CRONotary - Design Dokument

1 Architekturübersicht

- Frontend: Webanwendung für Nutzerinteraktion (Dokumentenupload, Signatur, Überprüfung)
- Backend: Server zur Hash-Generierung, Wallet-Kommunikation und Blockchain-Interaktion.
- Cronos-Blockchain: Speichert Hashes und Wallet-Signaturen in Smart Contracts.
- Wallet-Integration: Nutzer verbinden ihre Wallets (z.B. MetaMask) zur Signatur von Transaktionen.

2 Kernkomponenten

2.1 Frontend

- **Technologien**: React.js, TypeScript, Web3.js (für Wallet-Integration).
- **Funktionen**:
  - Dokumentenupload (Drag & Drop oder Dateiauswahl).
  - Anzeige des generierten Hashs.
  - Wallet-Verbindung zur Signaturbestätigung.
  - Überprüfungsseite mit Ergebnisanzeige (Echt“/Verändert“).
  - Werbebanner für Monetarisierung.

2.2 Backend

- **Technologien**: Node.js, Express.js, Cronos SDK.
- **Funktionen**:
  - Hash-Generierung mit `SHA-256` (garantiert Eindeutigkeit und Sicherheit).
  - Kommunikation mit Cronos-Blockchain (Speichern/Abrufen von Hashes).
  - Abwicklung von Transaktionsgebühren (z. B. 0.5 CRO pro Signatur).
  - API-Endpunkte für Frontend-Integration.
  - Automatische Auszahlung der Gebühren an den Smart-Contract Besitzer

2.3 Smart Contracts

- **Funktion**: Speicherung von Hash-Werten und zugehörigen Wallet-Adressen auf der Blockchain.
- **Sprache**: Solidity (kompatibel mit Cronos EVM).

3 Datenfluss

3.1 Signaturprozess (Nutzer A)

1. Dokument wird im Frontend hochgeladen.
2. Backend generiert SHA-256-Hash.
3. Nutzer bestätigt Transaktion via Wallet (Signatur + Gebührenzahlung).
4. Hash und Wallet-Adresse werden im Smart Contract gespeichert.
5. Gebühr wird automatisch an den Smart-Contract Depolyer ausgezahlt.

3.2 Überprüfungsprozess (Nutzer B)

1. Dokument wird hochgeladen.
2. Backend generiert Hash und fragt Blockchain ab.
3. Smart Contract vergleicht Hash mit gespeichertem Wert.
4. Ergebnis (Echt“/Verändert“) wird im Frontend angezeigt.

4 Klassendiagramm

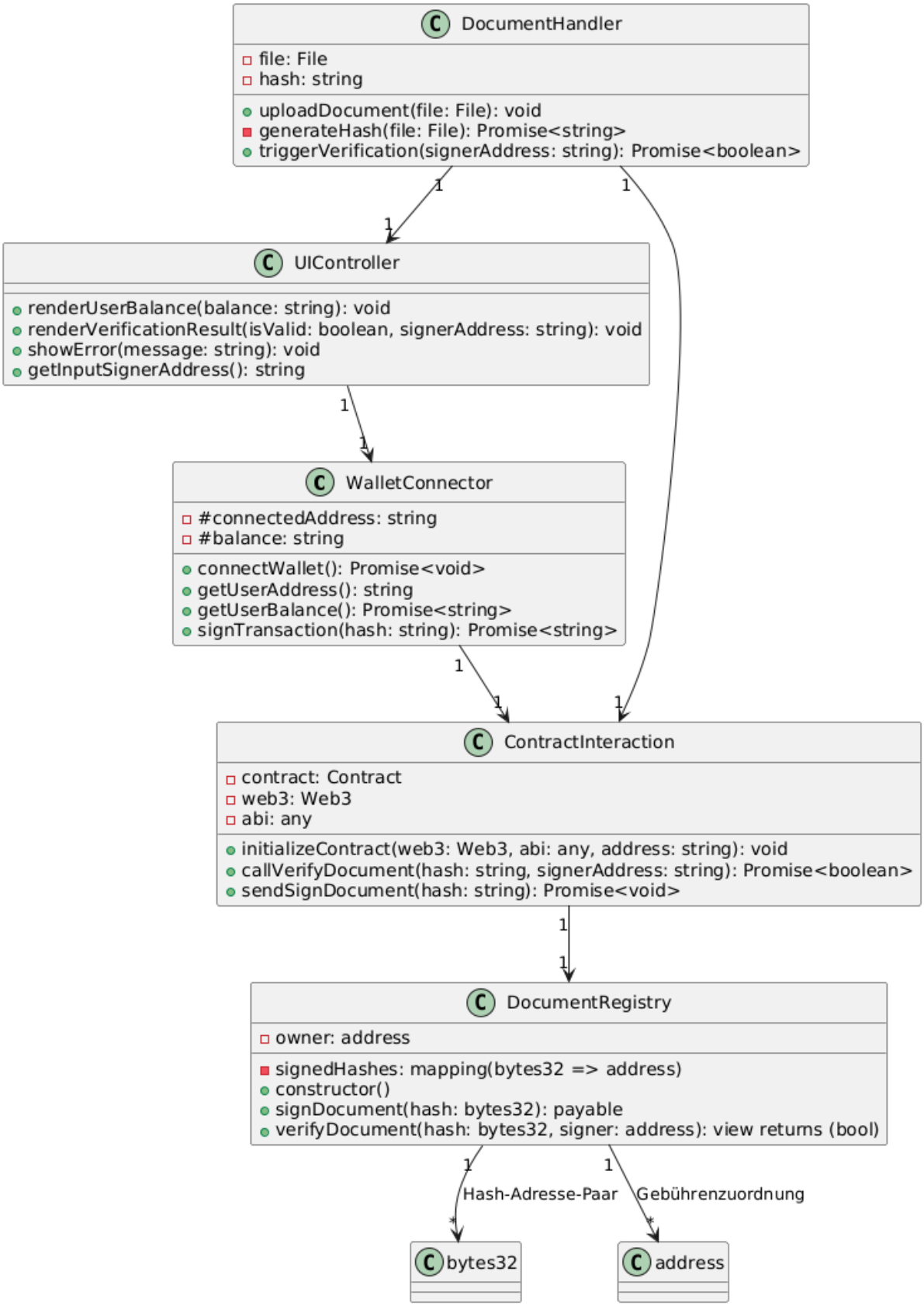

Figure 1: Klassendiagramm