Programul Operațional Competitivitate

CeCBiD-EOSC

Centru Cloud și Big Data pentru participarea la Cloud-ul European pentru Știință Deschisă (CeCBiD-EOSC)

Raport Tehnic 2.2

Federalizarea resurselor oferite de către CLOUDIFIN și alte centre Cloud, și dezvoltarea unei interfețe web pentru accesul utilizatorilor la resurse

Autori: Mihnea Dulea
Dragoș Ciobanu-Zabet
Bianca Neagu
Robert Poenaru
Ionuț Vasile

Versiunea: Finală (1.0)
Data: 31.07.2023
Distributie: Internă
Cod document: CBD.RT-2.2

Rezumat: Proiectul CeCBiD-EOSC a fost implementat în cadrul Departamentului Fizică Computațională și Tehnologia Informației din IFIN-HH între anii 2020-2023. Acest document raportează rezultatele obținute în cadrul Subactivității 2.2 – *"Federalizarea resurselor oferite de către CLOUDIFIN și alte centre Cloud, și dezvoltarea unei interfețe web pentru accesul utilizatorilor la aceste resurse"*, desfășurată în perioada 19.05.2021 – 31.07.2023. Sunt prezentate proiectarea, dezvoltarea si implementarea Sistemului Cloud Federalizat pentru cercetare (SCF) si a interfetelor de acces a utilizatorilor, de administrare, de monitorizare si de inventariere a resurselor SCF. Versiunea pilot a sistemului a fost testata prin interconectarea centrului CLOUDIFIN din cadrul Centrului de resurse Cloud si Big Data cu centrele Cloud ale INCD pentru Tehnologii Izotopice și Moleculare din Cluj-Napoca si Universitatii 'Ovidius' din Constanta.

Lista reviziilor

Data	Versiunea	Editor/Autor/Coautori	Sumarul modificărilor/completărilor principale
02.05.2023	0.1	M. Dulea	Structura raport, versiune preliminară Cuprins
09.05.2023	0.2	M. Dulea	Prefață; Cap. 1. Introducere;
15.05.2023	0.3	M.Dulea	Cap. 2 - Cerinte
06.06.2023	0.3	M. Dulea, D. Ciobanu-Zabet, I. Vasile	3. Specificatii tehnice; 4. Proiectarea platformei pilot
15.06.2023	0.4	R. Poenaru	8. Serviciu de inventariere si administrare …
03.07.2023	0.5	D. Ciobanu-Zabet, B. Neagu	5. Biroul virtual al utilizatorului
17.07.2023	0.6	D. Ciobanu-Zabet, I. Vasile	7. Implementare; 6. Monitorizarea centralizata
21.07.2023	0.7	D. Ciobanu-Zabet	9. Agenti; Anexa I
25.07.2023	0.8	B. Neagu	Anexa II – Manualul utilizatorului
28.07.2023	0 9	M. Dulea	Rezumat; Concluzii
31.07.2023	1.0	M. Dulea	Editare versiune finala

Prefață

Investițiile în infrastructura de Cloud computing și de Big Data în vederea maximizării potențialului de creștere a economiei digitale europene reprezintă una dintre direcțiile prioritare ale Strategiei Pieței Unice Digitale, care a fost stabilită în 2015 de către Comisia Europeană.[1]

Recunoscând capabilitațile de exploatare a fenomenului Big Data oferite de tehnologia Cloud, Comisia a lansat în aprilie 2016 Inițiativa Europeană Cloud[2], a carei implementare se bazeaza pe Cloud-ul European pentru Stiinta Deschisa (*European Open Science Cloud* – EOSC) si pe Infrastructura Europeana de Date (*European Data Infrastructure* – EDI).

In viziunea Comisiei Europene, EOSC trebuie să asigure pentru comunitatea științifică un mediu virtual sigur, deschis, capabil să ofere servicii de stocare, management, analiză, precum și de refolosire a datelor dincolo de frontiere și discipline stiintifice. În acest cadru, s-a recomandat infrastructurilor europene de cercetare (și în primul rând infrastructurilor ESFRI) sa promoveze reutilizarea datelor proprii pentru inovare și în scopuri educaționale prin sprijinirea conectării lor la EOSC.[3]

Parcursul de Implementare a EOSC[4] a stabilit liniile de acțiune pentru crearea unei federații pan-europene a infrastructurilor de date pentru cercetare, care să înlocuiască fragmentarea existentă cu soluții eficiente și ușor de utilizat pentru stocarea, găsirea, partajarea și refolosirea datelor. Direcțiile de acțiune propuse pentru implementarea modelului federalizat al EOSC privesc arhitectura sistemului, administrarea datelor, serviciile, accesul și interfețele de acces, regulile de participare, precum și guvernanța. Arhitectura EOSC cuprinde un nucleu federativ, care include resursele partajate ale EOSC, precum și multiple infrastructuri de date federate angajate în furnizarea de servicii către EOSC.

Începand din anul 2018, IFIN-HH a contribuit la implementarea infrastructurii EOSC prin intermediul Departamentului Fizică Computațională și Tehnologia Informației (DFCTI, https://cc.ifin.ro), care a participat, în calitate de asociat al coordonatorului, Fundația EGI[5], la proiectul H2020 EOSC-Hub[6] (2018-2020), destinat dezvoltării resurselor și serviciilor Cloud inițiale pentru EOSC. Sarcina DFCTI a fost de a furniza, prin intermediul centrului Cloud CLOUDIFIN[7], resurse pentru susținerea diferitelor comunități de utilizatori, precum și de a contribui cu servicii naționale la catalogul de servicii al EOSC, în conformitate cu regulile de angajare ale proiectului. În acest scop, EOSC-Hub a finanțat activitatea de management și operare a site-ului CLOUDIFIN, asigurând continuitatea furnizării acestor servicii.

Pentru a putea finanța realizarea masei critice de resurse necesară participării la EOSC, DFCTI a propus proiectul CeCBiD-EOSC in cadrul apelului POC 398/2018. Totodată, pentru continuarea implementării serviciilor specifice EOSC, DFCTI participă, incepînd din 2020, la proiectul H2020 EGI-ACE – *„Advanced Computing for EOSC"* (2020-2023), a cărui misiune este de a asigura servicii EOSC gratuite pentru cercetătorii din toate disciplinele științifice care necesită calcule intensive și Big Data. Astfel, proiectele EGI-ACE și CeCBiD-EOSC acționeaza complementar, la nivelul EU și, respectiv, național, pentru realizarea strategiei de integrare a infrastructurii de calcul și de date a IFIN-HH în EOSC.

Obiectivul general al proiectului CeCBiD-EOSC este creșterea capacității de cercetare în scopul ridicării nivelului de competitivitate științifică pe plan intern și internațional al IFIN-HH, prin modernizarea infrastructurii Cloud, extinderea infrastructurii masive de date și realizarea unui centru de date cu performanțe înalte, care să fie integrat în infrastructura Cloud Europeană

[1] "*A Digital Single Market Strategy for Europe*" - COM(2015) 192
[2] "*European Cloud Initiative – Building a competitive data and knowledge economy in Europe*" – COM (2016) 178
[3] "*Long-term sustainability of Research Infrastructures*" – SWD(2017) 323
[4] "*Implementation Roadmap for the European Open Science Cloud*" - SWD(2018) 83
[5] Fundatia EGI, https://www.egi.eu/about/egi-foundation/
[6] "*Integrating and managing services for the EOSC"*, https://www.eosc-hub.eu/
[7] Centrul de resurse Cloud al DFCTI, CLOUDIFIN, http://cloudifin.ifin.ro/

pentru Știința Deschisă.[8]

Totodata, proiectul propune o soluție tehnică pentru interconectarea la nivel național, în cadrul unui Cloud federalizat, a centrelor de tip Cloud privat dezvoltate în instituții aparținând sistemului de CDI, capabilă să ofere utilizatorilor acces printr-o interfață unică la resurse și servicii furnizate de aceste centre. Implementarea acestei soluții va eficientiza utilizarea resurselor Cloud de către grupurile de cercetători și va stimula semnificativ cooperarea între specialiștii în tehnologii informatice avansate.

Infrastructura realizată în cadrul proiectului va susține dezvoltarea unor activități de CDI în domeniile sistemelor de calcul paralel și distribuit, învațării automatizate, calculului științific și bioinformaticii, cu aplicații relevante pentru fizica materiei condensate, studiul interacției laser-materie, nanofizică și nanoelectronică.

Obiectivele specifice ale proiectului CeCBiD-EOSC au fost următoarele:

1. Realizarea unui centru performant de resurse Cloud și Big Data prin achiziționarea și instalarea de active corporale și necorporale necesare pentru derularea activităților de CDI prevăzute în proiect.

2. Dezvoltarea și diversificarea serviciilor furnizate la nivel european de catre centrul CLOUDIFIN în perspectiva integrării acestuia în EOSC.

3. Realizarea în cadrul centrului de resurse Cloud a unei soluții tehnice capabilă să interconecteze la nivel național, în cadrul unui sistem federalizat, centrele de tip Cloud privat dezvoltate în instituții aparținând sistemului de CDI, capabilă să ofere utilizatorilor acces printr-o interfață unica la resurse și servicii furnizate de către aceste centre.

4. Asigurarea condițiilor de susținere informațională în tehnologie Cloud și Big Data a participării institutului la colaborări internaționale de anvergură, precum și a noilor opțiuni strategice privind angajarea în direcții de cercetare emergente din spațiul științific internațional, cu relevanță socio-economică deosebită.

5. Dezvoltarea și implementarea de servicii și aplicații informatice pentru administrarea și funcționarea centrului de resurse, care utilizeaza tehnologii Cloud și Big Data pentru: satisfacerea cerințelor IT din faza operațională a proiectului Extreme Light Infrastructure - Nuclear Physics (ELI-NP); modelarea și simularea la nivel molecular a nano- și biosistemelor complexe; analiza datelor de secvențiere de nouă generație.

6. Realizarea condițiilor tehnice și asigurarea suportului de specialitate pentru obținerea și/sau îmbunătățirea de către parteneri economici, în special din cadrul Clusterul Tehnologic Magurele (MHTC), a unor produse și servicii în domenii de specializare inteligentă, care vor conduce la creșterea competitivității acestora.

7. Formarea și perfecționarea personalului calificat, precum și transferul de cunoștințe în domeniile Cloud computing și Big Data către personalul științific și tehnic din alte entitati ale sistemului de CDI.

8. Diseminarea rezultatelor proiectului prin participarea la conferințe (inter)naționale și publicarea de articole științifice în parteneriat public-privat.

Prin obiectivele sale, proiectul conduce la extinderea capacității resurselor Cloud și Big Data, precum și la îmbunătățirea calitativă și diversificarea serviciilor de calcul și de analiza de date pe care Centrul de Calcul Avansat din IFIN-HH le va oferi comunității științifice naționale și internaționale, contribuind prin aceasta la dezvoltarea sistemului național de CDI și la creșterea vizibilității la nivel european.

Rezultatele prevăzute ale proiectului sunt prezentate mai jos.

[8] Cerere de Finantare, proiect CeCBiD-EOSC

Nr.	Rezultat prognozat	Documentul în care a fost raportat
1.	Documentatia de achizitie a serviciilor de consultanta pentru elaborarea documentatiei tehnice necesare echiparii centrului de resurse Cloud si Big Data	RP2
2.	Contract de furnizare a serviciilor de consultanta pentru elaborarea documentatiei tehnice necesare echiparii centrului de resurse Cloud si Big Data	RP2
3.	Documentatie tehnica privind echiparea centrului de resurse Cloud si Big Data	RP3
4.	Documentatia de achizitie a serviciilor de informare si publicitate	RP1
5.	Contract de achizitie servicii de informare si publicitate	RP1
6.	Contracte de achizitie active corporale pentru echiparea centrului de resurse Cloud si Big Data	RP11
7.	Un comunicat de presa publicat la lansarea proiectului	RP1
8.	Un comunicat de presa publicat la finalizarea proiectului	RP13
9.	Pagina web a proiectului, publicata la adresa http://cecbid-eosc.nipne.ro	RP1
10.	Materiale de informare si publicitate (roll-up-uri, afise, rame afis, mape, banner).	RP1
11.	Sistem de procesare de date achizitionat si instalat.	RP13
12.	Sistem de stocare de date achizitionat si instalat.	RP13
13.	Switch pentru interconectarea sistemelor hardware achizitionat si instalat.	RP13
14.	Instalatie de climatizare achizitionata si instalata.	RP13
15.	Sistem UPS achizitionat si instalat.	RP13
16.	Rack-uri pentru gzduirea echipamentelor IT achizitionate si instalate.	RP13
17.	Tablouri electrice si retea electrica achizitionate si instalate.	RP13
18.	Servicii si aplicatii informatice pentru administrarea si monitorizarea centrului CLOUDIFIN, precum si pentru asigurarea accesului utilizatorilor.	RP13
19.	Interfata de acces al utilizatorilor la resurse oferite de centre Cloud multiple. Manual de utilizare.	RP13
20.	Servicii si aplicatii informatice pentru satisfacerea cerintelor IT din faza operationala initiala a proiectului ELI-NP. Raport stiintific si tehnic	RP13
21.	Servicii si aplicatii informatice pentru suportul activitatii de modelare si simulare a nanostructurilor complexe	RP13
22.	Servicii si aplicatii informatice pentru suportul activitatii de analiza a datelor de secventiere de noua generatie	RP13

23.	Studiu privind performantele centrului Cloud si Big Data. Manual de management	RP13
24.	Lucrări și comunicări științifice	RP13
25.	Fișe de post	RP1-13
26.	Documente de raportare (rapoarte de activitate / de progres; procese verbale ale intalnirilor echipei de management a proiectului)	RP1-13
27.	Cereri de plată/rambursare	RP4-13
28.	Raport de audit final al proiectului.	-

Proiectul CeCBiD-EOSC a demarat la data semnării contractului de finațare de catre Ministerul Educației și Cercetării (19.05.2020) si a fost finalizat in luna iulie 2023.

Proiectul CeCBiD-EOSC este cofinanțat din Fondul European de Dezvoltare Regională (FEDR) în baza contractului de finanțare încheiat cu Ministerul Educației și Cercetării în calitate de Organism Intermediar, în numele și pentru Ministerul Fondurilor Europene în calitate de Autoritate de Management.

Cuprins

1. Introducere 11

1.1 CONTEXT GENERAL ȘI NECESITATE 11
1.2 OBIECTIVELE SUBACTIVITĂȚII 2.2 11
1.3 REZULTATE PRECONIZATE 12
1.4 ETAPELE SUBACTIVITATII 12

2. Cerinte 13

2.1 CERINTELE UTILIZATORILOR 13
2.2 CONSTRANGERI LEGALE SI ORGANIZATIONALE 14
2.3 CERINTE DE SISTEM 14

3. Specificatiile tehnice ale Sistemului Cloud Federalizat 16

3.1 CONDITII DE INTEGRARE A CENTRELOR CLOUD IN SCF 16
3.2 SPECIFICATII DE SISTEM 16
3.3 POLITICI SI PROCEDURI 17

4. Proiectarea platformei pilot 18

4.1 ARHITECTURA SISTEMULUI 18
4.2 PROIECTAREA INTERFETEI GRAFICE A UTILIZATORILOR 19
4.3 PROCEDURA DE ALOCARE SI UTILIZARE A RESURSELOR 20

5. Biroul virtual al utilizatorului 21

5.1 PROIECTAREA SI DEZVOLTAREA COMPONENTELOR FRONTEND 21
5.2 PROIECTAREA SI DEZVOLTAREA COMPONENTELOR BACKEND 24

6. Monitorizarea centralizata a resurselor sistemului cloud federalizat 25

6.1 MONITORIZAREA RESURSELOR DE CATRE ADMINISTRATOR 26
6.2 MONITORIZAREA RESURSELOR DE CATRE UTILIZATORI 28

7. Implementarea SCF 29

7.1 MIGRAREA INFRASTRUCTRII CENTRALE DE MANAGEMENT 29
7.2 SECURIZAREA COMUNICARII DE MESAJE 29
7.3 ADMINISTRAREA CERERILOR DE ÎNREGISTRARE 29
7.4 ADMINISTRAREA UTILIZATORILOR SCF 30
7.5 ADMINISTRAREA PROIECTELOR DE CALCUL 32
7.6 ADMINISTRAREA PROIECTELOR DE CĂTRE UTILIZATOR 32

8. Serviciu de inventariere si administrare a pachetelor software 35

8.1 OBIECTIV 35
8.2 COMPONENTA BACKEND 35
8.3 COMPONENTA FRONTEND 39

9. Agenti 42

10. Concluzii 43

11. ANEXA I 44

12. ANEXA II – Manualul Utilizatorului 48

Lista figurilor

FIG. 4.1: Schema arhitecturii sistemului cloud federalizat 18
FIG. 5.1: Pagina principala a biroului virtual 21
FIG. 5.2: Pagina de management a proiectelor de calcul 22
FIG. 5.3: Interfata de management a resurselor cloud 23
FIG. 5.4: Raportarea statistica privind utilizarea resurselor Cloud 23
FIG. 6.1: Schema SCF, cuprinzand *DB Res* care primeste informatii de la agenti. 25
FIG. 6.2: Pagina de listare a serviciilor de calcul 27

Rezumat

Scopul acestui Raportului Tehnic

Scopul principal al raportului RT 2.2 este de a prezenta rezultatele obținute în cadrul Subactivității 2.2 a proiectului privind realizarea unei solutii tehnice de interconectare la nivel national a unor centre Cloud private din institutii apartinand sistemului public de CD sau de invatamant, care sa fie capabila sa ofere utilizatorilor acces printr-o interfata unica la resurse si servicii furnizate de catre acestea. Totodata, se raporteaza dezvoltarea interfeței web de acces al utilizatorilor la resursele federalizate.

Impact

Rezultatele Subactivitatii 2.2 raspund scopului Actiunii 1.1.2 a POC privind *„dezvoltarea unor retele de centre C-D, coordonate la nivel national si racordate la retele europene...*", prin implementarea unei platforme de interconectare a centrului CLOUDIFIN cu alte doua centre Cloud, realizandu-se astfel un sistem pilot de partajare a resurselor care poate fi extins la nivel national. Interfata unica de accesare a resurselor federalizate va facilita si spori utilizarea tehnologiei Cloud pentru colaborare in reteaua nationala de CDI, va stimula cresterea competitivitatii institutiilor participante si a numarului de proiecte elaborate in comun. Prin federalizarea resurselor Cloud, proiectul devine un vector de propagare al know-how-ului in tehnologii IT avansate catre zone clasificate de EU ca fiind mai putin dezvoltate, unde va contribui la facilitarea activitatii de cercetare, la atragerea si formarea de personal calificat

Conținutul Raportului Tehnic

In Introducerea raportului se motiveaza necesitatea realizarii *Sistemului Cloud Federalizat pentru cercetare* (SCF) si se descriu obiectivele, rezultatele asteptate si etapele Subactivitatii 2.2. In Cap. 2 se trec in revista cerintele utilizatorilor, cerintele de sistem si constrangerile legale sau organizationale care privesc functionarea SCF. Pe baza cerintelor, in capitolul urmator sunt prezentate specificatiile tehnice ale SCF si ale interfetei grafice de acces al utilizatorilor (IG), politicile de acces si proceduririle de inregistrare in sistem. Cap. 4 este dedicat proiectarii arhitecturii SCF si a IG, precum si definirii procedurii de alocare si utilizare a resurselor (de) catre solicitanti. In capitolul urmator este descrisa proiectarea si dezvoltarea software a componentelor biroului virtual al utilizatorilor, care asigura accesul acestora la resursele si serviciile Cloud necesare pentru desfasurarea activitatii de cercetare. Cap. 8 prezinta instrumentele de monitorizare a resurselor de catre administratorul si utilizatorii SCF. Implementarea SCF si a interfetelor de administrare si monitorizare asociate este descrisa in Cap. 7. Serviciul Cloud implementat pentru gestiunea pachetelor software instalate pe masinile virtuale este descris in Cap. 8, iar elementele software dezvoltate pentru comunicarea informatiei intre brokeri si infrastructura centrala de management a SCF sunt prezentate in Cap. 9 si Anexa I. In finalul documentului este atasat Manualul Utilizatorului SCF (Anexa II).

Concluziile Raportului Tehnic

Subactivitatea 2.2 se încheie cu realizarea versiunii pilot a Sistemului Cloud Federalizat pentru cercetare si a rezultatului planificat nr. 19 al proiectului - " *Interfata de acces al utilizatorilor la resurse oferite de centre Cloud multiple. Manual de utilizare*". Astfel, se indeplineste Obiectivul O3 al proiectului, "*Realizarea în cadrul centrului de resurse Cloud a unei soluții tehnice capabilă să interconecteze la nivel național, în cadrul unui sistem federalizat, centrele de tip Cloud privat dezvoltate în instituții aparținând sistemului de CDI, care să ofere utilizatorilor acces printr-o interfață unica la resurse și servicii furnizate de către aceste centre.*"

Solutia tehnica dezvoltata in cadrul proiectului permite transferul catre alte centre din tara, cu posibilitatea realizarii un sistem Cloud federalizat la nivel national.

1. Introducere

1.1 Context general și necesitate

Obiectivul principal al proiectului CeCBiD-EOSC este modernizarea infrastructurii Cloud și extinderea infrastructurii Big Data în vederea creșterii contribuției și vizibilității IFIN-HH în cadrul Cloud-ului European pentru Știinta Deschisă (EOSC).

Cel de-al doilea obiectiv general al proiectului este de a propune o solutie tehnica pentru interconectarea mai multor centre Cloud locale ale diferitelor institutii intr-un sistem federalizat.

Solutia propusa rezolva o problema de durata cu care se confrunta sistemul CDI national: absenta unui sistem de partajare coordonata a resurselor Cloud din tara dedicate cercetarii stiintifice. Desi scopul Actiunii 1.1.2, identic cu cel al competitiei Grid din 2008 lansata in cadrul Operatiunii 2.2.3 POSCCE, este *„dezvoltarea unor rețele de centre C-D, coordonate la nivel national si racordate la retele europene...*", la momentul depunerii propunerii de proiectului in Romania nu exista o retea de centre Cloud interconectate prin software analoaga retelei centrelor Grid, iar acordarea finantarii pe baza de proiecte institutionale individuale nu a favorizat colaborarea intre competitorii din aceeasi regiune de dezvoltare.

Proiectul raspunde cerintelor actiunii 1.1.2 prin dezvoltarea unei platforme de interconectare a centrului CLOUDIFIN cu doua centre Cloud din regiuni mai putin dezvoltate, realizandu-se astfel un sistem pilot de partajare a resurselor care poate fi extins la nivel national.

Prin aceasta se urmareste implementarea unui sistem de partajare coordonata a resurselor Cloud din țară dedicate cercetarii stiintifice, în special a centrelor Cloud care sunt finanțate prin proiectele POC propuse in cadrul apelului 398/2018, rezolvându-se astfel problema repartiției dezechilibrate a resurselor de calcul științific cu care se confruntă în prezent sistemul CDI național.

1.2 Obiectivele Subactivității 2.2

Conform Cererii de Finantare, obiectivul subactivitatii A2.2 este realizarea unei platforme pilot capabila sa interconecteze la nivel national, in cadrul unui *sistem cloud federalizat* (SCF), centre cloud independente dezvoltate in institutii apartinand sistemului de CDI. Sistemul trebuie sa asigure partajarea coordonata a resurselor cloud din tara dedicate cercetarii stiintifice, astfel incat capacitatea de calcul cloud oferita de SCF unui utilizator sa depaseasca disponibilul care poate fi asigurat acestuia de catre oricare dintre centrele cloud individuale conectate.

Interactiunea cu platforma pilot trebuie sa se faca prin interfete grafice (IG), care le vor oferi utilizatorilor acces la resursele furnizate de catre centre si servicii, iar administratorilor SCF accesul la mijloacele software necesare pentru managementul si monitorizarea infrastructurii de interconectare a centrelor.

In vederea implementarii si testarii platformei pilot au fost incheiate acorduri de colaborare cu doua institutii din tara care au depus proiecte de tip cloud in cadrul apelului POC 398/1/1: Universitatea "Ovidius" din Constanta si INCDTIM-Cluj.

Obiectivele specifice ale Subactivității 2.2 sunt următoarele:

1. Federalizarea resurselor oferite de catre CLOUDIFIN si alte centre Cloud

- se realizeaza prin instalarea si configurarea software-ului dezvoltat in cadrul proiectului precum si a serviciilor OpenStack care vor permite autorizarea, accesarea si monitorizarea resurselor tuturor centrelor conectate;

- utilizatorii sunt autorizati pe baza serviciilor specifice OpenStack, dupa care vor avea acces la resursele centrelor Cloud prin intermediul interfetei web;
- in functie de aplicatia software pe care opteaza sa o foloseasca, utilizatorul va fi transferat prin intermdiul sistemului de management al interfetei catre centrul care indeplineste aceasta cerinta si care dispune de resurse libere. In cazul in care nu exista resurse libere utilizatorul are posibilitatea de a trimite job-ul in asteptare catre centrul cel mai putin aglomerat.

2. Extinderea accesului utilizatorilor la resursele Cloud federalizate prin intermediul interfetei web.

Interfata extinsa va oferi utilizatorilor:

- posibilitatea de creare a tiparelor (*templates*) de masini virtuale (VM); template-urile vor fi preluate de catre serviciile OpenStack pentru crearea VM-urilor;
- salvarea template-urilor de VM intr-o baza de date in scopul refolosirii acestora;
- vizualizarea de informatii privind atat imaginile VM stocate cat si *flavor*-uri de VM-uri, oferita la generarea template-urilor
- posibilitatea verificarii starii resurselor cloud (status VM, volume de stocare, retele, etc);
- posibilitatea de lansare de job-uri paralele (MPI) atat pe cluster nativ (fizic) HPC, cat si pe cluster virtual (in Cloud), folosind un manager de job-uri integrat.

3. Optimizarea si validarea functionarii sistemului

1.3 Rezultate preconizate

- Platforme pilot capabila sa interconecteze la nivel national, in cadrul unui sistem cloud federalizat (SCF), centre cloud independente din sistemul de CDI.
- Interfata grafica (IG) de acces al utilizatorilor la resurse oferite de centre Cloud multiple.

1.4 Etapele subactivitatii

Realizarea subactivitatii 2.2 cuprinde urmatoarele etape:

1) Stabilirea cerintelor utilizatorilor, sistemului, precum si a constrangerilor legale / organizationale
2) Elaborarea specificatiilor SCF si IG, pe baza analizei cerintelor
3) Analiza resurselor existente si planificate din centrele cloud candidate la integrare
4) Proiectarea SCF si a IG
5) Dezvoltarea software a componentelor SCF si IG
6) Implementarea SCF si IG; conectarea centrelor cloud candidate la platforma pilot
7) Testarea SCF si a IG in diferite cazuri de utilizare si scenarii
8) Optimizarea si validarea sistemului

2. Cerinte

Cerintele utilizatorilor, sistemului si constrangerile externe de natura legala si organizationala au fost determinate in urma consultarii potentialilor beneficiari, a echipei tehnice a proiectului, precum si a documentatiei tehnice si juridice publicate.

2.1 Cerintele utilizatorilor

La nivelul utilizatorilor din domeniul cercetarii majoritatea cerintelor de calitate a serviciului sunt similare cu cele ale clientilor cloud-urilor comerciale. Exista insa si cerinte specifice care se datoreaza diferentelor dintre profilul acestor utilizatori si cel al clientilor obisnuiti ai furnizorilor de servicii cloud comerciale. Astfel, majoritatea utilizatorilor care provin din mediul stiintific folosesc o multitudine de aplicatii software din domeniul propriu de cercetare si au cunostinte avansate de programare si de management al aplicatiilor respective. Din acest motiv, atunci cand au nevoie de resurse suplimentare prefera migrarea catre sisteme cloud de tip IaaS care sa le confere libertate deplina privind reproducerea mediului de lucru initial. In acest scop, solicita personalizarea mediului virtual de lucru pentru portarea aplicatiilor respective impreuna cu toate dependintele acestora, iar aceasta operatiune necesita de multe ori suportul si interactia stransa cu managementul centrelor de resurse.

Majoritatea cerintelor de mai jos descriu performante ideale din punctul de vedere al utilizatorului SCF, care insa in practica nu se pot atinge intotdeauna datorita limitarilor fizice ale resurselor si serviciilor asigurate de sistem.

U1 *Acces*: autentificarea si autorizarea accesului la resurse/servicii sa fie facila dar sigura

U2 *Resurse adecvate*: caracteristicile tehnice ale resurselor hardware si software oferite la inregistrare sa fie cat mai apropiate de cele solicitate de catre utilizator

U3 *Servicii adecvate*: serviciile oferite sa corespunda exact necesitatilor utilizatorului

U4 *Adaptabilitate*: mediul virtual de lucru sa poata fi personalizat conform cerintelor utilizatorului

U5 *Alocarea resurselor*: procesul de alocare a resurselor sa fie cat mai simplu si intervalul de timp dintre solicitarea si alocarea acestora cat mai scurt

U6 *Disponibilitate*: serviciile oferite sa fie disponibile conform necesitatilor utilizatorului (ideal 24/7)

U7 *Integritatea datelor*: datele utilizate sau generate sa nu fie corupte, sa fie transferate integral si sa existe o gestiune coordonata a diferitelor versiuni de fisiere

U8 *Protejarea datelor*: utilizatorul beneficiaza de instrumente de reducere a riscului pierderii datelor

U9 *Securitate*: contul, aplicatiile, stocarea si comunicarea de date sa fie protejate la acces neautorizat

U10 *Drepturi legale*: drepturile de proprietate asupra datelor si de protectie a confidentialitatii pe care le are utilizatorul si/sau angajatorul acestuia trebuie sa fie asigurate conform legislatiei nationale

U11 *Integritatea serviciului*: perturbarea serviciilor SCF, a IG si a utilizarii aplicatiilor, care este datorata operatiunilor tehnice de rutina (mentenanta, actualizari software, etc.) trebuie sa fie minima

U12 *Costuri*: serviciile sa fie oferite gratuit sau la preturi minime in raport cu oferta de pe piata

U13 *Asistenta*: utilizatorul sa beneficieze de servicii de suport asigurate pe toata durata folosirii SCF

2.2 Constrangeri legale si organizationale

Acestea se aplica atat utilizatorilor cat si personalului de management cloud.

L1 *Utilizatori acceptati*: dat fiind caracterul de infrastructura nationala al SCF, sunt acceptati ca utilizatori ai acestuia doar persoanele fizice pentru care este aplicabila legislatia romana. Pentru utilizatorii internationali CLOUDIFIN ofera resurse in colaborare cu EGI.

L2 *Constrangeri legale*: toate operatiunile desfasurate in cadrul SCF si IG trebuie sa respecte legislatia nationala si reglementarile internationale, inclusiv privind protectia datelor personale (GDPR)

L3 *Constrangeri organizationale*: atat utilizarea resurselor cu care o entitate participa la SCF cat si accesul managementului centralizat al SCF la aceste resurse se fac cu acordul si respectarea regulamentelor de organizare si functionare a entitatii respective

2.3 Cerinte de sistem

Prin definitie, orice sistem cloud trebuie sa satisfaca o serie de cerinte de baza obligatorii, cum sunt disponibilitatea, scalabilitatea, elasticitatea, izolarea datelor si serviciilor, confidentialitatea, securitatea, etc. Aceste proprietati sunt asigurate in oricare dintre centrele cloud componente ale SCF prin intermediul managerului de resurse virtuale al sistemului cloud respectiv si nu vor fi prezentate aici. In continuare se vor discuta doar acele aspecte ale cerintelor tehnice si organizatorice care sunt specifice federalizarii centrelor cloud nationale in cadrul SCF.

S1 *Resurse fizice*: SCF trebuie sa dispuna de o capacitate minima de resurse dedicate, asigurata prin contributiile centrelor cloud componente.

S2 *Resurse virtuale*: SCF trebuie sa fie capabil sa ofere utilizatorilor posibilitatea de a alege diferite tipuri (*flavors*) de masini virtuale (*virtual machines* - VM) generate de catre centrele cloud componente

S3 *Interconectare*: centrele cloud componente trebuie sa poata comunica intre ele si cu managementul SCF printr-o retea internet sigura, cu latime de banda suficienta pentru transferul de date si sustinerea serviciilor

S4 *Interoperabilitate*: centrele cloud componente ale SCF trebuie sa fie compatibile, existand posibilitatea portarii aplicatiilor pe acele centre care dispun de resurse adecvate acestora

S5 *Managementul resurselor*: atat informatiile privind resursele SCF, furnizate de catre centrele componente, cat si alocarea acestora sunt gestionate centralizat

S6 *Monitorizare*: monitorizarea resurselor si serviciilor SCF se face centralizat

S7 *Accesul utilizatorilor*: inregistrarea, autentificarea si autorizarea utilizatorilor se realizeaza centralizat

S8 *Interfete*: accesul utilizatorilor, interactia acestora cu sistemul, administrarea si monitorizarea SCF se vor face prin intermediul unor interfete grafice (IG)

S9 *Comunitati de utilizatori*: pentru simplificarea alocarii resurselor virtuale, se definesc grupe / comunitati ale utilizatorilor care necesita acelasi tip de mediu virtual de lucru (de regula, o comunitate de utilizatori este legata de acelasi (sub)domeniu de cercetare si foloseste aceleasi aplicatii software)

S10 *Flexibilitate*: mediile virtuale de lucru servite utilizatorilor trebuie sa permita personalizarea conform cerintelor acestora (cf. U4)

S11 *Securitatea*: trebuie sa fie asigurata la nivelul centrelor de resurse componente ale SCF, la nivelul retelei internet de interconectare a acestora, la nivelul interfetelor grafice si la nivelul managementului central

S12 *Disponibilitatea serviciilor*: necesita asigurarea acesteia la nivel de infrastructura (centre de resurse, retea) si al managementului central

S13 *Politici si proceduri*: pentru implementarea si exploatarea SCF este necesara definirea de politici si proceduri comune, care sa fie respectate de catre personalul centrelor de resurse componente si de catre institutiile care le gazduiesc

3. Specificatiile tehnice ale Sistemului Cloud Federalizat

3.1 Conditii de integrare a centrelor Cloud in SCF

Pentru a putea fi integrate in SCF si a oferi in cadrul acestuia calitatea serviciului necesara, centrele de resurse cloud trebuie sa satisfaca urmatoarele cerinte tehnice si organizatorice:

I1 Sa dispuna de infrastructuri software cloud compatibile, capabile de interoperabilitate (S4);

I2 Sa fie conectate la aceeasi retea comuna de date de mare viteza, fiabila, printr-o legatura de date cu latime de banda minima garantata (S3);

I3 Sa ofere un nivel minim garantat de resurse dedicate SCF (S1);

I4 Resursele hardware dedicate SCF (pentru procesarea, stocarea si comunicarea locala de date) sa satisfaca cerinte minime de calitate, inclusiv pentru siguranta datelor;

I5 Sa respecte un nivel minim de disponibilitatea a serviciilor furnizate utilizatorilor SCF (S12);

I6 Sa permita instalarea de software client pentru federalizare si monitorizare centralizata (S5, S6);

I7 Administratorii centrelor trebuie sa adere la un ansamblu de politici si proceduri comune privind managementul serviciilor la nivelul SCF (S13).

3.2 Specificatii de sistem

Federalizarea centrelor de resurse cloud se realizează prin instalarea și configurarea software-ului dezvoltat în cadrul proiectului, precum si prin intermediul serviciilor asigurate de sistemele de management al resurselor virtuale ale centrelor.

De asemenea, interoperabilitatea (S4) este implementată de sistemele de management al resurselor virtuale ale centrelor componente, care oferă servicii compatibile (I1).

SCF este gestionat prin intermediul infrastructurii centrale de management (ICM), care asigură serviciile centrale privind: autentificarea si autorizarea utilizatorilor (S7); managementul resurselor (S5); monitorizarea resurselor si serviciilor furnizate utilizatorilor (S6), etc.

In cadrul ICM coordonarea utilizarii resurselor se realizează cu ajutorul unui sistem de informatii actualizate pe baza datelor primite de la agenți care ruleaza in centrele de resurse componente ale SCF.

Înregistrarea, autentificarea și autorizarea utilizatorilor se realizeaza prin intermediul unei interfațe grafice (IG) unice, a serviciilor asigurate de ICM și a unei baza de date in care se vor stoca credențialele utilizatorilor.

Monitorizarea disponibilitatii resurselor si serviciilor oferite de catre centrele locale, precum si resursele utilizate (masini virtuale, etc) se face prin intermediul unor aplicatii software client care vor rula local si vor transmite datele catre un server din cadrul ICM.

Fiecare centru component al SCF implementeaza si actualizeaza o baza de date locala, accesibila utilizatorilor prin intermediul ICM, cu informatii privind tipurile (*flavors*) de masini virtuale care pot fi generate in centrul respectiv (S2).

Prin intermediul IG utilizatorii vor avea posibilitatea de a crea tipare (*templates*) de masini virtuale (VM) cu sisteme de operare si aplicatii software specifice (S10), care vor fi stocate la nivel central in vederea refolosirii.

Atunci cand va fi necesara crearea VM-urilor respective de catre orice utilizator autorizat, tiparele vor fi preluate de catre serviciului de orchestrare al ICM.

Interfata grsfica a utilizatorului va afisa atat oferta de *flavours* disponibile in centrele SCF, cat si cea de *templates* de masini virtuale existente la nivel central, oferindu-i acestuia posibilitatea sa solicite folosirea celor convenabile.

3.3 Politici si proceduri

Politica de acces la resursele SCF este similara cu cea stabilita pentru centrul CLOUDIFIN in cadrul Subactivitatii 2.1. Activitatea fiecarui utilizator SCF se desfasoara exclusiv prin *proiecte de calcul*, care au scopuri, cerinte tehnice si perioade de desfasurare bine definite.

Procedurile de inregistrare a utilizatorilor si a proiectelor de calcul sustinute de SCF sunt aceleasi ca cele stabilite pentru centrul CLOUDIFIN, cu singura diferenta ca in acest caz resursele disponibile sunt distribuite in diferite centre integrate in SCF.

Prin urmare, administratorul SCF, care coordoneaza repartizarea resurselor, trebuie sa aloce o propunere de proiect de calcul unui centru care dispune de resursele solicitate, cu acordul managementului acestuia.

In practica, administratorul SCF poate negocia cu initiatorul proiectului de calcul adaptarea cerintelor la oferta disponibila, iar cu managementul centrelor de resurse adaptarea capacitatii oferite la cerintele utilizatorului.

Pentru respectarea cerintelor privind calitatea si disponibilitatea serviciului, a nivelului si calitatii minim acceptate de resurse dedicate, etc., se incheie în prealabil urmatoarele acorduri:

a) Acord privind nivelul operational, incheiat intre managementul SCF si cele ale centrelor de resurse;

b) Acord privind nivelul serviciilor, incheiat intre furnizorul si beneficiarii serviciului.

De asemenea, se pot incheia Memorandumuri de intelegere intre institutia coordonatoare a SCF si entitatile care administreaza centrele de resurse partenere.

4. Proiectarea platformei pilot

Proiectarea arhitecturii platformei pilot s-a realizat respectand cerintele si constrangerile prezentate in Cap. 2, luand in considerare urmatorii factori: specificatiile din Cap. 3; infrastructura centrelor care urmau să fie conectate; posibilitatea aplicării și extinderii dezvoltarilor software-ul din Subactivitatea 2.1 a proiectului pentru administrarea, monitorizarea si accesul utilizatorilor la resursele CLOUDIFIN.

S-a considerat ca premisele tehnice generale pentru realizarea platformei pilot sunt:

- Utilizarea exclusiva de software open source pentru implementarea infrastructurii, din motive de cost, de compatibilitate intre sistemele de management a resurselor virtuale, precum si de compatibilitate a acestora din urma cu software-ul dezvoltat pentru federalizare.
- Utilizarea software-ului OpenStack pentru furnizarea IaaS de catre centrele de resurse. Aceasta este o cerinta tehnica de interoperabilitate pentru centrele cloud care vor fi conectate la platforma pilot in cadrul acestui proiect. Pentru conectarea unor centre care utilizeaza platforme cloud diferite (de ex. OpenNebula) sunt necesare dezvoltari software care depasesc cadrul proiectului CECBID

4.1 Arhitectura sistemului

Arhitectura sistemului cloud federalizat este prezentata in Fig. 4.1, unde sunt reprezentate infrastructura centrala de management si doua centre de resurse.

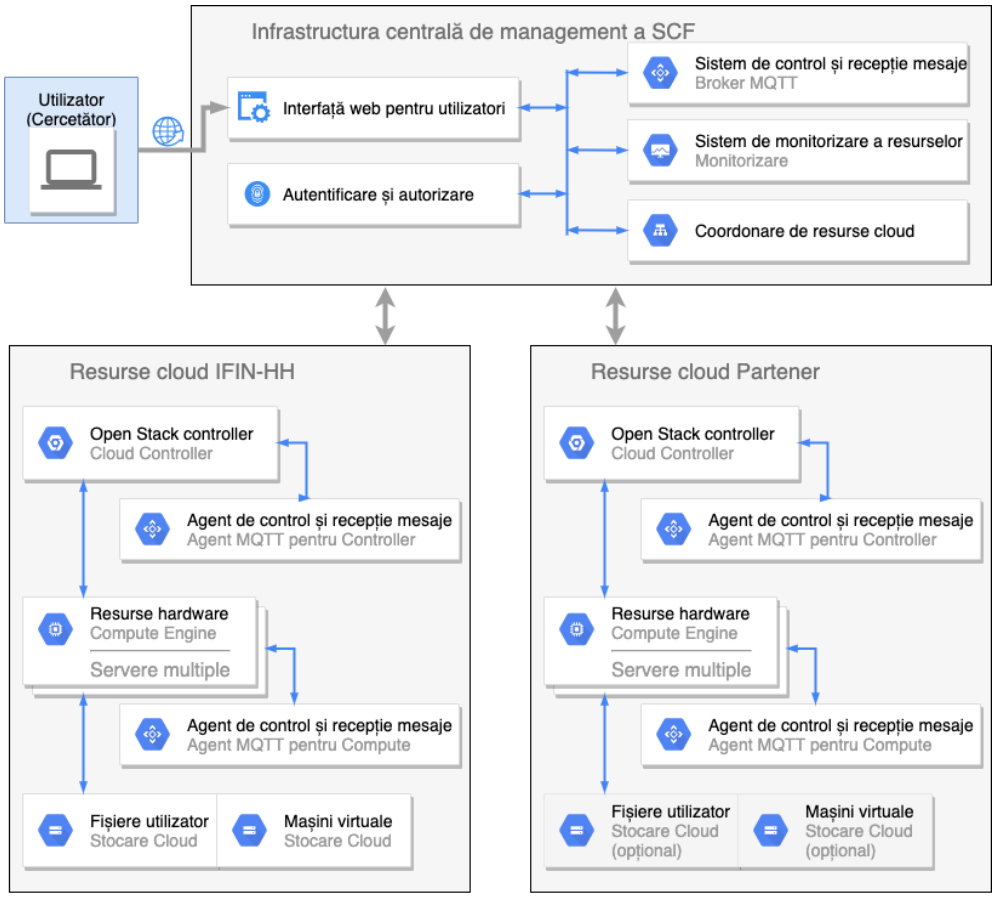


FIG. 4.1: Schema arhitecturii sistemului cloud federalizat

Pentru transportul mesajelor intre componentele SCF se utilizeaza protocolul MQTT, care va asigura comunicarea bidirectionala pentru sistemul de informatii si managementul de resurse. Astfel, se va dezvolta un *Sistem de control si receptie mesaje (SCRM)*, bazat pe protocolul MQTT, ce va comunica cu *Agentul de control si receptie mesaje* specializat, aflat pe controllerul OpenStack, cat si cu *Agentii* aflati pe serverele cu functie de Cloud Compute. Agentii MQTT sunt responsabili cu raportarea catre SCRM a gradului de ocupare a resurselor cloud, dar si de managementul acestora (agentul specific aflat pe controller). Prin intermediul acestui sistem, comenzile de interactiune cu VM-urile sunt transmise de catre utilizator catre centrele de resurse exclusiv prin interfata web a ICM.

Infrastructura centrala de management a SCF este gazduita in cadrul centrului CLOUDIFIN.

Conditiile de integrare a centrelor cloud in SCF sunt urmatoarele:

- Infrastructura de comunicare de date sa fie asigurata de Reteaua Nationala pentru Cercetare si Educatie RoEduNet, la care centrul CLOUDIFIN este conectat printr-o legatura cu latimea de banda de 10 Gb/sec, cu posibilitatea de upgrade pana la 100Gb/sec - latimea maxima disponibila in centrul de date al IFIN-HH. Latimea minima garantata de banda admisa pentru conectarea la platforma pilot va fi de 1 Gb/sec.
- Nivelul minim garantat de resurse dedicate SCF de un centru cloud sa fie negociat si specificat in Memorandumuri de intelegere incheiat intre IFIN-HH si entitatea care administreaza centrul respectiv.
- Siguranta locala a datelor sa fie asigurata prin RAID 6 de catre fiecare centru care furnizeaza servicii de stocare de date.
- Nivelul minim de disponibilitatea a serviciilor furnizate utilizatorilor va fi de 90% si va fi prevazut atat in Acordul privind nivelul operational, incheiat intre managementul SCF si cele ale centrelor de resurse, cat si in Acordul privind nivelul serviciilor, incheiat intre furnizor si utilizatori.
- Sistemul de operare acceptat este Linux, distributiile CentOS sau Ubuntu. Noi distributii pot fi acceptate in masura in care sunt acceptate in urma testelor, dar nu intra in scopul proiectului CECBID.
- Infrastructura de management a resurselor virtuale acceptata este OpenStack.

4.2 Proiectarea interfetei grafice a utilizatorilor

S-a considerat implementarea interfetei grafice a utilizatorilor SCF ca o extensie a interfetei grafice a utilizatorilor sistemului CLOUDIFIN, care a fost dezvoltata in cursul Subactivitatii 2.1.

Componentele/facilitatile adaugate sunt urmatoarele:

- Lista cumulativa a tipurilor (*flavours*) de masini virtuale generate de catre centrele cloud din SCF. Aceasta ii va fi disponibila utilizatorului in cadrul formularul de solicitare a resurselor de calcul, care va selecta tipurile necesare pentru proiectul de calcul. Utilizatorul va avea, de asemenea, posibilitatea sa solicite noi *flavours*, care nu se gasesc in lista prezentata.
- Lista centrala a template-urilor de VM-uri disponibile, selectabile din lista inclusa in formularul de solicitare a resurselor de calcul. Daca nu regaseste in lista un template convenabil, utilizatorul va putea crea unele noi (punctul urmator).
- Posibilitatea de creare de noi tipare de VM-uri; template-urile create vor fi preluate de catre serviciului de orchestrare HEAT pentru crearea VM-urilor respective.
- Posibilitatea de salvare si publicare a template-urilor de VM-uri create, in scopul refolosirii acestora.

- Posibilitatea verificarii starii resurselor cloud (starea VM-urilor create, volume de stocare, retele, etc).

4.3 Procedura de alocare si utilizare a resurselor

Atunci cand propune un nou proiect de calcul, solicitantul transmite cerintele sale privind resursele prin intermediul unuia din formularele web reprezentate din IG. Aceste cerinte se refera la caracteristici generale: timpul estimat de calcul, numarul minim de nuclee CPU (cores) necesare, memoria RAM minima per core, capacitatea de stocare necesara pe termen lung, bibliotecile si instrumentele software necesare, limbajul de programare, etc.

Urmeaza o etapa de brokeraj care, din motive de complexitate a SCF si pentru facilitarea interactiei cu solicitantul, este de preferat sa nu fie automat ci realizat de catre administratorul SCF. Conform cerintei de sistem S5 din raportul RP5-2.2, acesta gestioneaza prin intermediul infrastructurii centrale de management (ICM) informatiile privind resursele SCF, furnizate de catre centrele componente, si negociaza cu managerii locali alocarea resurselor catre utilizatori.

Cerintele solicitantului sunt transmise administratorului SCF, care consulta baza de date centrala a resurselor SCF si verifica existenta resurselor cu caracteristicile tehnice solicitate.

In cazul in care acestea exista, administratorul SCF ii contacteaza electronic pe managerii centrelor cloud care pot oferi resursele respective si verifica intervalul de disponibilitate a acestora.

Daca resursele sunt disponibile, administratorul aproba (din punct de vedere tehnic) propunerea de proiect a solicitantului si ii creaza acestuia contul de utilizator (daca acesta nu exista deja).

In cazul in care SCF nu dispune de resurse care sa corespunda exact cerintelor solicitantului, administratorul cauta in baza de date a resurselor centre care sa gazduiasca resurse cu caracteristici tehnice apropiate de cele ale solicitantului.

Daca acestea exista sau pot fi obtinute prin reconfigurari minimale ale resurselor existente, administratorul ii contacteaza pe managerii centrelor respective si, daca obtine acordul acestora, transmite solicitantului propunerea de modificare a cerintelor sale initiale de resurse cu resursele echivalente.

Daca solicitantul accepta noile conditii, administratorul aproba propunerea de proiect si ii creaza solicitantului un cont de utilizator al SCF.

Dupa aprobarea propunerii de proiect, utilizatorul se autentifica si isi acceseaza biroul virtual (BV), care ii ofera toate utilitatile necesare pentru utilizarea resurselor alocate.

5. Biroul virtual al utilizatorului

5.1 Proiectarea si dezvoltarea componentelor frontend

Biroul virtual cuprinde un set de pagini web cu informatii specifice fiecarui utilizator si accesibile doar de catre acesta, care indeplinesc urmatoarele functii:

a) Informare cu caracter general privind proiectele de calcul ale utilizatorului

In aceasta sectiune se listeaza proiectele in curs sau inactive la care participa sau a participat utilizatorul, preluate din baza de date centrala. Pentru fiecare proiect se afiseaza numele acestuia, apoi organizatia virtuala (VO) folosita, starea (activ/finalizat/suspendat), data de inceput si data de finalizare. Pentru fiecare proiect activ utilizatorul are posibilitatea sa acceseze pagina de management a acestuia, descrisa la punctul urmator.

Mai jos este prezentata pagina web principala dezvoltata pentru biroul virtual al utilizatorului.

FIG. 5.1: Pagina principala a biroului virtual

b) Managementul proiectelor de calcul

Pentru un proiect accesat de utilizator din pagina principala a BV se afiseaza mai intai resursele alocate de comun acord cu administratorul SCF in procesul de aprobare a proiectului (timp de calcul, nr. core, RAM/core, capacitate de stocare, software, etc.). Se afiseaza de asemente imaginile sablon (templates) de VM-uri la care utilizatorul are acces. Utilizatorul poate selecta noi template-uri din lista celor oferite in cadrul SCF, sau ii poate solicita administratorului SCF un nou tip de template ale carui caracteristici le specifica.

Pagina web de management a unui proiect de calcul („Analiza datelor de secventiere de noua generatie”) este prezentata in Fig. 5.2.

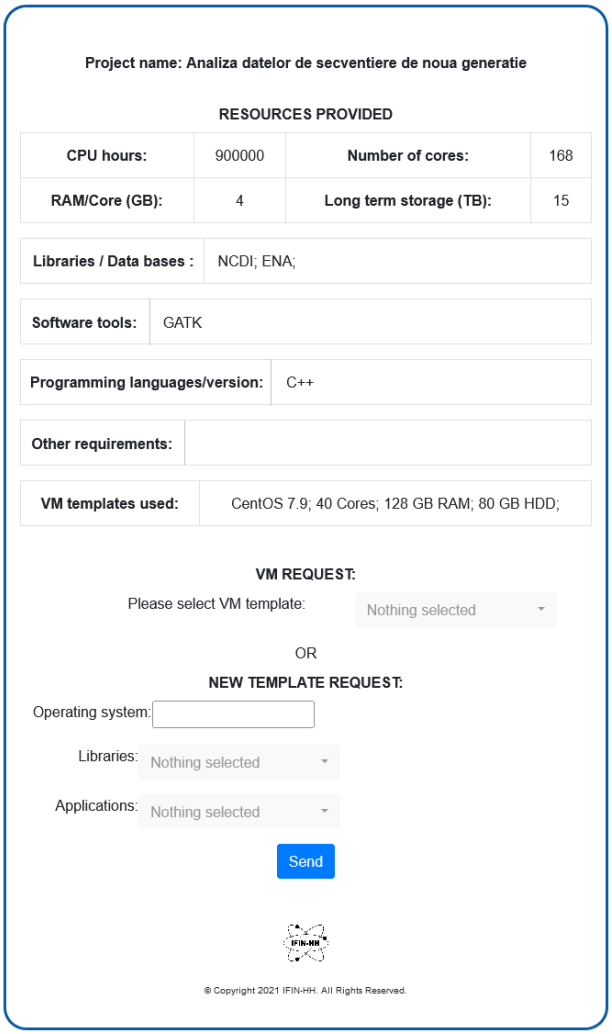


FIG. 5.2: Pagina de management a proiectelor de calcul

c) Managementul resurselor Cloud

Butonul „Manage resources” din pagina principala a BV trimite catre interfata de management a resurselor tuturor proiectelor Cloud active in care utilizatorul este inregistrat (in OpenStack numele proiectelor de calcul coincid cu numele VO-urilor folosite).

Interfata ii ofera utilizatorului informatii in timp real, obtinute de la agentii care ruleaza pe nodurile de calcul din site-urile Cloud ale SCF, privind masinile virtuale (VM) pe care le gestioneaza, organizatiile virtuale, data creerii VM-urilor, gradul de incarcare al resurelor CPU si memoria RAM folosita.

Din interfata de management, utilizatorul are posibilitatea sa-si editeze VM-urile, sa preia terminalul/consola oricarei masinii virtuale prin VNC (*Virtual Network Computing*), sa opreasca temporar, sa restarteze sau sa distruga orice VM.

Imaginea interfetei web dezvoltate pentru managementul resurselor Cloud alocate utilizatorului este reprodusa in Fig. 5.3.

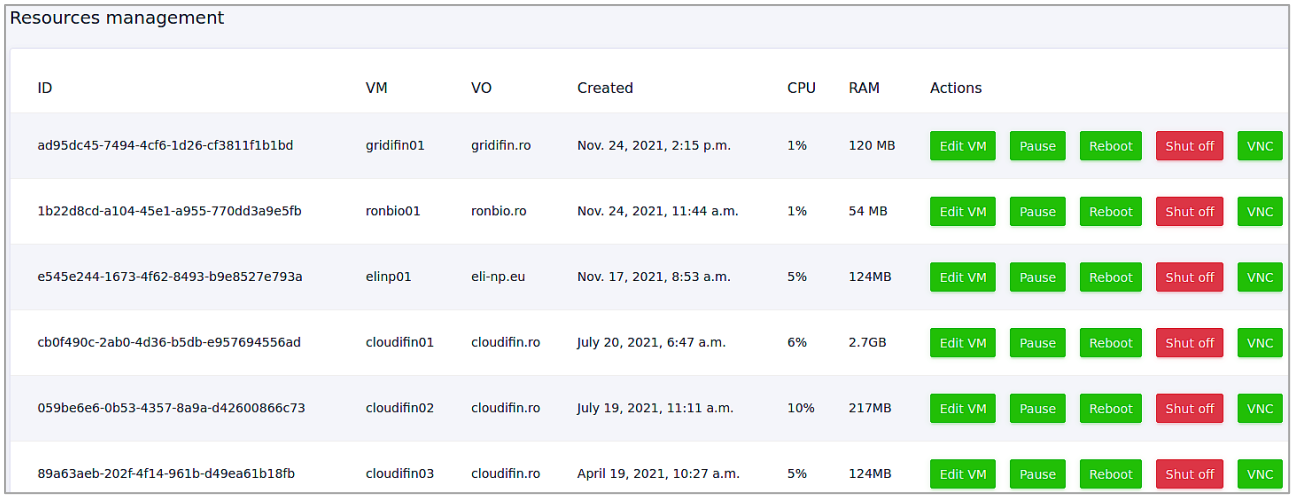


FIG. 5.3: Interfata de management a resurselor cloud

d) Furnizarea de statistici in timp real

Prin butonul „Live Statistics” se pot vizualiza in timp real date statistice privind resursele consumate de masinile virtuale, pe baza informatiilor transmise prin MQTT de agentii care ruleaza pe nodurile de calcul.

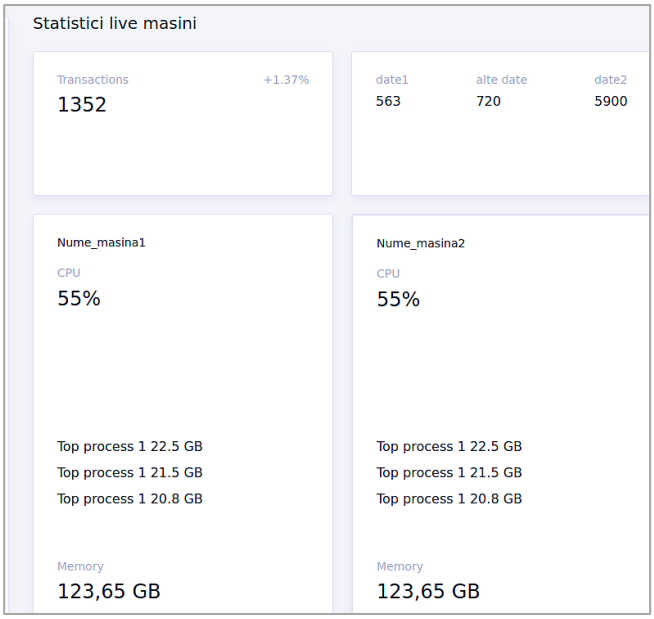


FIG. 5.4: Raportarea statistica privind utilizarea resurselor Cloud

e) Managementul profilului utilizatorului

Din pagina principala utilizatorul isi poate accesa pagina de management a profilului, unde poate actualiza parola de acces la cont, datele personale si de contact (institutia, functia/titlul, nr. telefon, adresa email).

f) Comunicare cu administratorul SCF

Butonul „Admin messaging” din pagina principala a BV trimite catre interfata web de comunicare cu administratorul SCF, care transmite informatia prin intermediul email.

g) Documentare

Din pagina principala a BV utilizatorul are acces la documentatia de sistem, la documentatia privind bibliotecile software disponibile si la manualul utilizatorului.

5.2 Proiectarea si dezvoltarea componentelor backend

Componentele software backend sunt dezvoltate folosind exclusiv software *open source*, in limbaj *python* si utilizand *Chart.js* pentru realizarea graficii statisticilor.

Testarea componentelor software dezvoltate s-a facut mai intai utilizand site-ul CLOUDIFIN, care gazduieste Infrastructura Centrala de Management (ICM) a SCF si un alt site cloud (Partner01) implementat ad-hoc in cadrul IFIN-HH cu respectarea design-ului de arhitectura al SCF prezentat in raportul anterior.

Monitorizarea disponibilitatii resurselor si serviciilor oferite de catre centrele locale, precum si a resurselor utilizate (masini virtuale, etc), s-a realizat prin intermediul unor aplicatii software client (agenti) care ruleaza local pe nodurile de calcul si transmit datele catre un server din cadrul infrastructurii ICM prin mesaje *publish* gestionate prin protocolul MQTT.

Functionalitatea agentilor de pe nodurile de calcul este asigurata utilizand biblioteca client python *paho-mqtt*.

Sistemul de control si receptie al mesajelor (SCRM) aflat pe controllerul OpenStack, comunica atat cu agentii de control si receptie a mesajelor, cat si cu agentii instalati pe nodurile de calcul. Acestia raporteaza catre SCRM gradul de ocupare a resurselor cloud, iar agentul aflat pe controller coordoneaza operatiunile de management de pe clienti.

Prin intermediul SCRM, comenzile de interactiune cu VM-urile sunt transmise de catre utilizator catre centrele de resurse exclusiv prin interfata web a ICM.

Fiecare centru component al SCF dispune de o baza de date locala furnizata de OpenStack si accesibila administratorului SCF prin intermediul ICM, cu informatii privind tipurile (*flavors*) de masini virtuale care pot fi generate in centrul respectiv.

ICM dispune de un sistem de informatii care sunt actualizate pe baza datelor primite de la agentii care ruleaza in centrele de resurse componente ale SCF.

Pentru testarea sistemului de transmitere a mesajelor intre ICM si site-ul cloud partener s-a monitorizat de pe CLOUDIFIN starea serviciilor cloud de pe un nod de calcul gestionat de cloud-ul partener (Partner01).

Procedura de monitorizare este descrisa mai jos.

Pe Partner01 ruleaza o aplicatie client python (agent), care asteapta un mesaj ce contine string-ul „status”. Dupa ce este validat mesajul, executa o comanda Linux care verifica starea serviciului cloud. Output-ul comenzii Linux este impachetat (*wrapped*) intr-o varibila „data” si transmis catre SCRM , unde este afisat in interfata utilizatorului.

Daca utilizatorului i s-a permis accesul atat la resurse din CLOUDIFIN cat si din Partner01, atunci acesta va putea obtine lista completa a starii serviciilor de pe masinile ambelor site-uri cloud, ca in imaginile prezentate in acest livrabil.

Atunci cand utilizatorul doreste actualizarea informatiilor privind starea serviciilor, utilizeaza butoanele „Status Service”. In absenta interventiei utilizatorului informatia se actualizeaza la un interval de timp fixat de catre administratorul SCF (in acest caz 1 min.)

6. Monitorizarea centralizata a resurselor sistemului cloud federalizat

Monitorizarea resurselor SCF se realizeza utilizand aplicatii software client (agenti) care ruleaza local pe controllere si noduri de calcul ale site-urilor si transmit datele catre un server din cadrul infrastructurii ICM prin mesaje *publish* gestionate prin protocolul MQTT.

Sistemul de control si receptie al mesajelor (SCRM) instalat pe un server din ICM comunica atat cu agentii de control si receptie a mesajelor (de pe controllere), cat si cu agentii instalati pe nodurile de calcul. Acestia raporteaza catre SCRM gradul de ocupare a resurselor cloud, iar agentii aflati pe controllere coordoneaza operatiunile de management de pe clienti. Prin intermediul SCRM, comenzile de interactiune cu VM-urile sunt transmise de catre utilizator catre centrele de resurse exclusiv prin interfata web a ICM.

ICM dispune de o baza de date a resurselor (*DB Res* in Fig.6.1) cu informatii care sunt actualizate pe baza datelor primite de la agentii care ruleaza in centrele de resurse (site-urile) componente ale SCF.

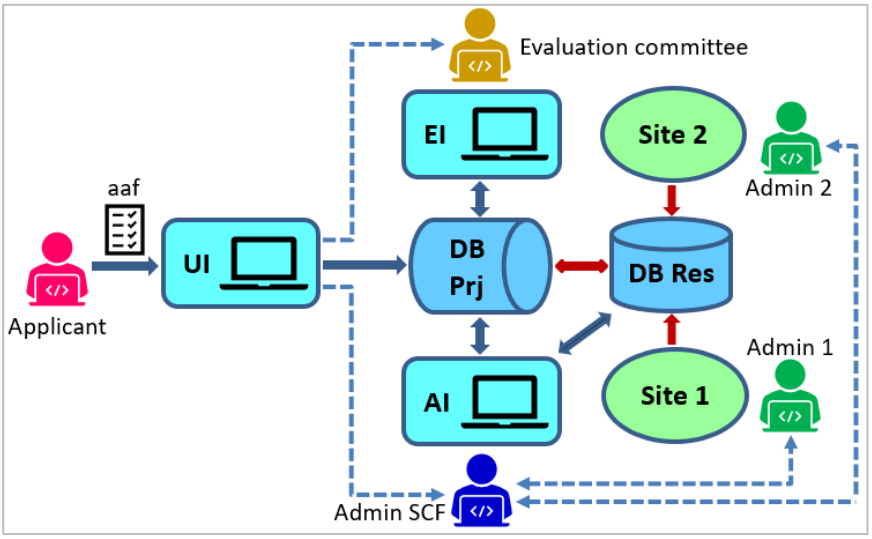


FIG. 6.1: Schema SCF, cuprinzand *DB Res* care primeste informatii de la agenti.

In figura mai sunt reprezentate interfetele grafice ale utilizatorilor (UI), evaluatorilor (EI), respectiv administratorului SCF (AI) si baza de date a proiectelor / utilizatorilor SCF (DB Prj)

Pentru ca interfata web a ICM sa fie informata in orice moment privind nivelul de ocupare a resurselor hardware din centrele cloud, in agentul care ruleaza pe controllere a fost dezvoltata o functie de descoperire a nodurilor de calcul. Procedeul de descoperire a arhitecturii cloud consta in urmatorii pasi: a) interfata web (dispecer) a ICM trasmite un mesaj mqtt catre agentul care ruleaza pe controller-ul centrului cloud; b) in urma receptionarii acestui mesaj, pe controller-ul partenerului se executata automat comanda "`openstack compute service list`", care va lista toate masinile din respectivul centru cloud; c) output-ul este transmis catre interfata web a ICM sub forma de string de date; d) in urma formatarii string-ului se genereaza un fisier csv, al carui continut va fi inserat in *DB Res*.

Continutul fisierului csv este de forma:

```
ID,Binary,Host,Zone,Status,State,UpdatedAt
1,nova-scheduler,ctrl-os,internal,enabled,up,2022-03-02T02:53:30.000000
23,nova-compute,bchs65,nova,enabled,up,2021-07-18T19:29:34.000000
24,nova-compute,bchs66,nova,enabled,up,2021-07-18T19:30:19.000000
```

Pentru ca agentul care ruleaza pe un controller sa poata executa comenzi OpenStack, acesta trebuie sa cunoasca o serie de variabile de sistem. In acest scop agentul necesita un fisier de configurare (`admin-openrc`) care sa contina toate variabilele de sistem necesare executrii comenzii ce listeaza toate resursele centrului cloud.

6.1 Monitorizarea resurselor de catre administrator

Pentru administratorul centrului cloud continutul fisierului de configurare este urmatorul:

```
[config]
OS_PROJECT_NAME=admin
OS_USERNAME=admin
OS_PASSWORD=********
OS_AUTH_URL=http://ctrl-os.nipne.ro:5000/v3
OS_IDENTITY_API_VERSION=3
```

Acest fisier trebuie editat de catre administrator, operatiunile permise fiind de management al utilizatorilor, al proiectelor si al masinilor virtuale

Doar administratorul centrului cloud detine un astfel de fisier de configurare care contine credentialele pentru accesarea propriului centru cloud.

Pentru a putea monitoriza resursele hardware si serviciile la nivel de resurse de calcul, functionalitatile agentul prezentat mai sus au fost extinse si pe nodurile de calcul. Rularea agentului pe noduri nu necesita fisierul de configurare, care este folosit doar pe controller.

In continuare se reproduce codul corespunzator functionalitatii din agent corespunzatoare setarii variabilelor din fisierul de configurare.

```
def on_message(mosq, obj, msg):
    if msg.topic == "cloud/cloudifin/topologie":
       if msg.payload == "req":
          config = configparser.ConfigParser()
          config.read('admin-openrc')
          username = config['config']['OS_USERNAME']
          auth_url = config['config']['OS_AUTH_URL']
          project_name = config['config']['OS_PROJECT_NAME']
          identity_api = config['config']['OS_IDENTITY_API_VERSION']
          passwd = config['config']['OS_PASSWORD']
          command =
("openstack","--os-username",username,"--os-auth-url",auth_url,"--os-project-
name",project_name,"--os-identity-api-version",identity_api,"--os-
password",passwd,"compute","service","list")
          p = subprocess.Popen(command, stdout=subprocess.PIPE)
          stdout, stderr = p.communicate()
          mqttc.publish("cloud/cloudifin/tolopogie/res", stdout)
```

Dupa parsarea datelor din fisierul de configurare, acestea sunt atribuite unor variabile care vor fi folosite ca parametri atunci cand se va executa comanda de listare a tuturor masinilor din cloud

```
command =
```

```
("openstack","--os-username",username,"--os-auth-url",auth_url,"--os-project-
name",project_name,"--os-identity-api-version",identity_api,"--os-
password",passwd,"compute","service", "list")
```

Sirul de date este apoi transmis pe canalul "`cloud/cloudifin/tolopogie/res`"

Brokerul mqtt receptioneaza mesajul si il formateaza sub forma unui fisier csv pe care il importa in baza de date.

In cazul site-ului CLOUDIFIN continutul acestui fisier este de forma:

4,nova-scheduler,cloud-ctrl.nipne.ro,internal,enabled,up,2023-05-11T07:57:58.000000

16,nova-conductor,cloud-ctrl.nipne.ro,internal,enabled,up,2023-05-11T07:57:58.000000

27,nova-consoleauth,cloud-ctrl.nipne.ro,internal,enabled,up,2023-05-11T07:57:51.000000

28,nova-compute,cloud-vm01-test,nova,enabled,down,2022-03-17T07:50:54.000000

31,nova-compute,cloudcompute01,nova,enabled,down,2022-03-17T07:51:05.000000

32,nova-compute,cloudcompute02,nova,enabled,up,2023-05-11T07:57:49.000000

33,nova-compute,cloudcompute03,nova,enabled,up,2023-05-11T07:57:56.000000

34,nova-compute,cloudcompute04,nova,disabled,down,2020-03-19T10:07:38.000000

35,nova-compute,cloudcompute05,nova,disabled,down,2020-03-19T10:09:42.000000

36,nova-compute,cloudcompute06,nova,disabled,down,2020-03-19T10:09:54.000000

37,nova-compute,cloudcompute07,nova,disabled,down,2020-03-19T10:10:08.000000

38,nova-compute,cloudcompute08,nova,disabled,down,2020-03-19T10:10:37.000000

Pentru celelalte site-uri cloud din SCF acest fisier are acelasi format, dar numele host-ului, controller-ului si ale nodurilor de calcul sunt diferite.

In interfata web datele importate sunt prezentate ca in Fig. 6.2.

LIST Compute Service

ID_Cloud	Service	Host	Zone	Status	State	Update	Edit	Delete
4	nova-scheduler	cloud-ctrl.nipne.ro	internal	enabled	up	2023-05-11T07:57:58.000000		
16	nova-conductor	cloud-ctrl.nipne.ro	internal	enabled	up	2023-05-11T07:57:58.000000		
27	nova-consoleauth	cloud-ctrl.nipne.ro	internal	enabled	up	2023-05-11T07:57:51.000000		
28	nova-compute	cloud-vm01-test	nova	enabled	down	2022-03-17T07:50:54.000000		
31	nova-compute	cloudcompute01	nova	enabled	down	2022-03-17T07:51:05.000000		
32	nova-compute	cloudcompute02	nova	enabled	up	2023-05-11T07:57:49.000000		
33	nova-compute	cloudcompute03	nova	enabled	up	2023-05-11T07:57:56.000000		
34	nova-compute	cloudcompute04	nova	disabled	down	2020-03-19T10:07:38.000000		
35	nova-compute	cloudcompute05	nova	disabled	down	2020-03-19T10:09:42.000000		
36	nova-compute	cloudcompute06	nova	disabled	down	2020-03-19T10:09:54.000000		
37	nova-compute	cloudcompute07	nova	disabled	down	2020-03-19T10:10:08.000000		
38	nova-compute	cloudcompute08	nova	disabled	down	2020-03-19T10:10:37.000000		

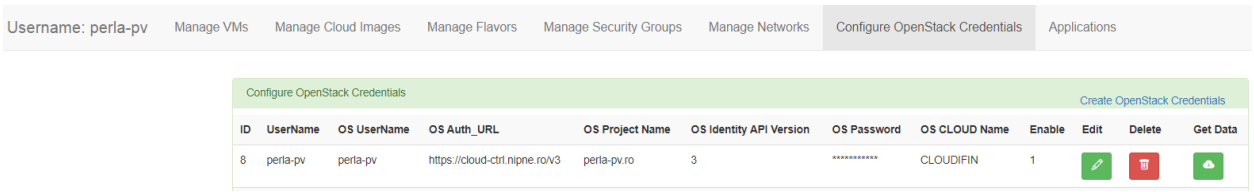
FIG. 6.2: Pagina de listare a serviciilor de calcul

6.2 Monitorizarea resurselor de catre utilizatori

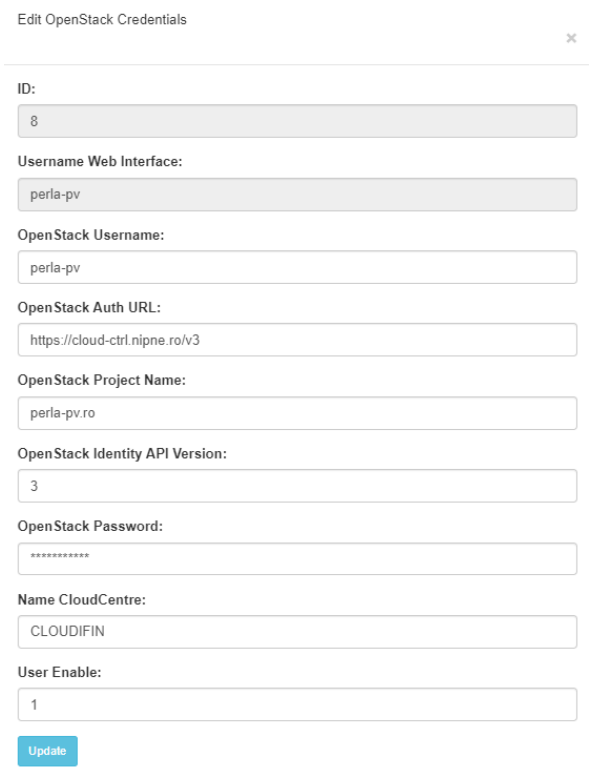
Pentru un utilizator procedura descrisa mai sus nu mai este valabila, deoarece ar necesita accesul acestuia prin protocol *ssh* la controllerul site-ului cloud, ceea ce ar constitu o problema de securitate.

Pentru a elimina operatiunile pe care ar fi trebuit sa le faca atat administratorul (crearea contului pentru utilizatorul care a fost acceptat in respectivul centru cloud) cat si utilizatorul (crearea fisierului de configurare dupa ce acesta s-a logat prin protocol *ssh* pe controller), a fost implemantata o functie care este apelata automat de sistem imediat dupa ce este acceptata cererea de creare si inrolare a utilizatorului respectiv intr-un proiect de calcul. Functia respectiva inseraza in baza de date a utilizatorilor credentialele de acces la respectivul centru cloud pentru care a optat sau a fost alocat automat.

Acest lucru poate fi vizualizat in tab-ul "*Configure OpenStack Credentials*" din contul utilizatorului. In coloana "*UserName*" este trecut numele de cont care va fi folosit pentru a accesa interfata web. In coloana "*OS UserName*" este trecut numele de utilizator care va fi folosit pt a accesa resursele centrului cloud. Aceste date sunt regasite si in e-mail-ul transmis automat solicitantului dupa ce cererea acestuia a fost acceptata.



Prin apasarea butonului "Edit" utilizatorul poate modifica credentialele de acces la resursele Cloud.



Dupa apasarea butonului „Update” credentialele de acces la resursele cloud sunt actualizate in baza de date a utilizatorilor. Operatiunile permise pentru un utilizator sunt de management al (imaginilor de) masini(lor) virtuale (creare, distrugere, suspendare VM-uri).

7. Implementarea SCF

7.1 Migrarea Infrastructrii Centrale de Management

Conform Cererii de Finantare, pentru realizarea platformei pilot a SCF, Infrastructura Centrala de Management (ICM) a fost conectata (prin mesagerie MQTT) la alte doua centre de resurse cloud din tara, site-ul cloud al Universitatii Ovidius din Constanta (implementat in cadrul proiectului POC *Ovidius Cyber Cloud* – OCC) si la cel mplementat la INCDTIM – Cluj prin proiectul POC cu codul 124698.

Atata timp cat cei doi parteneri nu au avut instalata infrastructura Cloud care va fi utilizata in regim de productie, testarea comunicarii intre ICM si client s-a facut in etapele anterioare ale proiectului utilizand site-ul CLOUDIFIN si un alt site cloud de test (*Partner01*) din IFIN-HH.

Apoi ICM a fost migrata pe echipamentele care vor fi utilizate in regim de productie: serverul *ccbd.ifin.ro*, pe care s-a instalat broker-ul MQTT Mosquitto; serverul *ccbd-db*, care gazduieste baza de date a resurselor cloud si se afla in reteaua privata; serverul de fisiere *repository* pe care sunt stocate, printre altele, imaginile de masini virtuale oferite utilizatorilor SCF.

Imaginile de masini virtuale gazduite de *repository* includ atat imaginile dezvoltate in cadrul proiectului CeCBiD pentru aplicatiile de nanofizica, de bioinformatica, sau pentru ELI-NP, cat si alte imagini care au fost dezvoltate de catre programatorii din DFCTI si care au fost incarcate in baza de date a aplicatiilor EGI AppDB (https://appdb.egi.eu).

7.2 Securizarea comunicarii de mesaje

Interfata ICM si agentii care ruleaza pe controllerele si pe nodurile centrelor de resurse cloud din SCF schimba mesaje prin intermediul brokerului MQTT. Brokerul Mosquitto filtreaza mesajele publicate pe un anumit subiect/canal (*topic*) astfel incat acestea sa fie receptionate doar de catre acei clienti care au fost in prealabil abonati (*subscribed*) la subiectul respectiv.

Comunicarea de mesaje intre ICM si centrul CLOUDIFIN a fost securizata la nivel local prin *iptables*. Deoarece pentru comunicarea de mesaje intre IFIN-HH si centrele de resurse cloud din tara se utilizeaza infrastructura de retea internet publica, au fost necesare solutii adecvate de securizare a comunicarii MQTT cu acestea.

Securizarea comunicarii MQTT intre nodurile retelei, intre ICM si site-urile cloud partenere din INCDTIM-Cluj, respectiv Universitatea Ovidius din Constanta (UOC) a fost realizata prin conexiuni dedicate punct la punct.

Intr-o prima faza, in care s-au efectuat teste de comunicare intre IFIN-HH si infrastructura cloud existenta la UOC, s-a implementat de catre IFIN-HH politica de securizare la nivel de firewall. Ulterior in SCF a fost implementata o solutie de conectare VPN punct-la-punct.

7.3 Administrarea cererilor de înregistrare

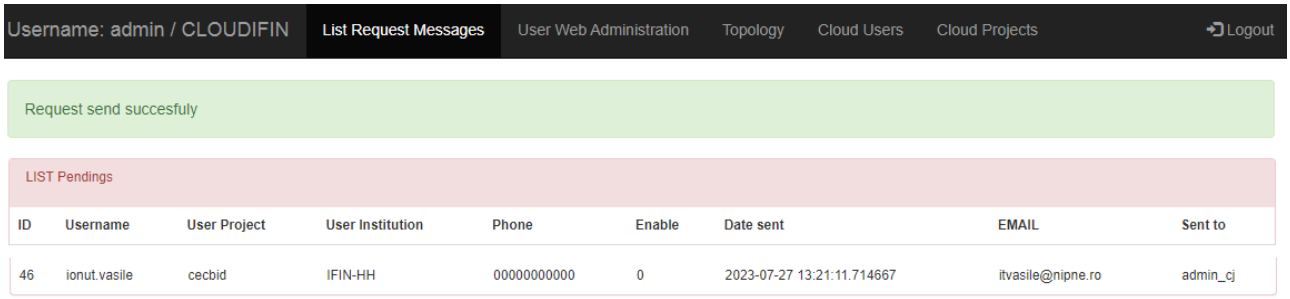
La solicitarea de înregistrare a sa și/sau a unui proiect de calcul în SCF, utilizatorul potențial completează același formular prezentat în Livrabilul 2.1, cu singura diferență că în acest caz resursele disponibile vor putea fi oferite de oricare din centrele integrate în SCF.

Mai întâi propunerea de proiect trebuie să primească avizul Comitetului de Evaluare.

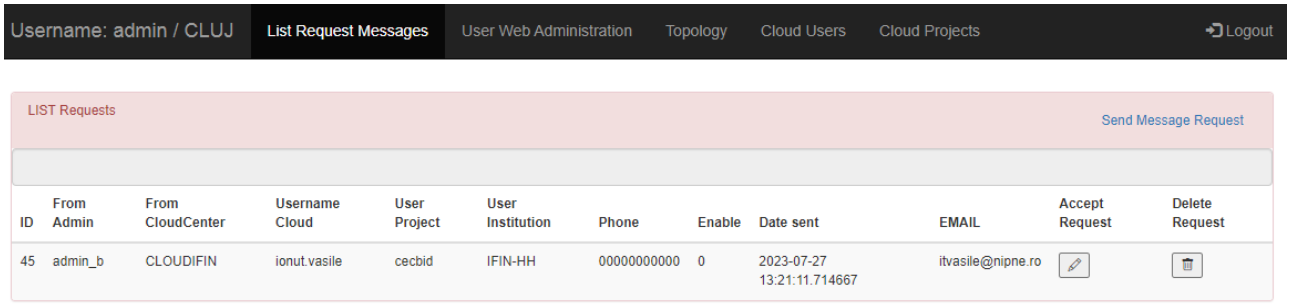
Apoi managerul SCF, care coordoneaza repartizarea resurselor cloud, trebuie sa încerce să aloce propunerea de proiect de calcul pe care a primit-o unui centru care dispune de resursele solicitate, cu acordul managementului centrului respectiv.

In acest scop, consultă mai întâi baza de date a resurselor SCF disponibile, care este actualizată în timp cuasireal de agentii ce rulează în centrele cloud. Dacă găsește un centru cu resurse disponibile adecvate, îi transmite o cerere managerului centrului respectiv privind solicitarea respectiva. Cererea se face completand un *"Request Form"* cu datele de identificare ale solicitantului, numele de utilizator atribuit (în acest exemplu "ionut.vasile"), date de contact, acronimul proiectului propus, etc.

Până la primirea răspunsului la această cerere, datele trimise vor figura la rubrica *"LIST Pendings"* din interfața de administrare a cererilor al managerului SCF (admin/CLOUDIFIN):



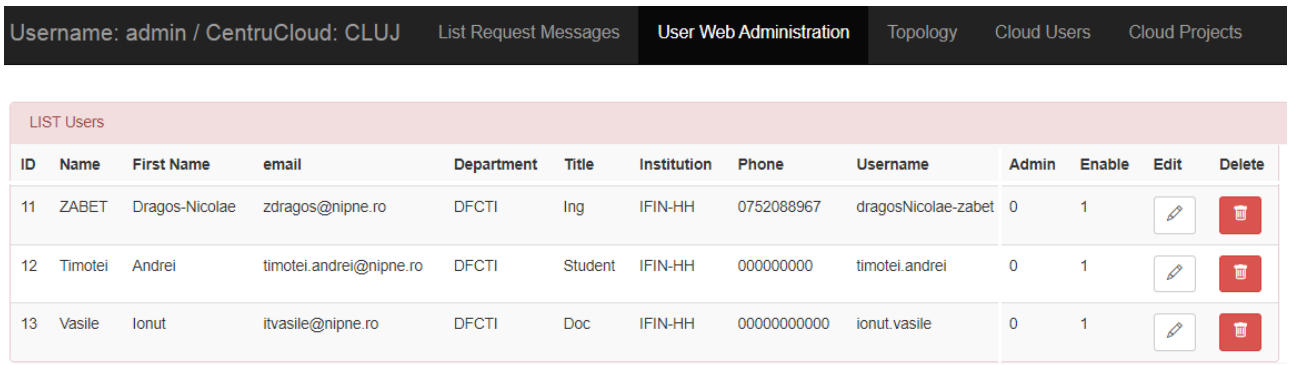
Cererea recepționată de către managerul centrului local (în acest caz identificat ca utilizatorul "admin/CLUJ" îi apare acestuia listată la rubrica *"LIST requests"* din interfața de administrare:



Admin/CLUJ consultă detaliile tehnice asociate cererii accesând baza de date a proiectelor de calcul. Dacă acceptă fără obiecții proiectul propus, apasă butonul *"Accept Request"*, iar in interfața de administrare a admin/CLOUDIFIN propunerea respectivă se mută automat de la rubrica *"LIST Pendings"* la rubrica *"LIST accepted"*.

7.4 Administrarea utilizatorilor SCF

Dupa acceptarea cererii, sistemul creaza automat cont pe platforma de calcul locală pentru solicitantul respectiv, cu datele de identificare trimise de admin/CLOUDIFIN:



În același timp, apasarea butonului *"Accept Request"* lansează un mesaj MQTT spre agentul care ruleaza pe controller-ul centrului cloud, pentru crearea unui cont de utilizator si pe OpenStack.

Pentru transmiterea sub forma de mesaj MQTT a tuturor variabilelor functiei `addUser()` asociata butonului de pe interfata web, variabilele au fost impachetate (*wrapped*) intr-un string (sir de caractere) *json* (s-a folosit functia python `json.dumps()`. String-ul json a fost creat astfel:

```
    dictionary =
{'name':name,'project':project,'domain':domain,'description':description,'email'
:email,'password':password}
    conv = json.dumps(dictionary)
    mqtt.publish("cloud/req/cloudifin/addUser", conv, 0)
```

In final se transmite pe canalul (*topic*) "`cloud/req/cloudifin/addUser`" un mesaj mqtt avand continut string-ul json "`conv`". Mesajul este transmis de pe interfata web pe agentii situati pe controller-ele centrelor cloud. Odata ce a ajuns pe agenti, mesajul este transformat sub forma de dictionar folosind functia `json.loads()`. S-a folosit aceasta metoda pentru a putea extrage cat mai facil variabilele de care este nevoie pentru construirea comenzii OpenStack de adaugare a unui utilizator.

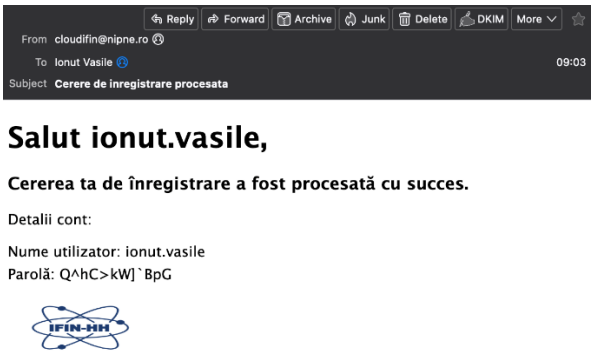
```
    dictionary = json.loads(msg.payload)
          name = dictionary['name']
          project = dictionary['project']
          domain = dictionary['domain']
          description = dictionary['description']
          email = dictionary['email']
          password = dictionary['password']
          command = ("openstack","--os-username",username,"--os-auth-
url",auth_url,"--os-project-name",project_name,"--os-identity-api-
version",identity_api,"--os-password",passwd,"user","create","--
project",project,"--password",password,name)
          p = subprocess.Popen(command, stdout=subprocess.PIPE)
```

unde "`command`" este comanda OpenStack de adaugare utilizator.

Output-ul dupa ce mesajul MQTT a fost receptionat si a fost rulata apoi comanda de creare a utilizatorului este următorul:

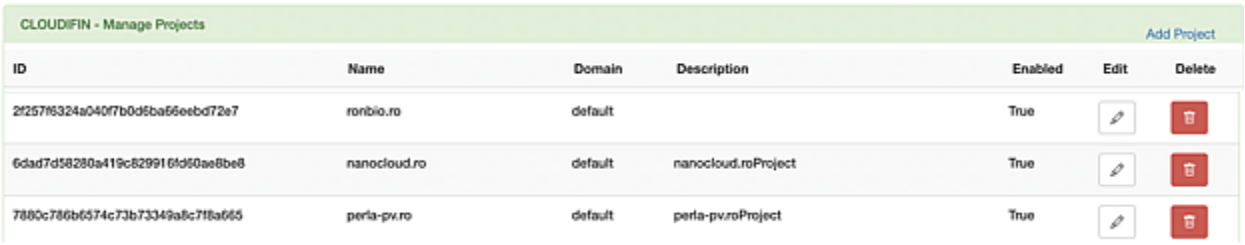
```
+---------------------+----------------------------------+
| Field               | Value                            |
+---------------------+----------------------------------+
| default_project_id  | f66088065ba4435fbd90c9ad8f4a3f9b |
| domain_id           | default                          |
| enabled             | True                             |
| id                  | 72af317ee5034a3e984c9d93612dc109 |
| name                | ionut.vasile                     |
| options             | {}                               |
| password_expires_at | None                             |
+---------------------+----------------------------------+
```

La finalizarea procedurii, solicitantul primeste un email cu datele de acces:

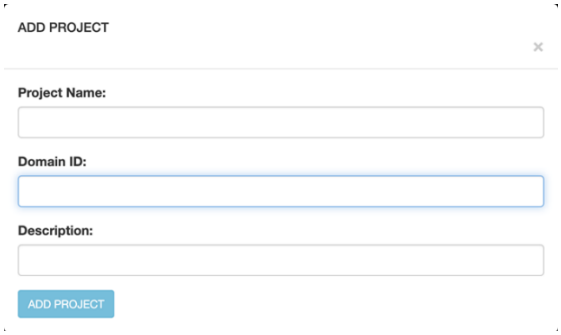


7.5 Administrarea proiectelor de calcul

Administratorul listeaza proiectele de calcul OpenStack prin intermediul interfetei web urmatoare (pentru exemplificare au fost folosite date reale):



Pentru adaugarea unui proiect de calcul nou administratorul utilizeaza link-ul "*Add Project*" si apoi completeaza formularul de mai jos. Datele din formular vor ajunge pe agentul care ruleaza pe controller. Acesta executa comanda OpenStack pentru adaugarea proiectului.



Pentru functia `addProject()` a fost creat string-ul json:

```
dictionary = {'name':name,'domain':domain,'description':description}
conv = json.dumps(dictionary)
mqtt.publish("cloud/req/cloudifin/addProject", conv, 0)
```

Mesajul mqtt avand payload-ul string-ul json "`conv`" si topic-ul "`cloud/req/cloudifin/addProjectr`" este transmis de interfata web catre agent. Ajuns pe agent, mesajul este transformat ca si in cazul functiei `addUser()` intr-un dictionar.

```
        dictionary = json.loads(msg.payload)
        name = dictionary['name']
        domain = dictionary['domain']
        description = dictionary['description']
        command = ("openstack","--os-username",username,"--os-auth-
url",auth_url,"--os-project-name",project_name,"--os-identity-api-
version",identity_api,"--os-password",passwd,"project","create","--
description",description,name,"--domain",domain)
        p = subprocess.Popen(command, stdout=subprocess.PIPE)
```

unde "`command`" este comanda OpenStack de adaugare proiect

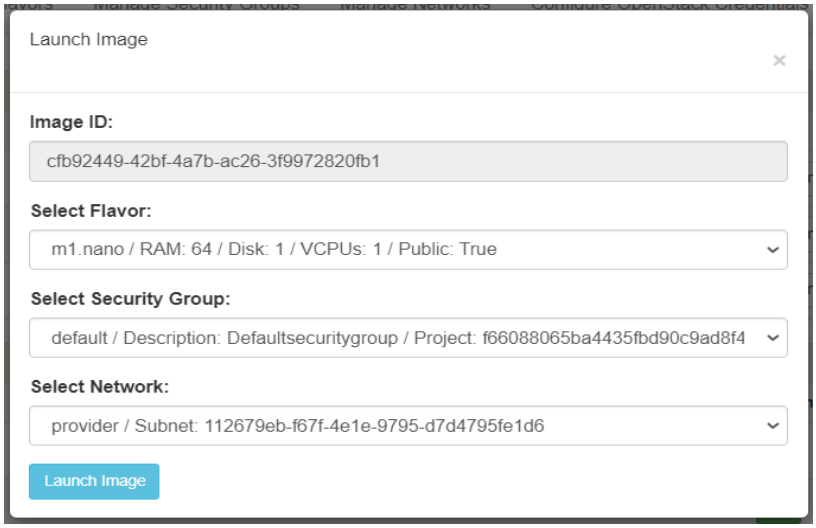
7.6 Administrarea resurselor de către utilizator

A fost programata interfata grafica a utilizatorului astfel incat dupa logare acesta sa poata vizualiza lista de imagini cu sisteme de operare la care are acces, si sa le poata utiliza la pornirea masinilor virtuale.

Toate informatiile legate de imagini (SO, flavor, security group, retele de comunicatie), sunt preluate din OpenStack in urma unor schimburi de mesaje MQTT.

Pentru exemplificare, interfata este reprodusa mai jos cu date reale:

Utilizatorul poate initializa VM-uri prin apasarea butonului "*Launch*". Dupa apasarea butonului respectiv se deschide o fereastra de unde acesta trebuie sa-si aleaga *Flavor*-ul, *Security Group*-ul si reteaua de comunicatie.



Dupa ce este apasat butonul "*Launch Image*" sistemul initializează mașina virtuală solicitată.

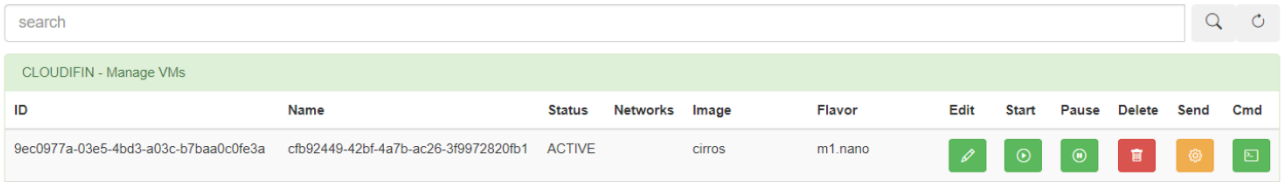
Se pot vizualiza informații privind mașina virtuală cu comenzile `openstack server list`

```
+--------------------------------------+--------------------------------------+--------+----------+----------------+-------------------+
| ID                                   | Name                                 | Status | Networks | Image          | Flavor            |
+--------------------------------------+--------------------------------------+--------+----------+----------------+-------------------+
| 9ec0977a-03e5-4bd3-a03c-b7baa0c0fe3a | cfb92449-42bf-4a7b-ac26-3f9972820fb1 | ACTIVE |          | cirros         | m1.nano           |
+--------------------------------------+--------------------------------------+--------+----------+----------------+-------------------+
```

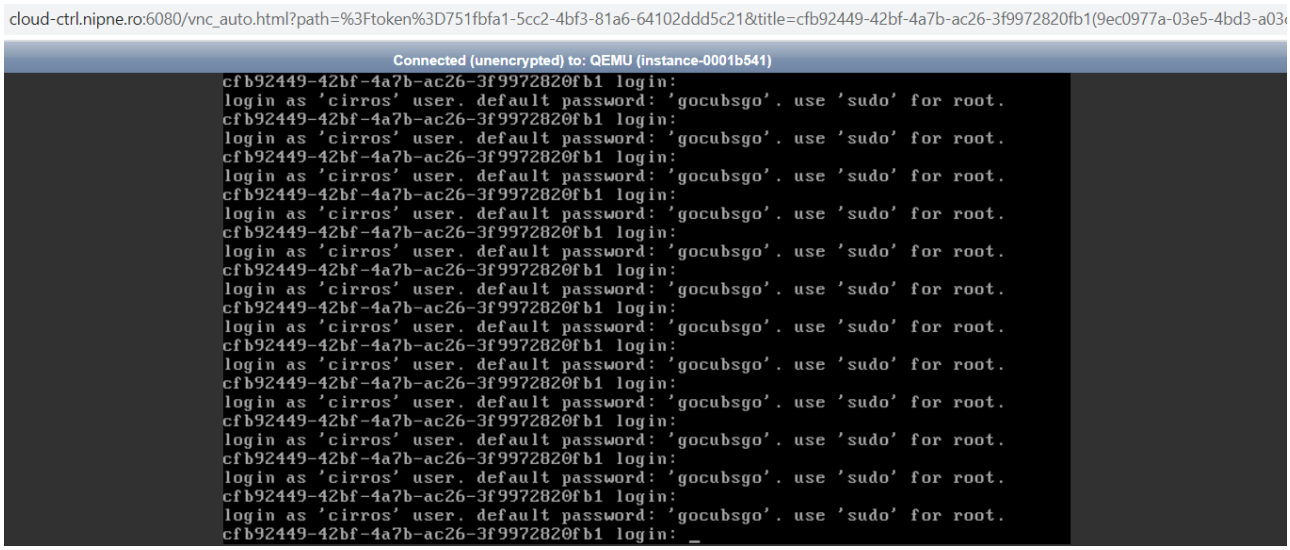
respectiv `openstack server show cfb92449-42bf-4a7b-ac26-3f9972820fb1`

```
+-----------------------------+------------------------------------------------------------+
| Field                       | Value                                                      |
+-----------------------------+------------------------------------------------------------+
| OS-DCF:diskConfig           | MANUAL                                                     |
| OS-EXT-AZ:availability_zone | nova                                                       |
| OS-EXT-STS:power_state      | Running                                                    |
| OS-EXT-STS:task_state       | None                                                       |
| OS-EXT-STS:vm_state         | active                                                     |
| OS-SRV-USG:launched_at      | 2023-07-26T09:04:44.000000                                 |
| OS-SRV-USG:terminated_at    | None                                                       |
| accessIPv4                  |                                                            |
| accessIPv6                  |                                                            |
| addresses                   |                                                            |
| config_drive                |                                                            |
| created                     | 2023-07-26T09:04:35Z                                       |
| flavor                      | m1.nano (0)                                                |
| hostId                      | 4e5255e23bf6d327dd657de4abf86d84615f6a6f1dc22c0b85ee6ef3   |
| id                          | 9ec0977a-03e5-4bd3-a03c-b7baa0c0fe3a                       |
| image                       | cirros (cfb92449-42bf-4a7b-ac26-3f9972820fb1)              |
| key_name                    | None                                                       |
| name                        | cfb92449-42bf-4a7b-ac26-3f9972820fb1                       |
| progress                    | 0                                                          |
| project_id                  | f66088065ba4435fbd90c9ad8f4a3f9b                           |
| properties                  |                                                            |
| status                      | ACTIVE                                                     |
| updated                     | 2023-07-26T09:04:44Z                                       |
| user_id                     | 26a49449216745849820d97021c294d08                          |
| volumes_attached            |                                                            |
+-----------------------------+------------------------------------------------------------+
```

Interfața grafică a utilizatorului îi permite acestuia administrarea mașinilor virtuale.



Accesul la mașinile virtuale respective se realizează prin *Virtual Network Computing* (VNC), utilizând un browser web:



8. Serviciu de inventariere si administrare a pachetelor software

8.1 Obiectiv

Scopul principal al acestui serviciu este de a-i asigura administratorului unei retele de calcul dezvoltata pe infrastructura OpenStack o interfata web care sa-i permita acestuia atat afisarea unor liste cu pachetele software instalate pe instantele de calcul virtualizate, cat si posibilitatea de actualizare a fiecare pachet.

In timp ce serviciul ii ofera administratorului posibilitatea de vizualizare + verificare + actualizare a versiunii software, utilizatorul fara drept de administrare are acces doar la functionalitatea de vizualizare.

In acest capitol, in cazul in care nu este precizat altfel, termenul de utilizator (sau user) se va referi la am-bele situatii (administrator/simplu utilizator).

Software-ul de baza utilizat pentru dezvoltarea sistemului a fost modulul Flask [2], care permite realizarea de aplicatii web utilizand limbajul Python, ceea ce ofera compatibilitate atat intre diferite arhitecturi hardware dar si sisteme de operare. Aplicatia principala programata in cadrul proiectului ofera mai multe functionalitati, printre care managementul de pachete de pe resursele de calcul, care va fi descris in cele ce urmeaza. Aceasta va rula in permanenta pe un server principal, adica pe un nod de calcul deschis pen-tru accesul web public (in continuare se va utiliza doar denumirea de server pentru a se face referire la nodul pe care este dezvoltat serviciul web Flask).

Infrastructura de vizualizare a pachetelor de pe VM-urile disponibile contine atat o componenta backend (modul de executie al functiilor dezvoltate in Python) dar si o componenta frontend (interfata grafica efectiva pe care utilizatorul o acceseaza). La fiecare conectare se vor putea vizualiza informatii care au fost create sau procesate in partea de backend, prin executarea unui grup de functii si module Python.

8.2 Componenta backend

Toate informatiile privind masinile virtuale disponibile sunt preluate din tabela '*Server*' a bazei de date a resurselor, *DBres*, descrisa in Sectiunea 3.

La accesarea paginii de management de pachete din cadrul aplicatiei principale Flask, utilizatorul va avea la dispozitie un buton prin apasarea caruia poate vizualiza intreaga lista de VM-uri (prezentata in Sectiunea 3). Aceasta lista se obtine prin interogarea tabelei ‚*Server*', proces realizat cu ajutorul functiei `refresh_instances()`.

Comunicarea dintre client (partea de interfata grafica) si server (functionalitatea cu modulele Pyhton) se realizeaza prin schimbul constant de evenimente asigurat de serviciul `socketIO`. Acest modul este necesar atat pentru aplicatia Flask:

```
from flask_socketio import SocketIO
from flask_socketio import emit
from flask import Flask, render_template
```

cat si pentru interfata web (programata in HTML + JavaScript + CSS):

```
<!-- use socketIO -->
<script
src="https://cdnjs.cloudflare.com/ajax/libs/socket.io/4.0.1/socket.io.js"
crossorigin="anonymous">
```

```
</script>
```

Un eveniment socketIO definit in interfata web fi configurat sa trasmita/emita o anumita comanda catre server, dar si viceversa. Astfel, prin apasarea butonului „`instance-refresher`”, un eveniment cu comanda „`refresh_instances`” va fi emis catre server:

```
$("#instance-refresher").click(() => {
console.log("Refreshing instances");
sio.emit("refresh_instances");
});
```

Serverul va raspunde acestui mesaj tot prin intermediul unui eveniment socketIO, executand un set de instructiuni in cadrul functiei `refresh_instances()`si finalizand cu transmiterea raspunsului catre client (aplicatia web):

```
@socketio.event
def refresh_instances():
    print('User requested VM list')
    active_vms = [(vm[0], vm[1]) for vm in vm_db.get_user_vms(VM_DB)]
    emit('instances', {'vms': active_vms})
```

Variabilele `vm[0]` si `vm[1]` contin ID-ul si, respectiv, numele masinilor virtuale ce sunt preluate din baza de date. Lista de VM-uri se transmite catre interfata web cu ajutorul comenzii *emit* din modulul socketIO. Odata ce lista este primita, aceasta este redata ca si continut HTML pe pagina activa. Intrucat editarea continua a sursei `.html` nu este o metoda eficienta, se va folosi pachetul jQuery [4]:

```
<!-- use jQuery -->
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>
```

Acesta este un pachet aditional limbajului JavaScript si permite accesarea doar a anumitor campuri (*tags*) din pagina HTML. In cazul afisarii masinilor virtuale, acestea vor fi dispuse sub forma unei liste selectabile (*dropdown-list*) in momentul in care instantele VM sunt primite de la serverul principal:

```
//save the instances list from the server as an array
sio.on("instances", (data) => {
  $("#vm-drp-list").empty();
  data.vms.forEach((element) => {
    let vm_name = element[1];
    let vm_id = element[0];
    $("#vm-drp-list").append(
      '<a class="dropdown-item" href="#">' + vm_name + "</a>"
    );
    vm_id_list.push(vm_id);
    vm_name_list.push(vm_name);
  });
});
```

Codul de mai sus verifica mai intai ca tabelul este gol inainte de initializare, iar variabilele `vm_id_list`, `vm_name_list` sunt utilizate pentru memorarea fiecarui ID, respectiv nume de VM, oferind posibilitatea de utilizare ulterioara a acestora. Prin comanda „append” (specifica JavaScript / jQuery) se adauga in dropdown-list numele ficarei masini virtuale, iar prin tagul HTML „<a>” elementul devine selectabil in pagina. Utilizatorul va putea selecta o VM specifica din lista de VM-uri (afisarea se face pe baza campului `Name` din baza de date initiala). Selectarea propiu-zisa a unei masini virtuale este pasul necesar pentru afisarea listei de pachete instalate pe acel sistem. Deoarece intregul set de pachete ce sunt instalate in mod standard (*by default*) pe o masina virtuala tip Linux RHEL [5] contine si multe servicii care nu sunt de interes utilizatorului, aplicatia va selecta doar acele pachete de dezvoltare (*development tools*) cu care acesta poate executa calcule si simulari fizice, matematice, etc. Codul de mai jos realizeaza selectarea unei VM din lista dropdown:

```
//actions when the user selects an item from vm-drp-list
  $("#vm-drp-list").on("click", "a", (e) => {
      selected_vm = e.target.text;
      let selected_vm_index = $("#vm-drp-list a").index(e.target) + 1;
      sio.emit("vm_selected", {
        vm_id: vm_id_list[selected_vm_index - 1],
        vm_name: vm_name_list[selected_vm_index - 1],
      });
```

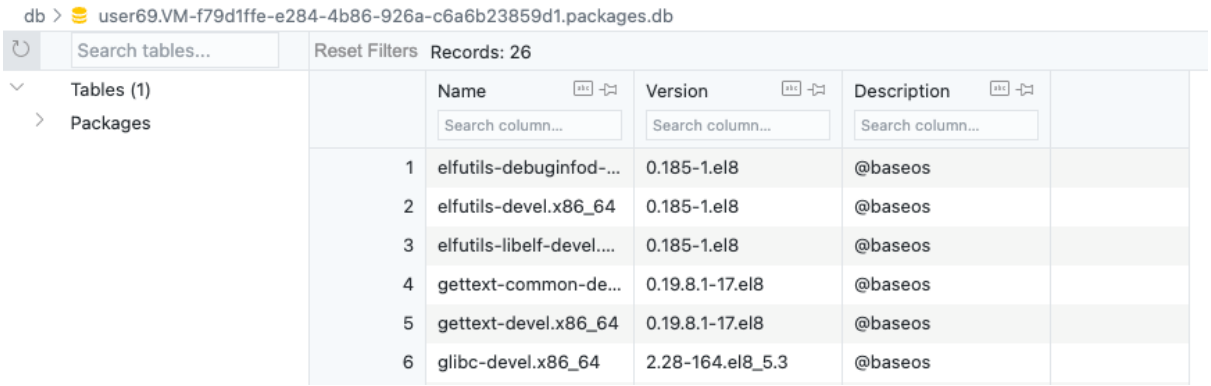
Alegerea de catre utilizator a unei anumite VM din lista se face dupa numele acestei masini, insa aplicatia identifica automat si ID-ul (prin `vm_id_list, vm_name_list`). In momentul selectarii propriu-zise a unui element din lista, un alt eveniment socketIO va fi emis catre server, prin care se va cere (*request*) informatia cu privire la pachetele masinii respective. Evenimentul „vm_selected" din codul de mai sus va trimite catre server un mesaj care va contine numele si ID-ul masinii, iar functia „sio.emit" este responsabila de crearea acestui eveniment. Ulterior, serverul va raspunde evenimentului prin functia urmatoare:

```
@socketio.event
def vm_selected(data):
    vm_id = data['vm_id']
    vm_name = data['vm_name']
    vm_packages = pack.get_vm_packages(USER_ID, vm_id) <--- sends command to VM for package retrieval
    ...
```

Odata ce serverul primeste informatiile cu privire la masina virtuala, prin intermediul protocolului (modulului) MQTT [6] va fi transmisa o comanda de tip terminal pe un canal (*topic*) unic de comunicare intre serverul principal si VM. In mod normal, un canal de comunicare mqtt are forma „`nume_canal_principal/nume_canal_secundar/...`". Comanda de terminal ce va fi executata are forma `yum list --installed | more | grep devel`, iar canalul mqtt este creat special pentru acel utilizator si acea masina virtuala.

Un exemplu test de topic mqtt pentru un utilizator denumit „user69" si pentru un VM selectat este:
`cloud/req/cloudifin/servers/user69/f79d1ffe-e284-4b86-926a-c6a6b23859d1`
cu structura (retea openstack)/(ID-ul unic al utilizatorului)/(ID-ul masinii virtuale).

In final, masina virtuala va primi un mesaj (sub forma unui strig binar ce va fi decodat in format UTF-8) care contine comanda de terminal. VM-ul va executa aceasta comanda, iar rezultatul din terminal va fi re-trimis catre serverul principal tot prin mqtt (via un topic diferit ‚res'). Serverul va stoca mesajul (dupa decodarea si parsarea acestuia) intr-un fisier de tip „.db". Structura fisierului este urmatoarea:


db > user69.VM-f79d1ffe-e284-4b86-926a-c6a6b23859d1.packages.db

Search tables... | Reset Filters | Records: 26

Tables (1)
Packages

	Name	Version	Description
	Search column...	Search column...	Search column...
1	elfutils-debuginfod-...	0.185-1.el8	@baseos
2	elfutils-devel.x86_64	0.185-1.el8	@baseos
3	elfutils-libelf-devel....	0.185-1.el8	@baseos
4	gettext-common-de...	0.19.8.1-17.el8	@baseos
5	gettext-devel.x86_64	0.19.8.1-17.el8	@baseos
6	glibc-devel.x86_64	2.28-164.el8_5.3	@baseos

Coloanele contin numele pachetului instalat, versiunea curenta, detalii suplimentare cu privire la acesta (daca este cazul). Tot acest fir de executie este implementat in functia `vm_packages = pack.get_vm_packages(USER_ID, vm_id)` din codul sursa.

Ca raspuns la cererea initiala `vm_selected` primita de la client, serverul emite un eveniment socketIO cu intreg continutul bazei de date cu pachete disponibile pe VM:

```
vm_packages = pack.get_vm_packages(USER_ID, vm_id)
emit('vm_packages', {'vm_packages': vm_packages})
```

Clientul va primi lista pachetelor si o va afisa sub forma unui tabel HTML. Implementarea pe partea de client care implementeaza aceasta actiune este reprodusa mai jos:

```
//wait for the server to send the packages on the VM that was previously selected
      sio.on("vm_packages", (data) => {
        // tuple list which contains the packages
        vm_packages = data["vm_packages"];
        $("#vm-name-title").html(vm_name_list[selected_vm_index - 1]);
        $("#vm-id-title").html(vm_id_list[selected_vm_index - 1]);
        //make the table visible
        $("#vm-table").css("display", "block");
        for (let i = 0; i < vm_packages.length; i++) {
          $("#vm-table > tbody").append(...)
```

Fiecare element din tabel este selectabil de catre utilizatori, iar administratorul va avea inca doua optiuni suplimentare:

O1. Verificarea existentei unei versiuni de software mai noua decat cea curenta
O2. Actualizarea unui pachet software la versiunea cea mai noua disponibila din biblioteca de pachete RHEL oficiala.

Cele doua operatiuni se pot executa pentru fiecare element din tabelul listei de pachete prin intermediul a doua butoane, “*Check*” si, respectiv, “*Update*”. Prin selectarea celor doua butoane de catre utilizator, doua evenimente declansatoare socketIO pot fi transmise catre serverul principal:

```
//check when the user clicks on the "check-button"
  $("#vm-table").on("click", "#check-update-button", (e) => {
    sio.emit("check_update", { vm_id: vm_id, package: package_name });
  });
  //check when the user clicks on the "update-button"
  $("#vm-table").on("click", "#update-button", (e) => {
    sio.emit("update", { vm_id: vm_id, package: package_name });
  });
```

La pasul urmator, serverul va comunica prin mqtt doua posibile comenzi de tip terminal ce vor ajunge sa fie executate pe VM respectiva:

```
def execute_check_update(userID, vm_id, package_name):
    print(f'From: {userID}')
    print(f'will check update for {package_name} on VM: {vm_id}')
    # shell command to be executed on the selected VM
    command = f'yum check-update | grep {package_name}'
    publish_command(userID, vm_id, command)
```

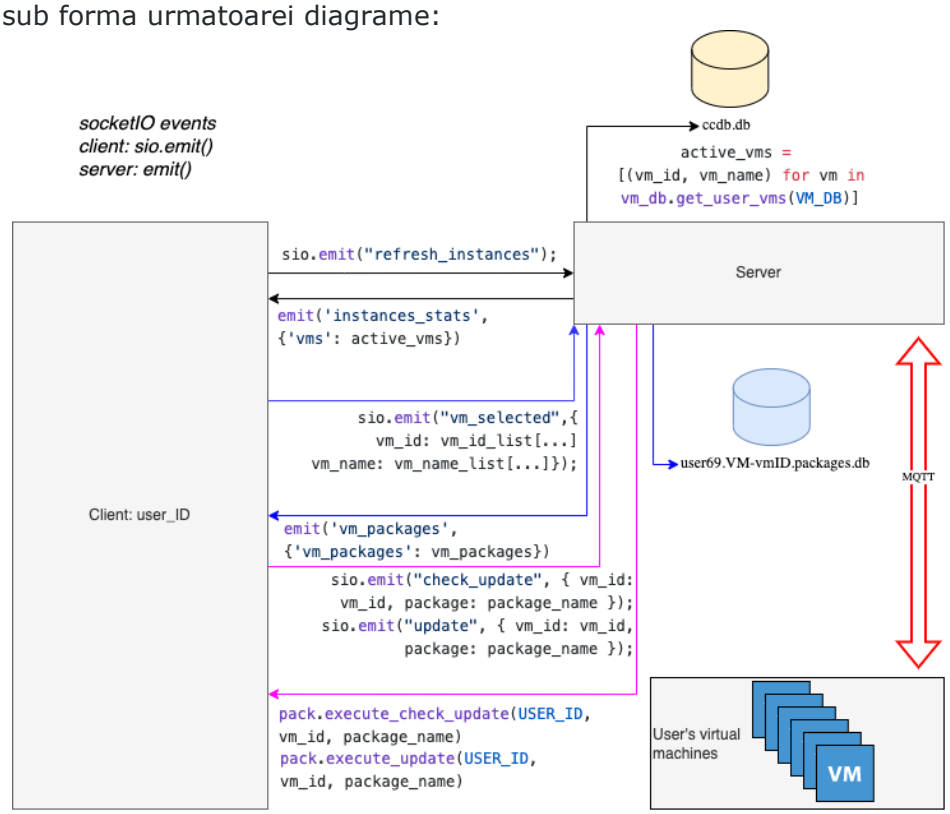
sau:

```
def execute_update(userID, vm_id, package_name):
    print(f'From: {userID}')
    print(f'will update {package_name} on VM: {vm_id}')
    # shell command to be executed on the selected VM
    command = f'yum update {package_name}'
    publish_command(userID, vm_id, command)
```

unde functia „`publish_command()`” este implementarea mqtt ce va transmite fie comanda „`f'yum check-update | grep {package_name}'`” sau „`f'yum update {package_name}'`“, spre a fi rulate pe masina virtuala.

Aceste comenzi vor verifica starea unui anume pachet sau vor porni un procedeu de actualizare tipic sistemului de operare Linux.

Intregul proces de executie si comunicare dintre client si server descrisa mai sus poate fi reprezentat sub forma urmatoarei diagrame:

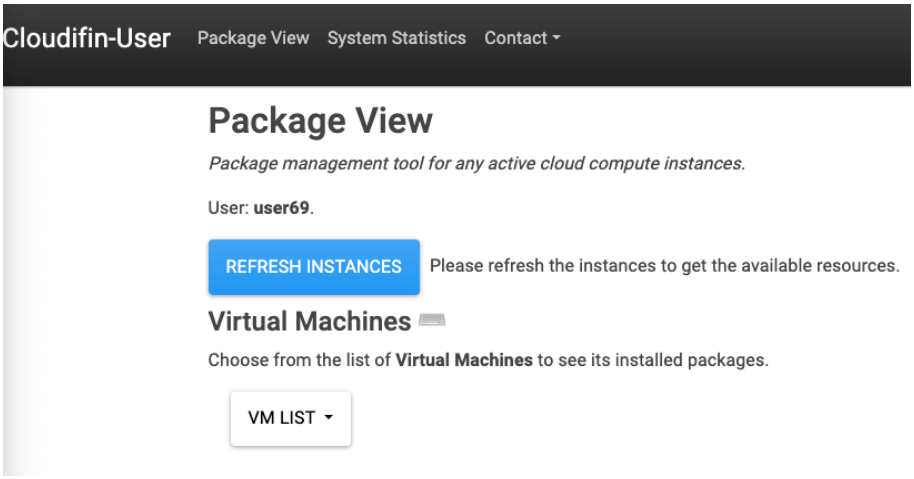


Fluxul de operatiuni descris in diagrama de mai sus contine si pasii de administrare (i.e. , verificare + actualizare pachete software), care nu sunt accesibili pentru utilizatorul fara drept de administrare. Utilizatorul obisnuit va putea accesa doar partea de listare a VM-urilor, oferite prin evenimentele „refresh_instances” (client) si „instances_stats” (server)..

8.3 Componenta frontend

Interfata grafica pe care utilizatorul o acceseaza este realizata prin intermediul aplicatiei web Flask. In continuare vor fi prezentate elementele cu care utilizatorii serviciului pot interactiona.

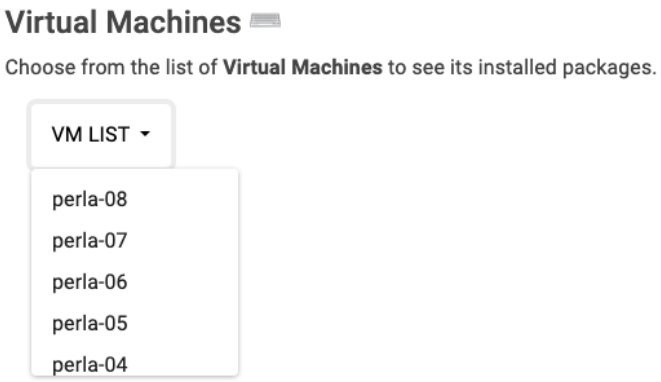
In captura de ecran de mai jos este prezentata pagina principala web *Package View* prin intermediul careia utilizatorul (*Cloudifin-User*) poate sa initiaze doar vizualizarea pachetelor software de pe masinile virtuale.



In situatia in care utilizatorul are si drept de administrator pe reteaua OpenStack (*Cloud-Admin*), pagina web initiala (*Package Management*) are un continut similar, dar permite si accesul la cele doua optiuni aditionale O1 si O2 care au fost prezentate in Sectiunea 5.1.

La accesarea paginii „*Package Management*”, se va afisa numele utilizatorului impreuna cu butonul „*Refresh Instances*”, la apasarea caruia evenimentul „`refresh_instances`” este emis de la client catre server.

Serverul va raspunde (prin evenimentul „`instances`”) cu lista de masini virtuale, iar aceasta se va vizualiza in meniul dropdown conform schemei de mai jos:

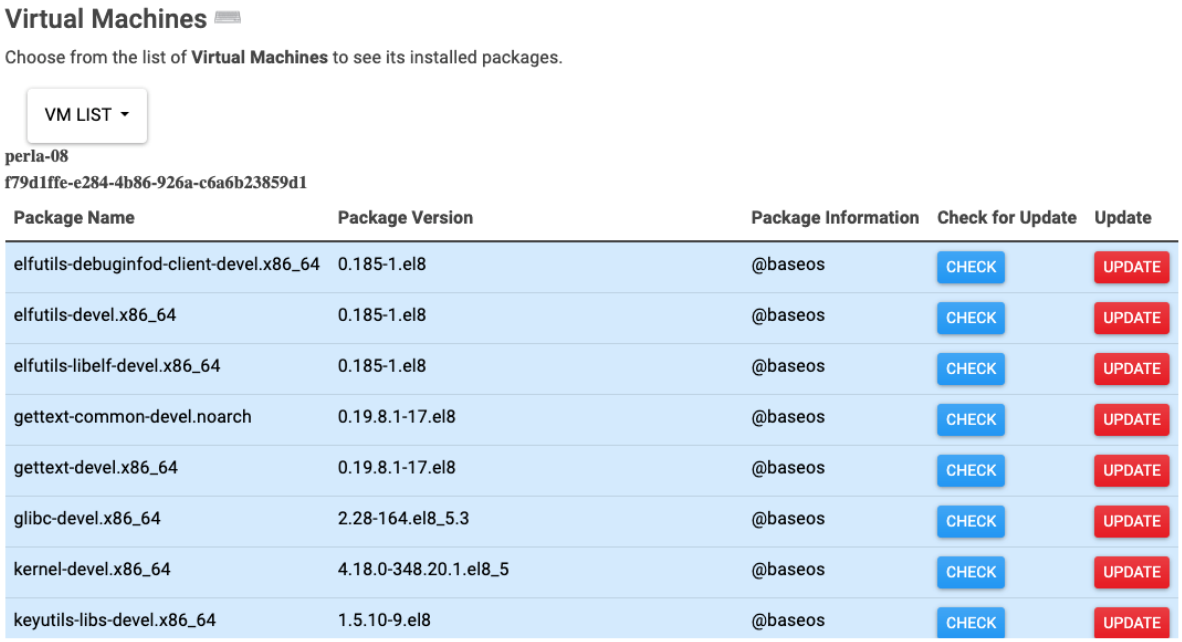


Simultan, pe terminalul serverului principal se va afisa un mesaj ca mai jos (util pentru dezvoltatorii aplicatiei web):

```
127.0.0.1 - - [26/May/2022 14:54:45] "POST /socket.io/?EIO=4&transport=polling&t=O40E1bi&sid=WysbAjGIh668gDC2AAAW HTTP/1.1" 200 -
User requested VM list
```

La selectarea unei masini virtuale din lista se va emite catre server evenimentul „`vm_selected`” ce va transmite inapoi catre client lista de pachete de pe masina corespunzatoare (via „`vm_packages`”).

De exemplu, daca administratorul selecteaza prima masina virtuala din lista de mai sus, pe interfata se afiseaza urmatoarele informatii:



Primele doua linii contin numele si ID-ul masinii virtuale selectate. In continuare, apare tabelul cu detalii si functiile de verificare/update software. Fiecare pachet dispune in mod separat de aceasta functionalitate. Un utilizator simplu va putea sa vada doar lista de pachete fara coloanele „*Check for Update*" si „*Update*".

La apasarea celor doua butoane, pe serverul principal vor aparea mesaje de tipul

```
User requested VM list
From: user69
will check update for elfutils-debuginfod-client-devel.x86_64 on VM: f79d1ffe-e284-4b86-926a-c6a6b23859d1
Will publish on the topic: cloud/req/cloudifin/servers//f79d1ffe-e284-4b86-926a-c6a6b23859d1
Connected to MQTT Broker!
```

(in cazul verificarii), sau

```
From: user69
will update elfutils-debuginfod-client-devel.x86_64 on VM: f79d1ffe-e284-4b86-926a-c6a6b23859d1
Connected to MQTT Broker!
Will publish on the topic: cloud/req/cloudifin/servers//f79d1ffe-e284-4b86-926a-c6a6b23859d1
```

(in cazul actualizarii).

Ultima linie din cele doua mesaje semnifica faptul ca o conexiune MQTT a fost creata catre masina virtuala, si ca se va publica pe canalul respectiv una dintre comenzile:

```
yum check-update | grep elfutils-debuginfod-client-devel.x86_64
yum update elfutils-debuginfod-client-devel.x86_64
```

9. Agenti

Informatiile privind resursele SCF sunt trimise catre broker-ul din ICM prin MQTT de catre agentii ce ruleaza pe site-urile cloud locale. In aceasta sectiune se prezinta codul dezvoltat in limbaj Python 2.7 pentru agentul care ruleaza pe controller-ul site-urilor locale (*'agent_controller_OCC-1.0.py'*) si fisierul de configurare *'admin-openrc'* ce trebuie editat cu credentialele site-ului cloud client. In versunea de productie clientii sunt site-urile CLOUDIFIN, OCC (administrat de Universitatea Ovidius din Constanta) si CLOUDITIM (implementat la INCDTIM-Cluj).

Pentru exemplificare este reprodus mai jos fisierul '*admin-openrc*' corespunzator site-ului OCC:

```
# 'admin-openrc'
[config]
OS_PROJECT_NAME=admin
OS_USERNAME=admin
OS_PASSWORD=***password***
OS_AUTH_URL=https://adresaControllerCloudOCC/v3
OS_IDENTITY_API_VERSION=3
```

Credentialele setate in fisier sunt utilizte doar local si nu sunt transmise catre ICM.

Variabila `'OS_AUTH_URL'` contine adresa controller-ului cloud al centrului de resurse OCC.

Valorile variabilelor din fisierul de credentiale ii sunt necesare agentului pentru a executa atat comenzi OpenStack de listare (servicii OpenStack, masini virtuale, flavor-uri, imagini de VM-uri gazduite de serviciul Glance, retele si subretele pentru VM-uri), cat si de pornire/stergere a VM-urilor.

Pentru descarcarea de pe ICM repository a imaginilor de masini virtuale prin intermediul MQTT, in codul agentului de controller au fost programate canale de mesagerie (*topics*) de tipul "`cloud/req/constanta/imageGet`".

Pentru testarea agentului intre ICM si CLOUDIFIN, in codul acestuia au fost programate initial doar comenzi de listare, iar dupa validarea agentului in testele dintre ICM si Univ. Ovidius au fost adaugate si comenzile de pornire / stergere / suspendare a masinilor virtuale.

In vederea rularii agentului trebuie instalate urmatoarele dependente software:

```
pip install paho-mqtt
pip install configparser
```

Pentru evitarea riscului de a modifica configuratia existenta, se recomanda folosirea `virtualenv`:

```
virtualenv agent-cloud
source agent-cloud/bin/activate
pip install paho-mqtt
pip install configparser
```

Agentul este lansat in executie de pe interfata ICM cu comenzile:

```
source agent-cloud/bin/activate
python agent_controller-1.0.py &
```

Codul complet al agentului este reprodus in Anexa I.

10. Concluzii

Rezultatul principal al Subactivitatii 2.2 este implementarea versiunii pilot a Sistemului Cloud Federalizat pentru cercetare (SCF) si a interfetei grafice de acces a utilizatorilor la acesta (care constituie rezultatului planificat nr. 19 al proiectului).

Infrastructura SCF, realizata prin programarea de cod original si utilizarea serviciilor OpenStack, este deocamdata singura solutie Cloud cunoscuta la nivel national care indeplineste cerinta scopului Actiunii 1.1.2 a POC privind *„dezvoltarea unor retele de centre C-D, coordonate la nivel national si racordate la retele europene..."*.

Pentru demonstrarea functionalitatii SCF, acesta a fost testat prin interconectarea centrului CLOUDIFIN din cadrul CCBD cu centrele Cloud ale INCD pentru Tehnologii Izotopice și Moleculare din Cluj-Napoca si Universitatii 'Ovidius' din Constanta.

Prin rezultatele obtinute se indeplineste Obiectivul O3 al proiectului, "*Realizarea în cadrul centrului de resurse Cloud a unei soluții tehnice capabilă să interconecteze la nivel național, în cadrul unui sistem federalizat, centrele de tip Cloud privat dezvoltate în instituții aparținând sistemului de CDI, care să ofere utilizatorilor acces printr-o interfață unica la resurse și servicii furnizate de către aceste centre.*"

Solutia tehnica dezvoltata in cadrul proiectului permite inglobarea altor centre Cloud ale entitatilor publice de CDI din tara, in perspectiva realizarii unui sistem Cloud federalizat la nivel national.

11. ANEXA I

```
"""
agent_controller-1.0.py
    -   compatibil python 2.7
    -   se executa pe controller-ul site-ului cloud
"""
import paho.mqtt.client as mqtt
import os, subprocess
import configparser
import json

# MQTT Settings
MQTT_Broker = "194.102.58.172"
MQTT_Port = 1883
Keep_Alive_Interval = 45
MQTT_Topic = "cloud/req/cloudifin/#"
hostname = os.uname()[1]

#Topic Subscribe
def on_connect(mosq, obj, flags, rc):
        mqttc.subscribe(MQTT_Topic, 0)

def on_message(mosq, obj, msg):
   if msg.topic == "cloud/req/cloudifin/project":
       if msg.payload == "req":
          config = configparser.ConfigParser()
          config.read('admin-openrc')
          username = config['config']['OS_USERNAME']
          auth_url = config['config']['OS_AUTH_URL']
          project_name = config['config']['OS_PROJECT_NAME']
          identity_api = config['config']['OS_IDENTITY_API_VERSION']
          passwd = config['config']['OS_PASSWORD']
          command = ("openstack","--os-username",username,"--os-auth-url",auth_url,"--os-
project-name",project_name,"--os-identity-api-version",identity_api,"--os-
password",passwd,"project", "list", "--long")
          p = subprocess.Popen(command, stdout=subprocess.PIPE)
          stdout, stderr = p.communicate()
          mqttc.publish("cloud/res/cloudifin/project", stdout)

    if msg.topic == "cloud/req/cloudifin/user":
       if msg.payload == "req":
          config = configparser.ConfigParser()
          config.read('admin-openrc')
          username = config['config']['OS_USERNAME']
          auth_url = config['config']['OS_AUTH_URL']
          project_name = config['config']['OS_PROJECT_NAME']
          identity_api = config['config']['OS_IDENTITY_API_VERSION']
          passwd = config['config']['OS_PASSWORD']
          command = ("openstack","--os-username",username,"--os-auth-url",auth_url,"--os-
project-name",project_name,"--os-identity-api-version",identity_api,"--os-
password",passwd,"user", "list","--long")
          p = subprocess.Popen(command, stdout=subprocess.PIPE)
          stdout, stderr = p.communicate()
          mqttc.publish("cloud/res/cloudifin/user", stdout)

    if msg.topic == "cloud/req/cloudifin/topologie":
       if msg.payload == "req":
          config = configparser.ConfigParser()
          config.read('admin-openrc')
          username = config['config']['OS_USERNAME']
          auth_url = config['config']['OS_AUTH_URL']
          project_name = config['config']['OS_PROJECT_NAME']
          identity_api = config['config']['OS_IDENTITY_API_VERSION']
          passwd = config['config']['OS_PASSWORD']
```

```
          command = ("openstack","--os-username",username,"--os-auth-url",auth_url,"--os-
project-name",project_name,"--os-identity-api-version",identity_api,"--os-
password",passwd,"compute", "service", "list")
          p = subprocess.Popen(command, stdout=subprocess.PIPE)
          stdout, stderr = p.communicate()
          mqttc.publish("cloud/res/cloudifin/topologie", stdout)

    if msg.topic == "cloud/req/cloudifin/flavor":
       if msg.payload == "req":
          config = configparser.ConfigParser()
          config.read('admin-openrc')
          username = config['config']['OS_USERNAME']
          auth_url = config['config']['OS_AUTH_URL']
          project_name = config['config']['OS_PROJECT_NAME']
          identity_api = config['config']['OS_IDENTITY_API_VERSION']
          passwd = config['config']['OS_PASSWORD']
          command = ("openstack","--os-username",username,"--os-auth-url",auth_url,"--os-
project-name",project_name,"--os-identity-api-version",identity_api,"--os-
password",passwd, "flavor", "list")
          p = subprocess.Popen(command, stdout=subprocess.PIPE)
          stdout, stderr = p.communicate()
          mqttc.publish("cloud/res/cloudifin/flavor", stdout)

if msg.topic == "cloud/req/cloudifin/addUser":
          config = configparser.ConfigParser()
          config.read('admin-openrc')
          username = config['config']['OS_USERNAME']
          auth_url = config['config']['OS_AUTH_URL']
          project_name = config['config']['OS_PROJECT_NAME']
          identity_api = config['config']['OS_IDENTITY_API_VERSION']
          passwd = config['config']['OS_PASSWORD']

          print(msg.payload)
          dictionary = json.loads(msg.payload)
          name = dictionary['name']
          project = dictionary['project']
          domain = dictionary['domain']
          description = dictionary['description']
          email = dictionary['email']
          password = dictionary['password']
          command = ("openstack","--os-username",username,"--os-auth-url",auth_url,"--os-
project-name",project_name,"--os-identity-api-version",identity_api,"--os-
password",passwd,"user","create","--project",project,"--password",password,name)
          p = subprocess.Popen(command, stdout=subprocess.PIPE)
          stdout, stderr = p.communicate()


    if msg.topic == "cloud/req/cloudifin/addProject":
          config = configparser.ConfigParser()
          config.read('admin-openrc')
          username = config['config']['OS_USERNAME']
          auth_url = config['config']['OS_AUTH_URL']
          project_name = config['config']['OS_PROJECT_NAME']
          identity_api = config['config']['OS_IDENTITY_API_VERSION']
          passwd = config['config']['OS_PASSWORD']

          print(msg.payload)
          dictionary = json.loads(msg.payload)
          name = dictionary['name']
          domain = dictionary['domain']
          description = dictionary['description']
          command = ("openstack","--os-username",username,"--os-auth-url",auth_url,"--os-
project-name",project_name,"--os-identity-api-version",identity_api,"--os-
password",passwd,"project","create","--description",description,name,"--domain",domain)
          p = subprocess.Popen(command, stdout=subprocess.PIPE)
          stdout, stderr = p.communicate()
```

```
    if msg.topic == "cloud/req/cloudifin/image":
       if msg.payload == "req":
          config = configparser.ConfigParser()
          config.read('admin-openrc')
          username = config['config']['OS_USERNAME']
          auth_url = config['config']['OS_AUTH_URL']
          project_name = config['config']['OS_PROJECT_NAME']
          identity_api = config['config']['OS_IDENTITY_API_VERSION']
          passwd = config['config']['OS_PASSWORD']
          command = ("openstack","--os-username",username,"--os-auth-url",auth_url,"--os-
project-name",project_name,"--os-identity-api-version",identity_api,"--os-
password",passwd, "image", "list")
          p = subprocess.Popen(command, stdout=subprocess.PIPE)
          stdout, stderr = p.communicate()
          mqttc.publish("cloud/res/cloudifin/image", stdout)

    if msg.topic == "cloud/req/cloudifin/security":
       if msg.payload == "req":
          config = configparser.ConfigParser()
          config.read('admin-openrc')
          username = config['config']['OS_USERNAME']
          auth_url = config['config']['OS_AUTH_URL']
          project_name = config['config']['OS_PROJECT_NAME']
          identity_api = config['config']['OS_IDENTITY_API_VERSION']
          passwd = config['config']['OS_PASSWORD']
          command = ("openstack","--os-username",username,"--os-auth-url",auth_url,"--os-
project-name",project_name,"--os-identity-api-version",identity_api,"--os-
password",passwd, "security", "group", "list")
          p = subprocess.Popen(command, stdout=subprocess.PIPE)
          stdout, stderr = p.communicate()
          mqttc.publish("cloud/res/cloudifin/security", stdout)

    if msg.topic == "cloud/req/cloudifin/network":
       if msg.payload == "req":
          config = configparser.ConfigParser()
          config.read('admin-openrc')
          username = config['config']['OS_USERNAME']
          auth_url = config['config']['OS_AUTH_URL']
          project_name = config['config']['OS_PROJECT_NAME']
          identity_api = config['config']['OS_IDENTITY_API_VERSION']
          passwd = config['config']['OS_PASSWORD']
          command = ("openstack","--os-username",username,"--os-auth-url",auth_url,"--os-
project-name",project_name,"--os-identity-api-version",identity_api,"--os-
password",passwd, "network", "list")
          p = subprocess.Popen(command, stdout=subprocess.PIPE)
          stdout, stderr = p.communicate()
          mqttc.publish("cloud/res/cloudifin/network", stdout)

    if msg.topic == "cloud/req/cloudifin/server":
       if msg.payload == "req":
          config = configparser.ConfigParser()
          config.read('admin-openrc')
          username = config['config']['OS_USERNAME']
          auth_url = config['config']['OS_AUTH_URL']
          project_name = config['config']['OS_PROJECT_NAME']
          identity_api = config['config']['OS_IDENTITY_API_VERSION']
          passwd = config['config']['OS_PASSWORD']
          command = ("openstack","--os-username",username,"--os-auth-url",auth_url,"--os-
project-name",project_name,"--os-identity-api-version",identity_api,"--os-
password",passwd, "server", "list", "--project", "admin")
          p = subprocess.Popen(command, stdout=subprocess.PIPE)
          stdout, stderr = p.communicate()
          mqttc.publish("cloud/res/cloudifin/server", stdout)

    if msg.topic == "cloud/req/cloudifin/imageGet":
       if not os.path.exists("glanceImage"):
          os.mkdir("glanceImage")
```

```
        fileName = msg.payload
        print(fileName)
        x     =     subprocess.Popen(['wget',     fileName,     '-P',     'glanceImage'],
stdout=subprocess.PIPE)
        output, err = x.communicate()

    if msg.topic == "cloud/req/cloudifin/importGlance":
        if not os.path.exists("glanceImage"):
            os.mkdir("glanceImage")
        fileName = msg.payload
        print(fileName)
        x     =     subprocess.Popen(['wget',     fileName,     '-P',     'glanceImage'],
stdout=subprocess.PIPE)
        output, err = x.communicate()

    if msg.topic == "cloud/req/cloudifin/admin":
        if msg.payload == "list-server":
            config = configparser.ConfigParser()
            config.read('admin-openrc')
            username = config['config']['OS_USERNAME']
            auth_url = config['config']['OS_AUTH_URL']
            project_name = config['config']['OS_PROJECT_NAME']
            identity_api = config['config']['OS_IDENTITY_API_VERSION']
            passwd = config['config']['OS_PASSWORD']
            command = ['openstack','--os-username',username,'--os-auth-url',auth_url,'--os-
project-name',project_name,'--os-identity-api-version',identity_api,'--os-
password',passwd,'server', 'list']
            p = subprocess.Popen(command, stdout=subprocess.PIPE)
            stdout, stderr = p.communicate()
            mqttc.publish("cloud/res/cloudifin/admin", stdout)

def on_subscribe(mosq, obj, mid, granted_qos):
    pass

mqttc = mqtt.Client()

# Assign event callbacks
mqttc.on_message = on_message
mqttc.on_connect = on_connect
mqttc.on_subscribe = on_subscribe

# Connect
mqttc.connect(MQTT_Broker, int(MQTT_Port), int(Keep_Alive_Interval))

# Continue the network loop
mqttc.loop_forever()
```

12. ANEXA II

MANUALUL UTILIZATORULUI SISTEMULUI CLOUD FEDERALIZAT

1. INTRODUCERE

Sistemul Cloud Federalizat (SCF), coordonat de catre *Centrul de resurse Cloud si Big Data* (CCBD), oferă utilizatorilor servicii de calcul, instrumente de dezvoltare software, servicii de stocare a datelor, mașini virtuale personalizate, interconectare avansată în rețea și sisteme de operare preconfigurate cu software științific personalizat. Serviciile sunt furnizate de site-ul CLOUDIFIN, care este gazduit de CCBD, si de site-urile Cloud ale institutiilor partenere in SCF.

CCBD a fost implementat in cadrul proiectului POC „*Centru Cloud și Big Data pentru participarea la Cloud-ul European pentru Știință Deschisă (CeCBiD-EOSC)*”, cofinanțat din Fondul European de Dezvoltare Regională (FEDR) în baza contractului încheiat cu Ministerul Educației și Cercetării în calitate de Organism Intermediar, în numele și pentru Ministerul Fondurilor Europene în calitate de Autoritate de Management.

CCBD este gazduit si administrat de Departamentul Fizica Computationala si Tehnologia Informatiei din cadrul Institutului National de Cercetare Dezvoltare pentru Fizica si Inginerie Nucleara Horia Hulubei.

2. INREGISTRAREA UTILIZATORILOR SI ALOCAREA RESURSELOR

Resursele SCF se aloca doar utilizatorilor inregistrati, pe durata realizarii unor proiecte de calcul asociate activitatii proprii de cercetare stiintifica.

Cererea de inregistrare pentru accesul la resursele si serviciile furnizate de SCF se face completând un formular electronic (AAF - *Access Application Form*) cu informatii privind: a) cerintele tehnice si aspectele stiintifice relevante ale proiectului de calcul propus; b) informatii de identificare si contact ale solicitantului si ale supervizorului acestuia.

Solicitarea este analizata către un Comitet de evaluare, care va lua deciziile adecvate tinand cont de valoarea stiintifica a proiectului, de cerintele tehnice, de resursele solicitate si disponibilitatea acestora in cadrul SCF.

Pe parcursul evaluarii, Comitetul de evaluare poate solicita informatii suplimentare despre diferite aspecte ale solicitarii.

Solicitarea poate fi acceptata asa cum a fost incarcata, acceptata cu anumite amendamente, sau respinsa, iar rezultatul evaluarii este comunicat solicitantului prin email.

Daca solicitarea este acceptata si utilizatorul este de acord cu conditiile impuse de Comitetul de evaluare, acesta trebuie sa accepte Politica de Confidentialitate si Conditiile de utilizare, inaine de a avea accesul garantat la resursele si serviciile furnizate de SCF pe durata desfasurarii proiectului. Solicitantul va fi inregistrat ca utilizator si i se va comunica prin email parola initiala de acces la cont.

3. INTERFATA DE ACCES LA SCF

Pentru acces se va folosi interfața web multifuncțională de la adresa https://ccbd.ifin.ro/, care este utilă în oricare dintre următoarele cazuri posibile:

1) accesarea resurselor prin nume de user/parolă de catre un utilizator deja înregistrat;
2) solicitarea de înregistrare ca utilizator împreună cu o propunere de proiect de calcul;

3) accesarea cu user/parolă de către un utilizator înregistrat pentru a propune un nou proiect de calcul;
4) solicitarea de înregistrare a unui nou utilizator pe un proiect de calcul existent;
5) solicitarea de acces în cazul pierderii parolei (*password recovery*).

Este exclusă posibilitatea înregistrării unui utilizator fără să existe cel puțin un proiect de calcul asociat acestuia.

Fig. 1: Pagina de acces a utilizatorilor

Actiunile pe care vizitatorii paginii trebuie sa le efectueze pentru a obtine rezultatele prezentate mai sus sunt descrise in continuare:

1) *accesarea resurselor unui proiect de calcul in curs* se face prin completarea campurilor "*Utilizator*" (de regula adresa e-mail de contact a acestuia) si "*Parola*", **dacă** este utilizator inregistrat;
2) *înregistrarea ca utilizator si propunerea unui nou proiect de calcul* se face prin selectarea butonului "*Proiect și utilizator nou*", **dacă** solicitantul nu a fost inregistrat;
3) *Înregistrarea unui nou proiect de calcul*, prin completarea campurilor "*Utilizator*", "*Parola*" si selectarea casetei "*Proiect nou*", **dacă** solicitantul este user inregistrat;

4) *înregistrarea ca utilizator pe un proiect de calcul existent* (prin selectarea link-ului "*Utilizator nou în proiect"*)*;*

5) *atribuirea unei noi parole* in cazul pierderii parolei, prin selectarea link-ului "*Recupereaza parola*"*.*

In cazurile 1 si 3, inainte de a da click pe butonul "*Trimite*", vizitatorul trebuie sa bifeze casuteele "*Am citit si accept Conditiile de Utilizare / Politica de Confidentialitate* ", dupa ce a citit continutul documentelor respective.

Interfata de acces furnizeaza, de asemenea, link-uri catre Politica de Acces, Manualul Utilizatorului, Conditii de Utilizare si Politica de Confidentialitate.

4. INREGISTRAREA UTILIZATORILOR SI A NOILOR PROIECTE DE CALCUL

Dupa ce a fost selectata optiunea „*Proiect si utilizator nou*" din Fig. 1, utilizatorul este directionat catre formularul prezentat in Fig. 2 de mai jos.

Campurile formularului sunt grupate in patru categorii de informatii:

1. Informatii Personale

Completand aceste campuri solicitantul se identifica si furnizeaza datele de contact.

2. Informatii Supervizor

Solicitantul introduce informatii despre coordonatorul proiectului de calcul pentru care a solicitat inregistrarea (coordonatorul care poate fi solicitantul sau alta persoana).

3. Detalii proiect

Se specifica titlul proiectului de calcul propus, aria de cercetare si perioada propusa pentru derularea acestuia.

Documentul continand descrierea stiintifica si detaliile tehnice ale proiectului, in format PDF, se incarca in baza de date folosind butonul "*Browse*".

4. Resursele solicitate

In aceasta sectiune a formularului solicitantul specifica natura si capacitatea resurselor de calcul necesare pentru derularea proiectului propus:

- numarul total de core-uri CPU,
- timpul estimat de utilizare al CPU si timpul total de utilizare a infrastructurii;
- memoria RAM necesara;
- capacitatea totala de stocare pe termen lung;
- biblioteci si aplicatii software necesare;
- limbajele de programare care vor fi utilizate.

Solicitarile suplimentare, de exemplu privind accesul la resurse GPGPU, por fi specificate intr-un camp text la rubrica "Alte cerinte specifice".

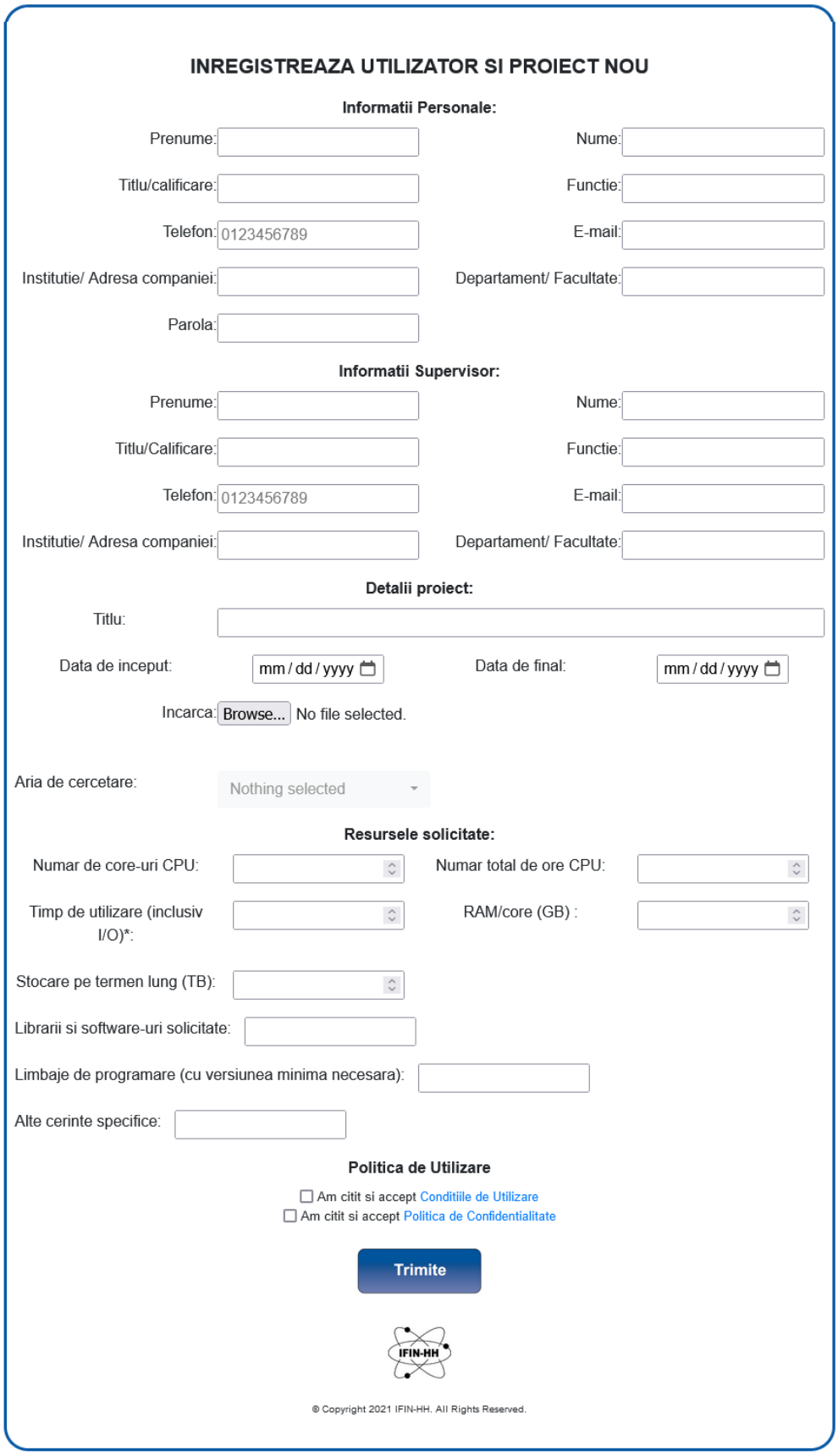


Fig. 2: Pagina/formularul de înregistrare a unui nou utilizator și a proiectului propus

5. INREGISTRAREA UNUI NOU PROIECT DE CATRE UN UTILIZATOR EXISTENT

In cazul in care utilizatorul are deja un cont creat anterior in cadrul unui proiect de calcul si doreste sa mai inregistreze un proiect, acesta completeaza datele de acces (“*Utilizator*” si “*Parola*”), si bifeaza caseta “*Proiect nou*” de pe pagina de acces.

Utilizatorul va fi directionat catre formularul de inregistrare a unui proiect nou, prezentat in Fig. 3.

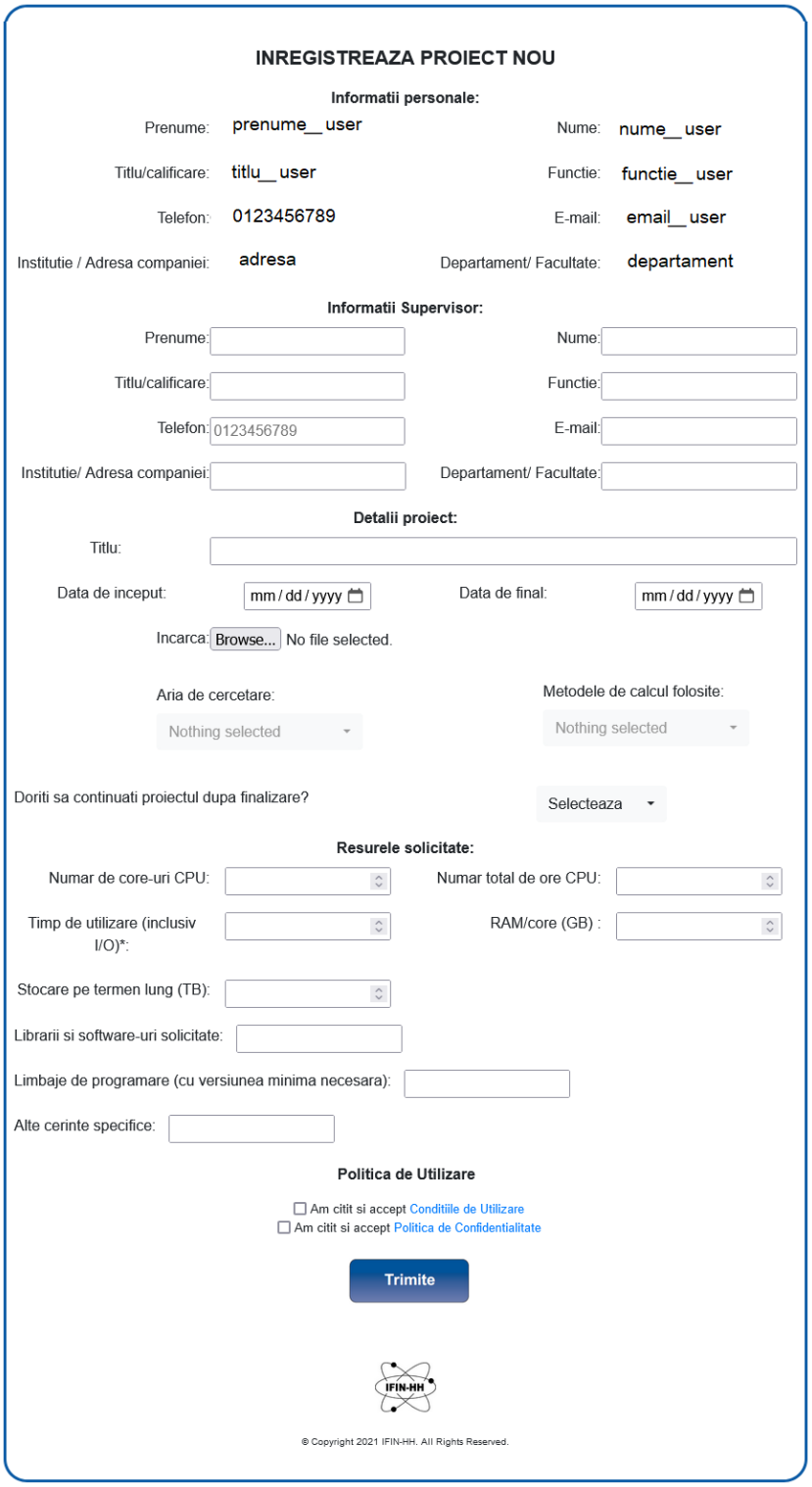


Fig. 3: Pagina de înregistrare a unui nou proiect de către un utilizator existent

Campurile din rubrica "*Informatii personale*" ale formularului sunt precompletate cu datele personale ale utilizatorului autentificat si sunt inactive.

Daca utilizatorul doreste modificarea informatiilor personale, acesta trebuie sa se logheze in contul sau.

Pentru toate formularele, datele completate sunt preluate de sistem doar daca utilizatorul bifeaza casetele prin care confirma ca a citit si accepta conditiile de utilizare a sistemului si, respectiv, politica de confidentialitate, care sunt publicate pe site-ul web.

6. INREGISTRAREA UNUI NOU UTILIZATOR PE UN PROIECT EXISTENT

In timp ce situatia ca un utilizator sa depuna mai multe proiecte a fost abordata in sectiunea 5, in aceasta sectiune este abordata situatia in care mai multi utilizatori se inscriu in acelasi proiect.

Se presupune ca a fost deja inregistrat un proiect de catre un alt utilizator si acum, din diferite motive, un alt user ar vrea sa acceseze resursele alocate proiectului.

Deoarece informatiile despre proiect sunt deja stocate in baza de date, mai sunt necesare doar urmatoarele informatii:

a) datele personale si de contact ale utilizatorului;

b) acordul directorului de proiect pentru a permite inregistrarea utilizatorului in echipa proiectului.

Interfata de inregistrare este orezentat in Fig.4.

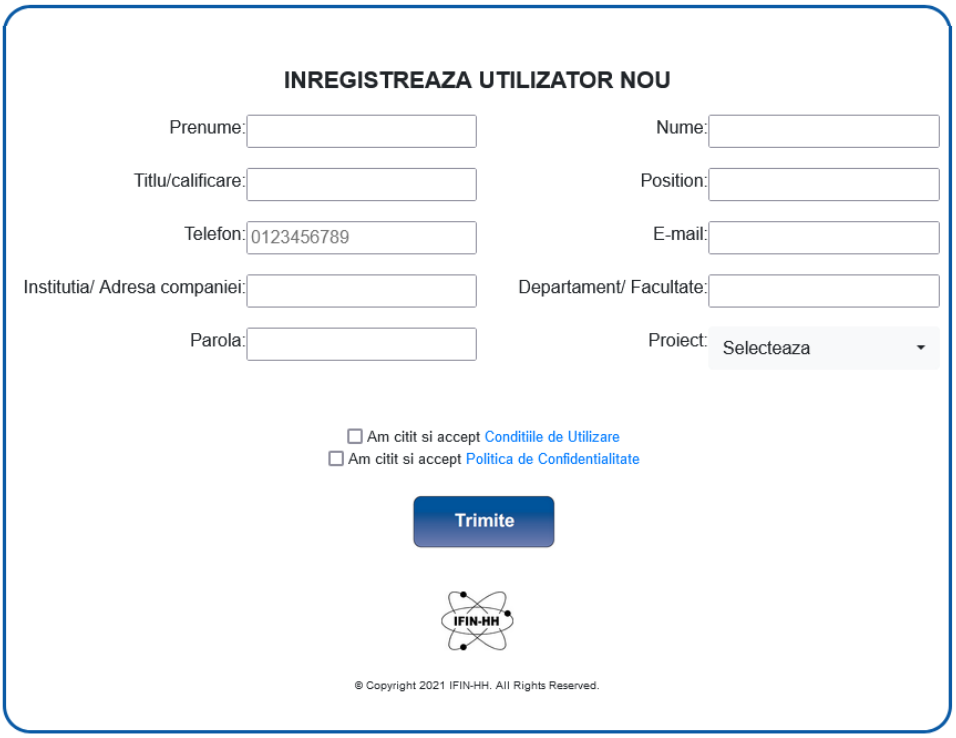


Fig. 4: Interfata de inregistrare a unui nou utilizator intr-un proiect existent

Inainte de a trimite formularul, aplicantul trebuie sa confirme ca a citit si accepta "*Conditiile de Utilizare "* si *"Politica de Confidentialitate "*.

Solicitarea este trimisa spre aprobare conducatorului proiectului de calcul respectiv.

7. SCHIMBAREA PAROLEI DE ACCES

Dacă utilizatorul a uitat/pierdut parola de acces, acesta va acționa butonul ”*Recupereaza parola*” din formularul din Fig . 1.

Fig. 6: Solicitarea recuperarii parolei

Utilizatorul va completa adresa de e-mail cu care a fost creat contul, și va acționa butonul ”*Trimite*” din formularul ”*Recuperare parola*”, demarând astfel procedura standard prin care isi confirma identitatea si poate stabili o noua parola de acces la interfata grafica.

7. ADMINISTRAREA RESURSELOR DE CATRE UTILIZATOR

Dupa autentificarea in intarfata din Fig. 1 (cazul 1 din Sectiunea 3), orice utilizator inregistrat este directionat automat catre interfata web a biroului sau virtual (BV), care afiseaza lista proiectelor in curs sau inactive la care participa sau a participat utilizatorul respectiv.

Fig. 4: Interfata biroului virtual al utilizatorului

Pentru fiecare proiect se afiseaza numele acestuia, apoi organizatia virtuala (VO) in cadrul careia se desfasoara proiectul, starea proiectului (activ/finalizat/suspendat), data de inceput si data de finalizare.

Pagina principala a BV mai cuprinde butoane catre managementul resurselor proprii, administrarea profilului utilizatorului, comunicarea cu administratorul SCF, statistici privind utilizarea resurselor alocate, precum si documentatie (privind sistemul, bibliotecile software, manualul utilizatorului).

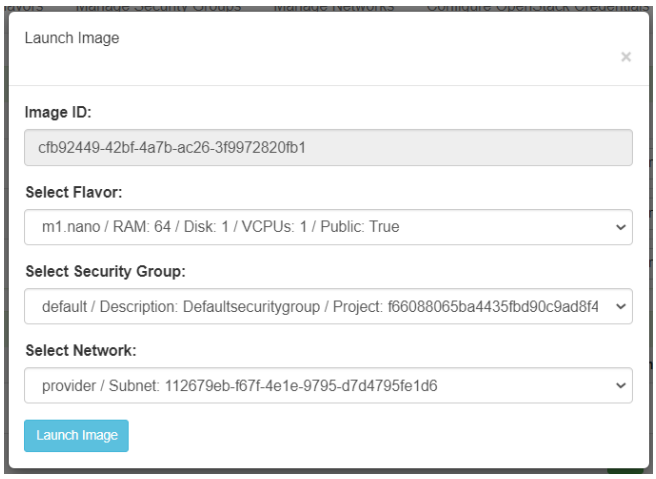
Prin actionarea butonului "*Manage resources*" se face accesul la interfata web prin intermediul careia se pot administra resursele proprii.

Tab-urile interfetei directioneaza catre: masini virtuale, imagini Cloud, flavours, grupuri de securitate, retele de comunicare de date, aplicatii, si poate configura credentiale OpenStack.

Se afiseaza lista de imagini cu sisteme de operare la care este permis accesul si care pot fi utilizate pentru pornirea masinilor virtuale.

Toate informatiile legate de imagini de sisteme de operare, flavor, grup de securitate, retele de comunicare, sunt preluate din OpenStack in urma unor schimburi de mesaje.

Initializarea masinilor virtuale se face prin apasarea butonului "*Launch*", care deschide o fereastra de unde se pot aleage Flavor-ul, grupul de securitate si reteaua de comunicare necesare:



Dupa ce este apasat butonul "*Launch Image*" sistemul initializeaza masina virtuala specificata.

Verificarea starii si a parametrilor masinii virtuale se poate face cu comenzi OpenStack, ca in exemplul de mai jos:

```
openstack server list
```

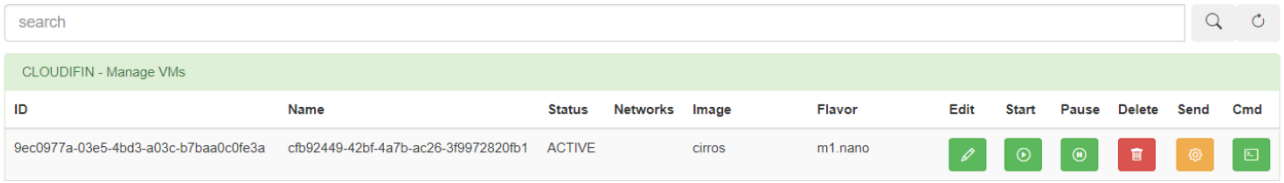
```
+--------------------------------------+--------------------------------------+--------+----------+----------------+------------------+
| ID                                   | Name                                 | Status | Networks | Image          | Flavor           |
+--------------------------------------+--------------------------------------+--------+----------+----------------+------------------+
| 9ec0977a-03e5-4bd3-a03c-b7baa0c0fe3a | cfb92449-42bf-4a7b-ac26-3f9972820fb1 | ACTIVE |          | cirros         | m1.nano          |
+--------------------------------------+--------------------------------------+--------+----------+----------------+------------------+
```

```
openstack server show cfb92449-42bf-4a7b-ac26-3f9972820fb1
```

care furnizeaza urmatoarele informatii:

```
+-----------------------------+--------------------------------------------------------------+
| Field                       | Value                                                        |
+-----------------------------+--------------------------------------------------------------+
| OS-DCF:diskConfig           | MANUAL                                                       |
| OS-EXT-AZ:availability_zone | nova                                                         |
| OS-EXT-STS:power_state      | Running                                                      |
| OS-EXT-STS:task_state       | None                                                         |
| OS-EXT-STS:vm_state         | active                                                       |
| OS-SRV-USG:launched_at      | 2023-07-26T09:04:44.000000                                   |
| OS-SRV-USG:terminated_at    | None                                                         |
| accessIPv4                  |                                                              |
| accessIPv6                  |                                                              |
| addresses                   |                                                              |
| config_drive                |                                                              |
| created                     | 2023-07-26T09:04:35Z                                         |
| flavor                      | m1.nano (0)                                                  |
| hostId                      | 4e5255e23bf6d327dd657de4abf86d84615f6a6f1dc22c0b85ee6ef3     |
| id                          | 9ec0977a-03e5-4bd3-a03c-b7baa0c0fe3a                         |
| image                       | cirros (cfb92449-42bf-4a7b-ac26-3f9972820fb1)                |
| key_name                    | None                                                         |
| name                        | cfb92449-42bf-4a7b-ac26-3f9972820fb1                         |
| progress                    | 0                                                            |
| project_id                  | f66088065ba4435fbd90c9ad8f4a3f9b                             |
| properties                  |                                                              |
| status                      | ACTIVE                                                       |
| updated                     | 2023-07-26T09:04:44Z                                         |
| user_id                     | 26a49449216745849 82d97021c294d08                            |
| volumes_attached            |                                                              |
+-----------------------------+--------------------------------------------------------------+
```

Tab-ul "*Manage VMs*" din interfata grafica de administrare a resurselor trimite catre urmatoarea pagina de comenzi (de editare, pornire/oprire VM, desfiintare, etc.)



Butonul "*Cmd*" permite accesul in interfata de linie de comanda la VM, care se face prin intermediul browser-ului web, ca in exemplul de mai jos: