Cuprins

[1 Motivație 2](#_Toc516817543)

[2 Introducere 3](#_Toc516817544)

[3 Tehnologii folosite 4](#_Toc516817545)

[3.1 Front-end 4](#_Toc516817546)

[3.1.1 HTML 4](#_Toc516817547)

[3.1.2 CSS 4](#_Toc516817548)

[3.1.3 Javascript 5](#_Toc516817549)

[3.1.4 AngularJS 5](#_Toc516817550)

[3.1.5 Bootstrap 6](#_Toc516817551)

[3.2 Back-end 7](#_Toc516817552)

[3.2.1 Node.js 7](#_Toc516817553)

[3.2.2 ExpressJS 8](#_Toc516817554)

[3.2.3 MongoDB 9](#_Toc516817555)

[3.2.4 Docker 10](#_Toc516817556)

[3.1 Google Cloud 12](#_Toc516817557)

[4 Structura aplicației 14](#_Toc516817558)

[5 Bibliografie 15](#_Toc516817559)

# Motivație

# Introducere

* Even if you have a fast computer, chances are you might still need more computing power. Cloud computing offers you that, computing power at your fingertips and, better yet, you decide how much of it to use and when.
* Compiling remotely ensures that your workstation is free to do other computing intensive tasks, like encoding/decoding, testing applications, rendering or number crunching of any kind. Or even if you just need it smooth enough to use your everyday applications.
* Additionally, sometimes you need to ensure that your program runs in different versions of the same platform, or different platforms and operating systems. Sure, virtualization can be done locally with a slew of software applications but this way, you don’t have to worry about setting up anything but the servers themselves. And lest not forget, they can be used for other tasks as well.
* There are arguably many cases where a remote compilation pays off, so let’s see how one would usually go about this.

# Tehnologii folosite

Pentru crearea aplicatiei web, au fost folosite tehnologiile ce formează Mean stack. Acesta este o abordare moderna a dezvoltării web-ului în care un singur limbaj, Javascript, rulează în fiecare nivel al aplicației, de la client la server la persistență. Cele patru tehnologii care fac parte din acest grup și a căror inițiale formează acronimul Mean sunt: MongoDB, ExpressJS, AngularJS și Node.js.

## Front-end

Front-endul reprezintă partea vizuală a aplicației cu care utilizatorii interacționează. Pentru crearea paginii și aspectului ei este nevoie de anumite tehnologii care împreună să afișeze informațiile necesare dar și să permită clientului să le filtreze, să le condiționeze și să le modifice după bunul plac.

Principalele tehnologii folosite sunt HTML pentru definirea structurii întregului site, CSS pentru stilizarea și aranjarea în pagină a elementelor create în scheletul HTML, și Javascript pentru a defini interacțiunile care să atragă utilizatorii.

### HTML

HTML (HyperText Markup Language) este un limbaj simplu folosit pentru a defini scheletul unei pagini web și a da ințeles conținutului prin încadrarea lui într-un anumit tip. Prin intermediul etichetelor ce definesc elementele, browser-ul recunoaște datele și le afișează ca paragraf, listă, tabel, titlu, formular, imagine, video, etc.

HTML este doar text, adică nu este compilat și poate fi citit de oameni. Acest limbaj oferă doar conținut static, de aceea trebuie folosit cu alte tehnologii de cele mai multe ori care să îi asigure și interactivitate.

### CSS

CSS (Cascading Style Sheets) este limbajul folosit pentru stilizarea elementelor din documentul HTML. Stilurile create în CSS sunt utilizate pentru a defini modul în care pagina web va fi afișată pe ecran, legând fiecare element de un set de caracteristici și proprietăți precum dimensiunea textului și a imaginilor, distanța dintre paragrafe, culoarea și poziția paragrafelor și a conținutului lor, grosimea marginilor unui tabel, umplutura, alinierea și fontul textului și multe alte opțiuni.

Legătura dintre elementul din fișierul HTML și stilul aplicat în fișierul CSS se face prin definirea unor clase și id-uri atașate etichetelor HTML. Astfel, cu ajutorul unui singur fișier CSS se pot stiliza mai multe pagini web atât timp cât fiecare face referința la acel fisier. În acest mod, se uniformizează aspectul întregului site, și se scapă de efortul de a modifica fiecare element în parte.

### Javascript

Javascript este un limbaj de programare folosit în special pentru dezvoltarea aplicațiilor web apărut din nevoia ca logica și inteligența să fie și pe partea de client. Acesta este procesat de browser-ul utilizatorului și nu de serverul web, eficientizând traficul dintre client si server. De exemplu, o funcție Javascript ar putea valida datele completate într-un formular înainte de a fi trimis spre server pentru a se asigura că toate câmpurile obligatorii au fost completate și toate datele au formatul corect.

Javascript este un limbaj interpretat, adică nu are nevoie să fie compilat sau preprocesat, având ca avantaj faptul că actualizarea codului sursă este mult mai ușoară.

Există două moduri de a folosi Javascript într-un fișier HTML. Primul este adăugarea întregului cod în cel HTML iar al doilea punerea codului în alt fișier Javascript și adăugarea unei referințe în codul HTML. Javascript este adăugat în HTML prin folosirea etichetelor *<script>* și *</script>*.

### AngularJS

AngularJS este un framework open-source bazat pe Javascript folosit pentru construirea aplicațiilor dinamice cu o singură pagină ce suportă structura MVC (Model View Controller). Acesta permite dezvoltatorului să extindă funcționalitatea din HTML dându-i abilitatea să creeze noi construcții prin intermediul directivelor din AngularJS.

AngularJS adaptează HTML-ul tradițional pentru a prezenta conținutul într-un mod dinamic prin intermediul two-way data-binding[[1]](#footnote-1) ce permite sincronizarea automată a modelelor și a vizualizărilor. Astfel, AngularJS disociază manipularea DOM[[2]](#footnote-2) de logica aplicației cu scopul de a imbunătăți performanța și testabilitatea.

Framework-ul implementează șablonul MVC pentru a separa prezentarea, datele și componentele din logică. Folosind injecția de dependență, acesta aduce serviciile tradiționale din partea de server, precum controllerele dependente de vizualizare, în partea de client a aplicației web. În acest fel, sarcinile pe care le are serverul sunt reduse considerabil.

Directivele sunt marcatori ai unui element din DOM care îi spune compilatorului AngularJS-HTML să atașeze un comportament anume acelui element sau să îl transforme. Dezvoltatorul poate să își creeze propriile directive pentru următoarele tipuri de elemente: directive de element, atribut, stil CSS, comentariu. Pe lângă acestea, în AngularJS există deja o serie de directive, precum: *ng-app*, *ng-controller*, *ng-view*, *ng-show, ng-if*, *ng-model*, *ng-repeat*, *ng-change*, *ng-click*, etc dar și multe directive implementate de alți dezvoltatori și făcute open-source precum: *ng-file-upload*, *angular-ui-bootstrap*, *angular-ui-router*, *angular-tree-control*, *angular-ui-ace*, etc.

Aplicația web este definită de un modul din AngularJS. Un modul este un recipient ce conține o colecție de funcții definite în fișierul Javascript. Unul sau mai multe controllere aparțin unui modul. Fiecare modul este identificat printr-un nume unic și poate să fie dependent de alte module. În AngularJS, fiecare vizualizare (pagina web) are asignat un singur modul prin intermediul directivei ng-app.

Pentru a lega datele și a modifica instant dintr-o parte în alta, în AngularJS se folosesc expresiile declarate de forma *{{}}*. Expresiile vor fi adăugate în șablonul HTML, iar textul va fi actualizat automat când este schimbat din model. Expresiile suportă filtre pentru a formata informația înainte de afișare, dar nu suportă control flow statements [[3]](#footnote-3). Pentru acestea se vor folosi directivele din AngularJS ca *ng-if*, *ng-repeat*, etc.

Aplicația este controlată de obiecte Javascript denumite controllere. Un controller conține un set de funcții, proprietăți și atribute, dar și un context numit *$scope* folosit în comunicarea cu vizualizările și prin intermediul căruia are loc two-way binding. Scopul controllerului este de a manipula elementele din arborele DOM, de a comunica cu serverul, de a valida inputuri sau de a face alte procesări necesare. Pentru a defini un controller al aplicatiei, se folosește directiva *ng-controller*.

Un serviciu în AngularJS este un singleton ce poate fi folosit de alte componente din aplicație ca directive, controllere, filtre sau alte servicii. Serviciile pot fi folosite când este nevoie ca fiecare componentă să aibă acces la o stare sau când este nevoie de o soluție pentru stocarea datelor. În AngularJS există 30 de servicii construite în mod implicit precum $http, *$timeout*, *$interval*, $*window*, etc. Pe lângă acestea, dezvoltatorul poate să își creeze propriile servicii.

### Bootstrap

Bootstrap este un framework open-source pentru front-end dezvoltat de Twitter. Acesta include șabloane de HTML și CSS pentru formulare, butoane, tabele, tipografie, modale, meniuri de navigare, carusel pentru imagini și multe altele, dar și plugin-uri de Javascript. Boostrap este folosit pentru că oferă abilitatea de a crea site-uri web responsive cât mai ușor.

Unul din cele mai căutate aspecte pe care le oferă Bootstrap este că înfățișarea site-ului web construit se adaptează în funcție de dispozitivul în care este vizualizată. Stilizarea CSS se reglează automat în funcție de dimensiunea ecranului, astfel încât chiar și pe telefon sau pe dispozitive de dimensiuni foarte mari, conținutul nu se va pierde sau blocurile nu se vor suprapune.

O altă funcționalitate populară din acest framework este că pagina poate fi împărțită în coloane de diferite dimensiuni, făcând mult mai ușoară organizarea și aranjarea div-urilor în pagină. Bootstrap împarte fiecare componentă în 12 coloane de dimensiuni egale, care sunt calculate în funcție de lățimea totală a dispozitivului. Astfel, componentele pot fi amplasate sub forma unei grile, iar acest lucru se face cu ajutorul claselor de forma *col-xs-4*, *col-sm-8*, *col-md-3*, *col-lg-6*, etc. Orice bloc poate fi împărțit pe mai multe coloane, de dimensiuni diferite, atât timp cât suma dimensiunilor este 12.

## Back-end

Back-endul este partea de server a aplicației, cea pe care utilizatorii nu o pot vedea, și cea care face comunicarea dintre browser și baza de date. Acesta este constituit din trei părți: serverul, aplicația găzduită pe el și baza de date care împreună, așteaptă cereri de la codul ce rulează pe browser, cereri făcute de utilizatori prin intermediul interfeței, și trimite răspunsul dorit.

Tehnologiile principale în crearea aplicației de back-end sunt Node.js și framework-ul ExpressJS. Baza de date utilizată este una de tip NoSQL, și anume MongoDB. Pentru că clientul care folosește aplicația ar putea trimite o bucată de cod menită să șteargă fișiere importante de pe server, am folosit Docker, creând un container în care codul clientului rulează, păstrând astfel siguranța serverului.

### Node.js

Node.js este o platformă de dezvoltare open-source ce execută cod Javascript pe partea de server. Este unul din cele mai populare limbaje atunci când este nevoie de o aplicație web real-time cu o conexiune persistentă între browser și server.

Unul din avantajele folosirii Node.js este că, fiind construit pe motorul Javascript V8 creat de Google pentru browserul Chrome, codul se execută foarte rapid. De asemenea, toate API-urile din Node.js se executa in mod asincron, adica nu blocheaza. Astfel, serverul nu asteapta sa primeasca raspunsul de la un API ca sa poata continua executia codului, ci merge mai departe la urmatorul call de API iar serverul poate verifica raspunsul anterior cu ajutorul unui mecanism de notificari de evenimente din Node.js.

Un alt avantaj al limbajului este că există numeroase module open-source ce pot fi downloadate prin suportul built-in de management al pachetelor folosind comanda *npm*. Ecosistemul acestor module este public, deci poate fi accesat și folosit de oricine, și de asemenea, oricine poate să iși publice propriile module care vor fi listate în repository-ul npm.

Dintre cele mai folosite module din Node.js se remarcă:

* Express – un framework de dezvoltare flexibil si minimal folosit de majoritatea aplicațiilor web în Node.js
* Hapi – un framework modular și simplu de folosit pentru crearea aplicațiilor web și de servicii
* Connect – un framework de server HTTP extensibil ce folosește plugin-uri cunoscute ca *middleware[[4]](#footnote-4)*; este fundația de bază pentru Express
* Socket.io și Sockjs – componentă server-side a două dintre cele mai comune componente websockets[[5]](#footnote-5)
* Pug (Jade) – unul din cele mai populare motoare de templating[[6]](#footnote-6)
* Mongodb – wrapper peste MongoDB ce oferă funcționalități de comunicare cu obiectele din baza de date
* Redis – bibliotecă de tip client ce oferă funcționalități pentru toate comenzile din Redis
* Lodash – bibliotecă ce oferă funcționalități de utilitate
* Forever – pachet folosit pentru asigurarea că un script Node.js rulează la infinit
* Bluebird – bibliotecă folosită pentru promisiuni cu performanțe foarte bune
* Moment – pachet folosit pentru parsarea, validarea, manipularea și formatarea variabilelor de tipul dată.

Orice proiect Node.js are un fișier numit *package.json* ce conține diverse informații despre proiect și autor precum:

* Numele și versiunea proiectului; De fiecare dată când codul este schimbat și se urcă modificările într-un sistem de version control, dezvoltatorul va crește versiunea codului.
* Calea către scriptul principal al proiectului și comanda necesară pentru rularea aplicației.
* Autorul proiectului
* O descriere a proiectului
* O listă de etichete după care proiectul poate fi găsit mai ușor în repository-ul npm
* Licența sub care e scris proiectul
* O listă de dependentț fără de care proiectul nu poate fi rulat. Fiecare dependență este o bibliotecă din repository-ul npm și pe lângă numele ei, poate fi menționată și versiunea pentru care proiectul poate fi executat. Pentru a instala toate dependențele deodată, se folosește comanda *npm install*. Instalarea se va face în mod recursiv pentru că fiecare bibliotecă are la rândul său un fișier *package.json* cu dependențele sale.

### ExpressJS

ExpressJS este un framework de Node.js pentru partea de server a unei aplicații web ce se remarcă prin faptul ca e foarte ușor de implementat funcționalitatea API-ului pentru fiecare rută existentă. Acesta oferă metode prin care se poate specifica o serie de funcții ce vor fi executate pentru un anumit verb HTTP[[7]](#footnote-7) și un anumit path atunci când clientul face o cerere și metode prin care se poate trimite un anumit view ca răspuns.

Există multe pachete de middleware compatibile cu ExpressJS care să adreseze majoritatea problemelor când vine vorba de o aplicație web. Dintre acestea, cele mai importante sunt cele care lucrează cu cookies, sesiuni, login, parametrii din url, informațiile trimise într-un request de tip POST, headere de securitate, etc.

Un exemplu de aplicație implementată în Express este cea din Fig. 1, unde se creează o instanță de aplicație care va asculta la portul 3000. Ea va avea o singură rută implementată, cea principală, astfel încât atunci când se va accesa *http://localhost:3000/*, răspunsul afișat va fi “Hello World”.

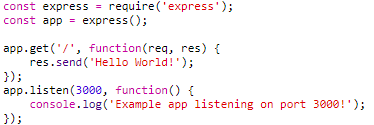


Fig. 1 – Exemplu Hello World în Express

### MongoDB

MongoDB este o bază de date open-source de tip NoSQL ce folosește un model de date orientat pe document. Acest tip de bază de date nu folosește tabele și linii ca în cele relaționale, ci este construit pe o arhitectură bazată pe colecții. O colecție este formată din mai multe documente iar un document este stocat în format BSON[[8]](#footnote-8), conținând mai multe perechi de tip cheie-valoare.

Asemenea altor baze de date NoSQL, MongoDB nu are nevoie de o schemă predefinită a valorilor, astfel că fiecare document poate conține câmpuri diferite iar tipurile fiecărui câmp nu trebuie declarat, și nici nu trebuie să fie la fel în toate documentele. Acest lucru conferă flexibilitate, ușurând modificarea imaginii de ansamblu a datelor.

MongoDB suportă o serie de funcționalități pentru interogarea, adăugarea, modificarea, ștergerea datelor precum find, insert, update, remove, etc. Pe lângă acestea, sunt disponibile două operații complexe de prelucrare a datelor: *aggregate* și *map reduce*.

Indexarea în MongoDB este una dintre conceptele comune cu bazele de date relaționale. Se pot pune indecși pe orice câmp din document, optimizând astfel interogările făcute asupra colecției. Orice colecție creată nouă are în mod implicit index pe câmpul \_id, câmp prezent în orice document și care trebuie să fie unic. Există mai multe tipuri de indecși în MongoDB: indecși pe un singur câmp, indecși compuși, indecși pe hash-uri, indecși geospațiali, indecși unici, indecși de text și indecși parțiali.

MongoDB oferă drivere pentru o serie numeroasă de limbaje, incluzând C, C++, C#, Java, Node.js, Perl, PHP, Python, Ruby și Scala. Librăria pentru Node.js se numește *mongodb*, și poate fi instalată cu pachetul npm. Funcțiile sunt asincrone, la fel cum sunt majoritatea funcțiilor în Node.js. Acestea pot fi însă executate în mod sincron, folosind *await*.

### Docker

Docker este o platformă software open-source creată pentru a rula aplicații într-un mediu izolat de restul sistemului. Acesta funcționează ca o mașină virtuală care ocupă mult mai puțin spațiu. Docker este un instrument ce împachetează, asigură și rulează containere în mod independent de sistemul de operare pe care este instalat.

“Definite printr-un singur fișier de configurare, containerele Docker pot fi refăcute de la zero în foarte scurt timp și vor funcționa identic indiferent de serverul pe care sunt refăcute. Este avantajoasă dezvoltarea aplicațiilor, local, folosind Docker pentru ca mai apoi să poată fi mutate în producție, fără prea multe surprize.” (Neagu, 2014)

O imagine de container este un pachet ușor, executabil și de sine stătător al unei bucăți de software ce include tot ce are nevoie pentru a o executa: cod, timp de rulare, instrumente și librării de sistem, setări. Fiind valabil și pentru Linux și pentru Windows, bucata de software conținută în container va rula în același mod de fiecare dată, indiferent de mediu. Containerele izolează software-ul de mediu, astfel încât diferențele dintre cadrul de dezvoltare și cadrul de producție nu vor mai conta și nu vor mai exista conflicte când mai multe echipe lucrează la același cod, deși au configurații diferite ale sistemelor.

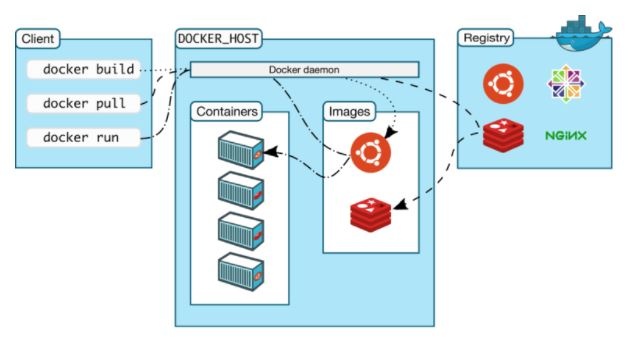


Fig. 2 – Arhitectura Docker

Așa cum se observă în Fig. 1, arhitectura Docker este una de tip client-server și este compusă din următoarele părți:

1. Clientul Docker – interfața prin care utilizatorul interacționează cu containerele
2. Obiectele Docker – principalele componente din Docker: containerele și imaginile
3. Daemon-ul Docker – un proces de background responsabil de recepționarea comenzilor și trimiterea lor spre containere prin intermediul liniei de comandă.
4. Registrul Docker – denumit și Docker Hub, locul unde imaginile de container sunt stocate și de unde pot fi luate.

Avantajele folosirii Docker față de mașini virtuale sunt:

* Containerele Docker sunt izolate și nu au nevoie de un hypervisor de hardware. Astfel, ele sunt mult mai mici și necesită mult mai puține resurse decât un VM.
* Docker este foarte rapid. Dacă pentru un VM durează câteva minute pentru a putea fi folosit, în cazul Docker, durează doar câteva secunde pentru a crea un container dintr-o imagine.
* Containerele pot fi împărțite între multipli membri ai unei echipe, aducând astfel portabilitate în pipeline-ul de dezvoltare. Acest lucru reduce erorile în care o bucată de cod funcționează într-un sistem, dar nu și în altul.

O imagine de Docker este un snapshot pe baza căruia se creează noi containere. Este o reprezentare a unui sistem de fișiere la care se adaugă librăriile specifice unui anumit sistem de operare. O nouă imagine de Docker se crează prin execuția unei serii de comenzi conținute într-un fișier numit Dockerfile. De exemplu, următoarele comenzi din Fig. 2 vor crea o nouă imagine care se construiește dintr-o bază de Ubuntu versiunea 16.06 la care se adaugă și librăriile MongoDB.

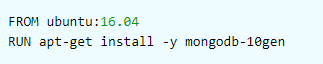


Fig. 3 – Exemplu de conținut pentru un fișier Dockerfile

Un container de Docker este o instanță în curs de rulare a unei imagini. O singură imagine poate fi folosită pentru a porni orice număr de containere. Imaginile sunt read-only, în timp ce containerele pot fi modificate. Modificările dintr-un container vor fi pierdute când el e oprit, dar aceste schimbări pot fi salvate într-o nouă imagine. În Fig. 3, se poate observa comanda folosită pentru a începe un container. În acest exemplu, se va downloada imaginea de Ubuntu versiunea 16.04 dacă ea nu este valabilă local deja, și apoi se va deschide un bash shell.



Fig. 4 – Comandă pentru a rula un container

## Google Cloud



Fig. 5 – Servicii oferite de Google Cloud

Platforma Google Cloud, oferită de Google, este o suită de servicii cloud printre care se numară:

* App Engine – PaaS[[9]](#footnote-9) pentru găzduire de aplicații
* BigQuery – IaaS[[10]](#footnote-10) pentru analiza bazelor de date de mărime mare
* Bigtable – IaaS pentru baze de date de tip NoSQL
* Cloud Datastore – DbaaS[[11]](#footnote-11) pentru baze de date orientate pe document
* Cloud Functions – FaaS[[12]](#footnote-12) pentru funcșii serverless[[13]](#footnote-13) care să fie declanșate de evenimente din cloud
* Cloud Machine Learning Engine – serviciu de învățare automată pentru antrenarea și executarea modelelor Tensorflow
* Cloud Pub/Sub – serviciu pentru publicarea și abonarea la mesaje și fluxuri de informații
* Compute Engine – IaaS pentru mașini virtuale
* Kubernetes Engine – serviciu de orchestrare a containerelor pentru Kubernetes
* Storage – IaaS pentru stocare de obiecte și fișiere

Google Compute Engine este o componentă IaaS din platforma Google Cloud ce permite utilizatorilor să iși creeze mașini virtuale în care să-și execute aplicațiile. Mașinile virtuale se deschid repede, au disc de stocare persistent și livrează performanță înaltă consistentă. Google Cloud permite utilizatorului să iși configureze VM-ul, putând opta pentru imaginile implicite dar și având opțiunea să iși creeze mașina virtuală după o imagine particularizată de Docker sau alte tehnologii de containizare.

Odată creată mașina virtuală, interfața Google Cloud permite accesul în mașină direct din browser. Aici mediul poate fi pregătit pentru a putea rula aplicația, instalând toate programele necesare cu ajutorul comenzilor specifice sistemului de operare.

Avantajele folosirii Google Cloud ca gazdă pentru o aplicaț­ie web sunt:

1. Acesta permite configurarea scalării automate. Asta înseamnă că traficul aplicației este urmărit, și atunci când apar deodată creșteri din cauza campaniilor de publicitate pe rețele de socializare sau pe canalele de televiziune, se creează automat mai multe instanțe ale aplicației care să poată face față la numărul mare de cereri de la utilizatori.
2. Google a investit într-un sistem de cablu de fibră optică pentru Google Cloud care îi oferă acces până la 10 Tbps[[14]](#footnote-14) din totalul de 60 Tbps viteză de transfer de date dintre Statele Unite și Japonia. Cablul ce are 9000 km și traversează oceanul Pacific este cablul subacvatic cu cea mai mare capacitate și are ca limite Oregon din Statele Unite și două locuri din Japonia.
3. Serviciile Google Cloud sunt cele mai ieftine comparativ cu alte servicii de găzduire în cloud. Așa cum se observă în Fig. 6, Google Cloud oferă mașini virtuale la fel de bune, dar la preț cu 30% mai mic decât alte platforme de găzduire web ca Amazon sau Microsoft Azure.

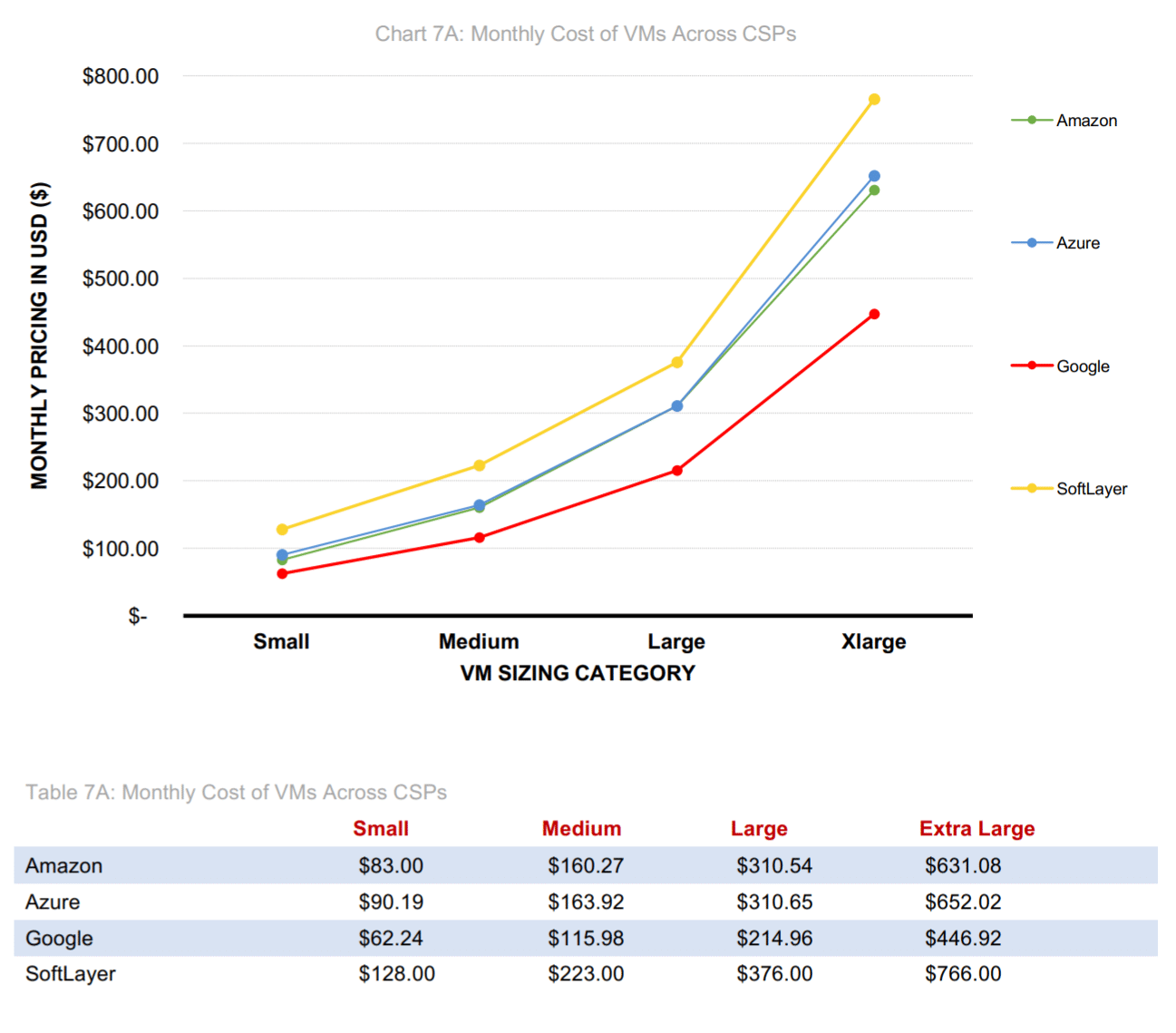


Fig. 6 – Prețurile lunare pentru mașini virtuale de la diferite platforme

# Structura aplicației

Aplicația OnlineCompiler oferă programatorilor posibilitatea de a executa și testa cod scris în diverse limbaje fără a fi nevoie să-și instaleze compilatoarele specifice în calculatorul propriu. Aceștia vor putea astfel scăpa de problemele lipsei de spațiu, scăderii performanței și creșterea folosirii CPU.

# Bibliografie

*Why the Hell Would You Use Node.js*. (2017, 2 24). Preluat pe 6 14, 2018, de pe medium: https://medium.com/the-node-js-collection/why-the-hell-would-you-use-node-js-4b053b94ab8e

*Docker Containers*. (fără an). Preluat pe 6 11, 2018, de pe aquasec: https://www.aquasec.com/wiki/display/containers/Docker+Containers

Jackson, B. (2018, 5 28). *Top 7 Advantages of Choosing Google Cloud Hosting*. Preluat pe 6 15, 2018, de pe kinsta: https://kinsta.com/blog/google-cloud-hosting/

Neagu, C. L. (2014, 8 21). *Îndrumar Docker*. Preluat pe 6 10, 2018, de pe clneagu: http://clneagu.ro/docker/

*Node.js - Introduction*. (fără an). Preluat pe 6 14, 2018, de pe tutorialspoint: https://www.tutorialspoint.com/nodejs/nodejs\_introduction.htm

Sahoo, P. (2015, 4 26). *Basic concepts in AngularJS*. Preluat pe 6 9, 2018, de pe c-sharpcorne: https://www.c-sharpcorner.com/UploadFile/dev4634/basic-concepts-in-angular-js/

Ularu, E.-G., & Puican, F. (2012, 12). *NOUA GENERAŢIE DE BAZE DE DATE NoSQL* . Preluat pe 6 10, 2018, de pe rria: https://rria.ici.ro/wp-content/uploads/2012/12/06-art.-Ularu-si-Puican-revizuit.pdf

1. Legarea datelor in două moduri – conceptul prin care orice modificare făcută în vizualizare (view) se propagă instant în model, și invers. [↑](#footnote-ref-1)
2. Document Object Model - definește o structură logică și standardizată a documentelor, o ordine prin care putem parcurge și edita elemente și conținut din documente HTML sau XML [↑](#footnote-ref-2)
3. Declaratii de flux a controlului – conditionale, bucle, exceptii [↑](#footnote-ref-3)
4. Software ce conectează sistemul client de cel back-end și "lipește" programele între ele [↑](#footnote-ref-4)
5. Protocol de comunicare, ce oferă canale de comunicare full-duplex peste o singură conexiune TCP [↑](#footnote-ref-5)
6. Template = șablon [↑](#footnote-ref-6)
7. GET, POST, PUT, DELETE, etc. [↑](#footnote-ref-7)
8. Binary JSON [↑](#footnote-ref-8)
9. Platform as a Service or Application Platform as a aService [↑](#footnote-ref-9)
10. Infrastructure as a Service [↑](#footnote-ref-10)
11. Database as a Service [↑](#footnote-ref-11)
12. Function as a Service [↑](#footnote-ref-12)
13. Model de execuție fără server [↑](#footnote-ref-13)
14. Terabiți pe secundă [↑](#footnote-ref-14)