Technical Design Document: Libro Book Management System

Introduction

The technical design document provides a detailed overview of the Libro Book Management System, outlining its architecture, components, design patterns, and technical considerations. It aims to guide the development team in implementing a robust and scalable ASP.NET API.

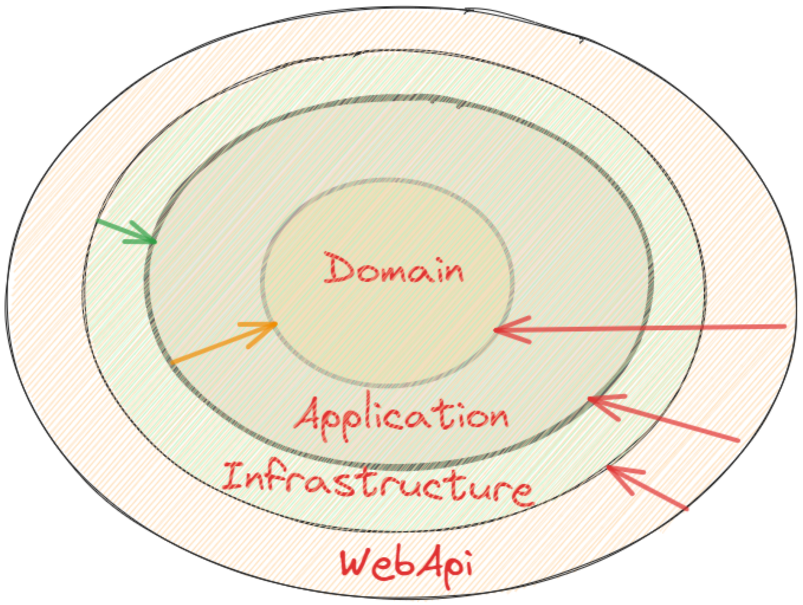
Architecture Overview:

The Libro Book Management System follows a clean architecture, promoting separation of concerns and modularity. It consists of distinct layers, including WebApi, Application, Domain, and Infrastructure, with clear boundaries and dependencies. This architecture enables testability, maintainability, and flexibility for future enhancements.

Design Patterns

The Libro Book Management System incorporates the following design patterns:

- Clean Architecture: The clean architecture principles are followed to separate concerns and keep the codebase flexible and adaptable to future changes. The architecture consists of layers with clear boundaries and dependencies, promoting testability and maintainability.



- CQRS (Command Query Responsibility Segregation): CQRS is implemented to separate read and write operations, optimizing each independently for improved scalability and performance. This approach allows for tailored architectures for handling commands (write operations) and queries (read operations).
- Domain-Driven Design (DDD): DDD is employed to align the project with the business domain and promote a common language and understanding between developers and domain experts. The focus is on modeling core domain concepts and behaviors, resulting in a more robust and adaptable software system.
- Service Pattern: The service pattern is utilized to encapsulate complex business logic and operations into reusable and cohesive service components. This promotes modularity, separation of concerns, and clean code organization.
- Repository Pattern: The repository pattern provides a layer of abstraction between the application and the data persistence layer. It enables a consistent and standardized way of accessing and manipulating data, improving code maintainability, testability, and scalability.
- Unit of Work Pattern: The unit of work pattern is utilized to manage transactions and ensure atomicity across multiple database operations. It provides a cohesive interface to perform multiple operations as a single unit, ensuring data consistency and integrity.

Technologies and Frameworks:

The API is built using ASP.NET Core, leveraging its rich set of features and libraries. It utilizes Entity Framework Core for data access, ensuring seamless interaction with the underlying database. Other technologies and frameworks, such as `Mapster` for object mapping and `FluentValidation` for input validation, are also incorporated to enhance development efficiency and code quality.

System Components:

The major components of the Libro Book Management System include Controllers, Services, Repositories, Models, and Data Access. Controllers handle incoming HTTP requests and orchestrate the flow of data. Services encapsulate business logic and perform operations on entities. Repositories provide an abstraction layer for data access, enabling separation between the domain and the underlying database. Models represent the entities and data structures used throughout the system, while Data Access handles database interactions.

API Endpoints:

The API exposes various endpoints to facilitate book management, user authentication, and system administration. These endpoints include functionalities such as book search, borrowing and returning books, user registration and authentication, and CRUD operations for books and authors. Each endpoint follows RESTful principles and adheres to the appropriate HTTP methods and status codes.

Data Storage:

The Libro Book Management System utilizes a relational database for data storage, with Entity Framework Core as the ORM (Object-Relational Mapping) tool. The database schema is designed to capture the entities and their relationships, ensuring data integrity and efficient querying. Proper indexing and optimization techniques are applied to enhance performance and scalability.

Authentication and Authorization

The Libro Book Management System implements JWT (JSON Web Token) authentication for secure user authentication and authorization. JWTs are used to securely transmit user information between the client and server, ensuring that only authorized users can access protected resources. The system assigns different roles and permissions to users based on their privileges, allowing for fine-grained access control.

Error Handling:

The API incorporates a robust error handling mechanism to provide meaningful error responses to clients. It includes proper exception handling, custom exception types, and global error handling middleware to ensure consistency and improve the user experience. Detailed error messages and appropriate HTTP status codes are returned to assist in troubleshooting and debugging.

```
namespace WebApi.Middleware;

public sealed class LoggerMiddleware : IMiddleware
{
    private readonly ILogger<LoggerMiddleware> _logger;

    public LoggerMiddleware(ILogger<LoggerMiddleware> logger)
    {
        _logger = logger;
    }

    public async Task InvokeAsync(HttpContext context, RequestDelegate next)
    {
        try
        {
            _logger.LogInformation($"Request: {context.Request.Method} {context.Request.Path}" +
                                   $" from {context.Connection.RemoteIpAddress}");

            await next(context);

            _logger.LogInformation($"Request completed: {context.Request.Method} {context.Request.Path}" +
                                   $" from {context.Connection.RemoteIpAddress}");
        }
        catch (Exception ex)
        {
            _logger.LogError(ex.Message,
                $"Error occurred while processing request: {context.Request.Method} {context.Request.Path}" +
                $" from {context.Connection.RemoteIpAddress}");

            throw;
        }
    }
}
```

and add the following dependencies to `Program.cs`:

```
app.UseMiddleware<LoggerMiddleware>();
```

Package Dependencies

The Libro Book Management System relies on several external packages and frameworks to support its functionality. These include:

- `ASP.NET Core`: The web framework used to build the API and handle HTTP requests.
- Entity Framework Core: The ORM (Object-Relational Mapping) tool used for data access and database interactions.
- `FluentValidation`: A package used for input validation, providing a fluent syntax for defining validation rules and improving code maintainability.
- `Mapster`: A package used for object mapping, simplifying the process of mapping objects between different types and structures.
- `StackExchange.Redis`: A package used for caching, improving application performance by reducing the need for frequent database queries.

Additional Features

The Libro Book Management System includes the following additional features:

- Reading Lists: Users can create and manage reading lists to keep track of books they are interested in or have already read.
- Book Reviews and Ratings: Users can provide reviews and ratings for books, allowing others to make informed decisions and discover new books.
- Notifications: Users can receive notifications about book availability, due dates, and other relevant updates.
- Book Recommendations: The system can provide personalized book recommendations based on user preferences, reading history, and other relevant factors.

Performance and Scalability:

Performance and scalability considerations are paramount in the Libro Book Management System. Caching mechanisms, such as Redis through the `StackExchange.Redis` package, are employed to store frequently accessed data and reduce database roundtrips. Performance optimizations, like query optimization and efficient data retrieval, are implemented to minimize response times and handle increased user loads.