**Script del Generador de Contraseñas**

**Descripción General**

Este proyecto es una aplicación completa para generar contraseñas aleatorias con diferentes niveles de seguridad. La aplicación permite al usuario personalizar las características de la contraseña y almacena un historial de contraseñas generadas en una base de datos MongoDB.

**Arquitectura**

**Stack Tecnológico**

* **Backend**: Node.js + Express
* **Frontend**: Angular
* **Base de Datos**: MongoDB Atlas

**Componentes del Sistema**

**1. Backend (Node.js + Express)**

El backend es una API RESTful que maneja la generación de contraseñas y el almacenamiento en la base de datos.

**Estructura de archivos principales:**

* server.js: Servidor Express principal
* .env: Variables de entorno (configuración de MongoDB y puerto)
* package.json: Dependencias y scripts

**Funcionalidades Principales del Backend:**

**Generación de Contraseñas**

function generatePassword(length, includeSpecialChars, includeNumbers, includeUppercase) {

const lowercase = 'abcdefghijklmnopqrstuvwxyz';

const uppercase = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';

const numbers = '0123456789';

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

let chars = lowercase;

if (includeUppercase) chars += uppercase;

if (includeNumbers) chars += numbers;

if (includeSpecialChars) chars += specialChars;

let password = '';

for (let i = 0; i < length; i++) {

const randomIndex = Math.floor(Math.random() \* chars.length);

password += chars[randomIndex];

}

return password;

}

**Modelo de datos**

const passwordSchema = new mongoose.Schema({

password: { type: String, required: true },

length: { type: Number, required: true },

includeSpecialChars: { type: Boolean, required: true },

includeNumbers: { type: Boolean, required: true },

includeUppercase: { type: Boolean, required: true },

createdAt: { type: Date, default: Date.now }

});

const Password = mongoose.model('Password', passwordSchema);

**Endpoints API**

1. **Generar Contraseña**

app.get('/generar-password', async (req, res) => {

try {

const length = parseInt(req.query.length) || 12;

const includeSpecialChars = req.query.special === 'true';

const includeNumbers = req.query.numbers === 'true';

const includeUppercase = req.query.uppercase === 'true';

const password = generatePassword(length, includeSpecialChars, includeNumbers, includeUppercase);

const newPassword = new Password({

password,

length,

includeSpecialChars,

includeNumbers,

includeUppercase

});

await newPassword.save();

res.json({ password, saved: true });

} catch (error) {

console.error('Error generating password:', error);

res.status(500).json({ error: 'Error generating password' });

}

});

1. **Obtener Historial de Contraseñas**

app.get('/passwords', async (req, res) => {

try {

const passwords = await Password.find().sort({ createdAt: -1 });

res.json(passwords);

} catch (error) {

console.error('Error fetching passwords:', error);

res.status(500).json({ error: 'Error fetching passwords' });

}

});

**2. Frontend (Angular)**

**Estructura de componentes principales:**

* PasswordService: Servicio para comunicación con la API del backend
* PasswordGeneratorComponent: Componente principal para la UI de generación

**PasswordService**

@Injectable({ providedIn: 'root' })

export class PasswordService {

private apiUrl = 'http://localhost:3000';

constructor(private http: HttpClient) { }

generatePassword(options: PasswordOptions): Observable<PasswordResponse> {

const { length, special, numbers, uppercase } = options;

return this.http.get<PasswordResponse>(`${this.apiUrl}/generar-password`, {

params: {

length: length.toString(),

special: special.toString(),

numbers: numbers.toString(),

uppercase: uppercase.toString()

}

});

}

getSavedPasswords(): Observable<SavedPassword[]> {

return this.http.get<SavedPassword[]>(`${this.apiUrl}/passwords`);

}

}

**PasswordGeneratorComponent**

export class PasswordGeneratorComponent {

passwordOptions: PasswordOptions = {

length: 12,

special: true,

numbers: true,

uppercase: true

};

generatedPassword: string = '';

copied: boolean = false;

savedPasswords: SavedPassword[] = [];

loading: boolean = false;

error: string | null = null;

constructor(private passwordService: PasswordService) {

this.loadSavedPasswords();

}

generatePassword(): void {

this.loading = true;

this.error = null;

this.copied = false;

this.passwordService.generatePassword(this.passwordOptions).subscribe({

next: (response) => {

this.generatedPassword = response.password;

this.loading = false;

this.loadSavedPasswords();

},

error: (error) => {

this.error = 'Error generating password. Please try again.';

this.loading = false;

console.error('Error generating password:', error);

}

});

}

loadSavedPasswords(): void {

this.passwordService.getSavedPasswords().subscribe({

next: (passwords) => {

this.savedPasswords = passwords;

},

error: (error) => {

console.error('Error loading saved passwords:', error);

}

});

}

copyToClipboard(): void {

if (!this.generatedPassword) return;

navigator.clipboard.writeText(this.generatedPassword)

.then(() => {

this.copied = true;

setTimeout(() => this.copied = false, 2000);

})

.catch(err => {

console.error('Error copying to clipboard:', err);

});

}

formatDate(dateString: string): string {

return new Date(dateString).toLocaleString();

}

}

**3. Interfaz de Usuario**

La interfaz permite:

* Configurar opciones (longitud, caracteres especiales, números, mayúsculas)
* Generar contraseñas
* Copiar al portapapeles
* Ver historial con detalles (longitud, opciones, fecha)

**Flujo de Trabajo**

1. El usuario configura opciones.
2. Se genera la contraseña vía API.
3. Se guarda en MongoDB.
4. Se muestra en la UI y se actualiza historial.

**Seguridad**

* Contraseñas almacenadas como texto plano (no recomendado para producción)
* Permite generar contraseñas seguras con distintas combinaciones

**Ejecución del Proyecto**

**Requisitos**

* Node.js (v16 o superior)
* Angular CLI
* Cuenta MongoDB Atlas

**Iniciar Backend**

cd backend

npm install

npm run dev

**Iniciar Frontend**

cd frontend

npm install

ng serve