Informe de Laboratorio 09

Tema: Laboratorio 09Angular

Nota

Estudiante	Escuela	Asignatura
Diego Alejandro Carbajal Gonzales, Jhonatan David Arias Quispe, Ricardo Mauricio Chambilla Perca	Escuela Profesional de Ingeniería de Sistemas	Programación Web 2 Semestre: II Código: 1702122

Laboratorio	Tema	Duración
09	Laboratorio 09Angular	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2024 - A	Del 01 de Julio 2024	Al 07 de Julio 2024

1. Actividades

- Se hizo la implementación del juego del Ahorcado usando Angular

2. Equipos, materiales y temas utilizados

- Sistema Operativo ArchCraft GNU Linux 64 bits Kernell
- Fedora
- Angular 17
- Android (Termux)
- NeoVim
- Git 2.42.0
- Cuenta en GitHub con el correo institucional.
- Latex

3. URL de Repositorio Github

- URL del Repositorio GitHub para clonar o recuperar.
- `https://github.com/JhonatanDczel/pweb2/tree/main/lab09`

4. Introducción

En este informe, se detalla el desarrollo de un juego de Ahorcado utilizando Angular. El proyecto fue realizado como parte de un laboratorio educativo, con el objetivo de poner en práctica habilidades en el desarrollo de aplicaciones web interactivas. Angular, un framework de JavaScript de desarrollo de aplicaciones web de una sola página (SPA), fue utilizado debido a su robustez y facilidad para gestionar componentes y estados.

5. Estructura del Proyecto

El proyecto consta de varios componentes que trabajan en conjunto para crear la experiencia de juego del Ahorcado. A continuación, se describe cada componente en detalle:

1. **Componente Hangman**
2. **Componente HangmanDisplay**
3. **Componente HangmanKeyboard**
4. **Componente HangmanQuestion**

Cada uno de estos componentes tiene su propio propósito y funcionalidad, y están diseñados para interactuar entre sí.

6. Componente Hangman

6.1. hangman.component.html

El archivo HTML de este componente define la estructura visual del juego. Utiliza varios subcomponentes y directivas de Angular para gestionar el flujo del juego.

```
<div class="hangman">
  <app-hangman-display (gameFinished)="onGameFinished()" [guesses]="guesses"
      [question]="question"></app-hangman-display>
  <ng-container *ngIf="!restartGameBtnShown else restartGameBtn">
    <app-hangman-question [guesses]="guesses" [question]="question"></app-hangman-question>
    <app-hangman-keyboard (keyPressed)="guess($event)" [question]="question"></app-hangman-keyboard>
  </ng-container>
</div>
<ng-template #restartGameBtn>
  <button class="btn" (click)="reset()">
    {{'>>Restart Game<<'}}
  </button>
</ng-template>
```

6.2. hangman.component.scss

El archivo SCSS define el estilo del componente, asegurando que el diseño sea flexible y esté centrado en la pantalla.

```scss
.hangman {
  display: flex;
  max-width: 960px;
  flex-direction: column;
  align-items: center;
}
```

6.3. hangman.component.ts

Este archivo contiene la lógica del juego. Gestiona el estado del juego, maneja las entradas del usuario y coordina la interacción entre los subcomponentes.

```ts
import { Location } from '@angular/common';
import { Component, OnInit } from '@angular/core';
import { HangmanService } from 'src/app/services/hangman.service';

@Component({
  selector: 'app-hangman',
  templateUrl: './hangman.component.html',
  styleUrls: ['./hangman.component.scss'],
})
export class HangmanComponent implements OnInit {
  question: string = '';
  questions: string[] = [];
  guesses: string[] = [];
  category: string = '';
  restartGameBtnShown = false;

  constructor(
    private hangmanService: HangmanService,
    private location: Location
  ) {}

  ngOnInit(): void {
    let jsonPath;
    const url = this.location.path();
    if (url.includes('jsonPath')) {
      jsonPath = url.split('jsonPath=')[1];
    }
    this.hangmanService.getQuestions(jsonPath).subscribe((response) => {
      this.questions = response.items;
      this.category = response.category;
      this.pickNewQuestion();
    });
  }

  guess(letter: string) {
    if (!letter || this.guesses.includes(letter)) {
      return;
    }
    this.guesses = [...this.guesses, letter];
  }

  dummyClick() {
    const key = prompt('Enter a key') || '';
    this.guess(key);
  }

```

```
  reset() {
    this.guesses = [];
    this.pickNewQuestion();
    this.restartGameBtnShown = false;
  }

  pickNewQuestion() {
    const randomIndex = Math.floor(Math.random() * this.questions.length);
    this.question = this.questions[randomIndex];
    console.log(this.question);
  }

  onGameFinished() {
    this.restartGameBtnShown = true;
  }
}
```

6.4. hangman.component.spec.ts

Este archivo contiene pruebas unitarias para asegurar que el componente funcione correctamente.

```
import { ComponentFixture, TestBed } from '@angular/core/testing';
import { HangmanComponent } from './hangman.component';

describe('HangmanComponent', () => {
  let component: HangmanComponent;
  let fixture: ComponentFixture<HangmanComponent>;

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      declarations: [ HangmanComponent ]
    }).compileComponents();
  });

  beforeEach(() => {
    fixture = TestBed.createComponent(HangmanComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

7. Componente HangmanDisplay

7.1. hangman-display.component.html

El archivo HTML de este componente contiene el SVG que representa el dibujo del Ahorcado. Utiliza las propiedades de entrada para actualizar el estado del dibujo según los errores del jugador.

```
<svg
  version="1.1"
  id="hangman"
  xmlns="http://www.w3.org/2000/svg"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  x="0px"
  y="0px"
  width="250px"
```

```
  height="350px"
  viewBox="0 0 250 350"
  enable-background="new 0 0 250 350"
  xml:space="preserve"
>
  <line
    id="base1"
    fill="none"
    stroke="#000000"
    stroke-width="10"
    stroke-miterlimit="10"
    x1="28"
    y1="326.034"
    x2="219"
    y2="326.034"
  />
  <!-- Ms  lneas  y figuras aqu -->
</svg>
```

7.2. hangman-display.component.scss

El archivo SCSS define estilos adicionales para el componente, incluyendo animaciones para el dibujo del Ahorcado.

```
.mistakes-count {
  text-align: center;
}

.man-bounce {
  animation: shimmy 1s infinite;
  animation-direction: alternate;
}

@keyframes shimmy {
  0% {
    transform: translate(0, 0);
  }
  100% {
    transform: translate(0px, 17px);
  }
}
```

7.3. hangman-display.component.ts

Este archivo contiene la lógica que determina el estado del dibujo del Ahorcado según las conjeturas del jugador.

```
import { Component, EventEmitter, Input, OnChanges, OnInit, Output, SimpleChanges } from
    '@angular/core';

@Component({
  selector: 'app-hangman-display',
  templateUrl: './hangman-display.component.html',
  styleUrls: ['./hangman-display.component.scss'],
})
export class HangmanDisplayComponent implements OnInit, OnChanges {
  @Input() guesses: string[] = [];
  @Input() question: string = '';
  @Output() gameFinished = new EventEmitter<boolean>();
  MAX_MISTAKES = 7;
```

```
  mistakesRemaining: number;
  success: boolean = false;

  constructor() {
    this.mistakesRemaining = this.MAX_MISTAKES;
  }

  ngOnChanges(changes: SimpleChanges): void {
    if (
      changes?.['question']?.currentValue &&
      changes?.['question'].currentValue !== changes?.['question'].previousValue
    ) {
      this.mistakesRemaining = this.MAX_MISTAKES;
      this.success = false;
    }
    const guessesCurrentValue = changes?.['guesses']?.currentValue;
    if (
      guessesCurrentValue &&
      guessesCurrentValue.length &&
      guessesCurrentValue !== changes['guesses'].previousValue
    ) {
      const char = [...guessesCurrentValue].pop();
      this.checkGuess(char);
    }
  }

  checkGuess(letter: string) {
    let didWin = true;
    this.mistakesRemaining -= this.wasGuessAMistake(letter);
    for (let i = 0; i < this.question.length; i++) {
        didWin = false;
        break;
      }
    }
    this.success = didWin;
    if (this.success || this.mistakesRemaining === 0) {
      this.gameFinished.emit(this.success);
    }
  }

  wasGuessAMistake(letter: string) {
    for (let i = 0; i < this.question.length; i++) {
      if (this.question[i].toLowerCase() === letter.toLowerCase()) {
        return 0;
      }
    }
    return 1;
  }

  ngOnInit(): void {}
```

7.4. hangman-display.component.spec.ts

Este archivo contiene pruebas unitarias para verificar el correcto funcionamiento del componente.

```
import { ComponentFixture, TestBed } from '@angular/core/testing';
import { HangmanDisplayComponent } from './hangman-display.component';

describe('HangmanDisplayComponent', () => {
  let component: HangmanDisplayComponent;
  let fixture: ComponentFixture<HangmanDisplayComponent>;

  beforeEach(async () => {
```

```
    await TestBed.configureTestingModule({
      declarations: [ HangmanDisplayComponent ]
    }).compileComponents();
  });

  beforeEach(() => {
    fixture = TestBed.createComponent(HangmanDisplayComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

8. Componente HangmanQuestion

8.1. hangman-question.component.html

Este archivo HTML muestra las letras de la palabra o frase que el jugador debe adivinar, ocultando las letras que aún no han sido adivinadas.

```
<div class="question">
  <div class="question-letter" *ngFor="let letter of question">
    <span>{{ guesses.includes(letter.toLowerCase()) ? letter : '_' }}</span>
  </div>
</div>
```

8.2. hangman-question.component.scss

Este archivo SCSS define estilos para el componente de la pregunta, incluyendo la apariencia de las letras y los guiones.

```
.question {
  display: flex;
  flex-wrap: wrap;
  justify-content: center;
  margin-bottom: 20px;
}

.question-letter {
  font-size: 2em;
  margin: 0 5px;
}
```

8.3. hangman-question.component.ts

Este archivo contiene la lógica del componente, que maneja la actualización de las letras mostradas según las conjeturas del jugador.

```
import { Component, Input, OnChanges, SimpleChanges } from '@angular/core';

@Component({
  selector: 'app-hangman-question',
  templateUrl: './hangman-question.component.html',
  styleUrls: ['./hangman-question.component.scss'],
```

```
})
export class HangmanQuestionComponent implements OnChanges {
  @Input() guesses: string[] = [];
  @Input() question: string = '';

  ngOnChanges(changes: SimpleChanges): void {
    // Este mtodo se llama cuando las propiedades de entrada cambian
  }
}
```

8.4. hangman-question.component.spec.ts

Este archivo contiene pruebas unitarias para verificar el correcto funcionamiento del componente.

```
import { ComponentFixture, TestBed } from '@angular/core/testing';
import { HangmanQuestionComponent } from './hangman-question.component';

describe('HangmanQuestionComponent', () => {
  let component: HangmanQuestionComponent;
  let fixture: ComponentFixture<HangmanQuestionComponent>;

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      declarations: [ HangmanQuestionComponent ]
    }).compileComponents();
  });

  beforeEach(() => {
    fixture = TestBed.createComponent(HangmanQuestionComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

9. Componente HangmanKeyboard

9.1. hangman-keyboard.component.html

Este archivo HTML muestra el teclado virtual que el jugador usa para adivinar las letras. Cada botón representa una letra y está deshabilitado si ya ha sido adivinada.

```
<div class="keyboard">
  <button
    *ngFor="let key of keys"
    [disabled]="guesses.includes(key)"
    (click)="keyPressed(key)"
    class="key"
  >
    {{ key }}
  </button>
</div>
```

9.2. hangman-keyboard.component.scss

Este archivo SCSS define estilos para el teclado virtual, asegurando que los botones sean claros y accesibles.

```scss
.keyboard {
  display: flex;
  flex-wrap: wrap;
  justify-content: center;
  max-width: 300px;
  margin: 20px 0;
}

.key {
  font-size: 1.5em;
  margin: 5px;
  padding: 10px;
  background-color: #ccc;
  border: none;
  border-radius: 5px;
  cursor: pointer;
}

.key:disabled {
  background-color: #aaa;
  cursor: not-allowed;
}
```

9.3. hangman-keyboard.component.ts

Este archivo contiene la lógica del componente, que maneja las entradas del teclado virtual y las envía al componente principal para procesarlas.

```ts
import { Component, EventEmitter, Input, OnInit, Output } from '@angular/core';

@Component({
  selector: 'app-hangman-keyboard',
  templateUrl: './hangman-keyboard.component.html',
  styleUrls: ['./hangman-keyboard.component.scss'],
})
export class HangmanKeyboardComponent implements OnInit {
  @Input() guesses: string[] = [];
  @Output() keyPressed = new EventEmitter<string>();

  keys: string[] = 'abcdefghijklmnopqrstuvwxyz'.split('');

  constructor() {}

  ngOnInit(): void {}

  onKeyPressed(key: string) {
    this.keyPressed.emit(key);
  }
}
```

9.4. hangman-keyboard.component.spec.ts

Este archivo contiene pruebas unitarias para verificar el correcto funcionamiento del componente.

```ts
import { ComponentFixture, TestBed } from '@angular/core/testing';
```

```typescript
import { HangmanKeyboardComponent } from './hangman-keyboard.component';

describe('HangmanKeyboardComponent', () => {
  let component: HangmanKeyboardComponent;
  let fixture: ComponentFixture<HangmanKeyboardComponent>;

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      declarations: [ HangmanKeyboardComponent ]
    }).compileComponents();
  });

  beforeEach(() => {
    fixture = TestBed.createComponent(HangmanKeyboardComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

10. Conclusión

El desarrollo de este proyecto permitió aplicar y reforzar conocimientos en Angular, incluyendo el manejo de componentes, servicios, y la comunicación entre componentes a través de `@Input` y `@Output`. El juego de Ahorcado resultante es una aplicación interactiva y educativa que demuestra cómo Angular puede ser utilizado para crear aplicaciones web complejas y bien estructuradas.

El informe detalla cada aspecto del proyecto, desde la estructura del código hasta los estilos y pruebas unitarias, proporcionando una guía comprensiva sobre cómo desarrollar un proyecto similar.

La experiencia adquirida a lo largo de este proyecto será valiosa para futuros desarrollos en Angular y otros frameworks de desarrollo de aplicaciones web.

11. Ejecución

11.1. Prueba en el buscador

Utilizando el browser Firefox, hemos entrado al server local de Angular, en la dirección `http://localhost:4200/`. para monitorera el comportamiento de la aplicación. Asi como poder detectar algunas fallas en la ejecución de la misma.

12. Commits más recientes

- A continuación se presentan los commits más importantes.

1. **commit 769dd24d99b2bc45c9dd464bc2a4ae88b146eaf5**
   Author: JhonatanDczel ¡jariasq@unsa.edu.pe¿
   Date: Sat Jul 6 21:47:16 2024 -0500
   Message: Creating services for the game's flow

2. **commit 99059428b63cfe95a5788ecfbc5f96c7e1ee5913**
   Author: JhonatanDczel ¡jariasq@unsa.edu.pe¿
   Date: Sat Jul 6 21:43:57 2024 -0500
   Message: Creando los demas componentes

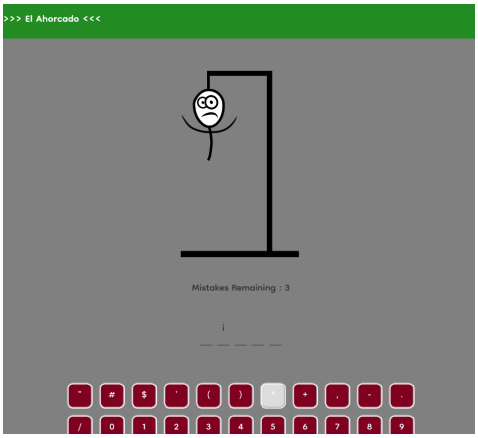

Figura 1: Resultado en el browser.

3. **commit 4c69553df89f8352c26f1e9c82bce6808f410d26**
   Author: JhonatanDczel ¡jariasq@unsa.edu.pe¿
   Date: Sat Jul 6 21:39:55 2024 -0500
   Message: Create component

4. **commit b7a54f62c3e6f18ad7948950753c54799b95bbed**
   Author: JhonatanDczel ¡jariasq@unsa.edu.pe¿
   Date: Sat Jul 6 21:35:30 2024 -0500
   Message: Subiendo assets y styles

5. **commit 8bb0d0582e9891a063388155c8691c6aed017817**
   Author: JhonatanDczel ¡jariasq@unsa.edu.pe¿
   Date: Sat Jul 6 21:33:12 2024 -0500
   Message: Inicializando el pry=oyecto en Angular

6. **commit 6b16ea98d63fc15bbe3578390a011cb3afda7124**
   Author: JhonatanDczel ¡jariasq@unsa.edu.pe¿
   Date: Sat Jul 6 21:32:25 2024 -0500
   Message: Inicializando el proyecto Hangman

7. **commit ebe30d1e1c4f9b0530b9fbb17260b8af059cdd7b**
   Author: JhonatanDczel ¡jariasq@unsa.edu.pe¿
   Date: Wed Jul 3 17:05:02 2024 -0500
   Message: Inicializando el laboratorio 09

8. **commit 1458d2a767f98b808008650cd110673a37bbf17a**
   Author: JhonatanDczel ¡jariasq@unsa.edu.pe¿
   Date: Wed Jun 12 21:20:07 2024 -0500
   Message: Creando el superusuario

13. Rúbricas

13.1. 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.
- 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	Repositorio se pudo clonar y se evidencia la estructura adecuada para revisar los entrega bles. (Se descontará puntos por error o onser vación)	4	X	4	
2. Commits	Hay porciones de código fuente asociado a los commits planificados con explicaciones deta lladas. (El profesor puede preguntar para re frendar calificación).	4	X	4	
3. Ejecución	Se incluyen comandos para ejecuciones y prue bas del código fuente explicadas gradualmente que permitirían replicar el proyecto. (Se des contará puntos por cada omisión)	4	X	1.5	
4. Pregunta	Se responde con completitud a la pregunta for mulada en la tarea. (El profesor puede pregun tar para refrendar calificación)	2	X	2	
5. Ortografía	El documento no muestra errores ortográficos. (Se descontará puntos por error encontrado)	2	X	1.5	
6. Madurez	El Informe muestra de manera general una evolución de la madurez del código fuente con explicaciones puntuales pero precisas, agre gando diagramas generados a partir del códi go fuente y refleja un acabado impecable. (El profesor puede preguntar para refrendar cali ficación).	4	X	3	
Total		20		16	