Salt-Monitor:
Aplicació de visualització i instal·lació de serveis al Sincrotró ALBA

Dídac Fuentes Valveny

Resum—Aquest treball es basa en desenvolupar una aplicació que permeti controlar les versions i l'estat del software instal·lat als ordinadors del sincrotró ALBA. Aquest softaware consisteix en un conjunt d'aplicacions creades específicament per el funcionament de les línies de llum del sincrotró i per tal de realitzar els experiments i les adquirir les dades d'aquets. Aquestes aplicacions es desenvolupen al sinctrotro ja que són molt especifiques i per tant, també s'han d'anar actualitzant constantment per tal de que el sincrotró estigui sempre a la última. L'objectiu és crear una aplicació funcional que visualitzi aquesta informació en temps real, comparant-la amb un repositori Git i permetent actualitzacions automàtiques dels serveis obsolets. El treball aconsegueix proporcionar un control de versions i seguiment continuat de les màquines del sincrotró per al departament de Control Systems i l'aplicació ha sigut instal·lada en els ordinadors del sincrotró perquè els treballadors la puguin utilitzar diàriament per realitzar la seva feina.

Paraules clau— SaltStack, Beamlines, Report, PyQt5, Git, CLI, YAML, MySQL, Docker

Resumen—Este trabajo se basa en desarrollar una aplicación que permita controlar las versiones y el estado del software instalado en las computadoras del sincrotrón ALBA. Este software consiste en un conjunto de aplicaciones creadas específicamente para el funcionamiento de las líneas de luz del sincrotrón y para realizar experimentos y adquirir datos. Estas aplicaciones se desarrollan en el sincrotrón ya que son muy específicas y, por lo tanto, también deben actualizarse constantemente para mantener el sincrotrón actualizado. El objetivo es crear una aplicación funcional que visualice esta información en tiempo real, comparándola con un repositorio Git y permitiendo actualizaciones automáticas de los servicios obsoletos. El trabajo logra proporcionar un control de versiones y seguimiento continuo de las máquinas del sincrotrón para el departamento de Control Systems, y la aplicación ha sido instalada en las computadoras del sincrotrón para que los trabajadores la puedan utilizar diariamente en sus labores.

Palabras clave— SaltStack, Beamlines, Report, PyQt5, Git, CLI, YAML, MySQL, Docker

Abstract—This work is based on developing an application that allows controlling the versions and status of the software installed on the computers of the ALBA synchrotron. This software consists of a set of applications specifically created for the operation of the synchrotron's beamlines and for conducting experiments and acquiring data. These applications are developed within the synchrotron as they are highly specialized and therefore require constant updates to keep the synchrotron up to date. The goal is to create a functional application that visualizes this information in real-time, comparing it with a Git repository and enabling automatic updates of outdated services. The work achieves providing version control and continuous monitoring of the synchrotron's machines for the Control Systems department, and the application has been installed on the synchrotron's computers for the daily use of the workers in their tasks.

Index Terms— SaltStack, Beamlines, Report, PyQt5, Git, CLI, YAML, MySQL, Docker

- *E-mail de contacte: didac.fuentes@gmail.com*
- *Treball tutoritzat per: Eduardo Cesar (Arquitectura de Computadors i Sistemes Operatius)*
- *Curs 2022/23*

1 INTRODUCCIÓ - CONTEXT DEL TREBALL

EL Sincrotró ALBA[1] és un institut de recerca científica ubicat al Parc Tecnològic del Vallès, molt a prop de la UAB, on podem trobar un accelerador d'electrons amb l'objectiu de generar llum de sincrotró i utilitzar aquesta llum per fer experiments que permeten estudiar materials a nivell atòmic i estudiar les seves propietats.

Al voltant de l'accelerador de partícules de l'ALBA podem trobar 12 laboratoris que reben la llum de sincrotró que es genera en l'accelerador i que s'anomenen línies de llum o Beamlines en anglès. En cada Beamline podem trobar diferents ordinadors (més de 100) que controlen motors i sensors per operar la línia. En cada ordinador hi ha instal·lat software específic desenvolupat pel departament de Control Systems del sincrotró i, aquest software (més de 30 serveis diferents), s'ha d'anar actualitzant constantment pel correcte funcionament dels experiments que es realitzen a les Beamlines i que necessiten aquest software per operar sensors i controlar diferents aparells.

El projecte va sorgir amb la necessitat de què el departament de Controls System pugues tenir una eina per visualitzar en temps real tot el software instal·lat en cada ordinador (màquines o hosts) i la seva versió i saber quines màquines i softwars (serveis) s'havien d'actualitzar.

El treball realitzat l'any passat per l'Anna Hidalgo[2] (estudiant d'enginyeria de Dades de la UAB el curs passat) va començar a desenvolupar aquest projecte dissenyant una base de dades on emmagatzemar la informació que es volia visualitzar i implementant una aplicació en Python per intentar visualitzar aquesta base de dades. Aquell treball va avançar molt en la solució del problema, però es va quedar a mitges perquè l'aplicació era molt senzilla i només es va arribar a provar en un entorn virtual i no en les Beamlines del sincrotró.
En aquest treball es continua implementant la mateixa solució creant la base de dades que ja va ser dissenyada i implementant una nova aplicació similar a la que es va fer l'any passat pero amb moltes més funcionalitats i que a més de visualitzar la base de dades també permet actualitzar els serveis. A continuació, en la seguent llista es poden veure totes les funcionalitats de l'aplicació i quines es trobaven ja implementades (JI) i quines s'han realitzat en aquest treball (AT):

- Llegir la base de dades i un repositori git. (JI)
- Visualitzar la comparació entre les dues fonts en un report. (JI)
- Filtrar les taules per serveis i/o hosts. (JI parcialment)
- Visualitzar en una altre report les versions dels serveis. (AT)
- Fer una versió de consola de l'aplicació (AT)
- Instal·lar o actualitzar serveis en l'aplicació (AT)
- Posar l'aplicació en producció (AT)

La resta d'aquest article està organitzat de la següent manera: En la secció 2 s'expliquen els objectius plantejats a l'inici del treball, la secció 3 explica la metodologia utilitzada per realitzar el treball, en la secció 4 trobem la planificació realitzada a l'inici del treball i en la secció 5 podem veure tot el desenvolupament realitzat al llarg dels últims mesos per tal que totes les parts de l'aplicació funcionin i es compleixin els objectius. Finalment, al final de l'article podem trobar els resultats i les conclusions d'aquest treball.

2 OBJECTIUS

El principal objectiu d'aquest treball és el de continuar desenvolupar l'aplicació que es va començar a implementar l'any passat per tal que els treballadors del departament de Controls Systems puguin fer un control de versions i un seguiment continuat i en temps real de l'estat de totes les màquines del sincrotró a partir de la creació de reports que es generaran a partir d'aquesta aplicació.
Per tal d'assolir l'objectiu principal, a l'inici del treball em vaig plantejar 4 objectius o fites per veure fins on podia arribar amb el temps disponible i els recursos disponibles:

- Tenir una aplicació funcional i que permeti crear taules per fer reports i tenir un control de versions de tots els servies i màquines de les beamlines del sincrotró.
- L'aplicació ha de permetre poder comparar la base de dades amb les versions instal·lades amb la informació d'un repositori git on es troba tota la informació relacionada amb els servis i hosts de les beamlines.
- L'aplicació ha de permetre filtrar els reports que genera segons un llistat de serveis i hosts específics i ha de mostrar els serveis que es troben obsolets.
- Poder visualitzar l'aplicació en una interfície gràfica i en una terminal o consola en dispositius linux.
- L'aplicació ha de permetre actualitzar els serveis amb una versió obsoleta de manera automàtica i integrada dins de la mateixa aplicació.
- Instal·lar l'aplicació a producció i comprovar el seu correcte funcionament de totes les funcionalitats anteriors en l'entorn real de les beamlines del sincrotró.

3 METODOLOGIA

Per fer aquest projecte he utilitzat la metodologia Waterfall o en cascada en la qual he anat desenvolupant les tasques del projecte de manera seqüencial. Per tal de complir amb els objectius he dividit el projecte en diferents tasques com es podrà veure més endavant en l'apartat de planificació i aquestes tasques les he anat desenvolupant de manera independent i seqüencialment. A l'acabar d'implementar cada tasca he comprovat el seu correcte funcionament i l'he integrat al conjunt del projecte. Moltes de les tasques tenen dependències entre elles

i en alguns casos s'han d'implementar de manera conjunta perquè funcionin correctament i és per això que la implementació d'algunes tasques ha sigut més costosa que d'altres, i per això el temps que he trigat a implementar-les ha variat d'1 setmana a 5 setmanes aproximadament com es podrà veure també més endavant en la planificació. Finalment, per controlar que el projecte avances correctament, una vegada al mes vaig avaluar la feina feta i vaig replantejar les prioritats de les tasques a fer per tal d'acabar el projecte i complir amb els objectius.

Durant tot el transcurs de la part d'escriure codi de l'aplicació he utilitzat un repositori git per tenir un control de versions de l'aplicació per tal de tenir tot el progres documentat i poder recuperar versions anteriors del codi en cas d'errors en algunes tasques.

4 Planificació

Seguint la metodologia explicada anteriorment, a l'inici del treball vaig crear un diagrama de Gantt, que podem veure en la fugura 1, amb totes les tasques que havia de realitzar per tal de complir amb els meus objectius. Les tasques les vaig planificar al inici del treball per complir amb els objectius plantejats i algunes corresponen a un objectiu concret i altres són més genèriques però que avancen en el desenvolupament de l'aplicació. Durant els mesos que he estat realitzant aquest treball, he anat realitzant les tasques complint els temps d'aquesta planificació i, en les últimes tasques, vaig poder avançar més ràpid del que vaig planificar i al final vaig poder acabar totes les tasques un mes abans del previst en aquesta planificació. Algunes tasques és superposen en alguns casos, ja que tenen dependències entre elles i les vaig realitzar alhora i intentat seguir la metodologia de desenvolupament en cascada.

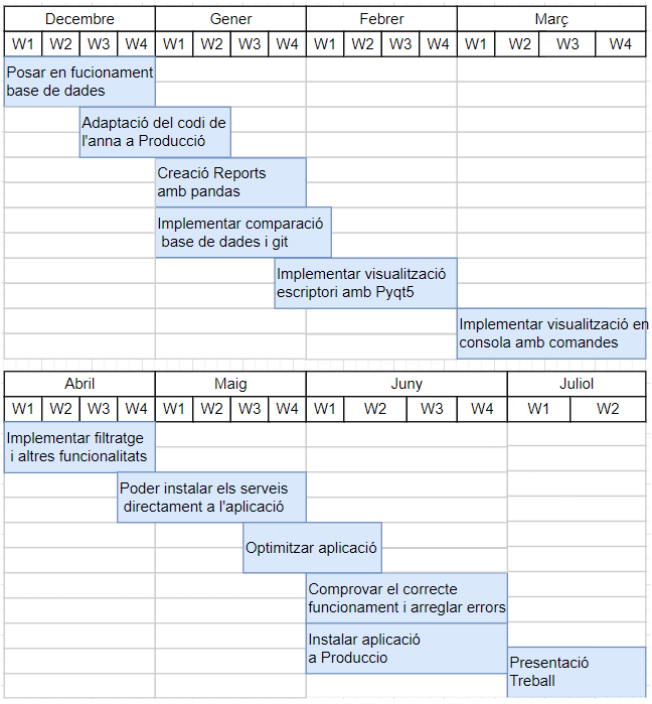

Fig. 1: Diagrama de Gantt del projecte

5 Desenvolupament

En aquest apartat aniré explicant el desenvolupament realitzat per totes les tasques mencionades anteriorment en el diagrama de Gantt, la seva implementació en la part de codi i les eines que he utilitzat per realitzar-les. En la figura 2 podem veure una visió global de com funciona l'aplicació que he implementat i que més endavant aniré explicant cada part. Aquesta arquitectura que he realitzat no té res a veure amb la que es va realitzar en el treball anterior, ja que encara que hi havia algunes parts que ja estaven implementades, per tal que funciones de manera més eficient i es pogués connectar amb les noves funcionalitats vaig haver de reestructurar l'aplicació tal com es veu en aquesta figura. De les dues fonts que s'obtenen la informació, trobem dues subclasses que agafen aquesta informació i en la classe report és comparen aquestes dues fonts per crear dos reports que es poden visualitzar tant en la terminal com en la interfície gràfica (GUI) de la mateixa manera que es pot fer amb la part d'instal·lació.

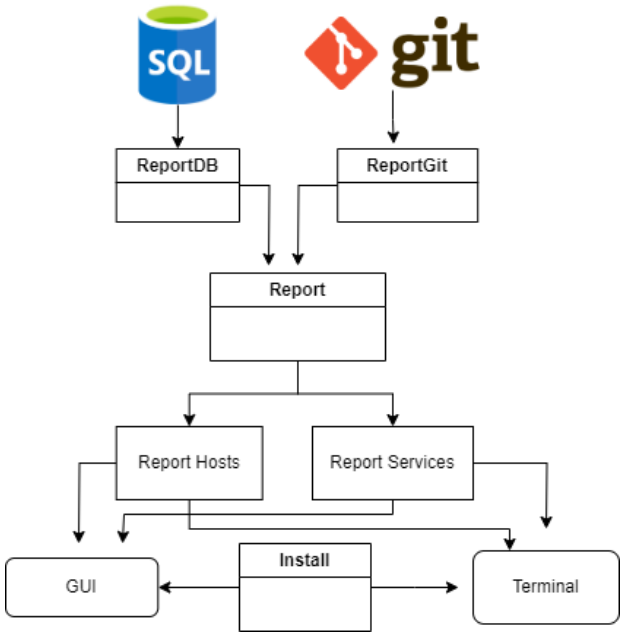

Fig. 2: Arquitecura de l'aplicació

5.1 Posada en funcionament de la base de dades

El primer de tot que vaig realitzar va ser crear i posar en funcionament la base de dades que va ser dissenyada en el treball anterior per tal de que l'aplicació pugui funcionar amb dades reals del sincrotró. Aquesta base de dades consta d'una única taula en la que cada vegada que s'instal·la o s'actualitza un servei s'emmagatzema tota la informació relacionada amb la instal·lació incloent el host, el servei i la seva versió. La base de dades s'ha creat amb MySQL[3] i es troba allotjada a una màquina virtual dintre de la intranet del sincrotró. Les instal·lacions dels serveis als hosts es realitzen amb una eina anomenada Saltstack[4] (de la qual explicaré el seu funcionament complert més endavant) que executa unes receptes amb instruccions en las que s'especifica com instal·lar cada

servei. Per tal que es registrin totes les instal·lacions a la base de dades vaig afegir una instrucció adicional a les receptes en la que s'especifica que cada vegada que es produeix una instal·lació es registri la seva informació a la base de dades. D'aquesta manera la base de dades s'omple automaticament cada vegada que s'instal·la o s'actualitza algun servei en algun host. La taula de la base de dades enmagatzema la següent informació:

TAULA 1: TAULA BASE DE DADES

ID	Domain	Subdomain	Host	Service
1	BL	BL01	pcbl0101	tango
2	BL	BL04	pcbl0401	taurus

Version	Who	Installed
2021202	User	10/10/2021 (09:26:03)
2020212	User	15/01/2022 (07:40:33)

- Domini i subdomini de la beamline on es troba el host en el que es produeix la instal·lació.
- Nom del host on s'està instalant el servei.
- Nom del servei que s'instal·la.
- Numeró de versiò del servei que s'està instal·lant.
- Usuari que ha efectuat la instal·lació.
- Data i Hora en la que s'ha realitzat la instal·lació.

5.2 Context i documentació del treball anterior

Per tal de començar a implementar la meva part de codi de l'aplicació vaig haver de documentar-me de tot el codi que ja es va realitzar l'any anterior i com funcionava aquella primera versió de l'aplicació. Com he explicat abans, la primera versió només funcionava en un entorn virtual basat en 7 dockers[5] que simulaven una beamline amb 5 hosts, un docker per la base de dades i un per executar l'aplicació. Aquesta aplicació estava desenvolupada amb la llibreria PyQt5[6] de Python que permet crear interfícies gràfiques amb la possibilitat d'interacturar amb l'usuari de diferents maneres. L'aplicació, que podem en la figura 3, només generava un report que es podia filtrar segons la beamline, el host i el servei i un cop seleccionats aquestes tres opcions creava una taula on les files són els serveis i les columnes els hosts.

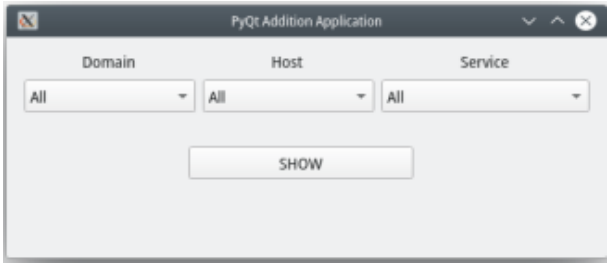

Fig. 3: Intefície gràfica que es va desenvolupar l'any passat

5.3 Creació dels Reports amb Pandas

Per la creació dels reports em vaig basar en la que ja es va fer l'any passat que consistia en una taula on les files eren els serveis i les columnes els hosts. Les caselles de la taula poden contenir el número de la versió del servei que es troba instal·lat en el host o poden estar buides si un servei no ha d'estar instal·lat en algun host. En la versió antiga només hi havia 2 colors per diferenciar les versions i en el report actual, que podem veure en la figura 4, podem trobar fins a 5 colors diferents amb els següents significats:

- **Stable:** La versió instalada en el host es la ultima disponible estable.
- **Unstable:** La versió instalada en el host es anterior a la ultima disponible estable i s'ha d'actualitzar.
- **Warning:** El servei no es troba instalat al host o registrat a la base de dades i ho haruria d'estar.
- **Frozen:** La versió instalada es diferent a la estable però no s'ha d'actualitzar ja que s'ha congelat a aquesta versió especifica per un motiu específic.
- **Installed:** Serveis instalats que no tenen número de versió i per tant es mostra la data de la ultima vegada que es van instalar en el host.

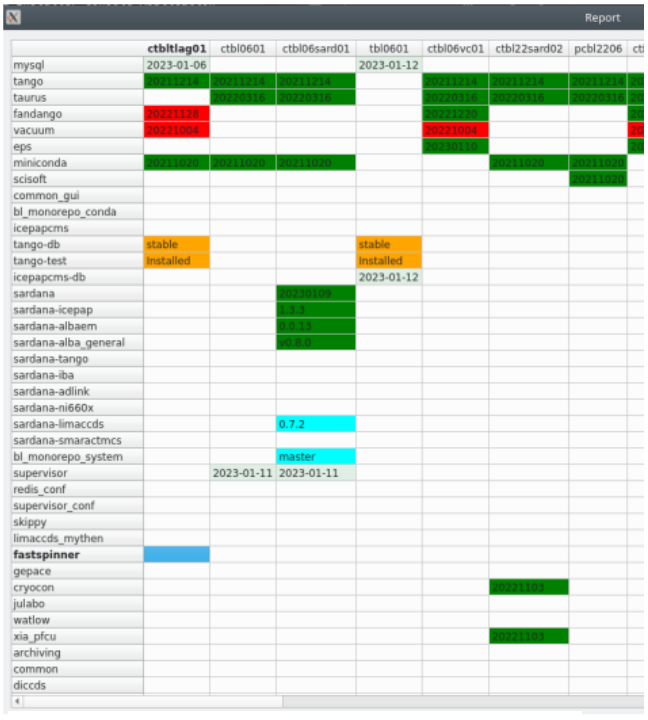

Fig. 4: Report principal que genera l'aplicació

També més endavant vaig crear un segon report que mostra totes les versions disponibles de cada servei. Aquest report només conté informació relacionada amb els serveis i consta d'una taula on les files són els serveis i les columnes els sistemes operatius de la versió sent de moment només tres: common que es refereix a què la versió és comuna per tots els sistemes operatius, Debian 9 i Debian 10 [7]. Aquesta taula, que es pot veure en la figura 5, conté el número de versió estable de cada servei amb els següents 3 colors:

	common	debian10	debian9
common	No version		
miniconda	20211020		
mysql	No version		
supervisor	No version		
icepapcms-db	No version		
tango		20211214	20211214
tango-db		20211214	20211214
tango-test	No version		
moco		20211020	20211020
taurus		20220316	20220316
sardana		20230109	20230109
timing		20211020	20211020
fandango		20221220	20221220
skippy		20211020	20211020
archiving			20221110
vacuum		20221110	20230120
eps		20230110	20230110
nhq_iseg			20211020
cyberstarx			20211020
oxfcryo700		20211020	20211020
limaccds_basler	20211130		
limaccds_web_bpm	20211020		
limaccds_mythen	20221107		
icepapcms	20230119		
scisoft	20211020		

Fig. 5: Segon Report que genera l'aplicació amb les versions dels serveis

• **Stable:** La versió estable del servei al sistema operatiu específic és la última disponible.
• **Warning:** La versió del servei estable del Sistema operatiu no es la última disponible i després de fer les proves pertinents s'hauria de posar com a estable la última versió.
• **No Version:** El Servei no té una versió especifica sino que es un paquet Debian que s'actualitza automaticament i que per tant no té versió.

5.4 Comparació base de dades amb repositori Git

Per obtenir la informació en temps real dels reports explicats en l'apartat anterior cal que l'aplicació estigui connectada a la base de dades explicada anteriorment i a un repositori Git[8] intern del sincrotró anomenat Salt-Base. En aquest repositori podem trobar tota la informació relacionada amb els serveis i les seves versions i, de cada host, podem trobar un arxiu on s'especifica quins serveis s'han d'instal·lar.

Per llegir aquestes dues fonts de dades, el que he fet en el codi Python ha sigut un sistema de classes que es connecten entre elles. El codi consta de dues subclasses, anomenades "reportgit" i "reportdb", i una principal anomenada "report" que recull les dues subclasses anteriors i compara la informació. En la classe "reportdb" em connecto remotament a la base de dades amb la llibreria de Python mysql-connector[9] i faig querys amb SQL per obtenir la informació. En la classe "reportgit" utilitzo la llibreria GitPython[10] per descarregar (o actualitzar si ja s'ha descarregat anteriorment) tot el repositori en una carpeta de l'ordinador local. Dintre de la carpeta utilitzo les llibreries Glob i Os per buscar i llegir els arxius on es troba la informació que necessito, que són els següents:

- Un arxiu per cada host amb el llistat de serveis que han d'anar instal·lats en el host.
- Un arxiu per cada servei amb el número de versió estable i un llistat amb la resta de versions antigues.

En la classe principal "report" utilitzo la llibreria Pandas[11] per guardar els reports finals i faig les següents comparacions entre les dues fonts per obtenir-los:

- En tres variables guardo un llistat de tots els hosts, serveis i dominis diferents que podem trobar tant en la base de dades com en el repositori per després poder filtrar els reports.
- Un panda Dataframe el qual conté el valor de la última versió obtinguda del últim registre de la base de dades corresponent al host i servei pertinent.
- Un panda Dataframe amb el color corresponent a cada casella del dataframe anterior el qual s'obté de comparar el valor de la casella amb el que trobem al repositori git.

5.5 Visualització amb Pyqt5 en interficie grafica

La interfície gràfica (o gui) de l'aplicació és completament nova respecte a la que es va realitzar l'any passat i en aquest cas a més d'utilitzar la llibreria PyQt5 per programar-la, també he utilitzat un software anomenat QTDesigner[12] que serveix per crear frontends d'aplicacions de manera dinàmica amb Qt i que després es pot exportar a codi Python i connectar amb la part de backend.

A continuació, en les figures 6 i 7, es poden veure les pantalles principals que generen els dos reports explicats anteriorment. En les dues pantalles es mostra la llegenda de colors de cada report per poder entendre el report que genera i totes les opcions de filtratge disponibles.

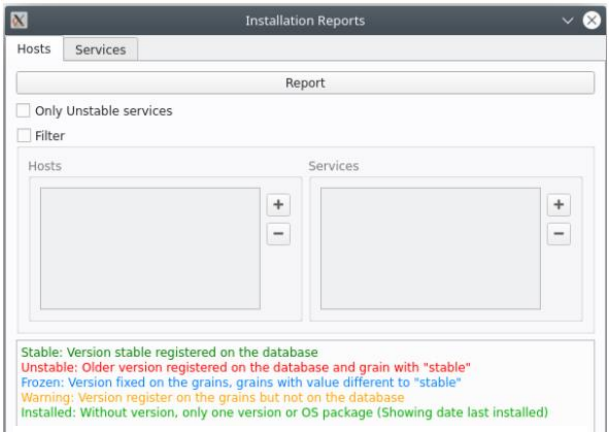

Fig. 6: Pantalla principal del report dels hosts

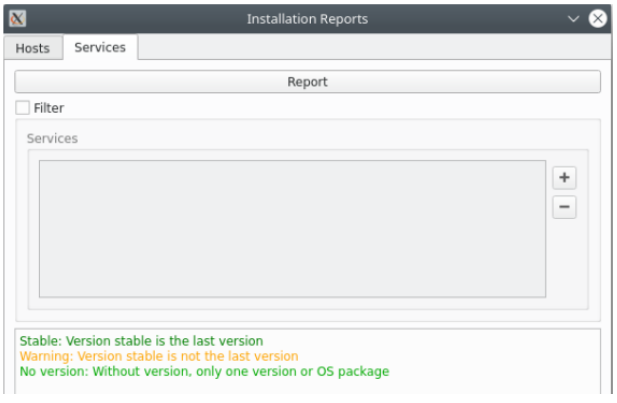

Fig. 7: Pantalla principal del report de les versions dels servis

5.6 Visualització amb Click en consola

Una vegada vaig tenir la part d'interfíce gràfica acabada, vaig començar a implementar la visualització dels reports en la terminal. Per aquesta part he utilitzat la llibreria Click[13] de Python que et permet crear comandes en la qual es pot assignar una funció a cada una. A continuació, en la figura 8, podem veure el missatge inicial quan s'executa l'aplicació i la comanda "help" que et mostra totes les comandes disponibles i que fa cada una.

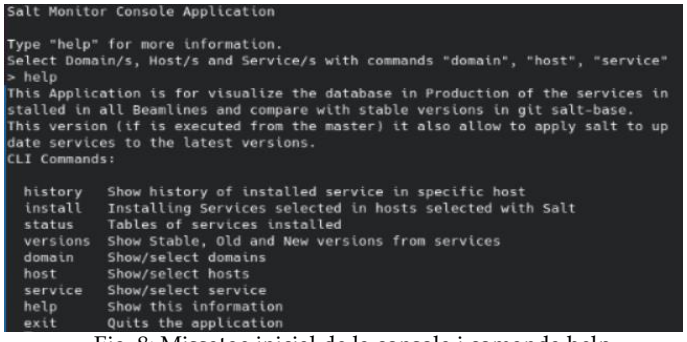

Fig. 8: Missatge inicial de la consola i comanda help

A continuació, en la figura 9, podem veure els dos principals reports ja explicats en l'apartat 5.3 en la seva versió de consola que han estat generats amb la llibreria tabulate[14] a partir d'un panda dataframe:

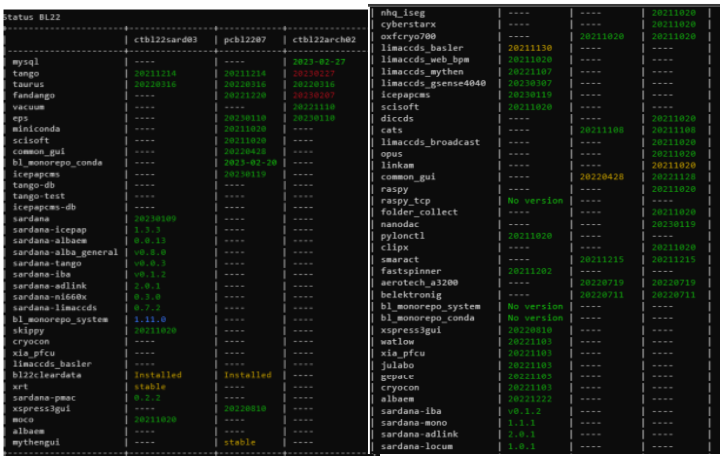

Fig. 9: Visualitzacions dels dos reports en versió consola

5.7 Implementació de filtratge i altres funcionalitats

Aquestes són totes les funcionalitats de filtratge que he implementat per visualitzar els reports de manera més simplificada i que es troben disponibles tant en la versió de gui com en la versió de consola:

- En el report principal es pot seleccionar l'opció de mostrar només els serveis que no estan estables. Aquesta opció genera un report amb només els serveis i hosts on la versió es troba obsoleta i, per tant, alguna casella és de color vermell. Aquesta opció es molt útil perquè de manera ràpida i senzilla es pot fer servir per saber quants i quins són els serveis que s'han d'actualitzar.
- En els dos reports hi ha una opció la qual et permet seleccionar un llistat de serveis per mostrar només un o varis d'ells en el report. Aquest llistat permet afegir i eliminar tants serveis com es vulgui i per defecte si no es selecciona cap es mostren tots al report.
- En el report principal a més de poder seleccionar els serveis també es pot seleccionar un llistat de hosts. Al llistat de hosts també apareixen els dominis de les beamlines els quals si els selecciones s'afegiran automaticàment al llistat tots els hosts de la beamline.

A continuació, en la figura 10, podem veure un exemple de report amb l'opció de mostrar només les versions no estables.

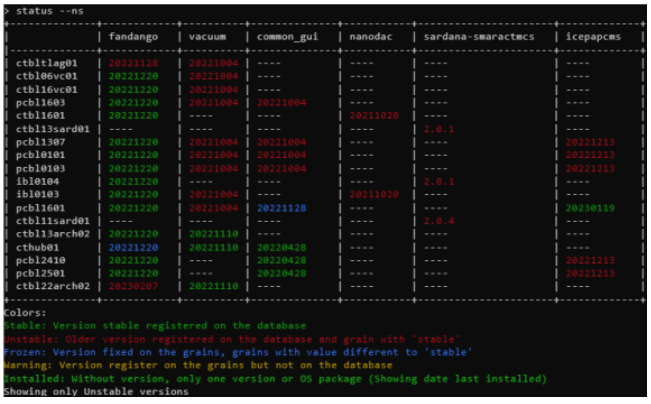

Fig. 10: Report principal en versió consola amb la opció ns

A continuació podem veure dues taules extres que es poden obtenir clicant qualsevol casella amb contingut als reports principals. Aquesta funcionalitat la vaig implementar per poder veure més informació adicional i que serveix com a historial.

La imatge de l'esquerra de la figura 11 és la que es genera al clicar una casella on contingui una versió instal·lada a un host. El primer element de la llista és el que es pot veure en el report i la resta són totes les instal·lacions anteriors que podem trobar a la base de dades sobre el servei i el host específic. Es mostra el número de versió i la data en la qual es va realitzar l'instal·lació.

En la imatge de la dreta de la figura 11 podem trobar el llistat de versions antigues d'un servei. Es genera al clicar qualsevol versió del segon report explicat a l'inici. Aquest llistat és molt útil per saber tot l'historial d'un servei específic en un sistema operatiu concret.

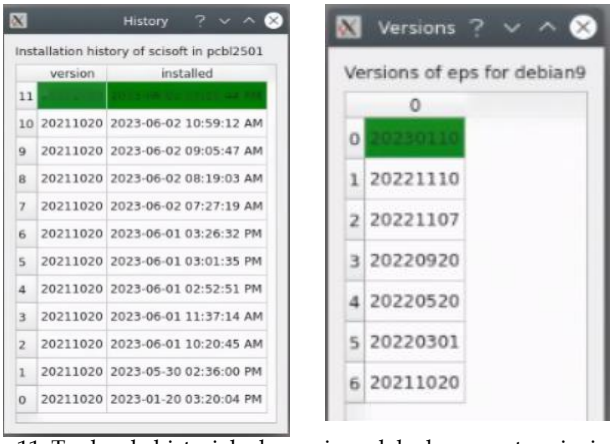

Fig. 11: Taules de historials de versions dels dos reports principals

5.8 Instalació dels serveis en l'aplicació

Finalment, després d'implementar totes les funcionalitats explicades anteriorment, només falta la part d'instal·lació la qual ha estat la part més costosa d'implementar, ja que em va caldre aprendre molta documentació sobre SaltStack que és l'eina que utilitzen al sincrotró per instal·lar tots els serveis de manera remota i automàtica.

SaltStack és un software lliure que serveix per realitzar instal·lacions de manera remota i centralitzada. El Salt Master és l'ordinador principal des d'on s'envien les receptes, que es llegeixen des del repositori git mencionat anteriorment, als Salt Minons que són tots els ordinadors els quals es poden controlar remotament des del Salt Master. Les receptes utilitzen el format YAML[15] i plantilles Jinja[16] on podem trobar totes les instruccions específiques per instal·lar un software concret. En el sincrotró totes les màquines de les beamlines que es poden veure en els reports són Salt Minions i el Salt Master és una màquina virtual del sincrotró creada específicament per aquesta funció i és des d'on s'executarà l'aplicació per actualitzar els serveis obsolets.

Aquest software s'executa a través de diferents comandes a la terminal i és com ho fan al sincrotró però també té una llibreria Python (anomenada salt[17]) oficial que permet programar les comandes directament en el codi Python i que es la que he utilitzat al meu codi per automatitzar el procés i poder instal·lar directament a través del frontend de la meva aplicació.

A continuació, en la figura 12, podem veure la intefície gràfica de la part d'instal·lació en la que podem trobar diferents opcions per especificar quins serveis es volen actualitzar i a quins hosts. També hi ha l'opció de test que si està activada només es realitza una simulació de la instal·lació i en el resultat et mostra si s'hagués produït algun error en el cas que s'hagués instal·lat. Es pot seleccionar la branca del repositori de la qual s'agafaran les receptes per instal·lar, però aquesta opció és només per casos especifics en els quals es fan proves concretes i per defecte és sempre master. Finalment, es pot escollir la carpeta on es guardaran els arxius amb el resultat de la instal·lació (un arxiu per cada host seleccionat) per consultar si s'ha produït algun error i comprovar les versions instal·lades de cada servei.

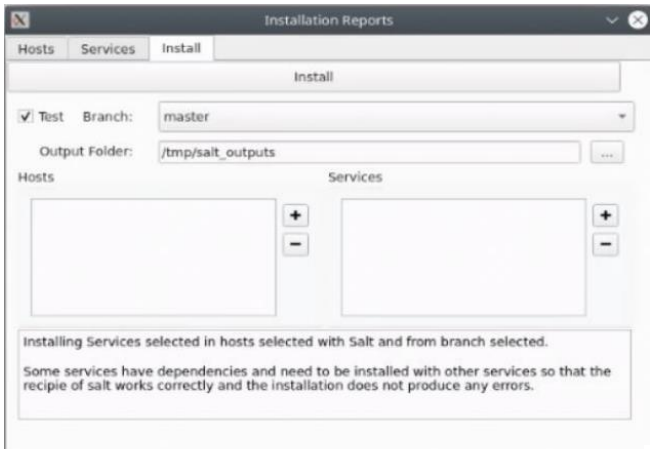

Fig. 12: Pantalla principal de la part d'instalació

En la figura 13 es mostra la pantalla de la interfíce gràfica en el cas d'actualitzar el servei "tango" en diferents hosts. Com es pot veure en aquest cas no s'ha produït cap error i simplement surten els noms dels arxius que ha generat l'aplicació amb el llistat de totes les instruccions que s'han executat i el seu resultat. Els noms dels arxius consten del nom del host al que pertany més la data en la qual s'ha produït la instal·lació per aquesta manera evitar sobreescriue un arxiu si es fan més d'una instal·lació en el mateix host.

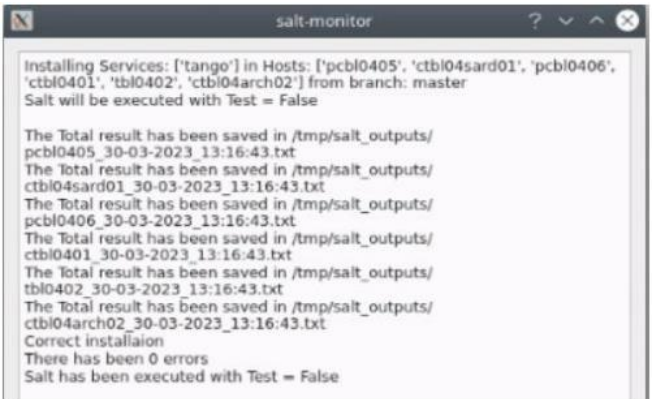

Fig. 13: Resultat en inteficie grafica de una instalació sense errors

En la figura 14 es pot veure com es mostra el resultat en la versió de consola i en aquest cas amb algun error en la instal·lació. Abans d'instal·lar hem seleccionat que volíem instal·lar el servei "sardana" en el host "ctblvm-sard01" i després hem executat la comanda install. Com es pot veure s'han produït 10 errors que es mostren a continuació i això és degut a que, com s'indica en un missatge abans de confirmar la instal·lació, alguns serveis no es poden instal·lar individualment perquè tenen dependències amb altres serveis i que, per tant, s'han d'instal·lar o actualitzar conjuntament. A l'executar la comanda amb test= True realment no s'han produït aquests errors perquè només s'ha simulat, però d'aquesta manera podem veure que hauria fallat i comprovar en les dependències dels errors que per poder instal·lar el servei de "sardana" són necessaris també els serveis de "tango" i "taurus".

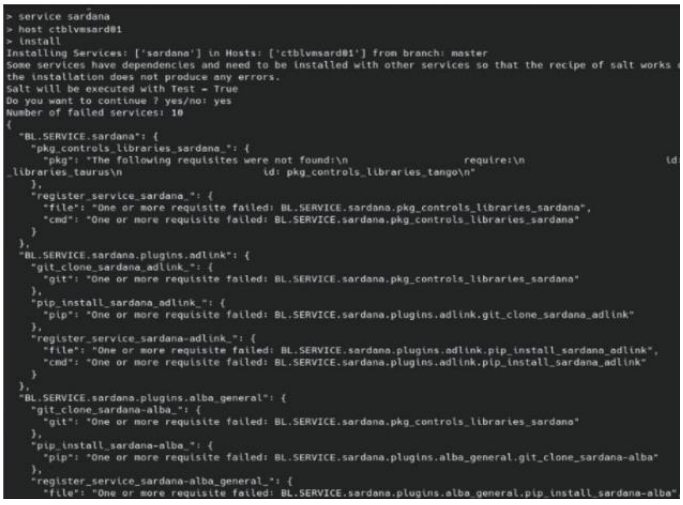

Fig. 14: Resultat en consola de una instalació amb alguns errors

6 Resultats

Un cop acabada la part d'implementació de totes les funcionalitats de l'aplicació vaig passar a la part de testeig i optimització. En aquesta part vaig trobar petits errors que s'em van passar per alt i que només és produïen en casos molt específics, però que vaig poder solucionar. En la part d'optimització vaig dedicar més temps perquè cada vegada que l'aplicació generava un report trigava uns segons i això o vaig poder reduir a tan sols 1,5 segons graciés a evitar haver de calcular cada vegada des de zero els reports i calcular únicament una vegada tota la informació i mostrar només la que es demana cada vegada que es vol generar un report. També la utilització de querys eficients de la llibreria Pandas i altres funcions d'aquesta llibreria i la de Numpy[18] van permetre reduir el temps considerablement. En la part d'instal·lació no vaig poder reduir el temps, ja que no depèn del meu codi sinó del temps que es triga a instal·lar cada servei que pot variar de uns segons i fins i tot alguns minuts. Tan sols vaig poder paral·lelitzar la instal·lació simultànea del mateix servei en diferents hosts alhora en el cas que es seleccioni més d'un host.

El resultat final de l'aplicació, després de comprovar que no tingués cap error i s'hagués optimitzat perquè fos eficient, es va decidir empaquetar per a la seva distribució interna al sincrotró. Vaig generar un paquet Conda[19] amb tot el codi incloent les dues versions de visualització i totes les dependències que té l'aplicació. Un cop creat el paquet es va instal·lar a producció en la màquina virtual Salt-Master que és l'únic lloc on funciona la part d'instal·lació i els treballadors es podran conectar remotament per utilitzar l'aplicació per actualitzar els serveis. La part de visualització dels reports es pot executar des de qualsevol ordinador intern del sincrotró perquè necessita accés a la base de dades i al repositori git que es troben a la intranet del sincrotró i amb el paquet creat de conda els treballadors poden instal·lar l'aplicació en el seu ordinador local per crear i visualitzar els reports i fer un control de versions que era l'objectiu principal pel qual es va decidir començar a desenvolupar aquesta aplicació ara fa més d'un any.

El nom escollit de l'aplicació va ser una decisió del departament de Controls Systems que són els que la faran servir i van decidir que **Salt-Monitor** era un bon nom, ja que fa referència al fet que serveix per monitoritzar tot el sistema d'instal·lacións que es fan amb el sistema de Salt. Jo crec que és un bon nom per aquesta aplicació que inicialment l'any passat tenia el nom provisional de Sicilia Installer, però que es va descartar perquè la funció principal de l'aplicació és la de crear reports i la part d'instal·lació és un afegit que inicialment no estava clar que es pogués fer, però que finalment he pogut desenvolupar i implementar.

Finalment, vull explicar que des del sincrotró porten unes setmanes utilitzant l'aplicació i m'han informat que els hi està sent molt útil i que els hi està estalviant molta feina, ja que era una aplicació que portaven demanant un temps des que el sistema de Salt va créixer molt amb més de 100 hosts i més de 30 serveis diferents.

7 Conclusions

Com a conclusió final del projecte puc afirmar que s'ha complert l'objectiu principal de tenir una aplicació funcional que els treballadors del sincrotró puguin utilitzar per fer la seva feina. També he pogut completar tots els objectius que es van plantejar a l'inici, inclòs l'objectiu d'instal·lar els serveis, el qual no estava segur de si el podria complir per la seva dificultat, però que al final vaig poder desenvolupar i acabar l'aplicació 3 setmanes abans de la planificació inicial.

Aquesta aplicació l'he desenvolupat perquè funcioni exclusivament en el sincrotró ALBA, ja que té moltes dependències específiques sobre el sistema que ja tenen implementat, però també l'he implementat amb una certa intel·ligencia i pugui ser capaç de tenir en compte futures ampliacions o actualitzacions del sistema d'instal·lació de serveis. Algun exemple d'aquesta intel·ligència de l'aplicació és la de què en qualsevol moment es poden afegir nous hosts, serveis o inclús noves beamlines, i el codi estarà preparat per continuar funcionant per mostrar cada vegada més informació a mesura que es vagi ampliant. També està preparat en cas que les màquines s'actualitzin a noves versions del sistema operatiu Debian, o inclús d'altres nous possibles sistemes operatius. Finalment, també està preparat en cas que en el repositori git es produeixin modificacions o s'hagi de crear una nova base de dades en una nova localització.

Desenvolupar aquest projecte m'ha servit en molts aspectes per aprendre moltes eines i tècniques de programació noves, i també com desenvolupar una aplicació pràcticament des de zero fins a la part final. Des del sincrotró ALBA m'han ensenyat molts coneixements que després em seran molt útils pel món laboral i, per tant, he adquirit molta experiència.

Agraïments

Voldria agrair a tots els companys del departament de Controls Systems del sincrotró ALBA que m'han ajudat a aprendre moltes coses i principalment al Roberto, el meu tutor de pràctiques al sincrotró que m'ha ajudat molt en tot el procés. També vull agrair al Eduardo Cesar el meu tutor a la universitat per tots els consells que m'ha donat en la realització dels informes i com millorar-los.

Bibliografia

[1] Pagina web oficial del Sincrotró ALBA: https://www.albasynchrotron.es/

[2] TFG de l'Anna Hidalgo realitzat el curs passat: https://ddd.uab.cat/record/264626

[3] Documentació sobre la base de dades relacional mysql: https://www.mysql.com/

[4] Pagina web oficial i documentació de l'eina SaltStack: https://saltproject.io/

[5] Documentació oficial dels Dockers: https://www.docker.com/

[6] Documentació oficial de la llibreria Python PyQt5: https://pypi.org/project/PyQt5/

[7] Pagina web oficial del sistema operatiu basat en Linux Debian: https://www.debian.org/

[8] Documentació relacionada amb els repositoris git: https://git-scm.com/

[9] Documentació oficial de la llibreria mysql-connector Python: https://dev.mysql.com/doc/connector-python/

[10] Documentació oficial de la llibreria GitPython: https://gitpython.readthedocs.io/en/stable/

[11] Documentació oficial de la llibreria de Python Pandas: https://pandas.pydata.org/

[12] Documentació oficial del software de disseny QT Designer: https://doc.qt.io/qt6/qtdesignermanual.html

[13] Documentació oficial de la llibreria Python Click: https://click.palletsprojects.com/

[14] Documentació oficial de la llibreria de Python tabulate: https://pypi.org/project/tabulate/

[15] Explicació del funcionament del format YAML: https://www.redhat.com/es/topics/automation/what-is-yaml

[16] Explicació del funcionament de les receptes Jinja de Salt: https://docs. saltproject.io/en/latest/topics/jinja/index.html

[17] Documentació oficial de la llibreria per Python de Salt: https://docs. saltproject.io/en/latest/ref/ clients/index.html

[18] Documentació oficial de la llibreria de Python Numpy: https://numpy.org/

[19] Documentació oficial del sistema de creació de paquets Conda: https://docs.conda.io/en/latest/