Informe de Laboratorio 09

Tema: Angular

Nota

Estudiante	Escuela	Asignatura
Alexandra Raquel Quispe Paul Andree Cari Jeferson Joao Basurco	Escuela Profesional de Ingeniería de Sistemas	Programación Web II Semestre: I Código: 20231001

Laboratorio	Tema	Duración
09	Angular	12 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2024 - A	Del 18 Junio 2024	Al 22 Junio 2023

1. Tarea

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.

```
npm install -g @angular/cli
```

Alternativamente, se puede utilizar un servidor web como Apache con Xampp. Luego, se debe descargar el archivo `data.json` y colocarlo en el directorio raíz del servidor web.

Para cada problema propuesto, se debe implementar un programa en Ajax y una página que realice las siguientes tareas:

2. Equipos, materiales y temas utilizados

- **Sistema Operativo:** Ubuntu GNU/Linux 23.04 Lunar Lobster 64 bits, Kernel 6.2.
- **Editor de texto:** VIM 9.0.
- **Entorno de desarrollo:** OpenJDK 64-Bits 17.0.7.
- **Control de versiones:** Git 2.39.2.
- **Repositorio:** Cuenta en GitHub con el correo institucional.

- **Tecnologías utilizadas:**
  - **Ajax:** Para la implementación de solicitudes asíncronas y la manipulación dinámica de datos.
  - **Google Charts:** Para la visualización gráfica de datos, incluyendo gráficos de líneas y gráficos comparativos.
  - **Python:** Utilizado para lanzar un servidor web local para pruebas.
  - **Servidor web:** SimpleHTTPServer (Python 2) o http.server (Python 3) para el desarrollo y prueba de aplicaciones web.

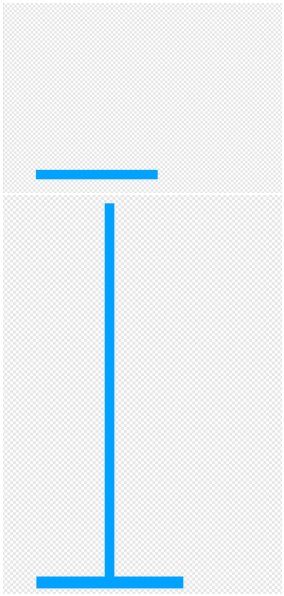
3. URL de Repositorio GitHub

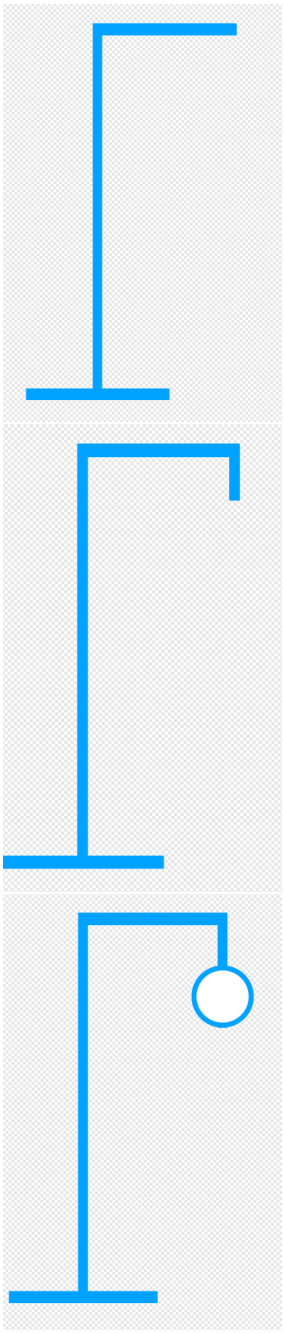
- URL del Repositorio GitHub para clonar o recuperar.
- `https://github.com/JefersonPWeb2/Pweb2-Lab09.git`
- Videos
- `https://docs.google.com/document/d/13EKPi-5QbAI2lYFi6uTgnXr7N8dtd6rU0wayuecb0CE/edit`

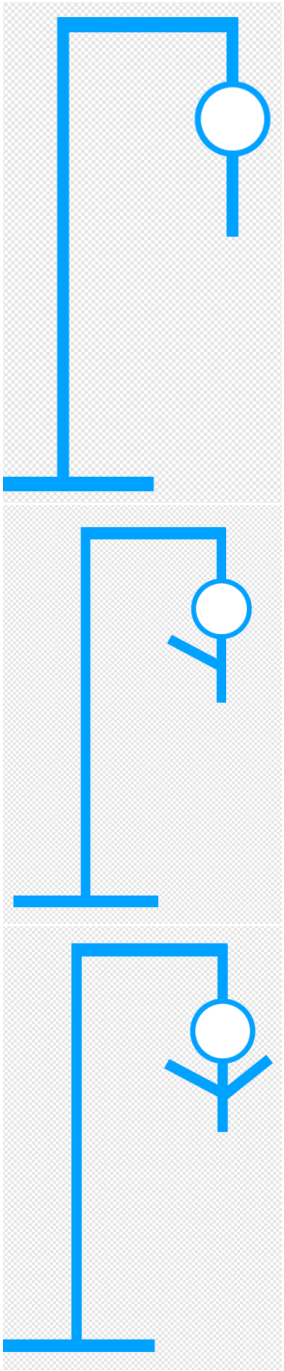
4. Actividades

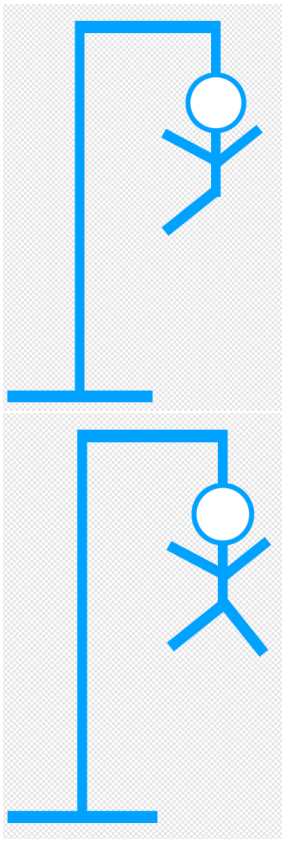
4.1. Actividad 1

- Creamos el directorio assets donde estaran las imagenes del ahorcado.









4.2. Actividad 2

- Creación de un alfabeto, método adivinar, mensaje final y reinicio del juego

```
import { Component } from '@angular/core';
import { CommonModule } from '@angular/common';
import { PalabrasService } from '../servicios/palabras.service';

@Component({
  selector: 'app-game-board',
  templateUrl: './game-board.component.html',
  styleUrls: ['./game-board.component.css'],
  standalone: true,
  imports: [CommonModule],
})
export class GameBoardComponent {
  alphabet: string[] = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'.split('');
  letrasAdivinadas: Set<string> = new Set();

  constructor(public palabrasService: PalabrasService) {}

  adivinar(letter: string) {
    this.letrasAdivinadas.add(letter);
    this.palabrasService.adivinar(letter.toLowerCase());
  }
```

```
  getEndMessage(): string {
    if (this.palabrasService.intentos <= 0) {
      return 'Perdiste! La palabra era: ' + this.palabrasService.palabraActual;
    } else {
      return 'Felicidades! Ganaste';
    }
  }

  reiniciarJuego() {
    this.letrasAdivinadas.clear();
    this.palabrasService.nuevaPalabra();
  }
}
```

4.3. Actividad 3

- Configurando la carpeta assets para incluir la carpeta hangman

4.4. Actividad 4

- Modificando el html de game-board para que muestre botones del alfabeto y mensajes de juego

```
<div class="game-board">
  <div class="word">
    <span *ngFor="let letter of palabrasService.palabraOculta">{{ letter }}</span>
  </div>
  <div class="keyboard">
    <button *ngFor="let letter of alphabet" (click)="adivinar(letter)"
        [disabled]="letrasAdivinadas.has(letter)">
      {{ letter }}
    </button>
  </div>
  <div class="status">
    <p>Intentos restantes: {{ palabrasService.intentos }}</p>
    <p *ngIf="palabrasService.juegoTerminado()">{{ getEndMessage() }}</p>
    <button *ngIf="palabrasService.juegoTerminado()" (click)="reiniciarJuego()">Reiniciar
        Juego</button>
  </div>
</div>
```

4.5. Actividad 5

- Agregando css al componente

```css
.game-board {
  text-align: center;
}

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

.keyboard button {
  margin: 5px;
  font-size: 1rem;
  padding: 10px;
}

.status {
  margin-top: 20px;
}
```

4.6. Actividad 6

- Creación de constructor y metodo getImagenURL para retornar una ruta

```
import { Component, Input } from '@angular/core';
import { CommonModule } from '@angular/common';
import { PalabrasService } from '../servicios/palabras.service';

@Component({
  selector: 'app-hangman-figure',
  templateUrl: './hangman-figure.component.html',
  styleUrls: ['./hangman-figure.component.css'],
  standalone: true,
  imports: [CommonModule],
})
export class HangmanFigureComponent {
  @Input() intentos: number = 0;

  constructor(public palabrasService: PalabrasService) {}

  getImageUrl(): string {
    const imageNumber = 9 - this.palabrasService.intentos;
    return `assets/hangman/${imageNumber}.png`;
  }
}
```

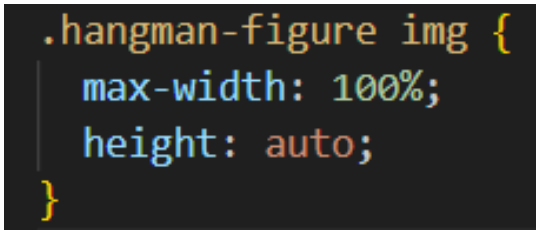
4.7. Actividad 7

- Mostrando imagen del ahorcado en la interfaz

```
<div class="hangman-figure">
  <img [src]="getImageUrl()" alt="Hangman Figure">
</div>
```

4.8. Actividad 8

- Agregando css al componente



4.9. Actividad 9

- Cambiando número de intentos

```
import { Injectable } from '@angular/core';
import { Subject } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class PalabrasService {

  private palabras: string[] = [
    'alexandra',
    'paul',
    'joao',
    'esternocleidomastoideo',
  ];

  palabraActual: string = '';
  palabraOculta: string[] = []; //es para mostrar las letras si se adivinan :D
  intentos: number = 9;

  intentosCambio = new Subject<number>();

  constructor() {
    this.nuevaPalabra();
  }

  getPalabras(): string[] {
    return this.palabras;
  }
```

```
  getPalabraAleatoria(): string {
    return this.palabras[Math.floor(Math.random() * this.palabras.length)];
  }

  nuevaPalabra() {
    this.palabraActual = this.palabras[Math.floor(Math.random() * this.palabras.length)];
    this.palabraOculta = Array(this.palabraActual.length).fill('_');
    this.intentos = 9;
    this.intentosCambio.next(this.intentos);
  }

  adivinar(letra: string) {
    let acierto = false;
    for (let i = 0; i < this.palabraActual.length; i++) {
      if (this.palabraActual[i] === letra) {
        this.palabraOculta[i] = letra;
        acierto = true;
      }
    }
    if (!acierto) {
      this.intentos--;
      this.intentosCambio.next(this.intentos);
    }
  }


  juegoTerminado(): boolean {
    return this.intentos <= 0 || this.palabraOculta.join('') === this.palabraActual;
  }
}
```

4.10. Actividad 10

- Importando los componentes e inicializando el número de intentos

```
import { Component } from '@angular/core';
import { GameBoardComponent } from './game/game-board/game-board.component';
import { HangmanFigureComponent } from './game/hangman-figure/hangman-figure.component';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css'],
  standalone: true,
  imports: [GameBoardComponent, HangmanFigureComponent]
})
export class AppComponent {
  title = 'Juego Ahorcado';
  intentos: number = 9;
}
```

5. Resultados

Figura 1: Iniciando el servidor de desarrollo de Angular

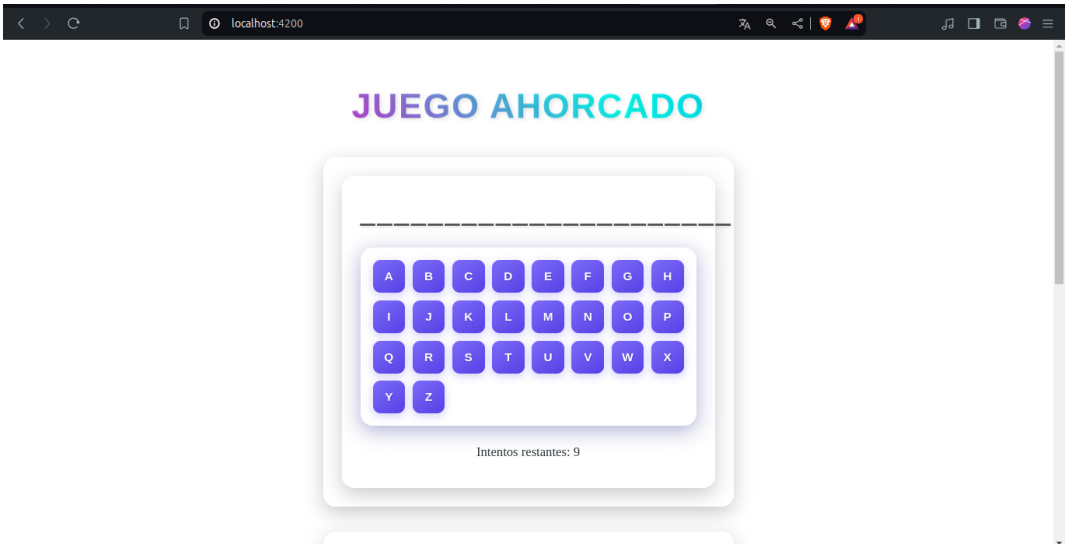
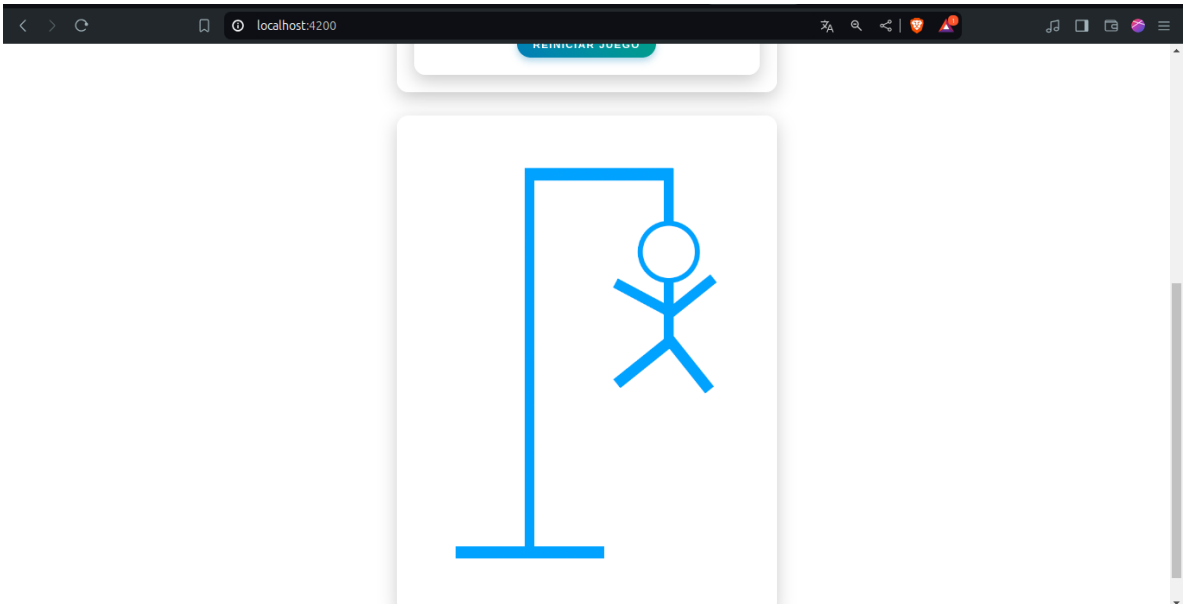


Figura 2: Resultado final del juego

Figura 3: Resultado final del juego


localhost:4200

5.1. 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 1: 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 2: 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	2	
3. Código fuente	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2	X	1	
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	2	
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		15	