell).

Aquest modul ha estat pensat per tal de ser instanciat passant per paràmetres la configuració per a la connexió, fent doncs un objecte o *Manager* que sigui el responsable de tota la interacció.

Proves del mòdul

En aquest cas la prova efectuada (Unittest) fa ús de diversos mètodes:

1. Es configura la connexió amb el servidor MinIO
2. Es crida el mètode per a esborrar tot el contingut del bucket
3. Es crea un nou fitxer anomenat *prova.zip*
4. Es fa un get del fitxer *prova.zip*
5. Es crida el mètode per a esborrar tot el contigut del bucket
6. Es validen totes les passes.

```
(massive_importer) somenergia@PC128:~/Documents/somenergia/massive_importer
$ python -W ignore -m unittest discover tests/ -p test_minio* -v
test_create_manager (lib.test_minio.TestMinioManager) ... ok
test_put_and_get_file (lib.test_minio.TestMinioManager) ... ok

----------------------------------------------------------------------
Ran 2 tests in 0.036s

OK
```

Figure 15: Test MinIO module.

9.4 Mòdul de la BD

Aquest mòdul conté mètodes de consulta i modificació de les taules Events i Imporfiles de la base de dades. Donat que utilitzen la llibreria de *PonyORM*, aquest mètodes s'encapçalen amb un *decorator* anomenat **db_session** que té la funció de guardar en memòria local tots els canvis que es realitzin fis a la finalització de la funció que el crida. En cas de haver anat to bé, s'efectuen els canvis. En cas d'existir algun error, aquest pot fer com si fós un rollback, sense haver afectat realment en cap moment la BD.

Proves del mòdul

En aquest cas les proves efectuades (Unittest) són:

1. Es configura la connexió amb el la Base de Dades
2. Es Creen les taules
3. Es Populen les taules amb elements invariables
4. Es Transformen Events en ImportFiles
5. Es Modifica l'estat dels ImportFiles
6. Es validen totes les passes.

```
(massive_importer) somenergia@PC128:~/Documents/somenergia/massive_importer
$ python -W ignore -m unittest discover tests/ -p test_db* -v
test_eventToImportFile (lib.test_db.TestDbUtils) ... ok
test_listEvents (lib.test_db.TestDbUtils) ... ok
test_updateState (lib.test_db.TestDbUtils) ... ok

----------------------------------------------------------------------
Ran 3 tests in 0.071s

OK
```

Figure 16: Test DB module.

9.5 Mòdul d'Alertes

El mòdul d'alertes és un petit fitxer que també s'ajusta a la idea vista amb la gestió de MinIO. És a dir, creem una classe "AlertManager" que un cop instanciada amb els paràmetres de configuració, queda disponible per a les seves funcionalitats.

En aquest cas, fins al moment només s'han contemplat dues funcionalitats: Per una banda l'enviament d'un missatge estàtic de test. D'altra banda un resum de l'estat de les dues taules de la Base de dades, especificant en cada cas si es tracta de casos pendents de importar, casos ja importats amb el seu respectiu estat.

Fem us de la llibreria *smtplib* la qual ens permet efectuar una connexió amb un servidor de correus SMTP de Google.

Proves del mòdul

En aquest cas la prova efectuada (Unittest) és:

1. Es configura la connexió amb el servidor mail
2. Es conforma un missatge
3. S'envia el missatge
4. Es tenca la connexió amb el servidor mail
5. Es valida l'enviament del mail.

```
(massive_importer) somenergia@PC128:~/Documents/somenergia/massive_importer
$ python -W ignore -m unittest discover tests/ -p test_alert* -v
test_alert_send (lib.test_alert.AlertTest) ... ok

----------------------------------------------------------------------
Ran 1 test in 1.638s

OK
```

Figure 17: Test Alert module.

9.6 Mòdul del ERP

Aquest mòdul com altres anteriorment, està ideat per a ser instanciat passant els parametres de configuració, de forma que queda establerta una connexió amb el ERP allotjat en un sevidor extern. A partir d'aqui, cridant als mètodes d'aquesta instància podem interactuar amb l'ERP. De fet la unica funcionalitat a data d'avui és la de importar fitxers cap a l'ERP.

Una de les limitacions que vam trobar a l'hora d'implementar aquesta funcionalitat és que la idea de fer importacions de forma *multithread* (en paral·lel) no és del tot viable, donat que hi ha un recurs del ERP que és exclusiu i no admet dues operacions a l'hora.

Aquest control doncs l'he dut a terme a partir d'un semàfor d'exclusió, que es bloqueja durant el procés crític mencionat, acte seguit el deixa anar. Així doncs tenim una concurrencia parcial dels processos.

Proves del mòdul

En aquest cas les proves efectuades (Unittest) són:

1. Es configura la connexió amb el servidor ERP
2. Es carrega en memòria un fitxer anomenat *proves.zip*
3. S'envia a través de la llibreria WizarImportAtrF1
4. Es valida la resposta del servidor.

```
(massive_importer) somenergia@PC128:~/Documents/somenergia/massive_importer
$ python -W ignore -m unittest discover tests/ -p test_erp* -v
test_import_wizard (lib.test_erpmanager.TestErpManager) ... /home/somenergi
a/Documents/somenergia/massive_importer/tests/data/prova.zip
ok

----------------------------------------------------------------------
Ran 1 test in 17.103s

OK
```

Figure 18: Test ERP module.

9.7 Tasques

Sense entrar en detall del projecte Scrapy, el qual veurem en un apartat següent, comptem amb una funció o tasca que és l'encarregada d'iniciar les diferents *Aranyes*. Aquest procés genera una sèrie de fitxers que a l'hora registren la seva activitat a la base de dades. A aquest primer procés li he dit *web_crawling*

És aqui on entra en joc la segona tasca, *check_new_events*, que és l'encarregada d'identificar que hi ha aquests nous fitxers, i inicia la importació cap al ERP a l'hora que manté actualitzat l'estat d'aquest procés a la base de dades.
Per ultim tenim la tasca de Resums (*summary*) de casos durant el darrer dia: es tracta de filtrar la informació que ens interessa dels ultims casos d'importació (tan pendents, com fets erronis o fets correctament) i posteriorment enviar-ho per correu al personal tècnic de la cooperativa.

Així doncs aquest mòdul de tasques és el punt de trobada de tots els mòduls anteriors. L'execució d'aquestes vindrà programada (modificable per l'administrador) d'entrada per una execució diària tal que primer es s'executin amb un cert temps de diferència en el següent ordre:

1. *web_crawling*
2. *check_new_events*
3. *summary*

web_crawling

Per tal d'integrar el projecte Scrapy al *massive_importer* s'ha creat un mòdul (és endevant el veurem en detall) anomenat *run_crawlers*. Aquest mòdul s'instancia inicialment. Així doncs, *web_crawling* unicament crida un mètode d'aquesta instancia, que és la que dispara el procés de crawling.

check_new_events

Entrant més en detall en aquesta tasca, el que veurem en primer lloc és la vida que segueix un arxiu, amb l'event que dispara en descarregar-se. Així doncs la decisió d'utiltizar un threadpool (Secció 5.3) ens permet d'entrada recorrer la taula d'Events, i per a cada element trobat, afegir un nou procés a la cua del threadpool, que (tenint en compte el semàfor que restringeix el recurs del ERP) intentarà fins a un numero fixat d'intents l'acció (per defecte 3).

En cas de reincorporar un procés a la cua de processos, aquest procés genera una nova cua de processos per ell sol, en lloc de posar-se al darrere dels demés. No obstant no interfereix en el recurs unic de l'ERP, donat que en tot moment es tracta amb un sol semàfor.
Aquesta tasca ha quedat pendent de refactorització des de la ronda R5, com es pot veure a la Secció 4 de planificació.

summary

En aquesta tasca, fem servir el mòdul d'Alertes i el de gestió de la base de dades. Així doncs, la tasca tot el que fa es llistar les entrades a la taula d'Events (la qual cosa ens indica que hi ha nous fitxers sense processar), i la taula de ImportFiles filtrant per la data del darrer dia, amb el

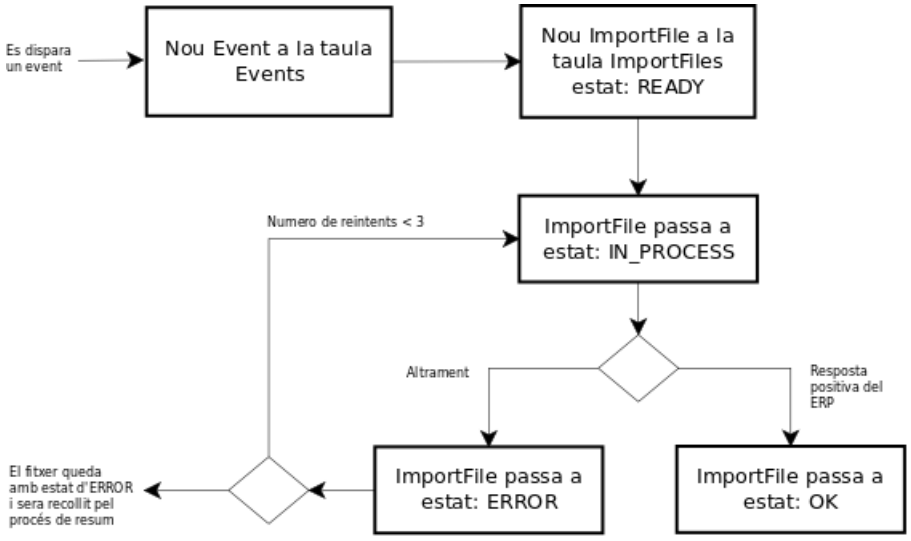


Figure 19: Diagrama de la vida d'un arxiu.

corresponent estat de cada fitxer.

Amb totes aquestes dades, es crea un missatge que s'enviarà a través del mòdul d'alertes, al correu que s'hagi configurat al programa.

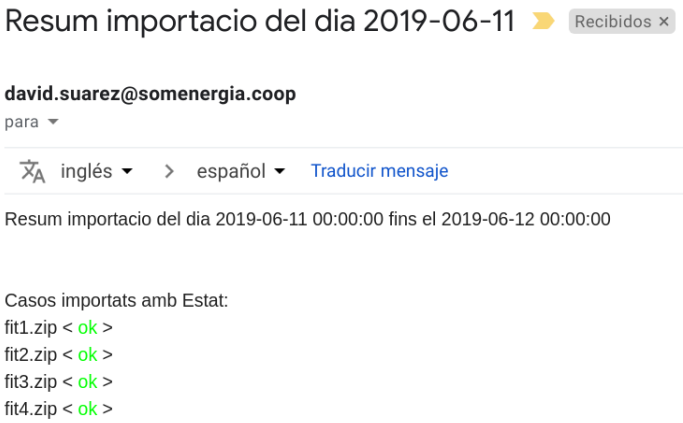


Figure 20: Exemple de missatge enviat per *summary*

Proves de les tasques

**IMPORTANT*: El fitxer de proves per a les tasques, anomenat *test_tasks.py* és un fitxer que fa proves destructives, és a dir que esborra contingut del bucket de MinIO com de les taules de la base de dades corresponents als fitxers de configuració de l'entorn que s'està executant. És important que per executar els testos, i aquest concretament, ens trobem a l'entorn de test.

En aquest cas les proves efectuades (Unittest) són:

1. Es configura la connexió amb el servidor MinIO
2. Es configura la connexió amb el servidor ERP (*Mock*)
3. Es configura la connexió amb la BD
4. Es crida un metode de neteja (S'esborren tots els objectes del bucket i totes les entrades tant la taula Events com a la taula ImportFiles)
5. Es pujen els fitxers *file1.zip*,*file2.zip*,*file3.zip* i *file4.zip* al bucket de minio
6. Es crida *test_web_crawling* (Obre un fitxer de proves i el puja al bucket)
7. Es crida *test_check_new_events* (Pujar tots els arxius pendents al ERP)
8. Es crida *test_summary* (Amb l'estat actual de la BD, s'envia un resum)
9. Es torna a cridar el mètode de neteja
10. Es validen totes les passes.

Com podem veure a la figura 49 la terminal també ens està mostrant el procés de Crawling, el qual es *verbose* i en iniciar el motor d'Scrapy ens està donant la informació de la configuració, des de la gestió de cookies, el User-Agent usat, entre molts altres.

```
(massive_importer) somenergia@PC128:~/Documents/somenergia/massive_importer$ python -W ignore -m unittest discover tests/ -p
 test_tasks* -v
test_check_new_events (importer.test_tasks.TestTasks) ... [Event['zips/fit1.zip'], Event['zips/fit2.zip'], Event['zips/fit3.
zip'], Event['zips/fit4.zip']]
ok
test_summary (importer.test_tasks.TestTasks) ... ok
test_web_crawling (importer.test_tasks.TestTasks) ... 2019-06-11 19:27:43 [scrapy.utils.log] INFO: Scrapy 1.6.0 started (bot
: scrapybot)
2019-06-11 19:27:43 [scrapy.utils.log] INFO: Versions: lxml 4.3.4.0, libxml2 2.9.9, cssselect 1.0.3, parsel 1.5.1, w3lib 1.2
0.0, Twisted 19.2.1, Python 3.6.7 (default, Oct 22 2018, 11:32:17) - [GCC 8.2.0], pyOpenSSL 19.0.0 (OpenSSL 1.1.1c  28 May 2
019), cryptography 2.7, Platform Linux-4.15.0-51-generic-x86_64-with-Ubuntu-18.04-bionic
2019-06-11 19:27:43 [scrapy.crawler] INFO: Overridden settings: {}
2019-06-11 19:27:43 [scrapy.extensions.telnet] INFO: Telnet Password: 308b78ee3bacdd92
2019-06-11 19:27:43 [scrapy.middleware] INFO: Enabled extensions:
['scrapy.extensions.corestats.CoreStats',
 'scrapy.extensions.telnet.TelnetConsole',
 'scrapy.extensions.memusage.MemoryUsage',
 'scrapy.extensions.logstats.LogStats']
2019-06-11 19:27:43 [scrapy.middleware] INFO: Enabled downloader middlewares:
['scrapy.downloadermiddlewares.httpauth.HttpAuthMiddleware',
 'scrapy.downloadermiddlewares.downloadtimeout.DownloadTimeoutMiddleware',
 'scrapy.downloadermiddlewares.defaultheaders.DefaultHeadersMiddleware',
 'scrapy.downloadermiddlewares.useragent.UserAgentMiddleware',
 'scrapy.downloadermiddlewares.retry.RetryMiddleware',
 'scrapy.downloadermiddlewares.redirect.MetaRefreshMiddleware',
 'scrapy.downloadermiddlewares.httpcompression.HttpCompressionMiddleware',
 'scrapy.downloadermiddlewares.redirect.RedirectMiddleware',
 'scrapy.downloadermiddlewares.cookies.CookiesMiddleware',
 'scrapy.downloadermiddlewares.httpproxy.HttpProxyMiddleware',
 'scrapy.downloadermiddlewares.stats.DownloaderStats']
2019-06-11 19:27:43 [scrapy.middleware] INFO: Enabled spider middlewares:
['scrapy.spidermiddlewares.httperror.HttpErrorMiddleware',
 'scrapy.spidermiddlewares.offsite.OffsiteMiddleware',
 'scrapy.spidermiddlewares.referer.RefererMiddleware',
 'scrapy.spidermiddlewares.urllength.UrlLengthMiddleware',
 'scrapy.spidermiddlewares.depth.DepthMiddleware']
2019-06-11 19:27:43 [scrapy.middleware] INFO: Enabled item pipelines:
[]
2019-06-11 19:27:43 [scrapy.core.engine] INFO: Spider opened
2019-06-11 19:27:43 [scrapy.extensions.logstats] INFO: Crawled 0 pages (at 0 pages/min), scraped 0 items (at 0 items/min)
2019-06-11 19:27:43 [scrapy.extensions.telnet] INFO: Telnet console listening on 127.0.0.1:6023
2019-06-11 19:27:43 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://www.example.com> (referer: None)
2019-06-11 19:27:43 [urllib3.connectionpool] DEBUG: Starting new HTTP connection (1): localhost:9000
2019-06-11 19:27:43 [urllib3.connectionpool] DEBUG: http://localhost:9000 "GET /zips?location= HTTP/1.1" 200 None
2019-06-11 19:27:43 [urllib3.connectionpool] DEBUG: http://localhost:9000 "PUT /zips/11-06-2019/prova.zip HTTP/1.1" 200 0
2019-06-11 19:27:43 [scrapy.core.engine] INFO: Closing spider (finished)
2019-06-11 19:27:43 [scrapy.statscollectors] INFO: Dumping Scrapy stats:
{'downloader/request_bytes': 214,
 'downloader/request_count': 1,
 'downloader/request_method_count/GET': 1,
 'downloader/response_bytes': 935,
 'downloader/response_count': 1,
 'downloader/response_status_count/200': 1,
 'finish_reason': 'finished',
 'finish_time': datetime.datetime(2019, 6, 11, 17, 27, 43, 827095),
 'log_count/DEBUG': 4,
 'log_count/INFO': 9,
 'memusage/max': 67981312,
 'memusage/startup': 67981312,
 'response_received_count': 1,
 'scheduler/dequeued': 1,
 'scheduler/dequeued/memory': 1,
 'scheduler/enqueued': 1,
 'scheduler/enqueued/memory': 1,
 'start_time': datetime.datetime(2019, 6, 11, 17, 27, 43, 191361)}
2019-06-11 19:27:43 [scrapy.core.engine] INFO: Spider closed (finished)
ok
2019-06-11 19:27:43 [urllib3.connectionpool] DEBUG: http://localhost:9000 "GET /zips/?max-keys=1000 HTTP/1.1" 200 None
2019-06-11 19:27:43 [urllib3.connectionpool] DEBUG: http://localhost:9000 "DELETE /zips/11-06-2019/prova.zip HTTP/1.1" 204 0
2019-06-11 19:27:43 [urllib3.connectionpool] DEBUG: http://localhost:9000 "DELETE /zips/fit1.zip HTTP/1.1" 204 0
2019-06-11 19:27:43 [urllib3.connectionpool] DEBUG: http://localhost:9000 "DELETE /zips/fit2.zip HTTP/1.1" 204 0
2019-06-11 19:27:43 [urllib3.connectionpool] DEBUG: http://localhost:9000 "DELETE /zips/fit3.zip HTTP/1.1" 204 0
2019-06-11 19:27:43 [urllib3.connectionpool] DEBUG: http://localhost:9000 "DELETE /zips/fit4.zip HTTP/1.1" 204 0

----------------------------------------------------------------------
Ran 3 tests in 2.903s

OK
```

Figure 21: Test tasks.

9.8 Projecte Scrapy

Un projecte Scrapy té una estructura predefinida. De fet, la llibreria incorpora una eina per a generar un arbre de directoris corresponent a un projecte d'exemple.

Listing 3: Iniciar projecte en Scrapy

```
scrapy startproject tutorial
```

Partint doncs d'aquesta estructura i incorporant un mòdul que hem anomenat *run_crawlers* que és el responsable de importar i executar els fitxers de les *Aranyes* com veurem a la secció 9.8.1.

D'altra banda hem volgut mantenir la idea de separar el codi de credencials i altres dades sensibles, que hem dit a terme amb el *massive_importer*. Així doncs, el directori *conf* conté el fitxer *credentials.yaml* que no figura al repositori i conté comptes d'usuari i contrassenyes per als diferents portals web.

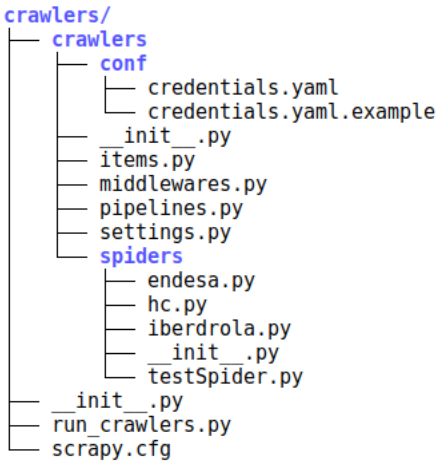


Figure 22: Arbre de directoris del projecte Scrapy.

9.8.1 Integració d'un projecte Scrapy

La integració d'un projecte Scrapy dins un altre projecte Python és una feina no estipulada, i podem trobar vàries formes de fer-ho. No obstant, hem de decidir l'arbre de directoris que volem tenir i respectar l'estructura d'un projecte Scrapy.

He inclòs el directori *Crawlers* que conté el projecte Scrapy mencionat a la seccio anterior al directori *massive_importer/massive_importer/*. (Arbre de directoris annexat)

L'element que ens serveis de nexe entre el projecte base i el de Crawlers, és el mòdul *run_spiders* que passem a comentar tot seguit:

9.8.2 run_crawlers

Aquest mòdul és l'encarregat de engegar el *motor* dels crawlers, cridant la llibreria *CrawlerRunner* i afegint-ne les aranyes seleccionades.

Té força importància dins el procés, donat que ens permet tenir d'una forma molt modular i configurable anar afegint aranyes a la tasca de *web crawling*. Inicialment es volia automatitzar les descarregues fent ús exclusiu de Scrapy. No obstant, com veurem tot seguit, Selenium ens aporta un millor resultat en termes de manteniment.

Dit això, la tasca de *run_crawlers.py* és llegir els valors de la configuració, on ens trobem quelcom similar al següent:

Listing 4: config.yaml

```
# Crawlers conf expecting 'Scrapy' or 'Selenium'
crawlers:
  testSpider: 'Scrapy'
  endesa: 'Selenium'
  endesa_r: ''
  iberdrola: ''
  hc: ''
  fenosa: ''
  viesgo: ''
  aseme: ''
  cide: ''
  conquense: ''
  begasa: ''
```

El sistema de control de quines aranyes s'han d'executar i quines no, compta amb dos punts de control: Per una banda, el fitxer de configuració *config.yaml* del *massive_importer* conté el llistat de crawlers disponibles, i es troben assignats a 'Scrapy' o 'Selenium' aquells que es vulguin executar.

El segon mecanisme de control és al propi mètode *crawl* del mòdul *run_spiders* el qual verifica que existeixi un arxiu corresponent al llistat de crawlers amb valor 'Scrapy' o 'Selenium'. En cas d'existir, importa el mòdul al projecte.
Acte seguit l'afegeix a la cua de crawlers (Scrapy porta l'eina de gestió de la cua, que és el *scrapy.crawler.CrawlerRunner*). I comença el procés de crawling.

Pel cas concret de Selenium, no es disposa de gestió de cua de processos de crawl, aixi doncs s'ha decidit gestionar-ho amb la llibreria *concurrent.futures.ThreadPoolExecutor*.

En tots dos casos, s'ha pensat una gestió d'excepcions que determinaran si un procés ha acabat malament, per a poder notificar l'acabament d'una tasca de forma controlada.

Aquest mòdul s'instancia des de la funció *web_crawling*, abans descrita a la secció de tasques 9.7. Finalment el mòdul també compta amb la funció *check_downloaded_files* que serà utilitzada per la tasca *summary* per determinar si els fitxers efectivament han estat descarregats.

9.9 Creació d'una aranya amb Scrapy

En primer lloc per tal d'unificar els noms, i poder executar qualsevol aranya, el nostre "gestor" de processos de crawling, el mòdul *run_crawlers* espera que comptem amb un mètode estàtic anomenat **run** que retorni una instància de la classe que dona nom al fitxer. De la forma següent:

Listing 5: run

```
def run():
    return Iberdrola()
class Iberdrola(scrapy.Spider):
    name = "iberdrola"
```

A part d'aquest detall de disseny, la construcció de la nostra aranya serà de la mateixa forma que ho fariem amb qualsevol projecte Scrapy. Com es veu al codi més amunt, necessitem declarar un nom per a l'aranya, així com un mètode inicial anomenat *start_requests*.

Listing 6: start_*equests*

```
def start_requests(self):
    urls = [
        'https://www.iberdrola.es/...
    ]
    for url in urls:
        yield scrapy.Request(url=url, callback=self.parse)
```

Tot és força intuïtiu, i no ens trobem amb cap problema, mentre no treballem amb JavaScript. Aquest darrer element, com s'ha comentat anteriorment, genera una serie de reptes, que es poden superar instal·lant i configurant correctament certs "plugins" per a Scrapy com és el cas de Splash.

Si seguim amb el desenvolupament del crawler, saltem del mètode inicial cap al definit com a 'callback'. Anar passant d'una funció a la següent no es fa en el moment de generar petincions web: aqui resideix el segon repte amb Scrapy, en aquest cas derivat del manteniment que pot suposar. Tot seguit mostrem com es construeix un formulari amb totes les dades que espera rebre el protocol HTTP:

```
    def seguiment(self, response):
        now = datetime.datetime.now()
        today = now.strftime("%d/%m/%Y")
        base_form = {
            'accion': response.xpath("//input
            'fec_recep_desde': "13/05/2019",#e").get(),
            'hor_recep_desde': "00", # respon
            'min_recep_desde': "00", # respon
            'seg_recep_desde': "00", # respon
            'fec_recep_hasta': "13/05/2019",#e").get(),
            'hor_recep_hasta': "23", # respon
            'min_recep_hasta': "59", # respon
            'seg_recep_hasta': "59", # respon
            'fec_descarga_desde': response.xp
            'hor_descarga_desde': response.xp
            'min_descarga_desde': response.xp
            'seg_descarga_desde': response.xp
            'fec_descarga_hasta': response.xp
            'hor_descarga_hasta': response.xp
            'min_descarga_hasta': response.xp
            'seg_descarga_hasta': response.xp
            'tipo': response.xpath("//select[
            'cod_emisora': "",
            'cod_destino': response.xpath("//A')]/@value").get(),
            'des_emisora': response.xpath("//
            'cod_grupo_emp_emisora': response
            'cod_grupo_emp_destino': response
            'grupo_proceso': response.xpath("),
            'cod_proceso': response.xpath("//
            'cod_paso': response.xpath("//sel
            'cod_baja': response.xpath("//sel
            'cod_solicitud': response.xpath("
            'sec_solicitud': response.xpath("
            'cups': response.xpath("//input[@
            'cod_descarga': response.xpath("/
            'pagina': response.xpath("//input
            'ventana': response.xpath("//inpu
        }
```

Com podem comprovar, si accedim al fitxer del crawler d'Iberdrola desenvolupat per Scrapy, veiem que fon forces passes les que s'han de seguir, i per tant forces *forms* que s'han de omplir.

Per acabar amb el crawler, el darrer pas el contindrem dins un 'try catch' per gestionar correctament l'error de descarrega, on s'acaba posant l'arxiu descarregat al bucket de minio:

Listing 7: step4

```
def step4(self, response):
    try:
        filename = ...
        filename = filename.split(';')[1].split('=')[1]
        filename = filename.strip('"\'')
        todayfolder = datetime.datetime.now().strftime("%d-%m-%Y")
        full_filename = '{}_{}'.format(Iberdrola.name, filename)

        filename = "%s/%s" % (todayfolder,full_filename)
    except Exception as err:
        logger.error(err)
    finally:
        yield minio_manager.\
            put_file(minio_manager.default_bucket, filename, response.body)
```

Així doncs després d'haver experimentat el temps que ens pot portat el desenvolupament d'un crawler amb Scrapy especialment si aquest ha d'incorporar inteacció obligatoria amb JavaScript, s'obre la porta a fer servir aranyes amb Selenium.

9.10 Creació d'una aranya amb Selenium

En aquest cas ens trobem amb un fitxer molt més compacte: Selenium compta amb una amplia llibreria pensada per interactuar amb qualsevol element de la web moderna. A més ens permet visualitzar (també és pot desactivar) el navegador web mentre es mou per la web.

En mòde de resum, exemplifiquem com es navega per la web amb Selenium, i el moment de la descarrega:

Listing 8: selenium$_{s}pider$

```
def start(self):
    try:
        self.driver.get("https://portal.edistribucion.com/login")
        WebDriverWait(self.driver,30).\
            until(EC.visibility_of_element_located((By.ID, "Login")))
        userbox = self.driver.find_element_by_id("username")
        userbox.send_keys(credentials['username'])
        passbox = self.driver.find_element_by_id("password")
        passbox.send_keys(credentials['password'])

        loginbt = self.driver.find_element_by_id("Login")
        loginbt.click()

        self.driver.get("https://portalede.endesa.es/apex/...")

        [...]

        self.wait_until_ready()
        self.driver.find_element_by_name("download_messages").click()

        filename = self.file_wait_download()

    except Exception as e:
        raise CrawlingProcessException(e)
    finally:
        self.driver.quit()
    try:
        newfilename = '{}_{}'.format(Endesa.name, filename)
        self.file_to_minio(filename, newfilename)
    except FileToBucketException as e:
        raise e
```

9.11 Proves finals

Fins ara com a proves només hem vist els testos integrats que avaluen el comportament dels mòduls de forma parcial. En aquest apartat ens centrarem en avaluar o provar el funcionament íntegre del projecte, en un entorn de testing.

Voldrem doncs simular el procés **d'importació de casos ATR** que diàriament es realitza de forma manual. Als Annexos trobem un exemple de com es realitza aquesta tasca en 11 passes.

Programació de les tasques

Dins d'aquest entorn de test, i per veure el correcte funcionament, configurarem les tasques iniciar-se pocs segons després de l'execució del *massive_importer*. L'*scheduling* per a aquesta prova és el següent (fitxer de configuració *test.py*):

```
TASKS = {
    'check_new_events': {
        'trigger': 'interval',
        'minutes': 100,
        'next_run_time': datetime.now() + timedelta(seconds=30)
    },

    'web_crawling': {
        'trigger': 'interval',
        'minutes': 100,
        'next_run_time': datetime.now() + timedelta(seconds=5)
    },

    'summary': {
        'trigger': 'interval',
        'minutes': 100,
        'next_run_time': datetime.now() + timedelta(seconds=100)
    }
}
```

Figure 23: Programació de tasques per l'entorn de test

Selecció de crawlers

També seleccionarem el crawler de Iberdrola per incloure'l a aquest procés. Per fer-ho, assignem a 'True' l'entrada de Iberdrola al fitxer *config.yaml*:

```
# Crawlers conf
crawlers:
  endesa: ''
  iberdrola: 'True'
  hc: ''
  fenosa: ''
  viesgo: ''
  aseme: ''
  cide: ''
  conquense: ''
  begasa: ''
```

Figure 24: Selecció de crawlers.

Execució del *massive_importer*

Listing 9: Execució del programa

```
pipenv run python main.py
```

Output

En aquesta primera part del output, veiem com es carreguen les 3 tasques, i es queda en repòs fins al cap de 1.14 segons (son els restants fins als 5 segons que hem posat de timedelta per a la primera tasca).

```
somenergia@PC128:~/Documents/somenergia/massive_importer$ pipenv run python main.py
[2019-06-12 02:55:26,792] [9738] [DEBUG][startup_configuration.build_app:29] Build app finished
[2019-06-12 02:55:26,792] [9738] [DEBUG][startup_configuration.add_jobs:34] Adding task: check_new_events
[2019-06-12 02:55:26,824] [9738] [INFO][base.add_job:433] Adding job tentatively -- it will be properly scheduled when the sch
eduler starts
[2019-06-12 02:55:26,824] [9738] [DEBUG][startup_configuration.add_jobs:36] Adding task: web_crawling
[2019-06-12 02:55:26,824] [9738] [INFO][base.add_job:433] Adding job tentatively -- it will be properly scheduled when the sch
eduler starts
[2019-06-12 02:55:26,825] [9738] [DEBUG][startup_configuration.add_jobs:38] Adding task: summary
[2019-06-12 02:55:26,825] [9738] [INFO][base.add_job:433] Adding job tentatively -- it will be properly scheduled when the sch
eduler starts
[2019-06-12 02:55:26,825] [9738] [INFO][main.main:14] Running massive importer...
[2019-06-12 02:55:26,825] [9738] [INFO][base._real_add_job:867] Added job "check_new_events" to job store "default"
[2019-06-12 02:55:26,825] [9738] [INFO][base._real_add_job:867] Added job "web_crawling" to job store "default"
[2019-06-12 02:55:26,825] [9738] [INFO][base._real_add_job:867] Added job "summary" to job store "default"
[2019-06-12 02:55:26,825] [9738] [INFO][base.start:159] Scheduler started
[2019-06-12 02:55:26,825] [9738] [DEBUG][base._process_jobs:926] Looking for jobs to run
[2019-06-12 02:55:26,825] [9738] [DEBUG][base._process_jobs:1006] Next wakeup is due at 2019-06-12 02:55:27.965636+02:00 (in 1
.140010 seconds)
```

Figure 25: Output 1/3.

En aquesta segon part de l'output veiem com s'executa la primera tasca, podem llegir com es carrega satisfactoriament el mòdul de Iberdrola i comença le procés de crawling. Segons després, el procés ha finalitzat i *massive_importer* torna a quedar-se en espera fins la següent tasca.

```
[2019-06-12 02:55:27,965] [9738] [DEBUG][base._process_jobs:926] Looking for jobs to run
[2019-06-12 02:55:27,967] [9738] [INFO][base.run_job:123] Running job "web_crawling (trigger: interval[1:40:00], next run at:
2019-06-12 02:55:27 CEST)" (scheduled at 2019-06-12 02:55:27.965636+02:00)
[2019-06-12 02:55:27,967] [9738] [DEBUG][tasks.web_crawling:25] Crawl process starting...
[2019-06-12 02:55:27,971] [9738] [DEBUG][base._process_jobs:1006] Next wakeup is due at 2019-06-12 02:55:52.965629+02:00 (in 2
4.999408 seconds)
[2019-06-12 02:55:27,977] [9738] [DEBUG][run_crawlers.crawl:22] Loaded iberdrola module
[2019-06-12 02:55:27,977] [9738] [DEBUG][run_crawlers.crawl:26] Starting iberdrola crawling...
[2019-06-12 02:55:52,300] [9738] [DEBUG][tasks.web_crawling:27] Process done!
[2019-06-12 02:55:52,300] [9738] [INFO][base.run_job:144] Job "web_crawling (trigger: interval[1:40:00], next run at: 2019-06-12 04:35:27 CEST)" ex
ecuted successfully
[2019-06-12 02:55:52,971] [9738] [DEBUG][base._process_jobs:926] Looking for jobs to run
[2019-06-12 02:55:52,972] [9738] [INFO][base.run_job:123] Running job "check_new_events (trigger: interval[1:40:00], next run at: 2019-06-12 02:55:
52 CEST)" (scheduled at 2019-06-12 02:55:52.965629+02:00)
```

Figure 26: Output 2/3.

En aquesta darrera part del output, veiem l'execució de la tasca *chec_new_events* i podem observar el valor de retorn del ERP, **GRCW_CO-0762_20190612003231 generated with result: True**.

També veiem l'execució satisfactoria de la tasca *summary*.

```
[2019-06-12 02:55:52,973] [9738] [DEBUG][base._process_jobs:1006] Next wakeup is due at 2019-06-12 02:57:02.965637+02:00 (in 69.994012 seconds)
[2019-06-12 02:55:52,973] [9738] [DEBUG][tasks.check_new_events:33] Import zips process stating...
[2019-06-12 02:55:52,986] [9738] [DEBUG][tasks.check_new_events:43] Afegit l'ImportFile 12-06-2019/GRCW_CO-0762_20190612025552.zip a la llista!
[2019-06-12 02:55:52,992] [9738] [DEBUG][minio_utils.get_file_content:38] Longitud: 23029
[2019-06-12 02:56:04,944] [9738] [DEBUG][tasks.check_new_events:58] 12-06-2019%2FGRCW_CO-0762_20190612025552.zip generated with result: True
[2019-06-12 02:56:04,945] [9738] [DEBUG][tasks.check_new_events:60] Process done!
[2019-06-12 02:56:04,952] [9738] [INFO][base.run_job:144] Job "check_new_events (trigger: interval[1:40:00], next run at: 2019-06-12 04:35:52 CEST)
" executed successfully
[2019-06-12 02:57:02,967] [9738] [DEBUG][base._process_jobs:926] Looking for jobs to run
[2019-06-12 02:57:02,968] [9738] [INFO][base.run_job:123] Running job "summary (trigger: interval[1:40:00], next run at: 2019-06-12 02:57:02 CEST)"
 (scheduled at 2019-06-12 02:57:02.965637+02:00)
[2019-06-12 02:57:02,975] [9738] [DEBUG][base._process_jobs:1006] Next wakeup is due at 2019-06-12 04:35:27.965636+02:00 (in 5904.997678 seconds)
[2019-06-12 02:57:04,613] [9738] [INFO][base.run_job:144] Job "summary (trigger: interval[1:40:00], next run at: 2019-06-12 04:37:02 CEST)" execute
d successfully
^C[2019-06-12 02:57:10,280] [9738] [INFO][main.main:17] Stopping massive importer...
[2019-06-12 02:57:10,282] [9738] [INFO][base.shutdown:191] Scheduler has been shut down
```

Figure 27: Output 3/3.

Comprovació manual

Per veure com s'han efectuats els canvis realment, passem a veure tant al bucket de MinIO, com a la taula de ImportFiles, com al correu d'Alerta enviat:

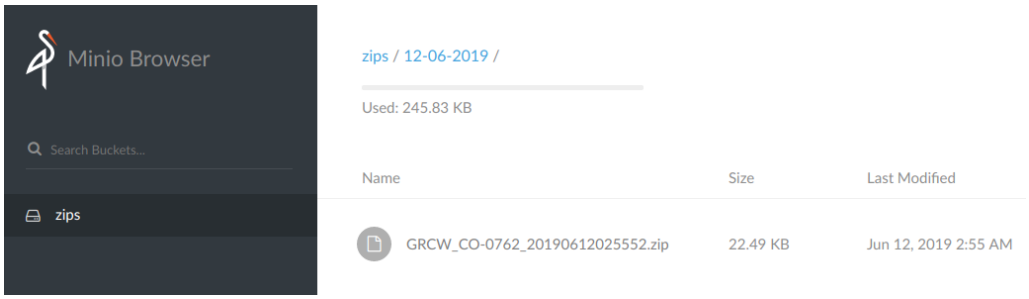


Figure 28: Bucket zips de MinIO.

Data Output | Explain | Messages | Notifications

	id [PK] integer	etag text	bucket text	name text	state text	size double pre	retries intege	msg text	created_at timestamp wi	modified_at timestamp with
1	13	0ba295...	zips	12-06-2019%2FGRCW_CO-0762_20190612025552.zip	ok	23029	0		2019-06-12...	2019-06-12 0...

Figure 29: Taula ImportFiles PostgreSQL.

Figure 30: Correu resum casos importats.

10 Implantació i resultats

Per a la implantació i la instal·lació dels diferents serveis, ho documento des d'un entorn local (en la pròpia màquina). No obstant, a l'hora de fer-ho sobre els servidors de producció, es farà mitjançant la comunicació amb l'administrador encarregat de gestionar-ho.

10.1 Instalació de la BD

Per a la instalació de *PostgreSQL* en un entorn debian, amb apt simplement hem d'executar des d'una terminal:

Listing 10: Instalació de PostgreSQL

```
apt-get install postgresql-10
```

Tot seguit passem a instalar un gestor visual de PostgreSQL: pgadmin4. Aquesta eina ens facilitarà l'accés i visualització de les taules:

Listing 11: Instalació de pgadmin4

```
apt-get install pgadmin4
```

Per tal d'accedir a Postgres, haurem de crear un usuari (com que es tracta d'un entorn local, no cal ser estrictes amb credencials). També passem a crear la base de dades que utilitzarem.

Listing 12: Configuració de la BD

```
sudo -u postgres psql postgres
sudo -u postgres createdb bucketevents_bd
```

Un cop fet això, si executem *pgadmin4* sen's obrirà des del navegador la finestra de gestió com mostrem a la Figura 13.

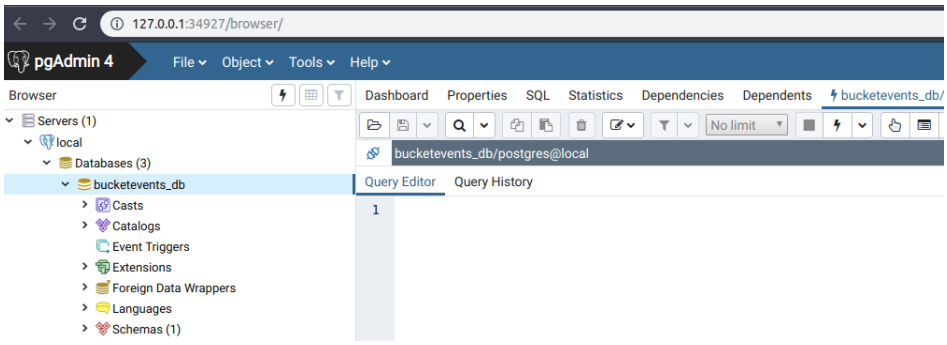


Figure 31: Interfície web pgadmin4.

10.2 Instalació del MinIO

La posada en marxa d'un servidor MinIO és tan senzill com es mostra a continuació. Si bé ara només accedirem localment, més endevant per a la implantació haurem de fer efectiva una configuració de permisos publics.

Listing 13: Instalació de MinIO

```
wget https://dl.minio.io/server/minio/release/linux-amd64/minio
chmod +x minio
./minio server /data
```

Un cop corrent el servidor, sens mostra la següent informació:

Figure 32: Servidor MinIO running.

MinIO amés, incorpora per defecte una interfície web que ens fa de client. De forma que si accedim al localhost, port 9000, veurem una còmode pagina web per a la gestió dels buckets i objectes:

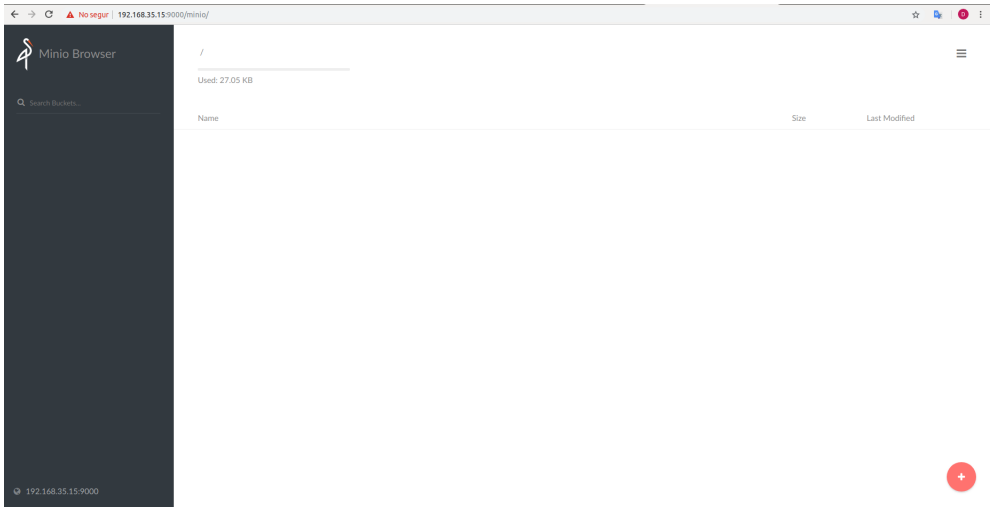


Figure 33: Interfície web de MinIO.

10.3 Configuració de MinIO

En primer lloc haurem de crear un bucket anomenat *zips*. MinIO compta amb un client, que hem de descarregar de forma independent al servidor, anomenat **mc**. La primera linia a continuació descarrega el client de minio, mentre que la segon afegeix el que ells anomenen "host". Un host no és més que una configuració per a diversos buckets. En aquest cas al nostre host li diem "minio".

Listing 14: Instalació del minio client

```
wget https://dl.minio.io/client/mc/release/linux-amd64/mcchmod +x minio
./mc config host add minio http://127.0.0.1:9000 NJHXYRAAILTKUVSXWDP9
IqoH+bHYxkfE8xvmLpMdaxjLGJkmhria3Qxi9Q73
```

Per tal de crear el bucket *zips* dins el host *minio* fem el següent:

Listing 15: Creació d'un Bucket

```
./mc mb minio/zips
```

Si volem establir una configuració, partint del fitxer en json amb la configuració actual del servidor, fem:

Listing 16: Configuració de minio

```
./mc admin config get minio/ > /tmp/myconfig
# editem el que volguem i posteriorment:
./mc admin config set myminio < /tmp/myconfig
```

En aquest cas, per establir la configuració de events mitjançant PostgreSQL, hem de introduïr o emplenar el camp corresponent, de la següent forma:

Listing 17: Configuració del postgres

```
"postgresql": {
                "1": {
                        "enable": true,
                        "format": "namespace",
                        "connectionString": "sslmode=disable",
                        "table": "events",
                        "host": "127.0.0.1",
                        "port": "5432",
                        "user": "postgres",
                        "password": "postgres",
                        "database": "bucketevents_db"
                }
```

Per tal de notificar els events dels arxius dipositats al bucket, MinIO utilitza (directament heretat de Amazon S3) unes cadenes de caràcters de configuració anomenats *arn*. En aquestes linies s'espeficica el tipus de servei que recollira l'event, així com el format de fitxers que l'ha d'accionar, etc. A continuació afegim una linia de arn per a fitxers .zip per postgres:

Listing 18: Events per format d'arxius

```
mc event add minio/zips arn:minio:sqs::1:postgresql --suffix .zip
```

En cas de voler determinar un host com a públic, el propi client de minio ens ho permet a través de la següent comanda:

Listing 19: Aplicar permisos publics

```
mc policy minio/zips public
```

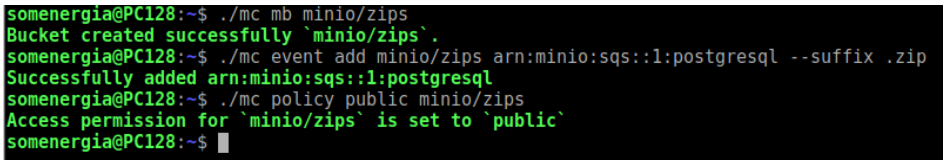


Figure 34: Minio havent aplicat permisos publics.

Per a fer la implantació als servidors on s'executarà el *massive_importer* he hagut de parlar amb l'administrador de sistemes de *Som Energia*, donat que es faran servir els Servidor ja existents tant pel PostgreSQL, com pel MinIO, com per la instància de *massive_importer*.

Així doncs havent sol·licitat un usuari concret, i la creació d'una base de dades amb les taules 'events' i 'import_file' ja comptem amb els paràmetres de configuració necessaris per a la BD.

De forma temporal serà el mateix servidor de PostgreSQL el que executi el *massive_importer*. S'ha tractat el tema en diverses reunions i ha sorgit la possibilitat de albergar en servidor apart, "scripts" o altres solucions, com el *massive_importer*.

Pel que fa al servidor de MinIO, actualment es comptava amb un servidor actiu, el qual només he hagut d'efectuar la configuració vista en la secció d'implementació (crear el bucket, assignar-li polítiques publoques, i configurar els events amb el postgres).

Com que actualment només disposem de dues *aranya* completament funcionals, ara per ara, el personal tècnic continuarà fent manualment les descàrregues de la resta de portals. No obstant, la pujada d'arxius al ERP, com ja hem parlat a la secció d'Anàlisi, la podem automatizar muntant al sistema de fitxers el bucket de minio a partir de FUSE.

Per tal de muntar la unitat, l'unic que hem de fer (despŕes d'haver fet la configuració de MinIO citada prèviament) és afegir la següent linia al fitxer **/etc/fstab**:

Listing 20: Configuració del postgres

```
s3fs#zips /home/somenergia/Escriptori/drop fuse
_netdev,allow_other,use_path_request_style,no_check_certificate
,url=https://direccioServidorMinio/ 0 0
```

Un cop reiniciat l'equip, tenim el bucket zip muntat a l'escriptori tal i com podem veure a la figura 35.

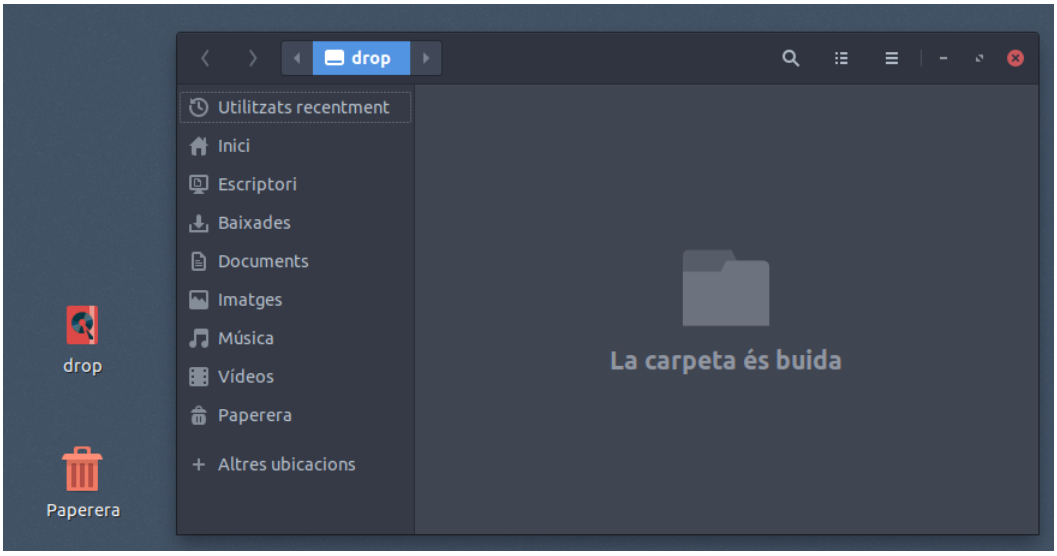


Figure 35: Bucket de MinIO muntat al sistema de fitxers amb FUSE.

10.4 Resultats en local

Amb tot l'entorn configurat, el servidor de MinIO funcionant, una instància del ERP esperant peticions i el PostgreSQL escoltant al port 5432, podem configurar l'execució del *massive_importer*.

Partint d'un fitxer de configuració on seleccionem l'aranya 'endesa' amb el valor de 'Selenium'. La configuració de les tasques és la següent:

Listing 21: tasques

```
TASKS = {
    'check_new_events': {
        'trigger': 'interval',
        'minutes': 99999,
        'next_run_time': datetime.now() + timedelta(seconds=120)
    },
    'web_crawling': {
        'trigger': 'interval',
        'minutes': 99999,
        'next_run_time': datetime.now() + timedelta(seconds=5)
    },
    'summary': {
        'trigger': 'interval',
        'minutes': 99999,
        'next_run_time': datetime.now() + timedelta(seconds=140)
    }
}
```

Així doncs només tindrem una execució, comprovarem que el comportament és el que esperem, i matarem el procés.

Executem amb la comanda:

Listing 22: Execució del programa

```
pipenv run python main.py
```

Podem observar com el log es queda aturat en el procés de crawling, i donat que no tenim desactivada la part gràfica del navegador de Selenium (opcional) podem veure el seguit de movimients que realitza a la web fins a generar la descàrrega: Podem comprovar com s'ha creat dins el bucket de

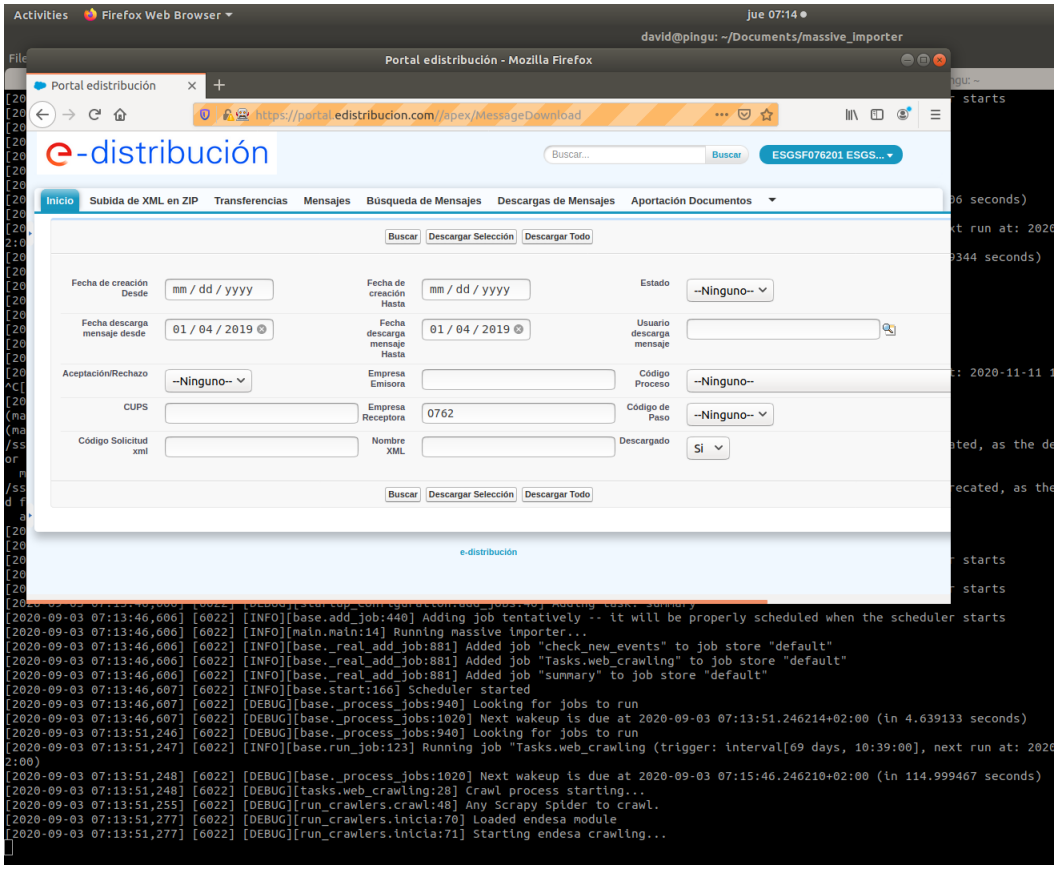


Figure 36: Log i procés de crawling

minio, una ruta que incorpora la data de la descarrega i el nom del fitxer descarregat.

Per altra banda, s'ha creat la respectiva entrada al fitxer de import_files que marca en estat 'OK' donat que la importació ja ha quedat en mans del ERP, de forma efectiva.

Per ultim comprovem que el sistema de resum *summary* funciona correctament:

Figure 37: Email enviat amb la tasca summary

10.5 Resultats en real

Partint de la implantació actual, dins un entorn de desenvolupament encara, hem estat realitzant proves de funcionament amb els arxius importats diariament, en paral·lel als resultats de producció comprovant la correctesa del procés.

El fet de descarregar els fitxers directes a l'unitat virtual en lloc de utilitzar el gestor de finestres del ERP, si bé no és el procés que més temps estalvia, estem parlant d'una reducció considerable de temps d'espera (a la resposta del ERP) i una gestió automàtica de reintents, en cas de fallada. Així doncs, com a sistema, afegeix el valor de l'automatització del procés de importació al erp, amb el re-intent automàtic per a aquelles descarregues que s'han de fer encara manualment:

D'altra banda, pels portals que comptin amb aranyes desenvolupades, la automatització és total, on només ha d'intervenir l'usuari en cas d'haver trobat algun error.

Tenim doncs un sistema igual que el que hem provat en l'entorn de Test, amb una descentralització dels diferents serveis, no obstant, el comportament és el mateix.

Donat que en aquest entorn les tasques les tenim programades tal com apareixen a la figura 39 i no disposo del temps per fer les proves sota aquest schedulling, he executat en l'entorn de develop les tasques amb la configuració descrita a la Figura 25 (mateixa configuració que a l'entorn de test).

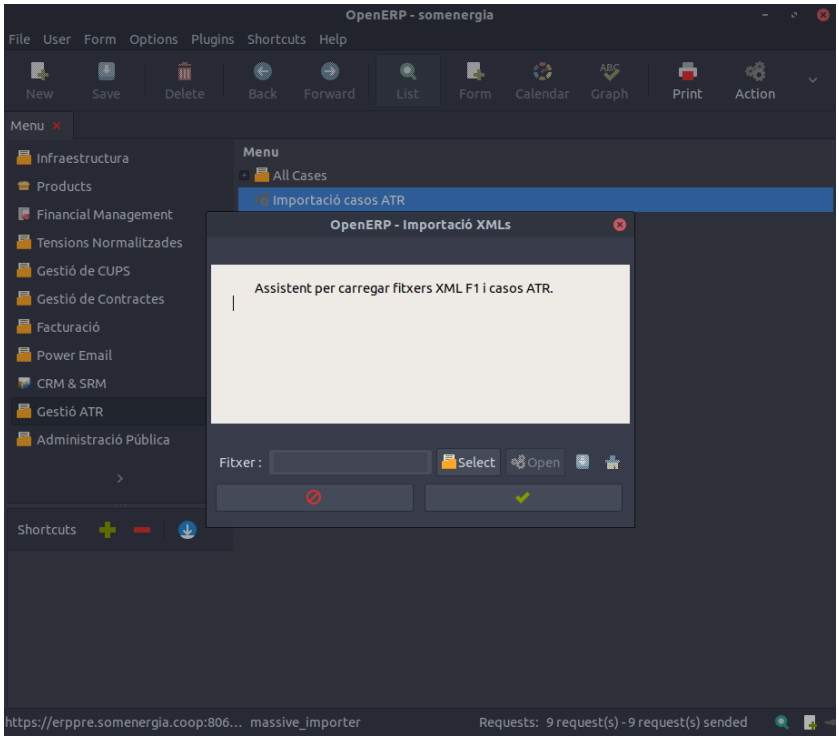


Figure 38: Assistent d'importació.

```
TASKS = {
    'web_crawling': {
        'trigger': 'cron',
        'hour': 21,
        'minutes': 0,
    },

    'check_new_events': {
        'trigger': 'cron',
        'hour': 22,
        'minute': 0,
    },

    'summary': {
        'trigger': 'cron',
        'hour': 23,
        'minutes': 30,
    }
}
```

Figure 39: Configuració de les tasques a *devel.py*

Output

```
[2019-06-12 04:06:59,933] [17185] [INFO][main.main:14] Running massive importer...
[2019-06-12 04:06:59,933] [17185] [INFO][base._real_add_job:867] Added job "check_new_events" to job store "default"
[2019-06-12 04:06:59,933] [17185] [INFO][base._real_add_job:867] Added job "web_crawling" to job store "default"
[2019-06-12 04:06:59,933] [17185] [INFO][base._real_add_job:867] Added job "summary" to job store "default"
[2019-06-12 04:06:59,933] [17185] [INFO][base.start:159] Scheduler started
[2019-06-12 04:06:59,933] [17185] [DEBUG][base._process_jobs:926] Looking for jobs to run
[2019-06-12 04:06:59,934] [17185] [DEBUG][base._process_jobs:1006] Next wakeup is due at 2019-06-12 04:07:03.127433+02:00 (in 3.
193458 seconds)
[2019-06-12 04:07:03,127] [17185] [DEBUG][base._process_jobs:926] Looking for jobs to run
[2019-06-12 04:07:03,130] [17185] [INFO][base.run_job:123] Running job "web_crawling (trigger: interval[1:40:00], next run at: 2
019-06-12 04:07:03 CEST)" (scheduled at 2019-06-12 04:07:03.127433+02:00)
[2019-06-12 04:07:03,130] [17185] [DEBUG][base._process_jobs:1006] Next wakeup is due at 2019-06-12 04:07:28.127424+02:00 (in 24
.999437 seconds)
[2019-06-12 04:07:03,130] [17185] [DEBUG][tasks.web_crawling:25] Crawl process starting...
[2019-06-12 04:07:03,141] [17185] [DEBUG][run_crawlers.crawl:22] Loaded iberdrola module
[2019-06-12 04:07:03,141] [17185] [DEBUG][run_crawlers.crawl:26] Starting iberdrola crawling...
[2019-06-12 04:07:05,430] [17185] [DEBUG][tasks.web_crawling:27] Process done!
[2019-06-12 04:07:05,431] [17185] [INFO][base.run_job:144] Job "web_crawling (trigger: interval[1:40:00], next run at: 2019-06-12 05:47:03 CES
T)" executed successfully
[2019-06-12 04:07:28,130] [17185] [DEBUG][base._process_jobs:926] Looking for jobs to run
[2019-06-12 04:07:28,132] [17185] [INFO][base.run_job:123] Running job "check_new_events (trigger: interval[1:40:00], next run at: 2019-06-12
04:07:28 CEST)" (scheduled at 2019-06-12 04:07:28.127424+02:00)
[2019-06-12 04:07:28,133] [17185] [DEBUG][base._process_jobs:1006] Next wakeup is due at 2019-06-12 04:08:38.127434+02:00 (in 69.995948 second
s)
[2019-06-12 04:07:28,133] [17185] [DEBUG][tasks.check_new_events:33] Import zips process stating...
[2019-06-12 04:07:28,148] [17185] [DEBUG][tasks.check_new_events:43] Afegit l'ImportFile 12-06-2019/GRCW_CO-0762_20190612040705.zip a la llist
a!
[2019-06-12 04:07:28,154] [17185] [DEBUG][minio_utils.get_file_content:38] Longitud: 23029
[2019-06-12 04:07:39,030] [17185] [DEBUG][tasks.check_new_events:58] 12-06-2019%2FGRCW_CO-0762_20190612040705.zip generated with result: True
[2019-06-12 04:07:39,031] [17185] [DEBUG][tasks.check_new_events:60] Process done!
[2019-06-12 04:07:39,035] [17185] [INFO][base.run_job:144] Job "check_new_events (trigger: interval[1:40:00], next run at: 2019-06-12 05:47:28
 CEST)" executed successfully
[2019-06-12 04:08:38,129] [17185] [DEBUG][base._process_jobs:926] Looking for jobs to run
[2019-06-12 04:08:38,130] [17185] [INFO][base.run_job:123] Running job "summary (trigger: interval[1:40:00], next run at: 2019-06-12 04:08:38
CEST)" (scheduled at 2019-06-12 04:08:38.127434+02:00)
[2019-06-12 04:08:38,131] [17185] [DEBUG][base._process_jobs:1006] Next wakeup is due at 2019-06-12 05:47:03.127433+02:00 (in 5904.997403 seco
nds)
[2019-06-12 04:08:39,927] [17185] [INFO][base.run_job:144] Job "summary (trigger: interval[1:40:00], next run at: 2019-06-12 05:48:38 CEST)" e
xecuted successfully
```

11 Conclusions i treball Futur

L'automatització és un repte al que s'enfronten totes les empreses. Si bé a data d'avui moltíssima gent se'n cuida de tasques tedioses i repetitives, sabem que les màquines no es cansen, i podem aprofitar el talent humà per a tasques més creatives.

En el cas concret de *Som Energia* ens trobem que l'equip tècnic té moltes tasques que impliquen el component humà i que difícilment poden ser automatitzades, però hi ha d'altres com és el cas del que pretenem suplir amb la implantació del *massive_importer* que no són de gust de ningú.

És per aquesta raó que el treball és valorat positivament tant per mi com per la resta de l'equip que m'ha envoltat aquestes darreres setmanes, donant-me doncs empenta per assolir-lo.

La dificultat d'implantar de forma definitiva, és a dir, en producció (pendent en les properes setmanes) va de la mà de la sensibilitat del procés. És per aquesta raó que es volen efectuar moltes proves i testos abans de entrar en producció.

La facilitat de trobar eines compatibles amb plataformes Linux, i en Python ha estat sorprenent. Contínuament ens trobem amb múltiples eines, totes elles molt ben valorades i vàlides per qualsevol necessitat. Això m'ha despertat un gran interès en seguir desenvolupant i aprenent en aquest entorn, així com en la cooperació en el desenvolupament. Tant a nivell d'empresa com a nivell de grups de treball ja siguin lúdics, sense ànim de lucre, etc.

Personalment ha estat una experiència molt grata. Treballar a aquesta cooperativa ha estat una agradable sorpresa i tinc la sort de poder continuar, en acabar les pràctiques.

Treball Futur

- En les properes setmanes s'aniran afegint noves *Aranyes* al programa.
- Refactoritzacions pendents així com les tasques vistes al backlog.
- Realització de proves exhaustives del programa.
- La implantació en producció del *massive_importer*.
- En un futur es preveu fer una interfície web, que ens permeti configurar el procés de Crawling, d'una forma més atractiva i simple.

Setembre 2020

En acabar les pràctiques a *Som Energia* vaig passar a formar part de l'equip IT de la cooperativa, enfocant-me al desenvolupament del programa de gestió (ERP). Donat que no s'ha avançat en el projecte com estava previst (per falta de prioritat davant d'altres tasques amb imperatius legals o altres prioritats) lligat al prop d'un any d'experiència i aprenentatge del que he gaudit dins aquest context, el treball futur esdevé força diferent:

- Integració de massive_importer com un mòdul de openERP.
- Desenvolupament dels crawlers restants amb l'eina Selenium.
- Anàlisi i desenvolupament d'un mòdul d'importació amb linies de importació per fitxer.
- Desenvolupament de la visualització de l'estat de la importació (en lloc d'un resum).
- Incoorporar les variables de configuracio al sistema openERP.

12 Bibliografia

References

[1] **Odoo**
https://github.com/odoo/odoo

[2] **Gisce-TI**
https://www.gisce.net/

[3] **Comissió Nacional dels Mercats i la Competència (CNMC)**
https://www.cnmc.es/ca/sobre-la-cnmc/que-es-la-cnmc

[4] **massive_importer:** Projecte per a la gestió de l'automatització de tasques
https://github.com/Som-Energia/massive_importer

[5] **somenergia-portaldownload:** Projecte de descarrega automatitzada basada en Selenium
https://github.com/Som-Energia/somenergia-portaldownload

[6] **Lliberalització del mercat elèctric:**
Ley 54/1997, de 27 de noviembre, del Sector Eléctrico.
https://www.boe.es/eli/es/l/1997/11/27/54

[7] **Crawler, spider o spiderbot:** Programari que inspecciona pàgines de forma automatitzada
https://en.wikipedia.org/wiki/Web_crawler

[8] **APScheduler:** A Python library that lets you schedule your Python code to be executed later, either just once or periodically.
https://apscheduler.readthedocs.io/en/latest/index.html#

[9] **MinIO:** An object storage server
https://min.io/

[10] **Scrapy:** Is an application framework for crawling web sites and extracting structured data
https://docs.scrapy.org/en/latest/news.html

[11] **Scrapy-splash:** Middleware for JavaScript integration
https://github.com/scrapy-plugins/scrapy-splash

[12] **SQLAlchemy:** Python SQL toolkit and Object Relational Mapper
https://www.sqlalchemy.org/library.html

[13] **PonyORM:** An advanced object-relational mapper
https://docs.ponyorm.org/

[14] **Gitkraken:** A Git Client
https://www.gitkraken.com/

[15] **Postman:** A complete API development environment
https://www.getpostman.com/

Appendices

A Instal·lació Client ERP

Per tal de descarregar el client que fa servir *Som Energia*, senzillament hem de clonar el repositori public `https://github.com/gisce/erpclient.git` de la següent forma:

Listing 23: clonar client openERP

```
git clone https://github.com/gisce/erpclient
```

Listing 24: obrir client openERP

```
cd erpclient/bin
./openerp-client.py
```

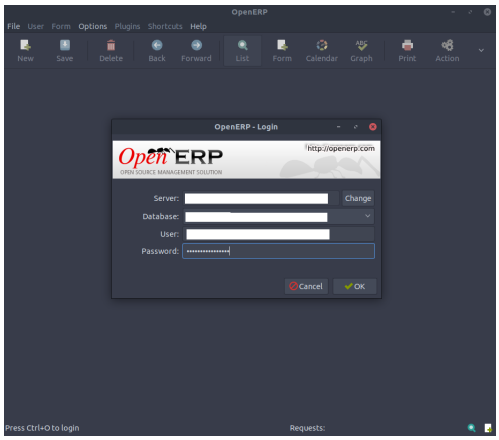


Figure 40: Client de openERP

B Arbre de directori del projecte

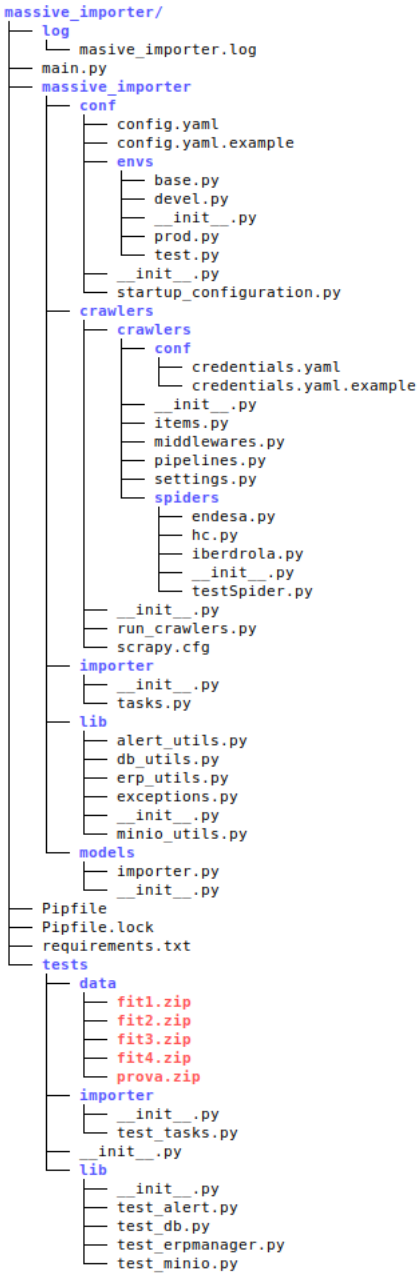


Figure 41: Arbre de directoris del projecte massive_importer.

C Rondes Trello

Figure 42: Ronda 0.

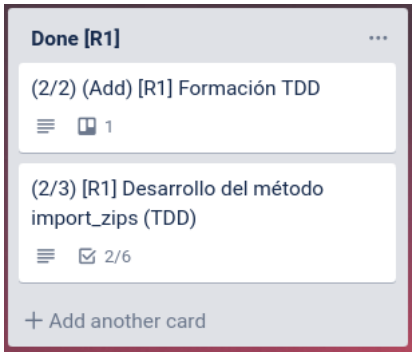


Figure 43: Ronda 1.

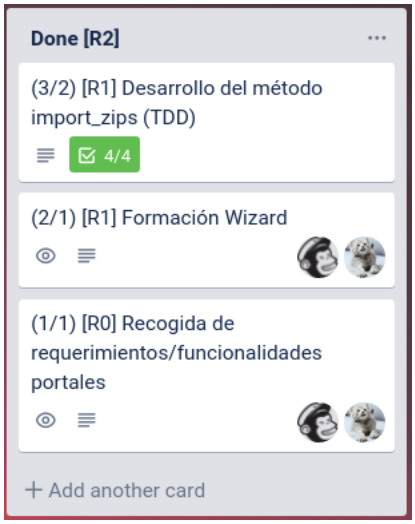


Figure 44: Ronda 2.

Figure 45: Ronda 3.

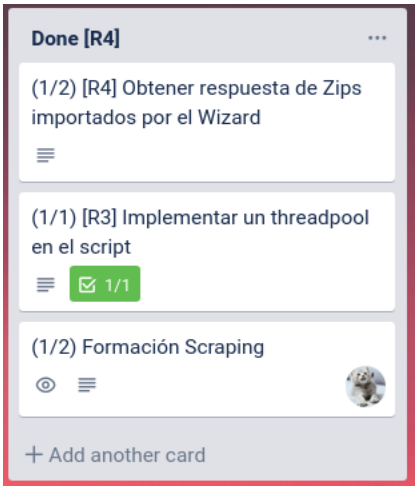


Figure 46: Ronda 4.

Figure 47: Ronda 5.

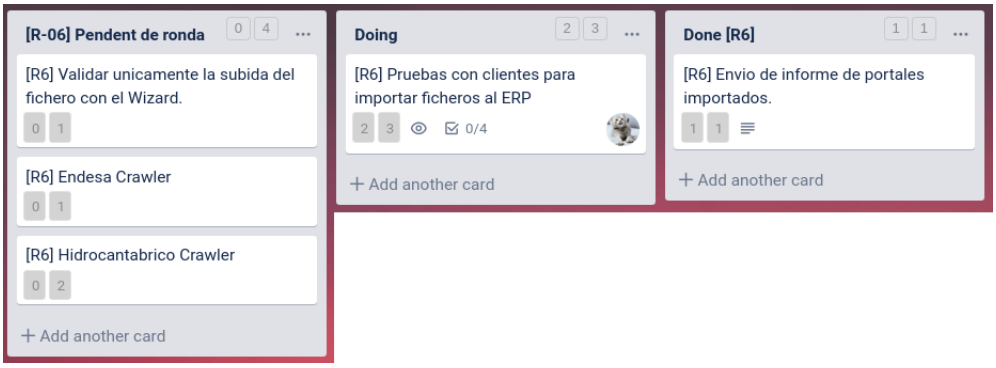


Figure 48: Ronda 6 (darrera ronda).

Figure 49: Rebost o backlog.

D Procés manual de descarrega per al portal d'Iberdrola

Figure 50: Pas1. Accés al portal web i autenticació.

Figure 51: Pas2. Accés al menú corresponent.

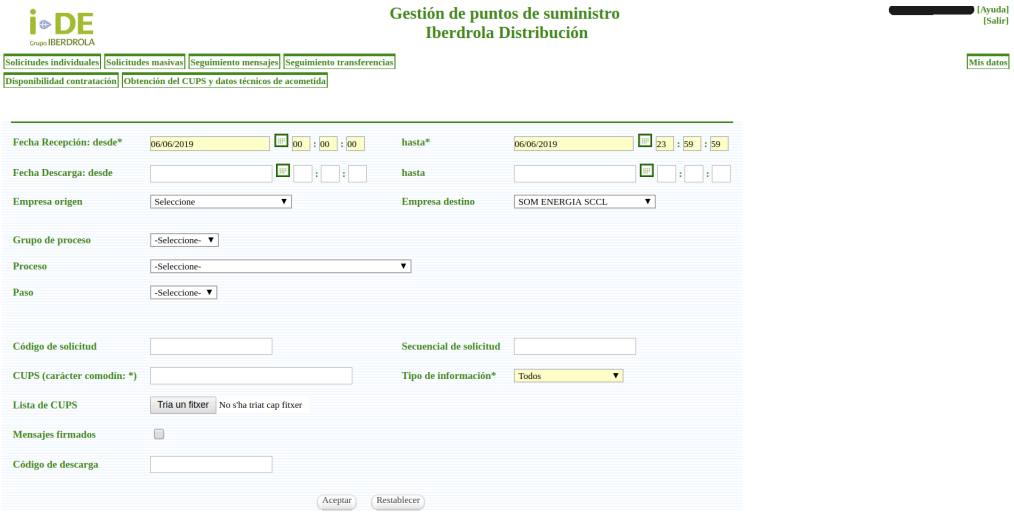


Figure 52: Pas3. Filtrat de cerca.

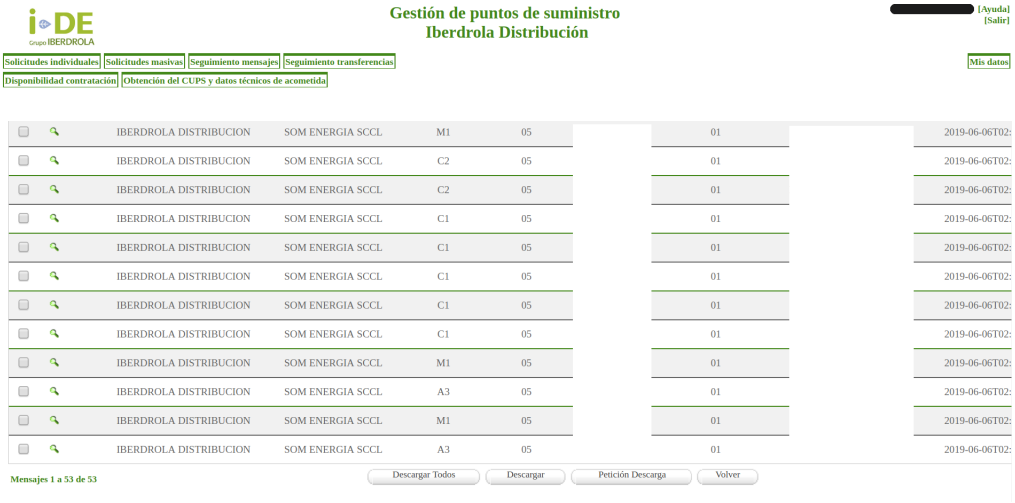


Figure 53: Pas4. Selecció d'elemnts a descarregar.

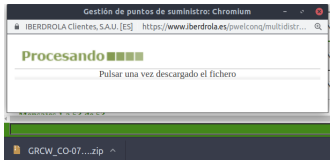


Figure 54: Pas5. Inici de la descarrega.

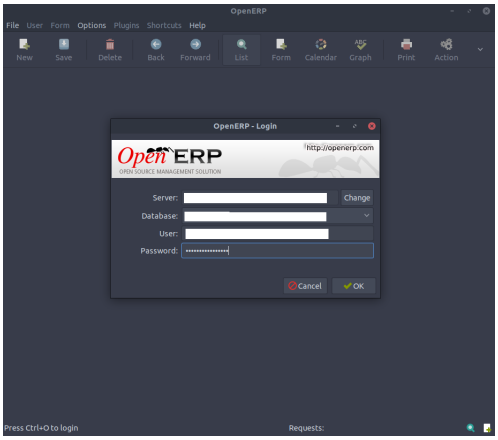


Figure 55: Pas6. Iniciar OpenERP amb credencials.

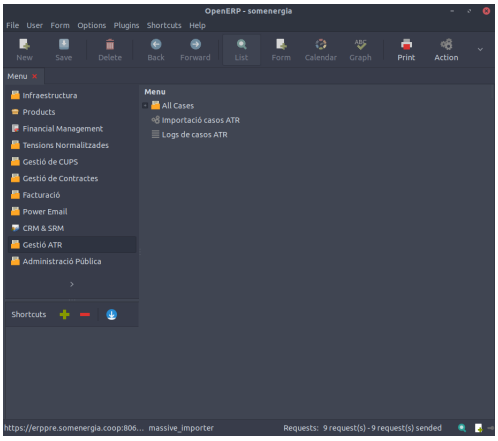


Figure 56: Pas7. Accedir al menú de casos ATR.

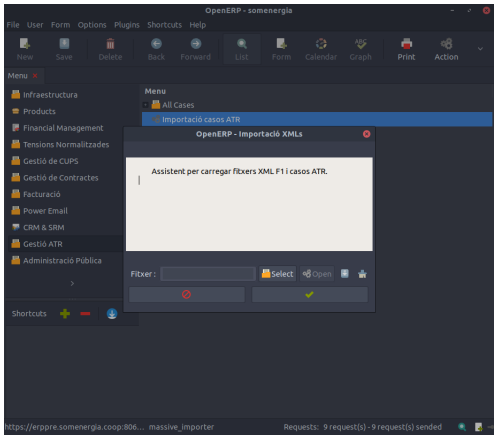


Figure 57: Pas8. Obrir procés d'importació.

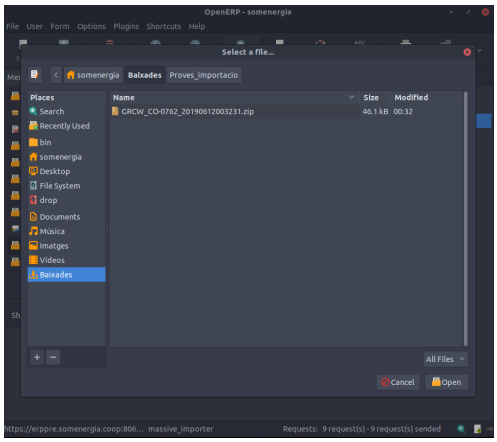


Figure 58: Pas9. Selecció del fitxer prèviament descarregat.

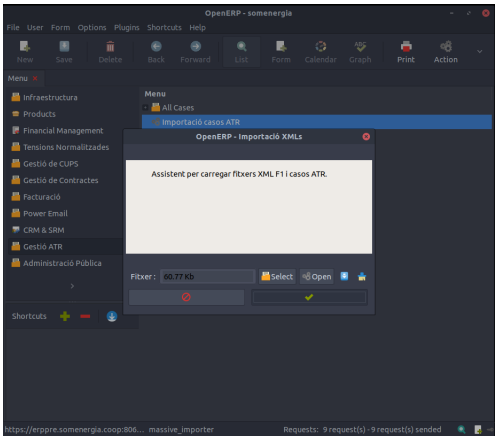


Figure 59: Pas10. Iniciar procés d'importació.

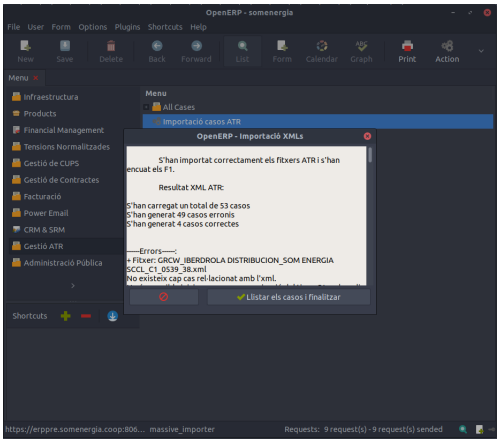


Figure 60: Pas11. Espera prolongada fins la resposta del ERP.