contents

General presentation of the topic..... ...2

Operating parameters..... 2

Minimum system requirements for running the website 2

Recommended browsers 2

Instructions for use for the user 3

The login screen 3

The chat page..... 5

Detailed description of the program..... 7

The login screen 7

Entry..... 7

Registration..... 11

The chat page..... 14

logout()..... 15

logout.php..... 15

The selection..... 15

The chat..... 19

Block diagram..... 24

Opportunities for further development..... 24

Librarianship..... 25

General presentation of the topic

The website implements a modern chat application in its design and in its simplicity. Its purpose is to create and/or maintain relationships between people through an online interface, which is very easily accessible to all of us these days. THE website is available to everyone, and you can create an account within a few seconds created.

I used HTML, CSS, PHP and JavaScript languages to prepare the thesis, a taking into account the appearance of the website and the security of user data. Although the page is not uses SQL databases, the user's data is protected. The "Program I talk about this in more detail in the "detailed description" paragraph.

Operating parameters

In order for the website to run correctly, it must be enabled in the user's browser the use of JavaScript and Cookies. In addition to this, from the point of view of technical requirements the website uses minimal computing power compared to today's systems.

The website was created at a resolution of 1920x1080, so it looks best there, but other it also works flawlessly on resolutions.

Minimum system requirements to run this website

- 512 MB RAM memory
- Intel Pentium 1.4 GHz
- 256 MB VRAM memory

Recommended browsers

While writing the website, I used a Chromium-based browser, it works best for them optimized for:

- Google Chrome
- Edge Chromium
- Opera
- Slimjet

- Torch Web

Although the website runs best on the above browsers, the following is non-Chromium based also tested on browsers:

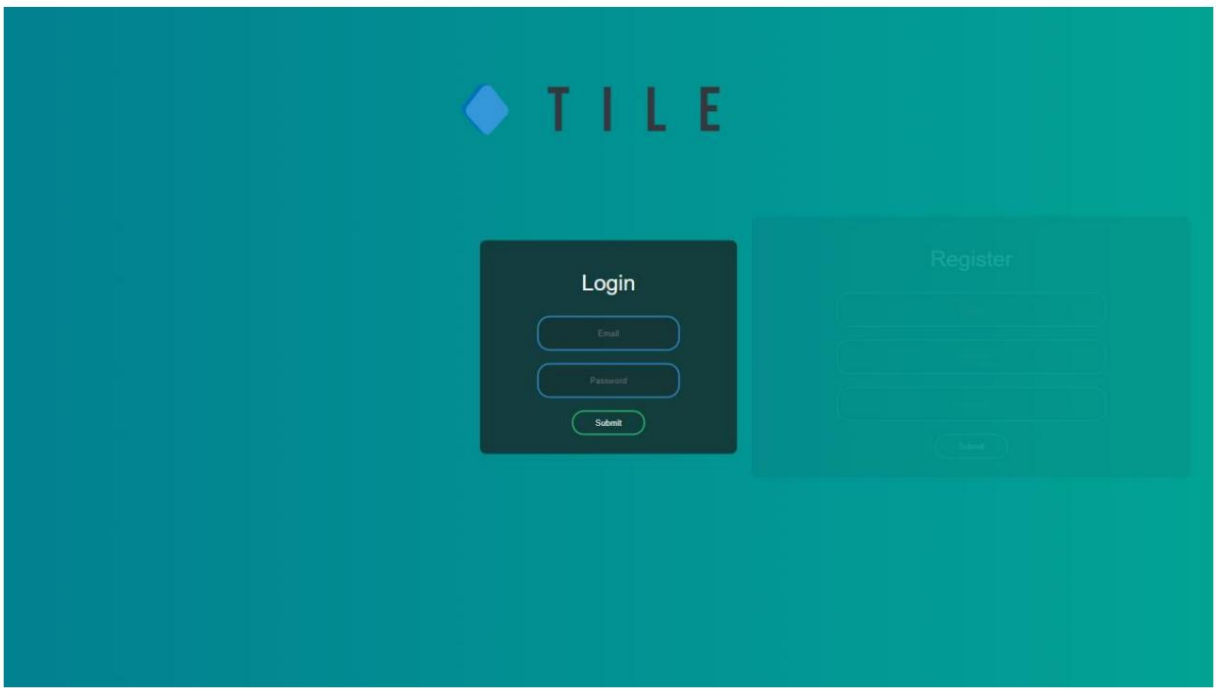
- Mozilla Firefox
- Safari
- Brave

Instructions for use for the user

The website is available at http://david.bartok.ro at any Internet-related address device.

The login screen

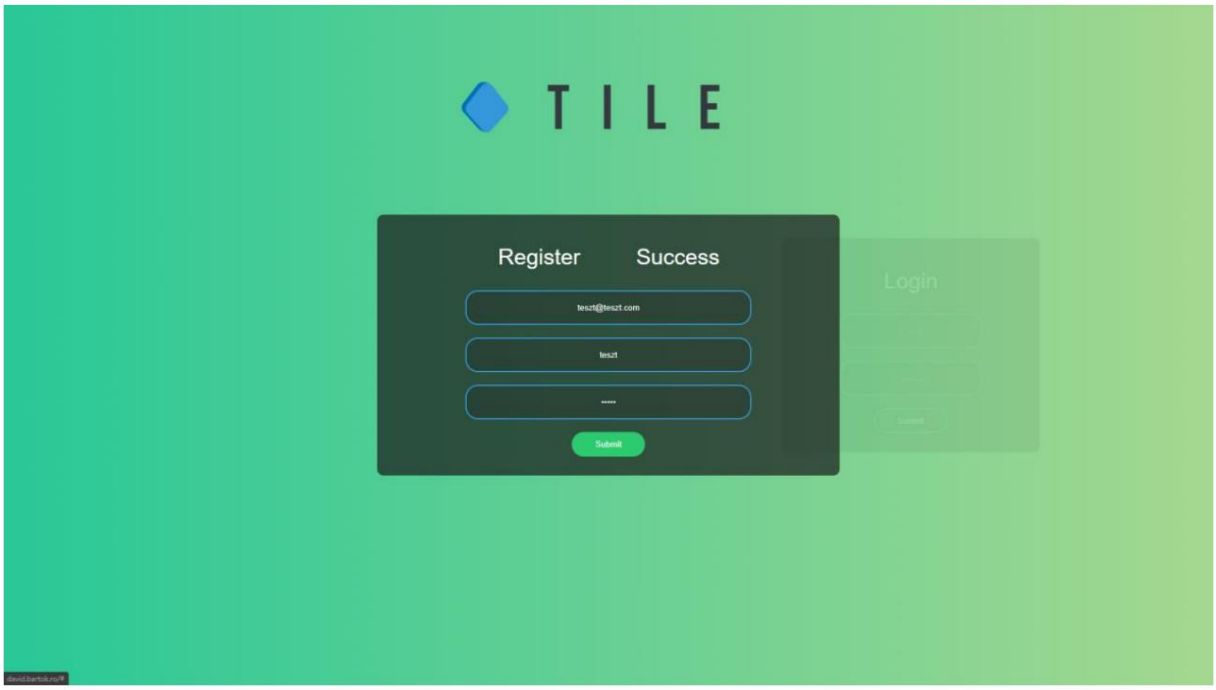
The main page greets you with 2 options: "Login" and "Register".



In the "Login" box, the user can only enter in that case

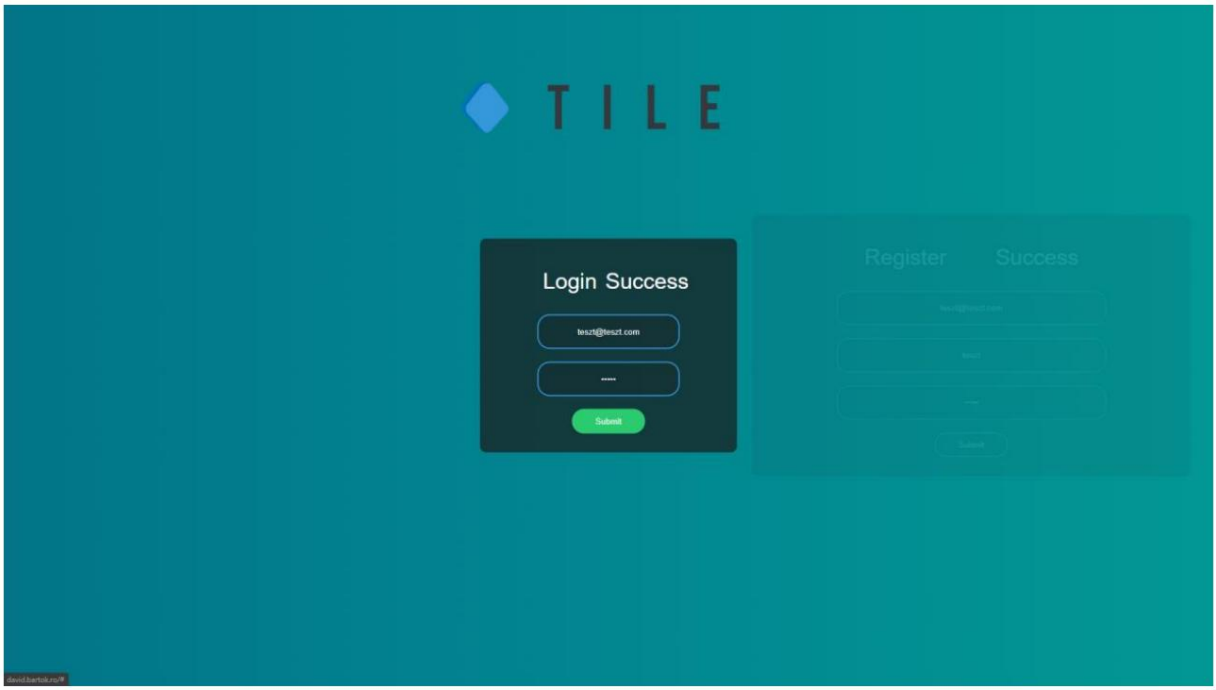
your login information if you already have an account. Otherwise, if we are talking about a new user, then you have to press the fainter "Register" (Hungarian: Registration).

Then, with a fluid and elegant animation, the "Register" cube replaces the "Login" cube
will be lost, giving the user the opportunity to open a new account. After the user enters the requested data,
and created a new account by pressing the "Submit" button.

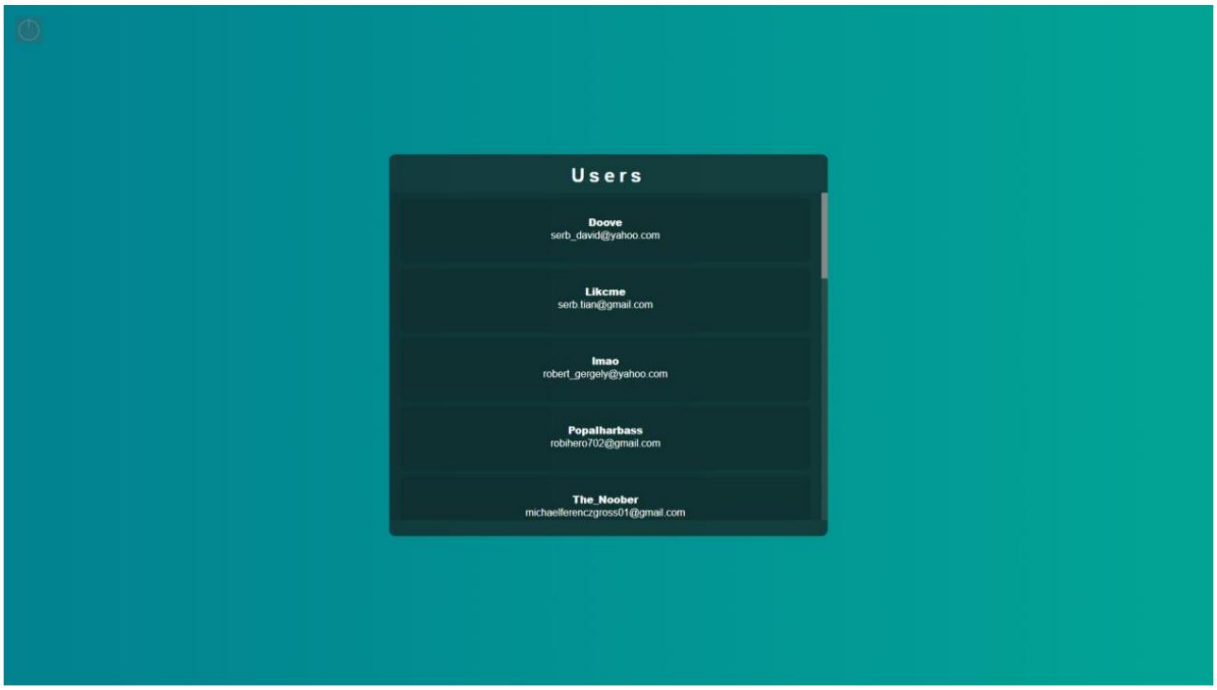


After opening an account, if the user wants to access his account, he must enter his information in the "Login"
cube, which can be called up in the same way as the "Register" cube by clicking on it. Then,
by entering the information requested during account creation and pressing the "Submit" button, the
user enters his account.

In addition, if the user has already created an account and has not logged out, then the page
you will be automatically logged into your existing account.

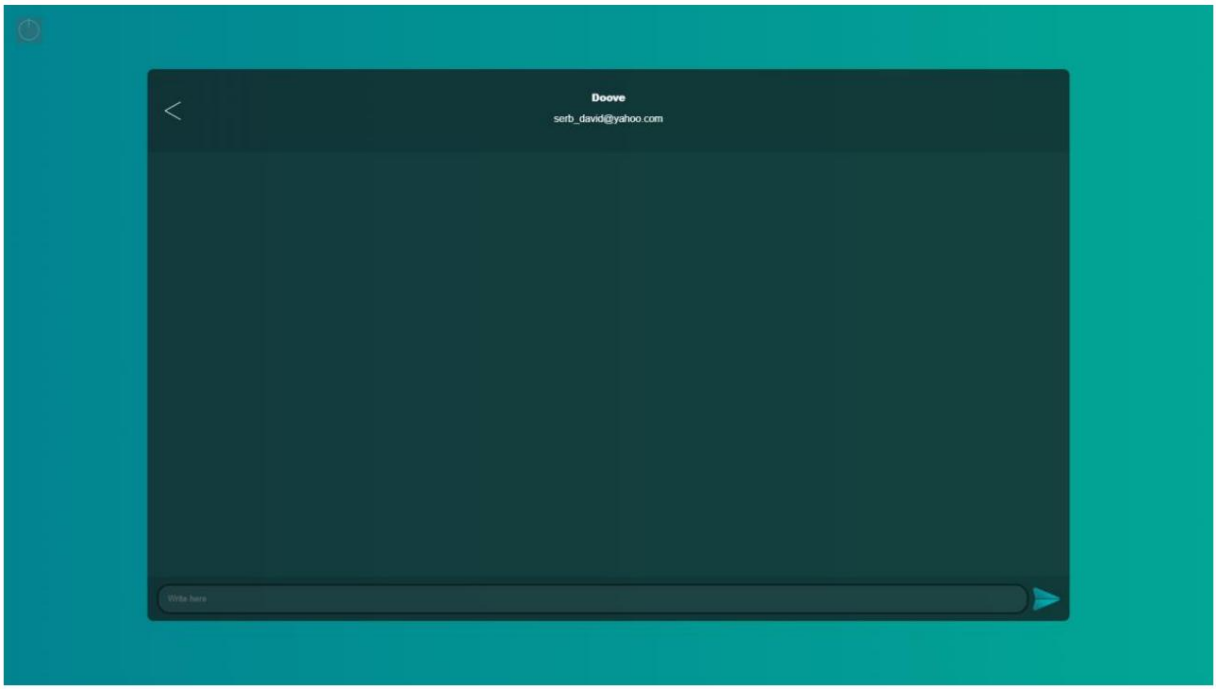


The chat page



On the next page, we are greeted by a list with all logged in users
user. By selecting an account from this list, the chat interface itself appears
with a specified person.

In addition to this, there is an exit button in the upper left corner, which the user can use
you can go back to the main page at any time, so that you will not be logged in automatically.



Here, intuitively, the user can write a message to the specified person, which is selected
person will receive it without delay.

Detailed description of the program

The login screen

Entry

```
<div class="access" id="login" onclick="HideRegister()">
    <script type="text/javascript">
        function login_failed() {
            document.getElementById("login_title").style.marginLeft = "25%";
            document.getElementById("login_title").style.left = "10%";
            document.getElementById("failed_title_login").style.opacity = "1";
        }

        function login_success() {
            setTimeout(() => {window.location.replace('home'); }, 1000);
            document.getElementById("login_title").style.marginLeft = "25%";
            document.getElementById("login_title").style.left = "10%";
            document.getElementById("failed_title_login").style.opacity = "0";
            document.getElementById("successful_title_login").style.opacity = "1";
        }
    </script>
    <div style="height: 90px;">
        <div id="login_title">
            <h2>Login</h2>
        </div>
        <div id="failed_title_login">
            <h2>Failed</h2>
        </div>
        <div id="successful_title_login">
            <h2>Success</h2>
        </div>
    </div>
    <form id="login_form" action="home\">
        <input type="email" id="login_email" name="email" placeholder="Email" required>
        <input type="password" id="login_passw" name="pass" placeholder="Password" required>
        <input type="submit" name="submit" value="Submit" onclick="return check_login()">
    </form>
</div>
```

First of all, if the user already has an account and has not logged out of it
button, the page will log in automatically.

The "Login" cube is made up of several cubes, the address where it says "Login"
"Login Failed" or "Login Success" in some cases), and it's also there
area for entering information, which includes two text boxes (for e-mail and code) and one
submit button.

To display this, if the user used the "Register" cube, then a
user just needs to press the "Login" box itself, which looks like a

would be in the background. Pressing this will start the JavaScript called "HideRegister()". function.

When the user presses the submit button, using the "onclick" function the "check_login()" JavaScript function is called.

Automatic login

```php
<?php
  session_start();
  if (isset($_SESSION["email"])) {
    header("Location: http://david.bartok.ro/home");
  }
?>
```

This Php-script is at the beginning of the page, check it if the user already has them fed data into the existing Cookies. If this is the case, you don't need to do it manually to log in, the site will do this automatically.

HideRegister()

```javascript
var log = document.getElementById("login");
var reg = document.getElementById("register");

function HideRegister() {
    reg.style.opacity = "0";
    log.style.opacity = "1";
    reg.style.width = "30%";
    reg.style.left = "80%";
    log.style.left = "50%";
    reg.style.opacity = "0.1";
}
```

This function assigns "Login" and "Register" to the variables "log" and "reg", and is working It removes the "Register" cube and reinforces the "Login" cube to give the illusion that that the "Login" cube is above the "Register" cube. Moves the "Register" cube all the way to the left, while bringing the “Login” cube to the center.

check_login()

```
function check_login() {
    var email=document.getElementById('login_email').value;
    var passw=document.getElementById('login_passw').value;
    var dataString='email='+ email + '&passw=' + passw;
    $.ajax({
        type:"post",
        url:"login.php",
        data:dataString,
        cache:false,
        success: function(html) {
            $('#scripts').html(html);
        }
    });
    return false;
}
```

"check_login()" uses Ajax (a web development technique that allows the web page exchange data with the server) is used to ensure that when the user sends the entered information with the "Submit" button, the page should not be reloaded because the Php script check the entered data. I used this to provide a more elegant solution for a simple problem.

In the code, the script takes the "email" and "passw" variables, whose values are the corresponding takes it from a text box.

The "dataString" variable takes this data, in such a way that it is triggered Php-script can read them.

Next we use Ajax, "url: 'login.php'," tells which Php file to must be accessed.

"success: function(html) {$('#scripts').html(html);}" ensures that what we write in the Php code, it appears in the element identified with the id "scripts".

login.php

```php
<?php session_start()?>
<?php
  $email = $_REQUEST['email'];
  $passw = $_REQUEST['passw'];
  if (((trim($email) != "") & ((strpos($email, "<")) == false) & ((strpos($email, ">")) ==
      false)) & ((trim($passw) != "") & ((strpos($passw, "<")) == false) & ((strpos($passw,
       ">")) == false))) {
    $file = file_get_contents('database.php');
    $verify = $email . " " . $passw." ";
    if (strpos($file, $verify) == true){
      echo "<script>login_success();</script>";
      $_SESSION["email"] = $email;
    } else {
    echo "<script>login_failed();</script>";
    }
  }
?>
```

The values of the "email" and "passw" variables are taken from the aforementioned "dataString". THE following "if" function ensures that the read values are not empty and not contain duckbills (< or >), due to code injection.

If the conditions are fulfilled, the Php-script checks the scanned data in the database their existence. If such an account already exists, we will start one on the website JavaScript function called "login_success()" (which we will talk about later) and saves the user's e-mail address in a Cookie. Conversely, if there is no such account, the a "login_failed()" function on a website.

login_success()

```javascript
function login_success() {
    setTimeout(() => {window.location.replace('home'); }, 1000);
    document.getElementById("login_title").style.marginLeft = "25%";
    document.getElementById("login_title").style.left = "10%";
    document.getElementById("failed_title_login").style.opacity = "0";
    document.getElementById("successful_title_login").style.opacity = "1";
}
```

The first line tells the website to switch pages after 1 second, enter the to the "home" folder where the chat page is located.

The following lines display "Success" in the title next to the text "Login". with a fluid movement.

login_failed()

```
function login_failed() {
    document.getElementById("login_title").style.marginLeft = "25%";
    document.getElementById("login_title").style.left = "10%";
    document.getElementById("failed_title_login").style.opacity = "1";
}
```

What this function does is that the word "Failed" appears next to "Login" in the title,

making the user aware that they have entered wrong data.

Registration

```
<div class="access" id="register" onclick="HideLogin()">
    <script type="text/javascript">
        function register_failed() {
            document.getElementById("register_title").style.marginLeft = "25%";
            document.getElementById("register_title").style.left = "10%";
            document.getElementById("failed_title_register").style.opacity = "1";
    }
        function register_success() {
            document.getElementById("register_title").style.marginLeft = "25%";
            document.getElementById("register_title").style.left = "10%";
            document.getElementById("failed_title_register").style.opacity = "0";
            document.getElementById("successful_title_register").style.opacity = "1";
        }
    </script>
    <div style="height: 90px;">
        <div id="register_title">
            <h2>Register</h2>
        </div>
        <div id="failed_title_register">
            <h2>Failed</h2>
        </div>
        <div id="successful_title_register">
            <h2>Success</h2>
        </div>
    </div>
    <form id="register_form">
        <input type="email" id="register_email" name="email" placeholder="Email" required>
        <input type="text" id="register_user" name="user" placeholder="Username" required>
        <input type="password" id="register_passw" name="pass" placeholder="Password" required>
        <input type="submit" name="submit" value="Submit" onclick="return check_register()">
    </form>
</div>
```

The "Register" cube, like the "Login" cube, consists of several cubes; the address where it says

to "Register" (and in certain cases "Register Failed" or "Register Success").

Below that is the area for entering information, which contains three text boxes

(of your e-mail, username and code).

To display this, the user only has to press the "Register" box itself,

which looks like it's in the background. Pressing this will start “HideLogin()”

named JavaScript function.

When the user presses the submit button, using the "onclick" function
the "check_register()" JavaScript function is called.

HideLogin()

```
var log = document.getElementById("login");
var reg = document.getElementById("register");

function HideLogin() {
    log.style.opacity = "0.1";
    reg.style.opacity = "1";
    reg.style.width = "32%";
    reg.style.left = "50%";
    log.style.left = "75%";
}
```

Here, we assign the "Login" and "Register" cubes to the "log" and "reg" variables, and with these
we are working. We fade out the "Login" cube and put it on the right side, and next to it a
We strengthen the "Register" cube and bring it to the center.

check_register()

```
function check_register() {
    var email = document.getElementById("register_email").value;
    var passw = document.getElementById("register_passw").value;
    var user = document.getElementById("register_user").value;
    var dataString = "email=" + email + "&passw=" + passw + "&user=" + user;
    $.ajax({
        type:"post",
        url:"register.php",
        data:dataString,
        cache:false,
        success: function(html) {
        $("#scripts").html(html);
        }
    });
    return false;
}
```

This function, similar to the "check_login()" function, uses Ajax in order to
so that the page does not refresh again when the data is processed. "email", "passw" and "user"
variables are filled with information from the corresponding text box, which will be
We load them into the "dataString" variable so that the "register.php" Php-script can process them.

register.php

```php
<?php
  $email = $_REQUEST["email"];
  $passw = $_REQUEST["passw"];
  $user = $_REQUEST["user"];
  if (((trim($email) != "") & ((strpos($email, "<")) == false) & ((strpos($email, ">")) ==
    false)) & ((trim($passw) != "") & ((strpos($passw, "<")) == false) & ((strpos($passw, "
    >")) == false))  & ((trim($user) != "") & ((strpos($user, "<")) == false) & ((strpos($
    user, ">")) == false))) {
    $myfile = file_get_contents("database.php");
    if (strpos($myfile, $email) == true){
      echo "<script>register_failed()</script>";
    } else {
      $file_main = fopen("database.php", "a");
      fwrite($file_main, $email." ".$passw." ".$user."\n");
      fclose($file_main);
      $file_eu = fopen("home/database.php", "a");
      fwrite($file_eu, $email." ".$user."\n");
      fclose($file_eu);
      echo "<script>register_success()</script>";
    }
  }
?>
```

The variables "email", "passw" and "user" are populated by the variable "dataString" (the former from an Ajax function) with the appropriate information.

The following "if" function checks that if the characters read are not empty and do not contain duckbills (< or >), avoiding code injection.

Next, the code checks to see if an account with the same email already exists use as entered by the user. If such a user does not already exist, then these are a variables are entered in the main database and only one of them is e-mail and username to a secondary database. Additionally, the web page starts a process called "register_success()". JavaScript function. Otherwise, if the login is not successful, then a a JavaScript function named "register_failed()" is launched on the website.

register_success()

```javascript
function register_success() {
    document.getElementById("register_title").style.marginLeft = "25%";
    document.getElementById("register_title").style.left = "10%";
    document.getElementById("failed_title_register").style.opacity = "0";
    document.getElementById("successful_title_register").style.opacity = "1";
}
```

This function displays "Success" next to "Register" in the title.

register_failed()

```
function register_failed() {
    document.getElementById("register_title").style.marginLeft = "25%";
    document.getElementById("register_title").style.left = "10%";
    document.getElementById("failed_title_register").style.opacity = "1";
}
```

This function, unlike the other one, displays next to "Register" in the title that
"Failed".

The chat page

```
<img src="logout.png" id="logout" onclick="logout()">
<div id="main_page">
    <h2 id="greeting">
        Users
    </h2>
    <div id="users">
        <?php
            for ($i=2; $i<$n; ++$i) {
                echo '<div id="user" onclick="return check_names(\''.$information[$i][1]
                    .'\',\''.$information[$i][2].'\')"><div id="user_name">'.$
                    information[$i][2].'</div><div id="user_email">'.$information[$i][1]
                    .'</div></div>';
            }
        ?>
    </div>
    <div id="header">
        <img src="back.png" id="back" onclick="transform_back()">
        <div id="header_user">
        </div>
        <div id="header_email">
        </div>
    </div>
    <div id="messages">
    </div>
    <div id="textpad">
        <form>
            <input type="text" placeholder="Write here" id="textbox" requred>
            <input type="submit" id="sendable" value="" onclick="return prepare()">
        </form>
    </div>
</div>
```

First of all, an exit button appears on the page, which, when pressed, starts a
called "logout()" JavaScript function.

The point of this page is to write a list that shows all of them
user, from which the logged in person can choose someone to chat with
by clicking on their name.

This page can be divided into two separate sections: selection and chat.

logout()

```
function logout() {
    $.ajax({
        type:"post",
        url:"logout.php",
        cache:false,
        success: function(html) {
            $('#scripts').html(html);
        }
    });
    return false;
}
```

Although this subroutine uses Ajax, it does not pass any information to the "logout.php" Php-script, because there is no shortage for that. All data/scripts that we will write in the Php-script are a It appears in an element with id "scipts".

logout.php

```
<?php
    session_start();
    session_unset();
    session_destroy();
    echo "<script>window.location.replace('http://david.bartok.ro');</script>";
?>
```

All this script does is inject JavaScript code into the page and return it to the user to the start page, so that the page will no longer automatically log in to the By using cookies, because we have emptied them.

The selection

```
<h2 id="greeting">
    Users
</h2>
<div id="users">
    <?php
        for ($i=2; $i<$n; ++$i) {
            echo '<div id="user" onclick="return check_names(\''.$information[$i][1]
                .'\',\''.$information[$i][2].'\')"><div id="user_name">'.$
                information[$i][2].'</div><div id="user_email">'.$information[$i][1]
                .'</div></div>';
        }
    ?>
</div>
```

To help the selection part, a code is run at the beginning of the web page, which is secondary to the page by selecting your database, enter the names of all users in a two-dimensional array, respectively your e-mail address. With this information, the web page creates a list that contains its elements into different cubes to make them easier to see and select.

Once the user has decided who they want to chat with, they can do so with a simple click can also reach When the user clicked on a specific person, the "onclick" function a so-called "check_names()" JavaScript function is started, which parameters are the selected person's e-mail and their name.

Scanning algorithm

```php
<?php
  session_start();
  if (isset($_SESSION["email"])) {
    $email = $_SESSION["email"];
  } else {
    header("Location: http://david.bartok.ro");
  }
  $information = array(array());
  $file = fopen("database.php","r");
  $i = 0;
  while(! feof($file)) {
    ++$i;
    $line = fgets($file);
    if ($i != 1) {
        $information[$i][1] = substr($line, 0, strpos($line, ' '));
        $information[$i][2] = trim(substr($line, strpos($line, ' ') + 1, strlen($line)));
      }
    $n=$i;
  }
?>
```

Before the selection, the Php-script checks if the user has cookies entered information, it means that you came to this page through login. Otherwise, if the given Cookie is empty, it means that the user did not access it legally page (since the user's e-mail is recorded in the Cookie upon entry) and then the Php-script takes the user back to the home page.

After that, the Php-script reads the secondary database line by line and notes separately the e-mails and usernames into a two-dimensional matrix (the "information" variable, that is the e-mail in the first place, and the username in the second place). The program notes the "n" the number of users as a variable.

check_names()

```
function check_names(email, user) {
    var dataString="email=" + email + "&user=" + user;
    $.ajax({
        type:"post",
        url:"search.php",
        data:dataString,
        cache:false,
        success: function(html) {
            $('#scripts').html(html);
        }
    });
    return false;
}
```

What the "check_names()" subroutine does is take its parameter variables and these put it in the "dataString" variable, which will be used by the "search.php" Php-script to work.

All deleted data/code deleted by Php-script in the Html code in the element identified with the id "scripts" displayed.

search.php

```php
<?php
    session_start();
    $user = $_REQUEST['user'];
    $contact = $_REQUEST['email'];
    $email = $_SESSION["email"];
    $_SESSION["contact"] = $contact;
    echo "<script>transform_main();</script>";
    echo "<script>document.getElementById('header_user').innerHTML = '".$user."';</script>";
    echo "<script>document.getElementById('header_email').innerHTML = '".$contact."';</script>";
    if ($email<$contact) {
        $myfile = "o0zTrb0[redacted]/".$email." - ".$contact;
    } else {
        $myfile = "o0zTrb0[redacted]/".$contact." - ".$email; }
    if (file_exists($myfile) == true) {
        $file = fopen($myfile, "r");
        while(! feof($file)) {
            $line = fgets($file);
            if (strpos($line, $email) != false) {
                $nline = fgets($file);
                echo "<script>document.getElementById('messages').innerHTML += '<div
                    class=margin><div class=message id=right>".trim($nline)."
                    </div></div></div>';</script>";
            }
            if (strpos($line, $contact) != false) {
                $nline = fgets($file);
                echo "<script>document.getElementById('messages').innerHTML += '<div
                    class=margin><div class=message id=left>".trim($nline)."
                    </div></div></div>';</script>";
            }
        }
        echo "<script>document.getElementById('messages').scrollTop =
            document.getElementById('messages').scrollHeight;</script>";
        unset($_SESSION["last_changed"]);
    }
?>
```

The variables "user" and "contact" take their values from the previous "dataString" and "email" variable from Cookies. The code then saves the e-mail of that person in a Cookie selected by the user. A JavaScript subprogram called "transform_main()" starts, which changes the page and actually switches from the selection section to the chat section into part.

Then the program enters the elements with the id "header_user" and "header_email" selected person's name and e-mail. The code to know what file to look for compares the email of the user and the selected person and sees if this is the file exists.

If this file already exists, the program reads it line by line, and depending on
from which e-mail this message originates, the message will be displayed on the appropriate page. The file in which
the user's messages look like this:

```
<?php header('Location: http://david.bartok.ro'); ?>
<teszt@teszt.com>
szia
<serb_david@yahoo.com>
Szia!
```

transform_main()

```
function transform_main() {
    document.getElementById("main_page").style.height = "70%";
    document.getElementById("main_page").style.width = "70%";
    document.getElementById("greeting").style.display = "none";
    document.getElementById("users").style.display = "none";
    document.getElementById("header").style.display = "block";
    document.getElementById("textpad").style.display = "block";
    document.getElementById("messages").style.display = "block";
}
```

With the help of this subroutine, the web page switches from the selection phase to the chat phase
phase. The canvas enlarges, the users and the greeting disappear, and the
header, message box and text box.

The chat

```
<div id="header">
    <img src="back.png" id="back" onclick="transform_back()">
    <div id="header_user">
    </div>
    <div id="header_email">
    </div>
</div>
<div id="messages">
</div>
<div id="textpad">
    <form>
        <input type="text" placeholder="Write here" id="textbox" requred>
        <input type="submit" id="sendable" value="" onclick="return prepare()">
    </form>
</div>
```

These elements are all hidden until the point when the selection page changes
section to the chat section. There is a back button in the corner of the chat window, which you can use

the user can return to the selection section with a JavaScript called "transform_back()".
with a subroutine.

Then at the bottom of the chat window there is a text box in which the user can enter his message.
By pressing the submit button, the message in the text box will be sent
Using the "prepare()" JavaScript subroutine.

When the user receives a message, this will be written out by the subroutine called "refresh()".
which searches for new messages every second using the "setInterval" JavaScript
using a function.

transform_back()

```
function transform_back() {
    document.getElementById("main_page").style.height = "45%";
    document.getElementById("main_page").style.width = "30%";
    document.getElementById("greeting").style.display = "block";
    document.getElementById("users").style.display = "block";
    document.getElementById("header").style.display = "none";
    document.getElementById("textpad").style.display = "none";
    document.getElementById("messages").style.display = "none";
    document.getElementById("messages").innerHTML = "";
}
```

With this subroutine, the web page returns to the selection phase. The canvas returns smaller
dimensions, the greeting and the list of users will appear again. The header and messages disappear
cube and the text box, and next to that, the cube of your message is reset to empty, ready to
post another user's messages.

prepare()

```
function prepare() {
    var message = document.getElementById('textbox').value;
    var dataString='message='+ message;
    $.ajax({
        type:"post",
        url:"write.php",
        data:dataString,
        cache:false,
        success: function(html) {
            $('#scripts').html(html);
        }
    });
    return false;
}
```

The "prepare()" subroutine is responsible for sending the data to the Php script that which will process them. The data from the text box goes into the "message" variable, then, entering this into the "dataString" variable, we send it to the "write.php" Php-script for processing. Then every script/text written by Php-script has the id "scripts". appears in element.

write.php

```
<?php
    session_start();
    $message = $_REQUEST["message"];
    if ((trim($message) != "") & ((strpos($message, "<")) == false) & ((strpos($message, ">")) ==
        false)) {
        $email = $_SESSION["email"];
        $contact = $_SESSION["contact"];
        if ($email<$contact)
            $myfile = "o0zTrb0[redacted]/".$email." - ".$contact;
        else
            $myfile = "o0zTrb0[redacted]/".$contact." - ".$email;
        if (file_exists($myfile) != true) {
            $file = fopen($myfile, "w");
            fwrite($file, "<?php header('Location: http://david.bartok.ro'); ?>\n");
            fclose($file);
        }
        file_put_contents($myfile, "<".$email.">\n".$message."\n", FILE_APPEND);
        echo "<script>document.getElementById('messages').innerHTML += '<div class=margin><div
            class=message id=right>".trim($message)."</div></div></div>';</script>";
        echo "<script>document.getElementById('messages').scrollTop =
            document.getElementById('messages').scrollHeight;</script>";
    }
    echo "<script>document.getElementById('textbox').value = '';</script>";
?>
```

We feed the data resulting from "dataString" into the "message" variable. Then the script examines so that the data is not empty and does not contain duckbills so that no code occurs injection. After that, we scan the e-mail of the user and the selected person, and to each other we compare to know what kind of stock we are working with.

If this file does not exist yet, it will be created now. It will have a header which
it will look like this: “<?php header('Location: http://david.bartok.ro'); ?>”. This is in that case
useful if one finds out where messages between users are stored, then
that if you try to read it from a browser, the Php server redirects you to the start page.

Then the message is entered in the correct text file, so that the person who sent it before his message
your e-mail will appear between two duckbills so that the program knows who sent it when it is read.

setInterval()

```
setInterval(refresh, 1000);
```

This is a function built into JavaScript, the essence of which is to start every second the
"refresh()" subroutine.

refresh()

```
function refresh() {
    $.ajax({
        type:"post",
        url:"refresh.php",
        cache:false,
        success: function(html) {
            $('#scripts').html(html);
        }
    });
    return false;
}
```

Although the "refresh()" subprogram uses Ajax, it does not pass any information to the Php-script,
just run it. All data/scripts written in the "refresh.php" Php-script will be displayed
in the element identified by id "scripts".

refresh.php

```php
<?php
    session_start();
    $email = $_SESSION["email"];
    $contact = $_SESSION["contact"];
    if ($email<$contact)
        $myfile = "o0zTrb0[redacted]/".$email." - ".$contact;
    else
        $myfile = "o0zTrb0[redacted]/".$contact." - ".$email;
    if (file_exists($myfile)) {
        if (isset($_SESSION["last_changed"]) == false)
            $_SESSION["last_changed"] = filemtime($myfile);

        if ($_SESSION["last_changed"] != filemtime($myfile)) {
            $file = fopen($myfile, "r");
            $skip = fgets($file);
            while(! feof($file)) {
                $try = fgets($file);
                if (strpos($try, "@") == true)
                    $sender = $try;
                if (!feof($file))
                    $message = fgets($file);
            }
            if (strpos($sender, $contact)){
                echo "<script>document.getElementById('messages').innerHTML += '<div
                    class=margin><div class=message id=left>".trim($message)."
                    </div></div></div>';</script>";
                echo "<script>document.getElementById('messages').scrollTop =
                    document.getElementById('messages').scrollHeight;</script>";
            }
        }
        $_SESSION["last_changed"] = filemtime($myfile);
    }
?>
```

The "email" and "contact" variables take the values of the Cookies and then compare them
select the correct text file. Checking to see if this exists, if it isn't already saved,
we save the exact time when this file was changed afterwards.

Then, if the post-saved change time is not equal to the real post-change time
time, then we know that the file has been changed compared to when we checked it afterwards.

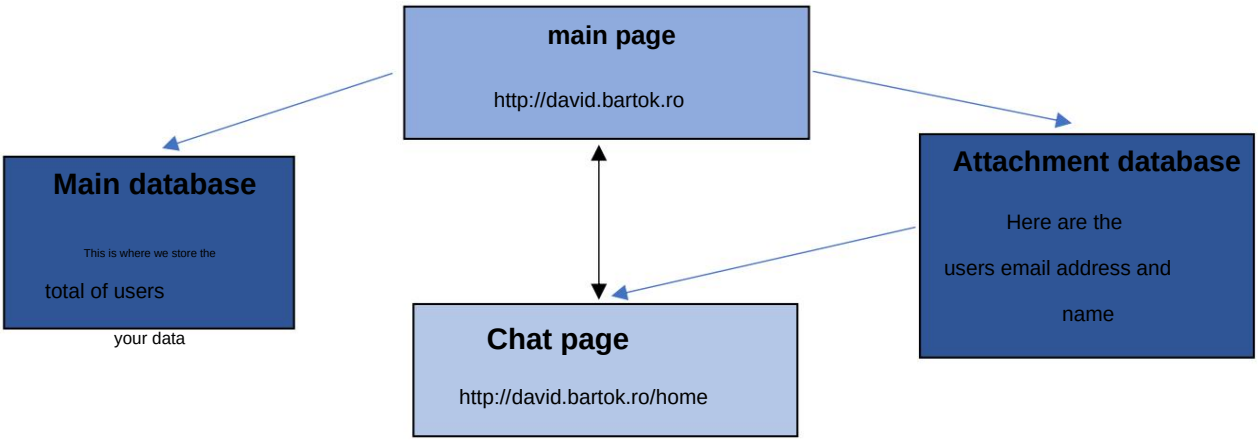
In this case, open the text file, skip the first line because of the header, and
we read all senders and messages into the "sender" and "message" variables into the "try" variable
with the help of

The purpose of this script is to remember the last sender and its message, which
so far he has done it.

If this last message is from the person the user is talking to, it means

that this message has not yet been written by the program, which will now be sent to the appropriate page, then automatically scroll down to it.

Block diagram



For the sake of the website's simplicity and elegance and for its purpose, it does not need to be more complicated be. The entire project consists of two websites that communicate using cookies.

The black arrow means the connection between the pages, so it is two-sided, the user believes

you can log out of your account at any time, returning to the main page.

The blue arrows represent the use of stocks:

From the main page, if someone does not yet have an account, create one and put their data into two databases they cost. The main database is only used for testing purposes at login, while the ext database to list users on the chat page.

Further development opportunities

- Possibility to send pictures
- Find out if someone sends a message even if the user is not chatting with them
- Find out if a user is online

Librarianship

Help related to programming:

- https://www.w3schools.com/xml/ajax_intro.asp
- https://stackoverflow.com/questions/1917576/how-do-i-pass-javascript-variables-to-php
- https://www.w3schools.com/css/default.asp
- https://www.w3schools.com/jsref/met_win_settimeout.asp
- https://www.w3schools.com/howto/howto_js_redirect_webpage.asp
- https://www.w3schools.com/howto/howto_css_transition_hover.asp
- https://www.w3schools.com/howto/howto_css_custom_scrollbar.asp
- https://www.w3schools.com/howto/howto_html_autocomplete_off.asp
- https://stackoverflow.com/questions/60494203/inter-affecting-objects-in-css