Software Requirements Specification (SRS) for the Certificate Management System

1. Introduction

1.1 Purpose

The purpose of this document is to outline the requirements for a Certificate Management System intended for universities, training centers, educators, and event organizers. The system will facilitate the creation, signing, and management of participation or completion certificates.

1.2 Scope

This system aims to automate the process of creating and issuing certificates. It allows organizations to create certificates, add participants, and generate personalized PDF documents featuring participant names. The system ensures digital signing of documents using the ED25519 algorithm and securely stores them in GridFS. Additionally, it offers certificate verification functionality to everybody and organization management features.

1.3 Intended Audience

This project is intended for:

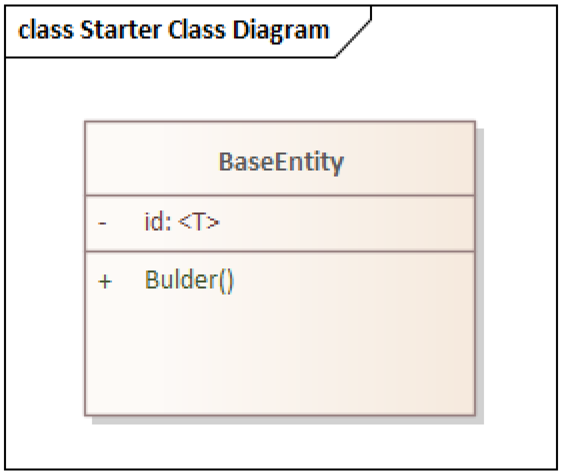
- Universities, Training Centers, and Educational Institutions: Organizations that need to issue certificates to students, participants, or attendees upon completion of courses, training sessions, or events.
- Educators and Event Organizers: Instructors, teachers, and coordinators who require an efficient system to create, manage, and distribute certificates to their audience.
- Participants and Certificate Recipients: Individuals who will receive certificates and may need to verify their authenticity through the system.

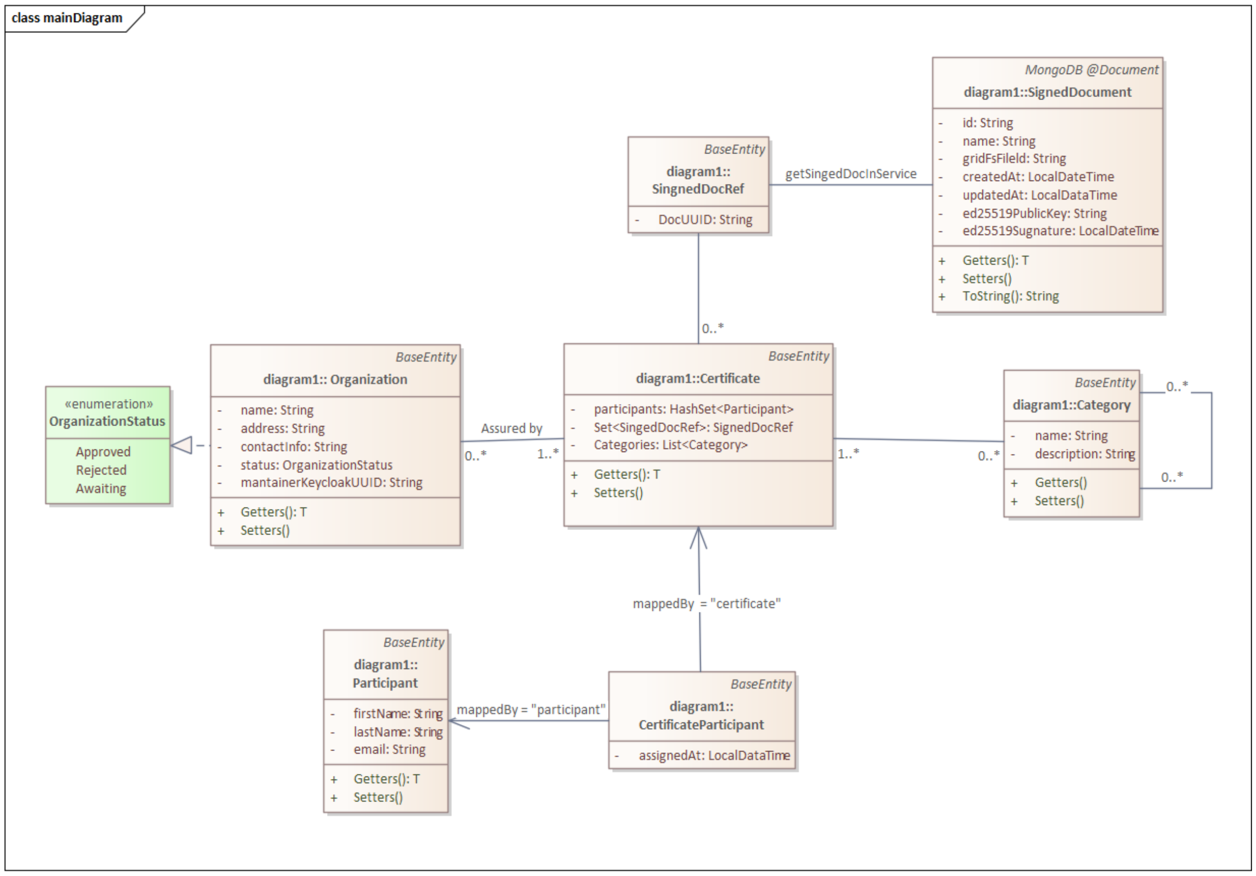
2. Overall Description

2.1 Product Perspective

The system is a web application based on a microservices architecture using Spring Boot. Data storage is managed using PostgreSQL databases and MongoDB (GridFS for file storage). Authentication and authorization are handled via Keycloak (OAuth2). Additional infrastructure includes Eureka Server for service discovery and Spring WebFlux Gateway for API routing.

2.2 Product Features





- Organization Creation: Users can create organizations, which can then be approved or rejected by an administrator.
- Certificate Management:
    - Create certificate templates with descriptions and categories.
    - Add participants by specifying names, surnames, and email addresses.

    - Generate personalized PDF certificates for each participant with their names included.
- Digital Signing and Storage:
    - Certificates are signed using the ED25519 algorithm via the Bouncy Castle library.
    - Signed documents are stored in GridFS with unique identifiers (UUIDs).
- Certificate Verification:
    - Any user can verify the authenticity of a certificate by uploading the certificate file and verifying it using the provided public key and signature.
- Organization Management:
    - Administrators can approve or reject organizations created by users.
    - The status of the organization is displayed to the user (approved, rejected, pending).
- Category Management:
    - Users can create and manage categories to classify certificates.
    - Supports hierarchical category structures (subcategories).

2.3 User Classes and Characteristics

1. Unregistered User:
    - Can verify the authenticity of certificates by uploading the certificate file and signature.
    - Has access to public information about certificates and organizations.
2. Registered User:
    - Can create organizations by providing necessary details (name, address, contact information).
    - Can create and manage certificates within their organizations.
    - Can add participants to receive certificates.
3. Administrator:
    - Approves or rejects organizations created by users.
    - Manages users and has full access to the system.

    - Can view and modify any certificates and organizations.

2.4 Operating Environment

- Server-Side:
    - Programming Language: Java 17+
    - Framework: Spring Boot
    - Databases: PostgreSQL, MongoDB (GridFS)
    - Authentication: Keycloak (OAuth2)
    - Service Discovery: Eureka Server
    - API Gateway: Spring WebFlux Gateway
- Infrastructure:
    - Docker is used for containerization of Keycloak, MongoDB, and PostgreSQL services.

2.5 Design and Implementation Constraints

- Security:
    - Private keys must be stored securely using keystores in P12 format.

2.6 Dependencies

- Keycloak: Successful operation for authentication and authorization.
- Databases: Availability and proper functioning of PostgreSQL and MongoDB databases.
- Libraries: Correct functioning of libraries for digital signatures (Bouncy Castle).

3. Specific Requirements

3.1 Functional Requirements

3.1.1 User Management

- Registration and Login:
    - Users must be able to register and log in to the system via Keycloak.
- Roles and Access Control:
    - The system must distinguish between roles: Unregistered User, Registered User, and Administrator.

    - Access to system functions is granted according to the user's role.

3.1.2 Organization Management

- Organization Creation:
    - Registered Users can create organizations by providing necessary details (name, address, contact information).
- Organization Approval:
    - Administrators can approve or reject organizations.
    - The status of the organization is displayed to the user.

3.1.3 Certificate Management

- Certificate Creation:
    - Registered Users can create certificate templates with descriptions and assign categories.
- Adding Participants:
    - Users can add a list of participants by specifying their names, surnames, and email addresses.
- Certificate Generation:
    - The system generates personalized PDF certificates for each participant.
    - Certificates are digitally signed using the ED25519 algorithm.

3.1.4 Storage and Verification

- Document Storage:
    - Signed certificates are stored in GridFS using unique identifiers (UUIDs).
- Certificate Verification:
    - Any user can upload a certificate file and verify its authenticity using the provided public key and signature, without accessing the actual file contents.

3.1.5 Category Management

- Category Creation:
    - Users can create and manage categories for classifying certificates.
- Category Hierarchy:

    - Supports hierarchical category structures (subcategories).

3.2 Non-Functional Requirements

3.2.1 Security

- Authentication and Authorization:
    - Use OAuth2 and Keycloak to ensure secure access.
- Key Storage:
    - Private keys must be stored in a secure keystore (P12 format).

3.3 Constraints

- System Limitations:
    - The system is not intended for automatic verification of participant data authenticity through external databases or services.

4. Data Model

4.1.1 BaseEntity<T>

- Fields:
    - id: Primary key of generic type T.

4.1.2 Organization

- Fields:
    - name: Organization name.
    - address: Physical address.
    - contactInfo: Contact information.
    - status: Approval status (approved, rejected, pending).
    - maintainerKeycloakUUID: ID of the user who created the organization.
    - certificates: Certificates issued by the organization.

4.1.3 Participant

- Fields:
    - name: First name.

  - surname: Last name.
  - email: Email address.
  - certificateParticipants: Certificates associated with the participant.

4.1.4 Category

- Fields:
  - description: Text description.
  - certificates: Certificates in this category.
  - parentCategory: Parent category, if any.
  - subCategories: Subcategories under this category.

4.1.5 CertificateParticipant

- Fields:
  - certificate: Reference to the associated Certificate entity.
  - participant: Reference to the associated Participant entity.
  - assignDate: Date and time when the certificate was assigned.

4.1.6 SignedDocRef

- Fields:
  - uuidOfDoc: UUID of the document stored in GridFS.

4.1.7 Certificate

- Fields:
  - description: Text description of the certificate.
  - issuers: Set of issuing organizations.
  - signedDocumentUUID: References to signed documents stored in GridFS.
  - categories: Categories the certificate belongs to.
  - certificateParticipants: Participants associated with the certificate.

4.1.8 SignedDoc

- Fields:

- id: UUID of the signed document.
- name: Name of the document.
- fileType: Type of the file (e.g., PDF).
- gridFsFileId: ID of the file in GridFS.
- createdAt: Date and time when the document was created.
- updatedAt: Date and time when the document was last updated.
- ed25519PublicKey: Public key used for the ED25519 signature.
- ed25519Signature: Digital signature of the document.