Implementação de Integridade de Documentos com Hash

Visão Geral

Esta implementação adiciona funcionalidades para garantir a integridade dos documentos no sistema de assinatura digital. Utilizamos hashes SHA-256 para verificar se os documentos não foram alterados após o upload ou após a assinatura.

Arquivos Criados/Modificados

1. **Novos Arquivos:**
   - `src/integrity/integrity.service.ts` - Serviço responsável pela geração e verificação de hashes
   - `src/migrations/1683456789000-IntegrityHashColumns.ts` - Migração para adicionar colunas de hash ao banco de dados
2. **Arquivos Modificados:**
   - `src/document/document.entity.ts` - Adicionadas colunas para armazenar hashes
   - `src/document/document.module.ts` - Adicionado IntegrityService aos providers
   - `src/document/document.service.ts` - Integrado IntegrityService no fluxo de upload e adicionado método para gerar documento final
   - `src/document/document.controller.ts` - Adicionados endpoints para verificar integridade

Explicação do Código Implementado

IntegrityService

O serviço de integridade (`IntegrityService`) é responsável por:

1. **Gerar hashes SHA-256** para documentos originais e assinados
2. **Armazenar hashes** no banco de dados
3. **Verificar a integridade** comparando o hash armazenado com o hash calculado em tempo real

Principais métodos: - `generateHash(buffer)`: Gera um hash SHA-256 a partir de um buffer - `storeOriginalHash(document, buffer)`: Gera e armazena o hash do documento original - `storeSignedHash(documentId, buffer)`: Gera e armazena o hash do documento assinado - `verifyIntegrity(documentId, isSigned)`: Verifica a integridade do documento comparando o hash armazenado com o hash calculado

Modificações na Entidade Document

Adicionamos dois novos campos à entidade Document: - `hashOriginal`: Armazena o hash SHA-256 do documento original - `hashSigned`: Armazena o hash

SHA-256 do documento final com assinaturas

Integração no Fluxo de Upload

O hash do documento original é gerado e armazenado automaticamente durante o upload do documento:

```
// No método create do DocumentService
savedDocument = await this.documentRepository.save(document);

// Gerar e armazenar hash do documento original
savedDocument = await this.integrityService.storeOriginalHash(savedDocument, file.buffer);
```

Geração do Documento Final

Foi adicionado um método `generateFinalDocument` ao DocumentService para gerar o documento final com assinaturas e armazenar seu hash:

```
async generateFinalDocument(documentId: number, finalDocumentBuffer: Buffer): Promise<Docume
  this.logger.log(`Generating final document with signatures for document ID: ${documentId}`

  // Gerar e armazenar hash do documento final assinado
  const document = await this.integrityService.storeSignedHash(documentId, finalDocumentBuf

  // Log audit trail
  await this.auditLogService.logAction(document.owner.id, 'GENERATE_FINAL_DOCUMENT', 'Docume
  return document;
}
```

Endpoints para Verificação de Integridade

Foram adicionados dois endpoints ao DocumentController:

1. **GET /api/documents/:id/integrity**
   - Retorna os hashes armazenados do documento (original e assinado)
   - Indica se o documento possui hash (isValid)
2. **POST /api/documents/:id/integrity/verify**
   - Recalcula o hash do documento original
   - Compara com o hash armazenado
   - Retorna o resultado da verificação

Como Testar

1. Executar a Migração

Primeiro, execute a migração para adicionar as colunas de hash ao banco de dados:

```
# Dependendo da configuração do projeto, use um dos comandos:
npm run typeorm:migration:run
# ou
npx typeorm migration:run
```

2. Testar Upload de Documento

Faça upload de um documento para verificar se o hash original está sendo gerado:

```
curl -X POST http://localhost:3000/api/documents \
  -H "Authorization: Bearer SEU_TOKEN_JWT" \
  -F "file=@caminho/para/documento.pdf" \
  -F "title=Documento de Teste" \
  -F "description=Descrição do documento"
```

3. Verificar Hash Original

Verifique se o hash original foi armazenado:

```
curl -X GET http://localhost:3000/api/documents/ID_DO_DOCUMENTO/integrity \
  -H "Authorization: Bearer SEU_TOKEN_JWT"
```

4. Verificar Integridade do Documento

Verifique a integridade do documento recalculando o hash:

```
curl -X POST http://localhost:3000/api/documents/ID_DO_DOCUMENTO/integrity/verify \
  -H "Authorization: Bearer SEU_TOKEN_JWT"
```

Considerações de Segurança

- Os hashes SHA-256 são considerados seguros para verificação de integridade
- A verificação de integridade ajuda a detectar alterações não autorizadas nos documentos
- Recomenda-se implementar medidas adicionais de segurança, como assinaturas digitais com certificados