Universitatea “Politehnica” din București
Facultatea de Electronică, Telecomunicații și Tehnologia Informației

Platforma software pentru managementul mașinilor virtuale

Inginerie software pentru comunicaţii

Electronică şi Telecomunicaţii
Reţele şi Software de Telecomunicaţii (ETC – RST)

Conducător științific
Conf. Dr. Ing. Eduard-Cristian POPOVICI

Absolvent
Tiberiu-Cristian CORNEANU

Anul 2019

Cuprins

Introducere

1. **Mașinile virtuale**
   1.1 Ce este o mașină virtuala

2. **Platforma Web**
   2.1 Angular
   2.2 Schema generala
   2.3 Baza de date

3. **Server Web**
   3.1 Java JPA
   3.2 JSON

4. **Securitate**
   4.1 HTTPS

5. **Cloud**
   5.1 Azure
   5.2 Vagrant
   5.3 Docker

6. **Testare/Utilizare**
   1.1 Nu am ceva concret

Concluzii

Lista acronimelor – se vor lista, în ordine alfabetică, toate acronimele folosite în text, împreună cu semnificația inițialelor și traducerea în limba română (dacă este cazul);

Acronim	Limba engleză	Limba română
VM	Virtual machine	Mașină virtuală
VMM	Virtual machine monitor or hypervisor	Supervizor de mașină virtuală
CPU	Central processing unit	Unitate centrală de prelucrare
OS	Operating system	Sistem de operare
CPU	Central processing unit	Unitate centrala de procesare
JPA	Java Persistence API	
DB	data base	
HTML	hypertext Markup Language	
CSS	Cascading Style Sheets	
JSON	JavaScript Object Notation	

Introducere – aceasta va conține motivația alegerii temei, gradul de noutate al temei (dacă este cazul), obiectivele generale ale proiectului, metodologia folosită, și va rezuma contribuția studentului și rezultatele obținute;

Scopul creării unei platforme ce se ocupa cu gestiunea unor mașini virtuale depinde foarte mult de domeniul în care este implementată o astfel de platformă de management însă toate aceste au un punct comun și anume gestionarea cât mai productivă a componentelor hardware. Prin gestionarea componentelor hardware se va realiza reducerea unor costuri cu privire la folosirea unei placi de memorie, a unui CPU, a unei plăci video sau a unui hard disk ce oferă specificații cât mai aproape de cerințele utilizatorului. Astfel folosind mașinile virtuale utilizatorul poate schimba oricând dispozitivele hardware de care are nevoie pentru a se mula cât mai aproape de specificațiile dorite având un cost cat mai scăzut.

O astfel de platformă poate fi folosită în cadrul învățământului universitar pentru gestionarea laboratoare ce necesită diferite specificații hardware. În acest moment astfel de platforme se află încă la început find dezvoltate în fiecare zi noi moduri prin care se poate efectua o gestiune mai eficienta a traficului de date.

Obiectivul implementării unei astfel de platforme în cadrul universitar este de a oferi: libertate cât mai mare a cadrelor didactice ce nu vor mai fi limitate de către dispozitivele hardware și posibilitatea studenților de a se conecta de oriunde prin intermediul internetului la mașinile virtuale oferite de către facultate. Totodată studenții nu se vor mai preocupa cu problemele întâmpinate la instalarea unor programe și referitoare la specificațiile necesare ale calculatorului personal, astfel reușind să salveze atât timp cât și bani.

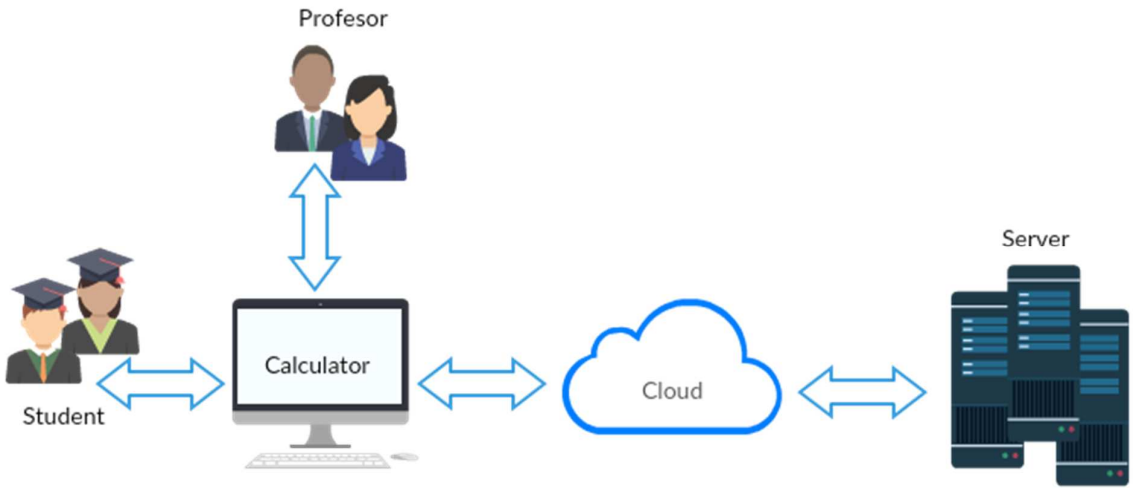
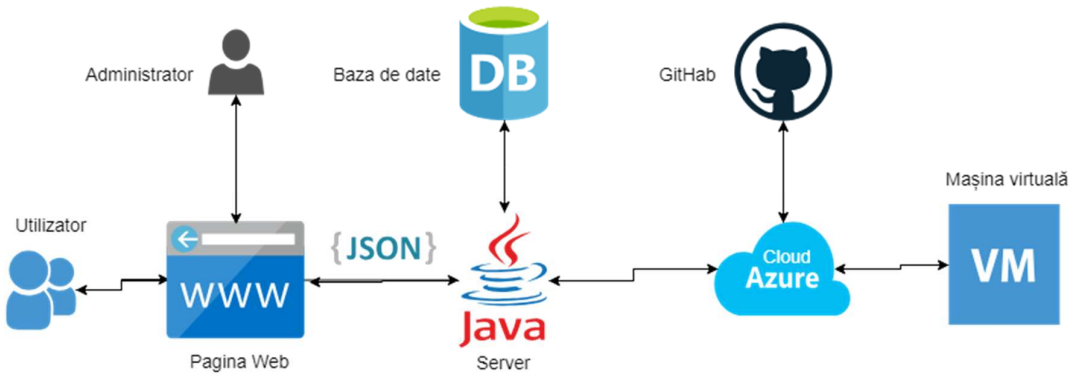


Fig a. Schema generala a proiectului

În vederea realizării acestui proiect s-a construi o platformă menita să ofere utilizatorilor săi o interacțiune cât mai plăcuta cu mașinile virtuale.

Platforma este creata cu ajutorul programului Angular codul find realizat în HTML, CSS și TypeScript, datele sunt menținute într-un sistem de baze de date relaționar SQL și legătura dintre platforma web și baza de date se face prin intermediul unui server web ce a fost construit utilizând limbajul Java JPA. Prin intermediul unui Shel Script se realizeaza mașina virtuală și se instalează programele necesare comanda făcându-se din intermediul paginii web.



Capitole – lucrarea va conține capitole numerotate crescător (introducerea nu se numerotează, primul capitol fiind capitolul 1);

1. Mașina Virtuală

1.1 Ce este o mașină virtuală?

O mașină virtuală este un fișier tip calculator care se comportă exact ca un calculator real asemenea celui pe care îl avem în posesia noastră. Într-o altă ordine de idei o mașină virtuală este un calculator creat cu ajutorul unui alt calculator fizic. Rulează asemănător ca oricare alt calculator într-o fereastră oferindu-i utilizatorului o experiență asemănătoare pe mașina virtuală ca cea pe care o are pe orice alt sistem de operare ce este executat pe un calculator. Mașina virtuala lucrează într-un mediu de joaca, numit sandbox, în care softwar-ul ce se afla instalat pe aceasta nu va interacționa sau va realiza schimbări pe calculatorul gazdă. Astfel se poate produce un mediu ideal de testare a unor sisteme de operare ca cele de tip beta, versiunea a doua a sistemului de operare inițial, accesarea de date ce au fost compromise în perfectă siguranță, crearea unor sisteme de operare de siguranță în cazul în care sistemul principal va fi compromis și prelucrarea unor noi aplicații.

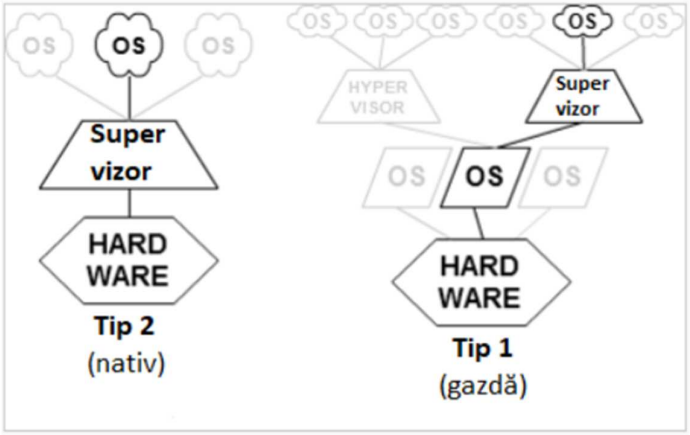
Mai multe mașini virtuale pot lucra simultan pe același calculator fizic. În cazul serverelor prelucrarea unor sisteme de operare multiple se face concomitent cu ajutorul unui software ce monitorizează activitatea lor, sub denumirea de VMM sau supervizor în timp ce calculatoarele desktop în mod obișnuit folosesc un sistem de operare pentru a rula celelalte sisteme de operare în ferestrele sale de programare [1].

Supervizorul este un software ce crează, gestionează și prelucrează mașinile virtuale denumite și mașini oaspete (guest machine) find instalate pe o mașină gazdă (host machine). Supervizorul separă sistemul de operare a unui calculator și

aplicațiile de hard. În cazul virtualizării permite sistemului fizic gazdă să opereze mai multiple mașini virtuale în calitate de vizitator pentru a maximiza eficiența resurselor calculatorului cum ar fi memoria, lățimea de bandă a rețelei și ciclul CPU [2].

Chiar daca VM-urile pot fi prelucrate pe același hardware fizic, sunt însa separate unele de altele ceea ce inseamnă că dacă un VM prezintă o eroare, se oprește sau prezintă un atac malițios acest lucru nu va afecta restul VM-urilor. VM-urile sunt, de asemenea, foarte mobile - deoarece sunt independente de hardware-ul subiacent, pot fi mutate sau migrate între servere virtualizate locale sau la distanță mult mai ușor decât aplicațiile tradiționale care sunt legate de hardware-ul fizic.

Există două tipuri de supervizori, ce se regăsesc sub denumirile de tipul 1 sau tipul 2.



Supervizorul de tip 1, este nativ și rulează direct pe hardware-ul gazdei pentru a controla hardware-ul și pentru a gestiona VM-urile clienților. Supervizorii moderni includ AntsleOs, Xen, serverul Oracle VM pentru SPARC, serverul Oracle VM pentru x86, Microsoft Hyper-V și ESX / ESXi de la VMware.

Supervizorul de tip 2, rulează pe un sistem convențional find găzudit la fel ca și alte aplicații din sistem. În acest caz, sistemul de operare gazdă rulează ca proces pe gazdă, în timp ce supervizorul separă sistemul de operare oaspete de sistemul de operare gazdă. Exemple de supervizori de tip 2 includ VMware Workstation, VMware Player, VirtualBox și Parallels Desktop pentru Mac.

În spațiul centrelor de date pentru întreprinderi, consolidarea a avut ca rezultat trei furnizori majori pe frontul supervizor: VMware, Microsoft și Citrix Systems.

https://medium.freecodecamp.org/a-beginner-friendly-introduction-to-containers-vms-and-docker-79a9e3e119b

https://www.networkworld.com/article/3243262/virtualization/what-is-a-hypervisor.html

What is a vm?

2. Platforma Web

2.1 Angular

Angular este o platformă care ușurează construirea de aplicații pe web. Unghiul combină șabloanele declarative, injectarea de dependență, instrumentele de la sfârșitul la sfârșit și cele mai bune practici integrate pentru a rezolva provocările legate de dezvoltare. Angular împuternicește dezvoltatorii să construiască aplicații care să trăiască pe web, pe mobil sau pe desktop.

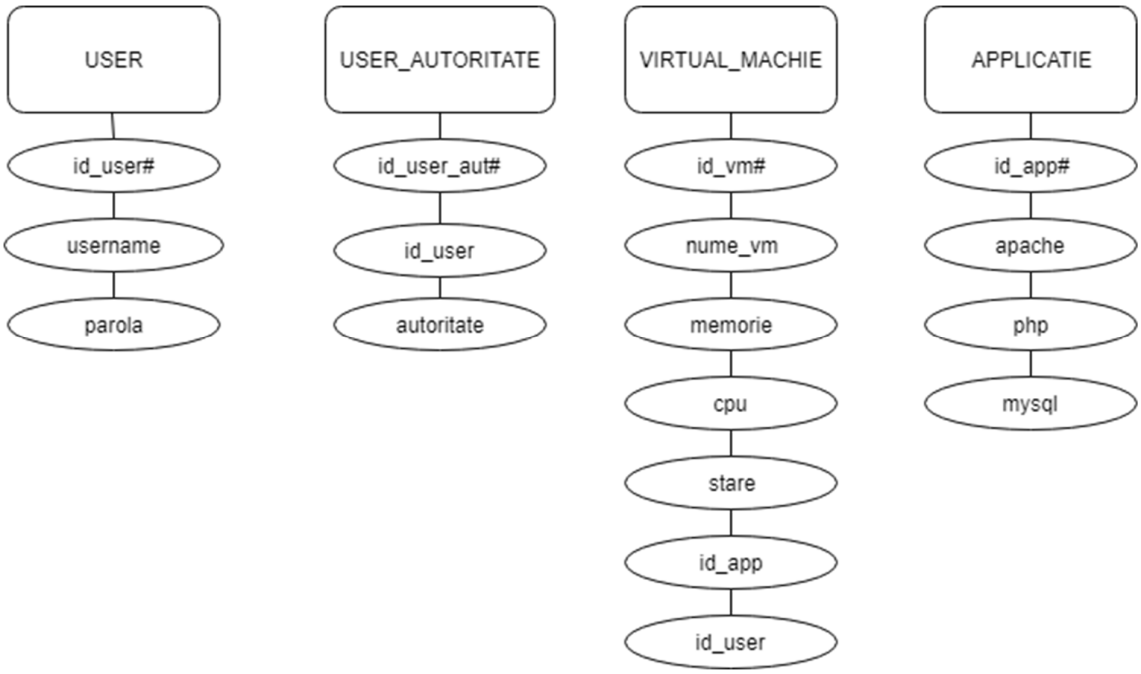
Blocurile de bază ale unei aplicații Angulare sunt NgModules, care oferă un context de compilare pentru componente. NgModulele colectează codul asociat în seturi funcționale; o aplicație unghiulară este definită de un set de NgModule. O aplicație are întotdeauna cel puțin un modul rădăcină care permite bootstrapping și de obicei are multe alte module de caracteristici.

2.2 Schema generala a aplicatiei Angular

De sters pozele din assets

2.3 Baza de date

O bază de date este un instrument pentru colectarea și organizarea informațiilor. Bazele de date pot stoca informații despre persoane, produse, comenzi sau orice altceva. Multe baze de date încep ca o listă într-un program de procesare a textului sau o foaie de calcul. Pe măsură ce lista se mărește, datele încep să conțină redundanțe și inconsistențe. Ele devin greu de înțeles sub formă de listă și există modalități limitate de a căuta sau a extrage subseturi de date pentru revizuire. După ce încep să apară aceste probleme, este o idee bună să transferați datele într-o bază de date creată de un sistem de gestionare a bazelor de date (DBMS), cum ar fi Access.



Un user se va autentifica în interiorul paginii Web și în urma unei interogări se va verifica dacă acesta se află sau nu în interiorul Bazei de Date. Dacă userul se afla în baza de date se va face o căutare în interiorul tabelei USER_AUTORITATE pe baza atributului id_user din tabela USER căutarea întorcând autoritatea userului (administrator, profesor, student). În tabela VIRTUAL_MACHINE avem prezent atributul “id_vm” ce reprezinta cheie primara in tabelă, numele mașinii virtuale reprezentate de atributul “nume_vm”, dimensiunea memoriei reprezentată de către atributul memorie, numărul de unități centrale de procesare reprezentat de atributul “cpu”, starea actuala a mașinii virtuale reprezentată de atributul ”stare” (ce poate fi: stop, run, cloning, deleting, rebuild, error, worning), identificatorul aplicațiilor ce sunt instalate pe mașina virtuală reprezentat de atributul “id_app” și identificatorul userului ce determina userul ce are acces la mașina virtuală reprezentat de atributul ”id_user”.

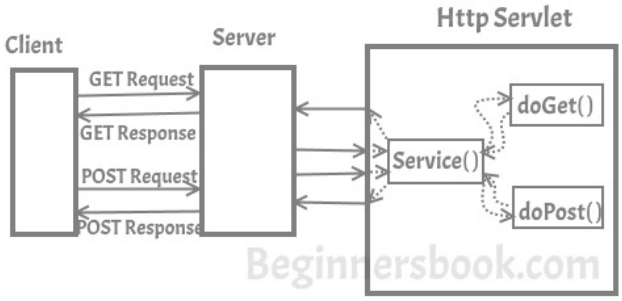
3. Server Web

3.1

Un servlet este o clasă scrisă în limbajul Java al cărei scop este generarea dinamică de date într-un server HTTP. O astfel de clasă poate crea atât conținut HTML, cât și documente XML, imagini, alte fișiere etc. În cea mai mare parte servleții sunt folosiți împreună cu protocolul HTTP, deși există posibilitatea de a utiliza servleți generici care nu sunt legați de un anumit protocol. Așadar prin „servlet” se înțelege de obicei „servlet HTTP”.

Un servlet nu poate fi executat direct de către server, ci de către un container web. Pentru a răspunde unei cereri de la un client (de obicei un browser) către un anumit servlet, serverul trimite mai departe cererea către container, care se ocupă

de instanțierea obiectului respectiv și de apelarea metodelor necesare. Exemple de containere web care suportă servleți includ Apache Tomcat (acesta poate rula și ca server de sine stătător, dar performanțele sale nu se compară cu cele ale unui server web dedicat precum Apache).



REST este un stil arhitectural care se bazează pe standardele web și pe protocolul HTTP. Într-o arhitectură bazată pe REST totul este o resursă. O resursă este accesată printr-o interfață comună bazată pe metode standard HTTP. Într-o arhitectură bazată pe REST aveți un server REST care oferă acces la resurse. Un client REST poate accesa și modifica resursele REST. Fiecare resursă ar trebui să suporte operațiuni comune HTTP. Resursele sunt identificate de ID-urile globale (care sunt, de obicei, URI-uri).

REST permite reprezentări diferite, cum ar fi text, XML, JSON etc. Clientul REST poate solicita o reprezentare specifică prin protocolul HTTP (negociere de conținut).

Metode HTTP

Metodele PUT, GET, POST și DELETE sunt utilizate în mod obișnuit în arhitecturile bazate pe REST. Următorul tabel oferă o explicație a acestor operațiuni.

GET definește accesul numai pentru citire al resursei fără efecte secundare. Resursa nu se modifică niciodată printr-o solicitare GET, de exemplu, solicitarea nu are efecte secundare (idempotent). PUT creează o nouă resursă. De asemenea, trebuie să fie idempotent. DELETE elimină resursele. Operațiile sunt idempotent. Ele se pot repeta fără să ducă la rezultate diferite.
POST actualizează o resursă existentă sau creează o nouă resursă.

3.2 JSON

Necesita o librarie https://code.google.com/archive/p/json-simple/downloads
Transmite date intre server si aplicatia web

4. Securitate

4.1 Metode de securizare a paginii web

Validarea imputului

Validări de input, trebuie verificat fiecare câmp în parte și să îi fie permis utilizatorului să introducă numai valori valide. Prin validarea de input, este limitat la intrarea unui alt tip de valoare. De exemplu, în câmpul zilei de naștere, un utilizator poate introduce numai în formatul DD - MM - AAAA. Un utilizator nu poate încerca nici măcar să introducă un șir sau un caracter.[4.3]

Se protejează de atacurile de tip Scripting Cross Site (CSS)

O aplicație web acceptă câteva intrări de la utilizatori care pot fi sub formă de comentarii, subiecte sau bloguri sub formă de HTML. Acceptarea HTML-ului complet poate fi periculoasă de la intrare. Deoarece permite JavaScript să fie executat, acest lucru poate cauza confidențialitatea cookie-urilor. Aceste cookie-uri vor permite unui intrus accesul la datele site-ului. Pentru a-l controla, există unele etichete HTML sau cod pe care le puteți personaliza, nu există nicio modalitate posibilă de a executa JavaScript, acesta va interzice formatarea. Puteți permite unele cod BB sau etichete BB, care sunt admise în mod frecvent pe forumuri.[4.3]

Paza împotriva SQL Injection

Injectarea SQL este una dintre cele mai bune atacurile de securitate cunoscute. Apare atunci când datele nu sunt bifate și aplicația nu scapă de caracterele folosite în șiruri de caractere. Folosind interogări parametrizate și instrucțiuni pregătite, puteți proteja SQL injectarea.[4.3]

Împiedicați spargerea parolelor cu sare

Md5 este cel mai popular algoritm de criptare sau functie pentru a preveni parola de catre dezvoltatorii PHP. Dă ruta imediat. Este o prevenire puternică, dar poate fi crăpată prin utilizarea forței brute.

Pentru a depăși această problemă, dezvoltatorii folosesc de multe ori "Sare".

Pentru a adăuga un text suplimentar înainte de a-l șterge și apoi faceți aceeași parolă furnizată de utilizator. Aceasta va mări lungimea parolei, deci și complexitatea acesteia. Împreună cu sare este o practică bună folosirea unui algoritm de tip hash, cum ar fi Sha1, Sha2, etc.[4.3]

5. Applications

5.1 Azure Cloud

Microsoft Azure este un serviciu de cloud computing creat de Microsoft pentru construirea, testarea, implementarea și gestionarea aplicațiilor și serviciilor printr-o rețea globală de centre de date gestionate de Microsoft. Acesta oferă software-ul ca serviciu (SaaS), platforma ca serviciu (PaaS) și infrastructura ca serviciu (IaaS) și suportă multe limbi, instrumente și cadre de programare diferite, inclusiv software și sisteme specifice Microsoft și terțe părți.

Mașini virtuale, infrastructură ca serviciu (IaaS), care permite utilizatorilor să lanseze mașini virtuale Microsoft Windows și Linux, precum și imagini de mașină preconfigurate pentru pachete software populare. Serviciile de aplicații, platforma ca serviciu (PaaS), permitând dezvoltatorilor să publice și să gestioneze cu ușurință site-uri Web.

Site-urile web, cu densitate mare de site-uri web, permit dezvoltatorilor să construiască site-uri folosind ASP.NET, PHP, Node.js sau Python și pot fi implementate folosind FTP, Git, Mercurial, Team Foundation Server sau încărcate prin portalul de utilizator.

5.2 Vagrant

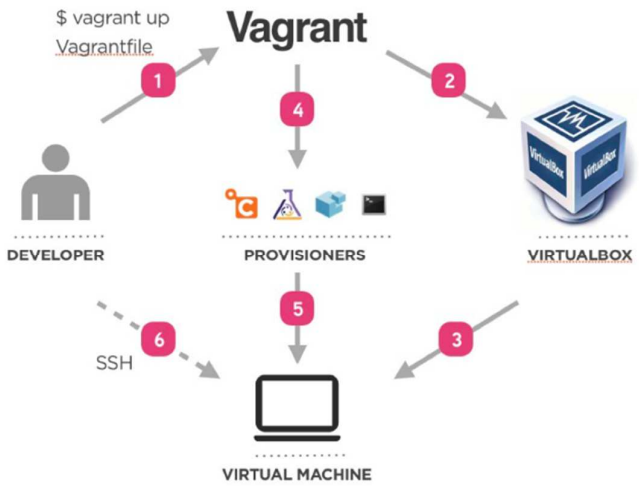
Vagrantul este un instrument pentru construirea și gestionarea mediilor de mașini virtuale într-un singur flux de lucru. Cu un flux de lucru ușor de utilizat și cu accent pe automatizare, Vagrant scade timpul de configurare a mediului de dezvoltare, crește paritatea producției și face ca "lucrările de pe mașina mea" să scuză o relicvă a trecutului.

Mașinile sunt furnizate de VirtualBox, VMware, AWS sau orice alt furnizor. Apoi, instrumentele de furnizare standard pentru industrie, cum

ar fi scripturile shell, Chef sau Puppet, pot instala și configura software-ul automat pe mașina virtuală.

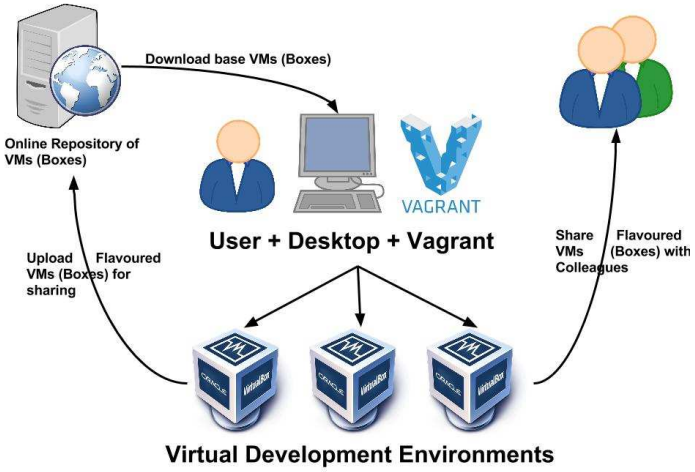
Vagrantul va izola dependențele și configurația utilizatorilor într-un singur mediu de unică folosință, fără a sacrifica niciunul dintre instrumentele cu care sunteți obișnuit (editori, browsere, depanatoare etc.). Odată creat un fișier Vagrant, trebuie doar să faci “vagrant up” și totul este instalat și configurat pentru ca tu să lucrezi. Alți membri ai echipei își creează mediile de dezvoltare din aceeași configurație, deci dacă lucrați pe Linux, Mac OS X sau Windows, toți utilizatorii rulează coduri în același mediu, față de aceleași dependențe, toate configurate la fel cale. Spuneți la revedere la "funcționează pe mașina mea" bug-uri.[4.4]

Vagrant vine cu câteva funcții “ascunse”: vagrant push și vagrant share. Prin comanda push poți face deployment pe un server de staging sau production dintr-o singură comandă (și câteva linii de configurare în Vagrantfile. Vagrant share este o alternativă pentru serverul de staging. În loc să urci proiectul pe un server de teste, pe care să vadă clientul site-ul, poți să-i trimiți un link de tipul http://ghastly-wombat-4051.vagrantshare.com pe care poate vedea exact ce lucrezi.



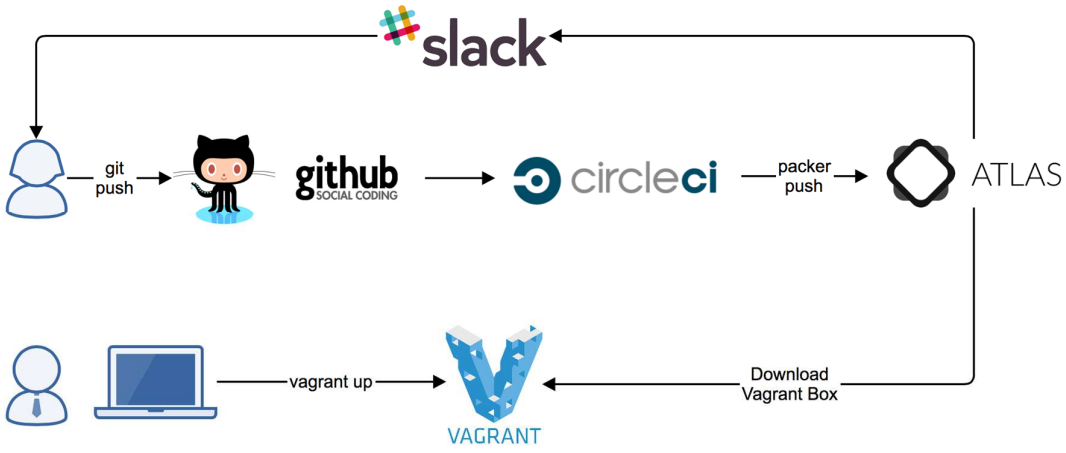
Se bazeaza pe VortualBox deoarece este gratuit, open-source și suficient de stabil. Odată instalat Vagrant și VirtualBox suntem gata de a porni.Comanda vagrant init <vagrant-box> 'poate fi folosită pentru a crea configurația inițială (Vagrantfile) și pentru a defini baza VM (cutie) cu care ar trebui construită. Rularea "vagrant up" va descărca un VM pre-existent dintr-un depozit web și o va porni. Acum VM este în funcțiune și un utilizator poate accesa prin intermediul comenzii “vagrant ssh”,

putând astfel să instaleze și să configureze VM-ul așa cum doresc și, de asemenea, să producă o cutie proprie din această imagine pentru a le împărtăși altora.



Odată ce utilizatorul “flavours” (customizează) un VM de dezvoltare, acesta poate să-l împărtășească direct cu colegii și deasemenea poate să îl încarce într-un repository pentru a face shar.[4.5]

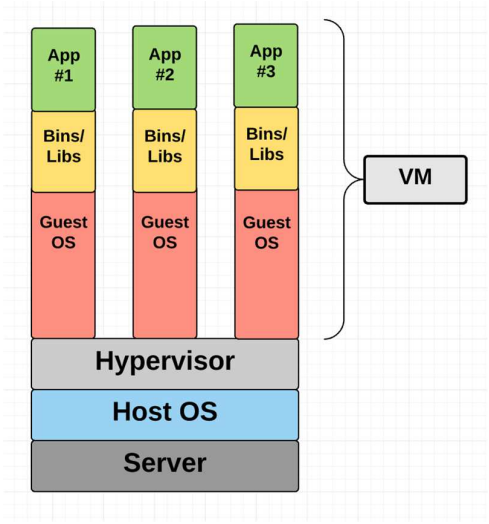
Putem utiliza Atlas pentru a construi Vagrant Boxes cu Packer in Cloud. Ca tip de scriere nu există un integratpor direct de GitHub pentru Atlas. Ca soluție putem adăuga un alt server CI ca un adeziv(legătura) între GitHub și Atlas pentru a face să funcționeze. Pentru acest exemplu am folosit CircleCI, primind notificări Slack din această construcție.[4.6]



5.3 Docker

Deoarece VM are un sistem virtual de operare propriu, hypervisor joacă un rol esențial în furnizarea VM-urilor cu o platformă pentru a gestiona și executa acest sistem de operare oaspete. Acesta permite computerelor gazdă să-și împărtășească resursele între mașinile virtuale care rulează în calitate de invitați.

După cum se poate observa în diagramă, VM-urile pachetează hardware-ul virtual, un kernel (adică OS) și spațiul de utilizator pentru fiecare VM nou.



Spre deosebire de un VM care oferă virtualizarea hardware, un container oferă virtualizare la nivel de sistem de operare prin abstractizarea "spațiului utilizator".

Pentru toate intențiile și scopurile, containerele arată ca un VM. De exemplu, au spațiu privat pentru procesare, pot executa comenzi ca root, au o interfață privată de rețea și o adresă IP, permit rute personalizate și reguli iptable, pot monta sisteme de fișiere și altele.

O mare diferență între containere și VM-uri este că recipientele * împart * nucleul sistemului gazdă cu alte containere.

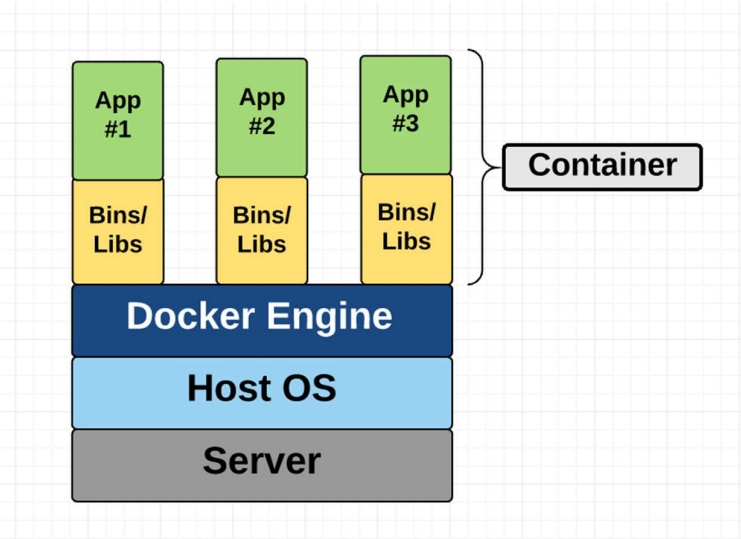


Diagrama arată că containerele ambalează doar spațiul utilizator, și nu nucleul sau hardware-ul virtual cum ar fi un VM. Fiecare container obține propriul spațiu de utilizator izolat pentru a permite mai multor containere să ruleze pe o singură mașină gazdă. Putem vedea că toată arhitectura de nivel de sistem de operare este împărțită între containere. Singurele părți care sunt create de la zero sunt coșurile și libs. Acest lucru face ca recipientele să fie atât de ușoare.

Docker este un proiect open source bazat pe containere Linux. Utilizează caracteristicile Kernel-ului Linux, cum ar fi spațiile de nume și grupurile de control, pentru a crea containere deasupra unui sistem de operare.

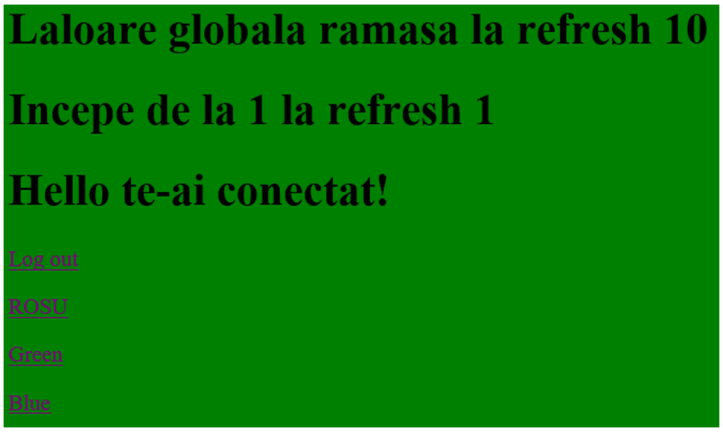
- mai usor de utilizat
- creste viteza
- modularitate și scalabilitate ușurează funcționarea aplicației în containere individuale. De exemplu, este posibil ca baza dvs. de date Postgres să fie difuzată într-un singur container și serverul dvs. Redis în altul, în timp ce aplicația dvs. Node.js se află într-o altă aplicație. Cu Docker, este mai ușor să conectați aceste recipiente împreună pentru a crea aplicația dvs., făcând-o ușor să scalați sau să actualizați componentele în mod independent în viitor.
- comunitate mare de Docker Hub unde poți găsi informații menite să te ajute în crearea imaginii de Docker

6. Testare/Utilizare

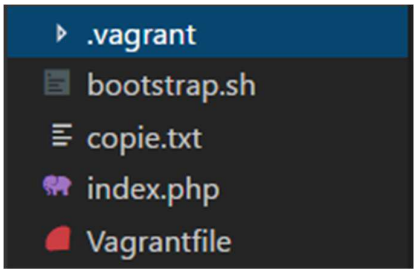
6.1 Aplicația în acest moment

Aplicatia de Java servlet face comunicarea cu Baza de Date verificand userul și parola pe baza cărora se face un salt către o nouă pagină Web.

Prin intermediul unui cookie se va păstra pe toata durata logării culoarea de backround aleasă de către utilizator.



În interiorul folderului creat se face instalarea mașinii virtuale alese rulând “Vagrantfile”, în urma execuției se va crea o mașina virtuala de Linux, Windows, etc. aleasă cu specificațiile dorite. Odată ce mașina este creată se va executa “bootstrap.sh” ce va indica aplicațiile dorite de utilizator a fi instalate pe noua mașină.

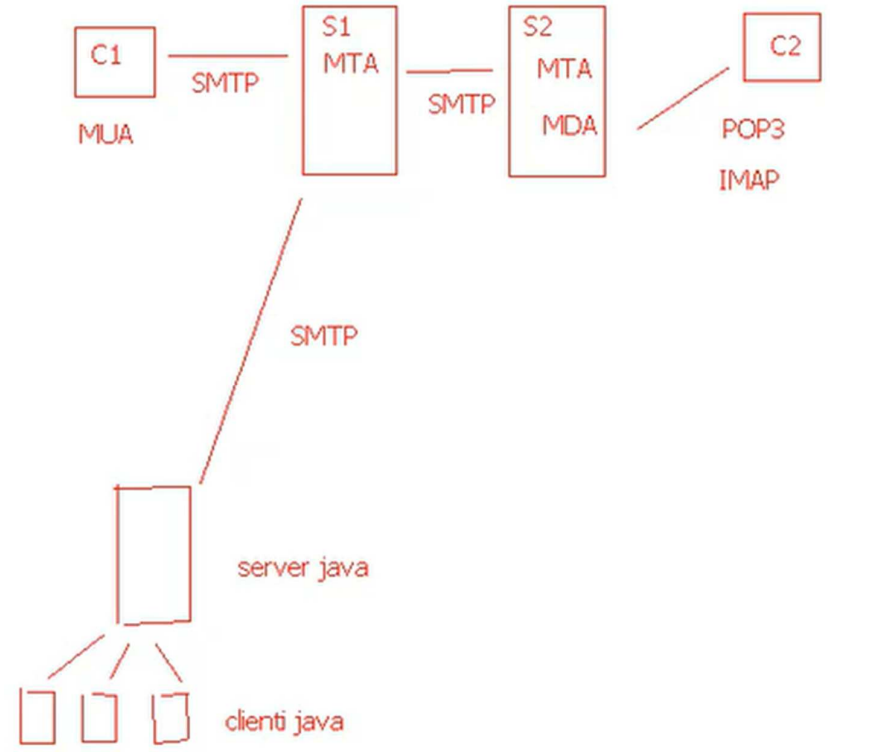


6.2 Probleme ale aplicației

Nu este efectuata legătura prin Rest API între Angular și Java servlet;
Nu este construită securitatea atât pe pagina Web cât și pe aplicație;
Nu se face callul de la pagina Web la Java pentru a fi creată mașina virtuală;
Codul din Vagrant este hardcodat și nu generic;
Nu se face extracția de cod Vagrant din GitHub;
Nu este implementat Docker;

7. Concluzii

Concluziile proiectului – în această parte a lucrării se regăsesc cele mai importante concluzii din lucrare, opinia personală privind rezultatele obținute în lucrare, precum și (opțional) potențiale direcții viitoare de cercetare legate de tema abordată. Concluziile proiectului nu se numerotează ca și capitol;



Comunicatia intre 2 utilizatori in vederea schimbului de informatii prin email(este necesar biblioteca mail.jar)

Bibliografie

1. Carte:

Exemplu:

[1] Neanderthal, H., Tranzistoare MOS cu grila de piatră, Editura Papyrus,
Stonehenge, 20000 î. Hr.

2. Articol dintr-o revistă sau din volumele unei conferințe:

Exemplu:

[2] Ivanovici, I. I. , “Electronic Nothingness”, în Journal of Emo Electronics, nr.
5/2012, pp.12-30.

3. Standarde, documente ale unor organizații:

Exemplu:

[3] RFC90210, A standard for waterless communications between whales,
International Standards Organization, Section XVIII, 2030

4. Documente disponibile exclusiv online:

[4.1] “What is a virtual machine?”, https://www.networkworld.com/article/3243262/virtualization/what-is-a-hypervisor.html, accesat ultima dată la data de: 03/11/2018;
[4.2] ”What is hypervisor?”,https://www.networkworld.com/article/3243262/virtualization/what-is-a-hypervisor.html, accesat ultima dată la data de: 03/11/2018;
[4.3] “Ways To Develop A Secure Your Web Application”, http://www.semiosissoftware.com/5-ways-develop-secure-web-application/, accesat ultima dată la data de: 15/01/2019;
[4.4] “Introduction to Vagrant”, https://www.vagrantup.com/intro/index.html;
[4.5] “Virtual Development Environments made easy”, https://www.mpcdf.mpg.de/about-mpcdf/publications/bits-n-bytes?BB-View=192&BB-Doc=159, accesat ultima dată la data de: 15/01/2019;
[4.6]”Git push to build Vagrant Boxes with Atlas”, https://stefanscherer.github.io/automate-building-vagrant-boxes-with-atlas/, accesat ultima dată la data de: 15/01/2019;
[4.7]”https://medium.freecodecamp.org/a-beginner-friendly-introduction-to-containers-vms-and-docker-79a9e3e119b”, https://medium.freecodecamp.org/a-beginner-friendly-introduction-to-containers-vms-and-docker-79a9e3e119b, accesat ultima dată la data de: 15/01/2019;

Anexe – acestea se numerotează în ordine crescătoare, începînd cu Anexa 1 (fără legătură cu Anexele din acest Regulament). Fiecare anexă se va menționa cel puțin o dată în textul lucrării, ca referință. Tipic, în anexe se includ scheme electrice complete, desene de cablaj imprimat, codul sursă scris de absolvent, fotografii ale machetei/ cablajului/ experimentelor (dacă există), anumite grafice (dacă sunt prea numeroase pentru a fi listate direct în cadrul capitolelor), diagrama Gantt a proiectului (opțională) etc. Nu se vor include în anexe materiale care nu reprezintă contribuția absolventului (de exemplu: cod sursă a unor biblioteci deja existente, foi de catalog a unor componente etc) decît în cazuri bine justificate și cu menționarea explicită a faptul că aceste materiale sînt pre-existente.

Reguli de redactare a lucrării