Centro De Ciencias Básicas

Ingeniería en Sistemas Computacionales

Seguridad en Sistemas

Práctica 1 – Primeros Pasos Cifrando

8 A

Profesor:

Mtro. Arturo Ocampo Silva

Ricardo Ramírez Torres, 349230

Índice

Introducción

Es bien sabido que el conocimiento es poder, a tal grado que la frase se ha convertido en un axioma en el argot popular. Esta distinción no es vano, el conocimiento nos permite tomar mejores decisiones, más informadas, que usualmente llevan a mejores resultados. En un mundo utópico, se buscaría que el conocimiento estuviera al alcance de todos, para que así la sociedad en general tomara un mejor rumbo. Tristemente, no vivimos en un mundo así; las intenciones de las personas no siempre concuerdan con lo que es mejor para los demás, y esta discrepancia ha resultado en un sinfín de amenazas de las que nos debemos cuidar.

Esta necesidad de divulgar selectivamente la información, engendrada principalmente en ámbitos políticos y militares, resultó en la invención de la encriptación: el arte de ocultar el significado de un mensaje para que solo el destinatario legítimo pueda comprenderlo. Aunque el uso de cifras rudimentarias se remonta a la antigüedad, hay una figura que destaca por la vigencia e impacto de sus contribuciones: Al Kindi. Mas allá de desarrollar nuevas técnicas de encriptación, Al-Kindi innovó en la desencriptación, publicando" Manuscrito sobre el descifrado de mensajes criptográficos" donde introdujo técnicas como el análisis de frecuencias, una de las primeras implementaciones prácticas de los métodos estadísticos. Asimismo, clasifica los tipos de encriptado en dos grandes grupos: por transposición y por sustitución.

Justamente ha sido Al Kindi quien ha demostrado la inseguridad de los métodos de cifrado utilizados en esta práctica, César y Atbash. El primero funciona desplazando los caracteres de la cadena inicial a través de un alfabeto acordado por un número dado de lugares. Por ejemplo, la cadena "HOLA" sería transformada en "JQNC" en una encriptación César con desplazamiento 2 y el alfabeto de las letras mayúsculas. El segundo, Atbash, funciona "espejeando" los valores según su posición en el alfabeto proporcionado: es decir, en un alfabeto con 15 caracteres el tercer carácter sería representado por el treceavo, el primero por el último, el séptimo por el noveno y así. Por ende, la misma cadena "HOLA" sería representada como "SLOZ".

Con esto mente, desarrollaremos un aplicación web que permita a los usuarios cifrar y descifrar mensajes de texto utilizando ambos métodos de cifrado.

Objetivo

El objetivo de esta práctica es desarrollar una aplicación web de cifrado tipo César y ATBASH, que permita a los usuarios encriptar y desencriptar mensajes de forma intuitiva.

Desarrollo

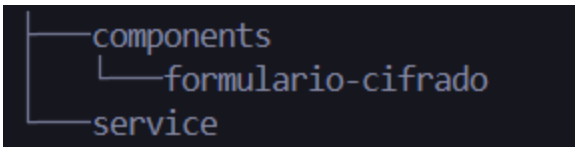
El desarrollo del proyecto será comentado de forma segura, como lo indican las instrucciones del proyecto.

Sección 0: Evaluación de requisitos

Al ser una aplicación web, Angular será utilizado como framework de desarrollo por varios motivos, entre los que destacan la familiaridad con la que contamos, su seguridad y flexibilidad.

0.1: Estructura

Nuestra aplicación web contará con los siguientes dos módulos, además de aquellos generados por defecto con cualquier nuevo proyecto de Angular: un servicio encargado del proceso de encriptación y desencriptación, y un componente que reciba el input del usuario.



Sección 1: Controlador Formulario

Este módulo se encarga de la lógica de envío y recepción de datos entre el usuario y el servicio.

1.1: Constructor

En el constructor declaramos que utilizaremos el ReactiveFormsModule de Angular, y le asignamos los valores por defecto, escogiendo una encriptación tipo César con el alfabeto alfanumérico. También asigna distintas restricciones al formulario, haciéndolo obligatorio y limitando la entrada de la sección de desplazamiento a solamente enteros.

```
//sección 1.1
  constructor(private fb: FormBuilder, private service: CifrarService) {
    this.form = this.fb.group({
      modulo: ['CESAR', Validators.required],
      operacion: ['CIFRAR', Validators.required],
      alfabeto:
['abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789',
Validators.required],
      desplazamiento: [3, [Validators.min(1) , Validators.pattern('^[0-9]*$'
]],
      mensaje: ['', Validators.required]
    });
  }
```

1.2: Ejecutar

Este modulo es el encargado de mandar los parámetros necesarios al servicio para realizar su encriptación o desencriptación, según lo elija el usuario. Asimismo, se asegura que se solicite la operación correcta con el tipo de cifrado correcto.

```
//sección 1.2
  ejecutar() {
    const { modulo, operacion, alfabeto, desplazamiento, mensaje } =
this.form.value;
    const esCifrado = operacion === 'CIFRAR';

    if (modulo === 'CESAR') {
      this.resultado = this.service.procesar1(mensaje, alfabeto, desplazamiento,
esCifrado);
    } else {
      this.resultado = this.service.procesar2(mensaje, alfabeto);
    }
  }
```

Sección 2: Vista Formulario

Este módulo se encarga de recibir la entrada del usuario y de mostrar los resultados en pantalla.

2.1: Parámetros de Cifrado

En esta sección se contienen todos los parámetros de cifrado como:

- Operación: Encriptar o Desencriptar.
- Tipo de Cifrado: AtBash o César.
- Alfabeto: Caracteres válidos.
- Desplazamiento (solamente César): la cantidad de lugares recorridos para encriptar o desencriptar.

```
<form [formGroup]="form" (ngSubmit)="ejecutar()" class="card p-4 shadow-sm">
    <div class="row">
      <div class="col-md-6 mb-3">
        <label class="form-label font-weight-bold">Seleccionar Tipo de
Cifrado:</label>
        <select formControlName="modulo" class="form-select">
          <option value="CESAR">Cifrado César</option>
          <option value="ATBASH">Cifrado Atbash</option>
        </select>
      </div>

      <div class="col-md-6 mb-3">
        <label class="form-label font-weight-bold">Acción:</label>
        <div class="btn-group w-100">
          <input type="radio" class="btn-check" formControlName="operacion"
value="CIFRAR" id="cifrar">
          <label class="btn btn-outline-success" for="cifrar">Cifrar</label>

          <input type="radio" class="btn-check" formControlName="operacion"
value="DESCIFRAR" id="descifrar">
          <label class="btn btn-outline-danger" for="descifrar">Descifrar</label>
        </div>
      </div>
    </div>

    <div class="mb-3">
      <label class="form-label">Conjunto de Caracteres (Base ASCII):</label>
      <input type="text" formControlName="alfabeto" class="form-control text-
monospace">
      <small class="text-muted">Solo se procesarán los caracteres incluidos en
esta lista.</small>
    </div>
@if (form.get('modulo')?.value === 'CESAR') {
    <div class="mb-3">
```

```
      <label class="form-label">Desplazamiento (Key):</label>
      <input type="number" formControlName="desplazamiento" class="form-control"
step="1"
        onkeypress="return event.charCode >= 48 && event.charCode <= 57"
placeholder="Ej: 3">
    </div>
    }
```

2.2: Entrada y envío

Esta sección contiene el textbox para que el usuario teclee la cadena a convertir, y el botón que manda a llamar a la función ejecutar, definida en la sección 1. Nótese que solamente se puede n enviar datos si se cumple con todos los parámetros de cifrado.

```
<div class="mb-3">
      <label class="form-label">Entrada de Texto:</label>
      <textarea formControlName="mensaje" class="form-control" rows="3"
placeholder="Escribe aquí..."></textarea>
    </div>

    <button type="submit" [disabled]="form.invalid" class="btn btn-dark btn-lg w-
100">
      PROCESAR {{ form.get('modulo')?.value }}
    </button>
  </form>
```

2.3 Resultado

Esta sección, contenida en un if, muestra los resultados de la encriptación al usuario.

```
@if (resultado) {
  <div class="mt-4 p-4 bg-white border rounded shadow-sm">
    <h4>Resultado Final:</h4>
    <p class="display-6 text-break">{{ resultado }}</p>
  </div>
  }
</div>
```

Sección 3: Servicio de Encriptación

Este módulo recibe parámetros, como la cadena a encriptar, el alfabeto y tipo de encriptación, y realiza las operaciones correspondientes.

3.1: Cesar

Esta sección utiliza las funciones de encriptar y desencriptar a través de César. Primero recibe el alfabeto y lo cuantifica para conocer el total de caracteres. Después, ajusta el desplazamiento según la acción, encriptar o desencriptar. Una vez hecho esto, separa el String de caracteres en un vector para ser recorrido según el desplazamiento y concatena estos caracteres en una nueva cadena, que devuelve como resultado.

```
procesar1(mensaje: string, alfabeto: string, desplazamiento: number, cifrar:
boolean): string {
    const L = alfabeto.length;
    const shift = cifrar ? desplazamiento : -desplazamiento;

    return mensaje.split('').map(char => {
      const index = alfabeto.indexOf(char);
      if (index === -1) return char;

      let nuevoIndex = (index + shift) % L;
      if (nuevoIndex < 0) nuevoIndex += L;

      return alfabeto[nuevoIndex];
    }).join('');
  }
```

3.2 AtBash

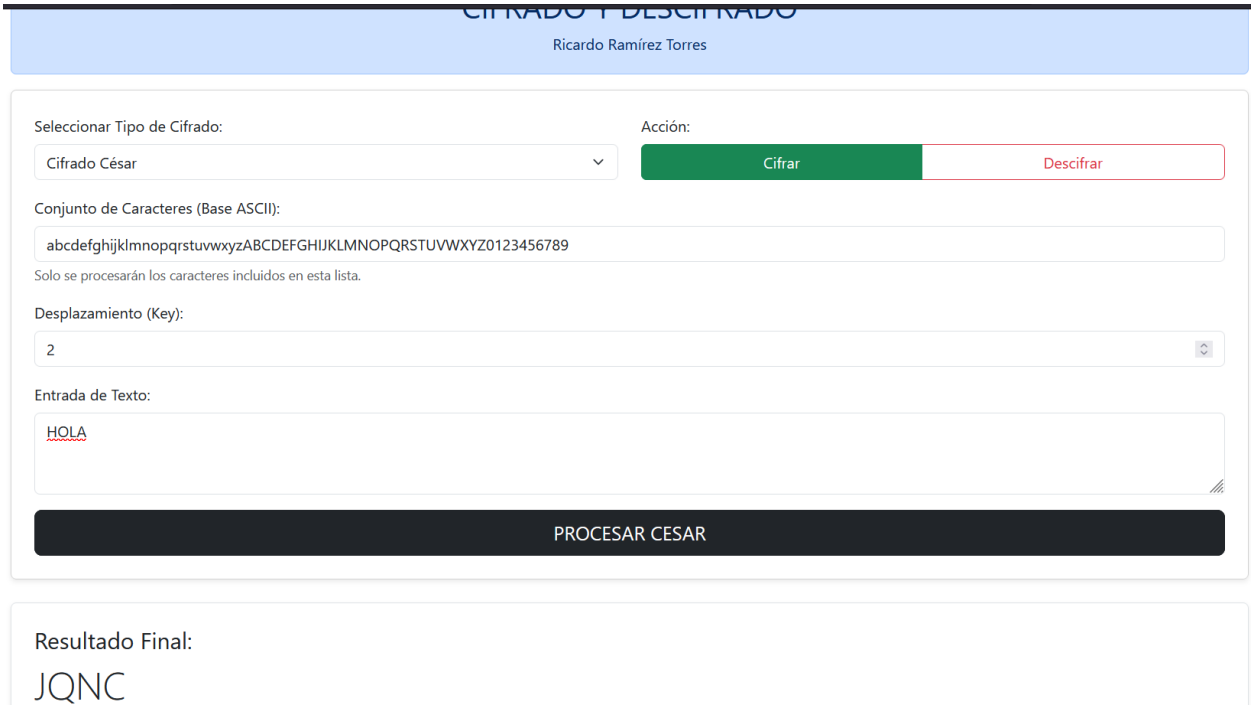
De forma similar, esta sección se encarga de cifrar y descifrar en base a AtBash, que son operaciones opuestas. Primero se obtiene también el largo del alfabeto y se separa en un vector, donde se buscan los caracteres y se ajusta el índice de su opuesto a través de una resta. Finalmente, se concatena el resultado.

```
procesar2(mensaje: string, alfabeto: string): string {
    const L = alfabeto.length;
    return mensaje.split('').map(char => {
      const index = alfabeto.indexOf(char);
      if (index === -1) return char;
      return alfabeto[(L - 1) - index];
    }).join('');
  }
```

Sección 4: Estilos

Los estilos fueron implementados a través de una instalación local de Bootstrap, previniendo la suplantación del enlace de su API.

Resultados



Enlace: https://practica1seguridadrrt.netlify.app/

Repositorio: https://github.com/ricardorzt/cifrados

Conclusión

Creo que el valor de esta actividad recae más en su desarrollo que en el producto final. La complejidad del código no fue exagerada, habiendo encontrado un buen lugar en otros cursos, como aquel de programación Web. No, el valor de esta práctica fue adentrarnos un poco a las prácticas de desarrollo de aplicaciones seguras, forzándonos a hacer uso de la documentación segura, y a limitar los comentarios en nuestro código.

Esto no quiere decir que los contenidos de la práctica no fueron de provecho, pues nos ayudaron a entender un poco más de la historia del cifrado y su relevancia histórica. Tuvimos que entender a profundidad ambos algoritmos para poder traducirlos a TypeScript. Es así que creo que la practica logró adentrarnos al tema de la seguridad por dos frentes: aquel de la encriptación y del desarrollo; por lo que la considero un ejercicio exitoso.

Bibliografía

- https://v17.angular.io/docs
- https://coehuman.uodiyala.edu.iq/uploads/Coehuman%20library%20pdf/HIS%20library%20%D9%83%D8%AA%D8%A8%20%D8%AA%D8%A7%D8%B1%D9%8A%D8%AE/%D9%83%D8%AA%D8%A8%20%D8%B3%D9%85%D8%A7%D9%87%D8%B1/Al-Kindi.pdf
- https://www.cs.ox.ac.uk/stephen.drape/materials/secret.pdf

- https://www.telsy.com/en/al-kindi-the-father-of-cryptanalysis/