Technical Documentation

Project: Digital Signature of PDF documents

Table of contents:

1. Project Overview

The Digital Signature System is a backend-oriented software solution developed using the C# programming language and the .NET framework. The purpose of the system is to enable digital signing of PDF documents in order to ensure document authenticity, integrity, and non-repudiation. By applying cryptographic techniques, the system allows documents to be signed in a secure and verifiable manner, making it suitable for use cases such as certificates, contracts, reports, and other official documents.

In addition to embedding a digital signature into PDF files, the system supports the optional insertion of a QR code. This QR code can be used to store verification data, metadata related to the signed document, or references to external validation mechanisms. A significant emphasis of the project is placed on software quality and maintainability, achieved through the application of clean architectural principles rather than focusing solely on functional implementation.

2. System Architecture

The system is designed according to the principles of Clean Architecture, also known as Onion Architecture. This architectural approach enforces a strict separation of concerns and ensures that dependencies flow inwards, toward the core abstractions of the system. High-level business rules are kept independent from low-level technical details, such as external libraries or frameworks.

The architecture is organized into four primary layers: Web.Api, Application.Services, Domain.Interfaces and Infrastructure.Services. Each layer has a well-defined responsibility and communicates with other layers exclusively through abstractions. This design minimizes coupling between components and improves long-term maintainability, testability, and extensibility.

3. Web.Api Layer

The Web.Api layer represents the presentation layer of the system. It is responsible for exposing the digital signature functionality through RESTful endpoints and handling client requests. This layer does not contain business logic and acts only as a mediator between external clients and the application services layer. By delegating all processing to application services, the Web.Api layer remains lightweight and focused solely on request handling and system integration.

4. Application.Services Layer

The Application.Services layer contains the application logic and is responsible for orchestrating the digital signature workflow. This layer implements the use cases of the system by coordinating interactions between domain abstractions and infrastructure services. It depends only on the interfaces defined in the Domain.Interfaces layer and remains unaware of concrete implementations.

The central component of this layer is the digital signature service, which controls the sequence of operations required to sign a document. It receives the input PDF document, invokes the digital signature generation process, requests modifications to the PDF, and optionally triggers QR code generation and insertion. By delegating all technical responsibilities to infrastructure services, this layer maintains a clear separation between business logic and implementation details.

5. Domain.Interfaces Layer

The Domain.Interfaces layer represents the core of the system and defines the fundamental contracts that describe the system's capabilities. This layer specifies what the system can do without providing any information about how those operations are implemented. It contains only interface definitions and does not depend on any external frameworks, libraries, or technical implementations.

Within this layer, interfaces are defined for digital signature operations, PDF processing, and QR code generation. These interfaces serve as a stable foundation for the entire system. Because no implementation details are present, this layer remains unaffected by changes in technology or infrastructure. All other layers depend on the contracts defined here, making this layer the most stable and critical part of the architecture.

6. Infrastructure.Services Layer

The Infrastructure.Services layer contains the concrete technical implementations required for the system to function. This layer is responsible for interacting with external libraries and handling low-level concerns such as PDF manipulation, cryptographic operations, and QR code generation. It implements the interfaces defined in the Domain.Interfaces layer, thereby fulfilling the contracts expected by the application layer.

This layer includes helper components for working with PDF documents and generating QR codes. Since all infrastructure logic is isolated within this layer, changes to libraries or technologies can be made without affecting the rest of the system. This design choice ensures that technical decisions do not propagate into business logic, preserving architectural integrity.

7. Digital Signature Workflow

The digital signing process follows a clearly defined and controlled workflow. Initially, a PDF document is provided as input to the system. A cryptographic digital signature is then generated based on the document content. This signature is embedded into the PDF file to ensure that any subsequent modification of the document can be detected. If required, a QR code is generated and inserted into the document to store additional verification information. Finally, the system outputs a digitally signed PDF document.

This workflow is managed by the application layer, which ensures that all steps are executed in the correct order and that each responsibility is delegated to the appropriate service.

8. Digital Signature Concepts

The system implements application-level digital signatures, focusing on preserving document integrity and verifying the authenticity of the signer. The use of cryptographic techniques ensures that the signed document cannot be altered without invalidating the signature. The abstraction of cryptographic operations allows the system to support future changes in algorithms or security standards without requiring modifications to business logic.

By isolating cryptographic concerns within infrastructure services, the system maintains flexibility and adaptability in a rapidly evolving security landscape.

9. Technologies Used

The system is implemented using the C# programming language and the .NET framework. It applies Clean Architecture principles and relies on interface-based design to enforce loose coupling between components. Core technical concepts include digital signatures, cryptography, PDF processing, and QR code generation. The selection of these technologies reflects the project's focus on robustness, maintainability, and real-world applicability.

10. Non-Functional Characteristics

The architectural design of the system promotes maintainability by clearly separating responsibilities across layers. Testability is improved through the use of interfaces, which allow dependencies to be easily mocked or replaced during testing. The system is also highly extensible, enabling new PDF libraries, cryptographic providers, or verification mechanisms to be introduced with minimal impact on existing code.

11. Limitations

The project does not include a graphical user interface and is intended to function as a backend or service-level component. Management of certificate authorities and advanced identity validation are outside the scope of the system. The implementation focuses specifically on digital signing functionality rather than full document lifecycle management.

12. Conclusion

The Digital Signature System demonstrates a structured and maintainable approach to implementing digital document signing using modern software architecture principles. By combining clean architectural design with practical cryptographic functionality, the project provides a solid foundation for secure PDF document processing. Its modular design makes it suitable for academic evaluation as well as for extension into real-world applications.