Registro Diario de Avances – Sprint 1 Aplicación Web para la gestión de itinerarios

**Fecha:**

**Autor:**

**Versión:** 1.0

# 1. Introducción

Este documento detalla el seguimiento y registro diario del Sprint 1 del proyecto Aplicación Web para la gestión de itinerarios, enfocado en la configuración inicial y desarrollo de la base de datos relacional para la gestión de turistas, usuarios e itinerarios. Se describen las tareas asignadas, su estado, y las estimaciones de esfuerzo realizadas por el equipo de desarrollo, con el objetivo de asegurar la correcta planificación y ejecución del sprint, garantizando la integridad y funcionalidad del sistema desde sus cimientos.

# 2. Lineamientos de Diseño

Para el desarrollo del Sprint 1, se establecieron los siguientes lineamientos de diseño:

* Crear una base de datos relacional eficiente y escalable que soporte la gestión de turistas, usuarios e itinerarios.
* Diseñar un modelo entidad-relación claro que permita definir las relaciones entre las distintas entidades del sistema.
* Implementar claves primarias y foráneas que aseguren la integridad referencial de los datos.
* Desarrollar scripts automatizados para la creación y replicación de la base de datos en diferentes entornos.
* Garantizar la validación y verificación de la integridad de los datos mediante pruebas y revisión continua.
* Documentar cada proceso y diseño para facilitar el mantenimiento y futuras extensiones del sistema.

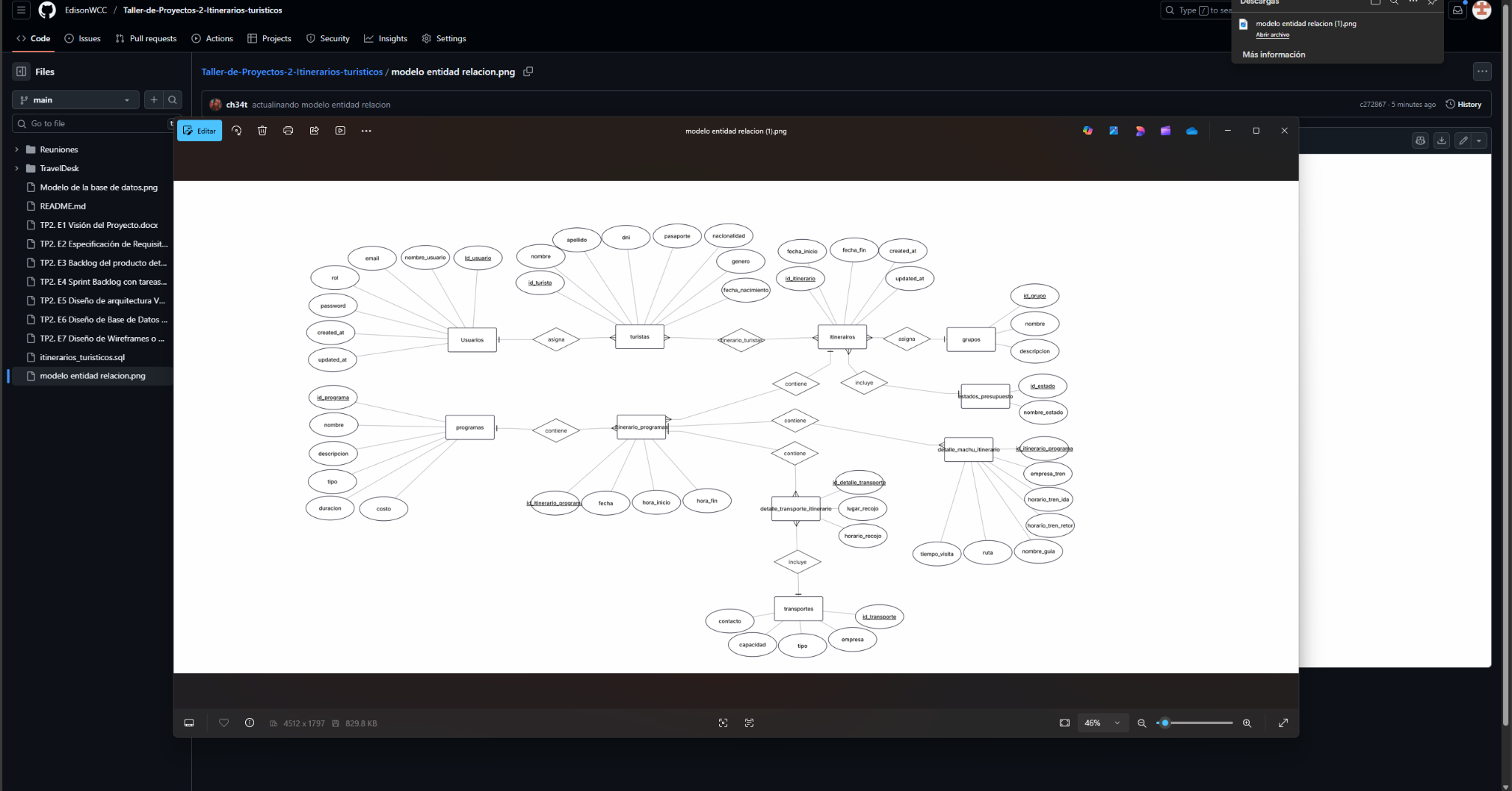
# 3. Historias de Usuario y Tareas

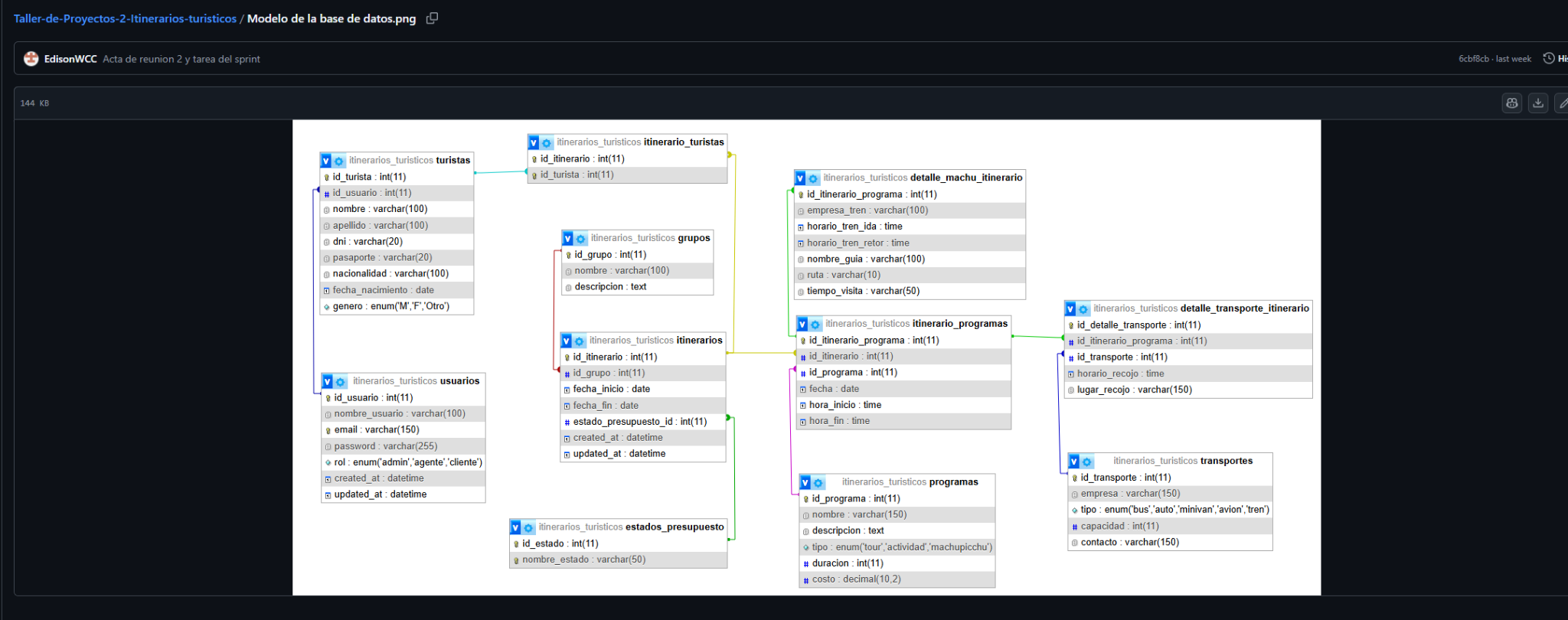
| **Backlog ID** | **Historia de Usuario** | **Tiempo Estimado** | **Fecha de Inicio** | **Fecha de Finalización** |
| --- | --- | --- | --- | --- |
| **HU-1** | Crear base de datos inicial | 24 hrs | 18/09/2025 | 22/09/2025 |
| **HU-2** | Definición de relaciones | 12 hrs | 19/09/2025 | 23/09/2025 |
| **HU-3** | Scripts de creación | 16 hrs | 21/09/2025 | 24/09/2025 |

# 4. Evidencias de Código

## 4.1 Historia de Usuario 1 — Crear base de datos inicial

Archivo: modelo entidad relacion.png/Modelo de la base de datos.png





## 4.2 Historia de Usuario 2 — Definición de relaciones

Archivo: Definición de relaciones.docx

Documentar relaciones y reglas de integridad

Relación Usuarios – Turistas

• Tabla padre: usuarios

• Tabla hija: turistas

• Llave foránea: turistas.id\_usuario → usuarios.id\_usuario

• Cardinalidad: 1 usuario puede estar vinculado a un turista.

• Regla de integridad: No puede existir un turista asociado a un usuario inexistente.

Relación Grupos – Itinerarios

• Tabla padre: grupos

• Tabla hija: itinerarios

• Llave foránea: itinerarios.id\_grupo → grupos.id\_grupo

• Cardinalidad: Un grupo puede tener varios itinerarios.

• Regla de integridad: Un itinerario no puede existir sin un grupo asociado.

Relación Estados\_presupuesto – Itinerarios

• Tabla padre: estados\_presupuesto

• Tabla hija: itinerarios

• Llave foránea: itinerarios.estado\_presupuesto\_id → estados\_presupuesto.id\_estado

• Cardinalidad: Un estado puede aplicarse a muchos itinerarios.

• Regla de integridad: Un itinerario siempre debe tener un estado de presupuesto válido.

Relación Itinerarios – Itinerario\_programas

• Tabla padre: itinerarios

• Tabla hija: itinerario\_programas

• Llave foránea: itinerario\_programas.id\_itinerario → itinerarios.id\_itinerario

• Cardinalidad: Un itinerario puede tener múltiples programas asociados.

Relación Programas – Itinerario\_programas

• Tabla padre: programas

• Tabla hija: itinerario\_programas

• Llave foránea: itinerario\_programas.id\_programa → programas.id\_programa

• Cardinalidad: Un programa puede ser usado en varios itinerarios.

Relación Itinerario\_programas – Detalle\_machu\_itinerario

• Tabla padre: itinerario\_programas

• Tabla hija: detalle\_machu\_itinerario

• Llave foránea: detalle\_machu\_itinerario.id\_itinerario\_programa → itinerario\_programas.id\_itinerario\_programa

• Cardinalidad: Relación 1:1 (cada itinerario de tipo Machu Picchu tiene un único detalle).

Relación Itinerario\_programas – Detalle\_transporte\_itinerario

• Tabla padre: itinerario\_programas

• Tabla hija: detalle\_transporte\_itinerario

• Llave foránea: detalle\_transporte\_itinerario.id\_itinerario\_programa → itinerario\_programas.id\_itinerario\_programa

• Cardinalidad: Un itinerario puede tener varios transportes asociados.

Relación Transportes – Detalle\_transporte\_itinerario

• Tabla padre: transportes

• Tabla hija: detalle\_transporte\_itinerario

• Llave foránea: detalle\_transporte\_itinerario.id\_transporte → transportes.id\_transporte

• Cardinalidad: Un transporte puede ser asignado a múltiples itinerarios.

Relación Itinerarios – Itinerario\_turistas

• Tabla padre: itinerarios

• Tabla hija: itinerario\_turistas

• Llave foránea: itinerario\_turistas.id\_itinerario → itinerarios.id\_itinerario

• Cardinalidad: Relación N:M (un itinerario puede tener varios turistas y un turista puede pertenecer a varios itinerarios).

Relación Turistas – Itinerario\_turistas

• Tabla padre: turistas

• Tabla hija: itinerario\_turistas

• Llave foránea: itinerario\_turistas.id\_turista → turistas.id\_turista

• Cardinalidad: Relación N:M, gestionada con clave primaria compuesta.

Implementar ON DELETE y ON UPDATE

Para asegurar consistencia se recomiendan las siguientes reglas:

• ON DELETE CASCADE en tablas de relación (itinerario\_turistas, itinerario\_programas, detalle\_machu\_itinerario, detalle\_transporte\_itinerario).

Esto asegura que al eliminar un itinerario, programa o turista, se eliminen automáticamente los registros relacionados.

• ON DELETE RESTRICT en entidades principales (usuarios, grupos, programas, transportes).

Esto evita borrar registros base si aún están en uso.

Ejemplo de implementación:

ALTER TABLE itinerario\_turistas

ADD CONSTRAINT fk\_itinerario FOREIGN KEY (id\_itinerario)

REFERENCES itinerarios(id\_itinerario)

ON DELETE CASCADE

ON UPDATE CASCADE;

ALTER TABLE itinerario\_turistas

ADD CONSTRAINT fk\_turista FOREIGN KEY (id\_turista)

REFERENCES turistas(id\_turista)

ON DELETE CASCADE

ON UPDATE CASCADE;

Validar integridad referencial

La integridad referencial garantiza que:

• No se pueden insertar registros en tablas hijas sin que exista la entidad en la tabla padre.

• No quedan registros huérfanos cuando se eliminan entidades principales.

• Se mantienen consistentes las claves primarias y foráneas ante actualizaciones.

Ejemplo de inserción inválida:

-- Error: turista inexistente

INSERT INTO itinerario\_turistas (id\_itinerario, id\_turista)

VALUES (1, 999);

Ejemplo de eliminación en cascada:

-- Eliminar un itinerario elimina automáticamente los turistas vinculados a ese itinerario

DELETE FROM itinerarios WHERE id\_itinerario = 1;

Resumen de relaciones y reglas

Tabla padre Tabla hija Cardinalidad ON DELETE ON UPDATE

usuarios turistas 1:1 RESTRICT CASCADE

grupos itinerarios 1:N RESTRICT CASCADE

estados\_presupuesto itinerarios 1:N RESTRICT CASCADE

itinerarios itinerario\_programas 1:N CASCADE CASCADE

programas itinerario\_programas 1:N RESTRICT CASCADE

itinerario\_programas detalle\_machu\_itinerario 1:1 CASCADE CASCADE

itinerario\_programas detalle\_transporte\_itinerario 1:N CASCADE CASCADE

transportes detalle\_transporte\_itinerario 1:N RESTRICT CASCADE

itinerarios itinerario\_turistas N:M CASCADE CASCADE

turistas itinerario\_turistas N:M CASCADE CASCADE

## 4.3 Historia de Usuario 3 — Scripts de creación

Archivo: itinerarios\_turisticos.sql

```sql

-- Crear base de datos

CREATE DATABASE IF NOT EXISTS itinerarios\_turisticos

DEFAULT CHARACTER SET utf8mb4

COLLATE utf8mb4\_general\_ci;

USE itinerarios\_turisticos;

-- Tabla: usuarios

CREATE TABLE usuarios (

id\_usuario INT(11) NOT NULL AUTO\_INCREMENT,

nombre\_usuario VARCHAR(100) NOT NULL,

email VARCHAR(150) NOT NULL UNIQUE,

password VARCHAR(255) NOT NULL,

rol ENUM('admin','agente','cliente') DEFAULT 'cliente',

created\_at DATETIME DEFAULT CURRENT\_TIMESTAMP,

updated\_at DATETIME DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

PRIMARY KEY (id\_usuario)

) ENGINE=InnoDB;

-- Tabla: grupos

CREATE TABLE grupos (

id\_grupo INT(11) NOT NULL AUTO\_INCREMENT,

nombre VARCHAR(100) NOT NULL,

descripcion TEXT DEFAULT NULL,

PRIMARY KEY (id\_grupo)

) ENGINE=InnoDB;

-- Tabla: estados\_presupuesto

CREATE TABLE estados\_presupuesto (

id\_estado INT(11) NOT NULL AUTO\_INCREMENT,

nombre\_estado VARCHAR(50) NOT NULL UNIQUE,

PRIMARY KEY (id\_estado)

) ENGINE=InnoDB;

-- Tabla: itinerarios

CREATE TABLE itinerarios (

id\_itinerario INT(11) NOT NULL AUTO\_INCREMENT,

id\_grupo INT(11) NOT NULL,

fecha\_inicio DATE NOT NULL,

fecha\_fin DATE NOT NULL,

estado\_presupuesto\_id INT(11) NOT NULL,

created\_at DATETIME DEFAULT CURRENT\_TIMESTAMP,

updated\_at DATETIME DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

PRIMARY KEY (id\_itinerario),

KEY (id\_grupo),

KEY (estado\_presupuesto\_id),

CONSTRAINT itinerarios\_ibfk\_1 FOREIGN KEY (id\_grupo) REFERENCES grupos (id\_grupo),

CONSTRAINT itinerarios\_ibfk\_2 FOREIGN KEY (estado\_presupuesto\_id) REFERENCES estados\_presupuesto (id\_estado)

) ENGINE=InnoDB;

-- Tabla: programas

CREATE TABLE programas (

id\_programa INT(11) NOT NULL AUTO\_INCREMENT,

nombre VARCHAR(150) NOT NULL,

descripcion TEXT DEFAULT NULL,

tipo ENUM('tour','actividad','machupicchu') NOT NULL,

duracion INT(11) DEFAULT NULL,

costo DECIMAL(10,2) NOT NULL,

PRIMARY KEY (id\_programa)

) ENGINE=InnoDB;

-- Tabla: itinerario\_programas

CREATE TABLE itinerario\_programas (

id\_itinerario\_programa INT(11) NOT NULL AUTO\_INCREMENT,

id\_itinerario INT(11) NOT NULL,

id\_programa INT(11) NOT NULL,

fecha DATE NOT NULL,

hora\_inicio TIME DEFAULT NULL,

hora\_fin TIME DEFAULT NULL,

PRIMARY KEY (id\_itinerario\_programa),

KEY (id\_itinerario),

KEY (id\_programa),

CONSTRAINT itinerario\_programas\_ibfk\_1 FOREIGN KEY (id\_itinerario) REFERENCES itinerarios (id\_itinerario),

CONSTRAINT itinerario\_programas\_ibfk\_2 FOREIGN KEY (id\_programa) REFERENCES programas (id\_programa)

) ENGINE=InnoDB;

-- Tabla: detalle\_machu\_itinerario

CREATE TABLE detalle\_machu\_itinerario (

id\_itinerario\_programa INT(11) NOT NULL,

empresa\_tren VARCHAR(100) DEFAULT NULL,

horario\_tren\_ida TIME DEFAULT NULL,

horario\_tren\_retor TIME DEFAULT NULL,

nombre\_guia VARCHAR(100) DEFAULT NULL,

ruta VARCHAR(10) DEFAULT NULL,

tiempo\_visita VARCHAR(50) DEFAULT NULL,

PRIMARY KEY (id\_itinerario\_programa),

CONSTRAINT detalle\_machu\_itinerario\_ibfk\_1 FOREIGN KEY (id\_itinerario\_programa) REFERENCES itinerario\_programas (id\_itinerario\_programa)

) ENGINE=InnoDB;

-- Tabla: transportes

CREATE TABLE transportes (

id\_transporte INT(11) NOT NULL AUTO\_INCREMENT,

empresa VARCHAR(150) NOT NULL,

tipo ENUM('bus','auto','minivan','avion','tren') NOT NULL,

capacidad INT(11) DEFAULT NULL,

contacto VARCHAR(150) DEFAULT NULL,

PRIMARY KEY (id\_transporte)

) ENGINE=InnoDB;

-- Tabla: detalle\_transporte\_itinerario

CREATE TABLE detalle\_transporte\_itinerario (

id\_detalle\_transporte INT(11) NOT NULL AUTO\_INCREMENT,

id\_itinerario\_programa INT(11) NOT NULL,

id\_transporte INT(11) NOT NULL,

horario\_recojo TIME DEFAULT NULL,

lugar\_recojo VARCHAR(150) DEFAULT NULL,

PRIMARY KEY (id\_detalle\_transporte),

KEY (id\_itinerario\_programa),

KEY (id\_transporte),

CONSTRAINT detalle\_transporte\_itinerario\_ibfk\_1 FOREIGN KEY (id\_itinerario\_programa) REFERENCES itinerario\_programas (id\_itinerario\_programa),

CONSTRAINT detalle\_transporte\_itinerario\_ibfk\_2 FOREIGN KEY (id\_transporte) REFERENCES transportes (id\_transporte)

) ENGINE=InnoDB;

-- Tabla: turistas

CREATE TABLE turistas (

id\_turista INT(11) NOT NULL AUTO\_INCREMENT,

id\_usuario INT(11) DEFAULT NULL,

nombre VARCHAR(100) NOT NULL,

apellido VARCHAR(100) NOT NULL,

dni VARCHAR(20) DEFAULT NULL,

pasaporte VARCHAR(20) DEFAULT NULL,

nacionalidad VARCHAR(100) DEFAULT NULL,

fecha\_nacimiento DATE DEFAULT NULL,

genero ENUM('M','F','Otro') DEFAULT NULL,

PRIMARY KEY (id\_turista),

KEY (id\_usuario),

CONSTRAINT turistas\_ibfk\_1 FOREIGN KEY (id\_usuario) REFERENCES usuarios (id\_usuario)

) ENGINE=InnoDB;

-- Tabla: itinerario\_turistas

CREATE TABLE itinerario\_turistas (

id\_itinerario INT(11) NOT NULL,

id\_turista INT(11) NOT NULL,

PRIMARY KEY (id\_itinerario, id\_turista),

KEY (id\_turista),

CONSTRAINT itinerario\_turistas\_ibfk\_1 FOREIGN KEY (id\_itinerario) REFERENCES itinerarios (id\_itinerario),

CONSTRAINT itinerario\_turistas\_ibfk\_2 FOREIGN KEY (id\_turista) REFERENCES turistas (id\_turista)

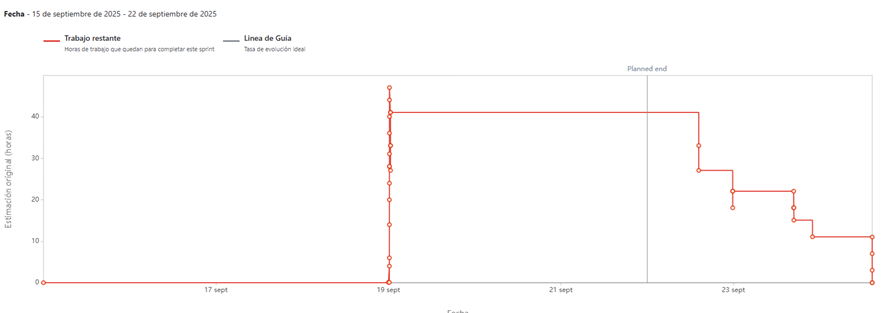
) ENGINE=InnoDB;

```

# 5. Burndown Chart

| **Backlog ID** | **Historia de Usuario** | **Tiempo Estimado** | **18/09/2025** | **19/09/2025** | **20/09/2025** | **21/09/2025** | **22/09/2025** | **23/09/2025** | **24/09/2025** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Dia1** | **Dia2** | **Dia3** | **Dia4** | **Dia5** | **Dia6** | **Dia7** |
| **HU-1** | Crear base de datos inicial | 24 | 6 |  |  |  | 14 | 4 |  |
| **HU-2** | Definición de relaciones | 12 | 4 |  |  |  |  |  | 8 |
| **HU-3** | Scripts de creación | 16 |  |  |  |  | 8 |  | 8 |
| **Tiempo de trabajo** | | 52 | 10 | 0 | 0 | 0 | 22 | 4 | 16 |

Burndown Chart



# 6. Análisis del Sprint

* El sprint comenzó con un esfuerzo total estimado de 52 horas para las historias de usuario asignadas.
* En los primeros días (18 y 19 de septiembre), se avanzó inicialmente en las tareas de mayor complejidad, como la creación de la base de datos inicial y la definición de relaciones.
* Hubo una pausa en el trabajo activo durante los días 20, 21 y 22 de septiembre, lo que podría indicar días de revisión, bloqueos o planificación.
* Posteriormente, se reanudó el trabajo con un enfoque significativo en la creación de la base de datos y en los scripts de creación, completando progresivamente las tareas asignadas.
* La distribución del trabajo fue irregular, pero se logró completar las tareas dentro del plazo establecido, reflejando una buena capacidad de adaptación del equipo.
* Se recomienda revisar los motivos de la pausa en medio del sprint para optimizar la continuidad del trabajo en futuros sprints.
* En general, el sprint fue exitoso al cumplir con los objetivos establecidos, manteniendo la calidad y la integridad del diseño de la base de datos.

# 7. Conclusiones

La pausa observada durante los días intermedios no afectó la entrega final, pero su análisis permitirá mejorar la continuidad y eficiencia en futuros sprints.

La documentación y definición de relaciones, junto con la creación y prueba de scripts, sentaron bases sólidas para el desarrollo posterior de funcionalidades del sistema.

Para siguientes sprints, se recomienda implementar mecanismos para minimizar interrupciones y mejorar la fluidez del trabajo diario.