Desenvolupament d'una eina d'anàlisis de logs per a detecció d'errors i seguiment de mostres

Albert Simon Roig

Resum– L'automatització total de laboratoris farmacèutics aviat serà una realitat. Mentre aquests laboratoris no estan llestos, s'utilitzen simuladors per provar el seu funcionament. Aquest projecte té com a objectiu comprovar si els simuladors funcionen correctament. Per a fer-ho s'utilitzen els logs, les traces i missatges que aquests generen i es centralitzen en una base de dades. Aquestes dades es tracten per poder generar més informació a partir de l'informació disponible. D'aquesta forma es pot detectar quin component no funciona correctament. El projecte també ofereix altres funcionalitats com el seguiment de mostres o mètriques de les simulacions. Aquestes dades permeten a l'usuari determinar si el laboratori esta ben dimensionat. El projecte es serveix en una aplicació web per facilitar l'accés a tots els usuaris.

Paraules clau– Laboratori, logs, traces, anàlisi, aplicació web, big data.

Abstract– Full automation of pharmaceutical laboratories will soon be a reality. While these labs are not ready, simulators are used to test their operation. This project aims to check if the simulators work properly. To do this, the logs, traces and messages that they generate are used and centralized in a database. This data is processed in order to generate more information from the available information. This way the program can detect which components are not working properly. The project also offers other features such as sample tracking or simulation metrics. These data allow the user to determine if the laboratory is well sized. The project is served in a web application to facilitate access to all users.

Keywords– Laboratory, logs, analysis, web application, big data.

1 INTRODUCCIÓ

AQUEST projecte neix amb l'objectiu de resoldre una necessitat real a l'empresa on treballo.

Hoffmann-La Roche és una empresa dedicada a crear solucions tecnològiques i serveis per a laboratoris d'anàlisis clínics, centres d'investigació, consultes mèdiques i farmàcies [1]. El projecte en el que em centro jo és l'automatització total de grans laboratoris d'anàlisis de fluids. Concretament em centro en la supervisió d'aquest. Aquests laboratoris han de ser capaços de tractar les mostres desitjades sense la necessitat de cap interacció humana.

Per poder comprovar el correcte funcionament de cada màquina del laboratori (tant els components de software com els components de hardware) i el seu comportament al combinar-les s'utlitzen simulacions. Aquestes representen un laboratori real sotmès a unes determinades condicions. El simulador permet triar l'estructura o *layout* del laboratori i les mostres que s'han d'analitzar. D'aquesta forma es pot assegurar un correcte funcionament del software abans d'instal·lar-lo en un entorn real.

El projecte es troba en una fase avançada de desenvolupament, tot i així encara hi ha algunes necessitats a cobrir. Actualment no hi ha cap sistema d'anàlisis de dades generades per les simulacions. Aquestes dades generades són el conjunt de missatges intercanviats entre els components del laboratori i el software central que els domina.

Sense la capacitat d'explotar les dades generades és impossible fer un rastreig d'una mostra de forma fàcil. Tampoc

• E-mail de contacte: albert.simonr@e-campus.uab.cat
• Menció realitzada: Tecnologies de la Informació
• Treball tutoritzat per: Rafael Fernández (dEIC)
• Curs 2019/20

és possible detectar les causes dels errors del software.
És desitjable saber què ha passat en el moment en que el simulador ha deixat de funcionar per un error i en instants anteriors. Amb aquesta informació es podria detectar quin mòdul del software té un funcionament erroni.

En aquest document s'explica amb detall el desenvolupament del projecte. Primer es plantegen els objectius i la metodologia utilitzada. Més endavant s'explica l'arquitectura de l'aplicació dissenyada i les tecnologies utilitzades per dur-la a terme.
Seguidament es repassen detalladament els resultats obtinguts en el projecte i quines tècniques s'han utilitzat per comprovar el correcte funcionament del software.
Finalment s'esmenten els pròxims passos del projecte, com es podria ampliar i millorar i s'exposen les conclusions.

2 ESTAT DE L'ART

Actualment hi ha diverses eines per construir un big data. La peça més important és la base de dades, aquí s'han de diferenciar les bases de dades relacionals de les no relacionals.
Tot i que els dos tipus de bases de dades podrien satisfer les necessitats del projecte s'ha optat per una base de dades no relacional.

ELK és un conjunt de projectes que permeten un fàcil tractament de grans volums de dades [2]. El primer projecte és Elasticsearch, una base de dades no relacional que permet buscar dades fàcilment utilitzant notació JSON. El segon projecte és Logstash, una eina de processament de dades que permet transformar les dades que es volen enviar a Elasticsearch de forma ràpida. Finalment, el tercer projecte dins ELK és Kibana, una aplicació web per visualitzar les dades emmagatzemades a Elasticsearch.

3 OBJECTIUS

L'objectiu final del projecte és facilitar el procés d'anàlisis de les simulacions realitzades. Per aconseguir-ho, s'han proposat uns quants objectius concrets:

- Permetre a l'usuari realitzar el seguiment de mostres a través d'una interfície gràfica. Aquest seguiment s'ha de poder visualitzar a través d'una línia del temps que ensenyi l'activitat de la mostra dins el laboratori (màquines per les que ha passat, segons que ha estat en cada màquina i nombre de desplaçaments dins el laboratori).

- Permetre a l'usuari capturar instantànies del sistema en moments determinats a través d'una interfície gràfica. Aquestes instantànies han de mostrar el número de mostres que hi ha dins de cada màquina del laboratori al moment desitjat.

- Permetre a l'usuari visualitzar informació i mètriques d'una simulació un cop aquesta ha acabat. Aquestes mètriques han de mostrar de forma representativa com ha funcionat la simulació.

Amb el compliment d'aquests tres objectius específics, els usuaris de l'aplicació han de ser capaços de satisfer les seves necessitats.

A part d'aquests objectius principals, també s'espera que l'aplicació compleixi una altre llista de requisits:

- Fàcil d'utilitzar per a tots els usuaris.

- Temps de resposta de l'aplicació raonables.

Al mes de maig, quan el projecte ja estava força avançat, durant una reunió de presentació de l'estat de l'aplicació es va decidir afegir alguna funcionalitat nova a petició del client (l'empresa). Aquestes funcionalitats són:

- **Llistar les mostres amb error:** A la pantalla de resum de la simulació, mostrar una llista de les mostres que han acabat el seu trajecte amb algun error. D'aquesta forma, l'usuari pot veure més ràpidament si hi ha un error al laboratori o en cas contrari, si les mostres s'han introduït malament en aquest.

- **Suport per a diverses simulacions:** En un primer moment no es tenia clar com abordar el problema de permetre dades de diverses simulacions simultàniament. En aquesta reunió es va acordar el mecanisme de filtratge.
  A la primera pantalla de la web, es mostra a l'usuari la llista de simulacions disponibles al sistema. L'usuari en tria una i en aquest moment es mostra el resum de la simulació seleccionada.

4 METODOLOGIA

Per al desenvolupament s'ha seguit una metodologia anomenada *scrum for one* [3]. Aquesta és una adaptació de *scrum* per a equips d'una sola persona. Manté els principis bàsics d'*scrum* i adapta la part de comunicació amb l'equip. Els punts destacables de *scrum for one* son els següents:

- **Divisió de la feina:** El desenvolupament es divideix en sprints i en tasques petites. S'arriba al resultat final de forma iterativa.

- **Revisió de l'sprint:** Al final de cada sprint es realitza una revisió de les tasques realitzades i les no realitzades. Aquesta revisió es guarda en un document i substitueix la reunió de tancament d'sprint de *scrum*.

- **Creació d'un diari de desenvolupament:** Aquest diari substitueix les reunions diàries *daily scrum*. En aquest s'apunta tot el progrés que es va realitzant.
  L'objectiu no és tenir una documentació molt extensa del treball realitzat cada dia si no una idea de com va la feina.
  Aquest diari s'actualitza durant el dia o al final d'aquest en comptes de a l'inici del següent dia com passa amb la *daily scrum*.

La duració dels sprints ha sigut de 15 dies i s'han realitzat un total de 8 sprints. Pel seguiment de les tasques s'ha utilitzat Trello[1] i pel control de versions s'ha utilitzat

[1] https://www.trello.com

Github[2].

La divisió de la feina en sprints ha sigut la següent:

1. **[09/03 - 20/03]** Formació en bases de dades no relacionals, Elastic stack, familiarització amb els orígens i formats de dades i disseny de la base de dades.
2. **[23/03 - 03/04]** Disseny i construcció de l'eina de subministrament de dades a la base de dades (*inserter*).
3. **[06/04 - 17/04]** Disseny de l'arquitectura del servidor (*application backend*), desenvolupar test del codi del servidor i implementació dels endpoints necessaris per assolir el primer objectiu (seguiment de mostres), implementació de test del codi desenvolupat.
4. **[20/04 - 01/05]** Implementació dels endpoints necessaris per assolir el segon objectiu (realitzar snapshots), decidir quina presentació es donarà a l'aplicació (web, aplicació escriptori...).
5. **[04/05 - 15/05]** Començar l'implementació del front end de l'aplicació (*web server*) i determinar si serà possible afegir objectius desitjables/augmentar l'abast del treball.
6. **[18/05 - 29/05]** Finalitzar l'objectiu 2 (realitzar snapshots), assolir una primera versió de l'objectiu 3 (mètriques de simulació) i implementar els nous requeriments acordats.
7. **[01/06 - 12/06]** Desenvolupar la versió final de la UI/frontend. Exploratory testing. Finalitzar l'objectiu 3 (mètriques de simulació) i totes les funcionalitats no acabades si en queden.
8. **[15/06 - 26/06]** Aconseguir una versió estable del software, realitzar desplegament de l'aplicació, finalitzar memòria i dossier.

5 Arquitectura de l'aplicació

S'ha dividit el projecte en tres parts. La primera part o *inserter* és el programa que s'encarrega de llegir els fitxers de logs d'una simulació, transformar les dades d'aquests i insertar-los a la base de dades.

La segona part o *application backend* és el servidor de dades. Aquest conté un conjunt d'endpoints que retornen les dades necessàries per assolir els objectius. Les dades que retorna les consulta de la base de dades. També fa transformacions en aquestes si és necessari.
Les respostes que proporciona aquesta part del projecte són en format JSON.

La tercera part o *web server* és l'aplcació amb la que treballarà l'usuari final. Aquesta té un format d'aplicació web. Només es comunica amb el servidor de dades.

A la figura 1 es pot observar un diagrama de l'arquitectura. S'ha adoptat aquest disseny per fer el projecte modular. Es podria canviar la presentació de l'aplicació fàcilment, per exemple a una aplicació d'escriptori. Per això només caldria programar les crides a l'*application backend* des de la nova aplicació. No caldria tractar ni modificar cap dada ja que aquestes ja les proporciona l'*application backend* en el format correcte.

En un futur també es podria canviar el motor de la base de dades i no s'hauria de modificar la presentació de l'aplicació. En aquest cas només s'hauria de modificar alguna part de l'*application backend*.

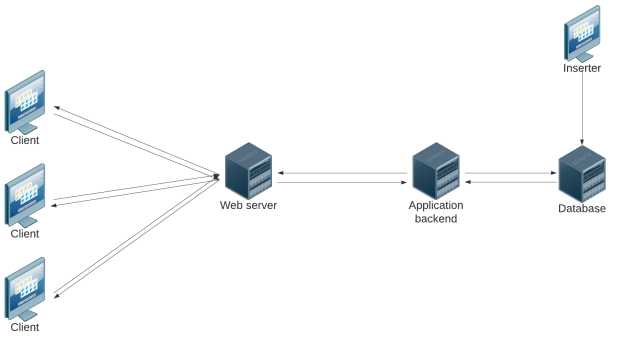


Fig. 1: Arquitectura del projecte.

6 Tecnologies utilitzades

A continuació es descriuran les tecnologies utilitzades per desenvolupar el projecte.

- **Base de dades:** Per a la base de dades s'ha escollit Elasticsearch. Aquesta és una part del projecte ELK [2].
  Elasticsearch està dissenyat per poder emmagatzemar grans quantitats de dades i poder-les filtrar fàcilment. Aquest motor de base de dades permet tenir les dades descentralitzades i replicades en diversos nodes. En aquest projecte s'utilitzarà un sol node.
- ***Inserter*:** Aquesta eina s'ha desenvolupat en python. S'han utilitzat algunes llibreries estàndards per dur-la a terme, sent *requests* [4] la més destacable. Aquesta llibreria ha sigut utilitzada per enviar les dades a insertar a Elasticsearch.
  Una altre llibreria molt utilitzada ha sigut *threading* [5]. Aquesta s'ha utilitzat per a paralelitzar la lectura de fitxers i l'inserció a la base de dades augmentant d'aquesta forma la velocitat del programa.
- ***Application backend*:** Aquesta eina també s'ha desenvolupat en python per les facilitats que ofereix a l'hora de manipular dades. L'eina té un format de servidor HTTP, per tant s'ha hagut d'utilitzar un framework per això. Hi ha diversos frameworks que ho permeten [6], en aquests cas s'ha triat Flask [7].
  Tots els endpoints serveixen dades en format JSON.
- ***Web server*:** Aquesta part del projecte també s'ha desenvolupat en python i l'ajuda de Flask [7].
  També es tracta d'un servidor HTTP. Aquest, a diferència de l'*application backend* serveix vistes, no dades. Les vistes s'han construït amb HTML, CSS i JavaScript.

[2]https://www.github.com

Les dades que mostren les vistes són les que serveix l'*application backend*. El *web server* crida als endpoints del *backend* per mostrar les dades demanades.

7 RESULTATS

El projecte ha sigut un èxit. Tots els objectius proposats s'han pogut complir, tant els originals com els proposats durant el desenvolupament.

Els requeriments secundaris també s'han complert majoritàriament, solament en un cas, el temps de resposta de l'aplicació és molt més elevat de l'esperat.

7.1 *Inserter*

Aquesta eina té un paper secundari dins del projecte. És menys utilitzada que les altres dos, ja que només es necessita per carregar una simulació al sistema.

Tot i trobar-se en un segon pla, és necessari que compleixi uns mínims de rendiment. En un inici, s'havia plantejat la possibilitat de visualitzar simulacions en temps real, és a dir, inserir cada log generat a la base de dades quan aquest es genera.
Per poder assolir aquest requeriment s'ha implementat una arquitectura d'alt rendiment amb l'ajuda de multithreading.

Les dades dels logs es troben en fitxers separats, normalment de 50MB. Quan s'inicia l'inserter, es llegeixen els els fitxers en paralel i es van enviant les dades transformades a la base de dades de mica en mica.
Aquesta arquitectura fa que l'usuari pugui configurar diversos paràmetres (figura 2). Els paràmetres es poden passar per línea de comandes o a través del fitxer de configuració.

```
PS D:\Documents\tfg\inserter> python.exe .\main.py --help
Usage: main.py [OPTIONS]

Options:
  -t, --threads INTEGER      Number of threads
  -e, --endpoint TEXT        Elasticsearch endpoint.
  -i, --index TEXT           Elasticsearch index.
  -s, --chunk-size INTEGER   Ammount of logs sent to elasticsearch in each
                             message

  -b, --backend TEXT         Application backend endpoint.
  --benchmark                Benchmark mode. Only for testing.
  -f, --files TEXT           Path to json file describing files to process.
  -c, --config-file TEXT     Path to config json file.
  --help                     Show this message and exit.
```

Fig. 2: Paràmetres permesos per a l'*inserter*

Alguns paràmetres configuren les dades que s'han de llegir o on es troba la base de dades i altres afecten directament al rendiment de l'eina. Aquests que afecten al rendiment són:

- **Threads:** Com indica el nom, defineix quants threads utilitzarà el programa per a llegir els arxius d'entrada.
  Un nombre massa petit de threads fa que el programa tingui un rendiment pobre. Certes operacions com l'enviament de dades a la base de dades requereixen esperar a una resposta, mentre s'espera una resposta altres threads poden realitzar altres operacions.
  Un nombre massa gran de threads fa que aquests competeixin pels recursos del PC. Això també es reflecteix amb un rendiment pobre.

- **Chunk size:** La base de dades espera missatges en format JSON per a inserir les dades desitjades. Aquests missatges poden contenir diverses entrades a indexar [8].
  Chunk size defineix la quantitat de logs que conté cada petició d'inserció a la base de dades. Com que aquestes es realitzen sobre HTTP, crides amb un sol log són molt poc eficients. Es poden enviar diversos logs amb el mateix cost d'establiment de connexió i enviament de dades.
  Si el paràmetre *chunk size* és molt gran, python ha de construir missatges molt grans i tractar amb strings molt llargs. En aquestes situacions, python té un rendiment molt pobre.
  Per tant, l'objectiu d'aquest paràmetre és trobar l'equilibri entre el cost de generar peticions HTTP i tractar strings de grans dimensions en python.

La millor combinació d'aquests paràmetres (entenent com a millor la que més logs per segon aconsegueix enviar a la base de dades) depèn del hardware on s'executi.
S'ha creat un script per realitzar *benchmarks* amb diferents combinacions i s'han obtinguts els resultats mostrats a la figura 3.

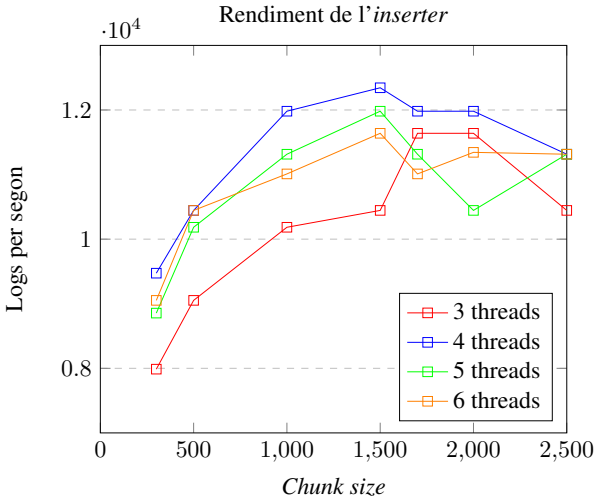


Fig. 3: Rendiment de l'*inserter*. L'eix x representa el valor del paràmetre *chunk size*. L'eix y mostra els logs per segon que s'inserten amb els paràmetres especificats (major és millor).

Sobre el hardware on s'ha executat (Intel Xeon E3-1505M v6[3] i 32 GB de ram), la millor combinació és **3 threads i *chunk size* de 1500**. Amb aquests paràmetres s'aconsegueix enviar $\approx$ 1300 logs per segon.
A la figura 3 es poden veure els efectes mencionats a la descripció de *chunk size*:

- Amb 6 threads mai s'aconsegueix un rendiment alt. És un nombre massa elevat pel CPU de l'equip.

- 3 threads no són suficients per aprofitar al màxim el processador. Es produeixen moltes estones d'espera de respostes.

[3] https://ark.intel.com/content/www/es/es/ark/products/97463/intel-xeon-processor-e3-1505m-v6-8m-cache-3-00-ghz.html

- Un *chunk size* menor a 1000 fa que s'aprofitin molt poc els missatges HTTP. L'esforç d'enviar i rebre resposta es podria aprofitar millor.

- Un *chunk size* major a 1700 fa que python perdi rendiment a l'hora de tractar amb strings. El temps de construcció del missatge a enviar a través d'HTTP és massa gran.

7.2 *Application backend*

L'*application backend* és la part del projecte que fa possible que els usuaris puguin visualitzat dades rellevants. Fa les consultes a Elasticsearch i transforma les dades obtingudes. És la part més important i més utilitzada del projecte.
Com ja s'ha comentat anteriorment, té un format de servidor web i s'ha implementat amb l'ajuda de Flask.

La llista d'endpoints disponibles d'aquest component és la següent:

- **/simulations:** Retorna una llista de noms i IDs de les simulacions carregades a la base de dades.

- **/simulation/[simulation_id]:** Retorna un conjunt d'informació vinculada a la simulació demanada (**simulation_id**). Aquesta informació és el nom de la simulació, data d'inici, data final i nombre de mostres analitzades.

- **/simulation/[simulation_name]/[simulation_id]:** Aquest endpoint s'utilitza per indexar una nova simulació. És molt més eficient guardar una llista de simulacions en un fitxer que no buscar totes les simulacions carregades a partir dels logs de la base de dades. A diferència dels alteres endpoints, no és un endpoint de consulta. No retorna dades.

- **/samples/[simulation_id]:** Retorna una llista de IDs de mostres agrupades pel camí que segueixen. Totes les mostres que segueixen un mateix camí passen més o menys per les mateixes màquines (hi poden haver petites variacions). El paràmetre **simulation_id** indica en quina simulació es busca.

- **/error-samples/[simulation_id]:** Retorna una llista de mostres que han acabat el seu trajecte amb errors. Les dades que s'inclouen a cada mostra són l'ID, el tipus d'error i el timestamp de quan s'ha detectat.

- **/sample/[sample_id]:** Retorna informació sobre la mostra consultada (**sample_id**). Aquesta informació és el temps que ha passat al laboratori, les màquines per les que ha passat i el temps que ha estat en cada una d'elles, el temps que ha estat viatjant entre màquines i una llista completa de logs que referencien a aquesta mostra.

- **/snapshot/[simulation_id]/[timestamp]:** Retorna una llista de les màquines de la simulació (**simulation_id**) amb el nombre de mostres que hi ha en elles en el moment que es tria (**snapshot**).

- **/timeline/[simulation_id]/[gap]:** Retorna les dades necessàries per construir una gràfica que mostra l'evolució del número de mostres que es troben a cada màquina del laboratori. El paràmetre **gap** indica quantes hores separen els punts de dades, per exemple amb gap=1 es mostrarà el nombre de samples de cada màquina al minut 0, 60, 120... i amb gap=0.5 es mostraran les dades de les màquines als minuts 0, 30, 60, 90, 120...

- **/log-count/[simulation_id]/[gap]:** Retorna les dades necessàries per construir una gràfica que mostra l'evolució dels missatges intercanviats entre les màquines del laboratori. El paràmetre **gap** indica quantes hores separen els punts de dades, per exemple amb gap=1 es mostrarà el nombre de missatges de cada tipus al minut 0, 60, 120... i amb gap=0.5 es mostraran les dades dels missatges als minuts 0, 30, 60, 90, 120...

- **/dev/reset:** Aquest endpoint s'ha utilitzat durant el desenvolupament i s'ha mantingut per l'utilitat que aporta. Aquest esborra totes les dades de la base de dades i regenera l'index on es guardaven. No es pot consultar des del *web server*, l'única forma d'accedir-hi és fent la petició directament des d'un navegador o eina que permet enviar peticions HTTP (curl, postman...).

Alguns endpoints retornen dades molt similars i es podrien ajuntar en un de sol. D'aquesta forma s'aconseguiria reduir el nombre de consultes que es fan a l'*application backend*. Aquesta acció no s'ha dut a terme ja que es reduiria el nombre de consultes però s'incrementaria el temps de resposta. Els endpoints *samples* i *error-samples* fan consultes diferents a Elasticsearch, per tant, si s'ajuntessin, el nou endpoint hauria de fer totes els consultes de cop.

En aquest component, el rendiment mínim acceptable es mesura en segons de resposta dels endpoints.
L'aplicació utilitza moltes crides AJAX [9], per aquesta raó, el temps de resposta pot ser una mica més alt sense que afecti a l'experiència d'usuari.

Com es pot observar a la figura 4 la majoria d'endpoints responen en menys de 5 segons. Els tres endpoints que no compleixen aquesta regla són els que retornen dades per mostrar en gràfiques. Aquests disposen d'un mecanisme de cache, que permet estalviar una gran quantitat de temps al resoldre peticions que ja han estat resoltes.

A la figura 4 es mostra el temps de resposta de la primera crida, en cas d'una crida ja resposta anteriorment, els endpoints *snapshot*, *timeline* i *log count* triguen aproximadament **3 segons** en retornar l'informació demanada.

Cal tenir en compte que el servidor s'ha executat en un ordinador amb hardware poc potent (Intel Core i3-6100[4] i 8 GB de ram). En un servidor amb hardware més avançat s'aconsseguirien temps de resposta molt més competents.

[4]https://ark.intel.com/content/www/es/es/ark/products/90729/intel-core-i3-6100-processor-3m-cache-3-70-ghz.html

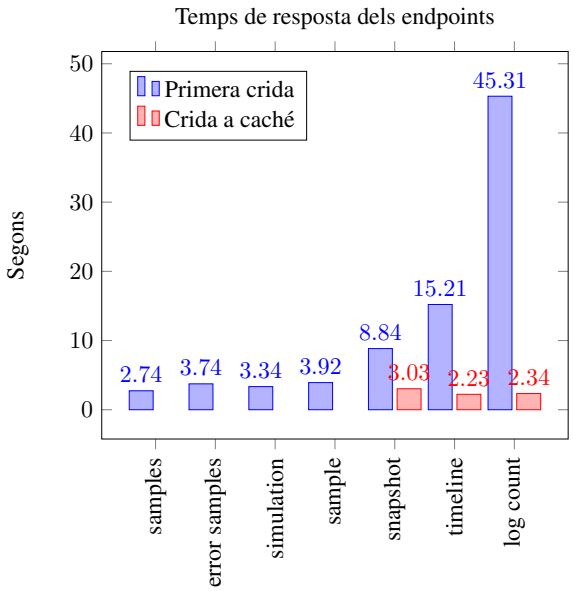


Fig. 4: Temps de resposta dels endpoints en segons. Menor és millor.

7.3 *Web server*

El *web server* és l'última part del projecte. Es tracta del frontend de l'aplicació web. Consulta els endpoints de l'*application backend* i mostra les dades dins les vistes de la pàgina web.

Les consultes a l'*application backend* es realitzen de forma asíncrona amb AJAX [9], de forma que el temps de càrrega de la web és inferior i l'experiència d'usuari és més fluida.

Aquesta part del projecte és la que menys complexitat té, senzillament s'encarrega de fer crides als endpoints de l'*application backend* i no realitza cap transformació a les dades rebudes.
El punt que més feina ha portat al desenvolupar aquesta part ha sigut el codi JavaScript, ja que s'han implementat una gran quantitat de funcions per contribuir a millorar l'experiència d'usuari.
Amb l'ajuda de Bootstrap [10] i jQuery s'ha creat una web moderna que segueix els estàndards de disseny actuals.

La pàgina web es divideix en diferents zones funcionals (figura 5). A la zona superior l'usuari pot canviar ràpidament de simulació o realitzar una busca de mostres per ID si ho desitja. A la zona lateral esquerra hi troba una llista reduïda de mostres. A la zona central és on hi troba més informació. Aquí és on es mostraran les dades que desitgi visualitzar, ja sigui un resum d'una simulació o les dades d'una mostra.

Com s'ha comentat anteriorment, s'han satisfet les necessitats proposades a l'inici del projecte. Les solucions ofertes han sigut les següents:

- **Realitzar seguiment de mostres:** A la figura 11 es pot veure la pantalla *sample*. En aquesta hi trobem una mica d'informació sobre la mostra seleccionada i una llista de trajectes que ha seguit. També es mostren aquests trajectes en una línia del temps on cada trajecte representa de forma proporcional el temps que ha passat la mostra al laboratori.
- **Capturar instantànies del sistema:** Des de la pantalla *simulació* (figura 6) es poden apreciar les diverses funcions que aquesta ofereix. La figura 7 mostra el resultat d'utilitzar la funció *snapshot*. En aquest moment es pot apreciar com les màquines P702-1 i P702-2 són les que més mostres contenen.
- **Visualitzar mètriques de la simulació:** Les mètriques de la simulació es poden obtenir des de la pantalla *simulació* també. Són les funcions *timeline* i *log count* que es veuen a la figura 6. Aquestes generen les gràfiques mostrades a les figures 9 i 10.

7.4 Resultats finals

El projecte s'ha encapsulat utilitzant contenidors Docker [11] per fer-lo fàcil de desplegar, mantenir i moure de màquina si és necessari.
Per crear els contenidors s'ha utilitzat **docker-compose**. Dins del fitxer de configuració s'han creat 3 serveis:

- Elasticsearch
- *Application backend*
- *Web server*

Com es pot observar, l'*inserter* no s'ha encapsulat utilitzant docker ja que no tindria sentit. Aquest s'executa cada vegada que es necessita, però no ofereix un servei com les altres dues parts del projecte.

El projecte satisfà les necessitats i és escalable. Permet l'incorporació de noves funcionalitats si així es requereix. L'empresa també està satisfeta amb el resultat i es té previst utilitzar-lo pròximament, per tant ha sigut un projecte beneficiós tant per l'estudiant com per la corporació.

8 Control de qualitat i test

Tot el codi del projecte s'ha sotmès a revisions i proves de qualitat. Al desenvolupar-se en una empresa, la qualitat de codi requerida és encara més alta.
La guia d'estil seguida ha sigut PEP8 [12], aquesta és un estàndard dins la comunitat de Python. Per realitzar la revisió automàtica del codi s'ha utilitzat el paquet *pylint*.

A part de l'adopció d'una guia d'estil i documentació del codi també s'ha realitzat un procés de testing del codi. Aquest s'ha format per unit testing i exploratory testing.

L'*inserter* s'ha sotmès principalment a unit testing. Totes les funcions de totes les classes s'han provat amb diversos casos de test. Per a fer-ho s'ha necessitat *mockejar* algunes classes o funcions de python. En total s'han programat 38 tests per assegurar que el funcionament d'aquesta part del projecte és correcte.
Cal destacar que l'*inserter* també s'ha provat de forma extensiva quan s'ha fet el *benchmarking* d'aquests. Per

trobar la millor configuració de paràmetres s'han realitzat proves que equivaldrien a exploratory testing.

L'*application backend* s'ha provat tant amb unit testing com amb exploratory tesing.
L'unit testing s'ha realitzat amb les llibreries *unittest* i *mock*. S'han cobert totes les funcions de totes les classes de forma que s'ha assolit un **statement coverage**.
La part d'exploratory testing s'ha fet amb un navegador web i amb postman. Aquest segon software facilita molt el procés de fer peticions a URLs i adjuntar dades a aquestes peticions. Amb aquesta part d'exploratory s'ha assegurat un correcte funcionament en casos reals.

Finalment, el *web server* s'ha provat solament amb exploratory testing. Es va plantejar implementar unit testing en aquesta part del projecte també però es va descartar la idea ja que totes les funcions del *web server* són extremadament curtes i simples. Solament es fan crides a endpoints (utilitzant la llibreria *requests*) i es retorna el resultat en format html.

9 PRÒXIMS PASSOS

Tot i haver completat tots els objectius plantejats, han sorgit altres objectius o petites funcionalitats que es podrien implementar en un futur. Aquesta feina la podria realitzar jo en els pròxims mesos per seguir el desenvolupament de l'eina. Les noves implementacions o millores plantejades són les següents:

- **Detecció d'errors potencials:** Actualment a l'aplicació es poden veure aquelles mostres que han acabat el seu trajecte amb algun error. També es poden veure els nivells d'ús de cada màquina del laboratori de la simulació.
  Faltaria un mòdul que mostri possibles errors. Si s'acumula un gran nombre de mostres en una màquina pot ser que aquesta hagi deixat de funcionar i no processi mostres, per tant està fent que el simulador funcioni incorrectament. Això facilitaria en gran part la feina de detectar si la simulació ha acabat correctament o no.

- **Llistar excepcions trobades:** Els logs que genera el simulador poden tenir un camp d'excepció generada. En casos que el simulador s'aturi de cop pot ser degut a que alguna màquina ha generat una excepció i ha deixat de funcionar.
  Aquestes excepcions es guarden a la base de dades però no es tracten. La nova funcionalitat seria afegir a la pantalla *simulacions* (figura 6) una llista d'excepcions trobades als logs. El format podria ser similar al de la llista de mostres amb errors (figura 8).

- **Millor integració amb el simulador:** Amb el model actual, quan una simulació acaba, deixa els arxius de logs en un directori específic del sistema. Si es volen utilitzar aquests logs per carregar-los amb l'*inserter*, s'ha d'entrar manualment a la màquina virtual, comprimir els fitxers i enviar-los a una màquina externa. Després descomprimir-los i especificar a l'*inserter* quins fitxers ha de llegir.
  La proposta d'aquest objectiu és integrar l'*inserter* dins la màquina virtual de les simulacions, de forma que quan el simulador acabi la seva execució, automàticament s'executi l'*inserter* i es puguin visualitzar els resultats a través de l'aplicació web. Aquesta millora representaria un gran canvi a l'hora de recollir les dades generades per les simulacions. Si es seguís per aquest camí, es podria automatitzar totalment el llançament de simulacions.

- **Millorar la navegació de l'aplicació web:** Les dades de l'aplicació web i la navegació entre pantalles es realitza utilitzant AJAX. Això redueix els temps de càrrega però també fa que la URL mai canvii. Totes les pàgines de la web es visualitzen utilitzant la mateixa URL.

  Això crea un problema a l'hora de compartir resultats amb altres persones. Si es vol investigar una mica sobre una simulació o una mostra, no es pot compartir un link que mostri les dades desitjades.
  Aquest problema es pot arreglar amb codi JavaScript i editant el *router* de l'aplicació web. Comportaria bastanta feina però el resultat seria notori.

- **Millorar les gràfiques de nombre de missatges:** Actualment, la gràfica de nombre de missatges transmesos (figura 10) aporta informació útil. Tot i així, encara podria aportar més informació si es permetés filtrar per camps dels missatges.
  Un clar exemple és quan es vol saber si una màquina del laboratori segueix funcionant correctament. Per descobrir-ho es pot mirar si aquesta enviar missatges del tipus *SampleContainerLeft* (els envia quan acaba de processar una mostra). Si es veu que en un cert moment ha deixat d'enviar missatges amb aquestes característiques és una prova inequívoca de que la màquina ha deixat de funcionar.

10 CONCLUSIONS

L'objectiu principal d'aquest projecte era facilitar la feina al personal encarregat de supervisar les simulacions dels laboratoris. Els resultats obtinguts han sigut els esperats. El projecte s'ha adaptat a les necessitats de l'empresa sense abandonar les idees principals plantejades a l'inici.
Gràcies a la metodologia *scrum for one* s'ha pogut fer un bon seguiment del progrés. Aquesta metodologia ha permès fer estimacions de temps bastant acurades.
L'error que s'ha provocat amb les prediccions ha permès incorporar noves funcionalitats demanades per l'empresa.

Dins l'empresa, aquest és un producte experimental. Tot i no formar part d'un altre projecte més gran, també es requereixen els estàndards de qualitat. Aquests estàndards han suposat un repte extra a l'hora de dur a terme el desenvolupament. S'han assolit amb la documentació del codi i unit testing.

L'arquitectura del projecte s'ha pensat de forma modular per a que sigui fàcil de modificar i escalar. S'han aplicat patrons de disseny que permeten la reutilització

de codi per poder així facilitar el desenvolupament i simplificar la complexitat del programa. Aquests patrons també fan que sigui més difícil l'existència de bugs poc comuns i facilita la fase de *testing*.

Personalment trobo que ha sigut molt reconfortant desenvolupar un projecte des de zero passant per totes les etapes. Cada pas ha sigut un repte, des del plantejament, passant pel disseny i l'implementació fins al tancament. He pogut consolidar la meva base de python i aprendre a utilitzar una gran quantitat de llibreries noves.
També voldria destacar que em motiva que aquest projecte no sigui simplement un treball de final de carrera, si no que també pot tenir continuïtat.

AGRAÏMENTS

Voldria dedicar aquest projecte a la meva família, que m'ha donat suport durant tota la carrera. També als meus companys que m'han ajudat a solucionar dubtes i han compartit coneixements.
També a aquelles persones de Roche que han supervisat la meva feina i m'han indicat quins eren els punts a reforçar.
Finalment a Rafael Fernández per l'ajuda que m'ha proporcionat a l'hora d'encaminar els informes d'aquest projecte.

REFERÈNCIES

[1] F. Hoffmann-La Roche Ltd. *Roche Diagnostics San Cugat*. https://www.roche.es/es_es/sobre_roche/nuestros-centros/roche-en-san-cugat.html. [Online] Accedit el 25 de febrer de 2020. 2018.

[2] Elasticsearch B.V. *What is the ELK Stack?* https://www.elastic.co/what-is/elk-stack. [Online] Accedit el 21 de febrer de 2020. 2018.

[3] Alex Andrew. *Scrum Of One: How to Bring Scrum into your One-Person Operation*. https://www.raywenderlich.com/585-scrum-of-one-how-to-bring-scrum-into-your-one-person-operation. [Online] Accedit el 27 de febrer de 2020. 2017.

[4] Kenneth Reith. *Requests: HTTP for Humans*. https://requests.readthedocs.io/en/master/. [Online] Accedit el 3 de març de 2020. 2020.

[5] Python Software Foundation. *Python threading documentation*. https://docs.python.org/3/library/threading.html. [Online] Accedit el 5 de març de 2020. 2020.

[6] Gareth Dwyer. *Flask vs Django: Why Flask Might Be Better*. https://www.codementor.io/@garethdwyer/flask-vs-django-why-flask-might-be-better-4xs7mdf8v. [Online] Accedit el 7 de març de 2020. 2017.

[7] Pallets. *Flask Documentation*. https://flask.palletsprojects.com/en/1.1.x/. [Online] Accedit el 7 de març de 2020. 2020.

[8] Elasticsearch B.V. *Elasticsearch Reference*. https://www.elastic.co/guide/en/elasticsearch/reference/current/docs-bulk.html#docs-bulk. [Online] Accedit el 18 de març de 2020. 2020.

[9] The jQuery Foundation. *jQuery Ajax documentation*. https://api.jquery.com/jquery.ajax/. [Online] Accedit el 21 de maig de 2020. 2020.

[10] Bootstrap team. *Bootstrap documentation*. https://getbootstrap.com/docs/4.1/getting-started/introduction/. [Online] Accedit el 28 d'abril de 2020. 2020.

[11] Docker. *Why Docker?* https://www.docker.com/why-docker. [Online] Accedit el 22 de maig de 2020. 2018.

[12] Python Software Foundation. *PEP 8 – Style Guide for Python Code*. https://www.python.org/dev/peps/pep-0008/. [Online] Accedit el 25 de març de 2020. 2013.

APÈNDIX

A.1 Imatges de l'aplicació web

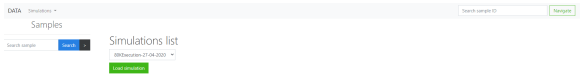


Fig. 5: Estructura de la pàgina web

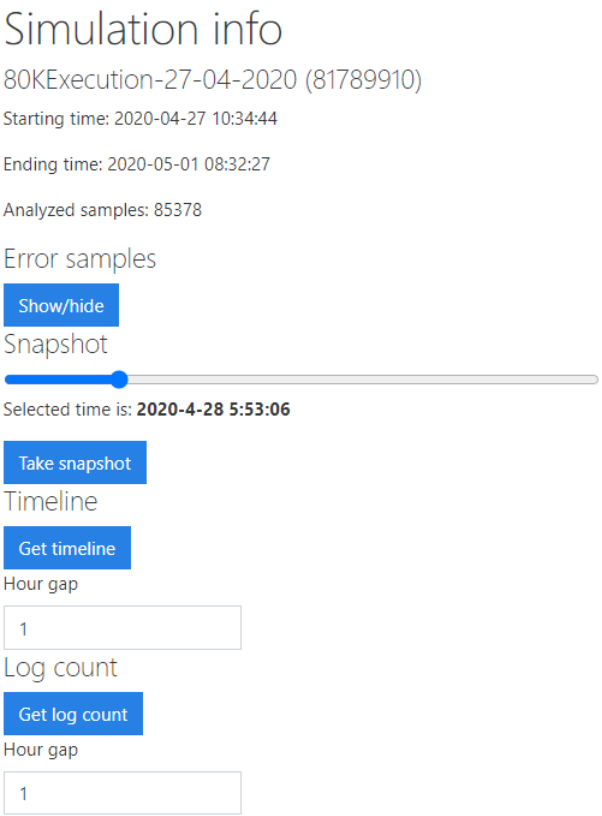


Fig. 6: Pantalla *simulacions*

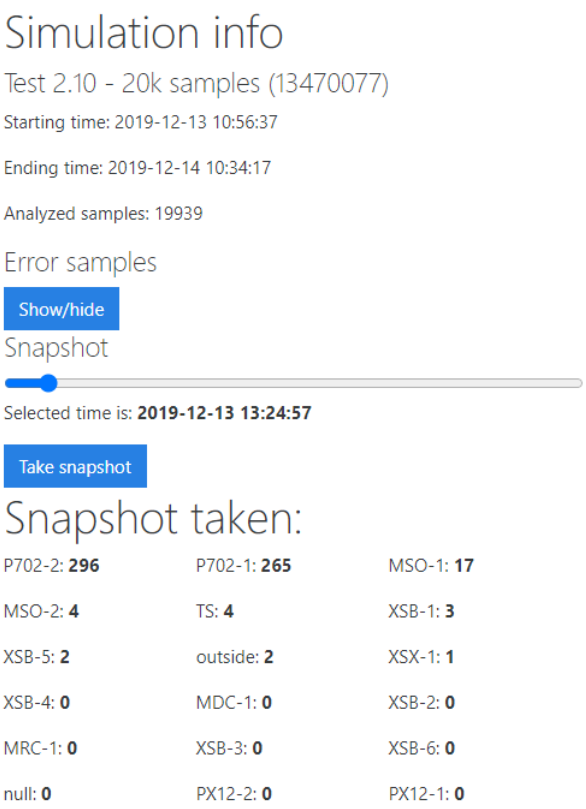
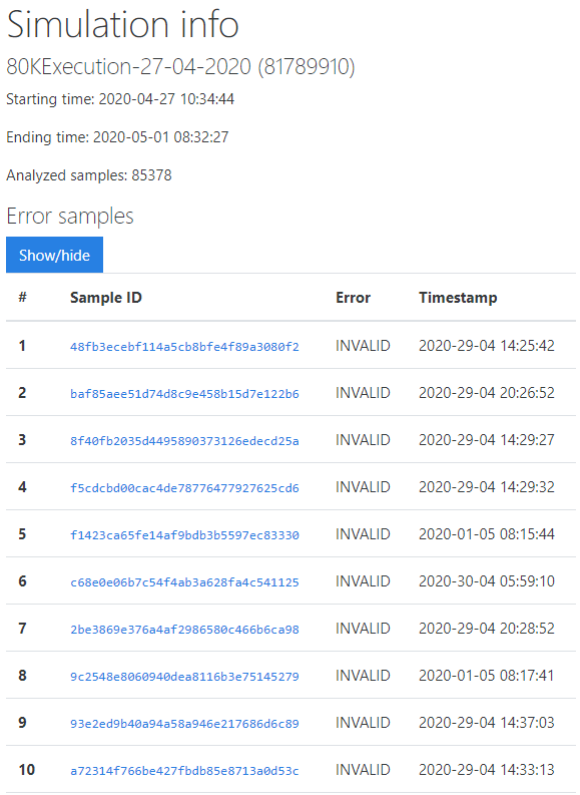


Fig. 7: Resultat de fer una *snapshot*



#	Sample ID	Error	Timestamp
1	48fb3ecebf114a5cb8bfe4f89a3080f2	INVALID	2020-29-04 14:25:42
2	baf85aee51d74d8c9e458b15d7e122b6	INVALID	2020-29-04 20:26:52
3	8f40fb2035d4495890373126edecd25a	INVALID	2020-29-04 14:29:27
4	f5cdcbd00cac4de78776477927625cd6	INVALID	2020-29-04 14:29:32
5	f1423ca65fe14af9bdb3b5597ec83330	INVALID	2020-01-05 08:15:44
6	c68e0e06b7c54f4ab3a628fa4c541125	INVALID	2020-30-04 05:59:10
7	2be3869e376a4af2986580c466b6ca98	INVALID	2020-29-04 20:28:52
8	9c2548e8060940dea8116b3e75145279	INVALID	2020-01-05 08:17:41
9	93e2ed9b40a94a58a946e217686d6c89	INVALID	2020-29-04 14:37:03
10	a72314f766be427fbdb85e8713a0d53c	INVALID	2020-29-04 14:33:13

Fig. 8: Llista de mostres amb errors

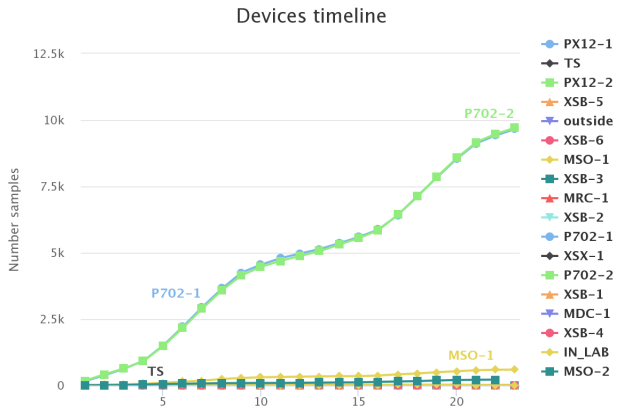


Fig. 9: Gràfica d'ús de les diferents màquines del laboratori

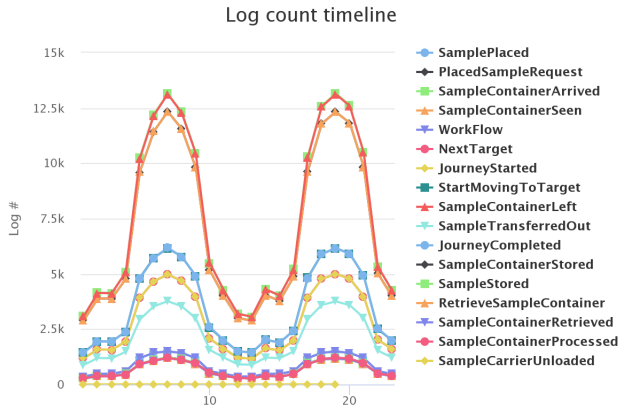


Fig. 10: Gràfica de nombre de missatges enviats

Fig. 11: Pantalla *sample*, on es veuen les trajectes realitzats per una mostra.