**Aplicatia mea pentru SO**

**Mediu rulare Linux:**

- Oracle VM VirtualBox

**Configurare Linux:**

sudo apt-get update

sudo apt-get upgrade

sudo apt-get install openssh-server openssh-client

service ssh start

service ssh status

sudo apt-get install apache2

sudo apache2ctl configtest

sudo ufw allow in "Apache Full"

sudo apt-get install mysql-server

mysql\_secure\_installation

//sudo apt-get install php libapache2-mod-php php-mcrypt php-mysql

sudo apt-get install apache2 php libapache2-mod-php php-mysqli

sudo systemctl restart apache2

sudo systemctl status apache2

sudo apt-get install libssh2-1 php-ssh2 -y

<https://www.cyberciti.biz/faq/linux-unix-running-sudo-command-without-a-password/>

- The forlder questions from mvc needs all permissions for the group “www-data”

**Configurare Apache**

sudo a2enmod rewrite

sudo service apache2 restart

New apache version has change in some way. If your apache version is 2.4 then you have to go to /etc/apache2/. There will be a file named apache2.conf. You have to edit that one(you should have root permission). Change directory text like this

<Directory /var/www/>

Options Indexes FollowSymLinks

AllowOverride All

Require all granted

</Directory>

service apache2 reload

**Configurare user MySQL server**

https://stackoverflow.com/questions/39281594/error-1698-28000-access-denied-for-user-rootlocalhost

$ sudo mysql -u root # I had to use "sudo" since is new installation

mysql> USE mysql;

mysql> CREATE USER 'YOUR\_SYSTEM\_USER'@’localhost’ IDENTIFIED BY 'YOUR\_PASSWORD';

mysql> GRANT ALL PRIVILEGES ON \*.\* TO 'YOUR\_SYSTEM\_USER'@'localhost';

mysql> UPDATE user SET plugin='mysql\_native\_password' WHERE User='YOUR\_SYSTEM\_USER';

mysql> FLUSH PRIVILEGES;

mysql> exit;

$ service mysql restart

E posibil ca mqsql sa aiba quota setat. Cand va atinge limita, va refuza sa porneasca.

**Limit execution time**

$ timeout x command -> makes any command run for x seconds.

**Limit user max processes limit**

<https://www.tecmint.com/set-limits-on-user-processes-using-ulimit-in-linux/>

- my set limit is 20 (in the .json config life)

**Limit user storage with quotas**

<https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/5/html/deployment_guide/ch-disk-quotas>

sudo apt install quota

add usrquota to etc/fstab

sudo mount -o remount /

quotacheck -cug /

sudo quotacheck -m -avug

sudo setquota dorin.haloca 20M 20M 0 0 /

Quotas don’t work for users with a number as username. They need at least one letter!

**Server pentru testare**

- imi voi configura eu prorpiul server cu care voi comunica prin SSH (fie utilizator va avea un cont de utilizator pe serverul Linux)

- fenrir

**Server pentru aplicatia web**

- Apache Linux

Nu am putut adauga libraria pentru SSH2 pe Windows.

- XAMPP Windows/Linux

**Limbaje**

- front-end: HTML+CSS

- back-end: PHP 7

**Autentificare**

- voi folosi autentificarea prin SSH la Fenrir

- resource **ssh2\_connect** ( string $host [, int $port = 22 [, array $methods [, array $callbacks ]]] )

- voi crea un utilizator pe masina mea de Linux folosind useradd, deoarece adduser nu imi permite sa creez utilizatori care contin punct (ex: dorin.haloca)

- voi stoca hash-ul parolei de la Fenrir si voi genera o parola aleatoarea pentru masina mea de Linux

**Utilizatori**

- guest user:

- poate vedea pagina de autentificare

- se poate autentifica

- normal user:

- can do anything the above user can

- can solve questions

- can post questions

- can logout

- can delete own questions

- can view own questions in My Questions page

- admin user:

- can do anything the above user can

- can post questions unconditionally

- can view all questions in All Questions page

- can mark question as valid/invalid

- can post news

- can delete news

**Capitole**

- comenzi bash

- scripturi

- programe C Linux

- id-ul unui capitol nu poate depasi 2 cifre (app/controllers/view\_questions.php:51)

**Adaugare capitol**

- se adauga capitolul in tabela chapters din baza de date

- se creeaza o tabela in baza de date chapter\_x, x fiind id-ul capitolului adaugat

- se adauga in controllers fisierele chapter\_x\_solve.php, chapter\_x\_submit.php, chapter\_x\_result.php

- se adauga in views fisierele chapter\_x\_solve.php, chapter\_x\_submit.php, chapter\_x\_result.php

- se adauga in sytlesheets fisierele chapter\_x\_solve.css, chapter\_x\_submit.css, chapter\_x\_result.css

**Adaugare intrebare:**

To reduce the change of the same command showing different outputs, it will be executed twice when the user tries to submit a question.

**Evaluarea**

**-** Initial voiam sa folosesc fenrir pentru evaluare, dar nu e bine sa trimit acolo solutia corecta, deci voi folosi masina mea Linux pentru executia comenzilor/programelor

- comenzi bash

- voi stoca comanda celui care adauga problema

- cand cineva ii rezolva problema, prin SSH ma voi conecta la fenrir (cu contul celui care rezolva problema) si voi rula pe rand solutia buna a problemei si solutia celui ce rezolva problema si voi compara rezultatle

- comenzile nu vor fi executate direct, ci dintr-un script pentru a preveni combinatii precum sleep 100; sleep 100; (timeout se aplica doar primului sleep 100)

- resource **ssh2\_exec** ( resource $session , string $command [, string $pty [, array $env [, int $width = 80 [, int $height = 25 [, int $width\_height\_type = SSH2\_TERM\_UNIT\_CHARS ]]]]] )

- in caz ca outputul este diferit, testez comenzile date pentru a vedea daca sunt la fel

- scripturi

- voi stoca codul scriptului celui care adauga problema

- cand cineva ii rezolva problema, prin SSH voi trimite scriptul pe fenrir (cu contul celui care rezolva problema) si il voi executa

- apoi voi trimite si voi executa solutia corecta a problemei

- voi compara cele 2 rezultate

- citirea si afisarea se vor face cu fisiere

- bool **ssh2\_scp\_send** ( resource $session , string $local\_file , string $remote\_file [, int $create\_mode = 0644 ] )

- la sfasit voi goli fiserul cu solutia de pe Fenrir (input+output)

- programe C Linux

- voi stoca codul celui care adauga problema

- voi trimite prin SCP pe fenrir (cu contul celui care adauga problema) si voi compila codul

- voi prelua fisierul cu codul compilat si il voi stoca

- cand cineva ii rezolva problema, prin SSH voi trimite codul programului pe fenrir (cu contul celui care rezolva problema), il voi compila si il voi executa

- apoi voi trimite solutia corecta compilata si o voi executa

- citirea si afisarea se vor face cu fisiere

- voi compara cele 2 rezultate

- bool **ssh2\_scp\_send** ( resource $session , string $local\_file , string $remote\_file [, int $create\_mode = 0644 ] )

- bool **ssh2\_scp\_recv** ( resource $session , string $remote\_file , string $local\_file )

- la sfasit voi goli fiserul cu solutia de pe Fenrir

- nu voi stoca programele compilate deoarece diferite compilatoarea pot compila diferit, voi stoca doar could si il compila pe masina mea Linux

**Evaluarea**

When user submits an answer, he will be redirected to the restults page. In case the answer was correct, he will see the author’s solutions. Otherwise, it will be hidden.

The user can report the question despite the correctness of the answer. In addition, a short message may be added.

Secutirty:

- all public functions in PHP classes must call check\_login() from controller, except for Home, Login and Logout

- check\_login() saves the user\_id, user (the username), password of the ssh connection (plain\_text), and the is\_admin status. At the end of the saving process, one more check is done to see whether the user is still logged in. ($\_SESSION[‘user’] is set)

Random function:

- As of PHP 7.1.0, **rand()** uses the same random number generator as [mt\_rand()](http://php.net/manual/ro/function.mt-rand.php).

- mt\_rand — Generate a random value via the Mersenne Twister Random Number Generator

- There is no need to seed the random number generator with [srand()](http://php.net/manual/en/function.srand.php) or **mt\_srand()** as this is done automatically.

**HTML Validation:**

- All html code was validated with <https://validator.w3.org/>

**Concurency per user(student) problem:**

- solved by using semaphores

- <https://brianscode.com/php-semaphore-example/>

**SQL Injection:**

- prevented by using prepared statements

**Solving questions:**

- only posted questions which weren’t marked as “Invalid” will be taken intro consideration

- the user will not get the same question twice in a row

**Bugs:**

Semaphores + sleep() prevent header() from executing in another tab. Header() executes when sleep() finishes execution. Ex: open two tabs of chapter\_x\_solve and click ‘Execute’ on both with a delay.

If there is only one questions suitable for a given chapter, when the user will want to solve questions for that chapter, an infinite loop will occur (the user can’t get the same question twice in a row)