

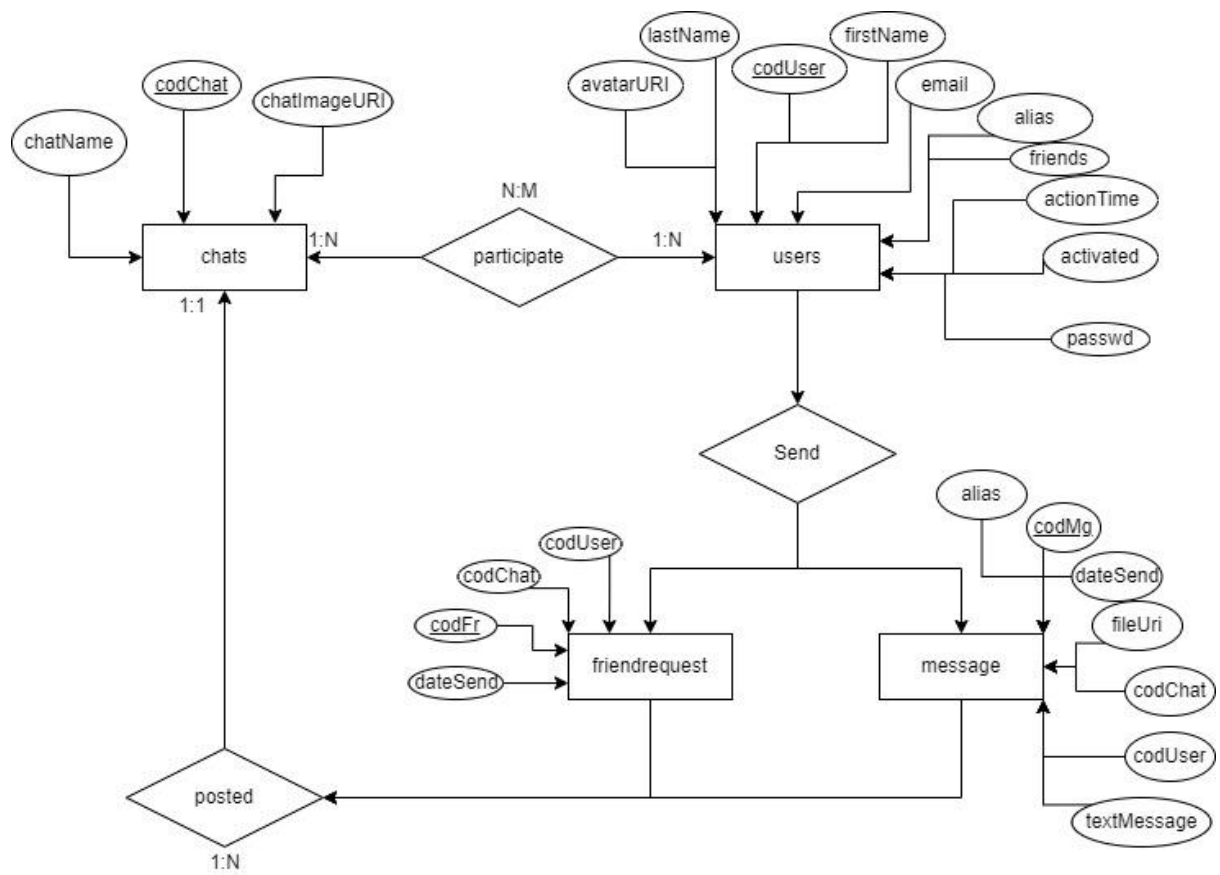
ChatApp Documentation

By: Félix Martínez Bendicho

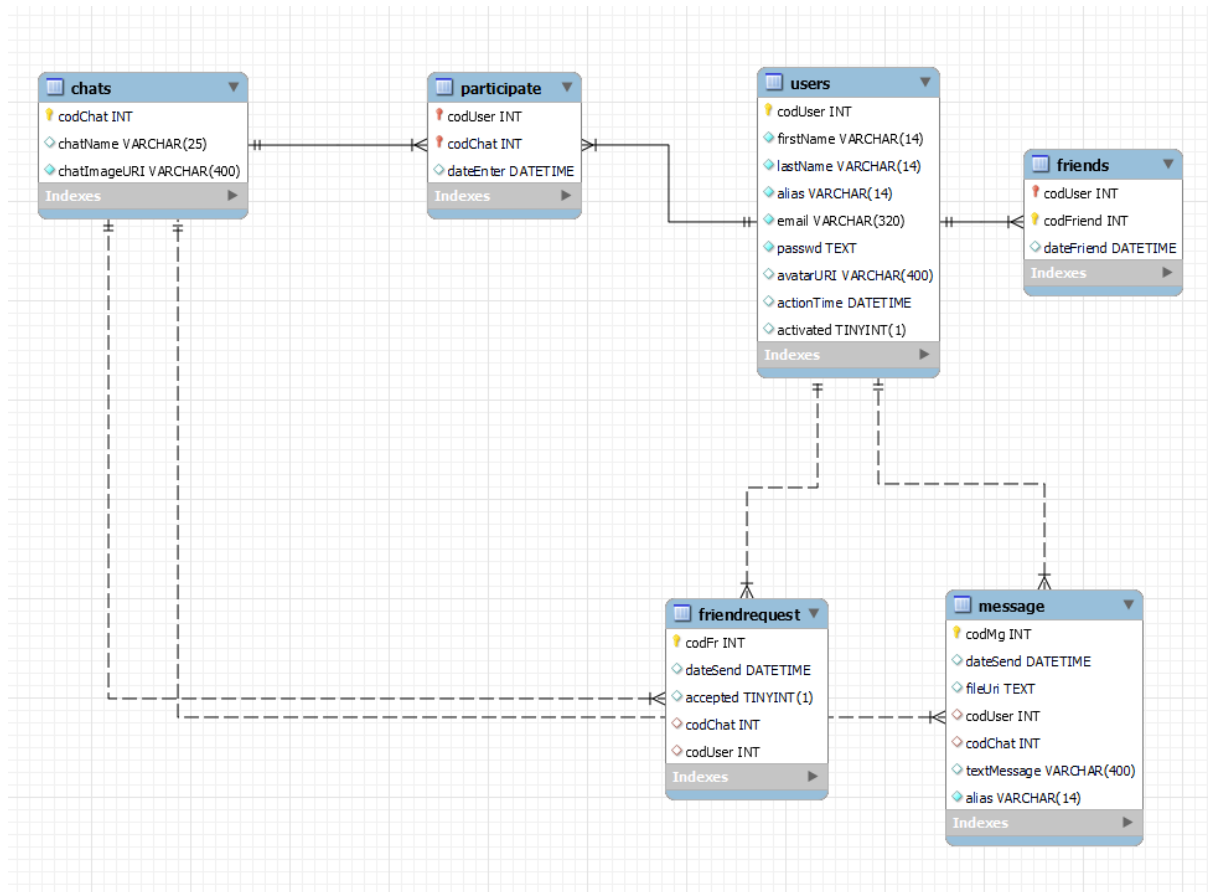
Extensions

ENLARGEMENT	MADE (Y/N)
A1 Self-registration	Y
A2 Password Recovery	Y
A3 Message to multiple recipients	Y
A4 Encrypted user password	Y
A5 User avatar	Y
A6 User profile	Y
A7 Friendship	Y
A8 Groups	Y
A9 Administration zone	Y
A10 Attached files	Y
A11 Images	Y
A12 Testing	N
A13 Outbox	Y
A14 Presentation	Y

Database diagram



E/R Scheme



User manual

For this project I tried to simulate the functionality of comun used chat apps like WhatsApp, Telegram, etc.

In order to achieve that I had to investigate how to use AJAX to provide the project with automatic page sections updates, designing UIs with Bootstrap and general use practices and programming capabilities of PHP.

As I mentioned, the main part of the web application remains in the *chats_home.php* file, where almost all the user data is loaded.

In order to easily implementate groups and private chats simultaneously, I followed a similar approach as popular chat apps; where messages are sent to chats, no matter if that chat remains only for private messaging with another user or if more than two users are in the same chat.

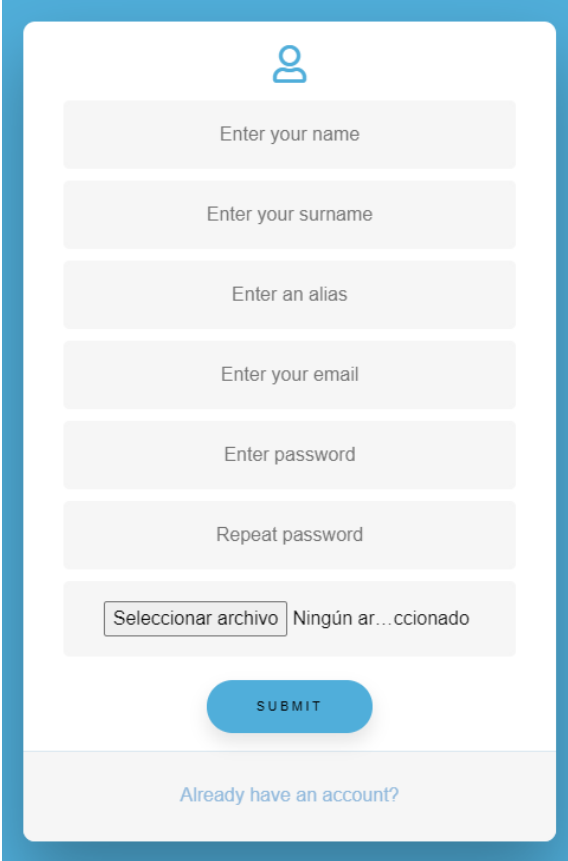
With that point of view, messages are sent to chats, where users participate. This has more implications, for example, in order to see what chats have unread

messages, you only have to see if the last message of the chat is unread for the logged user, if so, a *Unread messages* will be shown in the chats section.

I faced some problems while implementing AJAX. First of all, it isn't easy to communicate between PHP and JavaScript, for that I've used cookies and passed through get values. Also, AJAX is based in encoding and send data, so we need to construct a JSON file with the retrieved data from the DB and get that file from JavaScript, then parse that JSON with *JSON.parse()* and iterate each value in order to process it show it to the user in a UI integrated way.

Last of all, it became very time-consuming to create each HTML element with nodes in JavaScript. There are simpler ways to approach this, but I decided to get into it with nodes and I regret that decision.

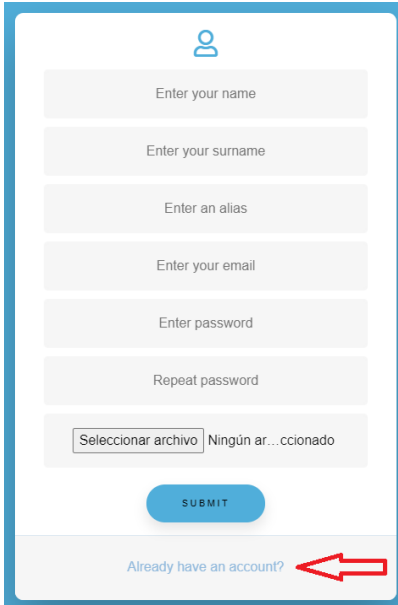
1. Register as new user:



A registration form with a blue border. At the top is a blue user icon. Below it are seven input fields: 'Enter your name', 'Enter your surname', 'Enter an alias', 'Enter your email', 'Enter password', 'Repeat password', and a file selection field with a button labeled 'Seleccionar archivo' and the text 'Ningún archivo seleccionado'. Below these is a blue 'SUBMIT' button. At the bottom is a link that says 'Already have an account?'.

- This is the default page.
- Avatar is optional.
- An email is sent to the provided email address, the user must confirm the operation within the next hour since he/she registered in order to successfully register.

2. Login:



A login form with a blue border, identical in layout to the registration form. It includes the same seven input fields and the 'SUBMIT' button. The 'Already have an account?' link at the bottom is highlighted with a red arrow pointing to it from the right.

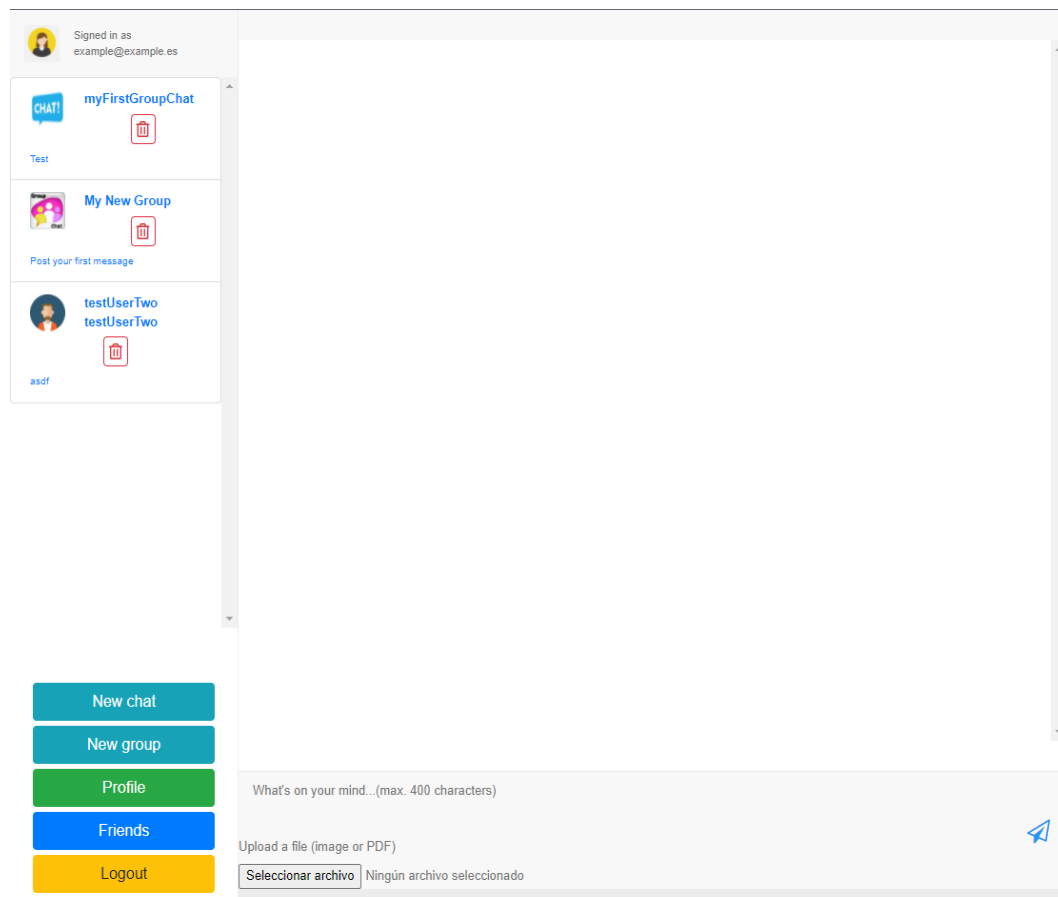
- If you are already a user,

click the link and login.

A login form with a blue border. At the top is a blue user icon. Below it are two light gray input fields: the first is labeled 'alias or email' and the second is labeled 'password'. Below the fields is a blue rounded button with the text 'SUBMIT'. At the bottom of the form are two links: 'Forgot password?' and 'Not yet a user?'.

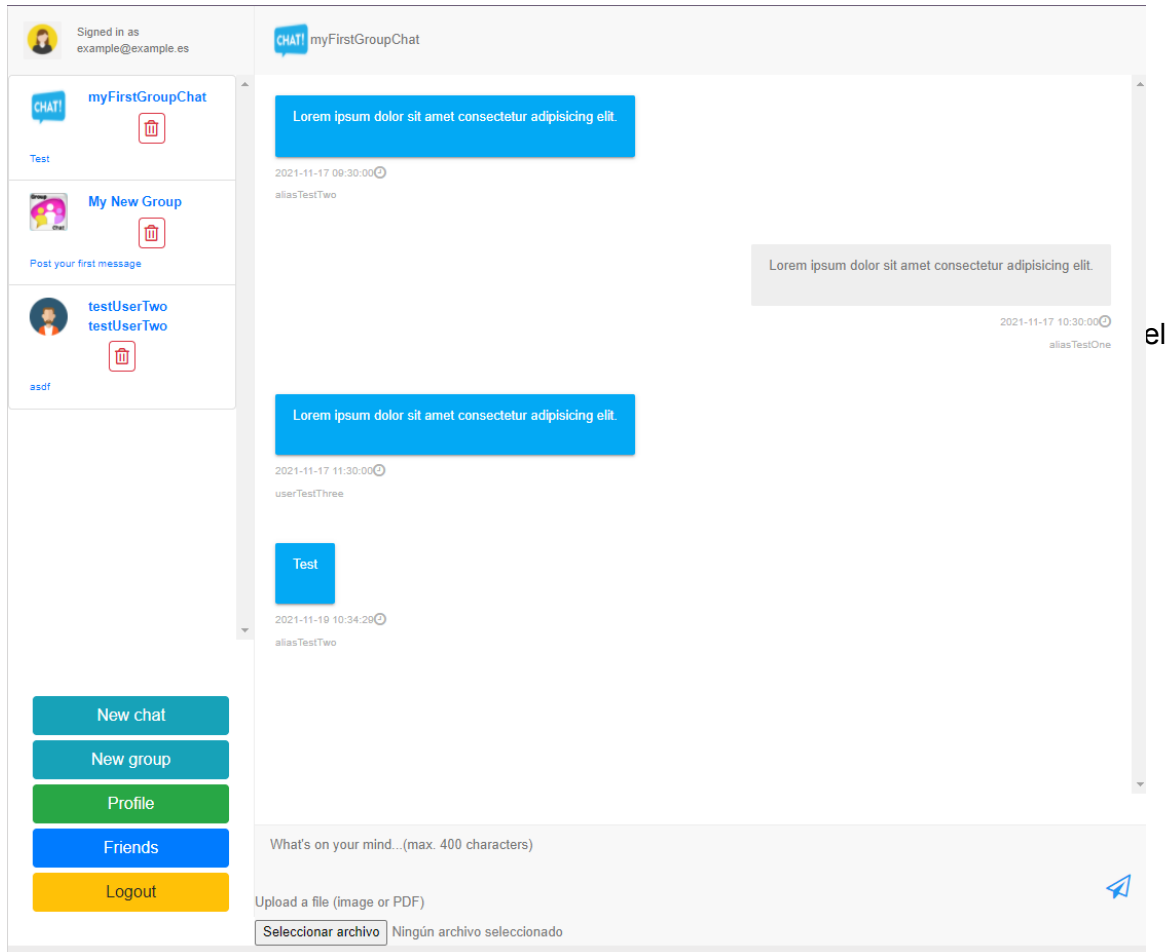
- When a chat from the left section is clicked, the messages from that chat will be shown in the right section.

3. Once you are logged in:

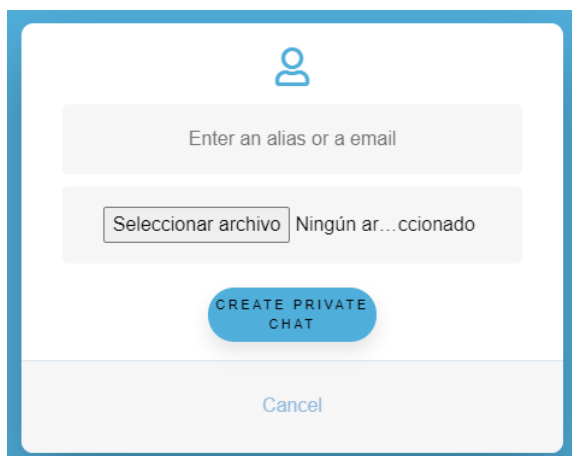


- Once you are logged in you will see the default home screen.

4. User actions in home screen:



a. New chat:



Then click the submit button, or click cancel to return to the home screen.

Enter a valid user alias or email.

In order to chat with someone you don't need to be friends. But you won't be able to see the full user profile.

To cancel the operation click "Return".

b. New group:




th someone you don't
But you won't be able to
ll user profile.

Enter valid user aliases or emails separated by commas.

You can add a group profile image and a group name, but they are optional.

c. Profile:

First Name	Last Name	Alias	Email	Avatar
testUserOne	testUserOne	aliasTestOne	example@example.es	
<input type="text" value="New first name"/>	<input type="text" value="New last name"/>	<input type="text" value="New alias"/>	<input type="text" value="New email"/>	<input type="button" value="Upload new avatar"/> <input type="button" value="Seleccionar archivo"/> <input type="text" value="Ningún archivo seleccionado"/>

then, you will be redirected to the login page.

Here you can modify your profile, enter the desired values in the correct inputs and click the "Submit Changes" button to update the profile info.

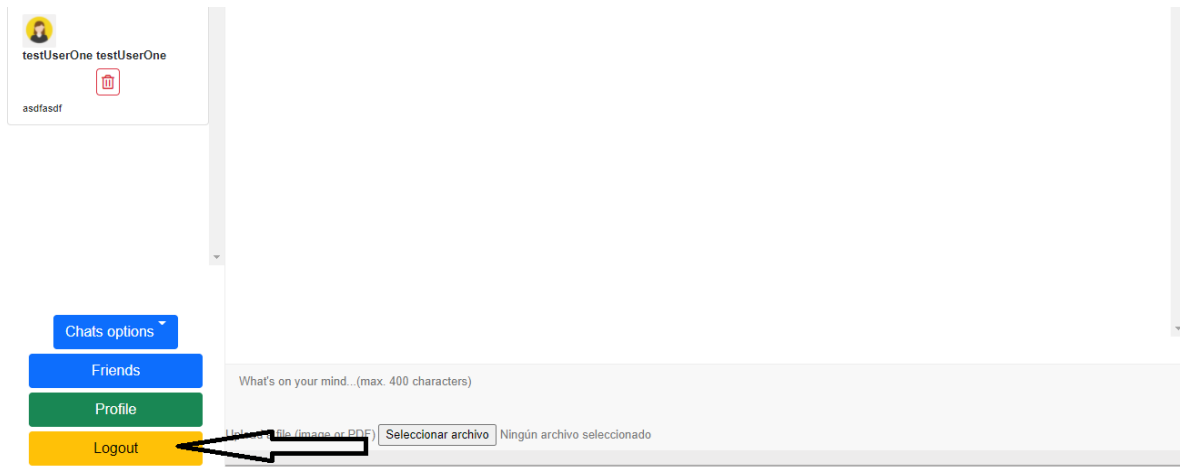
d. Friends:

First Name	Last Name	Alias	Email	Avatar	Friends since	Create chat
testUserTwo	testUserTwo	aliasTestTwo	example@example2.es		2021-11-22 11:22:42	<input type="button" value="Chat"/>

On this page you will see all your friends.
You can directly private chat with any of

them by clicking the “Chat” button.

e. Logout:
















To logout simply click the yellow “logout” button.

Provided data

The provided data includes:

1. Four activated users and the admin one.
 - a. Admin = user: admin / password: admin
 - b. User one = testUserOne / password: Password1234-
 - c. User one = testUserTwo / password: Password1234-
 - d. User one = userTestThree / password: Password1234-
 - e. User one = userTestFour / password: Password1234-

										activated		
										TRUE is converted to 1 and FALSE is converted to 0		
		codUser	firstName	lastName	alias	email	passwd	avatarURI	actionTime			
				1	admin	admin	admin	admin@admin.com	\$2y\$10\$HTkur1cOWj3MjPwJxzl.eDINpyRXgE1nqZytqcYPe1...	NULL	NULL	1
				2	testUserOne	testUserOne	aliasTestOne	example@example1.es	\$2y\$10\$0UksxQd.AfcjBwZP1cNgP.aLRLFBZbflOlqdv5TQg7T...	./uploads/avatars/image1.png	NULL	1
				3	testUserTwo	testUserTwo	aliasTestTwo	example@example2.es	\$2y\$10\$WAE1CFGJllc/QMtazTKmXezFmwLJJ/GJkLQG/cFb3GT...	./uploads/avatars/image2.png	NULL	1
				4	userTestThree	userTestThree	aliasTestThree	example@example3.es	\$2y\$10\$7KT5fKhPxbk4f9bHrMbZxeF72yO2v1h1rbbeHGAHFS...	./uploads/avatars/image1.png	NULL	1
				5	userTestFour	userTestFour	aliasTestFour	example@example4.es	\$2y\$10\$snVH8IA/EypmVuBN9YBM4.NQIDA5MfU8cH1AQJFB2JR...	./uploads/avatars/image2.png	NULL	1

2. Two friend requests:

- a. From aliasTestOne to aliasTestTwo (pending).
- b. From aliasTestThree to aliasTestFour (pending).

<div><div><div><div></div><div></div></div><div></div></div></div>					codFr	dateSend	accepted	codUser	codChat
<div><div><div></div></div></div>	<div><div><div></div></div></div> <div>Editar</div>	<div><div><div></div></div></div> <div>Copiar</div>	<div><div><div></div></div></div> <div>Borrar</div>	1	2021-11-24 09:33:31	0	4	3	
<div><div><div></div></div></div>	<div><div><div></div></div></div> <div>Editar</div>	<div><div><div></div></div></div> <div>Copiar</div>	<div><div><div></div></div></div> <div>Borrar</div>	6	2021-11-24 11:06:39	0	5	12	






















3. Eight messages:

- Two friend requests (text content = button).
- Three broadcast messages from aliasTestFour to the rest of the users.
- Three unread messages from aliasTestOne to the group chat, aliasTestTwo and aliasTestThree.

				codMg	dateSend	fileUri	codUser	codChat	textMessage	alias
<input type="checkbox"/>	Editar	Copiar	Borrar	1	2021-11-24 09:33:31	NULL	2	4	<a class="btn btn-primary" href="../friend_request...	aliasTestOne
<input type="checkbox"/>	Editar	Copiar	Borrar	3	2021-11-24 10:13:09	../uploads/attachments/image2.png	4	6	Test broadcast	aliasTestThree
<input type="checkbox"/>	Editar	Copiar	Borrar	4	2021-11-24 10:13:09	../uploads/attachments/image2.png	4	7	Test broadcast	aliasTestThree
<input type="checkbox"/>	Editar	Copiar	Borrar	5	2021-11-24 10:13:09	../uploads/attachments/image2.png	4	8	Test broadcast	aliasTestThree
<input type="checkbox"/>	Editar	Copiar	Borrar	23	2021-11-24 11:06:39	NULL	4	12	<a class="btn btn-primary" href="../friend_request...	aliasTestThree
<input type="checkbox"/>	Editar	Copiar	Borrar	24	2021-11-24 11:15:29	NULL	2	1	Test unread message	aliasTestOne
<input type="checkbox"/>	Editar	Copiar	Borrar	25	2021-11-24 11:15:36	NULL	2	6	Test unread message	aliasTestOne
<input type="checkbox"/>	Editar	Copiar	Borrar	26	2021-11-24 11:15:54	../uploads/attachments/Horarios DW2E.pdf	2	3	Test unread message with pdf	aliasTestOne

4. Seven chats:

- Four private chats:
 - Chat 1 = testUserOne and testUserTwo
 - Chat 6 = testUserOne and userTestThree
 - Chat 7 = testUserTwo and userTestThree
 - Chat 8 = userTestThree and userTestFour
- One group:
 - Chat 3 = userTestThree, testUserTwo and testUserOne
- Two friend requests:
 - Chat 4 = testUserTwo
 - Chat 12 = userTestFour

				codChat	chatName	chatImageURI
<input type="checkbox"/>	 Editor	 Copiar	 Borrar	1	privatechat	../uploads/chatsImage/default_chat_image/chat1.png
<input type="checkbox"/>	 Editor	 Copiar	 Borrar	6	privatechat	../uploads/chatsImage/default_chat_image/chat1.png
<input type="checkbox"/>	 Editor	 Copiar	 Borrar	7	privatechat	../uploads/chatsImage/default_chat_image/chat1.png
<input type="checkbox"/>	 Editor	 Copiar	 Borrar	8	privatechat	../uploads/chatsImage/default_chat_image/chat1.png
<input type="checkbox"/>	 Editor	 Copiar	 Borrar	3	My New Group	../uploads/chatsImage/default_chat_image/chat1.png
<input type="checkbox"/>	 Editor	 Copiar	 Borrar	4	Friend request	../uploads/chatsImage/default_chat_image/friend_req...
<input type="checkbox"/>	 Editor	 Copiar	 Borrar	12	Friend request	../uploads/chatsImage/default_chat_image/friend_req...

Technical explanation

1. Register a new user:

- The default page is the *login_new.php*, which allows a new user to be registered.
- The image files are checked via: *mime_content_type(string \$filename): string*; for improved security, instead of using *getimagesize()*.

```
if (is_uploaded_file($file['tmp_name'])) {  
    $mime_type = mime_content_type($file['tmp_name']);  
  
    $allowed_file_types = ['image/jpg', 'image/jpeg', 'png', 'image/png', 'image/gif'];  
    if (!in_array($mime_type, $allowed_file_types)) {  
        $errors[] = "Sorry, only JPG, JPEG, PNG & GIF files are allowed.";  
        $evOK = false;  
    }  
}
```

Also *is_uploaded_file(file)* is used, in order to only allow uploaded user files to be uploaded to the server.

If the file already exists the same file will be used.

- When a user is registered a *INSERT query* is done to the database, with the column *activated* = 0, then, when a user clicks in the email's link confirmation, it's redirected to the *login_already.php* page, where the code checks if it comes from an email and performs an UPDATE query to set *activated* to 1.
- In order to accept the register, the user must click the email link within the next hour after registering and receiving the email.

```
$currentTime = date('Y-m-d H:i:s');  
$maxAllowedTime = date('Y-m-d H:i:s', strtotime('+1 hour', strtotime($registerTime)));  
if ($currentTime > $maxAllowedTime) {  
    $sql_str = "DELETE FROM users WHERE codUser like '$id'";  
    $db->query($sql_str);  
    $errors[] = "User with email: $email, has not been activated within the maximum time allowed. Please, register the user again.";  
} else {  
    $sql_str = "UPDATE users SET activated = '1' WHERE codUser like '$id'";  
    $db->query($sql_str);  
    $errors[] = "User successfully registered, you can now login";  
}
```

2. Login as a user:

- A simple login. You can login both with the alias or email.

3. Chats home explanation:

- One you are logged in, you will see the *chats_home.php* with your user id in the URL. This way the code is simplified to pass to AJAX the *codChat* easily (see next).
- As in the private chats you have to see the other user profile, it doesn't matter what image to see, nevertheless a default one is selected.
- The left chats section and the right messages section are both synchronized using AJAX. The code below shows two functions, triggered by a *setInterval()* each 1000 milliseconds (1 second). This way both chats and messages are updated almost immediately for the user.

The user data in JSON format is generated by *home_data.php* file, the JS code in *chats_home* get it through *chats_messages_json.php* and *chats_data_json.php*, this way we can pass to that intermediate .php file some parameters that are then sent to *home_data.php* in the GET URL.

- Load messages function:

```
function loadMessages() {
    var codChat = check_cookie_name("codChat");
    // If the cookie is undefined, means that the user isn't in a chat, so it's not necessary to update the messages
    if (codChat != undefined) {
        var xhttp = new XMLHttpRequest();
        xhttp.onreadystatechange = function() {
            // Parse the JSON file from
            if (this.readyState == 4 && this.status == 200) {
                // Parse the JSON file from
                console.log(this.response);
                const DATA = JSON.parse(this.response);
                // Some problems with JSON indexes, but is working
                for (let i = 1; i <= DATA.length; i += 2) {
                    // Check if the last message is already added to the chat
                    if (!document.body.contains(document.getElementById("message" + (DATA.length - 1)))) {
                        };
                        xhttp.open("GET", "../chat_messages_json.php?chat=" + codChat, true);
                        xhttp.send();
                        return false;
                    }
                }
            }
        };
        loadMessages();
        setInterval(loadMessages, 1000);
    }
}
```

- Load chats function:

```

function loadChats() {
    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function() {
        if (this.readyState == 4 && this.status == 200) {
            // Parse the JSON file from
            const DATA = JSON.parse(this.response);
            for (let i = 0; i < DATA.length; i++) {
                // Check if the last chat is already added to the chat list
                if (!document.body.contains(document.getElementById(DATA[i][0]))) {
                    var codChat = check_cookie_name("codChat");
                    // Check if the user actually is inside a chat
                    for (let j = 0; j < DATA.length; j++) {
                        if (DATA[j][0] == codChat) {
                            codChat = undefined;
                        }
                    }
                    xhttp.open("GET", "../chats_data_json.php?chat=" + codChat, true);
                    xhttp.send();
                    return false;
                }
            }
        }
    };
    loadChats();
    setInterval(loadChats, 1000);
}

```

- As we can see, a WhatsApp approach was intended in this application.
- Most of the interface was made with Bootstrap.
- In the same *chats_home* page, I integrated a modal inside the *Chat options* dropdown. This allows it to be sent to multiple recipients without redirecting to another page.
This implementation was quite easy, as it's only a form inside a modal with the action setted to a .php file. See code below:

```

<div class="dropdown">
  <button class="btn btn-primary dropdown-toggle" type="button" id="dropdownMenuButton1" data-bs-toggle="dropdown" aria-expanded="false">
    Chats options
  </button>
  <ul class="dropdown-menu" aria-labelledby="dropdownMenuButton1">
    <li><a class="dropdown-item" href="/new_chat.php" role="button">New chat</a></li>
    <li><a class="dropdown-item" href="/new_group_chat.php" role="button">New group</a></li>
    <li>
      <!-- Button trigger modal -->
      <button type="button" class="dropdown-item" data-bs-toggle="modal" data-bs-target="#exampleModal">
        Broadcast list
      </button>
    </li>
  </ul>
</div>
<!-- Modal broadcast list -->
<div class="modal fade" id="exampleModal" tabindex="-1" aria-labelledby="exampleModallabel" aria-hidden="true">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="exampleModallabel">New broadcast list</h5>
        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
      </div>
      <form action="/send_multiple_recipients.php" method="POST" enctype="multipart/form-data">
        <div class="form-group m-2">
          <!-- Every available user -->
          <select class="form-select m-1" multiple aria-label="multiple select example" name="groupParticipants[]" id="groupParticipants">
            <?php
              $query = "SELECT alias, codUser FROM users";
              $result = $db->query($query);
              $users = $result->fetchAll(PDO::FETCH_ASSOC);
              foreach ($users as $user) {
                $alias = $user['alias'];
                $cod = $user['codUser'];
                if ($alias != "admin" && $cod != $_SESSION['codUser']) {
                  echo "<option value='$cod'>$alias</option>";
                }
              }
            <?>
          </select>
          <textarea placeholder="What's on your mind...(max. 400 characters)" name="messageText" maxlength="400" class="m-1"></textarea>
          <input type="file" name="broadcastImage">
        </div>
        <div class="modal-footer">
          <button type="submit" class="btn btn-primary">Send</button>
          <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">Close</button>
        </div>
      </form>
    </div>
  </div>
</div>

```

- As mentioned earlier, almost all the user data from this page is loaded with AJAX.

4. Admin zone

- A simple Bootstrap table with all the users data and the possibility to delete them.

5. Friend request

- To achieve this required me to make some queries, since my project is chat-based, each friend request must be a custom chat with a single message where two buttons appear, one to accept the friend request and another to reject it. Both buttons have *href* attributes that references the same *friend_request.php* file, the URLs have set parameters to know if the user had accepted or not.

The process will be the following:

create chat -> insert participate -> insert buttons message -> accepts / not -> delete chat

6. Emails

- As I wanted nice looking emails, I needed to create them in separate sections and join them in PHP as the email body.

To achieve that I just needed to create two separate HTML files as shown below:



And join them:

```
$secondPart = file_get_contents(VIEWS_DIR . "second_part_registration_email.html");  
$body = $firstPart . $urlPart . $secondPart;  
sendMail($mail, $body, $subject = "Confirmation Email", $configData[0], $configData[1]);
```

In both the registration and recover password emails, are sent URLs with parameters.

7. Attached files:

- Both images and documents (PDFs) are treated the almost same way. I made a function in PHP (see code below):

```
function uploadFile($file, $target_dir, $image)  
{
```

This function can both upload PDFs and images (PNGs, GIFs, SVGs, WEBPs and JPEGs).

As all the chats are loaded with AJAX, I check if a file exists and what type of file is to create a `` element or a `<embed>` one, see the code below:

```

function getExtension(filename) {
    var parts = filename.split('.');
    return parts[parts.length - 1];
}

function isImage(filename) {
    var ext = getExtension(filename);
    switch (ext.toLowerCase()) {
        case 'jpg':
        case 'jpeg':
        case 'jfif':
        case 'pjpeg':
        case 'pjp':
        case 'gif':
        case 'png':
        case 'svg':
        case 'webp':
            return true;
    }
    return false;
}

```

```

// If the message contains a compatible file URI do the following:
if (DATA[i]['fileUri'] != null && isImage(DATA[i]['fileUri'])) {
    // Is is a path to a image, create a img element
    var imgAttachment = document.createElement("img");
    imgAttachment.setAttribute("src", DATA[i]['fileUri']);
    imgAttachment.style.width = "100px";
    content.appendChild(imgAttachment);
    contentText.style.position = "relative";
    contentText.style.top = "20px";
} else if (DATA[i]['fileUri'] != null && isDoc(DATA[i]['fileUri'])) {
    // Is is a path to a pdf, create a embed of type pdf
    var previewText = document.createElement("p");
    previewText.innerHTML = "Document Preview";
    content.appendChild(previewText);
    var pdfAttachment = document.createElement("embed");
    pdfAttachment.style.paddingRight = "11px";
    pdfAttachment.setAttribute("src", DATA[i]['fileUri']);
    pdfAttachment.setAttribute("type", "application/pdf");
    pdfAttachment.style.width = "400px";
    pdfAttachment.style.height = "400px";
    content.appendChild(pdfAttachment);
    var downloadLink = document.createElement("a");
    downloadLink.style.display = "block";
    downloadLink.setAttribute("href", DATA[i]['fileUri']);
    downloadLink.innerHTML = "View PDF";
    content.appendChild(downloadLink);
    contentText.style.position = "relative";
    contentText.style.top = "20px";
}

```


RUN THE PROJECT

The project comes with the required vendor and autoload.php files. If by default it doesn't work you will need to install [composer](#) **inside the project folder chatapp_PHP** and run the following command in order to include PHPmailer: `composer require phpmailer/phpmailer.`