**Manual Técnico para la Aplicación de Generación de Contraseñas**

**Tabla de Contenidos**

1. Descripción General
2. Requerimientos del Sistema
3. Estructura de la Base de Datos
4. Configuración Inicial
5. Descripción del Código
6. Elección del Algoritmo
7. Método de Hasheado: BCRYPT
8. Pruebas de Seguridad
9. Documentación del Código

**Descripción General**

La **Aplicación de Generación de Contraseñas** es una herramienta diseñada para crear contraseñas seguras y únicas, adaptadas a las necesidades de los usuarios. La aplicación ofrece dos métodos de generación:

* **Contraseña Personalizada:** Permite a los usuarios seleccionar los tipos de caracteres y la longitud de la contraseña.
* **Contraseña estilo XKCD:** Genera contraseñas basadas en combinaciones de palabras aleatorias, siguiendo el principio de simplicidad y memorabilidad del cómic XKCD.

Además, la aplicación permite almacenar las contraseñas generadas en una base de datos, asociándolas con un código QR que representa la contraseña de forma segura.

**Requerimientos del Sistema**

Para desplegar y ejecutar correctamente la Aplicación de Generación de Contraseñas, se requiere el siguiente entorno y componentes:

**Hardware**

* **Servidor Web:**
  + Procesador: Intel Core i3 o superior.
  + Memoria RAM: 4 GB mínimo.
  + Almacenamiento: 10 GB libres para la base de datos y archivos de la aplicación.

**Software**

* **Servidor Web:**
  + **PHP:** Versión 7.4 o superior.
  + **Apache:** Preferentemente en entornos XAMPP o WAMP para desarrollo local.
* **Base de Datos:**
  + **MySQL:** Versión 5.7 o superior.
* **Bibliotecas y Frameworks:**
  + **JavaScript:**
    - Biblioteca HTML5 QR Code para escaneo de códigos QR.
    - Biblioteca jsPDF para la generación y descarga de contraseñas en formato PDF.
* **Otros:**
  + **Sistema Operativo:** Compatible con Apache y PHP (Windows, Linux, macOS).
  + **Navegador Web:** Actualizado para soportar las funcionalidades de JavaScript y HTML5.

**Estructura de la Base de Datos**

La base de datos utilizada por la aplicación se denomina **Generador** y está diseñada para almacenar de manera eficiente los datos de los usuarios y las contraseñas generadas.

**Tablas Principales**

### **Tabla: usuarios**

**Descripción**: Almacena la información personal y las credenciales de los usuarios.

**Campos**:

* id: Identificador único del usuario (INT, AUTO\_INCREMENT, PRIMARY KEY).
* nombre: Nombre del usuario (VARCHAR(255)).
* edad: Edad del usuario (INT).
* fecha\_nacimiento: Fecha de nacimiento del usuario (DATE).
* genero: Género del usuario (VARCHAR(10)).
* longitud: Longitud predeterminada para las contraseñas generadas (INT, por defecto 12).
* password: Contraseña cifrada del usuario (VARCHAR(255)).
* qr\_code: Ruta o URL al código QR de autenticación de 2 factores (VARCHAR(255)).
* email: Correo electrónico único del usuario (VARCHAR(255), UNIQUE).

### **Tabla: password\_manager**

**Descripción**: Contiene las contraseñas de los usuarios, asociadas a los servicios y los usuarios correspondientes.

**Campos**:

* id: Identificador único de la contraseña (INT, AUTO\_INCREMENT, PRIMARY KEY).
* user\_id: ID del usuario asociado a la contraseña (INT, FOREIGN KEY a usuarios(id)).
* service\_name: Nombre del servicio (VARCHAR(255)) al que pertenece la contraseña (ej: Gmail, Facebook).
* username: Nombre de usuario para el servicio (VARCHAR(255)).
* password: Contraseña cifrada almacenada para ese servicio (TEXT).
* created\_at: Fecha y hora de creación de la entrada (TIMESTAMP, por defecto CURRENT\_TIMESTAMP).

### **Tabla: mayusculas**

**Descripción**: Lista de caracteres en mayúsculas disponibles para la generación de contraseñas personalizadas.

**Campos**:

* id: Identificador único del carácter (INT, AUTO\_INCREMENT, PRIMARY KEY).
* caracter: Carácter en mayúscula (CHAR(1)).

### **Tabla: minusculas**

**Descripción**: Lista de caracteres en minúsculas disponibles para la generación de contraseñas personalizadas.

**Campos**:

* id: Identificador único del carácter (INT, AUTO\_INCREMENT, PRIMARY KEY).
* caracter: Carácter en minúscula (CHAR(1)).

### **Tabla: numeros**

**Descripción**: Lista de números disponibles para la generación de contraseñas personalizadas.

**Campos**:

* id: Identificador único del carácter (INT, AUTO\_INCREMENT, PRIMARY KEY).
* caracter: Un número (CHAR(1)).

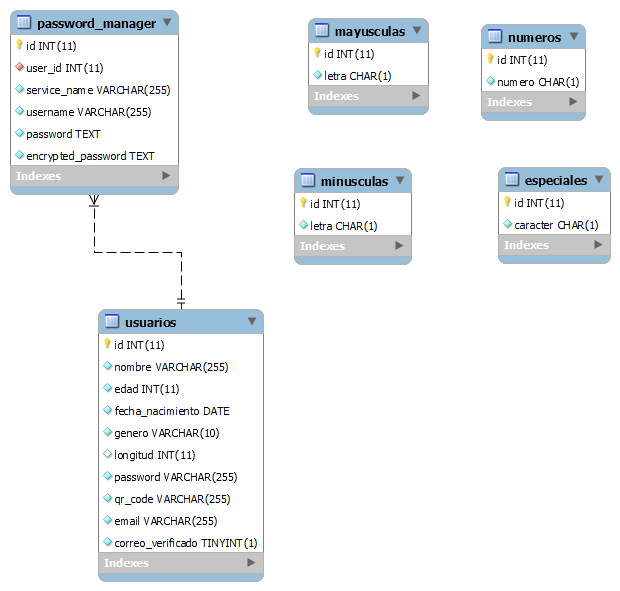
### **Tabla: especiales**

**Descripción**: Lista de caracteres especiales disponibles para la generación de contraseñas personalizadas.

**Campos**:

* id: Identificador único del carácter (INT, AUTO\_INCREMENT, PRIMARY KEY).
* caracter: Carácter especial (CHAR(1)).

**Diagrama de la Base de Datos**



**Configuración Inicial**

**1. Configuración de la Conexión a la Base de Datos**

* **Archivo de Configuración:** config.php

php

Copiar código

<?php

$servername = "localhost";

$username = "tu\_usuario";

$password = "tu\_contraseña";

$dbname = "Generador";

// Crear conexión

$conn = new mysqli($servername, $username, $password, $dbname);

// Verificar conexión

if ($conn->connect\_error) {

die("Conexión fallida: " . $conn->connect\_error);

}

?>

* **Pasos:**
  1. Abre el archivo config.php.
  2. Reemplaza tu\_usuario y tu\_contraseña con las credenciales de acceso a tu base de datos MySQL.
  3. Asegúrate de que el servidor de la base de datos esté activo.

**2. Carga de Tablas en MySQL**

* **Script SQL:** generador.sql

sql

Copiar código

CREATE DATABASE IF NOT EXISTS Generador;

USE Generador;

-- Tabla para almacenar datos del usuario y la contraseña generada

CREATE TABLE usuarios (

id INT AUTO\_INCREMENT PRIMARY KEY,

nombre VARCHAR(255) NOT NULL,

edad INT NOT NULL,

fecha\_nacimiento DATE NOT NULL,

genero VARCHAR(10) NOT NULL,

longitud INT DEFAULT 12,

password VARCHAR(255) NOT NULL,

qr\_code VARCHAR(255) NOT NULL,

email VARCHAR(255) NOT NULL UNIQUE

);

-- Tabla para letras mayúsculas

CREATE TABLE IF NOT EXISTS mayusculas (

id INT AUTO\_INCREMENT PRIMARY KEY,

letra CHAR(1) NOT NULL

);

-- Tabla para letras minúsculas

CREATE TABLE IF NOT EXISTS minusculas (

id INT AUTO\_INCREMENT PRIMARY KEY,

letra CHAR(1) NOT NULL

);

-- Tabla para números

CREATE TABLE IF NOT EXISTS numeros (

id INT AUTO\_INCREMENT PRIMARY KEY,

numero CHAR(1) NOT NULL

);

-- Tabla para caracteres especiales

CREATE TABLE IF NOT EXISTS especiales (

id INT AUTO\_INCREMENT PRIMARY KEY,

caracter CHAR(1) NOT NULL

);

CREATE TABLE password\_manager (

id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT NOT NULL,

service\_name VARCHAR(255) NOT NULL,

username VARCHAR(255) NOT NULL,

password TEXT NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES usuarios(id) ON DELETE CASCADE

);

-- Insertar datos iniciales en tablas auxiliares

INSERT INTO mayusculas (letra) VALUES

('A'), ('B'), ('C'), ('D'), ('E'), ('F'), ('G'), ('H'), ('I'), ('J'),

('K'), ('L'), ('M'), ('N'), ('O'), ('P'), ('Q'), ('R'), ('S'), ('T'),

('U'), ('V'), ('W'), ('X'), ('Y'), ('Z');

-- Letras minúsculas

INSERT INTO minusculas (letra) VALUES

('a'), ('b'), ('c'), ('d'), ('e'), ('f'), ('g'), ('h'), ('i'), ('j'),

('k'), ('l'), ('m'), ('n'), ('o'), ('p'), ('q'), ('r'), ('s'), ('t'),

('u'), ('v'), ('w'), ('x'), ('y'), ('z');

-- Números

INSERT INTO numeros (numero) VALUES ('0'), ('1'), ('2'), ('3'), ('4'), ('5'), ('6'), ('7'), ('8'), ('9');

-- Caracteres especiales

INSERT INTO especiales (caracter) VALUES ('!'), ('@'), ('#'), ('$'), ('%'), ('^'), ('&'), ('\*'), ('('), (')');

* **Pasos:**
  1. Abre tu herramienta de gestión de bases de datos (como phpMyAdmin).
  2. Crea una nueva base de datos llamada Generador.
  3. Ejecuta el script generador.sql para crear las tablas y poblar las tablas auxiliares con datos iniciales.

**3. Configuración de Permisos y CORS**

* **Permisos de Acceso a la Cámara:**
  + Asegúrate de que la aplicación web tenga permisos para acceder a la cámara del dispositivo. Esto generalmente se gestiona a través de la configuración del navegador.
* **Configuración de CORS:**
  + Para entornos de desarrollo, es esencial habilitar CORS (Cross-Origin Resource Sharing) para permitir solicitudes desde diferentes orígenes.
  + **Archivo .htaccess:**

apache

Copiar código

<IfModule mod\_headers.c>

Header set Access-Control-Allow-Origin "\*"

Header set Access-Control-Allow-Methods "GET,POST,OPTIONS"

Header set Access-Control-Allow-Headers "Content-Type"

</IfModule>

* + **Nota:** Para entornos de producción, reemplaza "\*" con los dominios específicos que tendrán acceso para mejorar la seguridad.

**Descripción del Código**

**1. PHP**

* **Archivo Principal:** index.php
  + **Funciones Principales:**
    - **Generación de Contraseñas Personalizadas:**
      * Recopila las opciones seleccionadas por el usuario (mayúsculas, minúsculas, números, caracteres especiales).
      * Genera una contraseña aleatoria basada en los criterios seleccionados.
      * Guarda la contraseña y los datos del usuario en la base de datos.
      * Genera un código QR que representa la contraseña.
    - **Manejo de Solicitudes AJAX:**
      * Recibe solicitudes asíncronas desde JavaScript.
      * Procesa la generación de contraseñas y devuelve respuestas en formato JSON.
  + **Ejemplo de Función de Generación de Contraseña:**

php

Copiar código

function generarContraseña($longitud, $usarMayusculas, $usarMinusculas, $usarNumeros, $usarEspeciales) {

$caracteres = '';

if ($usarMayusculas) {

$caracteres .= 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';

}

if ($usarMinusculas) {

$caracteres .= 'abcdefghijklmnopqrstuvwxyz';

}

if ($usarNumeros) {

$caracteres .= '0123456789';

}

if ($usarEspeciales) {

$caracteres .= '!@#$%^&\*()-\_=+[]{};:,.<>?/';

}

$password = '';

$max = strlen($caracteres) - 1;

for ($i = 0; $i < $longitud; $i++) {

$password .= $caracteres[rand(0, $max)];

}

return $password;

}

**2. JavaScript**

* **Archivo Principal:** app.js
  + **Funciones Principales:**
    - **Interacción con la Interfaz de Usuario:**
      * Maneja eventos de clic en botones como "Generar Contraseña" y "Crear Cuenta".
      * Valida los datos ingresados por el usuario antes de enviarlos al servidor.
    - **Generación y Escaneo de QR:**
      * Utiliza la biblioteca HTML5 QR Code para generar códigos QR que representan las contraseñas.
      * Implementa la funcionalidad para escanear códigos QR y recuperar contraseñas almacenadas.
    - **Solicitudes AJAX:**
      * Envía datos al servidor para la generación de contraseñas y la creación de cuentas.
      * Maneja las respuestas del servidor y actualiza la interfaz de usuario en consecuencia.
  + **Ejemplo de Solicitud AJAX:**

javascript

Copiar código

document.getElementById('generarBtn').addEventListener('click', function() {

const datos = {

nombre: document.getElementById('nombre').value,

edad: document.getElementById('edad').value,

fecha\_nacimiento: document.getElementById('fecha\_nac').value,

genero: document.getElementById('genero').value,

longitud: document.getElementById('longitud').value,

mayusculas: document.getElementById('mayusculas').checked,

minusculas: document.getElementById('minusculas').checked,

numeros: document.getElementById('numeros').checked,

especiales: document.getElementById('especiales').checked

};

fetch('generar.php', {

method: 'POST',

headers: {

'Content-Type': 'application/json'

},

body: JSON.stringify(datos)

})

.then(response => response.json())

.then(data => {

if(data.success) {

document.getElementById('password').value = data.password;

if(data.qr\_code) {

generarQRCode(data.qr\_code);

}

} else {

alert('Error al generar la contraseña.');

}

})

.catch(error => console.error('Error:', error));

});

**3. Bibliotecas Utilizadas**

* **HTML5 QR Code:**
  + **Descripción:** Biblioteca JavaScript para generar y escanear códigos QR.
  + **Integración:**
    - **Generación:** Convertir la contraseña en un código QR visible para el usuario.
    - **Escaneo:** Permitir a los usuarios recuperar contraseñas escaneando el código QR.
* **jsPDF:**
  + **Descripción:** Biblioteca JavaScript para generar archivos PDF en el cliente.
  + **Integración:**
    - Permitir a los usuarios descargar sus contraseñas en formato PDF para almacenamiento seguro.

**Elección del Algoritmo**

**Algoritmo de Generación de Contraseñas**

La seguridad de una contraseña depende en gran medida de su complejidad y aleatoriedad. Para garantizar la fortaleza de las contraseñas generadas, se ha seleccionado un algoritmo que cumple con los siguientes criterios:

1. **Aleatoriedad Criptográfica:**
   * Utiliza funciones de generación de números aleatorios seguros (random\_int en PHP) para evitar predicciones.
2. **Variedad de Caracteres:**
   * Combina diferentes tipos de caracteres (mayúsculas, minúsculas, números, especiales) para aumentar la entropía.
3. **Longitud Personalizable:**
   * Permite a los usuarios definir la longitud de la contraseña, favoreciendo longitudes recomendadas para mayor seguridad.

**Justificación de la Elección**

* **Seguridad:** El uso de funciones de generación de números aleatorios criptográficamente seguros asegura que las contraseñas sean impredecibles y resistentes a ataques de fuerza bruta.
* **Flexibilidad:** Permitir la personalización de la contraseña según las necesidades del usuario (longitud y tipos de caracteres) ofrece un equilibrio entre seguridad y usabilidad.
* **Memorabilidad:** La opción estilo XKCD facilita la creación de contraseñas que son fáciles de recordar sin sacrificar la seguridad, al combinar múltiples palabras aleatorias.

### **Método de Hasheado: BCRYPT**

**BCRYPT** es un algoritmo de hash diseñado para proteger contraseñas de forma segura. Sus principales características son:

1. **Salting Automático**: Genera un valor aleatorio único (sal) para cada contraseña, asegurando que contraseñas idénticas tengan hashes diferentes.
2. **Lentitud Controlada**: La "costante de trabajo" ajustable hace que el proceso de hash sea más lento, dificultando los ataques de fuerza bruta.
3. **Seguridad Mejorada**: Resistente a ataques de diccionario y de fuerza bruta gracias a su lentitud y el uso de sal.
4. **Compatibilidad con PHP**: PHP tiene soporte nativo para BCRYPT, con funciones como password\_hash() y password\_verify().

#### **Justificación de la Elección:**

* **Seguridad**: La combinación de sal y dificultad ajustable hace que BCRYPT sea altamente seguro.
* **Resistencia a Ataques**: Su lentitud lo hace resistente a ataques automatizados.
* **Fácil Implementación**: PHP soporta BCRYPT de forma nativa, facilitando su integración en proyectos.

**Pruebas de Seguridad**

La seguridad es una prioridad en el desarrollo de la Aplicación de Generación de Contraseñas. A continuación, se detallan las pruebas y medidas implementadas para garantizar la integridad y confidencialidad de los datos de los usuarios.

**1. Validación de Entrada**

* **Descripción:** Todas las entradas proporcionadas por el usuario son validadas y sanitizadas para prevenir inyecciones SQL y otros ataques de inyección.
* **Implementación:**
  + Uso de declaraciones preparadas (Prepared Statements) en PHP para todas las consultas a la base de datos.

php

Copiar código

$stmt = $conn->prepare("INSERT INTO usuarios (nombre, edad, fecha\_nacimiento, genero, longitud, password, qr\_code) VALUES (?, ?, ?, ?, ?, ?, ?)");

$stmt->bind\_param("sississ", $nombre, $edad, $fecha\_nacimiento, $genero, $longitud, $password, $qr\_code);

$stmt->execute();

**2. Almacenamiento Seguro de Contraseñas**

* **Descripción:** Aunque las contraseñas generadas son para uso en otras plataformas, se recomienda no almacenar contraseñas en texto plano.
* **Implementación:**
  + **Hashing:** Utilizar funciones de hashing seguras (como password\_hash en PHP) para almacenar contraseñas si fuese necesario.
  + **Encriptación del QR:** Asegurar que los códigos QR no puedan ser decodificados fácilmente sin autorización.

**3. Protección contra Ataques de Fuerza Bruta**

* **Descripción:** Implementar mecanismos para prevenir intentos de acceso no autorizados mediante ataques de fuerza bruta.
* **Implementación:**
  + **Limitación de Intentos de Inicio de Sesión:** Bloquear temporalmente la cuenta después de varios intentos fallidos.
  + **Captchas:** Integrar captchas en formularios sensibles para verificar que el usuario es humano.

**4. Uso de HTTPS**

* **Descripción:** Asegurar que todas las comunicaciones entre el cliente y el servidor estén cifradas.
* **Implementación:**
  + Configurar un certificado SSL/TLS en el servidor web para habilitar HTTPS.

**5. Pruebas de Penetración**

* **Descripción:** Realizar pruebas de penetración periódicas para identificar y remediar vulnerabilidades.
* **Implementación:**
  + Utilizar herramientas como OWASP ZAP o Burp Suite para escanear la aplicación en busca de vulnerabilidades comunes.

**6. Gestión de Sesiones Seguras**

* **Descripción:** Asegurar que las sesiones de usuario estén protegidas contra secuestros y fijación de sesiones.
* **Implementación:**
  + **Regeneración de IDs de Sesión:** Regenerar el ID de sesión después de iniciar sesión.
  + **Configuración de Cookies Seguras:** Marcar las cookies de sesión como HttpOnly y Secure.

**Documentación del Código**

Una documentación clara y detallada del código es esencial para el mantenimiento y la escalabilidad de la aplicación. A continuación, se describen las mejores prácticas implementadas:

**1. Comentarios en el Código**

* **Descripción:** Cada función y bloque de código relevante incluye comentarios descriptivos que explican su propósito y funcionamiento.
* **Ejemplo:**

php

Copiar código

/\*\*

\* Genera una contraseña aleatoria basada en los criterios proporcionados.

\*

\* @param int $longitud Longitud deseada de la contraseña.

\* @param bool $usarMayusculas Incluir letras mayúsculas.

\* @param bool $usarMinusculas Incluir letras minúsculas.

\* @param bool $usarNumeros Incluir números.

\* @param bool $usarEspeciales Incluir caracteres especiales.

\* @return string Contraseña generada.

\*/

function generarContraseña($longitud, $usarMayusculas, $usarMinusculas, $usarNumeros, $usarEspeciales) {

// Implementación de la función...

}

**2. Estructura de Directorios**

* **Descripción:** La aplicación sigue una estructura de directorios organizada para facilitar la navegación y el mantenimiento.
* **Estructura Recomendada:**

/appventas

│

├── /assets

│ ├── /css

│ └── /js

│

├── /includes

│ ├── db.php

│ ├── header.php

│ ├── footer.php

│ └── config.php

│

├── /models

│ ├── PasswordManager.php

│ └── User.php

│

├── /views

│ ├── dashboard.php

│ ├── login.php

│ ├── register.php

│ └── password\_manager.php

│

├── /controllers

│ ├── PasswordController.php

│ └── UserController.php

│

├── /routes

│ ├── add\_password.php

│ ├── delete\_password.php

│ ├── get\_passwords.php

│ ├── update\_password.php

│ └── login.php

│

└── /config

└── config.php

**3. Uso de Variables y Funciones Claras**

* **Descripción:** Nombres de variables y funciones descriptivos que reflejan su propósito, mejorando la legibilidad del código.
* **Ejemplo:**

javascript

Copiar código

function validarFormulario() {

// Validar campos del formulario

}

let longitudContraseña = document.getElementById('longitud').value;

**4. Modularización del Código**

* **Descripción:** Separar funcionalidades en módulos o archivos distintos para mejorar la mantenibilidad y reutilización del código.
* **Ejemplo:**
  + **Módulo de Generación de Contraseñas:** generar.php
  + **Módulo de Autenticación:** autenticar.php
  + **Módulo de Generación de QR:** Integrado en app.js

**5. Manejo de Errores**

* **Descripción:** Implementar manejo de errores robusto para capturar y gestionar excepciones de manera efectiva.
* **Ejemplo en PHP:**

php

Copiar código

try {

// Código que puede lanzar una excepción

} catch (Exception $e) {

error\_log($e->getMessage());

echo json\_encode(['success' => false, 'message' => 'Ocurrió un error.']);

}