



# Instituto Tecnológico Superior de Huauchinango

## Ingeniería en Sistemas Computacionales

### Reporte de Funcionalidades del Launcher "Radiant Realms"

### Administración de base de datos

Elaborado por (alumno):  
Luis David Trejo Fuentes

## Descripción General

El launcher "Radiant Realms" es una aplicación desarrollada en C# para la administración de un videojuego multijugador similar a League of Legends. Este launcher permite a los administradores y trabajadores gestionar datos de jugadores, añadir campeones y realizar diversas modificaciones en la base de datos. La interfaz está diseñada para ser accesible únicamente para usuarios con roles específicos (administradores y trabajadores).

## Funcionalidades Principales

### 1. Gestión de la Base de Datos

#### 1.1. Respaldo de la Base de Datos

Función: Permite crear un respaldo (backup) de la base de datos RadiantRealms.

Código Relacionado:

```
1 referencia
private void btnBackup_Click(object sender, EventArgs e)
{
    string backupFileName = txtBackupFileName.Text;
    if (string.IsNullOrEmpty(backupFileName))
    {
        MessageBox.Show("Please enter a backup file name.");
        return;
    }
    string backupQuery = $"BACKUP DATABASE [RadiantRealms] TO DISK = '{backupFileName}';

    ExecuteNonQuery(backupQuery);
    MessageBox.Show("Backup completed successfully.");
}
```

#### 1.2. Reducción del Log de la Base de Datos

Función: Permite reducir el tamaño del archivo de log de la base de datos.

Código Relacionado:

```

1 referencia
private void btnReduceLog_Click(object sender, EventArgs e)
{
    try
    {
        string reduceLogQuery = @"
        USE RadiantRealms;
        DBCC SHRINKFILE (N'RadiantRealms_log' , 1);
        BACKUP LOG RadiantRealms WITH TRUNCATE_ONLY;
        DBCC SHRINKFILE (N'RadiantRealms_log' , 1);";

        ExecuteNonQuery(reduceLogQuery);
        MessageBox.Show("Log file reduced successfully.");
    }
    catch
    {
        MessageBox.Show("No se pudo completar la operación");
    }
}

```

### 1.3. Creación de Roles

Función: Permite crear roles en la base de datos y asignarles usuarios.

Código Relacionado:

```

1 referencia
private void btnCreateRoles_Click(object sender, EventArgs e)
{
    string roleName = txtRoleName.Text;
    string userName = txtUserName.Text;
    if (string.IsNullOrEmpty(roleName) || string.IsNullOrEmpty(userName))
    {
        MessageBox.Show("Please enter a role name and a user name.");
        return;
    }
    string createRolesQuery = @"
    CREATE ROLE [{roleName}] AUTHORIZATION [dbo];
    ALTER ROLE [{roleName}] ADD MEMBER [{userName}];";

    ExecuteNonQuery(createRolesQuery);
    MessageBox.Show("Roles created successfully.");
}

```

## Seguridad y Autenticación

Función: Autenticación de usuarios para acceder al launcher, restringiendo el acceso a administradores y trabajadores. Utiliza codificación con hash y salt para almacenar y verificar contraseñas.

### Hash y Salt

Descripción: El hash y el salt se utilizan para proteger las contraseñas de los usuarios. El "salt" es un valor aleatorio que se añade a la contraseña antes de aplicar la función hash. Esto asegura que incluso

si dos usuarios tienen la misma contraseña, sus hashes serán diferentes. El hash es una función criptográfica unidireccional que convierte la contraseña (más el salt) en una cadena de caracteres única.

*Código para Generar Hash y Salt:*

```
public static class PasswordHelper
{
    public static string GenerateSalt()
    {
        byte[] salt = new byte[16];
        using (var rng = new System.Security.Cryptography.RNGCryptoServiceProvider())
        {
            rng.GetBytes(salt);
        }
        return Convert.ToBase64String(salt);
    }

    public static string HashPassword(string password, string salt)
    {
        using (var sha256 = System.Security.Cryptography.SHA256.Create())
        {
            byte[] saltedPassword = System.Text.Encoding.UTF8.GetBytes(password + salt);
            byte[] hash = sha256.ComputeHash(saltedPassword);
            return Convert.ToBase64String(hash);
        }
    }
}
```

### Código para Registrar Usuarios:

```
private void btnRegistrar_Click(object sender, EventArgs e)
{
    string username = txtUsername.Text;
    string password = txtPassword.Text;
    string email = txtEmail.Text;
    DateTime fechaNacimiento = dtpFechaNacimiento.Value;

    using (SqlConnection connection = new SqlConnection(connectionString))
    {
        connection.Open();

        // Generar un salt aleatorio
        string salt = BCrypt.Net.BCrypt.GenerateSalt(12);

        // Concatenar la contraseña con el salt y aplicar hashing
        string hashedPassword = BCrypt.Net.BCrypt.HashPassword(password + salt);

        // Obtener la fecha de registro actual
        DateTime fechaRegistro = DateTime.Now;

        // Llamar al procedimiento almacenado
        using (SqlCommand command = new SqlCommand("sp_RegistrarUsuario", connection))
        {
            command.CommandType = System.Data.CommandType.StoredProcedure;

            command.Parameters.AddWithValue("@Username", username);
            command.Parameters.AddWithValue("@PasswordHash", hashedPassword);
            command.Parameters.AddWithValue("@Salt", salt);
            command.Parameters.AddWithValue("@Email", email);
            command.Parameters.AddWithValue("@FechaNacimiento", fechaNacimiento);
            command.Parameters.AddWithValue("@FechaRegistro", fechaRegistro);

            try
            {
                command.ExecuteNonQuery();

                MessageBox.Show("Registro exitoso.");
                pnlRegistro.Visible = false;
            }
            catch (SqlException ex)
            {
                // Manejar cualquier error en la ejecución del procedimiento almacenado
                MessageBox.Show("Error al registrar usuario: " + ex.Message);
            }
        }
    }
}
```

### *Código para Autenticar Usuarios:*

```
1 referencia
private void btnIngresar_Click(object sender, EventArgs e)
{
    string username = txtUsernameLogin.Text;
    string password = txtPasswordLogin.Text;

    using (SqlConnection connection = new SqlConnection(connectionString))
    {
        connection.Open();

        // Obtener el hash y el salt almacenado en la base de datos para el usuario actual
        string getUserDataQuery = "SELECT PasswordHash, Salt FROM Usuarios WHERE Username = @Username";
        using (SqlCommand getUserDataCommand = new SqlCommand(getUserDataQuery, connection))
        {
            getUserDataCommand.Parameters.AddWithValue("@Username", username);

            using (SqlDataReader reader = getUserDataCommand.ExecuteReader())
            {
                if (reader.Read())
                {
                    string storedHashedPassword = reader["PasswordHash"].ToString();
                    string saltFromDatabase = reader["Salt"].ToString();

                    // Concatenar la contraseña proporcionada con el salt y verificar el hash
                    string hashedPasswordFromInput = (password + saltFromDatabase);

                    // Comparar el hash generado con el hash almacenado en la base de datos
                    if (BCrypt.Net.BCrypt.Verify(hashedPasswordFromInput, storedHashedPassword))
                    {
                        // La contraseña es válida
                        MessageBox.Show("Inicio de sesión exitoso.");

                        Launcher launcher = new Launcher();
                        launcher.Show();
                        this.Visible=false;
                    }
                    else
                    {
                        // La contraseña es incorrecta
                        MessageBox.Show("Inicio de sesión fallido. Verifica tu nombre de usuario y contraseña.");
                    }
                }
            }
        }
    }
}
```

### Tablas de la Base de Datos

La base de datos RadiantRealms contiene las siguientes tablas:

Amigos

Campeones

Jugadores

Partidas

Partidas\_Campeones

Usuarios

Resultados:

X


Nombre de usuario

Contraseña

Email

Fecha de nacimiento  
martes , 4 de juni

Registrar

The background of the registration screen features a vibrant, stylized illustration of various fantasy characters, including a large blue dragon-like creature, a knight in armor, and other magical beings, all set against a glowing blue and yellow circular backdrop. The title "Radidant Realms" is prominently displayed at the bottom in a stylized, gothic font.

INICIO PERFIL CAMPEONES

Luis David

Ocultar amigos

Nombre de usuario

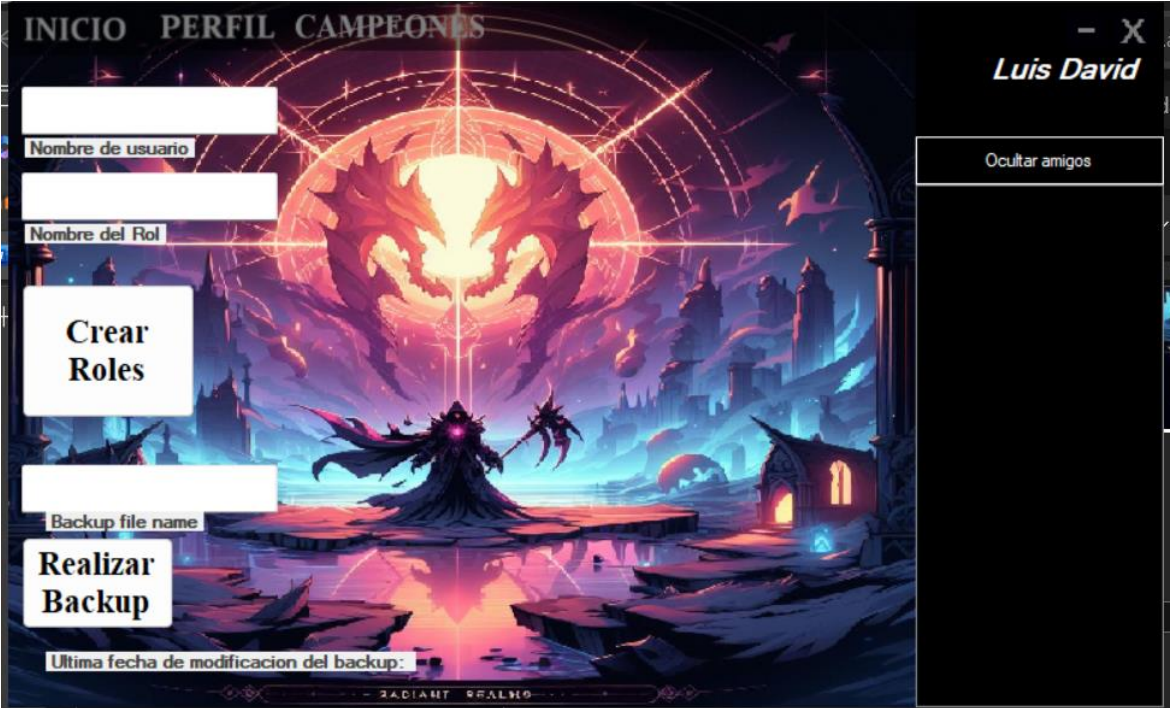
Nombre del Rol

Crear Roles

Backup file name

Realizar Backup

Ultima fecha de modificacion del backup:

The background of the profile screen is a dark, atmospheric illustration of a fantasy landscape with a large, glowing red and orange sun or moon in the center, casting a long shadow of a dragon-like creature. The scene is set in a dark, gothic-style environment with a small building and a bridge in the foreground. The title "Radidant Realms" is visible at the bottom.





## Conclusión

El launcher "Radiant Realms" proporciona una interfaz amigable y segura para que los administradores y trabajadores gestionen los datos de los jugadores y otros elementos del videojuego multijugador. Las funcionalidades incluyen respaldo y reducción de log de la base de datos, creación de roles e índices, y autenticación de usuarios mediante hash y salt para mayor seguridad.