Informe de Laboratorio 09

Tema: Angular

Nota

Estudiante(s)	Escuela	Asignatura
Mariel Alisson Jara Mamani mjarama@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Programación Web 2 Semestre: I Código: 1702122

Laboratorio	Tema	Duración
09	Angular	04 horas

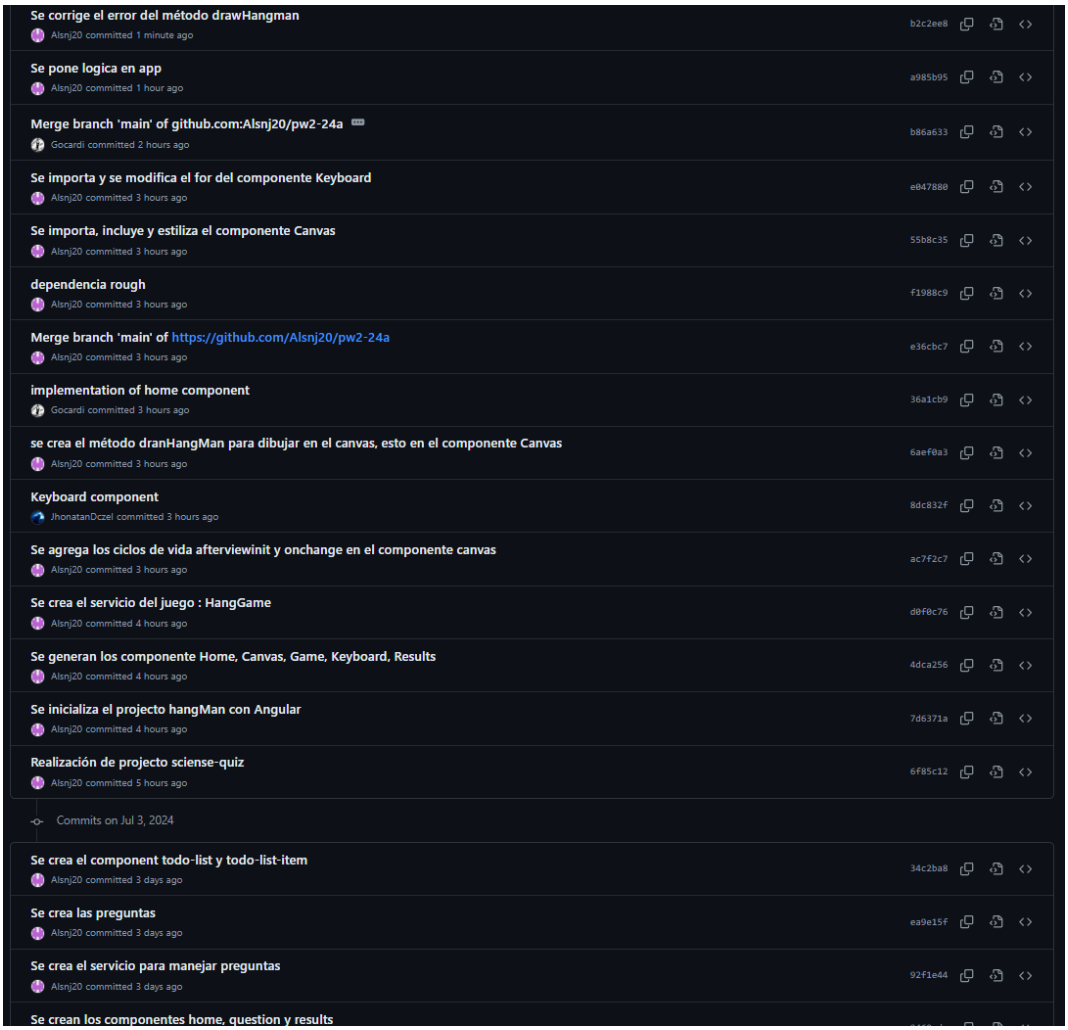
Semestre académico	Fecha de inicio	Fecha de entrega
2023 B	Del 3 Junio 2024	Al 6 Junio 2024

Laboratorio 09

1 Tarea

- **Ejercicio 1:**
  - 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.
- **Observaciones:**
  - Forme grupos de 2 a 4 personas
  - Reportar al profesor que logró culminar la tarea. La tarea debe ser compartida con el profesor en github (CarloCorrales010) y entregada usando el mismo url que se usó para clonar el repositorio. Además cada integrante del grupo deberá crear un video de 5 minutos en flipgrid explicando la aplicación.

2 Commits



Lista de commits realizados en el proyecto.

3 Equipos y materiales utilizados

- Cuenta en GitHub con el correo institucional.
- Sistema Operativo Microsoft Windows 10
- Visual Studio Code
- Git
- Windows PowerShell
- Angular
- Navegador Mozilla Firefox

4 Ejercicio Propuestos: Juego del ahorcado

Para el siguiente proyecto se ha creado una aplicación con Angular que implementa el juego del ahorcado. Se utiliza un elemento Canvas a través de la API de rough.js para dibujar el ahorcado. Para el diseño, se ha empleado Tailwind CSS. Se generaron los siguientes componentes:

- **app.component:** Componente principal que contiene la lógica del juego.
- **canvas.component:** Componente que contiene el canvas para dibujar el ahorcado.
- **keyboard.component:** Componente que contiene el teclado para seleccionar las letras.el juego.

4.1 Componente app

Este componente contiene la lógica del juego. Se encarga de seleccionar una palabra aleatoria del arreglo de palabras y de verificar si la letra seleccionada por el usuario se encuentra en la palabra. Además, se encarga de verificar si el usuario ha ganado o perdido el juego.

app.component.ts

```ts
import { Component } from '@angular/core';
import { RouterOutlet } from '@angular/router';
import { CanvasComponent } from './canvas/canvas.component';
import { KeyboardComponent } from './keyboard/keyboard.component';

@Component({
  selector: 'app-root',
  standalone: true,
  imports: [RouterOutlet, CanvasComponent, KeyboardComponent],
  templateUrl: './app.component.html',
  styleUrl: './app.component.css'
})
export class AppComponent {
  WORD:string = '';

  readonly WORD_LIST = [
    'ANGULAR',
    'JAVASCRIPT',
    'TYPESCRIPT',
    'PROGRAMMING',
    'DEVELOPMENT',
    'COMPONENT',
    'DIRECTIVE',
    'SERVICE',
    'MODULE',
    'ROUTING',
    'LIFE',
    'CYCLE',
    'HTTP',
    'CLIENT',
    'SERVER',
    'DATABASE',
    'QUERY',
    'RESPONSE',
    'REQUEST',
    'OBSERVABLE',
    'SUBSCRIPTION',
```

```
    'OPERATOR',
    'FUNCTION',
    'COMPILATION',
    'TRANSPILATION',
    'INTERPRETATION',
  ]

  readonly TITLE = 'Ahorcado';
  readonly MAX_TRIES = 6;
  tries_number = 0;

  constructor() {
    const random = Math.floor(Math.random() * this.WORD_LIST.length);
    this.WORD = this.WORD_LIST[random];
  }

  addTries() {
    this.tries_number += 1;

    if (this.tries_number === this.MAX_TRIES) {
      alert('Perdiste');
      this.WORD = this.WORD_LIST[Math.floor(Math.random() * this.WORD_LIST.length)];
      this.tries_number = 0;
    }
  }

}
```

- Componente principal que contiene la lógica del juego.

app.component.html

```
<app-canvas [step]="tries_number"></app-canvas>
<app-keyboard
  (addTries)="addTries()"
  [MAX_TRIES]="MAX_TRIES"
  [tries_number]="tries_number"
  [wordPrimary]="WORD"
></app-keyboard>
<router-outlet />
```

- Plantilla del componente **app**.
-

4.2 Componente canvas

Este componente contiene el canvas para dibujar el ahorcado. Se hace uso de la api rough.js para dibujar las figuras geométricas, esto en función de la varibles step pasada como propiedad de entrada @input. A medida que step incrementa, el dibulo del ahorcado se va completando, esto a traves de la función drawHangman.

canvas.component.ts

```ts
import { Component, ElementRef, ViewChild, Input, AfterViewInit, OnChanges, SimpleChanges } from '@angular/core'

// Declaracion para rough.js
declare var rough: any;

@Component({
  selector: 'app-canvas',
  standalone: true,
  imports: [],
  templateUrl: './canvas.component.html',
  styleUrl: './canvas.component.css'
})

export class CanvasComponent implements AfterViewInit, OnChanges {
  @ViewChild('itemCanvas', { static: false }) canvas: ElementRef<HTMLCanvasElement> | undefined;

  @Input() step: number = 0;
  private ctx: CanvasRenderingContext2D | null = null;
  roughCanvas: any;


  ngAfterViewInit(): void {
    if (this.canvas) {
      this.ctx = this.canvas.nativeElement.getContext('2d');
      this.roughCanvas = rough.canvas(this.canvas.nativeElement);
      this.drawHangman(this.step);
    }
  }

  ngOnChanges(changes: SimpleChanges): void {
    if (changes['step'] && !changes['step'].isFirstChange()) {
      this.drawHangman(this.step);
    }
  }

  drawHangman(step: number): void {
    if (this.canvas && this.ctx) {
      this.ctx?.clearRect(0, 0, this.canvas?.nativeElement.width, this.canvas?.nativeElement.height);


      if (0 <= step){
        this.roughCanvas.line(50, 450, 200, 450); // Base
        this.roughCanvas.line(125, 450, 125, 50);  // Poste vertical
        this.roughCanvas.line(125, 50, 280, 50);   // Poste horizontal
        this.roughCanvas.line(280, 50, 280, 100);  // Cuerda
        this.roughCanvas.circle(280, 140, 40);     // Cabeza
      }
      if (1 <= step){
```

```
49          this.roughCanvas.line(280, 180, 280, 350); // Cuerpo
50        }
51        if (2 <= step) {
52          this.roughCanvas.line(280, 350, 240, 450); // Pierna izquierda
53        }
54
55        if (3 <= step) {
56          this.roughCanvas.line(280, 350, 320, 450); // Pierna derecha
57        }
58        if (4 <= step) {
59          this.roughCanvas.line(280, 280, 220, 200); // Brazo izquierdo
60        }
61        if (5 <= step) {
62          this.roughCanvas.line(280, 280, 340, 200); // Brazo derecho
63        }
64
65        if (6 <= step) {
66          this.roughCanvas.circle(280, 140, 40);     // Cabeza
67          this.roughCanvas.line(270, 130, 260, 120); // Ojo derecho
68          this.roughCanvas.line(260, 130, 270, 120);
69          this.roughCanvas.line(290, 130, 300, 120); // Ojo izquierdo
70          this.roughCanvas.line(300, 130, 290, 120);
71          this.roughCanvas.line(270, 150, 300, 145, { roughness: 4 }); // Boca
72          this.roughCanvas.arc(280, 148, 20, 10, 0, 0.6 * Math.PI, true); // Lengua
73        }
74      }
75    }
76  }
```

- Componente para el canvas.

canvas.component.html

```
1  <div class="w-full max-w-md bg-white rounded-md xd">
2    <canvas #itemCanvas width="370" height="490"></canvas>
3  </div>
```

- Plantilla del componente **canvas**.
- La hoja de estilo esta aqui mismo por taiwlind.

4.3 Componente keyboard

Este componente contiene el teclado para seleccionar las letras. Se hace uso de la función selectLetter para seleccionar la letra y verificar si esta se encuentra en la palabra. Además, se verifica si el usuario ha ganado o perdido el juego.

keyboard.component.ts

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

@Component({
  selector: 'app-keyboard',
  standalone: true,
  imports: [],
  templateUrl: './keyboard.component.html',
})
export class KeyboardComponent implements OnInit {
  @Input() wordPrimary = '';
  @Input() MAX_TRIES = 6;
  @Input() tries_number = 0;

  @Output() addTries = new EventEmitter<void>();

  word: string[] = [];
  userWord: string[] = [];

  userLetter: string;

  tries: string[] = [];

  ngOnInit(): void {
    this.word = this.wordPrimary.toUpperCase().split('');
    this.userWord = this.word.map(() => ' ');
  }

  constructor() {
    this.userLetter = '';
  }

  changeLetter(event: Event) {
    this.userLetter = (event.target as HTMLInputElement).value;
  }

  userTry() {
    const letter = this.userLetter.toUpperCase();
    if (this.tries.includes(letter)) {
      return;
    }

    this.tries.push(letter);

    if (!this.word.includes(letter)) {
      this.addTries.emit();

      this.userLetter = '';
      return;
    }
```

```
50
51      this.word.forEach((l, i) => {
52        if (l === letter) {
53          this.userWord[i] = letter;
54        }
55      });
56
57      this.userLetter = '';
58    }
59  }
```

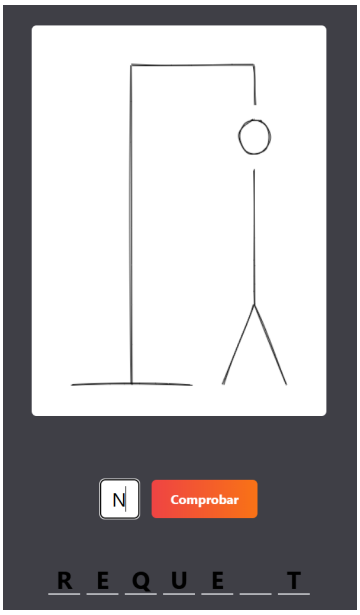
- Componente para el teclado.

keyboard.component.html

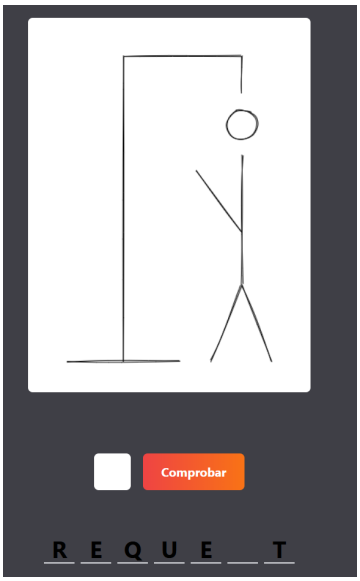
```
1  <div class="flex flex-col gap-10 mt-20 text-black">
2    <div class="flex justify-center gap-4">
3      <input
4        type="text"
5        name="letter"
6        id="letter"
7        [maxLength]="1"
8        class="w-12 px-4 py-2 rounded-md text-2xl"
9        value="{{userLetter}}"
10       (change)="changeLetter($event)"
11     >
12     <button class="bg-gradient-to-r from-red-500 to-orange-500 px-6 py-3 rounded-md font-bold text-white hover:s
13       Comprobar
14     </button>
15   </div>
16
17   <div class="flex justify-center gap-2">
18     @for (letter of userWord; track $index) {
19       <span class="w-10 border-b-2 h-14 text-center text-3xl flex justify-center items-end font-bold">
20         {{letter}}
21       </span>
22     }
23   </div>
24 </div>
```

- Plantilla del componente **keyboard**.

5 Pruebas



Prueba del juego del ahorcado.



Prueba 2 del juego del ahorcado.

6 URL del repositorio en GitHub

- `https://github.com/Alsnj20/pw2-24a/tree/main/lab09`

7 Estructura de laboratorio 09

- El contenido que se entrega en este laboratorio es el siguiente:

```
lab09/
   |--Angular/
      |--hangMan/
         |--src/
            |--app/
               |--app.component.ts
               |--app.component.html
               |--app.component.css
               |--canvas/
                  |--canvas.component.ts
                  |--canvas.component.html
               |--keyboard/
                  |--keyboard.component.ts
                  |--keyboard.component.html
                  |--keyboard.component.css
   |--Latex/
      |--linopinto_pw2_24a_lab09.tex
      |--linopinto_pw2_24a_lab09.pdf
      |--img/
          |--commits.png
          |--prueba.png
   |--sciense-quiz/
|--.gitignore
```

8 Rúbrica

Tabla: Rúbrica para contenido del Informe y evidencias

Contenido y demostración		**Puntos**	**Checklist**	**Estudiante**	**Profesor**
1. GitHub	Repositorio se pudo clonar y se evidencia la estructura adecuada para revisar los entregables. (Se descontará puntos por error o observación)	4	×	4	
2. Commits	Hay porciones de código fuente asociado a los commits planificados con explicaciones detalladas. (El profesor puede preguntar para refrendar calificación)	4	×	4	
3. Ejecución	Se incluyen comandos para ejecuciones y pruebas del código fuente explicadas gradualmente que permitirían replicar el proyecto. (Se descontará puntos por cada omisión)	4	×	4	
4. Pregunta	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación)	2	×	2	
7.Ortografía	El documento no muestra errores ortográficos. (Se descontará puntos por error encontrado)	2	×	1	
8. Madurez	El Informe muestra de manera general una evolución de la madurez del código fuente con explicaciones puntuales pero precisas, agregando diagramas generados a partir del código fuente y refleja un acabado impecable. (El profesor puede preguntar para refrendar calificación)	4	×	4	
Total		20	Completo	19	

9 Referencias

- `https://github.com/`
- `https://git-scm.com/`
- `https://www.w3schools.com/python/`