“ALEXANDRU IOAN CUZA” UNIVERSITY OF IAȘI

FACULTY OF COMPUTER SCIENCE



THESIS

Interactive web application-based learning for the

Operating Systems course

proposed by

Dorin Haloca

Session: July, 2019

Scientific Coordinator

Lect. dr. Vidrascu Cristian

“ALEXANDRU IOAN CUZA” UNIVERSITY OF IAȘI

FACULTY OF COMPUTER SCIENCE

Interactive web application-based learning for the

Operating Systems course

Dorin Haloca

Session: July, 2019

Scientific Coordinator

Lect. dr. Vidrascu Cristian

Avizat,

Îndrumător Lucrare de Licență

Titlul, Numele și prenumele \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Data \_\_\_\_\_\_\_\_\_\_\_\_ Semnătura \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**DECLARAȚIE privind originalitatea conținutului lucrării de licență**

Subsemntatul(a) ………………………………………………………………………………………

domiciliul în …………………………………………………………………………………………………..

născut(ă) la data de ………………..…., identificat prin CNP ………….……………..………………..., absolvent(a) al(a) Universității „Alexandru Ioan Cuza” din Iași, Facultatea de ………………………. specializarea …………………………………………………………, promoția …………………………., declar pe propria răspundere, cunoscând consecințele falsului în declarații în sensul art. 326 din Noul Cod Penal și dispozițiile Legii Educației Naționale nr. 1/2011 art.143 al. 4 si 5 referitoare la plagiat, că lucrarea de licență cu titlul: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_elaborată sub îndrumarea dl. / d-na \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, pe care urmează să o susțină în fața comisiei este originală, îmi aparține și îmi asum conținutul său în întregime.

De asemenea, declar că sunt de acord ca lucrarea mea de licență să fie verificată prin orice modalitate legală pentru confirmarea originalității, consimțind inclusiv la introducerea conținutului său într-o bază de date în acest scop.

Am luat la cunoștință despre faptul că este interzisă comercializarea de lucrări științifice in vederea facilitării fasificării de către cumpărător a calității de autor al unei lucrări de licență, de diploma sau de disertație și în acest sens, declar pe proprie răspundere că lucrarea de față nu a fost copiată ci reprezintă rodul cercetării pe care am întreprins-o.

Dată azi, ………………………… Semnătură student …………………………

DECLARAȚIE DE CONSIMȚĂMÂNT

Prin prezenta declar că sunt de acord ca Lucrarea de licență cu titlul „*Interactive web application-based learning for the Operating Systems course*”, codul sursă al programelor și celelalte conținuturi (grafice, multimedia, date de testetc.) care însoțesc această lucrare să fie utilizate în cadrul Facultății de Informatică.

De asemenea, sunt de acord ca Facultatea de Informatică de la Universitatea „Alexandru Ioan Cuza” din Iași, să utilizeze, modifice, reproducă și să distribuie în scopuri necomerciale programele-calculator, format executabil și sursă, realizate de mine în cadrul prezentei lucrări de licență.

Iași, *data*

Absolvent *Dorin Haloca*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(semnătura în original)

ACORD PRIVIND PROPRIETATEA DREPTULUI DE AUTOR

Facultatea de Informatică este de acord ca drepturile de autor asupra programelor-calculator, în format executabil și sursă, să aparțină autorului prezentei lucrări, *Dorin Haloca.*

Încheierea acestui acord este necesară din următoarele motive:

*[Se explică de ce este necesar un acord, se descriu originile resurselor utilizate în realizarea*

*produsului-program (personal, tehnologii, fonduri) și aportul adus de fiecare resursă.]*

Iași, *data*

Decan *Adrian Iftene* Absolvent *Dorin Haloca*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(semnătura în original) (semnătura în original)

CHAPTERS INDEX

[I. Problem description 16](#_Toc12116311)

[II. Configuring the server/servers 17](#_Toc12116312)

[1. General description 17](#_Toc12116313)

[2. Configuring the Linux machine used for execution of code 17](#_Toc12116314)

[3. Configuring the Apache server 19](#_Toc12116316)

[4. Configuring the MySQL server 21](#_Toc12116320)

[III. Requirements 23](#_Toc12116324)

[IV. Types of users in the web application 24](#_Toc12116325)

[V. Authentication process 25](#_Toc12116326)

[1. Components 25](#_Toc12116327)

[2. Scenarios 25](#_Toc12116330)

[VI. Back-end mechanisms 28](#_Toc12116336)

[1. Administrator panel 28](#_Toc12116337)

[2. Managing questions 28](#_Toc12116338)

[3. Code execution on the Linux machine 30](#_Toc12116345)

[4. User authentication and authorization 33](#_Toc12116351)

[5. PHP semaphores 34](#_Toc12116352)

[VII. Adding a new chapter 36](#_Toc12116353)

[1. MySQL Database 36](#_Toc12116354)

[2. Apache server 36](#_Toc12116355)

[VIII. Interface navigation 38](#_Toc12116359)

[IX. Deployment of the application 50](#_Toc12116360)

[X. Bugs 51](#_Toc12116361)

INTRODUCTION

I have chosen to create this web application for the Operating Systems course because the Databases course has one and from my point of view it aids the learning process. Instead of only solving problems by yourself, you can interact with your colleagues through the application and check your knowledge or simply learn through trial and error. For sure, the teacher wouldn’t have the necessary time, energy or imagination to create all the interesting questions students would post there.

The Databases course was the first time I experienced an interactive web application at the faculty and I enjoyed it. Of course, it had its flaws but I tried to fix some of them in my application. Unfortunately, I don’t think my web application is a truly reliable method of grading the students due to the complexity of the Linux operating system. As far as SQL is concerned, it’s way of working is simple and straightforward to assess. While some types of problems can be assessed without getting too deep into the Linux Kernel mechanisms, others require it. Thus, the assessment may be wrong, negatively impacting the students’ grades.

My application consists of an Apache Web Server which the student will interact with, a Linux machine which will handle the execution of code and a MySQL Database server. Of course, the web server will limit the extent to which a student will have access to the Linux environment.

My files for the Apache server use the Model-view-controller (MVC) architectural pattern. The model is the central component of the pattern. It is the application's dynamic data structure, independent of the user interface. It directly manages the data, logic and rules of the application. The view is any representation of information such as a chart, diagram or table. Multiple views of the same information are possible, such as a bar chart for management and a tabular view for accountants. The controller accepts input and converts it to commands for the model or view. [[1]](#footnote-1)

CONTRIBUTIONS

My contribution to the application mainly consisted of writing the code for the Apache Web Server files. I did the front-end as well as the back-end mechanism for the application to work properly while communicating with its components. It wasn’t easy to find proper methods of using and linking the 3 components of my application. In the beginning, I didn’t even know if want I wanted to create was feasible.

I also configured a Docker image to have all the necessary resources for my web application’s requirements.

As it is expected, I created the MySQL database structure and inserted some initial data in the tables. I also added some indexes to optimise the queries. Almost all tables have incremental ids columns so that when inserting a row in a table, you don’t need to specify an id.

# Problem description

My web application doesn’t necessarily solve a problem, but brings an improvement regarding learning the practical side of Operating Systems. As far as I remember, the laboratories of this faculty course could have been much more practical and interactive. Objectively, the teacher didn’t have enough time to thoroughly check what every coded until the end of the class.

Moreover, my application not only contributes to solving this issue, but it also offers other facilities to students. First of all, they get instant feedback for their proposed solutions. Secondly, they learn new ways of solving problems by learning from other students. Even though a student doesn’t know the answer or answer incorrectly, he or she can see the solution proposed by the question’s author.

The following chapters describe how my web application works and how to configure and deploy it.

# II. Configuring the server/servers

## General description

Initially, I thought about using the faculty’s server so the application would run the code on the students’ accounts. However, on further discussion with my scientific coordinator, I realised that uploading the question’s author code on the students’ account could lead to unwanted leaks and possible exploits. Therefore, I configure my own Linux machine for the application’s particular needs. I also added here instructions on how to configure the database server as well as the web server.

In previous versions, my application created a separate account on the Linux machine for every student using it to prevent write and execution conflicts. It was pretty costly to do this and also adjustments to users’ rights were required. However, after implementing Docker containers, one account is enough because I run the code in separate containers. By treating the code as plain text prior to mounting it in Docker, the overall security of the application improved drastically.

After modifying the web application to work with containers, I also no longer need to limit the amount of storage memory a student can use through quotas. This is a good thing because it didn’t work for users with only a number as username. They need at least one letter.

Docker was not the first virtualization solution I tried in order to improve the stability and security of the system. Before this, I tried the “chroot” command.

## Configuring the Linux machine used for execution of code

First we make sure that the Linux distribution is up-to-date

$ sudo apt-get update

$ sudo apt-get upgrade

Now we install and start the SSH service to connect remotely to the Linux machine and execute code.

$ sudo apt-get install openssh-server openssh-client

$ sudo service ssh start

Now we install Docker.

$ sudo snap install docker

### Configuring Docker

First, we create a group called “docker” so running a container can be done without root privileges. The $USER represents the account on the Linux machine which will be used through SSH.

$ sudo groupadd docker

$ sudo gpasswd -a $USER docker

We can use the “my\_ubuntu” image which I preconfigured and attached to the application package. Or we can configure a vanilla Ubuntu image for our needs:

$ docker pull ubuntu

$ docker run --name my\_ubuntu -it ubuntu bash

Now that we have an interactive shell opened, we install the “strace” command and thte gcc compiler. Note that we don’t need to use “sudo” anymore because we are logged in as root.

$ apt-get update

$ apt-get install strace

$ apt-get install gcc

$ exit

After installing those packages, we need to create an image out of that container.

$ docker commit my\_ubuntu

$ docker images

Copy the IMAGE ID of the first image from the list, with the “<none>” tag. We will change its name to “my\_ubuntu”.

$ docker tag $IMAGE\_ID my\_ubuntu

We don’t need anymore the container “my\_ubuntu” so we remove it. Note that the newly created image “my\_ubuntu” will be kept.

$ docker rm my\_ubuntu

Note that the web application may not be able to find Docker location. To solve this, you need to add its location in “/etc/environment”. The location of Docker and be found using $ which docker. In my case, I added “:/snap/bin” (ignore “”) at the end of string.

## Configuring the Apache server

This can run any operating system as long as it has the required libraries for my web application. Unfortunately, I couldn’t properly install the ssh2 PHP library on Windows, but I managed it instantly on Linux. Below I explain how I did it.

### 3.1. Installation

$ sudo apt-get install apache2

$ sudo apache2ctl configtest

$ sudo ufw allow in "Apache Full"

$ 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

$ sudo a2enmod rewrite

$ sudo service apache2 restart

If the 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

Note that the group “www-data” needs read and write permissions for the entire “mvc/app/” directory.

### 3.2. The JSON configuration file

It is located in mvc/app. This file contains some configuration parameters for the web application which are necessary to be correctly set within the deployment process.

First of all, the “app/local\_path” parameter refers to the location of the “mvc” directory, which contain all the web application components.

The “external\_shh” entry refers for example to the Faculty of Computer Science server which will be checked via SSH to make sure that a user who attempts to connect is indeed a student. The “check” parameter MUST be se to “true” when the application is deployed. In case of development purposes, it can be set to “false”. The other two parameters (“host” and “port”) are intuitive.

The “ssh” entry refers to the Linux machine used to execute code. The first two parameters (“host” and “port”) are intuitive. The next two parameters (“sudo\_user” and “sudo\_pass”) are credentials for a user who can use the sudo command on that Linux machine. The “timeout\_seconds” paramtere refers to the maximum amount in seconds a code can executed before being forcefully terminated. To disable this feature of the application, set it to “0”.

The “db” entry refers to the MySQL database server. The first parameter (“host”) is intuitive. The next two parameters (“user” and “pass”) are credentials for a user who can select, update and insert into the database. The last parameter (“name”) refers to the database name on the database server.

## Configuring the MySQL server

### 4.1. Installation

It can be installed on any operating system. I describe bellow how we install in on Linux.

$ sudo apt-get install mysql-server

$ sudo mysql\_secure\_installation

### 4.2. Creating a new user[[2]](#footnote-2)

It is a known fact that you don’t use the root user to access a database. Therefore, we need to create a normal user and to give it privileges. I describe bellow how we create a new user on Linux.

$ sudo mysql -u root

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

### 4.3. Importing the database

For the import I use the phpMyAdmin’s web interface. Go to Import>File to import and choose AplicatieSO.sql.

# III. Requirements

The current versions of the components my web application uses are listed below. They were the latest at the time of writing this. I didn’t test my application on any other versions.

Ubuntu: 18.04.2 LTS

Docker: 18.06.1-ce, build e68fc7a

MySQL database: 5.7.26-0ubuntu0.18.04.1 for Linux on x86\_64

Apache: 2.4.29

PHP: 7.2.19-0ubuntu0.18.04.1

OpenSSH: OpenSSH\_7.6p1 Ubuntu-4ubuntu0.3, OpenSSL 1.0.2n

GCC Compiler: Ubuntu 7.4.0-1ubuntu1~18.04.1

# IV. Types of users in the web application

1. Guest user
   1. Can access the authentication page
   2. Can log in
2. Normal user (student)
   1. Can do anything the above user can do
   2. Can view announcements
   3. Can solve questions
   4. Can report questions
   5. Can post questions according to set criteria
   6. Can delete his or her own questions according to set criteria
   7. Can view own posted questions in “My Questions” page
   8. Can view the message and the date of the report for those questions
   9. Can log out
3. Administrator (professor)
   1. Can do anything the above user can do
   2. Can post announcements
   3. Can delete announcements
   4. Can post questions unconditionally
   5. Can view all posted questions in “All Questions” page
   6. Can view all deleted questions in “All Questions” page
   7. Can access the administrator page (URL: mvc/public/admin)
   8. Can add administrators by user name
   9. Can remove administrators
   10. Can post chapters
   11. Can unpost chapters

# V. Authentication process

## Components

### 1.1. The MySQL database server

The username used for authenticating on my web application will be the one used to authenticate on an external SSH connection. It will be stored in the MySQL database server.

The password used for authenticating on my web application will be the one used to authenticate on an external SSH connection. Its hash, generated using the PHP default hashing algorithm[[3]](#footnote-3), will be stored in the MySQL database server.

### 1.2. The external SSH connection

For example, this is the server of the Faculty of Computer Science. This connection can be configured on the JSON configuration file. It is used to verify that a user is indeed a student of the faculty.

Even if this server is not online, the student will still be able to authenticate in most cases (see below scenarios) because the credentials are stored in the MySQL database server. The only thing that becomes impossible is the account creation process, if the application is configured to use this server, due to the impossibility to verify if a user is indeed a student.

## Scenarios

### 2.1. The student doesn’t have an account on my web application and the flag to check the external SSH connection is set to true.

My web application checks the MySQL database server in case there exists an account for that username. It will not find anything, so it initiates the account creation process:

My web application checks the external SSH connection to verify the credentials. If they are correct, a new account is created in my web application and on the Linux machine used to execute commands. The student is then automatically logged in.

### 2.2. The student doesn’t have an account on my web application and the flag to check the external SSH connection is set to false.

My web application checks the MySQL database server in case there exists an account for that username. It will not find anything, so it initiates the account creation process:

My web application doesn’t check the external SSH connection to verify the credentials. A new account is created in my web application and on the Linux machine used to execute commands. The student is automatically logged in.

### 2.3. The student does have an account on my web application, the flag to check the external SSH connection is the to true and the password is incorrect

My web application checks the MySQL database server in case there exists an account for that username. It will find the account and it initiates the authentication process:

My web application checks the password’s hash stored in the MySQL database to verify the credentials. The password is incorrect, so the application verifies the credentials using the external SSH connection. If the credentials are correct, a new hash is stored in the MySQL database and the student is logged in. Otherwise, an error message is displayed.

### 2.4. The student does have an account on my web application, the flag to check the external SSH connection is the to false and the password in incorrect

My web application checks the MySQL database server in case there exists an account for that username. It will find the account and it initiates the authentication process:

My web application checks the password’s hash stored in the MySQL database to verify the credentials. The password is incorrect, but the application doesn’t verify the credentials using the external SSH connection. A new hash is stored in the MySQL database and the student is logged in.

### 2.5. The student does have an account on my web application, the flag to check the external SSH connection is the to true and the password is correct

My web application checks the MySQL database server in case there exists an account for that username. It will find the account and it initiates the authentication process:

My web application checks the password’s hash stored in the MySQL database to verify the credentials. They are correct, so the student is logged in.

# VI. Back-end mechanisms

## Administrator panel

As the name implies, it can only be accessed by administrators at “public/admin”. There an administrator can grant or remove the administrator rights from other users. The administrators can also modify the status of chapters from posted to unposted or from unposted to posted.

## Managing questions

### 2.1. Answering questions

To answer questions, got to Choose Chapter and select one. While solving questions, you won’t come across any questions of yours or any questions marked as “Invalid” by an administrator. Even though the questions for that particular chapter are selected at random, you won’t get the same question twice in a row.

After you hit “Submit”, you will get to see the arguments, input file and the keyboard input of the author where it is the case. In case you answered correctly to the question, you will also get to see the author’s code. Otherwise, you can spend 3 correct answers to reveal it. This is useful especially if the problem seems impossible to you.

### 2.2. Answers assessment while solving questions

The correctness of an answer is based on the output the code gives, be it console or file. In case you only click “Execute”, both sources of output will be taken into consideration as follows: console output will be checked first and if it is missing, then the output file will be checked. However, in case you click “Submit”, you need to stick to the author’s specification’s regarding output. If the author specifies to write the output in a file, you need to do it like this. Otherwise, your answer won’t be taken into consideration. The same applies when the author demands console output.

### 2.3. Reporting questions

After you submit an answer, you have the choice to report a mistake in the question. This can mean the text or the code. The message of the report will be visible to the question’s author, but not the user who reported it. So a report is anonymous, unless an administrator checks it.

### 2.4. Posting questions

The process of posting questions is divided per chapter. The default criteria to post the first questions is to have answered correctly to at least 10 questions from that chapter. The second question posted requires the same number of corrects answers. After that, the cost gets multiplied by 2. In other words, 20 correct answers for the third question, 40 for the fourth one etc.

To change the default formula, change the code from mvc/app/models/Forumlas.php. Be careful to update the other formulas as well.

Questions which can’t get a consistent output after two consecutive runs won’t be accepted. This mechanism will be triggered by random numbers generations, time displaying functions etc. An error message regarding the non-determinism of the code will be displayed.

$this->execute($code,$args,$keybd,$input,true);

$aux\_output=$\_SESSION["exec\_msg"];

$this->execute($code,$args,$keybd,$input,true);

if(strcmp($aux\_output,$\_SESSION["exec\_msg"])!=0){

$exec\_msg=$this->session\_extract("exec\_msg",true);

$this->reload("Code is not deterministic!");

}

The metadata of a question, such as the chapter it belongs to, author, times attempted, times answered correctly, reports, date uploaded etc. are stored in the MySQL database. The text, code, arguments, input file, keyboard input of the question are stored in files located in mvc/app/questions. They have as name their id in the MySQL database. Their extensions are .code (what needs to be executed), .text (question’s requirements), .args (arguments), .keybd (keyboard input), .input (input file contents).

The uploading process has 3 steps. The first one consists in making an insert in the questions table with the status “pending”. Then, that entry’s id is read and the question’s files are created. When this is done, the status of the question in the database is changed to “posted”.

### 2.5. Deleting questions

The deleted questions are not removed from the system, but marked as “deleted” in the MySQL database. I chose to keep the deleted questions in case of a system error or unintentional operation. An administrator can restore any deleted question. This will update the number of posted and deleted questions for the user, consequently also updating the number of available correct answers.

If a question is deleted, the number of posted questions in the particular chapter for the user decrements and he or she gets back the spent right answers to post it. In other words, the user can immediately post again a question with only a penalty of 10 right answers. This measure greatly lowers the risk of reposting spam, thus, wasting storage space in the system.

### 2.6. Validating questions

Only the administrators can validate questions. This mechanism ensures a filtering of inappropriate questions which otherwise would get shown to students in the application. The validation status of a question can be updated as many times as necessary.

The default validation state is “Unvalidated”, which means that it has not been reviewed by an administrator yet. The administrators can mark a question as “Valid” or “Invalid”.

## Code execution on the Linux machine

### Preparation

The web application first writes the code and other necessary information in dedicated local files with specific extensions. They are located in “mvc/app/scp\_cache” and their unique names correspond to the user id who initiated the operation.

### Establishing the connection and sending the files

First we make use of the ssh2\_connect() method to connect to the Linux machine using a given $host and $port, with a default value of 22. It returns a resource which we will use in the authentication process.

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

In order to authenticate a user though SSH, we use the ssh2\_auth\_password() which takes as parameters the previous resource, a username and a password. In case the process was successful, it returns true. Otherwise, it return false.

ssh2\_auth\_password ( resource $session , string $username , string $password ) : bool

This method (ssh2\_scp\_send()) is able to send a $lolca\_file to the Linux machine identified by the $session variable using the Secure Copy Protocol. If the operation is completed successfully, it will return true and false otherwise.

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

### Executing the code

The ssh2\_exec() method is used to send a bash command for execution through SSH. It takes as arguments the resource returned by the ssh2\_connect() method and the command.

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

On top of that, I use the Docker functionalities, namely containers. The bash command I send for execution looks like this:

docker run --name " . $this->session\_user . " -v $(pwd)/" . $this->session\_user . ".sh:/code.sh -v $(pwd)/" . $this->session\_user . ".keybd:/code.keybd:ro -v $(pwd)/" . $this->session\_user . ".input:/code.input -v $(pwd)/" . $this->session\_user . ".output:/code.output -v $(pwd)/" . $this->session\_user . ".run:/code.run:ro --rm my\_ubuntu bash ./code.run

The “--name” argument means that the container will also have a name, that of the user initiating the code execution operation. The “-v” argument is used to mount a file in the container. The path before “/” represent the host’s files path and the path after “/” represents where it will be mounted in the container. Adding “:ro” makes the mounted file read-only. I change the unique name of the file stored on the host system to a generic “code.<extension>” so that it is easier to identify the files in container. The argument “--rm” tells the container to delete itself after executing the code. Otherwise, it would store the changes made. Instead of using a default ubuntu image, I use the “my\_ubuntu” image. The .run file I execute contains the following code:

chmod +x code.sh && ./code.sh $ARGUMENTS < code.keybd

First, I give the script file (.sh) the right to execute. If it was done successfully, I run it using the arguments from $ARGUMENTS if there are any. I am also able to feed the script the keyboard input stored in “code.keybd”. If I wouldn’t have a used an intermediate file (“code.run”), there would exist the risk of code injection through $ARGUMENTS, which are given by the user in the web form.

In order to get the output or errors after execution we need the ssh2\_fetch\_stream() method. The stream id can be SSH2\_STREAM\_STDIO or SSH2\_STREAM\_ERR. The stream\_get\_contents() returns the string.

ssh2\_fetch\_stream ( resource $channel , int $streamid ) : resource

stream\_get\_contents ( resource $handle [, int $maxlength = -1 [, int $offset = -1 ]] ) : string

### Checking system calls

For this I use the “strace” command. The “.run” file for the C programs which need to use forking looks this:

gcc code.c -o code.out && (strace -e trace=clone ./code.out $ARGUMENTS < code.keybd)

The strace commands looks for system calls involving process cloning. If no system call of this kind if detected during execution, an error message will be displayed indicating that the student did not use fork()in his/her code. To use the “strace” command in docker you need additional parameters:

docker run --cap-add=SYS\_PTRACE --security-opt seccomp=unconfined --security-opt apparmor=unconfined …

Another solution is to simply add --privileged, but it is recommended to avoid it.

### How the application limits the execution time

I use the “timeout” command to limit the execution of the “docker” command. Like this, students will not be wasting CPU resources on purpose or by mistake though an infinite loop. Moreover, without this time limit, the PHP lock (I will cover this topic later) would produce a deadlock for that particular student. Below is an example from PHP:

ssh2\_exec($this->connection, "timeout --signal=SIGKILL " . $ssh\_timeout\_seconds . " " . $docker\_command);

This mothod will trigger a pkill bash command after a given number of seconds ($ssh\_timeout\_seconds). The KILL signal which cannot be bypassed, unlike TERM.

## User authentication and authorization

For a user to be considered logged in, the ‘user’ field in the $\_SESSION variable needs to be set. Otherwise, the user is redirected to the login page.

if(isset($\_SESSION['user'])==false){

$this->view('home/login',['error\_msg' => $error\_msg]);

die;

}

The login process retrieves from the database the account’s details and stores them in the $\_SESSION variable. Those account’s details consist in the user id, user name and whether the user is an administrator.

The PHP code contains private functions, which can only be accessed by the Apache server and public ones which can be accessed by users via the URL. In case of the latter, additional steps are added to authenticate and authorise a user. I won’t talk about the form input sanitization I did because this is standard for any web application.

For any public function, first I check if the user is logged in and copy his or her details stored in the $\_SESSION variable into local variables. This prevents the user from initiating an action and quickly logging out so no account details get stored in the database for that particular action. For example, a user may write anonymous reports to questions if it wasn’t for this additional step.

If it is the case, I also check if the chapter a user wants to access is posted. Non-administrator users will not be able to access unposted chapters.

$this->check\_login();

$this->check\_chapter\_posted(self::CHAPTER\_ID);

$this->my\_sem\_acquire($this->session\_user\_id);

A non-administrator user also will not be able to view other users’ questions or deleted questions in the “view\_question” section of a chapter. Of course, the administrators will be able to see all the questions for a certain chapter and also perform certain operations on them unrestricted.

## PHP semaphores

The PHP semaphores also play a crucial role in the security and the stability of the web application.

$semaphore\_key = $user\_id;

$semaphore\_max = 1;// The number of processes that can acquire this semaphore

$semaphore\_permissions = 0666;// Unix style permissions for this semaphore

$semaphore\_autorelease = 1; // Auto release the semaphore if the request shuts down

//open a new or get an existing semaphore

$this->semaphore = sem\_get($semaphore\_key, $semaphore\_max, $semaphore\_permissions, $semaphore\_autorelease);

if(!$this->semaphore){

die( "Failed to get semaphore!");

}

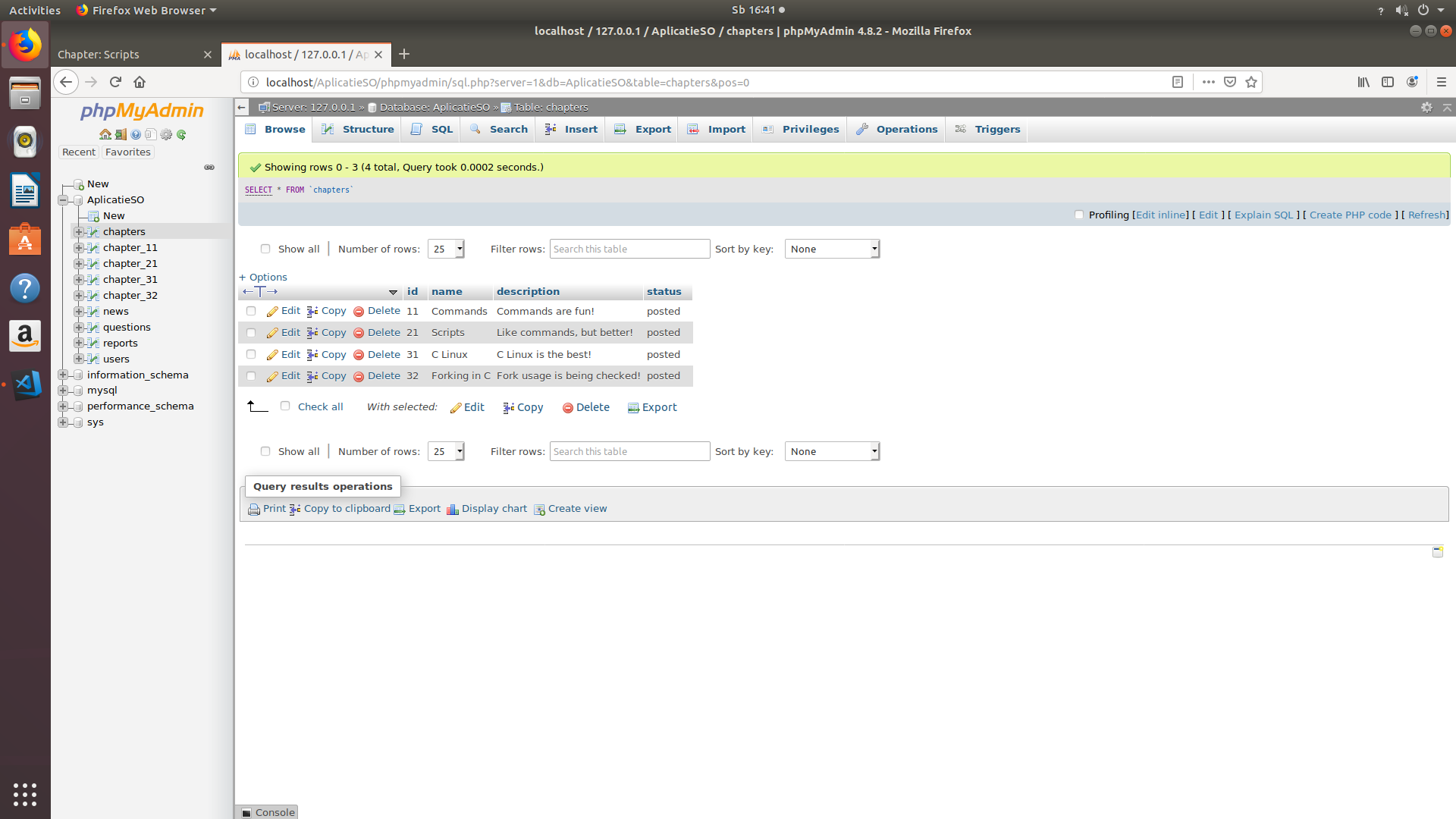
//acquire exclusive control

sem\_acquire($this->semaphore);

When a user performs an operation which is not thread-safe, the semaphore is used. The user won’t be able to use the same resource (the MySQL database or the SSH connection) from multiple web pages. Even logging in on two separate computers won’t bypass this intentional limitation because the semaphore uses the unique id of the user, not the session id. This prevents the user from quickly initiating multiple question posting operations before the system revaluates the eligibility of the user to post another question. The other user are not affected by the other users’ activity, as far as PHP semaphores are concerned.

# VII. Adding a new chapter

## 1. MySQL Database



1

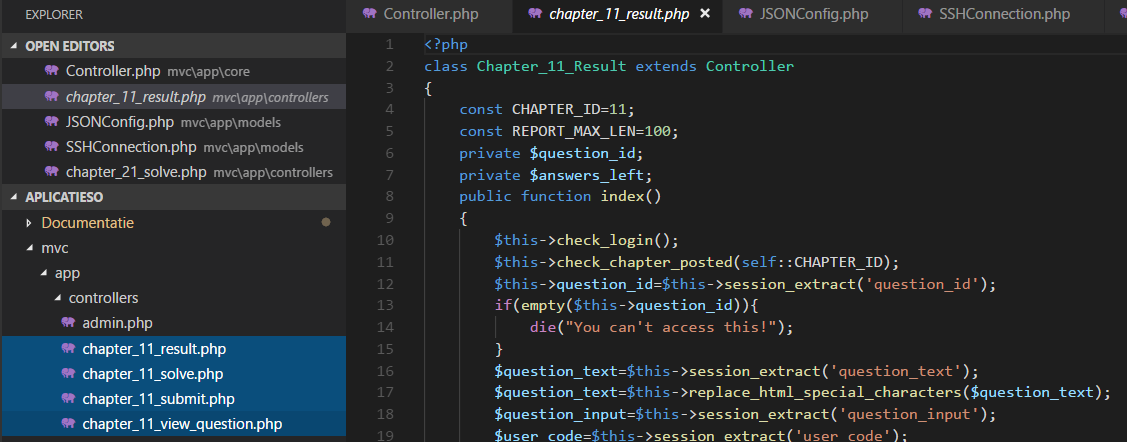
2

1. You navigate to the “chapters” table in the database.

2. You insert a new row with an id (first digit - chapter, second digit - subchapter), a name, a short description, and the chapter’s status. It can later be modified from the administrator page.

## 2. Apache server

### 2.1. Controllers



2

3

1

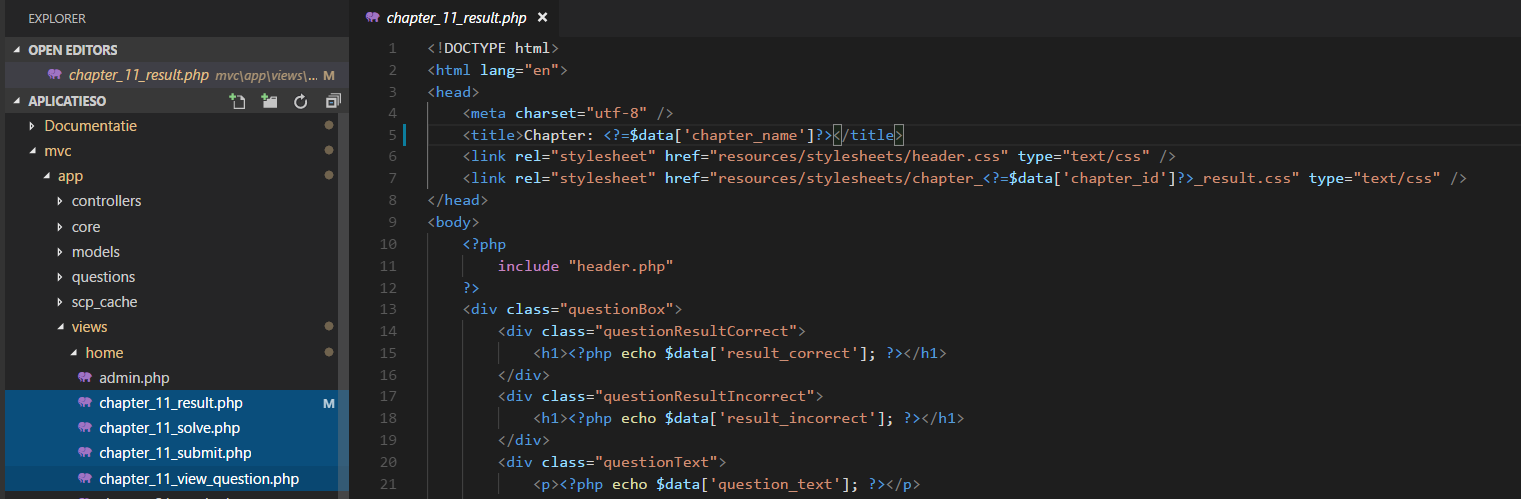
1. Create copies of the 4 files with specific sections (result, solve, submit and view\_question) for a given chapter from “mvc/app/controllers”. Change the id from their name (in this case “11”) to the corresponding id of the new chapter.

2. Rename the class from each copy of the files by changing the id to the new one.

3. Change the “CHAPTER\_ID” constant to the new one. You can also update the value for the other constants if you wish.

After you finished doing the essential previous steps, you change the files’ code to suit your needs.

### 2.2. Views



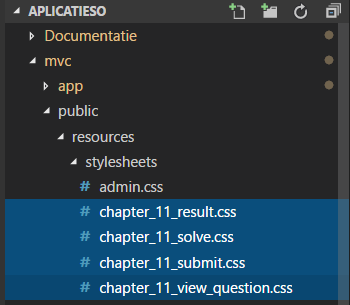
2

1

1. Create copies of the 4 files with specific sections (result, solve, submit and view\_question) for a given chapter from “mvc/app/views”. Change the id from their name (in this case “11”) to the corresponding id of the new chapter.

2. It is not crucial to modify the content of those files because the html for the web pages is generated dynamically by the controllers.

### 2.3. Stylesheets

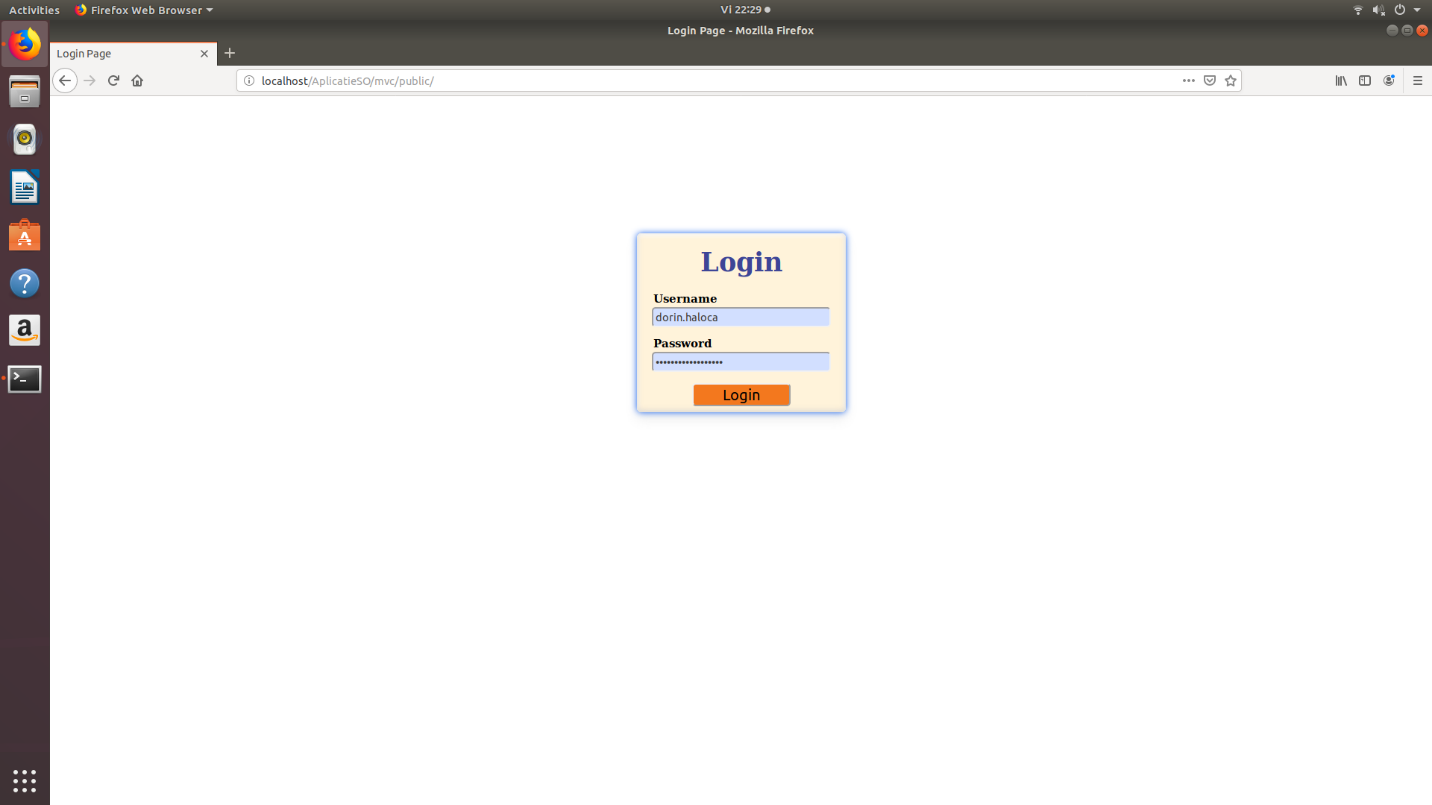
1. Create copies of the 4 files with specific sections (result, solve, submit and view\_question) for a given chapter from “mvc/public /resources/stylesheets”. Change the id from their name (in this case “11”) to the corresponding id of the new chapter.

1

1

# VIII. Interface navigation

When you first access the application, you are prompted to enter your credentials. There isn’t a registration page as the account is created automatically if you don’t have one. Of course, you need to be a student of the faculty to have your account created, if this check is set to true.



2

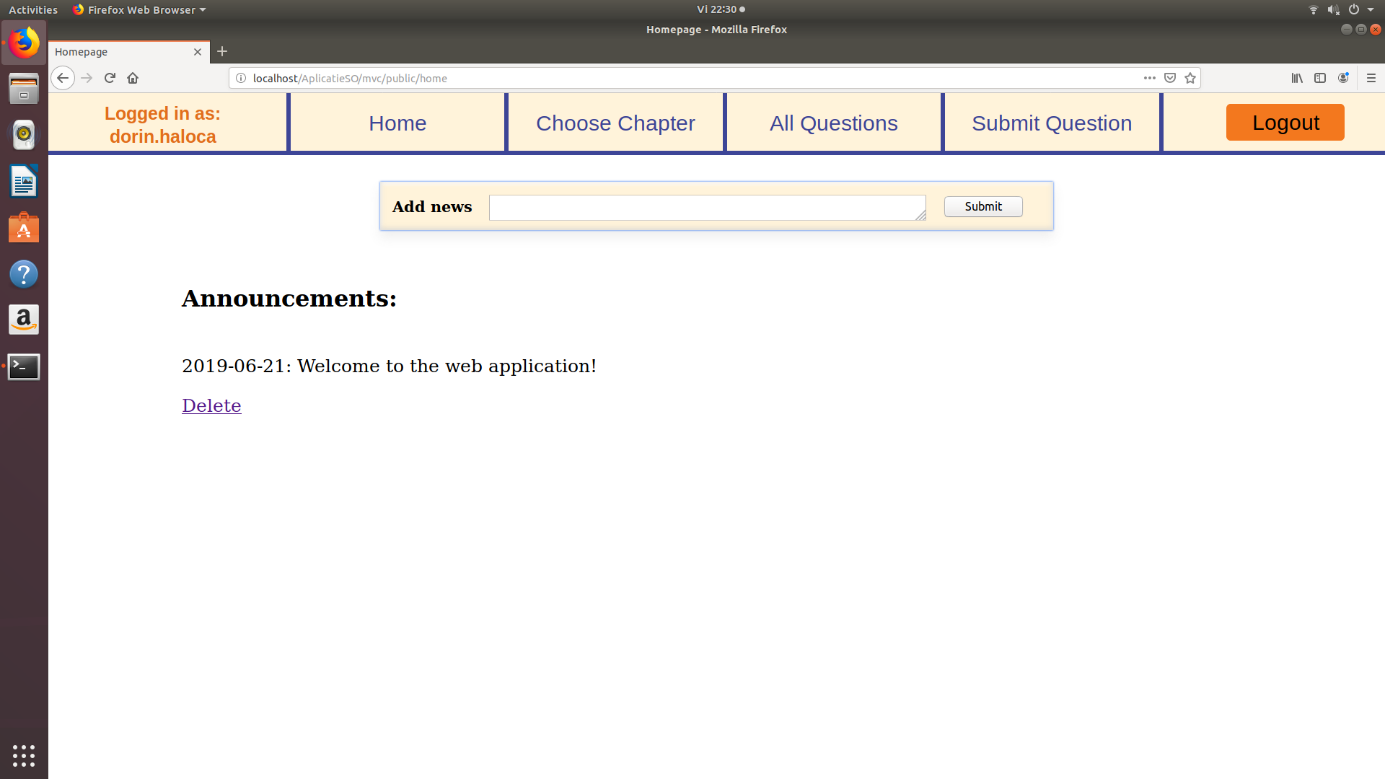
1

1. The credentials fields: username and password

2. The button to submit the credentials

If the credentials are correct, meaning you already have an account or an account was successfully created, you will be redirected to the home page of the web application.

Homepage



6

5

4

3

2

1

1. That is the username which was used to log in. It is displayed during the entire session and on every page of the web application.

2. That part of the header enables the user to navigate on the 4 sections of the web application: Home, Choose Chapter (there you solve questions), All Questions (view all the questions in the web application because dorin.haloca is an administrator; for normal users it is replaced by “My Questions”) and Submit Question (there you post questions in the web application).

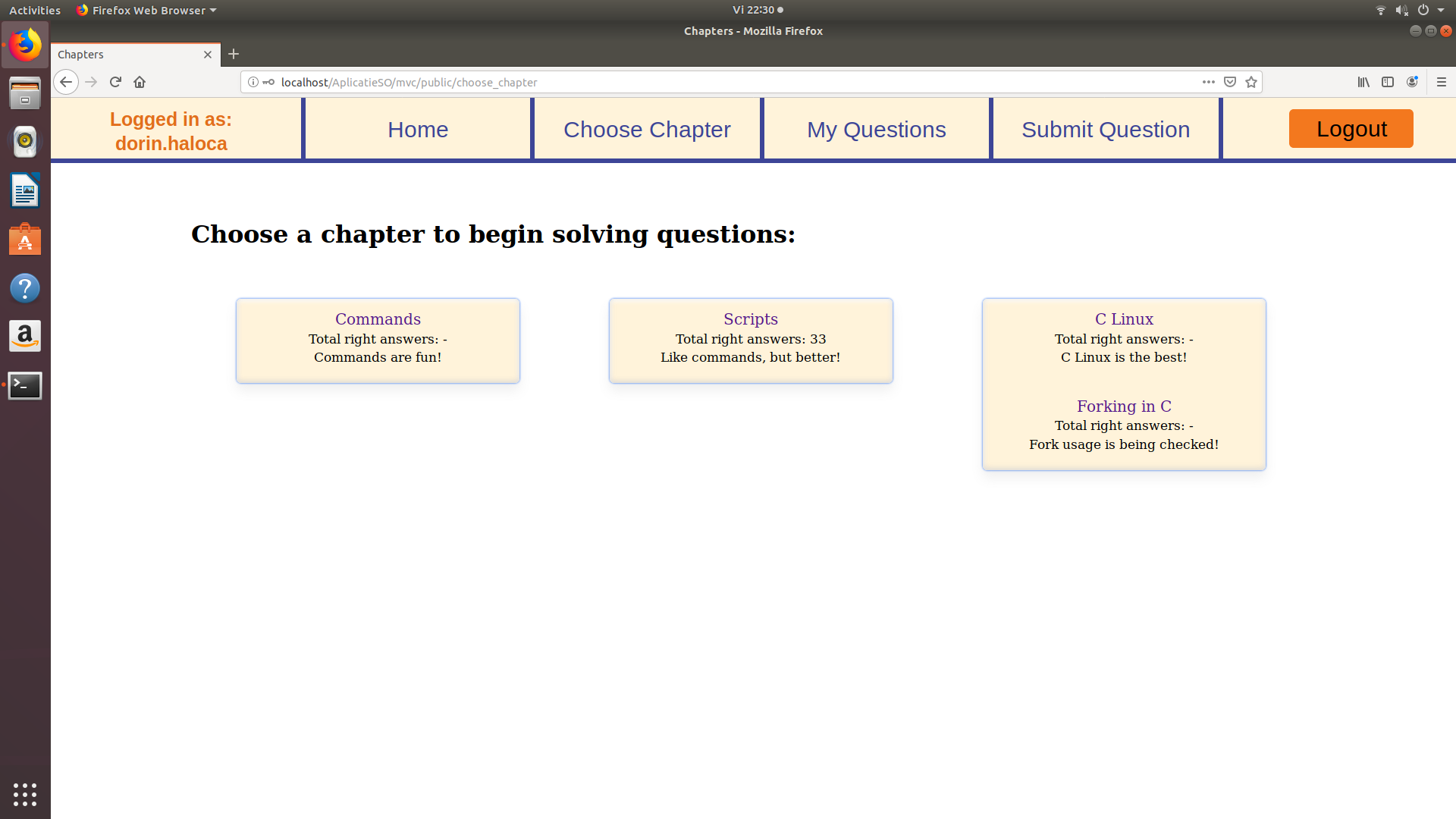
3. That button de-authenticates the user and redirects him or her to the login page.

4. That form is visible only to administrators and it enables them to post announcements on the home page.

5. There is the part of the home page where every user in the web application will see the announcements’ text and date when it was posted.

6. Only an administrator is able to see that button and delete that announcement.

Choose Chapter to solve



2

3

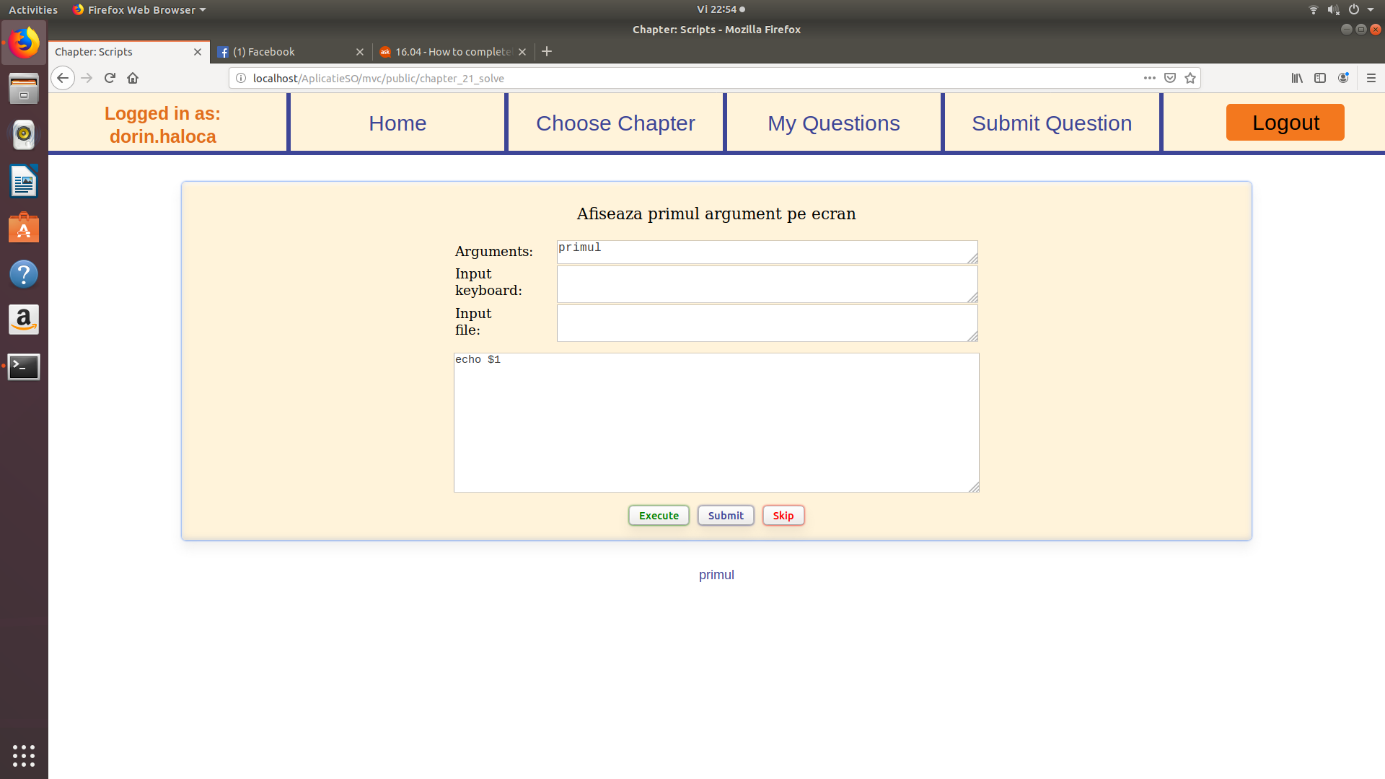
1

1. These are the chapters you can choose from when you want to solve questions. In total, there are 3 chapters: Bash Commands, Bash Scripts and C Linux.

2. A chapter can also have subchapters, like C Linux where you use forking.

3. Each chapter has a short description and a count for the total right answers. Note that this isn’t the number of right answers you can spend to post questions or do something else. Some chapters have a “-” instead of a number because that user didn’t yet access those chapters to attempt to solve questions.

Question solving page



6

5

4

3

2

1

1. That is the title of the chapter you are solving questions from.

2. That is the text of the question. There, the author specifies what to do and how to return the output.

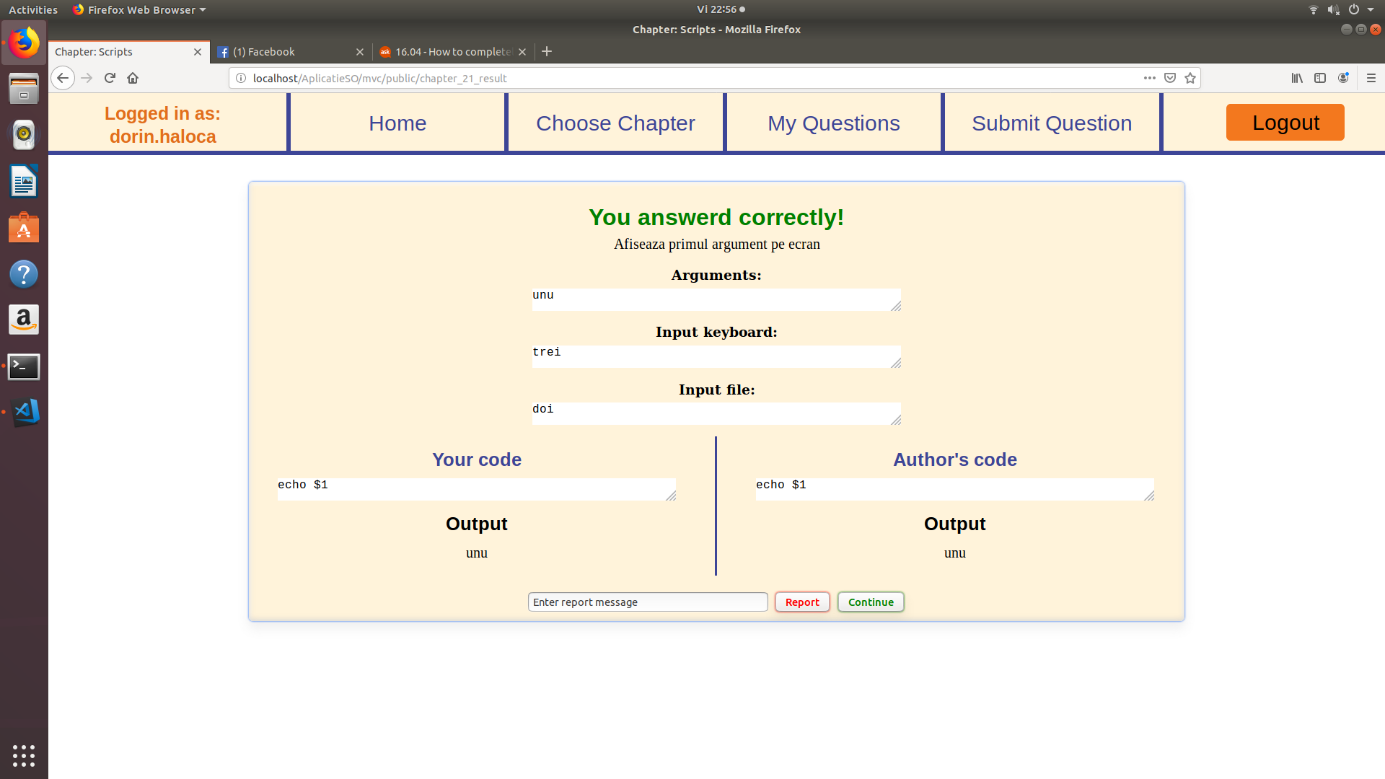
3. Those optional fields are used to feed the data to the code. These are used only for testing your code before submitting. When submitting, the author’s data will be fed to the user’s code. This fields have characters limits.

4. This is the field where you write your code. Needless to say, it is a mandatory field.

5. These are the controls for the form. The “Execute” button only simulates the execution of code with your data in the optional fields at 3. The “Submit” button tests your code, providing it has no errors, against the author’s code. The “Skip” button will give you a new question after a short delay.

6. That is the output of you code, regardless if it’s console output or file output. In case there are some errors, they will be displayed there too coloured in red. In case you see “Killed” in red, it means your code exceeded the time limit to execute and was forcefully terminated.

Result page - incorrect answer



3

2

1

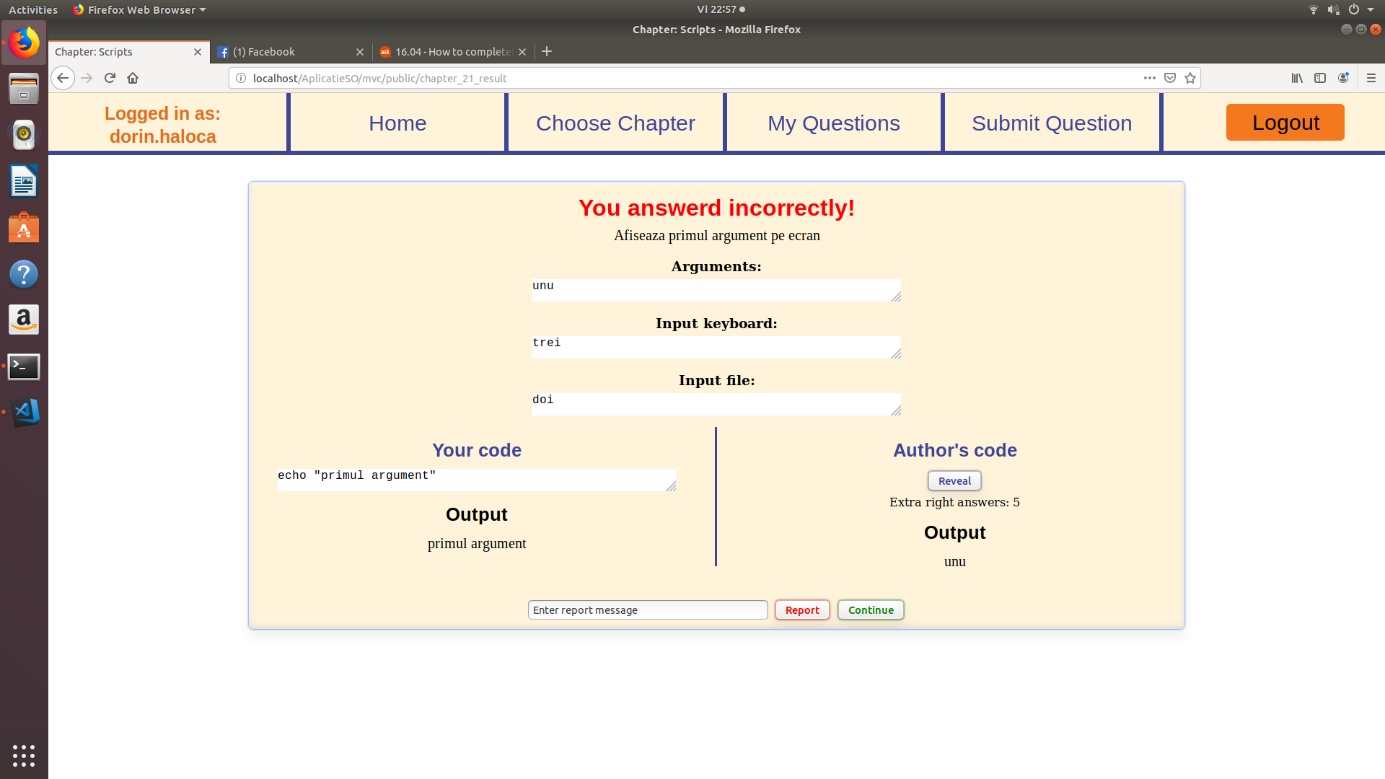
1. That is the author’s data which was fed to both your code and the author’s code.

2. You can report the question in case of a mistake of the author. You can also add a short message in the report field.

3. By pressing “Continue”, you will get a new question to solve.

Note that you can report a question regardless of the correctness of you answer.

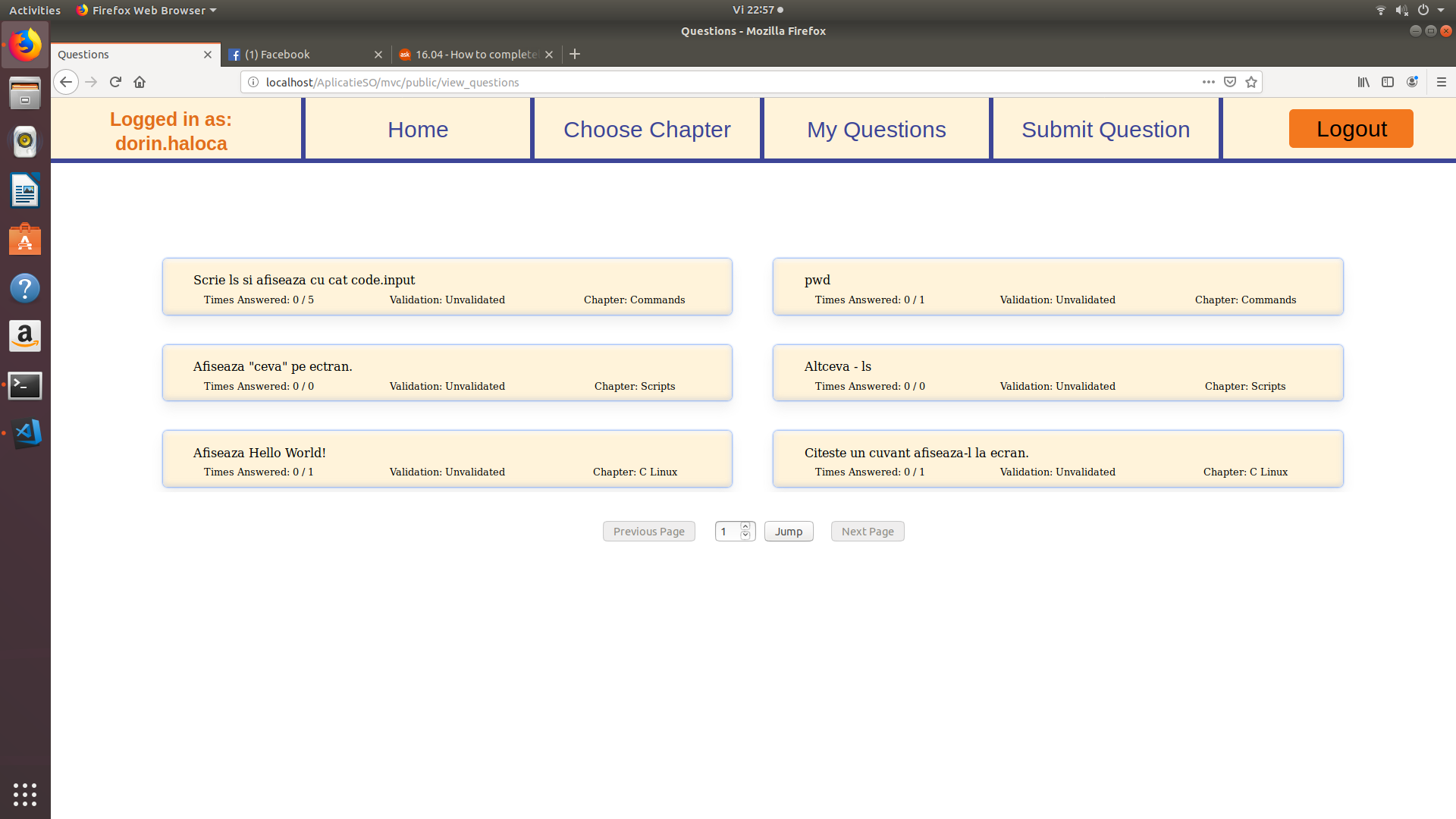
Result page - correct answer



1

1. If you didn’t answer correctly, you won’t see the author’s code by default. In order to reveal it, you need to spend some right answers for that chapter. Below the button, you see the number of available right answers you will have after the reveal.

My Questions



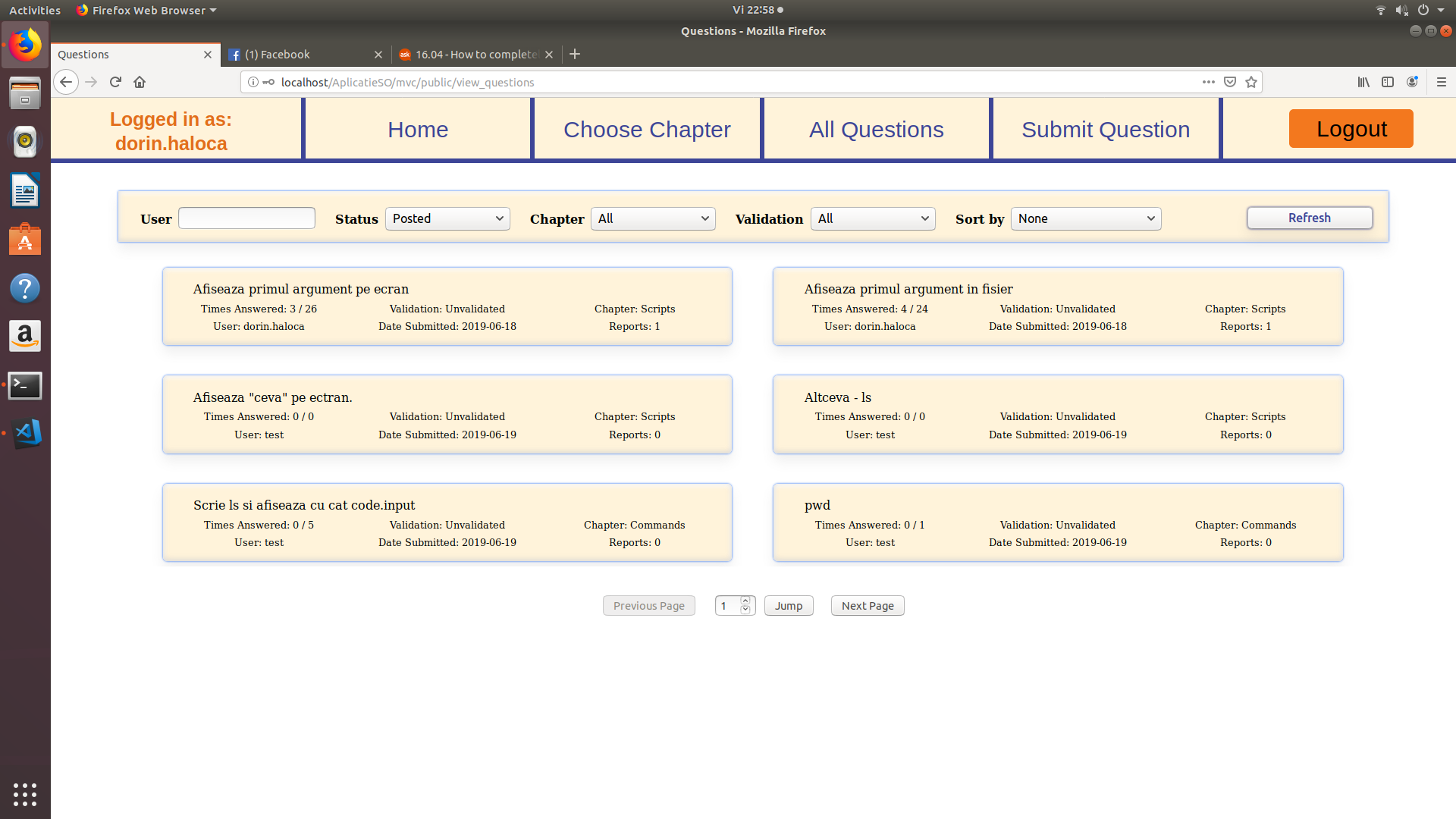
2

1

1. These are 6 of your posted questions in the entire web application. The first row of a cell represents the text of the question. The seconds row contains the number of times it was correctly solved and the number of attempts, the validation status and the chapter it belongs to. You can click on a cell to view additional details and to available actions.

2. These are the page navigation controls. The number field in the middle indicates the current page number. By modifying the value and pressing jump, you can go directly to your desired page. You can also press “Previous Page” or “Next Page” to navigate from page to page.

All Questions



1

3

2

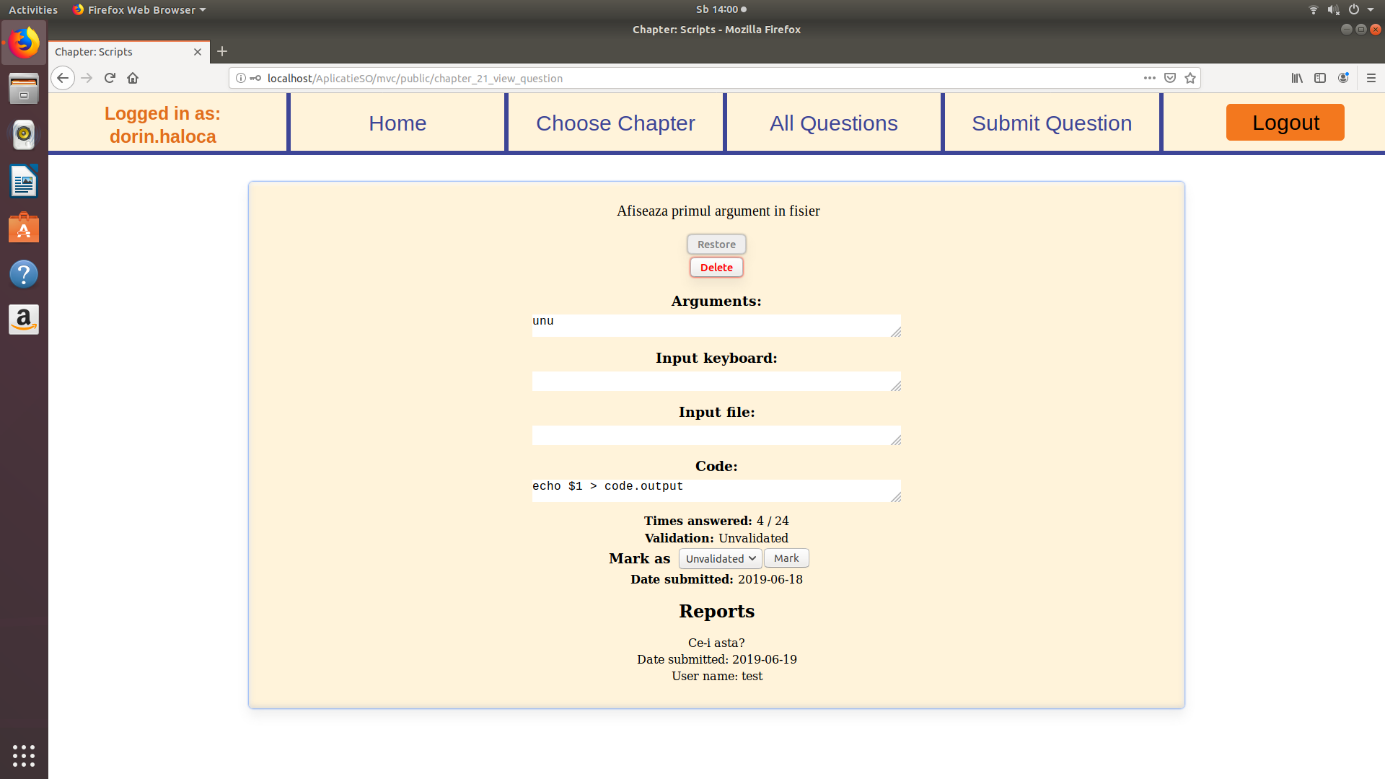
1. Instead of “My Questions”, the administrators have the “All Questions” header button.

2. The administrators can access the filtering menu. It can search questions by username, by status (“Posted” or “Deleted”), by Chapter (“All” or the name of the chapter/subchapter) or by Validation (“All” - all typed of validation, or “None” - unvalidated, “Valid” or “Invalid”). The administrators can also sort by “None” - nothing, “Reports Ascendant” - the question with least reports first, “Reports Descendant” - the question with most reports first, “Date Ascendant” - the first posted question first or “Date Descendant” - the last posted question first.

After the desired criteria are set, you press “Refresh” to refresh the results. The selected criteria will be kept selected in the form.

3. The administrators see an additional row in the preview cells of the questions. Namely, the user who posted the question, the submission date and the number of reports.

Question details - administrator



5

4

3

2

1

1. This is the text of the question.

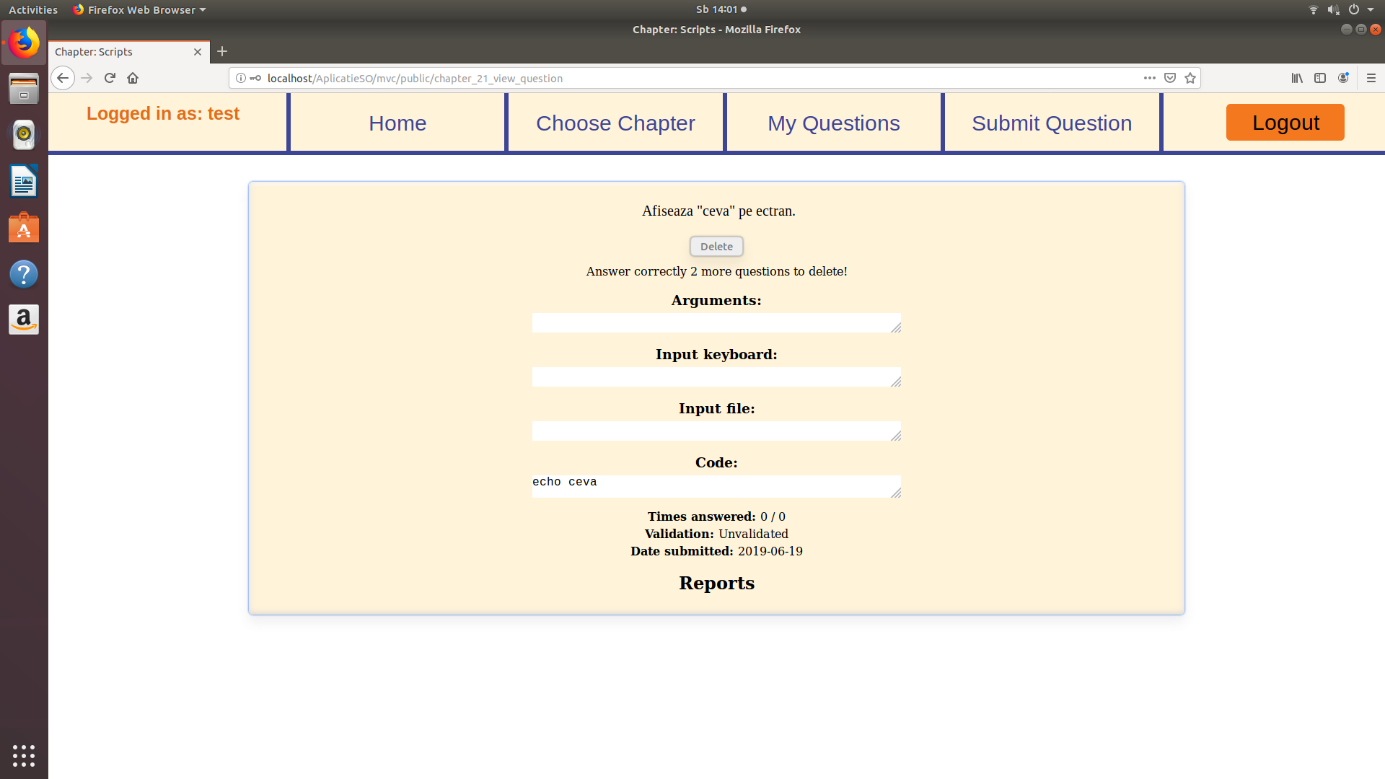
2. The “Restore” button is only visible to administrators. It marks the question as “posted” again if it was deleted. In our case it is not deleted because the button is grey.

3. The “Delete” button marks the question as “deleted”. The administrators can delete a question without any restrictions if the question is not deleted. If the question was deleted, the button would the grey.

4. The administrators have to ability to change the validation state. The normal user can only view the validation state.

5. This is a report from the user “test”. Only the administrator can see who submitted the report. A normal user would only see the first two rows (text and date).

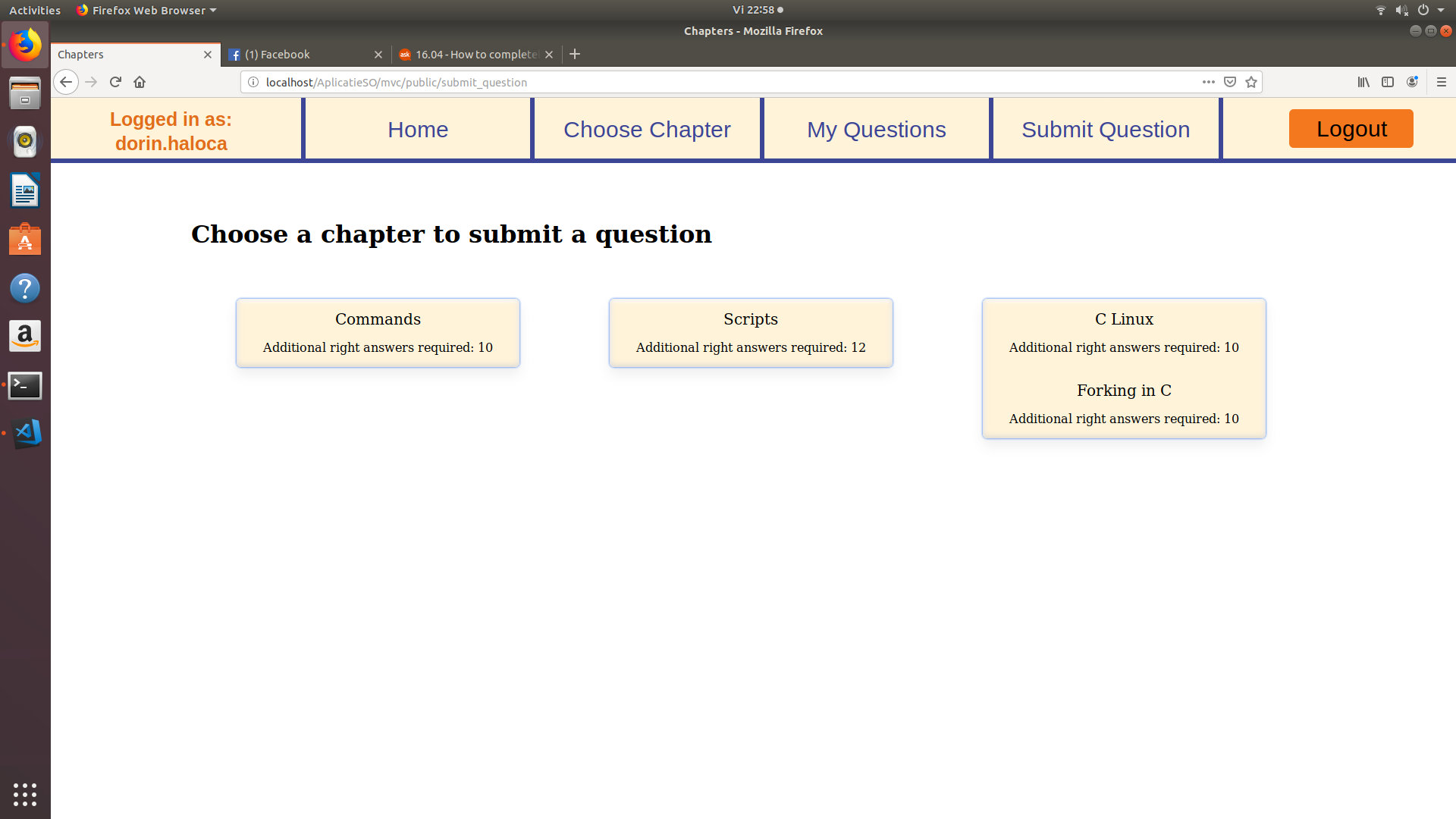
Question details - normal user



1

1. The normal user cannot delete the question yet because he or she doesn’t have the necessary number of right answers. Two more correct answers are required to delete this question. Then, the button will turn from grey to red.

Submit a question for a certain chapter



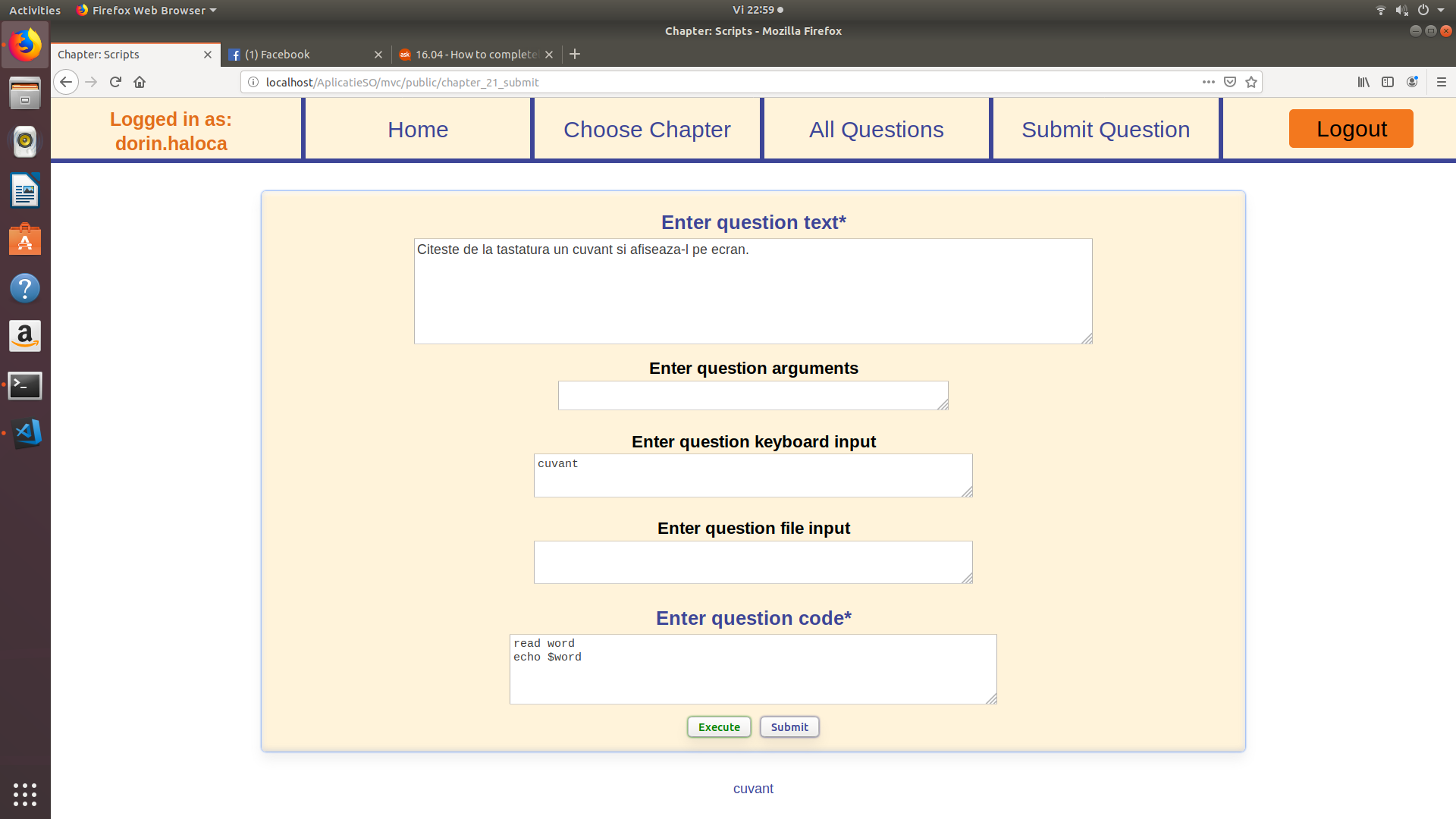
1

2

1. The list of available chapters is the same as for solving questions.

2. For a normal user, that row indication that he or she needs to answer correctly to more questions from that chapter before posting another question. In this case, the additional number required is 10. If the user answered correctly to enough questions, he or she will be able to click the chapter’s name and access the question submitting page. For an administrator, the row would look like “No need to answer questions” because the administrators can post questions unrestricted.

Question submitting page



5

4

3

2

1

1. This is the question’s text field which is mandatory (as indicated by the “\*”). This a maximum characters limit.

2. These are the optional field for the data fed to the code during execution. As you can see here, the only data fed will be the keyboard word “cuvant”.

3. This is the question’s code field which is mandatory as well and has a maximum characters limit.

4. When submitting a question, you can either press “Execute” to simulate the execution of you code for testing purposes, or press “Submit” to post your question in the web application.

5. This is where the output of the code will be displayed, regardless if it’s console output or file output. In case there are some errors, they will be displayed coloured in red. In case you see “Killed” in red, it means your code exceeded the time limit to execute and was forcefully terminated. In case you see “Code is not deterministic!” in red, it means your code doesn’t give the same output during two consecutive executions.

# IX. Deployment of the application

## 1. General information

The web application consists of 3 main components:

- Apache Web Server

- Linux Server - used for execution of code

- MySQL database server

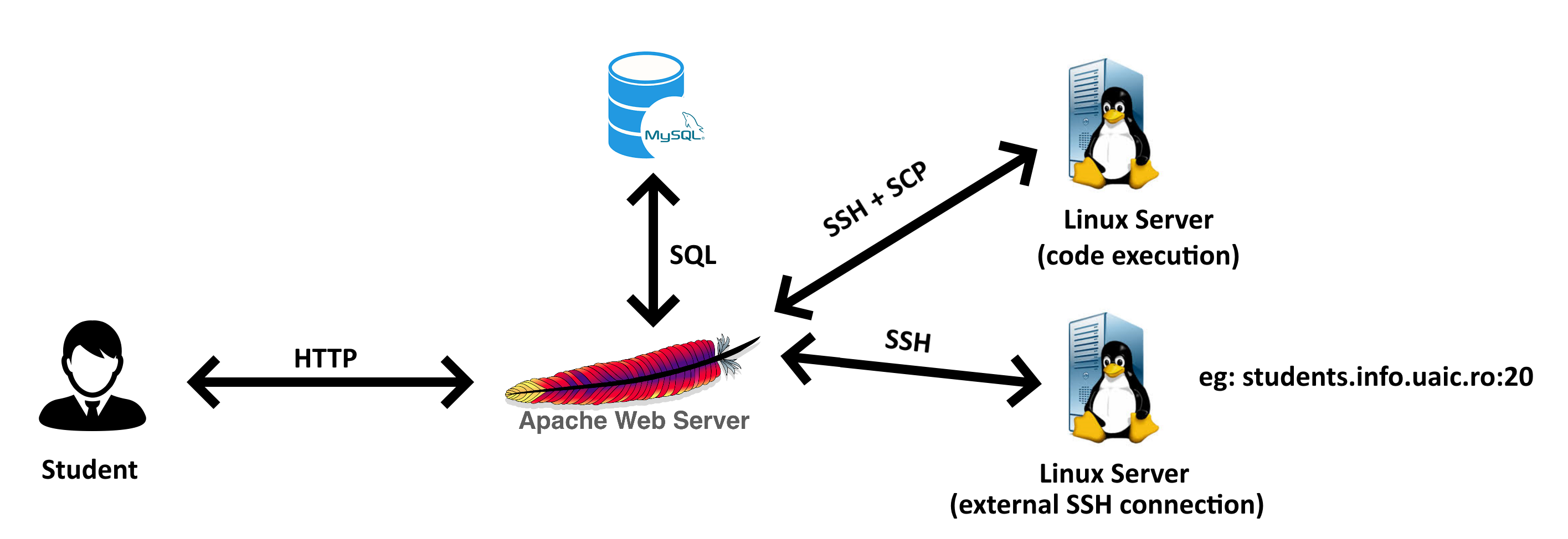
- External SSH connection (optional) - used to verify the identity of students

These components can be placed on a single computer without any problems. However, in case there is too much load for a single computer, the application can be split on 2 or 3 separate computers.

Only the machine which will be accessed via SSH to execute code requires a specific operating system, namely Linux. The other 2 components can be placed on any operating system that supports their operation.

Before deploying the application on the Apache server, read the second chapter (II. Configuring the server/servers) and implement the specified changes to the systems. Then, copy the “mvc” directory from my application’s package to the Apaches web pages’ directory. On Linux, it is the “html” directory of the Apache server, located in /var/www/html.

## 2. Interaction between components



3. How it will be done in the Faculty of Computer Science

The database server used will be the following: <https://students.info.uaic.ro/db/>

The rest of the deployment process is not yet established.

# X. Bugs

1. PHP semaphores in combination with the sleep x instruction in Linux makes the header() function in PHP misbehave. If we have two tabs opened with the chapter solving page, waiting for response from the Apache server, the instruction header() will make both tabs stop waiting, even though one tab was waiting for a response that would arrive later. Therefore, if we execute a sleep 5 and a sleep 10 for the same chapter simultaneously, only the tab with sleep 5 will receive a response while the other one will just stop waiting and receive nothing.

2. Sometimes, when pressing “Execute” or “Submit” for chapters based on C, an error message like “Output cannout be empy!” or “Could not send file execution” may appear. This is because of the SSH connection. Press the button again and it should work fine.

3. Because I use containers, the “ps” command will be pretty much unusable because it output will be almost empty. The same can be said about “who”. Even by modifying /var/run/utmp I couldn’t make it output something unless actual users were connected.

CONCLUSIONS

In the future, the web application should support the secure version of the HTTP protocol to ensure the security of the users’ credentials during the authentication process.

I couldn’t find a solution to enable students to also test commands regarding users, system processes or time. For this, I would need an environment like Docker container but with simulations of those dynamic operations so the students can learn to also manipulate that type of data. In addition, the environment should be initialized exact the same each time it is started so the PIDs are the same and the system clock starts from a predefined moment in time.

As far as I am concerned, I am pleased with how my web application turned out. In the beginning, I didn’t even know if I what I wanted to create was feasible. Only after three days of thorough research and short trials I started working on the real project. Initially, I wanted to do all my work on Windows due to its intuitive UI and my experience with it. However, I couldn’t install all the necessary components on it and so I migrated entirely to Linux, namely Ubuntu. Only one line of bash commands did all the things I couldn’t accomplish after many hours of looking on the internet for the Windows variant.

I learned a lot about about … <DE COMPLETAT AICI>

FLAWS FROM THE DB app I fixed

Even my scientific coordinator is aware that many things can be added and improved to my application and I agree with him. At the moment, the web application is very much usable and suitable for what it was designed to do. I hope that next year a student will choose to continue my work on this useful project.

în această parte a lucrării de licenţă se regăsesc cele mai importante concluzii din lucrare, opinia personală privind rezultatele obţinute în lucrare, precum şi potenţiale direcţii viitoare de cercetare legate de tema abordată. Concluziile lucrării nu se numerotează ca şi capitol.

BIBLIOGRAPHY

1. "*Operating System Concepts*", by Avi Silberschatz, Peter Baer Galvin and Greg Gagne, publisher John Wiley & Sons, Inc., 10th edition, 2018.

2. "*Modern Operating Systems*", by Andrew S. Tanenbaum, publisher Pearson, 4th edition, 2015.

3. <https://www.php.net>

4. <https://stackoverflow.com>

5. <https://linux.die.net>

6. <https://validator.w3.org>

7. <https://brianscode.com>

8. <https://profs.info.uaic.ro/~licenta/2019Jul/res/Ghid-LICENTA-2018.docx>

1. https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller [↑](#footnote-ref-1)
2. <https://stackoverflow.com/questions/39281594/error-1698-28000-access-denied-for-user-rootlocalhost> [↑](#footnote-ref-2)
3. The default PHP hashing algorithm will change in the future according to the latest security requirements [↑](#footnote-ref-3)