Registro Diario de Avances

**Fecha:** 01/09/2025

**Autor:**

**Danluis Enrique Romani Mora**

**Eduardo Chaparro Huaman**

**Fabrizcio Argandoña Montalvo**

**Versión:** 1.0

# 1. Introducción

En este documento se muestra el avance de tareas respectivas al Sprint 3 con lo que también se detallan las historias de usuarios pertenecientes y el esfuerzo realizado para completar cada una de ellas, con evidencias del código utilizado, burndown chart y el análisis respectivo al rendimiento del equipo.

# 2. Lineamientos de Diseño

Desarrollar una tercera versión funcional de la aplicación que permita a los usuarios:

* Inicio de sesión y acceso de los alumnos
* Mostrar mensajes en caso de error al momento de iniciar sesión.
* La posibilidad de recuperar la contraseña.
* La visualización del test de inteligencias múl tiples.

# 3. Historias de Usuario y Tareas

| Backlog ID | Historia de Usuario | Tiempo Estimado | Fecha de Inicio | Fecha de Finalización |
| --- | --- | --- | --- | --- |
| HU-1.4 | Acceso de alumnos al sistema | 10 | 1/10/2025 | 3/10/2025 |
| HU-1.5 | Mensaje de error por credenciales incorrectas | 4 | 1/10/2025 | /10/2025 |
| HU-1.6 | Recuperación de contraseña para alumnos | 8 | 1/10/2025 | /10/2025 |
| HU-1.12 | Confirmación visual de cierre de sesión | 3 | /10/2025 | /10/2025 |
| HU-2.1 | Visualización del test en la práctica | 14 | /10/2025 | /10/2025 |

# 4. Evidencias de Código

## 4.1 Historia de Usuario 1.4: Acceso de alumnos al sistema

Archivo Frontend : Login.component.html

<body>

<div class="overlay"></div>

<header>

<h1>INICIO DE SESIÓN</h1>

<h2>Ingrese sus datos</h2>

<div class="logoCole"></div>

</header>

<div class="login-container" [ngClass]="{'fade-mode': animar}">

<div style="display: flex; justify-content: space-between; gap: 10px;">

<button class="crear-button"

[ngClass]="{'active': tipoLogin === 'docente'}"

(click)="setLogin('docente')">Docentes</button>

<button class="crear-button"

[ngClass]="{'active': tipoLogin === 'alumno'}"

(click)="setLogin('alumno')">Alumnos</button>

<button class="crear-button"

[ngClass]="{'active': tipoLogin === 'admin'}"

(click)="setLogin('admin')">Administrador</button>

</div>

<input type="text" class="input-field" placeholder="Usuario" [(ngModel)]="usuario">

<input type="password" class="input-field" placeholder="Contraseña" [(ngModel)]="contrasena">

<button class="login-button" (click)="login()">Ingresar</button>

<p class="recuperar-texto" (click)="recuperar()">¿Olvidaste tu contraseña?</p>

</div>

</body>

Archivo Frontend : Login.component.ts

import { Component } from '@angular/core';

import { Router } from '@angular/router';

import { CommonModule } from '@angular/common';

import { HttpClientModule } from '@angular/common/http';

import { FormsModule } from '@angular/forms';

import { DocentesService } from '../services/docentes.service';

import Swal from 'sweetalert2';

@Component({

selector: 'app-login',

standalone: true,

imports: [CommonModule, HttpClientModule, FormsModule],

templateUrl: './login.component.html',

styleUrl: './login.component.css'

})

export class LoginComponent {

usuario: string = '';

contrasena: string = '';

tipoLogin: string = 'docente'; *// por defecto*

animar: boolean = false;

exito: boolean = false;

constructor(*private* router: Router, *private* docentesService: DocentesService) {}

setLogin(tipo: string) {

this.animar = true; *// activa animación*

setTimeout(() => {

this.tipoLogin = tipo;

this.animar = false; *// termina animación*

}, 300);

}

login() {

if (this.tipoLogin === 'docente') {

*// lógica para docentes*

if (this.usuario =='docente1' && this.contrasena == '1234') {

this.exito = true;

this.router.navigate(['inicio-docentes']);

}

} else if (this.tipoLogin === 'alumno') {

*// lógica para alumnos*

if (this.usuario =='alumno1' && this.contrasena == '1234') {

this.exito = true;

this.router.navigate(['inicio-alumnos']);

}

} else if (this.tipoLogin === 'admin') {

*// lógica para administrador*

if (this.usuario =='admin1' && this.contrasena == '1234') {

this.exito = true;

this.router.navigate(['inicio']);

}

}

if(this.exito==true)

{

Swal.fire({

title: 'Ingreso exitoso',

text: `Ingreso como ${this.tipoLogin}`,

icon: 'success',

confirmButtonText: 'Aceptar',

});

}

else

{

Swal.fire({

title: 'Error',

html: 'Usuario o contraseña incorrectos.<br>Estas intentando ingresar como <b>' + this.tipoLogin + '</b>' + '<br>Si no es correcto cambie de modo de ingreso.',

icon: 'error',

confirmButtonText: 'Aceptar',

});

}

}

recuperar() {

this.router.navigate(['/recuperar-contrasena']);

}

}

Backend donde se recibe las credenciales del alumno, verifica el rate limit y debería autenticar al usuario

@PostMapping("/login")

public ResponseEntity<Map<String, Object>> login(@Valid @RequestBody LoginRequest loginRequest, HttpServletRequest request) {

String clientIP = getClientIP(request);

Map<String, Object> response = new HashMap<>();

*// Rate limiting*

if (!rateLimitService.isLoginAllowed(clientIP) || !rateLimitService.isLoginAllowedByUser(loginRequest.getUsuario())) {

response.put("exito", false);

response.put("mensaje", "Demasiados intentos de login. Intente más tarde.");

auditoriaService.registrarAccion("LOGIN\_BLOQUEADO", "USUARIO", null, "Rate limit excedido", clientIP, loginRequest.getUsuario());

return ResponseEntity.status(429).body(response);

}

try {

response.put("exito", false);

response.put("mensaje", "Funcionalidad de autenticación no implementada.");

return ResponseEntity.status(501).body(response);

} catch (Exception e) {

auditoriaService.registrarLogin(loginRequest.getUsuario(), clientIP, false);

response.put("exito", false);

response.put("mensaje", "Error interno del servidor: " + e.getMessage());

return ResponseEntity.status(500).body(response);

}

}

## 4.2 Historia de Usuario 1.5: Mensaje de error por credenciales incorrectas

Archivo Frontend : Login.component.ts

if(this.exito==true)

{

Swal.fire({

title: 'Ingreso exitoso',

text: `Ingreso como ${this.tipoLogin}`,

icon: 'success',

confirmButtonText: 'Aceptar',

});

}

else

{

Swal.fire({

title: 'Error',

html: 'Usuario o contraseña incorrectos.<br>Estas intentando ingresar como <b>' + this.tipoLogin + '</b>' + '<br>Si no es correcto cambie de modo de ingreso.',

icon: 'error',

confirmButtonText: 'Aceptar',

});

}

## 4.3 Historia de Usuario 1.6: Recuperación de contraseña para alumnos

Archivo Frontend : recuperar-contrasena.component.html

<div class="overlay"></div>

<div class="panel-container">

<div class="panel-header">

<h2>Recuperar Contraseña</h2>

<button class="actividad-button cancelar" (click)="regresar()">Cancelar</button>

</div>

<form (ngSubmit)="confirmar()" #formContrasena="ngForm" class="formulario">

*<!-- Nueva contraseña -->*

<div class="form-group">

<label for="nueva">Nueva Contraseña</label>

<input type="password" id="nueva" name="nueva" [(ngModel)]="nuevaContrasena"

required minlength="6" pattern="^(?=.\*[0-9]).{6,}$" #nuevaCtrl="ngModel" />

<div class="error" \*ngIf="nuevaCtrl.invalid && (nuevaCtrl.dirty || nuevaCtrl.touched)">

<span \*ngIf="nuevaCtrl.errors?.['required']">La contraseña es obligatoria.</span>

<span \*ngIf="nuevaCtrl.errors?.['minlength']">Debe tener al menos 6 caracteres.</span>

<span \*ngIf="nuevaCtrl.errors?.['pattern']">Debe incluir al menos un número.</span>

</div>

</div>

*<!-- Confirmar contraseña -->*

<div class="form-group">

<label for="confirmar">Confirmar Contraseña</label>

<input type="password" id="confirmar" name="confirmar" [(ngModel)]="confirmarContrasena"

required #confirmarCtrl="ngModel" />

<div class="error" \*ngIf="confirmarCtrl.invalid && (confirmarCtrl.dirty || confirmarCtrl.touched)">

<span \*ngIf="confirmarCtrl.errors?.['required']">Debe confirmar la contraseña.</span>

</div>

<div class="error" \*ngIf="confirmarContrasena && confirmarContrasena !== nuevaContrasena">

Las contraseñas no coinciden.

</div>

</div>

*<!-- Botón -->*

<button class="actividad-button" type="submit"

[disabled]="formContrasena.invalid || nuevaContrasena !== confirmarContrasena">

Confirmar

</button>

</form>

</div>

Archivo Frontend : recuperar-contrasena.component.ts

import { Component } from '@angular/core';

import { Router } from '@angular/router';

import { FormsModule } from '@angular/forms';

import { CommonModule } from '@angular/common';

import Swal from 'sweetalert2';

@Component({

selector: 'app-recuperar-contrasena',

templateUrl: './recuperar-contrasena.component.html',

styleUrls: ['./recuperar-contrasena.component.css'],

standalone: true,

imports: [FormsModule,CommonModule]

})

export class RecuperarContrasenaComponent {

nuevaContrasena: string = '';

confirmarContrasena: string = '';

constructor(*private* router: Router) { }

regresar() {

this.router.navigate(['/login']);

}

confirmar() {

*// Validar campos vacíos*

if (!this.nuevaContrasena || !this.confirmarContrasena) {

Swal.fire({

icon: 'error',

title: 'Campos vacíos',

text: 'Debe completar ambos campos.'

});

*return*;

}

*// Validar longitud mínima*

if (this.nuevaContrasena.length < 6) {

Swal.fire({

icon: 'error',

title: 'Contraseña inválida',

text: 'La contraseña debe tener al menos 6 caracteres.'

});

*return*;

}

*// Validar que tenga al menos un número*

*const* regexNumero = /\d/;

if (!regexNumero.test(this.nuevaContrasena)) {

Swal.fire({

icon: 'error',

title: 'Contraseña insegura',

text: 'La contraseña debe contener al menos un número.'

});

*return*;

}

*// Validar que coincidan*

if (this.nuevaContrasena !== this.confirmarContrasena) {

Swal.fire({

icon: 'error',

title: 'Error',

text: 'Las contraseñas no coinciden.'

});

*return*;

}

*// Si todo es válido*

Swal.fire({

icon: 'success',

title: 'Éxito',

text: 'Contraseña cambiada con éxito.'

}).then(() => {

this.router.navigate(['/login']);

});

}

}

## 4.4 Historia de Usuario 1.12: Confirmación visual de cierre de sesión

Archivo Frontend : inicio.component.ts

logout() {

this.router.navigate(['login']);

Swal.fire({

title: 'Sesión cerrada',

text: 'Ahora no tienes acceso a la aplicación',

icon: 'success',

confirmButtonText: 'Aceptar',

});

}

Archivo Frontend : inicio.component.css

*/\* Contenedor principal \*/*

.overlay {

position: fixed;

top: 0;

left: 0;

width: 100%;

height: 100%;

background: rgba(47, 116, 87, 0.6);

z-index: -1;

}

.panel-container {

max-width: 90%;

margin: 30px auto;

padding: 20px;

background: rgba(255, 255, 255, 0.95);

border-radius: 15px;

box-shadow: 0 8px 20px rgba(0, 0, 0, 0.2);

font-family: 'Segoe UI', sans-serif;

text-align: center;

}

*/\* Título \*/*

.panel-container h2 {

margin-bottom: 20px;

color: #2f7457;

}

*/\* Contenedor de scrolls \*/*

.scrolls-container {

display: flex;

justify-content: space-between;

gap: 20px;

}

*/\* Cada lista con scroll \*/*

.scroll-box {

flex: 1;

max-height: 300px;

overflow-y: auto;

padding: 15px;

background: #fff;

border-radius: 12px;

box-shadow: 0 4px 12px rgba(0, 0, 0, 0.15);

transition: transform 0.2s ease;

}

.scroll-box:hover {

transform: scale(1.02);

}

*/\* Encabezados \*/*

.scroll-box h3 {

margin-bottom: 10px;

color: #377937;

}

*/\* Botones de cada elemento \*/*

.item-button {

width: 100%;

background: #338c92;

color: white;

padding: 10px;

border: none;

border-radius: 8px;

margin-bottom: 8px;

font-size: 0.95em;

cursor: pointer;

transition: transform 0.2s ease, background 0.3s ease, box-shadow 0.3s ease;

}

.item-button:hover {

background: #2e6883;

transform: scale(1.03);

box-shadow: 0 4px 10px rgba(0, 0, 0, 0.2);

}

.logout-button {

margin-top: 3%;

background: #e74c3c;

color: white;

padding: 10px;

border: none;

border-radius: 8px;

margin-bottom: 8px;

font-size: 0.95em;

cursor: pointer;

transition: transform 0.2s ease, background 0.3s ease, box-shadow 0.3s ease;

}

.logout-button:hover {

background: #c0392b;

transform: scale(1.03);

box-shadow: 0 4px 10px rgba(0, 0, 0, 0.2);

}

.panel-header {

display: flex;

flex-direction: column;

align-items: center;

}

.panel-header h2 {

margin: 0;

text-align: center;

}

.panel-header .actividad-button {

align-self: flex-end;

margin-top: 10px;

}

.actividad-button {

margin-top: 3%;

background: #e74c3c;

color: white;

padding: 10px;

border: none;

border-radius: 8px;

margin-bottom: 8px;

font-size: 0.95em;

cursor: pointer;

transition: transform 0.2s ease, background 0.3s ease, box-shadow 0.3s ease;

}

.actividad-button:hover {

background: #c0392b;

transform: scale(1.03);

box-shadow: 0 4px 10px rgba(0, 0, 0, 0.2);

}

Archivo Frontend : inicio.component.hmtl

<body>

<div class="overlay"></div>

<div class="panel-container">

<div class="panel-header">

<h2>Panel de Gestión</h2>

<button class="actividad-button" (click)="actividad()">Visualizar Actividad</button>

</div>

<div class="scrolls-container">

*<!-- Alumnos -->*

<div class="scroll-box">

<h3>Alumnos</h3>

<div \*ngFor="let alumno of alumnos" (click)="recuperarContrasena()">

<button class="item-button">{{ alumno }}</button>

</div>

</div>

*<!-- Docentes -->*

<div class="scroll-box">

<h3>Docentes</h3>

<div \*ngFor="let docente of docentes" (click)="recuperarContrasena()">

<button class="item-button">{{ docente }} </button>

</div>

</div>

*<!-- Pruebas -->*

<div class="scroll-box">

<h3>Pruebas</h3>

<div \*ngFor="let prueba of pruebas">

<button class="item-button">{{ prueba }}</button>

</div>

</div>

</div>

<button class="logout-button" (click)="logout()">Cerrar Sesión</button>

</div>

</body>

## 4.5 Historia de Usuario 2.1: Visualización del test en la práctica

Archivo Frontend : test-gardner.component.html

<body>

<div class="overlay"></div>

<div class="panel-container">

<h2>Test de Gardner</h2>

<div class="test-layout">

*<!-- Panel izquierdo: preguntas -->*

<div class="preguntas-section">

<form (ngSubmit)="onSubmit()">

<div \*ngFor="let pregunta of preguntasPaginadas" class="pregunta-card">

<p>{{pregunta.id}}. {{pregunta.texto}}</p>

<div class="opciones">

<label \*ngFor="let opcion of [1,2,3,4,5,6]" [class.selected]="pregunta.respuesta === opcion">

<input

type="radio"

[name]="'pregunta-' + pregunta.id"

[value]="opcion"

[(ngModel)]="pregunta.respuesta"

required

/>

{{ opcion }}

</label>

</div>

</div>

*<!-- Usa el template -->*

<ng-container \*ngTemplateOutlet="navegacion"></ng-container>

<button type="submit" class="item-button">Enviar Test</button>

</form>

</div>

*<!-- Panel derecho: índice -->*

<div class="indice-section">

<h3>Preguntas</h3>

<div class="indice-lista">

<button

\*ngFor="let pregunta of preguntas"

class="indice-item"

[class.respondida]="pregunta.respuesta !== null"

[class.activa]="pregunta.id >= inicioIndice && pregunta.id < inicioIndice + preguntasPorPagina"

(click)="irAPagina(pregunta.id)">

{{ pregunta.id }}

</button>

</div>

*<!-- Usa el mismo template -->*

<ng-container \*ngTemplateOutlet="navegacion"></ng-container>

</div>

</div>

</div>

*<!-- Template reutilizable -->*

<ng-template #navegacion>

<div class="navegacion">

<div class="nav-izquierda">

<button

type="button"

class="nav-button"

[class.oculto]="paginaActual <= 1"

(click)="anteriorPagina()"

>

Anterior

</button>

</div>

<div class="nav-derecha">

<button

type="button"

class="nav-button"

[class.oculto]="paginaActual >= totalPaginas"

(click)="siguientePagina()"

>

Siguiente

</button>

</div>

</div>

</ng-template>

</body>

Archivo Frontend : test-gardner.component.ts

import { Component } from '@angular/core';

import { Router } from '@angular/router';

import { CommonModule } from '@angular/common';

import { FormsModule } from '@angular/forms';

import Swal from 'sweetalert2';

@Component({

selector: 'app-test-gardner',

standalone: true,

imports: [CommonModule, FormsModule],

templateUrl: './test-gardner.component.html',

styleUrls: ['./test-gardner.component.css']

})

export class TestGardnerComponent {

preguntasPorPagina = 5;

paginaActual = 1;

preguntas = [

{ id: 1, texto: 'Me gusta resolver problemas matemáticos o lógicos.', respuesta: null },

{ id: 2, texto: 'Disfruto participar en actividades musicales.', respuesta: null },

{ id: 3, texto: 'Prefiero aprender a través de imágenes, diagramas o mapas mentales.', respuesta: null },

{ id: 4, texto: 'Puedo expresar mis ideas fácilmente con palabras.', respuesta: null },

{ id: 5, texto: 'Me resulta fácil trabajar en grupo.', respuesta: null },

{ id: 6, texto: 'Me gusta observar la naturaleza y aprender de ella.', respuesta: null },

{ id: 7, texto: 'Soy consciente de mis emociones y cómo influyen en mí.', respuesta: null },

{ id: 8, texto: 'Me gusta moverme y aprender haciendo actividades físicas.', respuesta: null },

{ id: 9, texto: 'Disfruto resolver acertijos y desafíos.', respuesta: null },

{ id: 10, texto: 'Me inspiro al escuchar música.', respuesta: null },

{ id: 11, texto: 'Me gusta leer libros.', respuesta: null },

{ id: 12, texto: 'Me gusta jugar videojuegos.', respuesta: null },

{ id: 13, texto: 'Me gusta jugar a los juegos de mesa.', respuesta: null },

{ id: 14, texto: 'Me gusta jugar a los juegos de mesa.', respuesta: null },

{ id: 15, texto: 'Me gusta jugar a los juegos de mesa.', respuesta: null },

{ id: 16, texto: 'Me gusta jugar a los juegos de mesa.', respuesta: null },

{ id: 17, texto: 'Me gusta jugar a los juegos de mesa.', respuesta: null },

{ id: 18, texto: 'Me gusta jugar a los juegos de mesa.', respuesta: null },

{ id: 19, texto: 'Me gusta jugar a los juegos de mesa.', respuesta: null },

{ id: 20, texto: 'Me gusta jugar a los juegos de mesa.', respuesta: null },

{ id: 21, texto: 'Me gusta jugar a los juegos de mesa.', respuesta: null },

];

constructor(*private* router: Router) {}

get totalPaginas() {

*return* Math.ceil(this.preguntas.length / this.preguntasPorPagina);

}

get inicioIndice() {

*return* (this.paginaActual - 1) \* this.preguntasPorPagina + 1;

}

get preguntasPaginadas() {

*const* inicio = (this.paginaActual - 1) \* this.preguntasPorPagina;

*return* this.preguntas.slice(inicio, inicio + this.preguntasPorPagina);

}

siguientePagina() {

if (this.paginaActual < this.totalPaginas) this.paginaActual++;

}

anteriorPagina() {

if (this.paginaActual > 1) this.paginaActual--;

}

irAPagina(id: number) {

this.paginaActual = Math.ceil(id / this.preguntasPorPagina);

}

onSubmit() {

*const* incompletas = this.preguntas.filter(p => p.respuesta === null);

if (incompletas.length > 0) {

Swal.fire({

icon: 'warning',

title: 'Faltan respuestas',

text: 'Por favor responde todas las preguntas antes de enviar.',

});

*return*;

}

console.log('Respuestas del test:', this.preguntas);

Swal.fire({

icon: 'success',

title: 'Respuestas guardadas',

text: 'El resultado se está calculando...',

timer: 2500,

showConfirmButton: false

}).then(() => {

this.router.navigate(['/inicio-alumnos']);

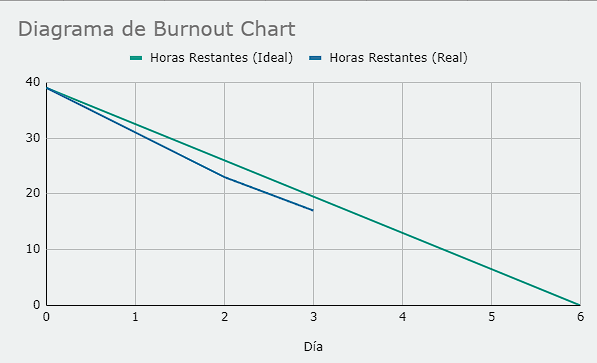
});

}

}

# 5. Burndown Chart

| Backlog ID | Historia de Usuario | Tiempo Estimado | 01/10/2025 | 02/10/2025 | 03/10/2025 | 04/10/2025 | 06/10/2025 | 07/10/2025 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Dia1** | **Dia2** | **Dia3** | **Dia4** | **Dia5** | **Dia6** |
| HU-1.4 | Acceso de alumnos al sistema | 10 | 5 | 4 | 1 |  |  |  |
| HU-1.5 | Mensaje de error por credenciales incorrectas | 4 | 2 | 0 | 1 |  |  |  |
| HU-1.6 | Recuperación de contraseña para alumnos | 8 | 1 | 4 | 2 |  |  |  |
| HU-1.12 | Confirmación visual de cierre de sesión | 3 |  |  | 1 |  |  |  |
| HU-2.1 | Visualización del test en la práctica | 14 |  |  | 1 |  |  |  |
| Tiempo de trabajo | | 39 | 31 | 23 | 17 |  |  |  |



# 6. Análisis del Sprint

# 7. Conclusiones