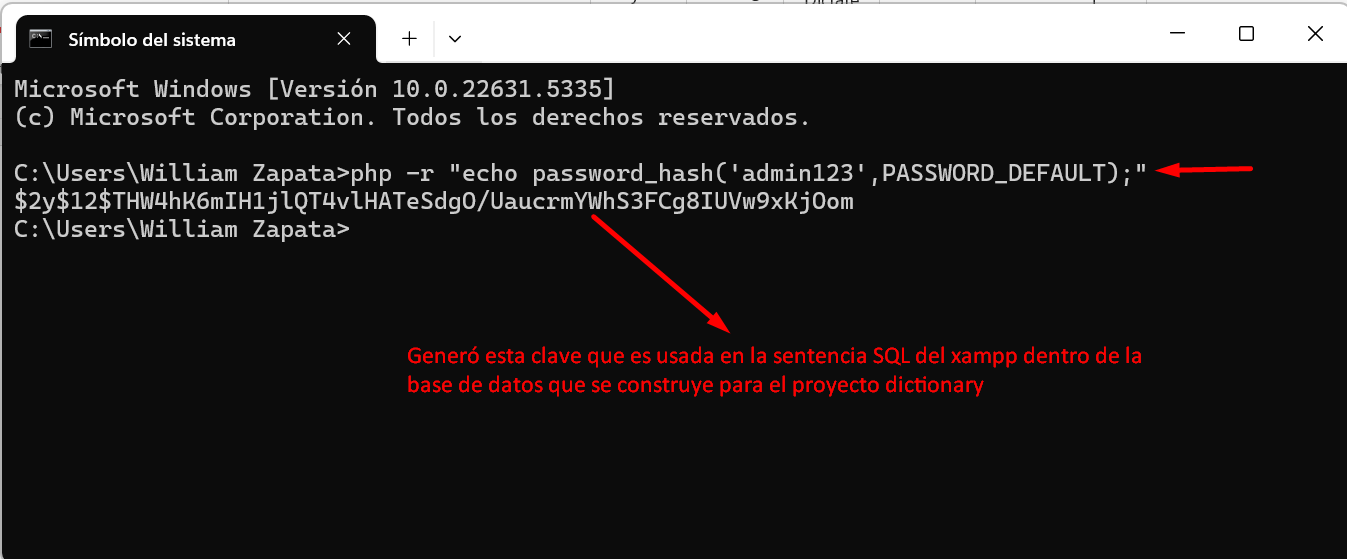
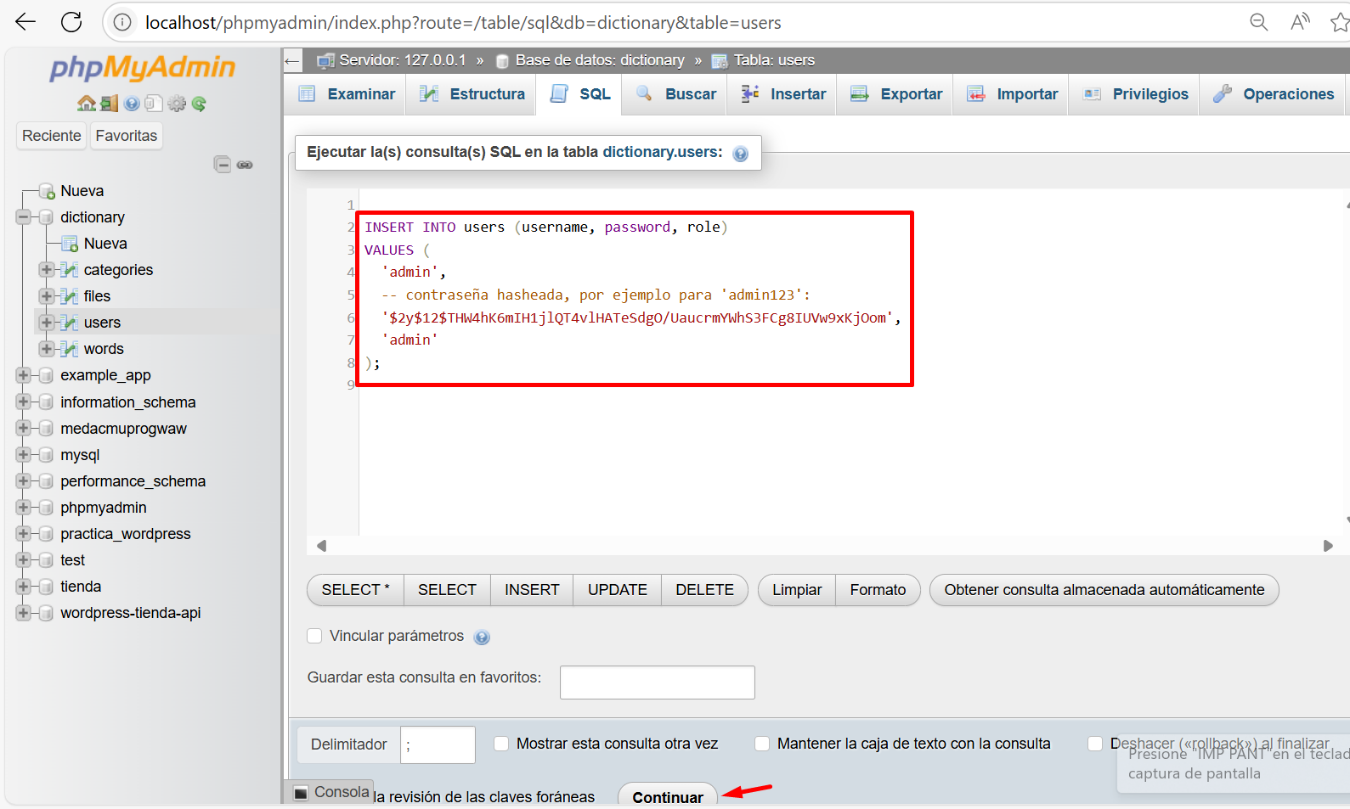
**DICTIONARY**

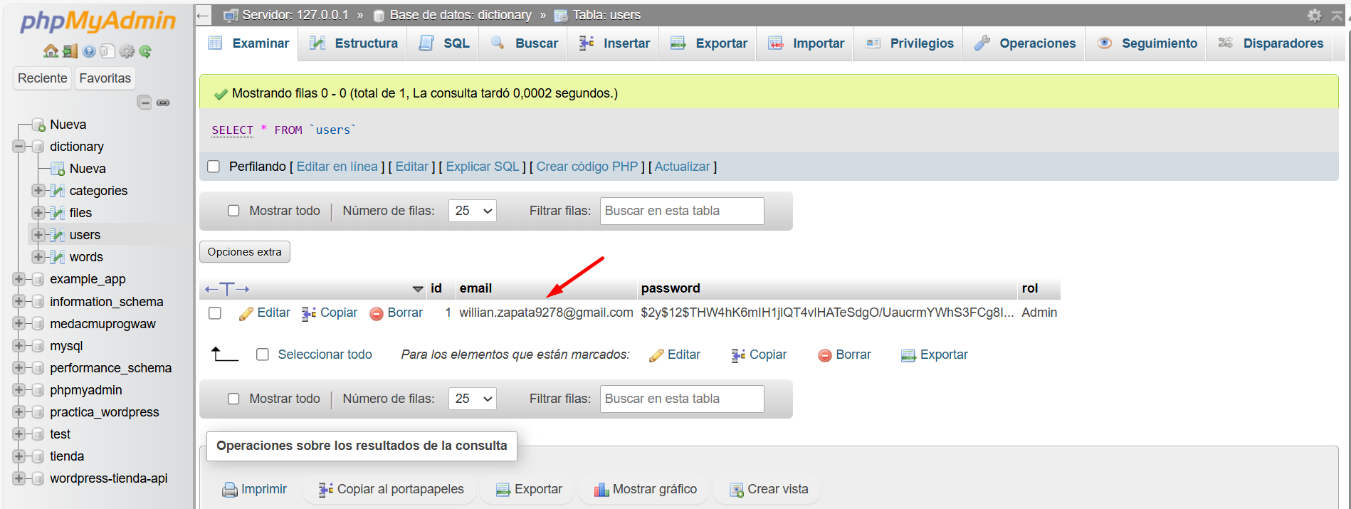
Para crear el hash o la contraseña para la creación del primer usuario admin se ejecutó el siguiente comando en la línea de comandos de Windows:





* Usuario: admin
* Contraseña: admin123

Por cuestiones de mejorar el diseño y accesibilidad se ha modificado el tipo de usuario por el siguiente:



* Usuario: willian.zapata9278@gmail.com
* Contraseña: admin123

**1-Base de datos**

Para este proyecto se ha creado la base de datos dictionary en el xampp usando MySQL con la siguiente estructura:

Tabla categories con las siguientes columnas:

* Id
* name

Tabla files con las siguientes columnas:

* Id
* name
* extension
* path(aquí va la ruta de la ubicación de las imágenes)

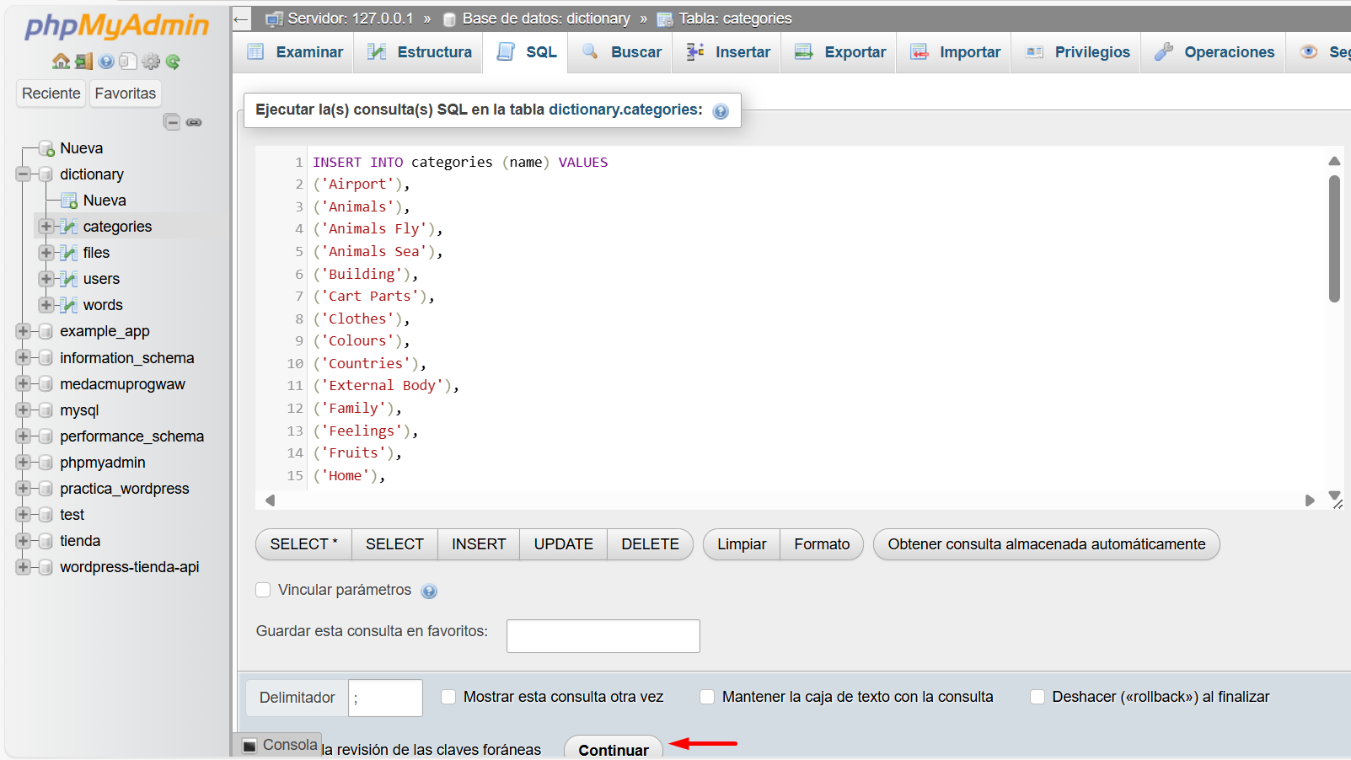
Tabla users con las siguientes columnas:

* Id
* Email
* Password
* Rol

Tabla words con las siguientes columnas:

* Id
* Word\_in (palabra que se ingresa)
* Meaning (traducción o significado)
* Date\_register
* Category\_id (foreign key de la tabla category)
* File\_id (foreign key de la tabla file)

Primera inserción de datos en la tabla categories:



**2-Estructura de archivos del proyecto**

|  |  |  |
| --- | --- | --- |
| Directorio | Subdirectorio | Archivos |
| dictionary | server/controller | Controller.php, ActionLogin.php, ActionLogout.php, ActionGetWord.php, ActionAddWord.php, ActionUpdateWord.php, ActionDeleteWord.php, ActionSearchWord.php, ActionGetWordsByCategory.php, authStatus.php |
| dictionary | server/daos | db.php, session.php |
| dictionary | public | adminPanel.php, dictionary.html, category.html, login.php, index.html |
| dictionary | public/images | airport, animals, animalsFly, ... (otras categorías) |
| dictionary | public/js | include.js, admin.js, dictionary.js, category.js |
| dictionary | public/css | style.css |
| dictionary | public/parts | navbar.html, footer.html |
| dictionary | public/thirdParty/bootstrap-5.3.3-dist/css | bootstrap.min.css |
| dictionary | public/thirdParty/bootstrap-5.3.3-dist/js | bootstrap.bundle.min.js |

Los archivos PHP en server/controller/ son los que realmente hacen el CRUD en la base de datos:

* ActionAddWord.php → crea una palabra.
* ActionGetWord.php → consulta una palabra por ID (para editarla).
* ActionUpdateWord.php → actualiza una palabra existente.
* ActionDeleteWord.php → elimina una palabra.

Estos archivos **reciben peticiones desde el frontend**, se comunican con la base de datos (vía daos/db.php), y **devuelven respuestas JSON** al navegador.  
Esto es el verdadero CRUD.

El archivo admin.js no hace CRUD directamente en la base de datos ... pero **orquesta todo el proceso desde el navegador del admin**. Es decir:

* Detecta cuando el admin da clic en "Agregar", "Editar" o "Eliminar".
* Toma los datos del formulario.
* Llama (vía fetch() o XMLHttpRequest) a los archivos PHP como:
  + ActionAddWord.php
  + ActionUpdateWord.php
  + etc.

Recibe la respuesta y actualiza dinámicamente la interfaz de adminPanel.php.

¿Cómo manejar las imágenes cuando se agrega o se va a editar una palabra?

Usar una intefaz personalizada para:

Leer los archivos desde public/images/ (lo hace PHP)

Mostrar las imágenes al admin

Dejar que el admin seleccione una

Enviar el path relativo con texto ($\_POST['image\_path'])

Guardarlo directamente en la base de datos