**3. Historias Técnicas**

**HT-001: Configuración del Entorno de Desarrollo**

* **Rol**: Desarrollador
* **Característica**: Infraestructura
* **Funcionalidad**: Configurar entorno de desarrollo
* **Razón**: Para establecer un ambiente de desarrollo consistente
* **Resultado**: Entorno configurado y documentado

**Escenario 1: Configuración de backend**

* **Criterio de Aceptación**: Backend configurado correctamente
* **Contexto**: Inicio del proyecto
* **Evento**: El desarrollador configura Django, PostgreSQL y dependencias
* **Resultado**: Backend funcional en entorno local
* **Comportamiento esperado**: El servidor de desarrollo se ejecuta sin errores y responde a peticiones

**Escenario 2: Configuración de frontend**

* **Criterio de Aceptación**: Frontend configurado correctamente
* **Contexto**: Inicio del proyecto
* **Evento**: El desarrollador configura Next.js y dependencias
* **Resultado**: Frontend funcional en entorno local
* **Comportamiento esperado**: La aplicación se compila y ejecuta sin errores

**HT-002: Diseño de Base de Datos**

* **Rol**: Desarrollador
* **Característica**: Persistencia de datos
* **Funcionalidad**: Diseñar esquema de base de datos
* **Razón**: Para almacenar y relacionar datos del sistema
* **Resultado**: Esquema de base de datos implementado

**Escenario 1: Diseño de modelos**

* **Criterio de Aceptación**: Modelos diseñados correctamente
* **Contexto**: Fase de diseño
* **Evento**: El desarrollador diseña los modelos de datos
* **Resultado**: Diagrama entidad-relación completo
* **Comportamiento esperado**: El diseño cubre todos los requerimientos de almacenamiento

**Escenario 2: Implementación de migraciones**

* **Criterio de Aceptación**: Migraciones implementadas correctamente
* **Contexto**: Fase de implementación
* **Evento**: El desarrollador crea y ejecuta migraciones
* **Resultado**: Tablas creadas en la base de datos
* **Comportamiento esperado**: Las migraciones se ejecutan sin errores y crean el esquema diseñado

**HT-003: Implementación de Autenticación JWT**

* **Rol**: Desarrollador
* **Característica**: Seguridad
* **Funcionalidad**: Implementar autenticación con JWT
* **Razón**: Para proteger el acceso al sistema
* **Resultado**: Sistema de autenticación seguro

**Escenario 1: Generación de tokens**

* **Criterio de Aceptación**: Tokens generados correctamente
* **Contexto**: Usuario inicia sesión
* **Evento**: El sistema genera tokens de acceso y refresco
* **Resultado**: Tokens JWT válidos entregados al cliente
* **Comportamiento esperado**: Los tokens contienen la información necesaria y tienen la duración configurada

**Escenario 2: Validación de tokens**

* **Criterio de Aceptación**: Tokens validados correctamente
* **Contexto**: Usuario realiza petición autenticada
* **Evento**: El sistema valida el token de acceso
* **Resultado**: Acceso concedido o denegado según validez
* **Comportamiento esperado**: Las peticiones con tokens válidos son procesadas, las inválidas son rechazadas

**HT-004: Implementación de Control de Acceso Basado en Roles**

* **Rol**: Desarrollador
* **Característica**: Seguridad
* **Funcionalidad**: Implementar RBAC
* **Razón**: Para restringir acceso según rol del usuario
* **Resultado**: Permisos configurados por rol

**Escenario 1: Definición de permisos**

* **Criterio de Aceptación**: Permisos definidos correctamente
* **Contexto**: Fase de diseño de seguridad
* **Evento**: El desarrollador define permisos por rol
* **Resultado**: Matriz de permisos documentada
* **Comportamiento esperado**: Cada rol tiene asignados los permisos necesarios para sus funciones

**Escenario 2: Implementación de middleware de autorización**

* **Criterio de Aceptación**: Middleware implementado correctamente
* **Contexto**: Fase de implementación
* **Evento**: El desarrollador implementa middleware de autorización
* **Resultado**: Acceso controlado según rol
* **Comportamiento esperado**: Las peticiones son filtradas según los permisos del rol del usuario

**HT-005: Integración con API del SRI**

* **Rol**: Desarrollador
* **Característica**: Integración externa
* **Funcionalidad**: Integrar con API del SRI
* **Razón**: Para validar y autorizar facturas electrónicas
* **Resultado**: Facturas autorizadas por el SRI

**Escenario 1: Configuración de credenciales**

* **Criterio de Aceptación**: Credenciales configuradas correctamente
* **Contexto**: Fase de configuración
* **Evento**: El desarrollador configura credenciales del SRI
* **Resultado**: Sistema configurado para comunicarse con el SRI
* **Comportamiento esperado**: Las credenciales se almacenan de forma segura y son accesibles para la integración

**Escenario 2: Envío de facturas**

* **Criterio de Aceptación**: Facturas enviadas correctamente
* **Contexto**: Generación de factura
* **Evento**: El sistema envía la factura al SRI
* **Resultado**: Factura procesada por el SRI
* **Comportamiento esperado**: El sistema recibe y almacena la respuesta del SRI (autorización o rechazo)

**HT-006: Implementación de API RESTful**

* **Rol**: Desarrollador
* **Característica**: Arquitectura
* **Funcionalidad**: Implementar API RESTful
* **Razón**: Para permitir comunicación entre frontend y backend
* **Resultado**: API documentada y funcional

**Escenario 1: Diseño de endpoints**

* **Criterio de Aceptación**: Endpoints diseñados correctamente
* **Contexto**: Fase de diseño
* **Evento**: El desarrollador diseña los endpoints de la API
* **Resultado**: Documentación de la API completa
* **Comportamiento esperado**: Los endpoints cubren todas las funcionalidades requeridas

**Escenario 2: Implementación de serializers**

* **Criterio de Aceptación**: Serializers implementados correctamente
* **Contexto**: Fase de implementación
* **Evento**: El desarrollador implementa serializers para modelos
* **Resultado**: Datos serializados correctamente
* **Comportamiento esperado**: Los datos se transforman correctamente entre JSON y objetos del modelo

**HT-007: Configuración de Entorno de Producción**

* **Rol**: DevOps
* **Característica**: Infraestructura
* **Funcionalidad**: Configurar entorno de producción
* **Razón**: Para desplegar la aplicación en producción
* **Resultado**: Sistema desplegado y accesible

**Escenario 1: Configuración de servidor**

* **Criterio de Aceptación**: Servidor configurado correctamente
* **Contexto**: Fase de despliegue
* **Evento**: El DevOps configura el servidor de producción
* **Resultado**: Servidor listo para despliegue
* **Comportamiento esperado**: El servidor tiene todas las dependencias instaladas y configuradas

**Escenario 2: Configuración de CI/CD**

* **Criterio de Aceptación**: Pipeline CI/CD configurado correctamente
* **Contexto**: Fase de automatización
* **Evento**: El DevOps configura pipeline de integración y despliegue
* **Resultado**: Despliegues automatizados
* **Comportamiento esperado**: Los cambios en el repositorio desencadenan pruebas y despliegue automático

**HT-008: Implementación de Pruebas Automatizadas**

* **Rol**: Desarrollador
* **Característica**: Calidad
* **Funcionalidad**: Implementar pruebas automatizadas
* **Razón**: Para garantizar la calidad del código
* **Resultado**: Suite de pruebas completa

**Escenario 1: Implementación de pruebas unitarias**

* **Criterio de Aceptación**: Pruebas unitarias implementadas correctamente
* **Contexto**: Fase de desarrollo
* **Evento**: El desarrollador implementa pruebas unitarias
* **Resultado**: Cobertura de pruebas adecuada
* **Comportamiento esperado**: Las pruebas verifican el comportamiento de componentes individuales

**Escenario 2: Implementación de pruebas de integración**

* **Criterio de Aceptación**: Pruebas de integración implementadas correctamente
* **Contexto**: Fase de desarrollo
* **Evento**: El desarrollador implementa pruebas de integración
* **Resultado**: Interacciones entre componentes verificadas
* **Comportamiento esperado**: Las pruebas verifican el comportamiento del sistema como un todo

**HT-009: Optimización de Rendimiento**

* **Rol**: Desarrollador
* **Característica**: Rendimiento
* **Funcionalidad**: Optimizar rendimiento del sistema
* **Razón**: Para garantizar tiempos de respuesta adecuados
* **Resultado**: Sistema con rendimiento optimizado

**Escenario 1: Optimización de consultas**

* **Criterio de Aceptación**: Consultas optimizadas correctamente
* **Contexto**: Fase de optimización
* **Evento**: El desarrollador optimiza consultas a la base de datos
* **Resultado**: Tiempos de respuesta mejorados
* **Comportamiento esperado**: Las consultas se ejecutan en tiempo aceptable incluso con gran volumen de datos

**Escenario 2: Implementación de caché**

* **Criterio de Aceptación**: Caché implementado correctamente
* **Contexto**: Fase de optimización
* **Evento**: El desarrollador implementa sistema de caché
* **Resultado**: Respuestas más rápidas para datos frecuentes
* **Comportamiento esperado**: Los datos frecuentemente accedidos se sirven desde caché

**HT-010: Implementación de Logs y Monitoreo**

* **Rol**: DevOps
* **Característica**: Operaciones
* **Funcionalidad**: Implementar sistema de logs y monitoreo
* **Razón**: Para detectar y resolver problemas rápidamente
* **Resultado**: Sistema monitoreado con logs centralizados

**Escenario 1: Configuración de logs**

* **Criterio de Aceptación**: Logs configurados correctamente
* **Contexto**: Fase de implementación
* **Evento**: El DevOps configura sistema de logs
* **Resultado**: Logs centralizados y accesibles
* **Comportamiento esperado**: Todas las acciones relevantes se registran con nivel de detalle adecuado

**Escenario 2: Configuración de alertas**

* **Criterio de Aceptación**: Alertas configuradas correctamente
* **Contexto**: Fase de implementación
* **Evento**: El DevOps configura sistema de alertas
* **Resultado**: Alertas enviadas ante problemas
* **Comportamiento esperado**: Los problemas críticos generan alertas inmediatas