Informe de Laboratorio 09

Tema: Angular

Nota

Integrantes	Escuela	Asignatura
Miguel Angel Alvarez Choque 20230477 Rodrigo Alexander Fernández Huarca 20230465 Eduardo Joel Cuno Salazar 20231497 Jose Maria Ticona Saure 20233482	Escuela Profesional de Ingeniería de Sistemas	Programación Web II Semestre: I

Laboratorio	Tema	Duración
09	Angular	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2024 - A	Del 03 de julio 2024	Al 05 de julio 2024

RESULTADOS Y PRUEBAS

1. EJERCICIOS RESUELTOS:

Ejercicio propuesto:

- *Se desea crear un proyecto con Angular que implemente el juego del ahorcado. Se tendrá un arreglo con posibles palabras a adivinar por parte del usuario. La interfaz la dejamos a gusto de ustedes los programadores.*

1.1. Parte 1:

- En esta parte se instaló Angular CLI se creo un nuevo proyecto en Angular, se corrió la aplicación y se creó los componentes necesarios.

1.1.1. evidencias:

```
npm install -g @angular/cli
ng new ahorcado
cd ahorcado
ng serve
ng generate component hangman
```

1.2. Parte 2:

- En esta segunda parte se trabajo en el archivo *src/app/hangman/hangman.component.ts* para poder definir las posibles palabras y los estados del juego.

1.3. Evidencias:

```
D:\UNSA\Repositorios\Pweb2\pw2_grupo\Pweb-6\Laboratorio_09 • Contiene elementos resaltados
@Component({
  selector: 'app-hangman',
  templateUrl: './hangman.component.html',
  styleUrls: ['./hangman.component.css']
})
export class HangmanComponent implements OnInit {
  palabras: string[] = ['palabraPrueba', 'mensaje', 'unsa', 'programa', 'developer'];
  palabraActual: string = '';
  letrasAdivinadas: string[] = [];
  intentosRestantes: number = 7;
  imagenAhorcado: string = '';

  constructor() { }

  tabnine: test | explain | document | ask
  ngOnInit(): void {
    this.nuevaPalabra();
  }

  tabnine: test | explain | document | ask
  nuevaPalabra() {
    this.palabraActual = this.palabras[Math.floor(Math.random() * this.palabras.length)];
    this.letrasAdivinadas = Array(this.palabraActual.length).fill('_');
    this.intentosRestantes = 7;
    this.actualizarImagenAhorcado();
  }
```

```
  adivinar(letra: string) {
    if (this.palabraActual.includes(letra)) {
      for (let i = 0; i < this.palabraActual.length; i++) {
        if (this.palabraActual[i] === letra) {
          this.letrasAdivinadas[i] = letra;
        }
      }
    } else {
      this.intentosRestantes--;
      this.actualizarImagenAhorcado();
    }
  }

  tabnine: test | explain | document | ask
  actualizarImagenAhorcado() {
    this.imagenAhorcado = `assets/images/${7 - this.intentosRestantes}.svg`;
  }
}
```

1.4. Parte 3:

- En esta parte se realizó la creación de la interfaz del usuario.
- Se completo el HTML en *hangman.component.html* y el CSS en *hangman.component.css*.

1.4.1. Evidencias:

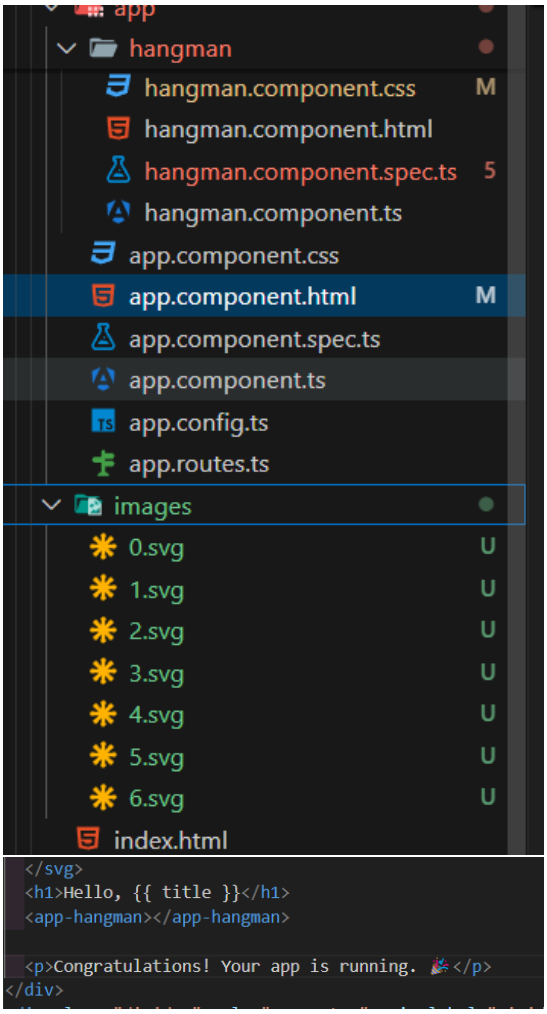
```html
<p>hangman works!</p>
<div class="hangman-container">
    <h1>Hangman Game</h1>
    <img [src]="imagenAhorcado" alt="Ahorcado">
    <div class="palabra">
      <span *ngFor="let letra of letrasAdivinadas">{{ letra }}</span>
    </div>
    <div class="letras">
      <button *ngFor="let letra of 'abcdefghijklmnopqrstuvwxyz'" (click)="adivinar(letra)">
        {{ letra }}
      </button>
    </div>
    <button (click)="nuevaPalabra()">Nueva Palabra</button>
</div>
```

```css
.hangman-container {
    text-align: center;
}

.palabra {
    font-size: 2rem;
    margin: 20px;
}

.letras {
    display: flex;
    flex-wrap: wrap;
    justify-content: center;
    margin-top: 20px;
}

.letras button {
    margin: 5px;
    padding: 10px;
    font-size: 1.2rem;
}
```

1.5. Parte 4:

- En esta parte se añadió las imagenes necesarias para el juego. Tambien se añadio el componente a la aplicación .

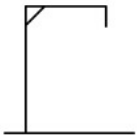
1.5.1. Evidencias:



1.6. Resutados:

- El resultado del juego es el sigueinte:

JUEGO DEL AHORCADO



Palabra a adivinar: _ _ _ _ _ _ _ _ _ _

Intentos restantes: 6

CONCLUSIONES

- Aprender Angular te equipa con las habilidades necesarias para desarrollar aplicaciones web modernas y eficientes. Su enfoque en la simplicidad, la rapidez de desarrollo y la seguridad lo convierte en una excelente elección tanto para principiantes como para desarrolladores experimentados.
- En este laboratorio profundizamos en los aspectos basicos y escenciales de angular para poder usarlo en proyectos futuros.

METODOLOGÍA DE TRABAJO

- Para el presente trabajo se repartió cada parte del trabajo a un integrante del equipo. Los roles principales fueron:
  - Desarrollo de las actividades y ejercicios del laboratorio:
    - José Maria - Parte 1
    - Rodrigo - parte 2
    - Eduardo - parte 3
    - Miguel - parte 4

REFERENCIAS Y BIBLIOGRAFÍA

Para el trabajo se ocuparon fuentes básicas para entender el uso de Django. Se procede a compartir las fuentes bibliograficas:

- `https://docs.djangoproject.com/es/3.2/`
- `https://docs.djangoproject.com/es/3.2/ref/models/fields/#field-types`
- `https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Tutorial_local_library_website`

1.7. URL'S del repositorio:

- URL del Repositorio GitHub donde se elaboró el trabajo del laboratorio.
- `https://github.com/ELGRANn/Pweb-6.git`
- URL personal de cada integrante del grupo.
  - Miguel Angel Alvarez Choque:
  - `https://github.com/miguelnodjan/pw2_24a.git`
  - Eduardo Joel Cuno Salazar:
  - `https://github.com/ELGRANn/pw2-24a.git`
  - Rodrigo Alexander Fernández Huarca:
  - `https://github.com/RdrigoFH/pw2-24a.git`
  - Jose Maria Ticona Saure:
  - `https://github.com/joseticonasaure/pw2-24a.git`

1.8. Estructura de laboratorio 7

```
|------Laboratorio_09
  |------ahorcado
      |------public
      |------src
             |------app
             |        |------hangman
             |------images
```

2. Rúbricas

2.1. Entregable Informe

Tabla 2: Tipo de Informe

Informe		
Latex	El informe está en formato PDF desde Latex, con un formato limpio (buena presentación) y fácil de leer.	**Nota**
Observaciones	Respetar la estructura de organización para la ubicación de los entregables. Por cada observación dentro del informe se le descontará puntos. Se debe incluir el código fuente latex del informe	

2.2. Rúbrica para el contenido del Informe y demostración

- El alumno debe marcar o dejar en blanco en celdas de la columna **Checklist** si cumplio con el ítem correspondiente.
- Si un alumno supera la fecha de entrega, su calificación será sobre la nota mínima aprobada, siempre y cuando cumpla con todos lo items.
- El alumno debe autocalificarse en la columna **Estudiante** de acuerdo a la siguiente tabla:

Tabla 3: Niveles de desempeño

	Nivel			
Puntos	Insatisfactorio 25 %	En Proceso 50 %	Satisfactorio 75 %	Sobresaliente 100 %
2.0	0.5	1.0	1.5	2.0
4.0	1.0	2.0	3.0	4.0

Tabla 4: Rúbrica para contenido del Informe y demostración

Contenido y demostración		Puntos	Checklist	Estudiante	Profesor
1. GitHub	Hay enlace URL activo del directorio para el laboratorio hacia su repositorio GitHub con código fuente terminado y fácil de revisar.	2	X	2	
2. Commits	Hay capturas de pantalla de los commits más importantes con sus explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4	X	3	
3. Código fuente	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2	X	2	
4. Ejecución	Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente.	2	X	1	
5. Pregunta	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación).	2	X	2	
6. Fechas	Las fechas de modificación del código fuente estan dentro de los plazos de fecha de entrega establecidos.	2	X	2	
7. Ortografía	El documento no muestra errores ortográficos.	2	X	1.5	
8. Madurez	El Informe muestra de manera general una evolución de la madurez del código fuente, explicaciones puntuales pero precisas y un acabado impecable. (El profesor puede preguntar para refrendar calificación).	4	x	3	
Total		20		16.5	