Memoria técnica - BookMarket

1. Introducción

Descripción del dominio del problema

El proyecto aborda el dominio de la compraventa de libros usados entre particulares. Se busca facilitar la publicación y búsqueda de libros con una interfaz web simple y un flujo de compra inicial mediante carrito.

Objetivos de la aplicación

- Permitir alta y autenticación de usuarios.
- Gestionar libros y categorías mediante operaciones CRUD.
- Ofrecer un carrito funcional para añadir y quitar libros.
- Mantener la información persistida con Doctrine sobre base de datos relacional.

2. Análisis del sistema

Funcionalidades principales

- Registro de usuarios con validación y hash de contraseña.
- Inicio/cierre de sesión.
- Gestión de libros (crear, listar, editar, eliminar).
- Gestión de categorías.
- Visualización y gestión del carrito personal por usuario.

Casos de uso / flujo principal

1. El usuario se registra en `/register`.
2. Inicia sesión en `/login`.
3. Accede al catálogo de libros.
4. Añade libros al carrito desde listado o detalle.
5. Revisa su carrito y elimina libros si lo desea.

3. Diseño

Entidades y campos

- **User**: id, nombre, email, password, rol, fechaRegistro.
- **Book**: id, titulo, autor, descripcion, precio, estado, fechaPublicacion.
- **Categoria**: id, nombre, descripcion.
- **Carrito**: id, fechaCreacion, total.
- **Pedido**: id, fecha, total, estado.
- **Mensaje**: id, contenido, fecha.

Justificación del modelo de datos

- Un usuario puede publicar múltiples libros y tener múltiples pedidos.
- Cada libro pertenece a una categoría.
- Cada usuario tiene un único carrito activo (1:1).
- Un carrito puede tener múltiples libros, y un libro puede aparecer en distintos carritos (N:M), suficiente para este alcance académico.

Estructura general del proyecto

- `src/Entity`: entidades Doctrine.
- `src/Controller`: controladores por módulo.
- `src/Form`: formularios Symfony.
- `templates/`: vistas Twig por funcionalidad.
- `config/`: seguridad, rutas, Doctrine y configuración de framework.

4. Implementación

Controladores más relevantes

- `RegistroController`: alta de usuario, hash de contraseña y creación del carrito inicial.
- `SecurityController`: login/logout.
- `BookController`: CRUD de libros.
- `CarritoController`: listado, añadir y quitar libros del carrito, recálculo de total.

Formularios y validaciones

Se usan formularios Symfony con validaciones como `NotBlank`, `Email`, `Positive` y `Length` para asegurar que los datos de entrada sean correctos antes de persistir.

Acceso a base de datos mediante Doctrine

La persistencia se implementa con Doctrine ORM usando entidades anotadas con atributos PHP. Los datos se guardan con `persist()` + `flush()` y se recuperan por repositorios.

5. Conclusiones

Dificultades encontradas

- Ajustar relaciones bidireccionales entre usuario y carrito.
- Integrar un carrito funcional en un proyecto inicialmente orientado al CRUD.
- Unificar validaciones de formulario y entidad para el registro.

Aprendizajes adquiridos

- Diseño de dominio básico en Symfony con Doctrine.
- Seguridad y autenticación con roles.
- Implementación de flujos funcionales más allá del CRUD (carrito de compra).
- Documentación técnica estructurada para defensa del proyecto.