Ideas

Sacado de la tabla 1

**1. Almacenamiento de la información de los estudiantes**

* **Problema identificado:** La información de los estudiantes se almacena de forma local en la PC del instituto académico, lo que presenta riesgos en términos de seguridad, integridad y accesibilidad. Además, el historial de solicitudes atendidas y no atendidas también se guarda de manera centralizada, lo que podría generar posibles pérdidas de datos o vulnerabilidades en el caso de fallos del sistema o accesos no autorizados.
* **Impacto:** Este enfoque centralizado y dependiente de dispositivos locales puede dar lugar a la manipulación, pérdida o acceso indebido a la información sensible.

**2. Áreas encargadas de la emisión de certificados académicos**

* **Problema identificado:** El proceso de emisión de certificados está fragmentado entre diversas áreas (TI, Secretaría, Dirección), lo que implica una serie de pasos manuales y de validación que pueden ser propensos a errores, demoras y falta de transparencia. Además, cada área tiene un acceso limitado a la información, lo que puede generar descoordinación y falta de seguimiento en tiempo real.
* **Impacto:** La falta de centralización y de un proceso automatizado puede resultar en inconsistencias en los certificados, retrasos en la emisión y dificultades para auditar el proceso de emisión de documentos.

**3. Áreas encargadas de la verificación de los certificados académicos**

* **Problema identificado:** El proceso de verificación de certificados involucra múltiples actores, como el área de admisiones y una entidad externa para la validación notarial. Este enfoque aumenta la complejidad del proceso, ya que depende de la correcta transmisión de información entre diversas entidades, lo que puede generar ineficiencias y problemas de autenticidad.
* **Impacto:** El manejo manual de la información y la dependencia de terceros para la validación de los certificados puede generar errores en la verificación, demoras y la posibilidad de que los documentos no sean verificados correctamente o en el tiempo adecuado.

**Propuesta con Blockchain:**

* Utilizar blockchain para centralizar, asegurar y automatizar el proceso de certificación y verificación. Los certificados podrían ser registrados como transacciones en una blockchain, lo que garantizaría su autenticidad, transparencia e inmutabilidad. Cada certificado se validaría y verificara de forma instantánea mediante una cadena de bloques, eliminando los problemas de centralización, errores manuales y la dependencia de terceros para la validación notarial.

Aquí tienes una versión mejorada de tu problema identificado, incorporando los puntos que mencioné antes:

"Actualmente, no existe un mecanismo ágil ni seguro para verificar la validez de los certificados digitales emitidos, lo que genera vulnerabilidades en términos de autenticidad y transparencia. El proceso de emisión y validación de estos certificados está fragmentado y depende de múltiples actores y sistemas no integrados, lo que aumenta la posibilidad de errores, demoras y manipulaciones. Además, la falta de un sistema centralizado y automatizado dificulta la verificación en tiempo real, creando incertidumbre en los usuarios que requieren confirmación rápida y confiable."

**En tu caso, algunos stakeholders podrían ser:**

1. **Instituciones de Educación Superior** – Son las principales beneficiarias del sistema, ya que buscan una solución para certificar documentos digitales de manera segura.
2. **Estudiantes y Graduados** – Usuarios finales que necesitan documentos certificados, como diplomas o certificados de estudios.
3. **Empresas y Empleadores** – Entidades que verificarán la autenticidad de los documentos.
4. **Autoridades Académicas (Rectoría, Secretaría Académica)** – Encargadas de la validación y emisión de documentos oficiales.
5. **Departamento de TI de la Universidad** – Implementará y mantendrá la integración del sistema en la infraestructura de la institución.
6. **Organismos Reguladores** – Entidades gubernamentales o educativas que regulan el uso de tecnología en certificaciones académicas.
7. **Desarrolladores y equipo técnico** – Quienes construyen y mantienen el sistema blockchain y la aplicación.

En tu caso, el **director** podría desempeñar el rol de **stakeholder clave y tomador de decisiones**. Su papel puede incluir:

1. **Patrocinador o Avalador del Proyecto** – Puede apoyar la implementación del sistema y facilitar recursos.
2. **Autoridad en la Aprobación de Procesos** – Supervisaría la adopción de la certificación digital en la institución.
3. **Interlocutor con Otros Departamentos** – Puede coordinar con admisiones, sistemas y otros equipos para asegurar la integración del sistema.
4. **Representante Institucional** – Su respaldo podría ser clave para la aceptación y credibilidad del proyecto dentro de la universidad.

Sí, cuando el director firma el certificado, está **validando y autenticando** el documento, asegurando que la información contenida en él es oficial y legítima. Su firma actúa como un **sello de aprobación institucional**, lo que garantiza que el certificado ha sido emitido por la autoridad correspondiente y cumple con los requisitos académicos y administrativos.

En el contexto de tu tesis, esta validación podría trasladarse al **registro en blockchain**, donde la firma del director equivaldría a una **firma digital o un hash en la red**, garantizando su autenticidad e inalterabilidad. ¿Estás considerando incluir un mecanismo de firma digital en tu prototipo?

**Diagramas de caso de uso del sistema**

¡Perfecto! Para tu prototipo de certificación de documentos en blockchain, un **diagrama de casos de uso** te ayudará a visualizar la interacción entre los usuarios y el sistema.

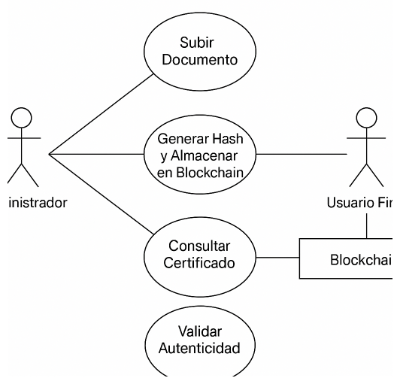
Voy a diseñar el diagrama con los siguientes actores y casos de uso:

**🎭 Actores:**

1. **Administrador**: Responsable de subir y certificar documentos.
2. **Usuario Final**: Consulta y verifica la autenticidad del documento.
3. **Blockchain**: Recibe y almacena el hash del documento.

**📌 Casos de Uso:**

✅ **Subir Documento** (Administrador)  
✅ **Generar Hash y Almacenar en Blockchain** (Sistema)  
✅ **Consultar Certificado** (Usuario Final)  
✅ **Validar Autenticidad** (Sistema y Blockchain)



**🎭 Actores del Sistema**

1. **Administrador** 👨‍💼
   * Responsable de cargar documentos y generar su certificación en blockchain.
2. **Usuario Final** 👤
   * Consulta y verifica la autenticidad de los documentos certificados.
3. **Blockchain** ⛓️
   * Actúa como un sistema externo donde se almacena el hash único del documento para su validación.

**📌 Casos de Uso y Descripción**

✅ **Subir Documento** (*Administrador → Sistema*)

* El administrador carga un documento en formato digital (por ejemplo, PDF).

✅ **Generar Hash y Almacenar en Blockchain** (*Sistema → Blockchain*)

* El sistema calcula un hash único del documento y lo registra en la blockchain.
* No almacena el archivo, solo el hash para garantizar seguridad e integridad.

✅ **Consultar Certificado** (*Usuario Final → Sistema*)

* Un usuario ingresa un código o archivo para verificar su certificación.

✅ **Validar Autenticidad** (*Sistema → Blockchain*)

* El sistema consulta la blockchain para comparar el hash ingresado con el registrado.
* Si coinciden, el documento es válido; si no, se muestra una alerta de posible falsificación.

**📌 Relación entre los elementos**

* **El administrador interactúa con el sistema** para cargar y certificar documentos.
* **El usuario final interactúa con el sistema** para verificar autenticidad.
* **El sistema se comunica con la blockchain** para registrar y validar los certificados.

Desarrollo

**📁 Contexto general del archivo**

Este archivo define funciones relacionadas con los certificados de idioma: emitir, verificar, obtener, invalidar y descargar en PDF.

Se usa:

* **MongoDB** (a través del modelo Certificate) para guardar los certificados.
* **Blockchain (Ethereum)** usando ethers.js para registrar en un contrato inteligente.
* **PDFKit** para generar certificados en formato PDF.

**🔐 Configuración inicial**

js

CopiarEditar

const fs = require('fs');

const path = require('path');

const PDFDocument = require('pdfkit');

const Certificate = require('../models/Certificate');

const { ethers } = require('ethers');

const contractABI = require('../contract/contractABI.json');

const contractAddress = require('../contract/contractAddress.json');

* Importas las librerías necesarias.
* Cargas el **modelo de certificados**, el **ABI del contrato**, y su **dirección**.

**⚙️ Configurar conexión con la blockchain:**

js

CopiarEditar

const provider = new ethers.JsonRpcProvider('http://localhost:8545');

* Conectas a una red local de Ethereum (puede ser Ganache o Hardhat local).

js

CopiarEditar

if (!process.env.PRIVATE\_KEY) {

console.error('❌ No se definió PRIVATE\_KEY en el archivo .env');

process.exit(1);

}

* Aseguras que exista una clave privada para firmar transacciones.

js

CopiarEditar

const wallet = new ethers.Wallet(process.env.PRIVATE\_KEY, provider);

const contract = new ethers.Contract(contractAddress.address, contractABI, wallet);

* Creas una **billetera (wallet)** y una instancia del contrato inteligente que te permite emitir o invalidar certificados en la blockchain.

**🧾 issueCertificate**

js

CopiarEditar

exports.issueCertificate = async (req, res) => {

const { instituteName,facultyName, studentName, cedula, language, level } = req.body;

* Recibes los datos del certificado desde el frontend o cliente.

js

CopiarEditar

const tx = await contract.issueCertificate(...);

await tx.wait();

* Llamas a la función issueCertificate del **contrato inteligente**, que guarda los datos en blockchain.
* Esperas a que la transacción se confirme (wait()).

js

CopiarEditar

const newCert = new Certificate({...});

await newCert.save();

* Guardas el certificado también en **MongoDB**, incluyendo la fecha y estado de validez (valid: true).

**✅ verifyCertificate**

js

CopiarEditar

const cert = await Certificate.findOne({ cedula });

* Busca en la base de datos MongoDB el certificado por número de cédula.

js

CopiarEditar

res.json({ valid: cert.valid });

* Devuelve si el certificado está válido o no (true / false).

**📋 getCertificate**

js

CopiarEditar

const cert = await Certificate.findOne({ cedula });

* Busca el certificado por cédula y lo devuelve completo como JSON.

**📚 getAllCertificates**

js

CopiarEditar

if (req.user.role !== 'admin') { ... }

* Verifica que solo los usuarios con rol **admin** puedan ver todos los certificados.

js

CopiarEditar

const certificates = await Certificate.find();

* Recupera todos los certificados de MongoDB.

**📄 downloadCertificate**

js

CopiarEditar

const doc = new PDFDocument();

doc.pipe(res);

* Crea un nuevo documento PDF y lo envía como respuesta.

js

CopiarEditar

doc.text(`Instituto: ${cert.instituteName}`);

...

doc.text(`Estado: ${cert.valid ? 'Válido' : 'No válido'}`);

* Escribe la información del certificado en el PDF.

js

CopiarEditar

if (fs.existsSync(logoPath)) {

doc.image(logoPath, ...);

}

* Agrega el **logo** si existe.

**❌ invalidateCertificate**

js

CopiarEditar

const tx = await contract.invalidateCertificate(cedula);

await tx.wait();

* Llama a la función del contrato inteligente para invalidar (anular) el certificado.

js

CopiarEditar

const cert = await Certificate.findOneAndUpdate(

{ cedula },

{ valid: false },

{ new: true }

);

* Actualiza el campo valid del certificado a false en MongoDB.

**📌 Resumen general**

| **Función** | **¿Qué hace?** |
| --- | --- |
| issueCertificate | Emite un certificado (en blockchain + base de datos). |
| verifyCertificate | Verifica si un certificado es válido. |
| getCertificate | Devuelve un certificado por cédula. |
| getAllCertificates | Devuelve todos los certificados (solo admin). |
| downloadCertificate | Genera y descarga un PDF del certificado. |
| invalidateCertificate | Invalida un certificado (en blockchain + base de datos). |

El comando:

bash

CopiarEditar

npm run dev

significa: **ejecutar el script llamado dev que está definido en tu archivo package.json**.

**¿Qué hace exactamente?**

Veamos un ejemplo típico de un package.json en un proyecto frontend (como con **Vite**, **Next.js**, etc.):

json

CopiarEditar

{

"name": "frontend",

"scripts": {

"dev": "vite"

},

...

}

Aquí, cuando tú haces npm run dev, estás ejecutando vite, que:

* Arranca un servidor de desarrollo
* Recarga la página automáticamente al guardar cambios
* Muestra la app en una URL como http://localhost:5173 (u otro puerto)

Otro ejemplo con Next.js:

json

CopiarEditar

{

"scripts": {

"dev": "next dev"

}

}

En este caso, npm run dev lanza el servidor de desarrollo de **Next.js**.

**¿Cómo saber qué hace npm run dev en tu caso?**

1. Abre el archivo frontend/package.json.
2. Busca esta sección:

json

CopiarEditar

"scripts": {

"dev": "aquí irá el comando que se ejecuta"

}

¿Quieres que lo revise por ti? Puedes copiar aquí el contenido de tu package.json o solo la parte de "scripts".

**Dijiste:**

{ "name": "fronend", "private": true, "version": "0.0.0", "type": "module", "scripts": { "dev": "vite", "build": "vite build", "lint": "eslint .", "preview": "vite preview" }, "dependencies": { "axios": "^1.9.0", "react": "^19.1.0", "react-dom": "^19.1.0", "react-router-dom": "^7.6.0" }, "devDependencies": { "@eslint/js": "^9.25.0", "@types/react": "^19.1.2", "@types/react-dom": "^19.1.2", "@vitejs/plugin-react": "^4.4.1", "autoprefixer": "^10.4.21", "eslint": "^9.25.0", "eslint-plugin-react-hooks": "^5.2.0", "eslint-plugin-react-refresh": "^0.4.19", "globals": "^16.0.0", "postcss": "^8.5.3", "tailwindcss": "^3.4.17", "vite": "^6.3.5" } }

**ChatGPT dijo:**

Gracias por compartir tu package.json. Con base en esto, aquí te explico exactamente qué hace cada comando:

**✅ Comando principal:**

bash

CopiarEditar

npm run dev

**¿Qué hace?**

Ejecuta:

bash

CopiarEditar

vite

Esto **inicia el servidor de desarrollo de Vite**, mostrando tu aplicación React en una URL local como:

arduino

CopiarEditar

http://localhost:5173/

Vite es un **empaquetador rápido** para aplicaciones frontend modernas (como React), que recarga en caliente y es ideal para desarrollo.

**📁 ¿Qué contiene tu proyecto?**

Tu carpeta frontend es un proyecto frontend en React, usando:

* **React 19**
* **Vite** como herramienta de desarrollo
* **Tailwind CSS** para estilos
* **ESLint** para análisis estático de código
* **React Router DOM** para navegación de páginas
* **Axios** para peticiones HTTP

**🧠 Otros comandos útiles:**

* npm run build: compila tu app para producción (crea archivos estáticos optimizados en dist/)
* npm run preview: inicia un servidor para **ver el resultado del build**
* npm run lint: analiza el código para encontrar errores de estilo o sintaxis con ESLint

**🚀 En resumen**

Si corres:

bash

CopiarEditar

cd frontend

npm install # instala dependencias

npm run dev # inicia tu app en modo desarrollo (React + Vite)

¡Tu aplicación estará lista para desarrollar y probar en el navegador!

¿Quieres que te ayude a agregar autenticación, rutas, o conexión con smart contracts también?