

Universidad Nacional del Altiplano

Facultad de Ingeniería Mecánica Eléctrica, Electrónica y de Sistemas

Escuela Profesional de Ingeniería de Sistemas



Base de Datos II

Seguridad en CRUD Chat de Universidad con Usuarios

Presenta: Jorge G. Olarte

Codigo: 215167

Docente: Ing. Robert Romero



Puno-Perú

Enero 30, 2024



Tabla de Contenidos

Tabla de Contenidos.....	2
Introduccion.....	4
Previsualización del Proyecto (capturas interfaz):.....	5
Registro de Nuevo Usuario.....	5
Probar validación email lenguaje Regular(*@gmail.com).....	5
Probar validación de captcha:.....	6
Login del Usuario Creado.....	7
Inscripción a Cursos con el Usuario Creado (Departamento de Medicina).....	9
Chat con CRUD para cada Usuario.....	11
Podemos ver mensajes de otros usuarios en el chat de Anatomia Humana.....	11
Publicar Mensaje (se puede editar y eliminar solo si es del usuario).....	12
Editar Mensaje (Solo si es del Usuario).....	12
Logout apretando el nombre.....	13
Base de Datos Usada:.....	15
Diseño.....	15
Triggers para auditar modificaciones en la base de datos.....	16
Procedures.....	16
Explicación (capturas código):.....	18
Tree.....	18
PHP Imagen Captcha:.....	19
Php validar Captcha.....	19
Compara si el input es igual al pin del captcha generado.....	19
Revisar Sesiones activas.....	20
Conexion General.....	20
Php para Borrar Mensajes de un Chat.....	21
Php para Editar Mensajes de un Chat.....	21
Inscripcion a un Curso.....	22
Evitar Sql Injections con el uso de prepare y bind_param.....	22
Php para mostrar Mensajes de Chat(Editar / Eliminar).....	23
Evitar Sql Injections con el uso de prepare y bind_param.....	23
Php para Mostrar Departamentos Disponibles.....	24
Evitar Sql Injections con el uso de prepare y bind_param.....	24
Php que muestra cursos disponibles por departamento.....	25
Evitar Sql Injections con el uso de prepare y bind_param.....	25



Php que muestra Cursos a los que esta inscrito un usuario.....	26
Evitar Sql Injections con el uso de prepare y bind_param.....	26
Php para login.....	27
Validacion si usuario existe.....	27
Uso de sanitize_input.....	27
Evitar Sql Injections con el uso de prepare y bind_param.....	27
Php para logout:.....	28
Php para publicar mensaje en Chat.....	28
Evitar Sql Injections con el uso de prepare y bind_param.....	28
Php para Registro.....	29
Validacion si usuario no existe.....	29
Password Hash.....	29
Proteccion contra SQL Injections.....	29
AJAX solo funciones Importantes:.....	30
Ajax Mostrar Usuario si esta iniciado.....	30
Ajax Uso de Lenguaje Regular para validar Email.....	31
Ajax Uso de Lenguaje Regular para Validar Usuario.....	31
Ajax Registro con Captcha y llama Funciones de Validacion.....	32
Ajax Login con Captcha.....	33
Mejoras que se pueden realizar.....	34
Conclusion.....	35
1. Diseño y Estructura de la Base de Datos:.....	35
2. Seguridad de la Base de Datos:.....	35
3. Auditoría y Registro de Cambios:.....	35
4. Funcionalidades Avanzadas:.....	36



Introduccion

El Sistema de Gestión Académica es una plataforma web integral diseñada para la administración eficiente de información académica. El proyecto se enfoca en la seguridad de la base de datos, **garantizando la integridad de los datos** y protegiendo la privacidad de los usuarios, evitando **sniffers bots y sql injections**. De manera resumida el proyecto tiene:

Gestión de Departamentos y Cursos: El sistema permite la creación, modificación y eliminación de departamentos y cursos académicos. La información es almacenada de manera segura en la base de datos, utilizando **consultas preparadas** y transacciones **atómicas**.

Registro y Autenticación Seguros: Los usuarios se registran de forma segura con contraseñas encriptadas mediante el **algoritmo SHA-512**. El uso de **captchas** durante el registro y la autenticación garantiza la protección contra ataques automatizados.

Auditoría y Registro de Cambios: La implementación de **triggers** y **procedimientos almacenados** permite registrar cambios en la base de datos, **proporcionando un historial detallado de las operaciones realizadas**. Esto facilita la identificación de actividades sospechosas y la auditoría del sistema.

Chat Integrado en Cursos: Los estudiantes tienen acceso a un sistema de **chat** en tiempo real dentro de los cursos a los que están inscritos. Esta función promueve la colaboración y la interacción entre los participantes.

Gestión de Comentarios Segura: Los estudiantes pueden **eliminar y modificar solo sus propios comentarios**, proporcionando un control adicional sobre su participación en el sistema.



Previsualización del Proyecto (capturas interfaz):

Registro de Nuevo Usuario

Registro de usuario Guillermo departamento de Medicina (permite tener acceso solo a chats de cursos de medicina) funcional conectado y validado con base de datos y lenguaje regular..

A screenshot of a web application's 'Register' form. The form is titled 'Register' and contains the following fields: 'Username' with the value 'Guillermo', 'Password' with masked characters '*****', 'Email' with the value 'guillermo@gmail.com', and a 'Department' dropdown menu currently showing 'Medicine'. Below the dropdown is a captcha image. At the bottom of the form is a blue 'Register' button and a link that says 'Already have an account? Login'.

Probar validación email lenguaje Regular(*@gmail.com)

A screenshot of the same 'Register' form, but with an error message displayed. The 'Email' field now contains '*@gmail.com'. A modal dialog box is overlaid on the form, titled 'localhost', with the message 'Invalid email. Please enter a valid email address.' and an 'OK' button. The other fields, including the 'Department' dropdown (still set to 'Medicine') and the captcha (showing 'N1KJL'), remain visible.

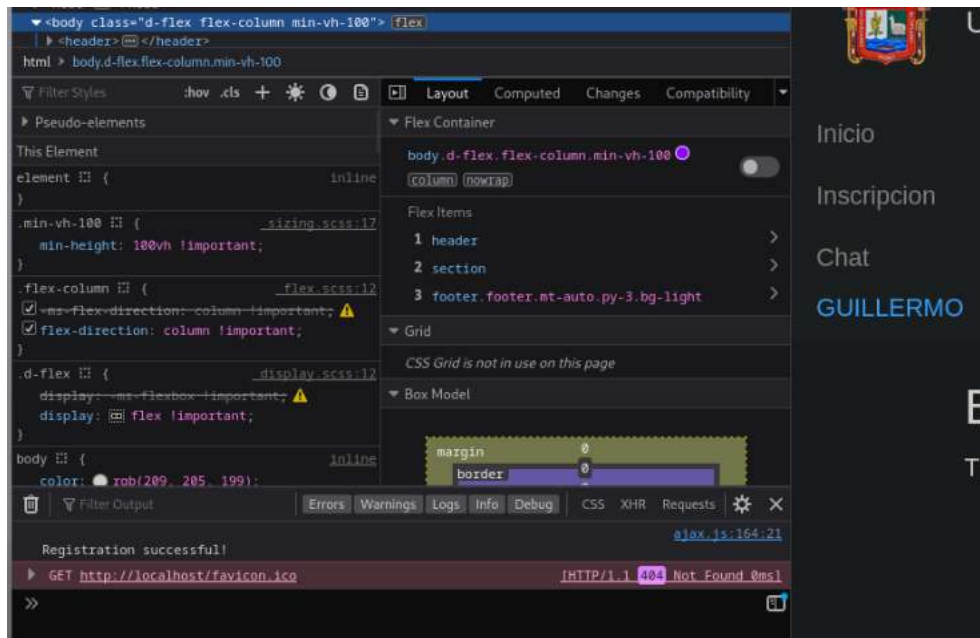


Probar validación de captcha:

A screenshot of a registration form on a dark background. The form has three input fields: 'Email:' with 'guillermo@gmail.com', 'Department:' with 'Medicine', and 'Captcha:' with 'AEQB3'. A modal error message box is overlaid on the right, showing a 'localhost' icon, the text 'Invalid Captcha. Please try again.', and an 'OK' button.































Probar registro correcto

- 1 En la izquierda se ve console.log de Registro Exitoso
- 2 En la derecha se muestra el Usuario (en este caso Guillermo) que después del registro automáticamente inicia sesión y se muestra su nombre(si en caso no haya sesion iniciada muestra boton de login/register.



En la base de datos:

Extra options

↔ T ↔		-	StudentID	Username	Password	Email	DepartmentID	
<input type="checkbox"/>	 Edit	 Copy	 Delete	1	milagrito	\$2y\$10\$5NsfnmK.Nmzwbp2qyAqgUuGTYjPfOT0QjP6q/1SMZv...	jorge@gmail.com	2
<input type="checkbox"/>	 Edit	 Copy	 Delete	2	milagrito2	\$2y\$10\$reaLY0XSsaPJRNc9VZw5qufIN8TvMCTOL/Bb72MURXp...	test2@gmail.com	1
<input type="checkbox"/>	 Edit	 Copy	 Delete	3	milagrito3	\$2y\$10\$3BE8DxDzYzRcQ53mALEKOEEdI0uGvvpuV8sWLizUIw...	test3@gmail.com	1
<input type="checkbox"/>	 Edit	 Copy	 Delete	4	milagrito4	\$2y\$10\$gb040Z4FtVzGThKCZfX2OGQ65T.mOwMeNIMpQ6Rrnf...	test4@gmail.com	1
<input type="checkbox"/>	 Edit	 Copy	 Delete	5	milagrito5	\$2y\$10\$MuYUB7/rr1U1MbcbgvDpgOXzA5NzFmMLxBhdYPIKCn...	test5@gmail.com	1
<input type="checkbox"/>	 Edit	 Copy	 Delete	6	milagrito6	\$2y\$10\$K5vAZXX2qgYnanzs0E8qB.IFVzqE9URbOpjl4xEU7rb...	test6@gmail.com	1
<input type="checkbox"/>	 Edit	 Copy	 Delete	7	milagrito19	\$2y\$10\$8KDWRAe/k70.6JfqdhyIOela3za5.4t6TPdRmZLUp6f...	milagrito19@gmail.com	1
<input type="checkbox"/>	 Edit	 Copy	 Delete	8	desSoft	\$2y\$10\$vnYuqk1Cy.14gEC/gHFYIOuqqj135ax5JfmrjC/KxT...	desSoft@gmail.com	2
<input type="checkbox"/>	 Edit	 Copy	 Delete	12	milagrito12	\$2y\$10\$5VnoqR6iIjrA5boZ16aiUeStWrWjZt5AsgZz.3nto/...	123@gmail.com	1
<input type="checkbox"/>	 Edit	 Copy	 Delete	13	Guillermo	\$2y\$10\$SCUT6FVjzdE2BRyIfg8R2.SiVnKRT6knnLBBTnu.Eac...	guillermo@gmail.com	1

Login del Usuario Creado

Login con cuenta guillermo@gmail.com captcha y validacion(por backend) funcional



Universidad Nacional del Altiplano Inicio Inscripción Chat

Login

Email:

Password:

Captcha:

Login

Don't have an account? [Register](#)

Inicio de Session(se ve usuario)

File View

HTML CSS JS XHR Fonts Images Media WS Other

Status	Met...	Domain	File	Initiator	Type	Transferred	Size
200	POST	localhost	validate_captcha.php	xhr	html	320 B	7 B
200	POST	localhost	login.php	jQuery 3.6.4	html	331 B	18 B
200	GET	localhost	index.html	ajax.googleapis.com	html	cached	2.67 kB
200	GET	localhost	check_session.php	jQuery 3.6.4	json	361 B	56 B
200	GET	code.jquery.com	jquery-3.5.1.slim.min.js		script	cached	0 B
200	GET	cdn.jsdelivr.net	popper.min.js		script	cached	0 B
200	GET	stackpath.bootstrapcdn.com	bootstrap.min.js		script	cached	0 B
200	GET	code.jquery.com	jquery-3.6.4.min.js		script	cached	0 B
200	GET	localhost	ajax.js		script	3.63 kB	13.09 kB
404	GET	localhost	favicon.ico	FaviconLoader	html	cached	271 B
200	GET	localhost	check_session.php	jQuery 3.6.4	json	361 B	56 B
200	GET	localhost	fetch_departments.php	jQuery 3.6.4	json	304 B	108 B
200	GET	localhost	get_dep_courses.php	jQuery 3.6.4	json	451 B	145 B
200	GET	localhost	get_user_courses.php	jQuery 3.6.4	html	313 B	1 B

14 requests 16.63 kB / 6.07 kB transferred Finish: 274 ms DOMContentLoaded: 246 ms Load: 248 ms

Filter Output Errors Warnings Logs Info Debug CSS XHR Requests

Login successful!

GET http://localhost/favicon.ico [HTTP/1.1] 404 Not Found [mal]

Universidad Nacional

Inicio
Inscripcion
Chat
GUILLERMO

Bienvenido al

This is the home page of the



Inscripción a Cursos con el Usuario Creado (Departamento de Medicina)

Ya que el usuario esta registrado al departamento de medicina solo puede inscribirse a cursos de esta carrera:

Chats Disponibles al registrarse (Vacío ya que hay que inscribirse primero) ;

The screenshot shows a web interface for the Universidad Nacional del Altiplano. At the top left is the university's logo and name. At the top right are navigation links: 'Inicio', 'Inscripcion', and 'Chat GUILLERMO'. The main heading is 'Selecciona un Curso para el Chat'. Below this, there is a label 'Cursos Disponibles:' followed by a dropdown menu. The dropdown menu is currently empty. Below the dropdown is a green button labeled 'Ingresar al Chat'.

Inscripcion:

The screenshot shows a web interface for the Universidad Nacional del Altiplano. At the top left is the university's logo and name. At the top right are navigation links: 'Inicio', 'Inscripcion', and 'Chat GUILLERMO'. The main heading is 'Inscripción en Cursos'. Below this, there is a label 'Cursos Disponibles:' followed by a dropdown menu. The dropdown menu shows 'Anatomia Humana' as the selected option. Below the dropdown is a green button labeled 'Inscribirse'.

Tenemos solo acceso a cursos de medicina (departamento del usuario)



Universidad Nacional del Altiplano

Inicio Inscripción Chat GUILLERMO

Inscripción en Cursos

Cursos Disponibles:

- Anatomía Humana
- Anatomía Humana
- Farmacología Básica
- Introducción a la Cirugía

Inscripcion Realizada

Universidad Nacional del Altiplano

Inicio Inscripción Chat GUILLERMO

Inscripción en Cursos

Cursos Disponibles:

- Anatomía Humana

Inscribirse

locathert

SISTEMA

OK

Chats disponibles despues de la inscripcion a curso

Universidad Nacional del Altiplano

Inicio Inscripción Chat GUILLERMO

Selección de un Curso para el Chat

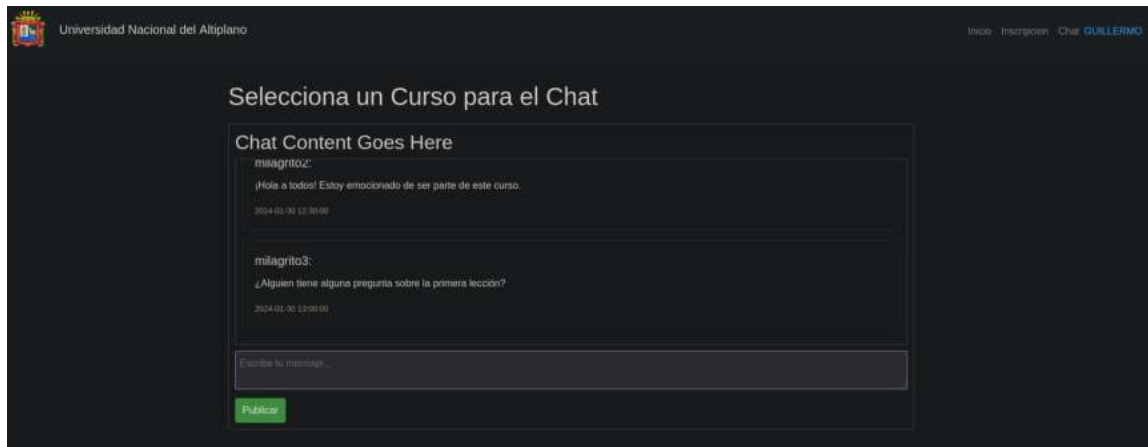
Cursos Disponibles:

- Anatomía Humana
- Anatomía Humana



Chat con CRUD para cada Usuario

Podemos ver mensajes de otros usuarios en el chat de Anatomia Humana



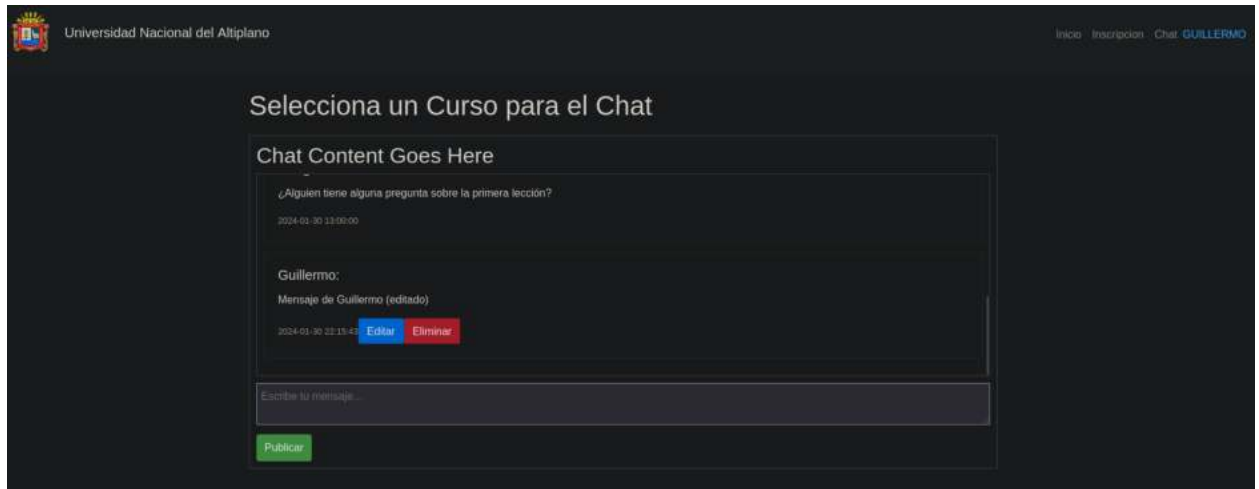


Publicar Mensaje (se puede editar y eliminar solo si es del usuario)

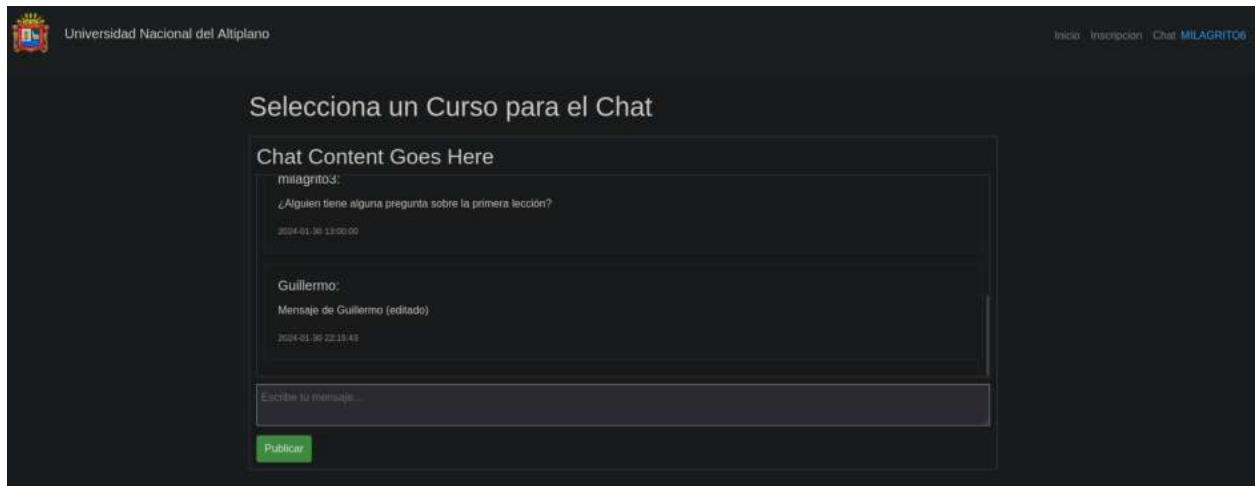
The screenshot shows the 'Selecciona un Curso para el Chat' page. Under 'Chat Content Goes Here', there is a message from Guillermo: '¿Alguien tiene alguna pregunta sobre la primera lección?' dated 2024-01-30 13:00:00. Below it, another message from Guillermo is shown with 'Mensaje de Guillermo' and a timestamp of 2024-01-30 23:15:47. This message has 'Editar' and 'Eliminar' buttons. At the bottom, there is a text input field 'Escribe tu mensaje...' and a 'Publicar' button.

Editar Mensaje (Solo si es del Usuario)

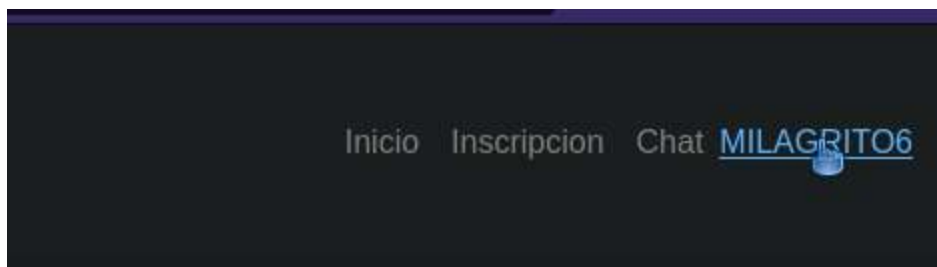
This screenshot is similar to the previous one, but with an edit dialog box open. The dialog box has a title 'localhost' and a subtitle 'Edit the post content:'. It contains a text input field with the text 'Mensaje de Guillermo (editado)'. There are 'Cancel' and 'OK' buttons at the bottom of the dialog. The background chat interface remains the same, showing the message from Guillermo with the 'Editar' button highlighted.



Ver mensaje desde otro usuario (Milagrito6 departamento de medicina)



Logout apretando el nombre.

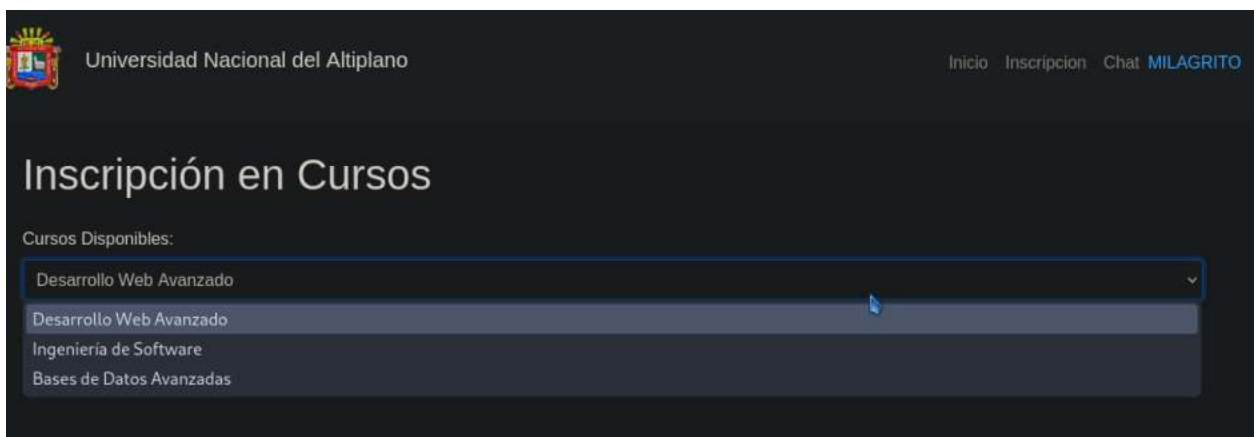




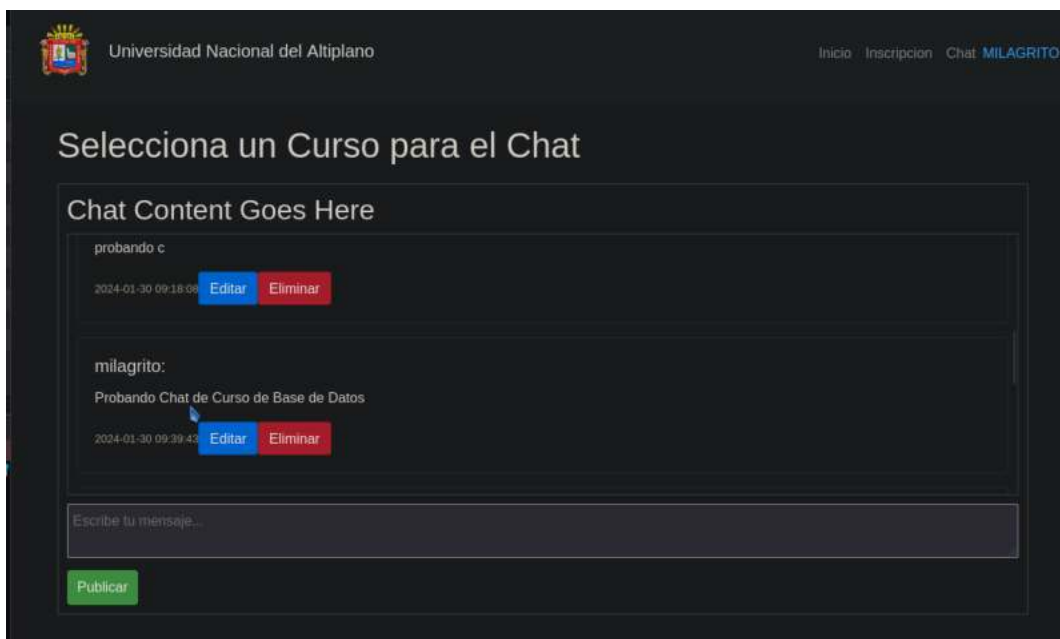
Se puede ver que se procesa un archivo.php para el logout y ahora muestra el botón de login/registro de nuevo



Ejemplo de otro departamento (Usuario Milagrito de Ingeniería de Software)



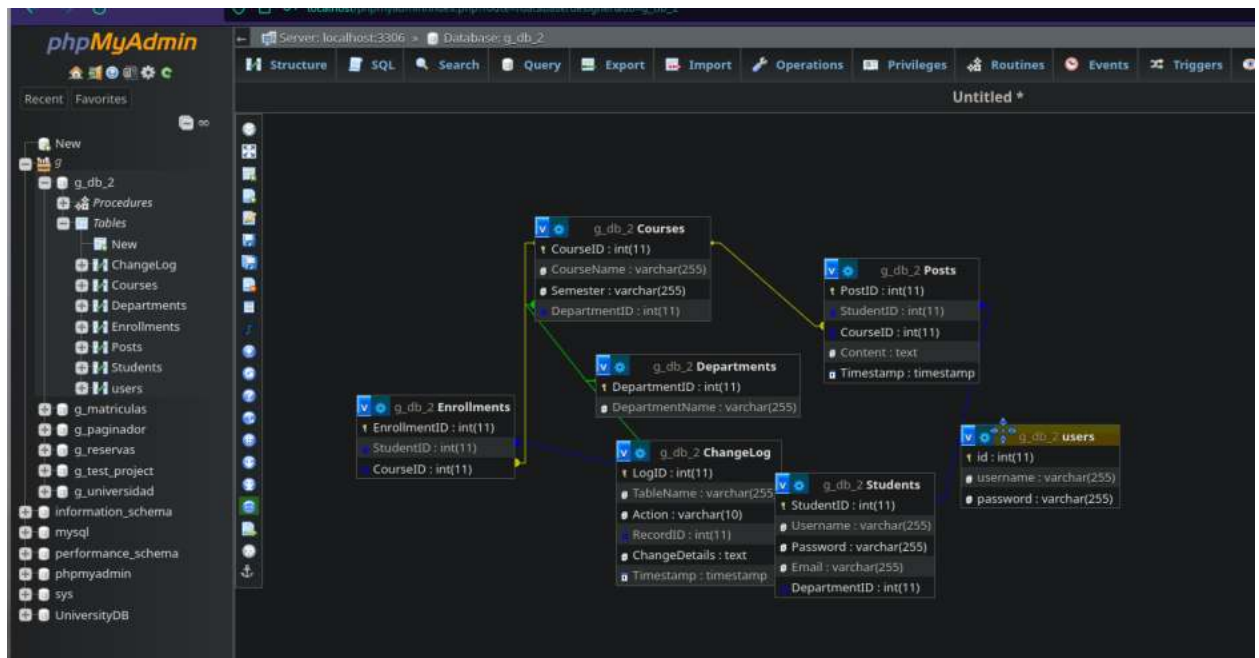
Chat Base de Datos (Departamento de Ingeniería de Software)





Base de Datos Usada:

Diseño





Triggers para auditar modificaciones en la base de datos

Server: localhost:3306 Database: g_db_2

Structure SQL Search Query Export Import Operations Privileges

Triggers

Check all Export Drop

Name	Table	Time	Event	
<input type="checkbox"/> AfterDeletePostsTrigger	Posts	AFTER	DELETE	Edit Export Drop
<input type="checkbox"/> AfterInsertUpdateDeletePostsTrigger	Posts	AFTER	INSERT	Edit Export Drop
<input type="checkbox"/> AfterUpdatePostsTrigger	Posts	AFTER	UPDATE	Edit Export Drop

Procedures

Server: localhost:3306 Database: g_db_2

Structure SQL Search Query Export Import Operations Privileges

Routines

Check all Export Drop

Name	Type	Returns	
<input type="checkbox"/> DeletePost	PROCEDURE		Edit Execute Export Drop
<input type="checkbox"/> EnrollStudentInCourse	PROCEDURE		Edit Execute Export Drop
<input type="checkbox"/> GetPostsInCourse	PROCEDURE		Edit Execute Export Drop
<input type="checkbox"/> InsertPost	PROCEDURE		Edit Execute Export Drop
<input type="checkbox"/> RemoveStudentFromCourse	PROCEDURE		Edit Execute Export Drop



Cada uno cumple atomicidad

The screenshot shows a database IDE with a list of stored procedures on the left and an 'Edit' window in the center. The 'Edit' window contains the following SQL code:

```
3 DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET exitHandler = TRUE;
4
5 START TRANSACTION;
6
7 INSERT INTO Enrollments (StudentID, CourseID) VALUES (studentID,
8
9 IF exitHandler THEN
10     ROLLBACK;
11 ELSE
12     COMMIT;
13 END IF;
14 END
```

The code is line-numbered from 3 to 14. The IDE interface includes buttons for 'Edit', 'Execute', 'Export', and 'Drop' for each procedure in the list. The 'Edit' window has a 'Go' button and a 'Close' button at the bottom right.



Explicación (capturas código):

Tree

Estructura del Proyecto (Frontend / Backend)

```
db_2
├── .git
├── U backend
│   ├── U captcha.jpg
│   ├── U captcha.php
│   ├── • check_session.php
│   ├── conexion.php
│   ├── delete_post.php
│   ├── edit_post.php
│   ├── enroll.php
│   ├── U fetch_departments.php
│   ├── U fuentexd.ttf
│   ├── U get_dep_courses.php
│   ├── • get_posts.php
│   ├── U get_user_courses.php
│   ├── login.php
│   ├── logout.php
│   ├── publish_post.php
│   ├── register.php
│   └── U validate_captcha.php
├── • frontend
│   ├── assets
│   │   └── logos
│   ├── css
│   │   └── style.css
│   ├── js
│   │   ├── JS • ajax.js
│   │   └── JS app.js
│   ├── pages
│   │   └── index.html
│   └── README.md
└── NvimTree_1 [-]
```



PHP Imagen Captcha:

```
Explorer
db_2
  .git
  U backend
    U captcha.jpg
    U captcha.php
    * check_session.php
    * conexion.php
    * delete_post.php
    * edit_post.php
    * enroll.php
    * U fetch_departments.php
    * U fuentexd.ttf
    * U get_dep_courses.php
    * U get_posts.php
    * U get_user_courses.php
    * login.php
    * logout.php
    * publish_post.php
    * register.php
    * U validate_captcha.php
  frontend
    assets
      logos
    css
      style.css
    js
      * ajax.js
      * app.js
    pages
      * index.html
  README.md

captcha.php > @ generateRandomString
<?php
session_start();

// Function to generate a random string for captcha
function generateRandomString($length = 5) {
    $characters = '1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZ';
    $charactersLength = strlen($characters);
    $randomString = '';
    for ($i = 0; $i < $length; $i++) {
        $randomString .= $characters[rand(0, $charactersLength - 1)];
    }
    return $randomString;
}

// Generate CAPTCHA
$captchaString = generateRandomString();

// Store CAPTCHA in session
$_SESSION['captcha_code'] = $captchaString;

// Path to the Background Image on your server
$backgroundImagePath = './captcha.jpg';

// Load the Background Image
$backgroundImage = imagecreatefromjpeg($backgroundImagePath);

// Create a 40 image of the same size as the background image
$image = imagecreatetruecolor(imagesx($backgroundImage), imagesy($backgroundImage));

// Copy the background image to the new image
imagecopy($image, $backgroundImage, 0, 0, 0, 0, imagesx($backgroundImage), imagesy($backgroundImage));

// Define colors
$textColor = imagecolorallocate($image, 0, 0, 0); // Text in black

// Draw lines and dots for additional distortion
for ($i = 0; $i < 5; $i++) {
    imageline($image, mt_rand(0, imagesx($image)), mt_rand(0, imagesy($image)), mt_rand(0, imagesx($image)), mt_rand(0, imagesy($image)), $textColor);
    imagesetpixel($image, mt_rand(0, imagesx($image)), mt_rand(0, imagesy($image)), $textColor);
}
```

Php validar Captcha

Compara si el input es igual al pin del captcha generado

```
Explorer
db_2
  .git
  U backend
    U captcha.jpg
    U captcha.php
    * check_session.php
    * conexion.php
    * delete_post.php
    * edit_post.php
    * enroll.php
    * U fetch_departments.php
    * U fuentexd.ttf
    * U get_dep_courses.php
    * U get_posts.php
    * U get_user_courses.php
    * login.php
    * logout.php
    * publish_post.php
  frontend
    assets
      logos
    css
      style.css
    js
      * ajax.js
      * app.js
    pages
      * index.html
  README.md

validate_captcha.php
1 <?php
session_start();

if (isset($_POST['captcha'])) {
    $enteredCaptcha = strtoupper($_POST['captcha']);
    $actualCaptcha = strtoupper($_SESSION['captcha_code']);

    if ($enteredCaptcha === $actualCaptcha) {
        echo 'success';
    } else {
        echo 'error';
    }
} else {
    echo 'error';
}
?>
```



Revisar Sesiones activas

Check_session.php

```
1 <?php
2 session_start();
3 $response = array();
4
5 if (isset($_SESSION['user'])) {
6     $response['loggedin'] = true;
7     $response['userDetails']['username'] = htmlspecialchars($_SESSION['user'], ENT_QUOTES, 'UTF-8'); // Sanitize user input
8 } else {
9     $response['loggedin'] = false;
10 }
11
12 header('Content-Type: application/json');
13 echo json_encode($response);
14 ?>
```

Conexion General

conexion.php

```
1 <?php
2 $servername = "localhost";
3 $username = "root";
4 $password = "root";
5 $dbname = "g_db_2";
6
7 $conn = new mysqli($servername, $username, $password, $dbname);
8
9 if ($conn->connect_error) {
10     die("Error de conexión: " . $conn->connect_error);
11 }
12 ?>
```



Php para Borrar Mensajes de un Chat

```
1 <?php
2 session_start();
3 require("conexion.php");
4
5 if ($_SERVER["REQUEST_METHOD"] === "POST" && isset($_POST['postID'])) {
6     $postID = $_POST['postID'];
7
8     // DELETE FROM Posts WHERE PostID = ?
9     $deletePostQuery = "DELETE FROM Posts WHERE PostID = ?";
10    $stmtDeletePost = $conn->prepare($deletePostQuery);
11    $stmtDeletePost->bind_param("i", $postID);
12
13    if ($stmtDeletePost->execute()) {
14        echo "success";
15    } else {
16        echo "error";
17    }
18
19    $stmtDeletePost->close();
20 } else {
21     echo "error";
22 }
23
24 $conn->close();
25
26 ?>
```

Php para Editar Mensajes de un Chat

```
1 <?php
2 session_start();
3 require("conexion.php");
4
5 if ($_SERVER["REQUEST_METHOD"] === "POST" && isset($_POST['postID']) && isset($_POST['content'])) {
6     $postID = $_POST['postID'];
7     $newContent = $_POST['content'];
8
9     // UPDATE Posts SET Content = ? WHERE PostID = ?
10    $editPostQuery = "UPDATE Posts SET Content = ? WHERE PostID = ?";
11    $stmtEditPost = $conn->prepare($editPostQuery);
12    $stmtEditPost->bind_param("si", $newContent, $postID);
13
14    if ($stmtEditPost->execute()) {
15        echo "success";
16    } else {
17        echo "error";
18    }
19
20    $stmtEditPost->close();
21 } else {
22     echo "error";
23 }
24
25 $conn->close();
26
27 ?>
```




Inscripcion a un Curso

Evitar Sql Injections con el uso de prepare y bind_param

```
db_2 Explorer
├── .git
├── U backend
├── U captcha.jpg
├── U captcha.php
├── U check_session.php
├── U conexion.php
├── U delete_post.php
├── U edit_post.php
├── U enroll.php
├── U fetch_departments.php
├── U fuente.txt
├── U get_dep_courses.php
├── U get_posts.php
├── U get_user_courses.php
├── U login.php
├── U logout.php
├── U publish_post.php
├── U register.php
├── U validate_captcha.php
├── U frontend
├── assets
├── logos
├── css
├── style.css
├── js
├── JS ajax.js
├── JS app.js
├── pages
├── index.html
└── README.md

enroll.php
require("conexion.php");

if ($_SERVER["REQUEST_METHOD"] === "POST" && isset($_POST['enroll'])) {
    $username = $_SESSION['user'];
    $selectedCourses = isset($_POST['courses']) ? $_POST['courses'] : [];

    // Eliminar inscripciones anteriores del estudiante para los cursos seleccionados
    $getStudentIDQuery = "SELECT StudentID FROM Students WHERE Username = ?";
    $stmtStudentID = $conn->prepare($getStudentIDQuery);
    $stmtStudentID->bind_param("s", $username);
    $stmtStudentID->execute();
    $resultStudentID = $stmtStudentID->get_result();
    $rowStudentID = $resultStudentID->fetch_assoc();
    $studentID = $rowStudentID['StudentID'];

    // Eliminar inscripciones anteriores del estudiante para los cursos seleccionados
    $deleteEnrollmentsQuery = "DELETE FROM Enrollments WHERE StudentID = ? AND CourseID = ?";
    $stmtDelete = $conn->prepare($deleteEnrollmentsQuery);
    $stmtDelete->bind_param("ii", $studentID, $courseID);

    foreach ($selectedCourses as $courseID) {
        $stmtDelete->execute();
    }

    $stmtDelete->close();

    // Inscribir al estudiante en los cursos seleccionados
    $enrollQuery = "INSERT INTO Enrollments (StudentID, CourseID) VALUES (?, ?)";
    $stmtEnroll = $conn->prepare($enrollQuery);
    $stmtEnroll->bind_param("ii", $studentID, $courseID);

    foreach ($selectedCourses as $courseID) {
        $stmtEnroll->execute();
    }

    $stmtEnroll->close();
    $conn->close();

    echo "success";
    exit();
}
```



Php para mostrar Mensajes de Chat(Editar / Eliminar)

Evitar Sql Injections con el uso de prepare y bind_param

```
db_2 Explorer
├── .git
├── U backend
│   ├── U captcha.jpg
│   ├── U captcha.php
│   ├── check_session.php
│   ├── conexion.php
│   ├── delete_post.php
│   ├── edit_post.php
│   ├── enroll.php
│   ├── U fetch_departments.php
│   ├── U get_dep_courses.php
│   ├── U get_posts.php
│   ├── U get_user_courses.php
│   ├── login.php
│   ├── logout.php
│   ├── publish_post.php
│   ├── register.php
│   └── U validate_captcha.php
├── frontend
│   ├── assets
│   │   ├── logos
│   │   ├── style.css
│   │   └── js
│   │       ├── ajax.js
│   │       ├── app.js
│   │       ├── pages
│   │       └── index.html
│   └── README.md
└── U get_posts.php

// Verificar si se proporcionó el ID del curso
$courseID = isset($_GET['courseID']) ? $_GET['courseID'] : null;

if ($courseID == null) {
    echo "Error: No se proporcionó el ID del curso.";
} else {
    // Preparar la consulta SQL para evitar inyecciones
    $getPostsQuery = "SELECT Posts.PostID, Posts.Content, Students.Username, Posts.Timestamp
    FROM Posts
    INNER JOIN Students ON Posts.StudentID = Students.StudentID
    WHERE Posts.CourseID = ?
    ORDER BY Posts.Timestamp ASC";

    $stmt = $conn->prepare($getPostsQuery);
    $stmt->bind_param("i", $courseID);
    $stmt->execute();
    $result = $stmt->get_result();

    // Mostrar los posts en formato HTML (puedes personalizar esto según tus necesidades)
    while ($row = $result->fetch_assoc()) {
        echo "<div class='card mb-3' id='post_" . $row['PostID'] . "'>";
        echo "<div class='card-body'>";
        echo "<h3 class='card-title'>" . $row['Username'] . "</h3>";

        // Content element
        echo "<p class='card-text'>" . $row['Content'] . "</p>";

        // Timestamp element
        echo "<small class='text-muted'>" . $row['Timestamp'] . "</small>";

        // Check if the post belongs to the current user
        if ($SESSION['user'] == $row['Username']) {
            // Edit button
            echo "<button type='button' class='btn btn-primary edit-button' onclick='editPost(" . $row['PostID'] . ")'>Editar</button>";

            // Delete button
            echo "<button type='button' class='btn btn-danger delete-button' onclick='deletePost(" . $row['PostID'] . ")'>Eliminar</button>";
        }
    }
}
```



Php para Mostrar Departamentos Disponibles

Evitar Sql Injections con el uso de prepare y bind_param

```
db_2 Explorer
├── git
├── u backend
│   ├── u captcha.jpg
│   ├── u captcha.php
│   ├── check_session.php
│   ├── conexion.php
│   ├── delete_post.php
│   ├── edit_post.php
│   ├── enroll.php
│   └── u fetch_departments.php
├── u frontend.ttf
├── u get_dep_courses.php
├── u get_posts.php
├── u get_user_courses.php
├── login.php
├── logout.php
├── publish_post.php
├── register.php
├── u validate_captcha.php
├── frontend
├── assets
│   ├── logos
├── css
├── js
├── js ajax.js
├── js app.js
├── pages
├── index.html
└── README.md

fetch_departments.php
<?php
// Connect to MySQL database
<?php $conn = mysqli_connect('localhost', 'root', '', 'db_2');

// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

// Prepare statement
$query = "SELECT * FROM Departments";
$stmt = mysqli_prepare($conn, $query);

// Bind parameters (none in this case)
// Bind parameters (none in this case)

// Execute the statement
mysqli_stmt_execute($stmt);

// Get result set
$result = mysqli_stmt_get_result($stmt);

// Check for errors
if (!$result) {
    die("Error fetching departments");
}

// Fetch the data into an associative array
$departments = array();
while ($row = mysqli_fetch_assoc($result)) {
    $departments[] = $row;
}

// Return the data as JSON
header('Content-Type: application/json');
echo json_encode($departments);

// Close the statement and database connection
mysqli_stmt_close($stmt);
mysqli_close($conn);
?>
```




Php que muestra cursos disponibles por departamento

Evitar Sql Injections con el uso de prepare y bind_param

```
Explorer
db_2
  .git
  U backend
  U captcha.jpg
  U captcha.php
  U check_session.php
  U conexion.php
  U delete_post.php
  U edit_post.php
  U enroll.php
  U fetch_departments.php
  U fuente.txt
  U get_dep_courses.php
  U get_posts.php
  U get_user_courses.php
  U login.php
  U logout.php
  U publish_post.php
  U register.php
  U validate_captcha.php
  frontend
  assets
  logos
  css
  style.css
  js
  JS ajax.js
  JS app.js
  pages
  index.html
  README.md

get_dep_courses.php
1
<?php
session_start();
require('conexion.php');

// Verificar si el usuario está logueado
$username = $_SESSION['user'];
$departmentQuery = "SELECT DepartmentID FROM Students WHERE Username = ?";
$stmtDept = $conn->prepare($departmentQuery);
$stmtDept->bind_param("s", $username);
$stmtDept->execute();
$resultDept = $stmtDept->get_result();
$rowDept = $resultDept->fetch_assoc();
$departmentID = $rowDept['DepartmentID'];

// Mostrar cursos disponibles para el departamento del estudiante
$availableCoursesQuery = "SELECT * FROM Courses WHERE DepartmentID = ?";
$stmtCourses = $conn->prepare($availableCoursesQuery);
$stmtCourses->bind_param("i", $departmentID);
$stmtCourses->execute();
$resultCourses = $stmtCourses->get_result();

// Crear un array para almacenar la información de los cursos
$courses = [];

// Almacenar la información de los cursos en el array
while ($rowCourse = $resultCourses->fetch_assoc()) {
    $courses[] = [
        'id' => $rowCourse['CourseID'],
        'name' => $rowCourse['CourseName'],
    ];
}

// Convertir el array a formato JSON y enviarlo como respuesta
header('Content-Type: application/json');
echo json_encode($courses);

$stmtDept->close();
$stmtCourses->close();
$conn->close();
```



Php que muestra Cursos a los que esta inscrito un usuario

Evitar Sql Injections con el uso de prepare y bind_param

```
Explorer
db_2
  .git
  . U backend
    U captcha.jpg
    U captcha.php
    . check_session.php
    . conexion.php
    . delete_post.php
    . edit_post.php
    . enroll.php
    U fetch_departments.php
    U fuentexd.ttf
    U get_dep_courses.php
    . get_posts.php
    U get_user_courses.php
    . login.php
    . logout.php
    . publish_post.php
    . register.php
    U validate_captcha.php
  . frontend
    . assets
    . logos
    . css
      style.css
    . js
      . ajax.js
      . app.js
    . pages
      index.html
    README.md

get_user_courses.php
<?php
session_start();
require('conexion.php');

// Obtener el ID del usuario logueado
$username = $_SESSION['user'];

$stmtStudentIDQuery = "SELECT studentID FROM Students WHERE Username = ?";
$stmtStudentID = $conn->prepare($stmtStudentIDQuery);
$stmtStudentID->bind_param('s', $username);
$stmtStudentID->execute();
$resultStudentID = $stmtStudentID->get_result();
$rowStudentID = $resultStudentID->fetch_assoc();
$studentID = $rowStudentID['studentID'];

// Obtener cursos a los que está inscrito el estudiante
$enrolledCoursesQuery = "SELECT c.CourseID, c.CourseName FROM Courses c
INNER JOIN Enrollments e ON c.CourseID = e.CourseID
WHERE e.StudentID = ?";

$stmtEnrolledCourses = $conn->prepare($enrolledCoursesQuery);
$stmtEnrolledCourses->bind_param('i', $studentID);
$stmtEnrolledCourses->execute();
$resultEnrolledCourses = $stmtEnrolledCourses->get_result();

// Construir opciones de cursos para la respuesta AJAX
$options = '';
while ($rowEnrolledCourse = $resultEnrolledCourses->fetch_assoc()) {
    $options .= "<option value='". $rowEnrolledCourse['CourseID'] . "'>". $rowEnrolledCourse['CourseName'] . "</option>";
}

$stmtStudentID->close();
$stmtEnrolledCourses->close();
$conn->close();

echo $options;
?>
```



Php para login

Validacion si usuario existe

Uso de sanitize_input

Evitar Sql Injections con el uso de prepare y bind_param

```
login.php
<?php
session_start();
require("conexion.php");

// ... (code omitted) ...

function sanitize_input($data) {
    return htmlspecialchars(trim($data));
}

// Login User
if ($_SERVER["REQUEST_METHOD"] === "POST" && isset($_POST['login'])) {
    $email = sanitize_input($_POST['loginEmail']);
    $password = sanitize_input($_POST['loginPassword']);

    // Basic validation
    if (empty($email) || empty($password)) {
        echo "Please provide both email and password.";
    } else {
        // Check if email exists
        $check_email_query = "SELECT * FROM Students WHERE Email=?";
        $stmt = $conn->prepare($check_email_query);
        $stmt->bind_param("s", $email);
        $stmt->execute();
        $result = $stmt->get_result();

        if ($result->num_rows > 0) {
            // Verify password
            $row = $result->fetch_assoc();
            if (password_verify($password, $row['Password'])) {
                $_SESSION['user'] = $row['Username']; // Use the username or any other identifier
                echo "Login successful!";
            } else {
                echo "Incorrect password.";
            }
        } else {
            echo "Email not found.";
        }
    }

    $stmt->close();
}
```



Php para logout:

```
1 <?php
2 session_start();
3 session_destroy();
4 session_unset();
5 exit();
6 // Redirect to the home page if applicable
7 header('Location: index.php'); // Change this to the appropriate URL.
8 exit();
9 ?>
```

Php para publicar mensaje en Chat

Evitar Sql Injections con el uso de prepare y bind_param

```
1 <?php
2 session_start();
3 require("conexion.php");
4
5 // Verificar si el usuario está logueado
6 if ($SERVER["REQUEST_METHOD"] === "POST" && isset($_POST['content']) && isset($_POST['courseID'])) {
7     $content = $_POST['content'];
8     $courseID = $_POST['courseID'];
9
10    $username = $_SESSION['user'];
11    $departmentQuery = "SELECT StudentID FROM Students WHERE Username = ?";
12    $stmtDept = $conn->prepare($departmentQuery);
13    $stmtDept->bind_param("s", $username);
14    $stmtDept->execute();
15    $resultDept = $stmtDept->get_result();
16    $rowDept = $resultDept->fetch_assoc();
17    $studentID = $rowDept['StudentID'];
18
19    // $studentID = $_SESSION['studentID']; // Ya estás obteniendo el StudentID del usuario actual arriba
20
21    $publishPostQuery = "INSERT INTO Posts (StudentID, CourseID, Content) VALUES (?, ?, ?)";
22    $stmtPublishPost = $conn->prepare($publishPostQuery);
23    $stmtPublishPost->bind_param("iis", $studentID, $courseID, $content);
24
25    if ($stmtPublishPost->execute()) {
26        echo "success";
27    } else {
28        echo "error";
29    }
30
31    $stmtPublishPost->close();
32 } else {
33     echo "error";
34 }
35
36 $conn->close();
37 ?>
```



Php para Registro

Validacion si usuario no existe

Password Hash

Proteccion contra SQL Injections



```
db_2
  git
  db_2
    backend
      u captcha.jpg
      u captcha.php
      u check_session.php
      u conexion.php
      u delete_post.php
      u edit_post.php
      u enroll.php
      u fetch_departments.php
      u fuente.txt
      u get_dep_courses.php
      u get_posts.php
      u get_user_courses.php
      u login.php
      u logout.php
      u publish_post.php
      u register.php
      u validate_captcha.php
    frontend
      assets
        logos
      css
      js
      js
        ajax.js
      app.js
      pages
      index.html
  README.md

register.php
function sanitize_input($data) {
    return htmlspecialchars(trim($data));
}

// Register user
if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['register'])) {
    $username = sanitize_input($_POST['username']);
    $email = sanitize_input($_POST['email']);
    $password = password_hash($_POST['password'], PASSWORD_DEFAULT);
    $departmentID = intval($_POST['department']); // Assuming 'department' is the name attribute in the form

    // Check validation
    if (empty($username) || empty($email) || empty($_POST['password']) || empty($departmentID)) {
        echo "Please provide all required information.";
    } else {
        // Check if username or email already exists
        $check_user_query = "SELECT * FROM Students WHERE Username=?";
        $check_email_query = "SELECT * FROM Students WHERE Email=?";

        $stmt_user = $conn->prepare($check_user_query);
        $stmt_user->bind_param("s", $username);
        $stmt_user->execute();
        $result_user = $stmt_user->get_result();

        $stmt_email = $conn->prepare($check_email_query);
        $stmt_email->bind_param("s", $email);
        $stmt_email->execute();
        $result_email = $stmt_email->get_result();

        if ($result_user->num_rows > 0) {
            echo "Username already exists. Please choose a different username.";
        } elseif ($result_email->num_rows > 0) {
            echo "Email already exists. Please use a different email address.";
        } else {
            // Register the user
            $stmt_insert = $conn->prepare("INSERT INTO Students (Username, Password, Email, DepartmentID) VALUES (?, ?, ?, ?)");
            $stmt_insert->bind_param("sssi", $username, $password, $email, $departmentID);

            if ($stmt_insert->execute()) {
                // Redirect to login page
            }
        }
    }
}
```

AJAX solo funciones Importantes:

Ajax Mostrar Usuario si esta iniciado

```
JS ajax.js
1 // login register

$(document).ready(function () {
    // Check session status
    checkSessionStatus();

    // Register user
    function checkSessionStatus() {
        $.ajax({
            url: '/db_2/backend/check_session.php', // PHP file that checks session status
            type: 'GET',
            dataType: 'json', // Expected JSON response
            success: function (response) {
                // Update session status
                if (response.loggedIn) {
                    $('#session-container').html('<a href="/db_2/frontend/index.html" id="logout"> ' + response.userDetails.username.toUpperCase() + '</a>');
                    $('#login-register-container').hide();
                    $('#loginDropdown').hide();
                } else {
                    $('#session-container').html('');
                    $('#login-register-container').show();
                    $('#loginDropdown').show();
                }
            }
        });
    }
});
```




Ajax Uso de Lenguaje Regular para validar Email

```
31
32
33 //var emailFilter=/^[a-zA-Z0-9_-]+@[a-zA-Z0-9-]{1,64}(\.[a-z]{2,4}|[a-z]{2,3}\.[a-z]{2})$/i;
34
35 var emailFilter=/^[a-zA-Z0-9_-]+@[a-zA-Z0-9-]{1,64}(\.[a-z]{2,4}|[a-z]{2,3}\.[a-z]{2}){1,5}$/i;
36
37 var validEmail=emailFilter.test(email);
38
39
40 if (validEmail!=false) {
41     //alert('Mail valido');
42     return true;
43 }
44
45 else {
46     // alert('Mail no valido');
47     return false;
48 }
49
50 //alert('Oops!\n'+email);
51
52 } // fin validemail
53
```

Ajax Uso de Lenguaje Regular para Validar Usuario

```
90
91
92 function validespacios(user) {
93     //var emailFilter=/^[a-zA-Z0-9_-]+@[a-zA-Z0-9-]{1,64}(\.[a-z]{2,4}|[a-z]{2,3}\.[a-z]{2})$/i;
94     //alert('valido no es necesario');
95
96     user = $.trim(user);
97
98
99     var userFilter=/^[a-zA-Z0-9]{4,15}$/i;
100
101     //var userfilter=/^[a-zA-Z0-9]{1,10}$/i;
102
103     var validuser=userFilter.test(user);
104
105
106     if (validuser!=false) {
107         //alert('valido');
108         return true;
109     }
110
111     else {
112         //alert('no valido');
113         return false;
114     }
115
116 }
```



Ajax Registro con Captcha y llama Funciones de Validacion

```
JS ajax.js > @ ready() callback > @ submit() callback > @ success
128 $('#register-form').submit(function (event) {
129     event.preventDefault();
130
131     // Get the user input
132     var username = $('#username').val();
133     var email = $('#email').val();
134     var password = $('#password').val(); // 'password' is declared but its value is never read.
135     var department = $('#department').val(); // Assuming 'department' is the name attribute in the form // 'department' is declared but its value is never
136
137     // Validate username and email
138     if (!validespacios(username)) {
139         alert('Invalid username. Please enter a valid username without spaces.');
```

```
140         return;
141     }
142
143     if (!validemail(email)) {
144         alert('Invalid email. Please enter a valid email address.');
```

```
145         return;
146     }
147
148     // Get the captcha code entered by the user
149     var enteredCaptcha = $('#captcha').val();
150
151     // Validate the captcha using AJAX
152     $.ajax({
153         url: '/db_2/backend/validate_captcha.php',
154         type: 'POST',
155         data: { captcha: enteredCaptcha },
156         success: function (response) {
157             if (response === 'success') {
158                 // Captcha is valid, proceed with registration
159                 // Serialize the form data
160                 var formData = $('#register-form').serialize();
161
162                 // Submit registration form
163                 $.ajax({
164                     url: '/db_2/backend/register.php',
165                     type: 'POST',
166                     data: formData,
167                     success: function (data) {
```




Ajax Login con Captcha

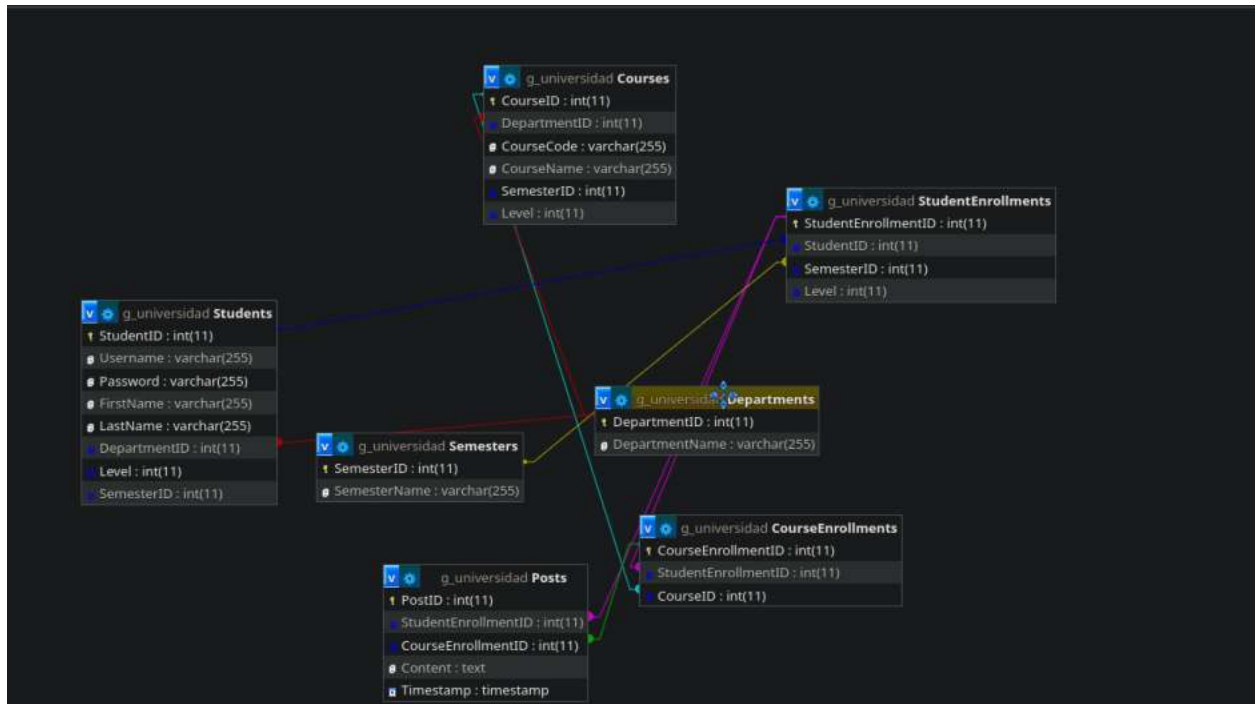
```
JS ajax.js > @ ready() callback > @ ready() callback > @ submit() callback > @ success

258 $(document).ready(function () {
259     $('#login-form').submit(function (e) {
260         e.preventDefault();
261
262         // Get the captcha code entered in the form
263         var enteredCaptcha = $('#captcha').val();
264
265         // Validate the captcha using ajax
266         $.ajax({
267             url: '/db_2/backend/validate_captcha.php',
268             type: 'POST',
269             data: { captcha: enteredCaptcha },
270             success: function (response) {
271                 if (response === 'success') {
272                     // Captcha is valid, proceed with login
273                     // Serialize the form data
274                     var formData = $('#login-form').serialize();
275
276                     $.ajax({
277                         url: '/db_2/backend/login.php',
278                         type: 'POST',
279                         data: formData, // Use the serialized form data
280                         success: function (data) {
281                             console.log(data);
282                             // After login, we check the session status
283                             document.location.href = '/db_2/frontend/index.html';
284                             checkSessionStatus();
285                         }
286                     });
287                 } else {
288                     alert('Invalid Captcha. Please try again.');
```



Mejoras que se pueden realizar

Expandir Base de datos





Conclusion

El proyecto de base de datos representa un hito significativo en la implementación de un sistema de gestión académica robusto y seguro. A lo largo del desarrollo, se han aplicado principios fundamentales de diseño y seguridad para garantizar la integridad de los datos, la confidencialidad de la información y la eficiencia operativa. A continuación, se presenta una conclusión general que destaca los aspectos clave del proyecto:

1. Diseño y Estructura de la Base de Datos:

Se ha diseñado una estructura de base de datos coherente y optimizada que refleja los requisitos y la lógica del sistema de gestión académica.

Las relaciones entre las tablas se han establecido de manera precisa para garantizar la integridad referencial y la coherencia de los datos.

2. Seguridad de la Base de Datos:

Se han implementado medidas de seguridad robustas para proteger la base de datos contra ataques maliciosos y vulnerabilidades de seguridad.

El uso de consultas preparadas, transacciones atómicas y encriptación de contraseñas garantiza la confidencialidad y la integridad de los datos sensibles.

3. Auditoría y Registro de Cambios:

Se ha incorporado un sistema de registro de cambios que permite realizar un seguimiento detallado de las operaciones realizadas en la base de datos.



Los triggers y procedimientos almacenados se han utilizado para registrar eventos importantes, lo que facilita la auditoría del sistema y la identificación de actividades sospechosas.

4. Funcionalidades Avanzadas:

El sistema incluye características avanzadas, como chat en tiempo real dentro de los cursos y gestión segura de comentarios, que promueven la colaboración y la participación activa de los usuarios.

La implementación de estas funcionalidades se ha realizado de manera segura, asegurando que los usuarios puedan interactuar de forma protegida y eficiente.

En resumen, el proyecto de base de datos no solo cumple con los requisitos funcionales del sistema de gestión académica, sino que también establece un estándar de seguridad, eficiencia y usabilidad.