PrestAula

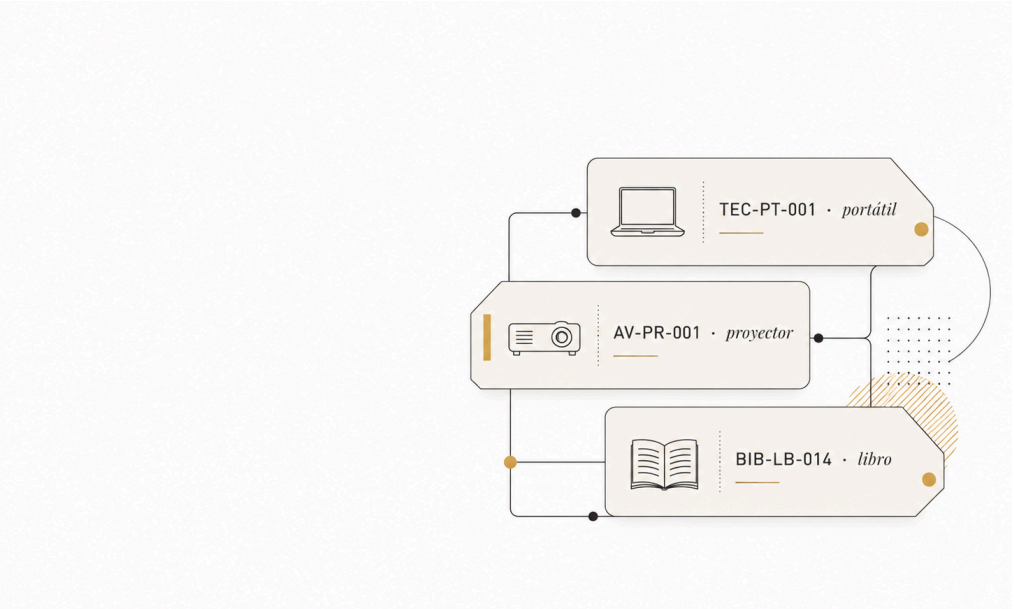
Diseño e integración de una base de datos SQLite en una aplicación web con Bun y TypeScript

Guión didáctico para la UF1845

Acceso a datos en aplicaciones web del entorno servidor

Certificado profesional IFCD0210 — Desarrollo de aplicaciones con tecnologías web
Módulo formativo MF0492_3 — Programación web en el entorno servidor
Unidad formativa UF1845 — Acceso a datos en aplicaciones web del entorno servidor
Modalidad Presencial **Nivel** 3

Nombre y apellidos: [Samuel López López]
Curso: SSCE0110 — Habilitación para la docencia en grados A, B y C
Fecha: [08/07/2026]



Índice

Recorrido de aprendizaje

01 Entender Identificar el problema: una web que muestra datos, pero no los conserva.

02 Diseñar Decidir qué información necesita PrestAula y cómo se relaciona.

03 Guardar Crear una base de datos SQLite con recursos, personas y préstamos.

04 Conectar Vincular la aplicación web (Bun y TypeScript) a SQLite.

05 Gestionar Crear, consultar, editar y controlar préstamos, devoluciones y errores.

06 Comprobar Probar la aplicación y verificar que responde a situaciones reales.

01 Contexto de la propuesta

De una web temporal a una aplicación que recuerda

Identificación

Elemento	Información
Certificado profesional	IFCD0210 · Desarrollo de aplicaciones con tecnologías web
Módulo formativo	MF0492_3 · Programación web en el entorno servidor
Unidad formativa	UF1845 · Acceso a datos en aplicaciones web del entorno servidor
Nivel	3
Modalidad	Presencial
Duración de la unidad	90 horas

Punto de partida

En la unidad formativa anterior, el alumnado ha desarrollado una aplicación web básica capaz de mostrar información y recoger datos mediante formularios.

Sin embargo, esa información todavía no se conserva cuando la aplicación se reinicia. Los recursos, las personas usuarias y los préstamos aparecen solo como datos de ejemplo escritos directamente en el código.

La pregunta que guía esta propuesta es sencilla:

> **¿Cómo conseguimos que una aplicación web guarde, recupere y actualice información de forma organizada?**

El caso práctico: PrestAula

PrestAula es una aplicación web para gestionar el préstamo de recursos de un centro de formación.

El sistema permitirá registrar y consultar recursos como portátiles, proyectores, cámaras, libros, tabletas, adaptadores o auriculares; identificar a las personas usuarias; y controlar los préstamos, devoluciones y la disponibilidad de cada recurso.

Durante la unidad, el alumnado diseñará una base de datos SQLite y la conectará con una aplicación web desarrollada con Bun y TypeScript.

Antes y después

Antes: web temporal	Después: PrestAula conectada
Los datos están escritos directamente en el código.	Los datos se guardan en una base de datos SQLite.
La información desaparece al reiniciar la aplicación.	La información se conserva y puede recuperarse después.

Antes: web temporal	Después: PrestAula conectada
Solo se muestran recursos de ejemplo.	Se gestionan recursos, personas usuarias y préstamos reales.
Los formularios no guardan cambios permanentes.	Los formularios crean, modifican y consultan información.
La disponibilidad se controla manualmente.	La aplicación calcula qué recursos están disponibles o prestados.

> **IDEA CLAVE**
>
> Una aplicación útil no solo muestra datos: también debe poder guardarlos, organizarlos y recuperarlos cuando sea necesario.

02 Presentación y objetivos de aprendizaje

Aprender a convertir datos en una aplicación útil

PrestAula parte de una aplicación web sencilla desarrollada previamente. A lo largo de esta propuesta, el alumnado aprenderá a transformar esa web inicial en una aplicación capaz de conservar información, consultarla cuando sea necesaria y evitar errores habituales durante la gestión de recursos y préstamos.

El objetivo no es memorizar comandos aislados, sino comprender cómo una necesidad real —prestar un proyector, registrar una devolución o localizar un recurso— se traduce en datos, tablas, consultas y acciones dentro de una aplicación web.

01 · Comprender y diseñar

1 Identificar la información que necesita almacenar una aplicación de gestión de préstamos de recursos en un centro de formación.

2 Diseñar un modelo de datos básico para PrestAula, diferenciando categorías, recursos, personas usuarias y préstamos, así como las relaciones entre ellos.

02 · Crear y conectar

3 Crear una base de datos SQLite con tablas relacionadas, aplicando claves principales, claves foráneas y restricciones básicas de integridad.

4 Realizar consultas SQL sencillas para registrar, consultar, modificar y eliminar información relacionada con recursos, personas usuarias y préstamos.

5 Conectar una aplicación web desarrollada con Bun y TypeScript a una base de datos SQLite para mostrar y gestionar información persistente.

03 · Aplicar y comprobar

6 Implementar operaciones CRUD en PrestAula, incorporando formularios, validaciones y mensajes de error comprensibles.

7 Aplicar reglas básicas de funcionamiento, como impedir préstamos duplicados, controlar las devoluciones y conservar el historial de los recursos del centro.

IDEA CLAVE

Aprender a acceder a datos no consiste solo en guardar información: consiste en crear aplicaciones capaces de responder de forma ordenada a situaciones reales.

03 PrestAula necesita recordar

¿Qué ocurre cuando una aplicación guarda información?

Una web puede mostrar datos, recibir información mediante formularios y responder a las acciones de una persona usuaria. Sin embargo, si esos datos solo existen mientras la aplicación está abierta, desaparecen cuando se reinicia.

PrestAula necesita conservar la información de los recursos del centro, las personas que los solicitan y los préstamos realizados. Para conseguirlo, la aplicación utilizará una base de datos SQLite.

Datos temporales y datos persistentes

Datos temporales	Datos persistentes
Solo existen mientras la aplicación está funcionando.	Se conservan aunque la aplicación se cierre o reinicie.
Pueden estar escritos directamente en el código.	Se guardan en una base de datos organizada.
Sirven para hacer pruebas rápidas.	Permiten gestionar información real a lo largo del tiempo.
No conservan el historial de cambios.	Permiten consultar recursos, préstamos y devoluciones anteriores.

EJEMPLO

María López registra el préstamo de un proyector:

1. Selecciona el recurso y añade la fecha prevista de devolución.
2. PrestAula comprueba que el proyector está disponible.
3. La aplicación guarda el préstamo en SQLite.
4. El proyector pasa a mostrarse como **prestado**.
5. El historial conserva quién lo solicitó y cuándo debe devolverlo.

El recorrido de un dato en PrestAula

¿Qué ocurre cuando una aplicación guarda información?

IDEA CLAVE

La base de datos permite que PrestAula recuerde la información y la use cuando vuelva a necesitarla.

Vocabulario básico

- **Dato:** información concreta, como el nombre de un recurso o una fecha.
- **Base de datos:** sistema organizado para guardar información.
- **Persistencia:** capacidad de conservar los datos aunque la aplicación se cierre.
- **Formulario:** pantalla que permite introducir o modificar información.

04 La información que gestiona PrestAula

Cuatro tipos de información, una sola necesidad

Para gestionar préstamos de forma organizada, PrestAula no guarda toda la información en una única lista. Cada tipo de dato tiene su propio espacio y se relaciona con los demás cuando es necesario.

01 · Categorías

Sirven para agrupar los recursos del centro y facilitar su localización.

Ejemplos de categoría
Tecnología
Material audiovisual
Biblioteca
Material de aula

Las categorías se crean como datos iniciales y se seleccionan al registrar cada recurso.

02 · Recursos

Son los objetos concretos que el centro puede prestar.

Recurso	Código	Ubicación habitual
Portátil Lenovo ThinkPad 01	`TEC-PT-001`	Aula de informática 1
Proyector Epson EB-X06	`AV-PR-001`	Almacén audiovisual
Cámara Canon EOS 2000D	`AV-CA-002`	Aula polivalente 1
Libro *HTML y CSS*	`BIB-LB-014`	Biblioteca / sala de estudio

Cada recurso tiene un código de inventario único, una categoría, una ubicación habitual y un estado interno.

03 · Personas usuarias

Son las personas que pueden solicitar recursos del centro.

Dato	Ejemplo
Nombre y apellidos	María López García
Correo electrónico	`maria.lopez@centro.es`
Tipo de persona	Docente
Estado	**Activa**

Una persona inactiva conserva su historial, pero no puede solicitar nuevos préstamos.

04 · Préstamos

Un préstamo conecta a una persona concreta con un recurso concreto durante un periodo de tiempo.

Dato	Ejemplo
Recurso	Proyector Epson EB-X06
Persona usuaria	María López García
Fecha de préstamo	12/03/2027
Fecha prevista de devolución	14/03/2027
Estado	**Activo**

EJEMPLO

María López solicita el proyector `AV-PR-001` para impartir una sesión formativa. PrestAula consulta los datos del recurso, comprueba que está disponible y crea un préstamo con las fechas acordadas.

IDEA CLAVE

Separar la información permite que PrestAula encuentre, actualice y relacione los datos sin duplicarlos ni generar contradicciones.

Cuatro tipos de información

Cada tipo de dato tiene su propio espacio y se relaciona con los demás.

01 · Categorías

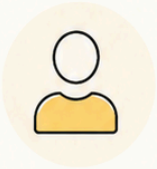
Agrupan y organizan los recursos.

- Tecnología
- Material audiovisual
- Biblioteca
- Material de aula

02 · Recursos

Objetos concretos que se pueden prestar.

Portátil Lenovo 01	TEC-PT-001	Aula informática 1
Proyector Epson	AV-PR-001	Almacén audiovisual
Cámara Canon	AV-CA-002	Aula polivalente 1
Libro HTML y CSS	BIB-LB-014	Biblioteca



03 · Personas usuarias

Personas que pueden solicitar recursos.

Nombre	María López García
Correo	maria.lopez@centro.es
Tipo	Docente
Estado	Activa

04 · Préstamos

Conectan una persona con un recurso.

Recurso	Proyector Epson
Persona	María López García
Préstamo	12/03/2027
Devolución	14/03/2027
Estado	Activo

05 Organizar la información antes de programar

El modelo de datos de PrestAula

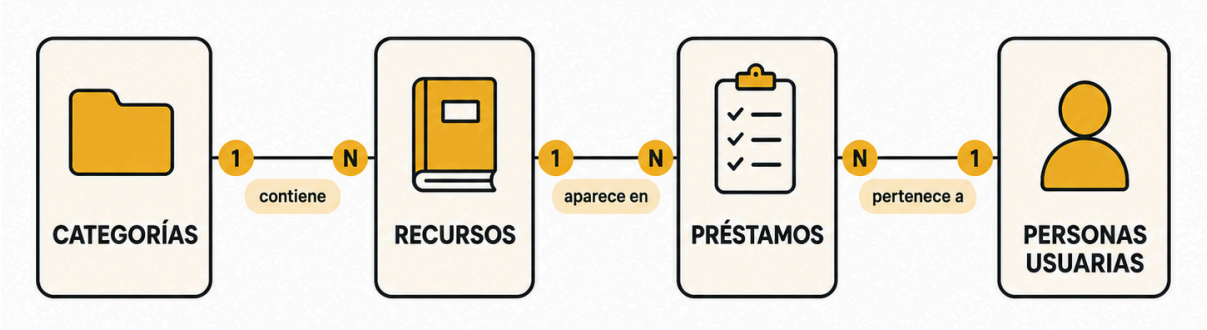
Antes de crear tablas o escribir código, es necesario decidir cómo se organizará la información. Este diseño previo se denomina **modelo de datos**.

El modelo permite responder a preguntas como:

- ¿Qué información debe guardar la aplicación?
- ¿Qué datos pertenecen a cada recurso?
- ¿Cómo se relaciona un préstamo con una persona y un material?
- ¿Qué información no debe repetirse?

En PrestAula, la información se organiza en cuatro tablas relacionadas: categorías, recursos, personas usuarias y préstamos.

Diagrama entidad-relación



Lectura del modelo

- Una **categoría** puede incluir varios recursos. Por ejemplo, *Material audiovisual* incluye proyectores, cámaras y auriculares.
- Un **recurso** puede aparecer en muchos préstamos a lo largo del tiempo.
- Una **persona usuaria** puede tener varios préstamos registrados.
- Cada **préstamo** une a una persona concreta con un recurso concreto.

Dos conceptos importantes

Concepto	Explicación sencilla
Clave principal	Dato que identifica de forma única cada registro. Por ejemplo, el identificador de un recurso.
Clave foránea	Dato que conecta una tabla con otra. Por ejemplo, el préstamo guarda qué recurso y qué persona están relacionados.

EJEMPLO

María López solicita el proyector `AV-PR-001`. PrestAula no necesita volver a escribir todos los datos de María ni del proyector: crea un préstamo que conecta ambos registros.

IDEA CLAVE

Diseñar bien las relaciones evita duplicar información y ayuda a que la aplicación mantenga los datos organizados.

06 De la idea a SQLite

Una base de datos dentro de PrestAula

Hasta ahora hemos decidido qué información necesita guardar PrestAula y cómo se relacionan los datos. El siguiente paso es crear un lugar donde esa información pueda conservarse.

Para este proyecto utilizaremos **SQLite**.

SQLite es una base de datos ligera que se guarda en un único archivo. No necesita instalar ni administrar un servidor independiente, por lo que resulta adecuada para aprender cómo funciona el acceso a datos y para desarrollar una primera versión funcional de PrestAula.

> IDEA CLAVE
> SQLite será el archivo que permitirá a PrestAula recordar los recursos, las personas usuarias y los préstamos registrados.

De una tabla visual a una tabla real

La tabla `recursos` representa cada objeto concreto que puede prestarse en el centro.

Campo	Para qué sirve
`id`	Identifica cada recurso de forma única.
`codigo_inventario`	Permite localizar el recurso mediante un código visible, como AV-PR-001.
`nombre`	Indica qué recurso es.
`categoria_id`	Relaciona el recurso con una categoría existente.
`ubicacion`	Indica dónde se guarda habitualmente.
`estado`	Señala si está activo, en mantenimiento o dado de baja.
`fecha_alta`	Registra cuándo se incorporó al centro.

> IDEA CLAVE
> **En lenguaje normal:** «Queremos guardar un proyector con un código único, indicar que pertenece al material audiovisual y saber que su ubicación habitual es el almacén audiovisual.»

En código SQL

```sql
CREATE TABLE recursos (
  id INTEGER PRIMARY KEY,
  codigo_inventario TEXT NOT NULL UNIQUE,
  nombre TEXT NOT NULL,
  categoria_id INTEGER NOT NULL,
  ubicacion TEXT NOT NULL,
  estado TEXT NOT NULL
    CHECK (estado IN ('activo', 'mantenimiento', 'baja')),
  fecha_alta TEXT NOT NULL,

  FOREIGN KEY (categoria_id) REFERENCES categorias(id)
);
```

¿Qué está haciendo este código?

- `CREATE TABLE recursos` crea una tabla llamada recursos.
- `INTEGER PRIMARY KEY` genera un identificador único para cada recurso.
- `NOT NULL` indica que ese dato es obligatorio.
- `UNIQUE` evita que dos recursos compartan el mismo código de inventario.
- `FOREIGN KEY` conecta cada recurso con su categoría.
- `CHECK` limita el estado a las opciones permitidas.

RECUERDA

El código no se escribe para memorizarlo de una vez. Primero se decide qué necesita guardar la aplicación y después se traduce esa decisión a una tabla.

07 Consultar y modificar datos con SQL

SQL: el lenguaje para trabajar con la información

Una vez creada la base de datos, PrestAula necesita poder consultar y modificar la información que guarda. Para ello se utiliza **SQL**, un lenguaje que permite indicar a SQLite qué datos queremos obtener o qué cambios queremos realizar.

Las operaciones principales forman el acrónimo **CRUD**:

Acción	En PrestAula	Instrucción SQL
Crear	Registrar un nuevo recurso	`INSERT`
Consultar	Mostrar recursos disponibles	`SELECT`
Modificar	Cambiar la ubicación de un recurso	`UPDATE`
Eliminar	Eliminar un dato de prueba o gestionar una baja	`DELETE`

> IDEA CLAVE
>
> **En lenguaje normal:** «Queremos mostrar solo los recursos activos que no tengan un préstamo pendiente.» PrestAula consulta los recursos, comprueba su estado y revisa si existe algún préstamo activo asociado.

En código SQL

SQL

```sql
-- Consultar los recursos disponibles
SELECT
  r.codigo_inventario,
  r.nombre,
  c.nombre AS categoria,
  r.ubicacion
FROM recursos r
JOIN categorias c ON c.id = r.categoria_id
LEFT JOIN prestamos p
  ON p.recurso_id = r.id AND p.estado = 'activo'
WHERE r.estado = 'activo' AND p.id IS NULL
ORDER BY r.nombre;

-- Registrar un recurso nuevo
INSERT INTO recursos (
  codigo_inventario, nombre, categoria_id,
  ubicacion, estado, fecha_alta
)
VALUES ('AV-PR-001', 'Proyector Epson EB-X06', 2,
  'Almacén audiovisual', 'activo', '2027-03-12');
```

Resultado de la consulta

Código	Recurso	Categoría	Ubicación
AV-PR-001	Proyector Epson EB-X06	Material audiovisual	Almacén audiovisual
TEC-PT-001	Portátil Lenovo ThinkPad 01	Tecnología	Aula de informática 1

IDEA CLAVE

SQL permite consultar y actualizar la información de forma precisa. Cada acción visible en PrestAula —registrar, buscar, editar o eliminar— se traduce en una operación sobre la base de datos.

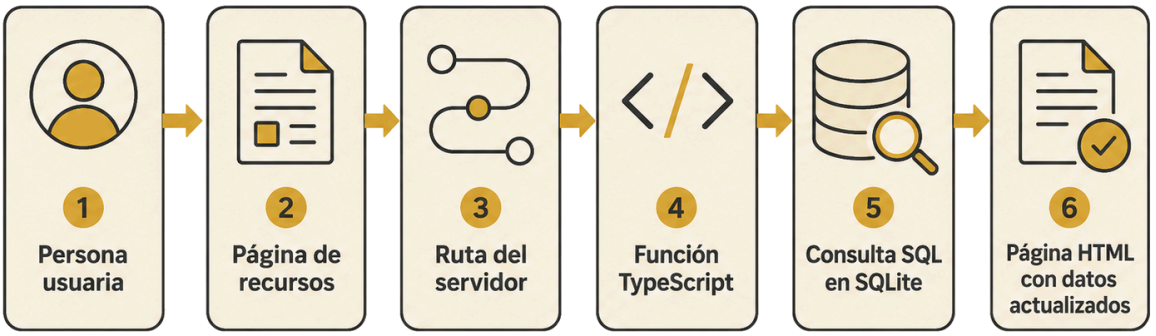
08 Conectar PrestAula con SQLite

De la base de datos a la página web

SQLite ya puede guardar información, pero todavía necesitamos que PrestAula sea capaz de utilizarla dentro de la aplicación web. En este proyecto, cada tecnología tiene una función clara:

Tecnología	Función dentro de PrestAula
HTML	Muestra formularios, tablas y mensajes a la persona usuaria.
Bun	Ejecuta la aplicación en el servidor.
TypeScript	Organiza la lógica de la aplicación.
SQLite	Guarda los recursos, personas usuarias y préstamos.

El recorrido de una consulta



IDEA CLAVE

En lenguaje normal: «Cuando una persona abre la página de recursos, PrestAula debe consultar SQLite y mostrar los recursos disponibles.»

En código TypeScript

TypeScript

```typescript
import { Database } from "bun:sqlite"

const db = new Database("prestaula.sqlite")

export function obtenerRecursosDisponibles() {
  return db.query(`
    SELECT r.codigo_inventario, r.nombre, r.ubicacion
    FROM recursos r
    LEFT JOIN prestamos p
      ON p.recurso_id = r.id AND p.estado = 'activo'
```

```
    WHERE r.estado = 'activo' AND p.id IS NULL
    ORDER BY r.nombre
  `).all()
}
```

¿Qué hace este código?

1. Abre el archivo de base de datos de PrestAula.
2. Ejecuta una consulta SQL para localizar recursos disponibles.
3. Devuelve los resultados a la ruta de la aplicación.
4. La página HTML utiliza esos datos para crear el listado que verá la persona usuaria.

> **RECUERDA**
> La aplicación web no necesita inventar los datos cada vez que carga una página: los consulta en SQLite y los muestra cuando son necesarios.

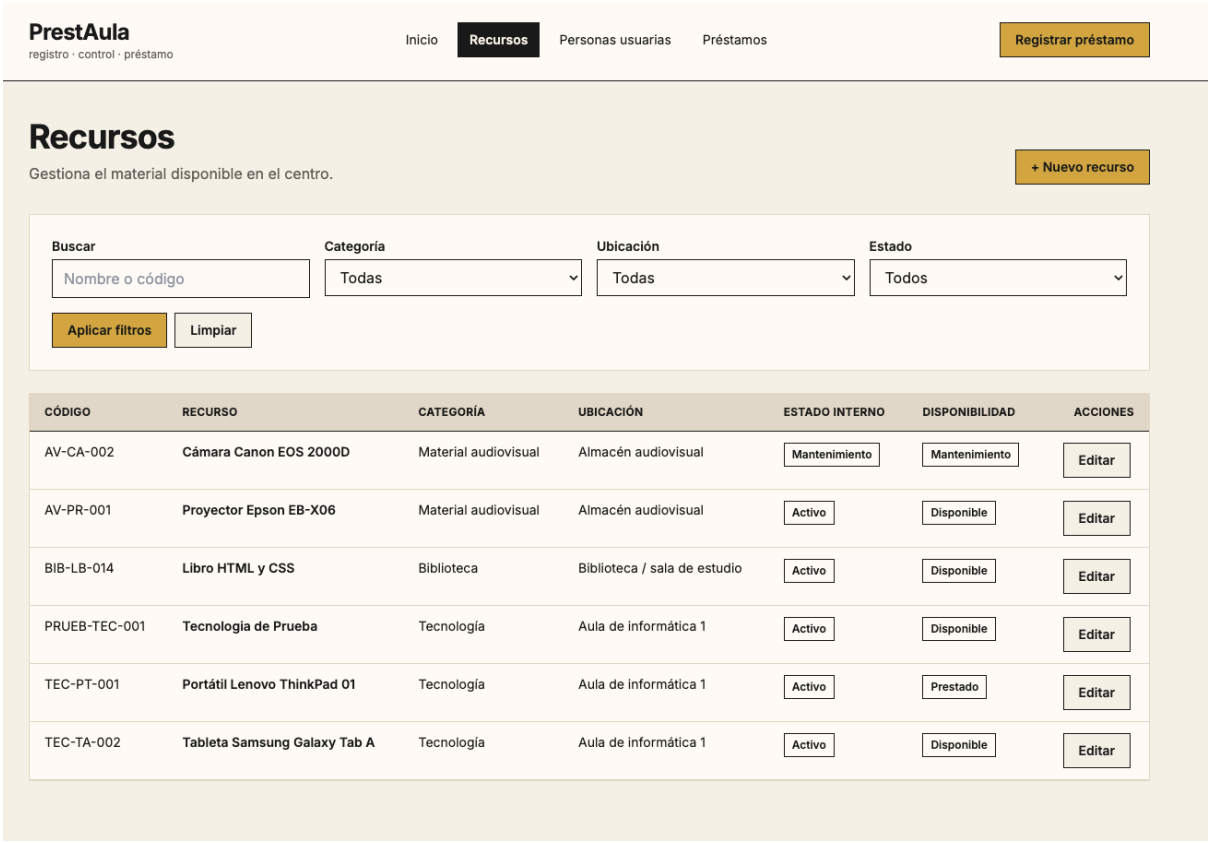
09 CRUD de recursos

Gestionar el inventario del centro

El primer CRUD completo de PrestAula se aplica a los recursos. El alumnado empieza por una información fácil de identificar —un portátil, un proyector o un libro— antes de trabajar con préstamos, que requieren relacionar varios datos.

Acción en la aplicación	Operación	Ejemplo
Ver el listado	Consultar	Mostrar los recursos del centro.
Nuevo recurso	Crear	Registrar una cámara recién incorporada.
Editar	Modificar	Cambiar la ubicación habitual de un proyector.
Dar de baja	Actualizar / eliminar	Marcar un recurso como no disponible.

Pantalla principal de recursos



IDEA CLAVE

En lenguaje normal: «Al guardar un recurso, PrestAula debe comprobar que el código de inventario no exista ya y, después, registrar los datos en SQLite.»

En código TypeScript

TypeScript

```typescript
type NuevoRecurso = {
  codigoInventario: string
  nombre: string
  categoriaId: number
  ubicacion: string
  estado: "activo" | "mantenimiento" | "baja"
  fechaAlta: string
}

export function crearRecurso(recurso: NuevoRecurso) {
  const existe = db
    .query("SELECT 1 FROM recursos WHERE codigo_inventario = ?")
    .get(recurso.codigoInventario)

  if (existe) {
    throw new Error("Ya existe un recurso con este código.")
  }

  db.query(`
    INSERT INTO recursos (
      codigo_inventario, nombre, categoria_id,
      ubicacion, estado, fecha_alta
    ) VALUES (?, ?, ?, ?, ?, ?)
  `).run(
    recurso.codigoInventario, recurso.nombre,
    recurso.categoriaId, recurso.ubicacion,
    recurso.estado, recurso.fechaAlta
  )
}
```

IDEA CLAVE

El CRUD no consiste únicamente en añadir y borrar datos. En una aplicación real, PrestAula debe proteger el historial: un recurso con préstamos asociados se da de baja o se marca como inactivo antes de eliminarse.

10 CRUD de personas usuarias

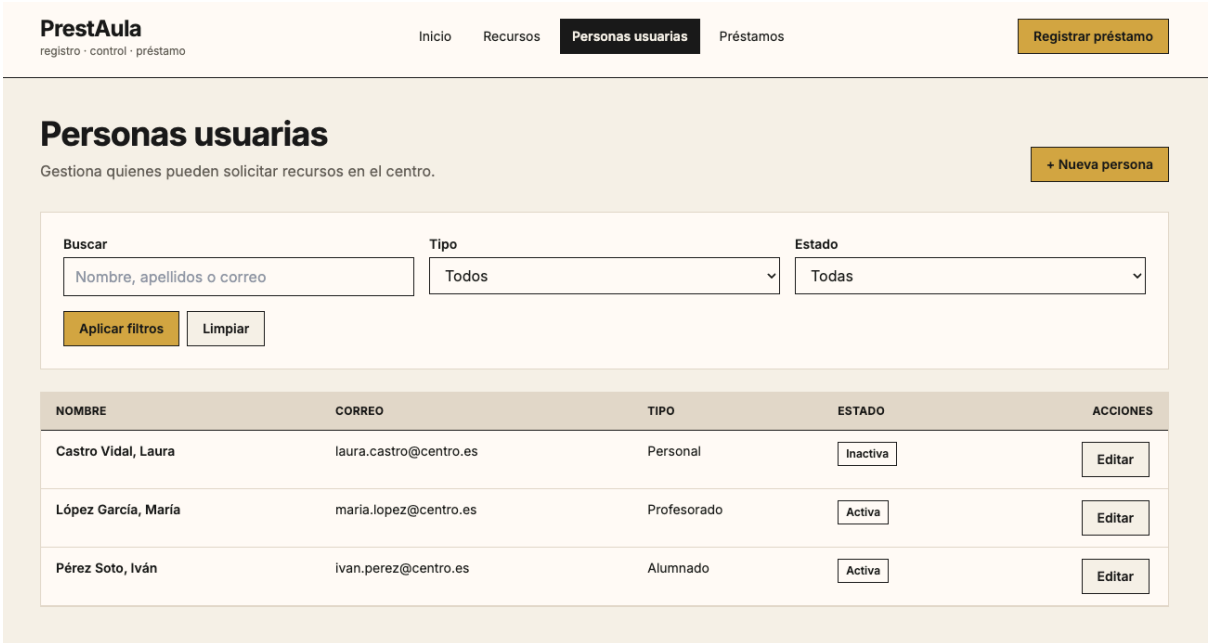
Gestionar quién puede solicitar recursos

Después de trabajar con los recursos del centro, PrestAula aplica el mismo patrón a las personas usuarias. Esta parte permite consolidar las operaciones CRUD antes de introducir los préstamos, que requieren relacionar información de varias tablas.

Cada persona usuaria dispone de una ficha con sus datos de contacto, su tipo de relación con el centro y su estado de actividad.

Acción en la aplicación	Ejemplo en PrestAula
Consultar	Ver el listado de alumnado, profesorado y personal del centro.
Crear	Registrar a una nueva docente.
Editar	Corregir un correo electrónico o un teléfono.
Desactivar	Impedir nuevos préstamos sin eliminar el historial.
Eliminar	Solo cuando no existen préstamos asociados.

Pantalla de personas usuarias



IDEA CLAVE

En lenguaje normal: «Antes de registrar una persona, PrestAula debe comprobar que tiene nombre, correo y tipo. Además, no puede haber dos personas con el mismo correo.»

En código TypeScript

TypeScript

```typescript
type NuevaPersona = {
  nombre: string
  apellidos: string
  email: string
  telefono?: string
  tipo: "alumnado" | "profesorado" | "personal"
}

export function crearPersona(persona: NuevaPersona) {
  const existe = db
    .query("SELECT 1 FROM personas_usuarias WHERE email = ?")
    .get(persona.email)

  if (existe) {
    throw new Error("Este correo ya está registrado.")
  }

  db.query(`
    INSERT INTO personas_usuarias (
      nombre, apellidos, email, telefono, tipo, activa
    ) VALUES (?, ?, ?, ?, ?, 1)
  `).run(
    persona.nombre, persona.apellidos, persona.email,
    persona.telefono ?? null, persona.tipo
  )
}
```

RECUERDA

Desactivar una persona no elimina su información. Conserva su historial de préstamos, pero evita que pueda solicitar nuevos recursos.

11 Registrar préstamos

Un préstamo conecta personas y recursos

Un préstamo es el registro que relaciona a una persona usuaria con un recurso concreto durante un periodo de tiempo.

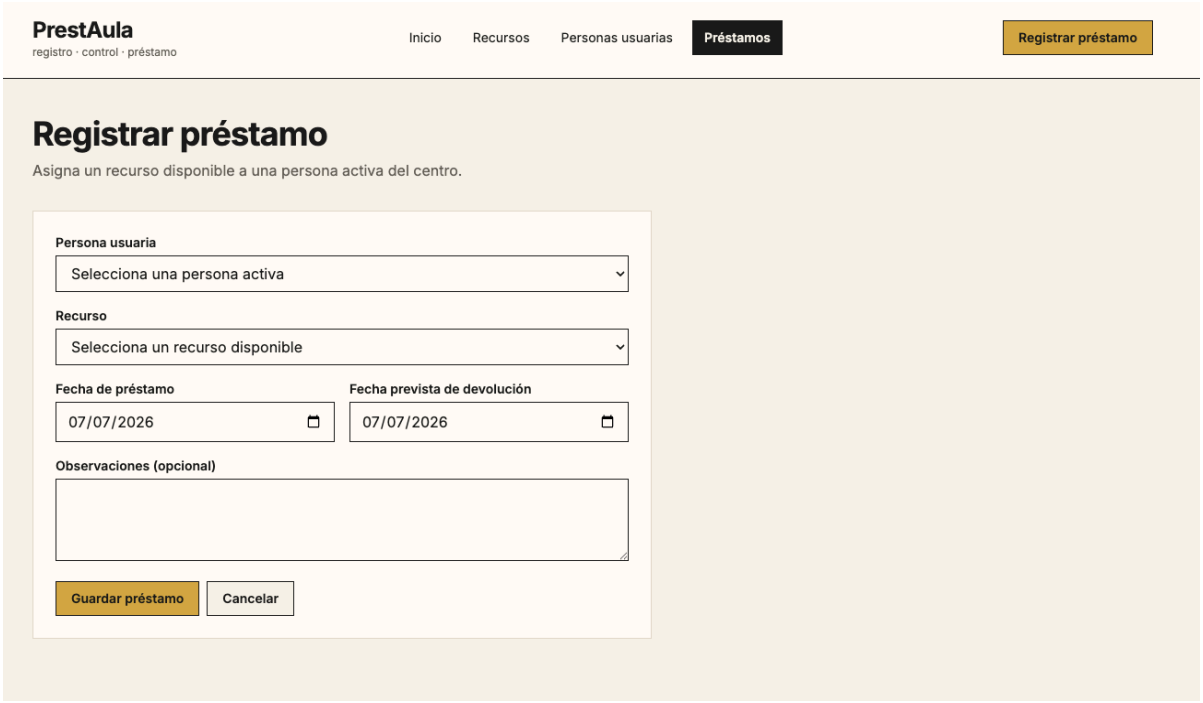
PrestAula no crea recursos ni personas nuevas al registrar un préstamo. Utiliza datos que ya existen en la base de datos y comprueba que ambos cumplen las condiciones necesarias.

PrestAula utiliza datos ya existentes y crea el registro del préstamo.

Información necesaria

Dato	Ejemplo
Persona usuaria	María López García
Recurso	Proyector Epson EB-X06
Fecha de préstamo	12/03/2027
Fecha prevista de devolución	14/03/2027
Observaciones	Incluye mando a distancia y cable HDMI.

Formulario de préstamo



IDEA CLAVE

En lenguaje normal: «PrestAula debe comprobar que la persona está activa, que el recurso está disponible y que la fecha prevista de devolución no es anterior a la de préstamo.»

En código TypeScript

TypeScript

```typescript
export function crearPrestamo(datos: {
  recursoId: number
  personaId: number
  fechaPrestamo: string
  fechaPrevista: string
  observaciones?: string
}) {
  if (datos.fechaPrevista < datos.fechaPrestamo) {
    throw new Error(
      "La fecha prevista no puede ser anterior."
    )
  }

  const recursoDisponible = db.query(`
    SELECT r.id FROM recursos r
    LEFT JOIN prestamos p
      ON p.recurso_id = r.id AND p.estado = 'activo'
    WHERE r.id = ? AND r.estado = 'activo' AND p.id IS NULL
  `).get(datos.recursoId)
```

```
20
21    if (!recursoDisponible) {
22      throw new Error("El recurso no está disponible.")
23    }
24
25    db.query(`
26      INSERT INTO prestamos (
27        recurso_id, persona_id, fecha_prestamo,
28        fecha_prevista_devolucion, estado, observaciones
29      ) VALUES (?, ?, ?, ?, 'activo', ?)
30    `).run(
31      datos.recursoId, datos.personaId, datos.fechaPrestamo,
32      datos.fechaPrevista, datos.observaciones ?? null
33    )
34  }
```

IDEA CLAVE

Un préstamo no duplica datos. Conecta una persona y un recurso que ya están registrados en PrestAula.

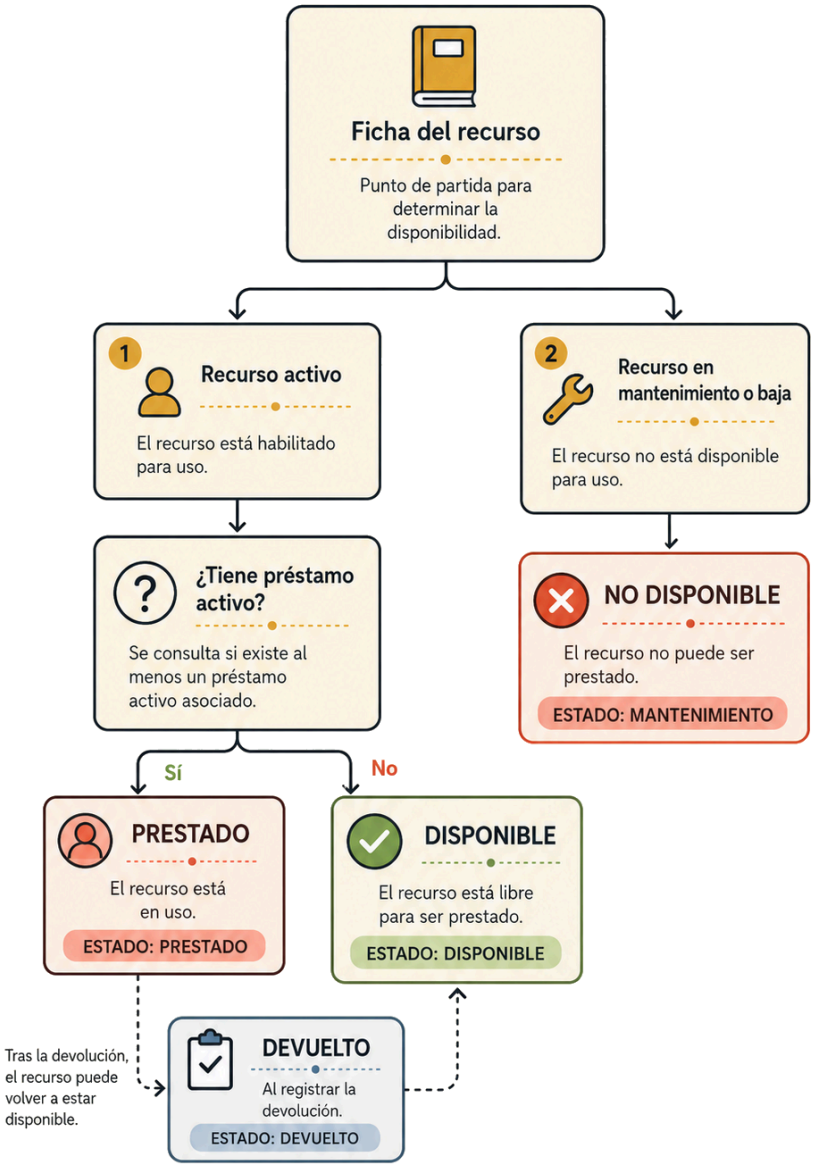
12 Devoluciones, disponibilidad y reglas

Cerrar el préstamo sin perder el historial

Cuando un recurso regresa al centro, PrestAula no elimina el préstamo. Registra la fecha real de devolución y cambia su estado a `devuelto`.

De esta forma, el historial se conserva y el recurso vuelve a poder aparecer como disponible para otra persona.

Cómo se calcula la disponibilidad



La disponibilidad no se escribe manualmente dentro de la ficha del recurso. PrestAula la calcula consultando los préstamos activos asociados.

Registrar una devolución

```typescript
export function registrarDevolucion(
  prestamoId: number,
  fechaDevolucion: string
) {
  const resultado = db.query(`
    UPDATE prestamos
    SET estado = 'devuelto', fecha_devolucion_real = ?
    WHERE id = ? AND estado = 'activo'
  `).run(fechaDevolucion, prestamoId)

  if (resultado.changes === 0) {
    throw new Error("No se ha podido registrar la devolución.")
  }
}
```

Después de esta acción:

5. El préstamo queda registrado como devuelto.
6. Se conserva la fecha real de devolución.
7. El recurso vuelve a aparecer como disponible.
8. El historial permite saber quién lo utilizó y cuándo.

Reglas que protege PrestAula

Situación	Respuesta de la aplicación
Recurso ya prestado	No permite crear un segundo préstamo activo.
Persona inactiva	No permite registrar nuevos préstamos.
Fecha incorrecta	Muestra un mensaje antes de guardar.
Recurso en mantenimiento	Lo excluye de los recursos disponibles.
Recurso con historial	Recomienda darlo de baja antes que eliminarlo.

RECUERDA

Resumen parcial

SQLite permite conservar la información de PrestAula.

Los recursos, las personas y los préstamos se organizan en tablas relacionadas.

Las operaciones CRUD permiten crear, consultar, modificar y controlar los datos.

Las validaciones evitan errores antes de que afecten a la información del centro.

IDEA CLAVE

Una aplicación bien diseñada no solo registra acciones: también protege los datos y ayuda a tomar decisiones correctas.

13 Actividades y metodología didáctica

Aprender construyendo PrestAula

Las actividades se organizan de forma progresiva. Cada una parte de una necesidad concreta del centro de formación y añade una mejora visible a la aplicación.

El alumnado no trabaja con ejercicios aislados: avanza paso a paso hasta convertir una web inicial en una aplicación capaz de guardar y gestionar información.

Actividades de aprendizaje

Actividad 1 · Identificar la información necesaria

Objetivo: reconocer qué datos debe guardar PrestAula.

Caso: una docente solicita un proyector para una sesión formativa y debe devolverlo dos días después.

Resultado esperado: listado razonado de datos necesarios: recurso, código de inventario, persona, fechas, ubicación y estado.

Actividad 2 · Diseñar el modelo de datos

Objetivo: organizar la información antes de crear la base de datos.

En parejas, el alumnado completa el diagrama entidad-relación de PrestAula, indicando las tablas, los campos principales y las relaciones entre categorías, recursos, personas usuarias y préstamos.

Resultado esperado: diagrama revisado y preparado para traducirse a tablas SQLite.

Actividad 3 · Crear la base de datos y datos iniciales

Objetivo: crear las tablas necesarias e introducir ejemplos de prueba.

El alumnado crea la base de datos SQLite, añade las categorías iniciales y registra varios recursos y personas usuarias.

Resultado esperado: archivo SQLite funcional con datos de demostración.

Actividad 4 · Conectar una pantalla con SQLite

Objetivo: sustituir los datos de ejemplo de la web por datos reales.

A partir de una plantilla inicial, el alumnado conecta la página de recursos con SQLite y muestra los recursos disponibles en una tabla HTML.

Resultado esperado: listado de recursos generado desde la base de datos.

Actividad 5 · **Resolver situaciones reales**

Objetivo: aplicar reglas de funcionamiento y controlar errores.

El alumnado recibe casos como: intentar prestar un recurso ya prestado; registrar una devolución con fecha incorrecta; solicitar material con una persona inactiva; dar de baja un recurso con préstamos anteriores.

Resultado esperado: aplicación de validaciones y mensajes claros para cada situación.

Actividad 6 · **Consolidación final: ¿qué haría PrestAula?**

Objetivo: fijar lo aprendido.

A partir de seis situaciones breves, el alumnado identifica qué dato interviene, qué operación CRUD corresponde, qué validación debe aplicarse y cuál debe ser la respuesta de la aplicación.

Resultado esperado: ficha de comprobación individual corregida en pareja y puesta en común de los errores más frecuentes..

Metodología didáctica

La propuesta se desarrolla mediante una metodología activa basada en proyecto. PrestAula funciona como hilo conductor durante toda la unidad: el alumnado analiza una necesidad real, diseña una solución y la convierte en una aplicación web funcional.

Principios metodológicos

Aprendizaje basado en proyecto — el trabajo se organiza alrededor de PrestAula, una aplicación con una finalidad reconocible: controlar préstamos de recursos en un centro de formación.

De lo simple a lo complejo — primero se trabaja con recursos, después con personas usuarias y finalmente con préstamos, devoluciones y reglas de disponibilidad.

Explicación + demostración + práctica

Trabajo individual y colaborativo

- Análisis y diseño del modelo de datos en parejas.
- Desarrollo de ejercicios individuales.
- Puestas en común para revisar decisiones y errores frecuentes.
- Resolución colaborativa de casos prácticos.

Atención a diferentes ritmos

- Plantillas iniciales para quien necesite apoyo.
- Ejemplos de consultas y estructuras parcialmente completadas.
- Retos de ampliación para quien avance más rápido.

14 Evaluación, cierre y demo funcional

Evaluar el proceso y el resultado

La evaluación será continua, práctica y centrada en evidencias de aprendizaje. No se valorará únicamente que la aplicación funcione: también se tendrá en cuenta la capacidad para analizar el problema, diseñar el modelo de datos, aplicar reglas de funcionamiento y explicar las decisiones tomadas.

E1 Modelo de datos y creación de la base de datos `evidencia práctica`

A partir de un supuesto práctico, el alumnado identifica entidades, atributos, relaciones, claves e índices básicos; crea las tablas necesarias mediante SQL y documenta el modelo de datos.

Qué se comprueba

- Identificación correcta de recursos, personas usuarias y préstamos.
- Diseño de relaciones coherentes.
- Creación de tablas con campos adecuados.
- Uso básico de claves principales, foráneas y restricciones.
- Claridad en la documentación del modelo.

E2 Acceso a datos desde PrestAula `evidencia práctica`

El alumnado conecta la aplicación web con SQLite e implementa operaciones de consulta, alta, modificación y gestión de préstamos, aplicando validaciones y tratamiento de errores.

Qué se comprueba

- Conexión correcta entre Bun, TypeScript y SQLite.
- Consulta y representación de datos en la web.
- Uso de operaciones CRUD.
- Registro de préstamos y devoluciones.
- Validaciones comprensibles y coherentes.

Rúbrica simplificada

Criterio	Iniciado	Conseguido	Destacado
Modelo de datos	Identifica parte de la información necesaria.	Organiza entidades y relaciones correctamente.	Justifica decisiones y previene duplicidades.
SQL y SQLite	Crea consultas o tablas con apoyo.	Crea y consulta tablas relacionadas.	Aplica restricciones y consultas con autonomía.

Criterio	Iniciado	Conseguido	Destacado
Integración web	Muestra datos de ejemplo.	Consulta y guarda datos reales desde la app.	Integra el CRUD con validaciones claras.
Reglas de PrestAula	Detecta algunos errores.	Impide situaciones incorrectas básicas.	Gestiona errores y conserva el historial.
Comunicación técnica	Describe parcialmente lo realizado.	Explica el funcionamiento con claridad.	Relaciona decisiones técnicas con necesidades reales.

De una web inicial a una aplicación que gestiona información

PrestAula muestra cómo una aplicación web puede evolucionar desde una interfaz con datos de ejemplo hasta una herramienta capaz de guardar, consultar y actualizar información de forma organizada. A lo largo de la propuesta, el alumnado:

- identifica qué información necesita gestionar un centro de formación;
- diseña un modelo de datos con recursos, personas usuarias y préstamos;
- crea una base de datos SQLite con tablas relacionadas;
- conecta una aplicación web desarrollada con Bun y TypeScript;
- implementa operaciones CRUD;
- aplica validaciones para evitar errores y conservar el historial.

IDEA FINAL

Una base de datos no es solo un lugar donde se almacenan datos. Es la base que permite que una aplicación recuerde, relacione información y responda correctamente a situaciones reales.

Demo funcional

PrestAula · Sistema de préstamo de recursos

Explora la aplicación funcional y consulta las operaciones trabajadas durante la unidad.

Enlace: https://prestaula.samuhlo.dev/

[VER DEMO]

Reto de ampliación

Añade una nueva consulta a PrestAula que permita mostrar:

> Los recursos cuya fecha prevista de devolución se encuentra dentro de los próximos siete días.

Esta mejora puede incorporarse al panel de inicio como aviso para facilitar el seguimiento de los préstamos activos.

ANEXO Glosario y bibliografía

Conceptos para revisar y fuentes de consulta.

Glosario básico

Término	Definición
Base de datos	Sistema organizado para guardar información y recuperarla cuando sea necesaria.
Campo	Cada dato que se guarda dentro de una tabla, por ejemplo nombre o fecha.
Clave foránea	Campo que conecta un registro de una tabla con otro registro de otra tabla.
Clave principal	Dato que identifica de forma única cada registro de una tabla.
CRUD	Siglas de crear, consultar, modificar y eliminar información.
Dato	Información concreta, como el nombre de un recurso o una fecha.
Formulario	Pantalla que permite introducir o modificar información.
Persistencia	Capacidad de conservar los datos aunque la aplicación se cierre o reinicie.
Registro	Conjunto de campos que describe una entidad concreta, como un recurso o un préstamo.
SQL	Lenguaje utilizado para crear, consultar y modificar datos en una base de datos.
Tabla	Estructura que organiza registros del mismo tipo en filas y campos.

Bibliografía y webgrafía

- Boletín Oficial del Estado. **Real Decreto 1531/2011, de 31 de octubre**, por el que se establecen certificados de profesionalidad de la familia profesional de Informática y Comunicaciones.
- Bun. **Documentación oficial de Bun y del módulo `bun:sqlite`**.
- SQLite. **Documentación oficial del lenguaje SQL y creación de tablas**.
- Materiales elaborados para la propuesta didáctica PrestAula: modelo de datos, diagramas, actividades, ejemplos SQL, fragmentos TypeScript y diseño visual.