# Sprint 1 - Fundación del Sistema

**Duración**: 2 semanas (02/09/2025 - 15/09/2025)
**Tipo**: Entrega académica
**Objetivo**: Establecer la base del sistema con autenticación y gestión básica de personal **Capacity**: 80 horas (2 desarrolladores x 40 horas/semana)
**Sprint Goal**: Sistema de autenticación funcional y CRUD básico de trabajadores operativo

---

## Sprint Backlog Overview

| Work Item Type | Count | Story Points | Status |
|---|---|---|---|
| User Story | 7 | 18 | New |
| Task | 28 | - | New |
| Bug | 0 | - | - |

---

## Matriz de Riesgo Calórica - Sprint 1

### Metodología de Evaluación

- **Riesgo Técnico** (1-5): Complejidad de implementación
- **Impacto de Negocio** (1-5): Criticidad para el funcionamiento del sistema
- **Esfuerzo** (1-5): Story Points y tiempo estimado

### Epic: Gestión de Identidad y Acceso

**HU-033: Login de usuario ★ CRÍTICO (5 SP) - [EN AZURE]**

**Parent Feature**: Autenticación Básica
**Assigned To**: [Developer 1]
**Priority**: 1

**User Story:**
Como usuario del sistema quiero poder iniciar sesión con mi email y contraseña para acceder a las funcionalidades del sistema

**Acceptance Criteria:**

- [x] Formulario de login con email y contraseña
- [x] Validación de credenciales en backend
- [x] Generación y persistencia de token JWT
- [x] Redirección a dashboard después de login exitoso
- [x] Manejo de errores de autenticación con mensajes claros

**Development Tasks:**

- [x] **Task 1.1**: Crear modelo de Usuario en base de datos (4h)
  - *Description*: Definir tabla usuarios con campos email, password hash, rol, timestamps
  - *Assigned To*: [Developer 1]
  - *Area*: Backend\Base-Datos
- [x] **Task 1.2**: Implementar endpoint POST /auth/login (6h)
  - *Description*: Controlador para autenticar usuario y generar JWT
  - *Assigned To*: [Developer 1]
  - *Area*: Backend\API
- [x] **Task 1.3**: Crear componente LoginForm en React (4h)
  - *Description*: Formulario con validaciones frontend
  - *Assigned To*: [Developer 2]
  - *Area*: Frontend\Autenticacion
- [x] **Task 1.4**: Implementar hook useAuth para gestión de estado (3h)
  - *Description*: Context y hook para manejar autenticación global
  - *Assigned To*: [Developer 2]
  - *Area*: Frontend\Autenticacion
- [ ] **Task 1.5**: Testing de login con casos válidos e inválidos (3h)
  - *Description*: Pruebas unitarias y de integración
  - *Assigned To*: [Developer 1]
  - *Area*: Testing

---

**HU-034: Logout de usuario ★ CRÍTICO (2 SP) - [EN AZURE]**

**Parent Feature**: Autenticación Básica
**Assigned To**: [Developer 2]
**Priority**: 1

**User Story:**
Como usuario autenticado quiero poder cerrar mi sesión para proteger mi información al salir del sistema

**Acceptance Criteria:**

- [ ] Botón de logout visible en header
- [ ] Invalidación del token JWT en backend
- [ ] Limpieza del estado local de autenticación
- [ ] Redirección automática a página de login

**Development Tasks:**

- [ ] **Task 2.1**: Crear endpoint POST /auth/logout (2h)
  - *Description*: Endpoint para invalidar token JWT

    - *Assigned To*: [Developer 1]
    - *Area*: Backend\API

- [ ] **Task 2.2**: Implementar componente LogoutButton (2h)

    - *Description*: Botón con confirmación y manejo de estado
    - *Assigned To*: [Developer 2]
    - *Area*: Frontend\Autenticacion

- [ ] **Task 2.3**: Configurar limpieza automática de rutas protegidas (2h)

    - *Description*: Guard de rutas que valide token válido
    - *Assigned To*: [Developer 2]
    - *Area*: Frontend\Autenticacion

---

## 👥 Epic: Administración de Personal

**HU-001: Crear registro de trabajador ⭐ CRÍTICO (5 SP) - [EN AZURE]**

**Parent Feature**: CRUD de Trabajadores
**Assigned To**: [Developer 1]
**Priority**: 1

**User Story:**
Como administrador quiero registrar un nuevo trabajador para gestionar el personal de la empresa agrícola

**Acceptance Criteria:**

- [ ] Formulario completo con datos personales y laborales
- [ ] Validación de cédula y email únicos
- [ ] Selección de cargo y departamento desde catálogos
- [x] Guardado exitoso en base de datos
- [ ] Confirmación visual de registro exitoso

**Development Tasks:**

- [x] **Task 3.1**: Diseñar modelo de Trabajador en BD (3h)

    - *Description*: Tabla trabajadores con todos los campos requeridos
    - *Assigned To*: [Developer 1]
    - *Area*: Backend\Base-Datos

- [x] **Task 3.2**: Crear endpoint POST /trabajadores (5h)

    - *Description*: API para crear trabajador con validaciones
    - *Assigned To*: [Developer 1]
    - *Area*: Backend\API

- [ ] **Task 3.3**: Desarrollar FormularioTrabajador component (6h)

    - *Description*: Formulario completo con validaciones frontend

    - *Assigned To*: [Developer 2]
    - *Area*: Frontend\Personal

- ☐ **Task 3.4**: Implementar validaciones de negocio (2h)
    - *Description*: Validar cédula única, formato email, campos requeridos
    - *Assigned To*: [Developer 1]
    - *Area*: Backend\Validaciones

---

## 👥 Epic: Administración de Personal

**HU-001: Crear registro de trabajador ⭐ CRÍTICO (5 SP) - [EN AZURE]**

**Parent Feature**: CRUD de Trabajadores
**Priority**: 1

**User Story:**
Como administrador quiero registrar un nuevo trabajador para gestionar el personal de la empresa agrícola

**Acceptance Criteria:**

- ☐ Formulario completo con datos personales y laborales
- ☐ Validación de cédula y email únicos
- ☐ Selección de cargo y departamento desde catálogos
- ☐ Guardado exitoso en base de datos
- ☐ Formulario completo con datos personales y laborales
- ☐ Validación de cédula y email únicos
- ☐ Selección de cargo y departamento desde catálogos
- ☑ Guardado exitoso en base de datos
- ☐ Confirmación visual de registro exitoso
    - *Assigned To*: [Developer 1]
- ☐ **Task 3.2**: Crear endpoint POST /trabajadores (5h)
    - *Description*: API para crear trabajador con validaciones
    - *Area*: Backend\API
- ☐ **Task 3.3**: Desarrollar FormularioTrabajador component (6h)
    - *Assigned To*: [Developer 2]
    - *Area*: Frontend\Personal
    - *Description*: Validar cédula única, formato email, campos requeridos

- *Assigned To*: [Developer 1]
- *Area*: Backend\Validaciones

**HU-002: Asignar información laboral al trabajador (3 SP) - [EN AZURE]**

**Parent Feature**: CRUD de Trabajadores
**Assigned To**: [Developer 2]
**Priority**: 2

**User Story:**
Como administrador quiero asignar información laboral específica al trabajador para mantener datos completos y actualizados

**Acceptance Criteria:**

**Development Tasks:**

- *Description*: Agregar campos laborales a tabla trabajadores
- *Assigned To*: [Developer 1]
- *Area*: Backend\Base-Datos
- ☐ **Task 4.2**: Crear endpoint PATCH /trabajadores/:id/info-laboral (3h)
  - *Description*: API para actualizar información laboral específica
  - *Assigned To*: [Developer 1]
- ☐ **Task 4.3**: Desarrollar FormularioInfoLaboral component (4h)
  - *Description*: Formulario específico para datos laborales
  - *Assigned To*: [Developer 2]

---

**HU-000: Consulta de empleados (3 SP) - [EN AZURE]**

**Assigned To**: [Developer 2]
**Priority**: 2

Como usuario quiero consultar la lista de empleados para ver información del personal y realizar búsquedas específicas

**Acceptance Criteria:**

- ☐ Filtros por departamento, cargo y estado
- ☐ Búsqueda por nombre, cédula o email
- ☐ Vista detallada al hacer click en empleado **Development Tasks:**
- ☐ **Task 5.1**: Crear endpoint GET /trabajadores con filtros (4h)
  - *Description*: API con paginación, filtros y búsqueda

    - *Assigned To*: [Developer 1]
    - *Area*: Backend\API

- ☐ **Task 5.2**: Desarrollar componente ListaTrabajadores (5h)
    - *Description*: Lista con paginación, filtros y búsqueda
    - *Assigned To*: [Developer 2]

- ☐ **Task 5.3**: Crear componente DetalleTrabajador (3h)
    - *Description*: Modal o página con información completa
    - *Area*: Frontend\Personal

---

**Parent Feature**: Gestión de Roles
**Assigned To**: [Developer 1]
**Priority**: 1

**User Story:**
Como administrador quiero crear roles de usuario para organizar la estructura de permisos del sistema

**Acceptance Criteria:**

- ☐ CRUD completo de roles (crear, ver, editar, eliminar)
- ☐ Definición de permisos por módulo
- ☐ Roles predefinidos: Administrador, Supervisor, Trabajador
- ☐ Validación de nombres de roles únicos

**Development Tasks:**

- ☐ **Task 6.1**: Crear modelo Rol y tabla de permisos (3h)
    - *Description*: Tablas roles y roles_permisos con relaciones
    - *Assigned To*: [Developer 1]
    - *Area*: Backend\Base-Datos

- ☐ **Task 6.2**: Implementar endpoints CRUD para roles (4h)
    - *Description*: API completa para gestión de roles
    - *Assigned To*: [Developer 1]
    - *Area*: Backend\API

- ☐ **Task 6.3**: Crear componente GestionRoles (3h)
    - *Description*: Interface para CRUD de roles
    - *Assigned To*: [Developer 2]
    - *Area*: Frontend\Admin

---

**HU-035: Registro de nuevo usuario (3 SP) - [EN AZURE]**

**Parent Feature**: Gestión de Usuarios
**Assigned To**: [Developer 1]
**Priority**: 1

**User Story:**
Como administrador quiero registrar nuevos usuarios en el sistema para que puedan acceder según su rol asignado

**Acceptance Criteria:**

- ☐ Formulario de registro con campos: nombre, email, contraseña, rol
- ☐ Validación de email único en el sistema
- ☐ Encriptación segura de contraseñas
- ☐ Asignación de rol durante el registro
- ☐ Confirmación de creación exitosa

**Development Tasks:**

- ☐ **Task 6.4**: Crear endpoint POST /usuarios/registro (4h)
  - *Description*: API para crear nuevo usuario con validaciones
  - *Assigned To*: [Developer 1]
  - *Area*: Backend\API
- ☐ **Task 6.5**: Desarrollar FormularioRegistroUsuario component (4h)
  - *Description*: Formulario completo para registro de usuarios
  - *Assigned To*: [Developer 2]
  - *Area*: Frontend\Admin

---

# 🏗 Infrastructure & Setup Tasks

## Environment Setup

- ☐ **Task 7.1**: Configurar proyecto backend Express + TypeScript (4h)
  - *Description*: Setup inicial con estructura de carpetas, dependencias básicas
  - *Assigned To*: [Developer 1]
  - *Area*: DevOps\Setup
- ☐ **Task 7.2**: Configurar proyecto frontend React + TypeScript (3h)
  - *Description*: Create React App con TypeScript, estructura de carpetas
  - *Assigned To*: [Developer 2]
  - *Area*: DevOps\Setup
- ☐ **Task 7.3**: Configurar base de datos MySQL (3h)
  - *Description*: Instancia local, configuración de conexión, migraciones básicas
  - *Assigned To*: [Developer 1]
  - *Area*: Backend\Base-Datos

- [ ] **Task 7.4**: Configurar CORS y middleware básico (2h)
    - *Description*: Middleware de autenticación, CORS, body-parser, logging
    - *Assigned To*: [Developer 1]
    - *Area*: Backend\API

### Development Tools

- [ ] **Task 8.1**: Configurar variables de entorno (.env) (1h)
    - *Description*: Configuración para desarrollo, staging y producción
    - *Assigned To*: [Developer 1]
    - *Area*: DevOps\Configuration
- [ ] **Task 8.2**: Setup de Axios y servicios API (2h)
    - *Description*: Cliente HTTP configurado con interceptors
    - *Assigned To*: [Developer 2]
    - *Area*: Frontend\Services
- [ ] **Task 8.3**: Configurar React Router y estructura de rutas (2h)
    - *Description*: Rutas principales, guards de autenticación
    - *Assigned To*: [Developer 2]
    - *Area*: Frontend\Routing

---

## 📊 Sprint Metrics & Tracking

### Velocity Tracking

- **Planned Story Points**: 18
- **Planned Task Hours**: 88
- **Daily Standup**: 9:00 AM (lunes a viernes)
- **Sprint Review**: 15/09/2025 - 2:00 PM
- **Sprint Retrospective**: 15/09/2025 - 3:00 PM

### Definition of Ready (DoR)

Una User Story está lista para desarrollo cuando:

- [ ] Acceptance criteria claros y verificables
- [ ] Dependencies identificadas y resueltas
- [ ] Tasks técnicas definidas con estimaciones
- [ ] Mockups o wireframes disponibles (si aplica)
- [ ] Testing approach definido

### Definition of Done (DoD)

Una User Story está completa cuando:

- [ ] Todos los acceptance criteria cumplidos

- [ ] Código revisado por peer (Pull Request aprobado)
- [ ] Pruebas unitarias implementadas (mín. 80% cobertura)
- [ ] Pruebas de integración ejecutadas exitosamente
- [ ] Documentación API actualizada (si aplica)
- [ ] Deployed a ambiente de staging
- [ ] Validado por Product Owner

---

## Sprint Planning Details

### Sprint Capacity Calculation

| Developer | Capacity (hours) | Availability | Adjusted Hours |
|---|---|---|---|
| Developer 1 | 40h | 100% | 40h |
| Developer 2 | 40h | 100% | 40h |
| **Total** | **80h** | - | **80h** |

### Task Distribution

| Developer | User Stories | Tasks | Estimated Hours |
|---|---|---|---|
| Developer 1 | HU-033, HU-001, HU-005 | 12 tasks | 42h |
| Developer 2 | HU-034, HU-002, HU-000 | 12 tasks | 38h |

### Risk Assessment

| Risk | Impact | Probability | Mitigation |
|---|---|---|---|
| Base de datos compleja | High | Medium | Simplificar modelo inicial |
| Integración Frontend-Backend | Medium | Low | API-first approach |
| Tiempo insuficiente | High | Medium | Priorizar HU críticas |
| Dependencias externas | Low | Low | Mock services iniciales |

---

## Daily Standup Template

### Questions para cada developer:

1. **¿Qué hiciste ayer?** - Tasks completadas
2. **¿Qué harás hoy?** - Tasks planeadas
3. **¿Hay impedimentos?** - Bloqueos técnicos o de proceso
4. **¿Necesitas ayuda?** - Colaboración requerida

### Tracking Board Columns:

- **To Do** - Tasks no iniciadas

- **In Progress** - Tasks en desarrollo
- **In Review** - Pull requests pendientes
- **Testing** - En proceso de QA
- **Done** - Completadas y validadas

---

## 📋 Sprint Review Agenda

### Demo Flow (20 minutos total):

1. **Autenticación** (5 min)
    - Login exitoso y fallido
    - Logout y limpieza de sesión
    - Navegación con usuario autenticado
2. **Gestión de Trabajadores** (10 min)
    - Crear nuevo trabajador
    - Asignar información laboral
    - Consultar lista con filtros
    - Ver detalle de trabajador
3. **Gestión de Roles** (5 min)
    - Crear roles básicos
    - Asignar permisos
    - Validaciones de seguridad

### Acceptance Criteria Review:

- [ ] Todas las User Stories demostrables
- [ ] Criterios de aceptación validados
- [ ] Performance aceptable (< 2seg load time)
- [ ] UI/UX intuitiva para usuarios finales
- [ ] Error handling visible y user-friendly

---

## 🔧 Technical Debt & Improvements

### Identified Technical Debt:

- [ ] **Security**: Implementar rate limiting en APIs
- [ ] **Performance**: Optimizar queries de base de datos
- [ ] **Testing**: Aumentar cobertura de tests e2e
- [ ] **Documentation**: Completar documentación técnica

### Next Sprint Improvements:

- [ ] Implementar middleware de logging estructurado
- [ ] Configurar pipeline CI/CD básico

- [ ] Agregar validaciones más robustas
- [ ] Mejorar manejo de errores global