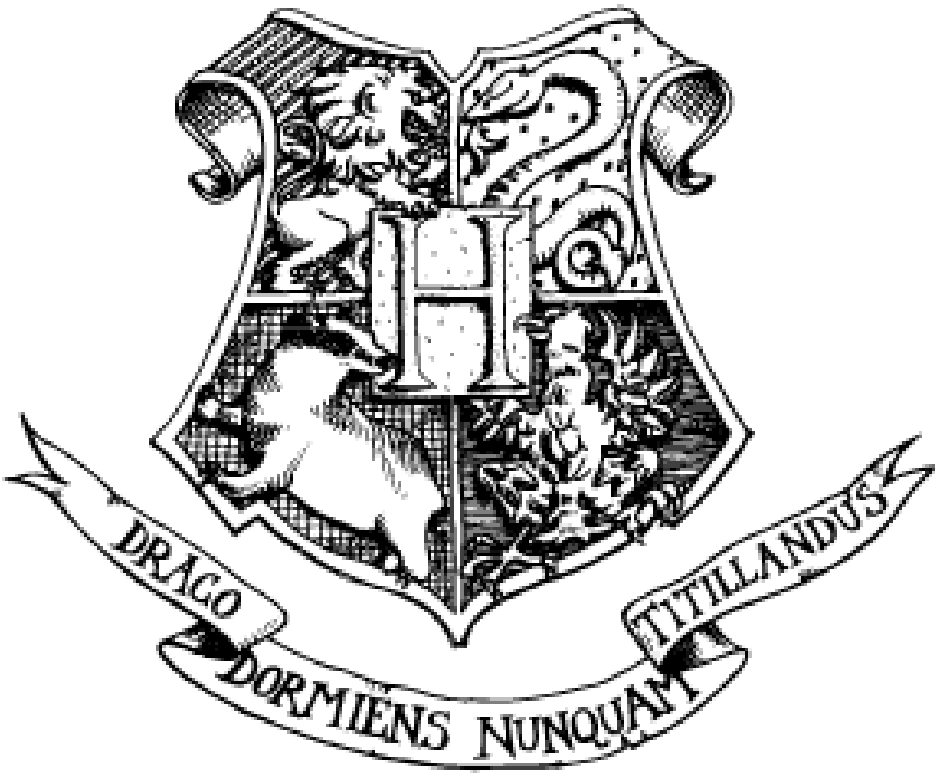
DISEÑO TÉCNICO DEL SISTEMA

Hogwarts



Nombre del fichero:	DOC_DisenoTecnicoDelSistemaHogwarts.pdf
Fecha de versión y versión:	05/04/2024 – 1.0
Desarrollador/a:	Alison Nicole Robles Heredia
Institución o empresa:	Escuela Hogwarts de Magia y Hechicería
Aplicación web:	Sistema de Gestión de Base de Datos de la Escuela Hogwarts de Magia y Hechicería

Historial de revisiones

Fecha	Autor/a	Descripción
04/04/2024	Alison Nicole Robles Heredia	Actualización de la versión 1.0
05/04/2024	Alison Nicole Robles Heredia	Creación de los CRUDs.

ÍNDICE DEL CONTENIDO

1. Arquitectura de la solución. 4
1.1 Dependencias. 4
1.1.1 Librerías. 4
1.2 Entornos. 4
2. Infraestructura empleada. 4
2.1 Plataforma Software. 4
3. NameSpace Alumnos. 4
3.1 Clase Alumnos. 4
3.1.1 Constructores. 4
3.1.2 Métodos. 5

1. Arquitectura de la solución.

La clase Alumnos se trata de una clase desarrollada en el proyecto Hogwarts de la Escuela Hogwarts de Magia y Hechicería.

1.1 Dependencias.

Tiene dependencia con las clases conexión y respuestas.

1.1.1 Librerías.

No se emplean librerías.

1.2 Entornos.

La aplicación está siendo ejecutada en un entorno proporcionado por XAMPP, con MySQL como tu sistema de gestión de bases de datos.

2. Infraestructura empleada.

2.1 Plataforma Software.

Se ha empleado JavaScript y PHP versión 8.2.12.

3. NameSpace Alumnos.

3.1 Clase Alumnos.

A través de la clase Alumnos heredera de la clase Conexión.

3.1.1 Constructores.

Se hace uso de un constructor heredado de la clase Conexión:

$$function\ \_construct(\{$$

$$\$listadatos = \$this \rightarrow datosConexion(\ \ );$$

$$foreach(\$listadatos\ as\ \$key => \$value)\{$$

$$\$this \rightarrow server = \$value['server'];$$

$$\$this \rightarrow user = \$value['user'];$$

$$\$this \rightarrow password = \$value['password'];$$

$$\$this \rightarrow database = \$value['database'];$$

$$\$this \rightarrow port = \$value['port'];$$

$$\$this \rightarrow conexion = new\ mysqli(\$this \rightarrow server, \$this - user, \$this \rightarrow password,$$

$$\$this \rightarrow database, \$this \rightarrow port);$$

$$if(\$this \rightarrow conexion \rightarrow connect\_errno){$$

$$echo \text{ Algo va mal con la conexión.};$$

$$die(\ \ ); \}\}$$

$$\})$$

3.1.2 Métodos.

```
/**
 * Método para listar los alumnos por páginas de 10
 * @param int $pagina El número de página a mostrar
 * @return array Un array con los datos de los alumnos
 */
public function listaAlumnos($pagina = 1)
{
  $cantidad = 10;
  $inicio = ($pagina - 1) * $cantidad;

  $query = "SELECT * FROM " .$this-> table ." LIMIT $inicio,$cantidad";
  $datos = parent::obtenerDatos($query);
  return $datos;
}

/**
 * Método para obtener los datos de un alumno específico
```

```
     * @param string $codigoAlumno El código del alumno a buscar
     * @return array Los datos del alumno encontrado
     */
public function obtenerAlumno($codigoAlumno)
{
  $query = "SELECT * FROM " . $this->table . " WHERE codigoAlumno
           = '$codigoAlumno'";
  return parent::obtenerDatos($query);
}

    /**
     * Método para agregar un nuevo alumno
     * @param string $json Los datos del alumno en formato JSON
     * @return array La respuesta de la solicitud
     */
public function post($json)
{
  $_respuestas = new respuestas;
  $datos = json_decode($json, true);

  if (
    !isset($datos['nombre']) || !isset($datos['apellidos']) || !isset($datos['codigoCasa']) ||
    !isset($datos['edad']) || !isset($datos['curso'])
  ) {
    return $_respuestas->error_400();
  } else {
    $this->nombre = $datos['nombre'];
    $this->apellidos = $datos['apellidos'];
    $this->codigoCasa = $datos['codigoCasa'];
    $this->edad = $datos['edad'];
    $this->curso = $datos['curso'];
```

```
        $resp = $this-> insertarAlumno();

        if ($resp) {
          $respuesta = $_respuestas-> response;
          $respuesta["result"] = array(
            "codigoAlumno" => $resp
          );
          return $respuesta;
        } else {
          return $_respuestas-> error_500();
        }
      }
    }

      /**
       * Método para insertar un nuevo alumno en la base de datos
       * @return int|string El código del alumno insertado o 0 si falla
       */
    public function insertarAlumno()
    {
      $query = "INSERT INTO " . $this-
              > table . " (nombre, apellidos, codigoCasa, edad, curso) VALUES
      ('" . $this-> nombre . "', '" . $this-> apellidos . "', '" . $this-
              > codigoCasa . "', '" . $this-> edad . "', '" . $this-> curso . "')";
      $resp = parent::nonQueryId($query);

      if ($resp) {
        return $resp;
      } else {
        return 0;
      }
```

```
}

    /**
     * Método para actualizar los datos de un alumno
     * @param string $json Los datos del alumno en formato JSON
     * @return array La respuesta de la solicitud
     */
public function put($json)
{
  $_respuestas = new respuestas;
  $datos = json_decode($json, true);
  if (!isset($datos['codigoAlumno'])) {
    return $_respuestas-> error_400();
  } else {
    $this-> codigoAlumno = $datos['codigoAlumno'];

    if (isset($datos['nombre'])) {
      $this-> nombre = $datos['nombre'];
    }
    if (isset($datos['apellidos'])) {
      $this-> apellidos = $datos['apellidos'];
    }
    if (isset($datos['codigoCasa'])) {
      $this-> codigoCasa = $datos['codigoCasa'];
    }
    if (isset($datos['edad'])) {
      $this-> edad = $datos['edad'];
    }
    if (isset($datos['curso'])) {
      $this-> curso = $datos['curso'];
```

```
    }

    $resp = $this->modificarAlumno();

    if ($resp) {
      $respuesta = $_respuestas->response;
      $respuesta["result"] = array(
        "codigoAlumno" => $this->codigoAlumno
      );
      return $respuesta;
    } else {
      return $_respuestas->error_500();
    }
  }
}

    /**
     * Método para modificar los datos de un alumno en la base de datos
     * @return int|string El número de filas afectadas o 0 si falla
     */
public function modificarAlumno()
{
  $query = "UPDATE " . $this->table . " SET nombre = '" . $this-
          >nombre . "', apellidos = '" . $this->apellidos . "', codigoCasa
          = '" . $this->codigoCasa . "', edad = '" . $this->edad . "', curso
          = '" . $this->curso . "' WHERE codigoAlumno = '" . $this-
          >codigoAlumno . "'";
  $resp = parent::nonQuery($query);
  print_r($resp);

  if ($resp >= 1) {
```

```
        return $resp;
    } else {
        return 0;
    }
}

    /**
     * Método para eliminar un alumno de la base de datos
     * @param string $json Los datos del alumno en formato JSON
     * @return array La respuesta de la solicitud
     */
public function delete($json)
{
    $_respuestas = new respuestas;
    $datos = json_decode($json, true);
    if (!isset($datos['codigoAlumno'])) {
        return $_respuestas-> error_400();
    } else {
        $this-> codigoAlumno = $datos['codigoAlumno'];
        $resp = $this-> eliminarAlumno();

        if ($resp) {
            $respuesta = $_respuestas-> response;
            $respuesta["result"] = array(
                "codigoAlumno" => $this-> codigoAlumno
            );
            return $respuesta;
        } else {
            return $_respuestas-> error_500();
        }
```

```
    }
}

    /**
     * Método privado para eliminar un alumno de la base de datos
     * @return int|string El número de filas afectadas o 0 si falla
     */
private function eliminarAlumno()
{
    $query = "DELETE FROM " . $this-> table . " WHERE codigoAlumno
             = '" . $this-> codigoAlumno . "'";
    $resp = parent::nonQuery($query);

    if ($resp >= 1) {
        return $resp;
    } else {
        return 0;
    }

}
```