Registro Diario de Avances – Sprint 1 SISTEC

**Fecha:01/102025**

**Autor:Raurau Quispe, Juan Boris**

**Versión:** 3.0

# 1. Introducción

Este documento detalla el seguimiento diario del Sprint 1 del proyecto SISTEC, centrado en el desarrollo de funcionalidades básicas. Se incluyen actividades diarias, tareas completadas, evidencias de código, un burndown chart y un análisis del rendimiento del equipo.

# 2. Lineamientos de Diseño

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

El Sprint 1 se enfocó en obtener una **primera versión funcional y estable** del sistema SISTEC que permita:

**✔ Funcionalidades clave del Sprint 1**

* Iniciar y cerrar sesión de manera segura.
* Asignar accesos mediante roles.
* Registrar, editar, listar y eliminar equipos.
* Visualizar alertas generadas por RFID.
* Cambiar estado de alertas y registrar historial.

**✔ Consideraciones técnicas**

* Backend implementado con PHP y MySQL.
* Validaciones estrictas para RFID, nombres y roles.
* Seguridad mediante password\_verify, sesiones y control de acceso.
* Comunicación con fetch/AJAX para actualizaciones dinámicas.
* Código estructurado modularmente (carpetas /equipos/, /alertas/, /includes/).

# 3. Historias de Usuario y Tareas

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Backlog ID | Historia de Usuario | Tiempo Estimado | Fecha de Inicio | Fecha de Finalización |
| HU-1.1 | Iniciar Sesión en el Sistema | 2.5 h | 23/09/2025 | 23/09/2025 |
| HU-1.2 | Control de Acceso por Roles | 3 h | 24/09/2025 | 24/09/2025 |
| HU-1.3 | Cierre de Sesión Seguro | 1.5 h | 25/09/2025 | 25/09/2025 |
| HU-2.1 | Registrar Equipos | 3 h | 26/09/2025 | 26/09/2025 |
| HU-2.2 | Editar Equipos | 3 h | 27/09/2025 | 27/09/2025 |
| HU-2.3 | Eliminar Equipos | 2.5 h | 28/09/2025 | 28/09/2025 |
| HU-3.1 | Visualizar Alertas | 3 h | 29/09/2025 | 29/09/2025 |
| HU-3.2 | Cambiar Estado de Alerta | 2.5 h | 30/09/2025 | 30/09/2025 |
| HU-3.3 | Consultar Historial de Cambios | 3 h | 01/10/2025 | 01/10/2025 |

# 4. Evidencias de Código

## 4.1 Historia de Usuario 1.1 — Iniciar Sesión en el Sistema

**Archivo: db.php**

<?php

class DB {

    private static $conn;

    public static function conn() {

        if (!self::$conn) {

            $cfg = require \_\_DIR\_\_ . '/../../config/database.php';

            try {

                self::$conn = new mysqli($cfg['host'], $cfg['user'], $cfg['pass'], $cfg['name']);

            } catch (\mysqli\_sql\_exception $e) {

                $msg = $e->getMessage();

                $code = (int)$e->getCode();

                $unknownDb = ($code === 1049) || (stripos($msg, 'unknown database') !== false);

                if (!$unknownDb) { throw $e; }

                // Bootstrap sin BD

                $bootstrap = new mysqli($cfg['host'], $cfg['user'], $cfg['pass']);

                if ($bootstrap->connect\_error) {

                    http\_response\_code(500);

                    exit('Error de conexión.');

                }

                $schemaPath = \_\_DIR\_\_ . '/../../database/normalized\_schema.sql';

                if (file\_exists($schemaPath)) {

                    $sql = file\_get\_contents($schemaPath);

                    if ($sql !== false) {

                        if (!$bootstrap->multi\_query($sql)) {

                            http\_response\_code(500);

                            exit('Error al inicializar la base de datos.');

                        }

                        while ($bootstrap->more\_results() && $bootstrap->next\_result()) { /\* avanzar \*/ }

                    }

                } else {

                    $bootstrap->query("CREATE DATABASE IF NOT EXISTS `" . $cfg['name'] . "` CHARACTER SET utf8mb4 COLLATE utf8mb4\_general\_ci");

                    $bootstrap->query("USE `" . $cfg['name'] . "`");

                }

                $bootstrap->close();

                // Reintentar conexión

                self::$conn = new mysqli($cfg['host'], $cfg['user'], $cfg['pass'], $cfg['name']);

                // Semilla ADMIN si no hay usuarios

                if (!self::$conn->connect\_error) {

                    if ($r = self::$conn->query("SHOW TABLES LIKE 'usuarios'")) {

                        if ($r->num\_rows > 0) {

                            $c = self::$conn->query("SELECT COUNT(\*) AS c FROM usuarios");

                            $row = $c ? $c->fetch\_assoc() : ['c'=>0];

                            if (intval($row['c'] ?? 0) === 0) {

                                $ridRes = self::$conn->query("SELECT id FROM roles WHERE clave='admin' LIMIT 1");

                                $rid = $ridRes && $ridRes->num\_rows ? intval($ridRes->fetch\_assoc()['id']) : null;

                                if ($rid) {

                                    $usuario = 'ADMIN';

                                    $hash = password\_hash('ADMIN12345', PASSWORD\_DEFAULT);

                                    $stmt = self::$conn->prepare("INSERT INTO usuarios (usuario, contrasena, role\_id) VALUES (?, ?, ?)");

                                    if ($stmt) { $stmt->bind\_param('ssi', $usuario, $hash, $rid); $stmt->execute(); }

                                }

                            }

                        }

                    }

                }

            }

            if (self::$conn->connect\_error) { http\_response\_code(500); exit('Error de conexión.'); }

        }

        return self::$conn;

    }

}

**Archivo: login.php**

<!DOCTYPE html>

<html lang="es">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>Login - SISTEC</title>

  <link rel="stylesheet" href="/sistemap/public/assets/css/style.css">

  <style>

    body {

      font-family: Arial, sans-serif;

      background: linear-gradient(to right, #003366, #005580);

      display: flex;

      justify-content: center;

      align-items: center;

      height: 100vh;

      margin: 0;

    }

    .login-card {

      background-color: white;

      padding: 30px;

      border-radius: 15px;

      box-shadow: 0 8px 16px rgba(0,0,0,0.2);

      width: 320px;

      text-align: center;

    }

    .login-card h2 { margin-bottom: 18px; color: #003366; }

    .login-card input[type="text"], .login-card input[type="password"] {

      width: 100%; padding: 10px; margin-bottom: 12px; border: 1px solid #ccc; border-radius: 8px;

    }

    .login-card button { width: 100%; padding: 10px; background-color: #003366; color: white; border: none; border-radius: 8px; font-weight: bold; cursor: pointer; }

    .login-card button:hover { background-color: #005580; }

    .error { background-color: #ffdddd; color: #cc0000; border: 1px solid #cc0000; padding: 10px; margin-bottom: 12px; border-radius: 8px; }

  </style>

  </head>

<body>

  <?php require\_once \_\_DIR\_\_ . '/../../Core/Session.php'; Session::start(); $csrf = Session::csrfToken(); ?>

  <form class="login-card" method="POST" action="/sistemap/public/login">

    <h2>🔐 Iniciar sesión</h2>

    <?php if (!empty($error)): ?>

      <div class="error"><?= htmlspecialchars($error) ?></div>

    <?php endif; ?>

    <input type="text" name="usuario" placeholder="Usuario" required>

    <input type="password" name="contrasena" placeholder="Contraseña" required>

    <input type="hidden" name="\_csrf" value="<?= htmlspecialchars($csrf) ?>">

    <button type="submit">Ingresar</button>

  </form>

</body>

</html>

## 4.2 Historia de Usuario 1.2 — Control de Acceso por Roles

**Archivo: autenticación**

<?php

class AuthController extends Controller {

    public function showLogin(){

        if (Auth::check()) { header('Location: /sistemap/public/dashboard'); exit; }

        $this->render('auth/login', [], false);

    }

    public function login(){

        $conn = DB::conn();

        $usuario = $\_POST['usuario'] ?? '';

        $contrasena = $\_POST['contrasena'] ?? '';

        if (!Session::validateCsrf($\_POST['\_csrf'] ?? '')) {

            $this->render('auth/login', ['error' => 'Token CSRF inválido'], false);

            return;

        }

        // Detectar si existen tablas normalizadas (roles)

        $isNorm = false;

        if ($r = $conn->query("SHOW TABLES LIKE 'roles'")) { $isNorm = $r->num\_rows > 0; }

        if ($isNorm) {

            $stmt = $conn->prepare('SELECT u.id, u.usuario, u.contrasena, r.clave AS rol FROM usuarios u JOIN roles r ON r.id = u.role\_id WHERE u.usuario = ?');

        } else {

            $stmt = $conn->prepare('SELECT id, usuario, contrasena, rol FROM usuarios WHERE usuario = ?');

        }

        $stmt->bind\_param('s', $usuario);

        $stmt->execute();

        $res = $stmt->get\_result();

        if ($res && $res->num\_rows === 1) {

            $u = $res->fetch\_assoc();

            if (password\_verify($contrasena, $u['contrasena'])) {

                Auth::login($u['usuario'], $u['rol']);

                header('Location: /sistemap/public/');

                exit;

            }

        }

        $this->render('auth/login', ['error' => 'Usuario o contraseña incorrectos'], false);

    }

    public function logout(){

        Auth::logout();

    }

}

**Archivo: roles**

<?php

class UsuariosController extends Controller {

    private function requireRoles(array $roles) {

        Auth::requireLogin();

        $rol = Auth::role();

        if (!in\_array($rol, $roles)) {

            if ($\_SERVER['REQUEST\_METHOD'] === 'GET') {

                header('Location: /sistemap/public/');

            } else {

                header('Content-Type: application/json');

                http\_response\_code(403);

                echo json\_encode(['success' => false, 'error' => 'Acceso denegado']);

            }

            exit;

        }

    }

    public function index() {

        $this->requireRoles(['admin']);

        $usuarios = Usuario::all();

        $this->render('usuarios/index', [

            'usuarios' => $usuarios,

            'rol' => Auth::role(),

            'usuario' => Session::get('usuario')

        ]);

    }

    public function store() {

        try {

            $this->requireRoles(['admin']);

            header('Content-Type: application/json');

            if (!Session::validateCsrf($\_POST['\_csrf'] ?? '')) { http\_response\_code(419); echo json\_encode(['success'=>false,'error'=>'Token CSRF inválido']); return; }

            $usuario = strtoupper(trim($\_POST['usuario'] ?? ''));

            $contrasena = $\_POST['contrasena'] ?? '';

            $rol = $\_POST['rol'] ?? '';

            if ($usuario === '' || $contrasena === '' || $rol === '') { echo json\_encode(['success' => false, 'error' => '⚠️ Complete todos los campos obligatorios.']); return; }

            if (!preg\_match('/^[A-ZÁÉÍÓÚÑÜ]+$/u', $usuario)) { echo json\_encode(['success' => false, 'error' => 'El nombre de usuario solo debe contener letras (sin espacios).']); return; }

            if (!(strlen($contrasena) >= 10 && preg\_match('/[A-Za-z]/', $contrasena) && preg\_match('/[0-9]/', $contrasena))) { echo json\_encode(['success' => false, 'error' => 'La contraseña debe tener mínimo 10 caracteres y combinar letras y números.']); return; }

            if (!in\_array($rol, ['admin','tecnico'], true)) { echo json\_encode(['success'=>false,'error'=>'Rol inválido.']); return; }

            if (Usuario::existsByUsuario($usuario)) { echo json\_encode(['success' => false, 'error' => 'El usuario ya existe.']); return; }

            $res = Usuario::create($usuario, $contrasena, $rol);

            $ok = is\_array($res) ? ($res['ok'] ?? false) : (bool)$res;

            if ($ok) { echo json\_encode(['success' => true, 'message' => '✅ Usuario agregado correctamente.']); }

            else {

                $conn = DB::conn();

                $dbErr = method\_exists($conn, 'error') ? trim($conn->error) : '';

                $mErr = is\_array($res) ? ($res['err'] ?? '') : '';

                $detalle = $mErr !== '' ? $mErr : $dbErr;

                echo json\_encode(['success' => false, 'error' => ($detalle !== '' ? ('Error al registrar: ' . $detalle) : 'Error al registrar.')]);

            }

        } catch (Throwable $ex) {

            http\_response\_code(500);

            echo json\_encode(['success'=>false,'error'=>'Excepción: '.$ex->getMessage()]);

        }

    }

    public function update() {

        try {

            $this->requireRoles(['admin']);

            header('Content-Type: application/json');

            if (!Session::validateCsrf($\_POST['\_csrf'] ?? '')) { http\_response\_code(419); echo json\_encode(['success'=>false,'error'=>'Token CSRF inválido']); return; }

            $id = intval($\_POST['id'] ?? 0);

            $usuario = $\_POST['usuario'] ?? null;

            $contrasena = $\_POST['contrasena'] ?? null;

            $rol = $\_POST['rol'] ?? null;

        if ($id <= 0) { echo json\_encode(['success'=>false,'error'=>'ID inválido']); return; }

        if ($usuario !== null && $usuario !== '') {

            $usuario = strtoupper(trim($usuario));

            if (!preg\_match('/^[A-ZÁÉÍÓÚÑÜ]+$/u', $usuario)) {

                echo json\_encode(['success' => false, 'error' => 'El nombre de usuario solo debe contener letras (sin espacios).']);

                return;

            }

        }

        if ($contrasena !== null && $contrasena !== '' && !(strlen($contrasena) >= 10 && preg\_match('/[A-Za-z]/', $contrasena) && preg\_match('/[0-9]/', $contrasena))) {

            echo json\_encode(['success' => false, 'error' => 'La contraseña debe tener mínimo 10 caracteres y combinar letras y números.']);

            return;

        }

        if ($rol !== null && $rol === '') { echo json\_encode(['success'=>false,'error'=>'El rol no puede estar vacío.']); return; }

        if ($rol !== null && !in\_array($rol, ['admin','tecnico'], true)) { echo json\_encode(['success'=>false,'error'=>'Rol inválido.']); return; }

            // En modo legado, si se envía rol y la columna no existe, crearla para compatibilidad

            if ($rol !== null && $rol !== '' && !Usuario::isNormalizedStatic() && !Usuario::hasColumnStatic('usuarios','rol')) {

                DB::conn()->query("ALTER TABLE usuarios ADD COLUMN rol VARCHAR(20) NOT NULL DEFAULT 'tecnico'");

            }

            $res = Usuario::update($id, $usuario, $contrasena, $rol);

            $ok = is\_array($res) ? ($res['ok'] ?? false) : (bool)$res;

            if ($ok) { echo json\_encode(['success' => true, 'message' => 'Usuario actualizado correctamente.']); }

            else {

                $conn = DB::conn();

                $dbErr = method\_exists($conn, 'error') ? trim($conn->error) : '';

                $mErr = is\_array($res) ? ($res['err'] ?? '') : '';

                $detalle = $mErr !== '' ? $mErr : $dbErr;

                echo json\_encode(['success' => false, 'error' => ($detalle !== '' ? ('Fallo al actualizar: ' . $detalle) : 'No hay cambios o fallo al actualizar.')]);

            }

        } catch (Throwable $ex) {

            http\_response\_code(500);

            echo json\_encode(['success'=>false,'error'=>'Excepción: '.$ex->getMessage()]);

        }

    }

    public function delete() {

        $this->requireRoles(['admin']);

        header('Content-Type: application/json');

        if (!Session::validateCsrf($\_POST['\_csrf'] ?? '')) { http\_response\_code(419); echo json\_encode(['success'=>false,'error'=>'Token CSRF inválido']); return; }

        $id = intval($\_POST['id'] ?? 0);

        $ok = Usuario::delete($id);

        echo json\_encode($ok ? ['success' => true, 'message' => 'Usuario eliminado.'] : ['success' => false, 'error' => 'Error al eliminar.']);

    }

}

## 4.3 Historia de Usuario 1.3 — Cierre de Sesión Seguro

**Archivo: logout.php**

public function logout(){

        Auth::logout();

    }

## 4.4 Historia de Usuario 2.1 — Registrar Equipos

**Archivo: equipos/agregar.php**

public function store() {

        $this->requireRoles(['admin']);

        header('Content-Type: application/json');

        if (!Session::validateCsrf($\_POST['\_csrf'] ?? '')) { http\_response\_code(419); echo json\_encode(['success'=>false,'error'=>'Token CSRF inválido']); return; }

        $data = [

            'numero\_tarjeta\_rfid' => strtoupper(trim($\_POST['numero\_tarjeta\_rfid'] ?? '')),

            'nombre' => strtoupper(trim($\_POST['nombre'] ?? '')),

            'numero\_serie' => strtoupper(trim($\_POST['numero\_serie'] ?? '')),

            'ubicacion' => strtoupper(trim($\_POST['ubicacion'] ?? '')),

        ];

        // Validaciones

        if ($data['numero\_tarjeta\_rfid']==='' || $data['nombre']==='' || $data['numero\_serie']==='' || $data['ubicacion']==='') {

            echo json\_encode(['success'=>false,'error'=>'⚠️ Todos los campos son obligatorios.']); return;

        }

        if (!preg\_match('/^[A-Z0-9]+$/', $data['numero\_tarjeta\_rfid'])) {

            echo json\_encode(['success'=>false,'error'=>'El RFID debe ser alfanumérico (sin espacios).']); return;

        }

        // Nombre: solo letras y espacios

        if (!preg\_match('/^[A-ZÁÉÍÓÚÑÜ ]+$/u', $data['nombre'])) {

            echo json\_encode(['success'=>false,'error'=>'El nombre del equipo solo debe contener letras y espacios.']); return;

        }

        // Número de serie: letras, números y guiones

        if (!preg\_match('/^[A-Z0-9-]+$/', $data['numero\_serie'])) {

            echo json\_encode(['success'=>false,'error'=>'El número de serie solo puede tener letras, números y guiones.']); return;

        }

        if (!preg\_match('/^LABORATORIO\s+\d{3}$/', $data['ubicacion'])) {

            echo json\_encode(['success'=>false,'error'=>'Ubicación inválida. Ejemplo válido: LABORATORIO 101']); return;

        }

        $ok = Equipo::create($data);

        echo json\_encode($ok ? ['success'=>true] : ['success'=>false,'error'=>'Error al registrar.']);

    }

## 4.5 Historia de Usuario 2.2 — Editar Equipos

**Archivo: equipos/actualizar.php**

public function update() {

        $this->requireRoles(['admin']);

        header('Content-Type: application/json');

        if (!Session::validateCsrf($\_POST['\_csrf'] ?? '')) { http\_response\_code(419); echo json\_encode(['success'=>false,'error'=>'Token CSRF inválido']); return; }

        $id = intval($\_POST['id'] ?? 0);

        $data = [

            'numero\_tarjeta\_rfid' => strtoupper(trim($\_POST['numero\_tarjeta\_rfid'] ?? '')),

            'nombre' => strtoupper(trim($\_POST['nombre'] ?? '')),

            'numero\_serie' => strtoupper(trim($\_POST['numero\_serie'] ?? '')),

            'ubicacion' => strtoupper(trim($\_POST['ubicacion'] ?? '')),

        ];

        if ($id<=0) { echo json\_encode(['success'=>false,'error'=>'ID inválido']); return; }

        // Validaciones para actualización parcial: solo validar campos enviados (no vacíos)

        if ($data['numero\_tarjeta\_rfid']!=='' && !preg\_match('/^[A-Z0-9]+$/', $data['numero\_tarjeta\_rfid'])) {

            echo json\_encode(['success'=>false,'error'=>'El RFID debe ser alfanumérico (sin espacios).']); return;

        }

        if ($data['nombre']!=='' && !preg\_match('/^[A-ZÁÉÍÓÚÑÜ ]+$/u', $data['nombre'])) {

            echo json\_encode(['success'=>false,'error'=>'El nombre del equipo solo debe contener letras y espacios.']); return;

        }

        if ($data['numero\_serie']!=='' && !preg\_match('/^[A-Z0-9-]+$/', $data['numero\_serie'])) {

            echo json\_encode(['success'=>false,'error'=>'El número de serie solo puede tener letras, números y guiones.']); return;

        }

        if ($data['ubicacion']!=='' && !preg\_match('/^LABORATORIO\s+\d{3}$/', $data['ubicacion'])) {

            echo json\_encode(['success'=>false,'error'=>'Ubicación inválida. Ejemplo válido: LABORATORIO 101']); return;

        }

        // Confirmar que al menos un campo válido fue enviado

        if ($data['numero\_tarjeta\_rfid']==='' && $data['nombre']==='' && $data['numero\_serie']==='' && $data['ubicacion']==='') {

            echo json\_encode(['success'=>false,'error'=>'No hay cambios para actualizar.']); return;

        }

        $ok = Equipo::update($id, $data);

        echo json\_encode($ok ? ['success'=>true] : ['success'=>false,'error'=>'Error al actualizar.']);

    }

## 4.6 Historia de Usuario 2.3 — Eliminar Equipos

**Archivo: equipos/eliminar.php**

public function delete() {

        $this->requireRoles(['admin']);

        header('Content-Type: application/json');

        if (!Session::validateCsrf($\_POST['\_csrf'] ?? '')) { http\_response\_code(419); echo json\_encode(['success'=>false,'error'=>'Token CSRF inválido']); return; }

        $id = intval($\_POST['id'] ?? 0);

        if ($id<=0) { echo json\_encode(['success'=>false,'error'=>'ID inválido']); return; }

        $ok = Equipo::delete($id);

        echo json\_encode($ok ? ['success'=>true] : ['success'=>false,'error'=>'Error al eliminar.']);

    }

## 4.7 Historia de Usuario 3.1 — Visualizar Alertas

**Archivo: alertas/listar.php**

public function index() {

        $this->requireRoles(['admin']);

        $equipos = Equipo::all();

        $this->render('equipos/index', [

            'equipos' => $equipos,

            'rol' => Auth::role(),

            'usuario' => Session::get('usuario')

        ]);

    }

## 4.8 Historia de Usuario 3.2 — Cambiar Estado de Alerta

**Archivo: alertas/cambiarestado*****(Parte visual para modificar estados)***

public function atender() {

        Auth::requireLogin();

        header('Content-Type: application/json');

        if (!Session::validateCsrf($\_POST['\_csrf'] ?? '')) { http\_response\_code(419); echo json\_encode(['success'=>false,'error'=>'Token CSRF inválido']); return; }

        $id = intval($\_POST['id'] ?? 0);

        if ($id <= 0) { http\_response\_code(400); echo json\_encode(['success'=>false,'error'=>'ID inválido']); return; }

        $user = Session::get('usuario');

        $ok = Alerta::setEstado($id, 'ATENDIDO', $user);

        if ($ok) { echo json\_encode(['success'=>true,'message'=>'Alerta marcada como ATENDIDA']); }

        else { http\_response\_code(500); echo json\_encode(['success'=>false,'error'=>'No se pudo actualizar la alerta']); }

    }

public function revertir() {

        Auth::requireLogin();

        header('Content-Type: application/json');

        if (!Session::validateCsrf($\_POST['\_csrf'] ?? '')) { http\_response\_code(419); echo json\_encode(['success'=>false,'error'=>'Token CSRF inválido']); return; }

        $id = intval($\_POST['id'] ?? 0);

        if ($id <= 0) { http\_response\_code(400); echo json\_encode(['success'=>false,'error'=>'ID inválido']); return; }

        $user = Session::get('usuario');

        $ok = Alerta::setEstado($id, 'NO\_ATENDIDO', $user);

        if ($ok) { echo json\_encode(['success'=>true,'message'=>'Alerta revertida a NO ATENDIDO']); }

        else { http\_response\_code(500); echo json\_encode(['success'=>false,'error'=>'No se pudo actualizar la alerta']); }

    }

## 4.9 Historia de Usuario 3.3 — Consultar Historial de Cambios

**Archivo: alertas/cambiarestado(tabla historial alertas)**

public function historial() {

        Auth::requireLogin();

        header('Content-Type: application/json');

        $id = intval($\_GET['id'] ?? 0);

        if ($id <= 0) { http\_response\_code(400); echo json\_encode(['success'=>false,'error'=>'ID inválido']); return; }

        $items = Alerta::historial($id);

        echo json\_encode(['success'=>true,'data'=>$items], JSON\_UNESCAPED\_UNICODE);

    }

# 5. Burndown Chart

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Backlog ID** | **Historia de Usuario** | **Tiempo Estimado** | **23/09/2025** | **24/09/2025** | **25/09/2025** | **26/09/2025** | **27/09/2025** | **28/09/2025** | **29/09/2025** | **30/09/2025** | **01/10/2025** |
| **Dia1** | **Dia2** | **Dia3** | **Dia4** | **Dia5** | **Dia6** | **Dia7** | **Dia8** | **Dia9** |
| HU-1.1 | Iniciar Sesión | 2.5 | 2.5 | - | - | - | - | - | - | - | - |
| HU-1.2 | Control de Acceso | 3 | 3 | 0 | - | - | - | - | - | - | - |
| HU-1.3 | Logout | 1.5 | 1.5 | 0 | - | - | - | - | - | - | - |
| HU-2.1 | Registrar Equipos | 3 | - | 3 | - | - | - | - | - | - | - |
| HU-2.2 | Editar Equipos | 3 | - | 3 | - | - | - | - | - | - | - |
| HU-2.3 | Eliminar Equipos | 2.5 | - | - | 2.5 | - | - | - | - | - | - |
| HU-3.1 | Listar Alertas | 3 | - | - | - | 3 | - | - | - | - | - |
| HU-3.2 | Cambiar Estado | 2.5 | - | - | - | - | 2.5 | - | - | - | - |
| HU-3.3 | Historial | 3 | - | - | - | - | - | 3 | - | - | - |
| **Tiempo de trabajo** | | 37,5 | 3 | 2 | 5 | 5 | 5 | 5 | 2 | 4 | 3 |

# 6. Análisis del Sprint

* Se cumplió con todas las tareas estimadas.
* El progreso fue constante y sin bloqueos mayores.
* Las estimaciones fueron precisas y se logró un buen ritmo de trabajo.

# 7. Conclusiones

El Sprint 1 concluyó exitosamente con todas las funcionalidades básicas implementadas. Se generaron componentes reutilizables, se probaron los flujos de usuario, y el código quedó preparado para futuras funcionalidades como edición y múltiples listas.

El equipo demostró capacidad de entrega puntual y buena gestión de riesgos menores. Las métricas indican que la carga fue bien distribuida y los objetivos del sprint se cumplieron completamente.