Proposal for Blockchain-Based Applications Architecture, Case of Study: Electronic Certification System

Angel Rendon[0000−0003−3900−9582]

Universidad Nacional de Colombia
amrendonsa@unal.edu.co
http://unal.edu.co/

Abstract. Blockchain has become an appealing technology for the software industry as the SmartContracts appeared as another of its applications alongside to cryptocurrencies. Technologies such as the Ethereum Virtual Machine (EVM) have opened a whole new landscape for developers to come up with more efficient solutions taking advantage of its qualities (e.g., decentralization, transparency, traceability, auditability, etc).
As the technology gains a more important role, a need for having a set of good practices rises as key element in software development.
This paper aims to explore topics that motivates this work: what has been done at the architectural side, as well as showing our own experience exploring the development of a blockchain certification tool for Universidad Nacional de Colombia's Engineering Faculty extension courses diplomas, using Solidity, deploying the SmartContract to the Ropsten test net (including the tools used for such purpose), and integrating it into a microservices architecture for our architecture proposal.
At the end, this paper will briefly show what are the next stages for the project under analysis and conclusions.

Keywords: Blockchain · Software architecture · Applications · Architectural views.

1 Introduction

Hearing of people faking studies in their curriculum as they are running for public positions is not new news [1] [2] [3] [4] [5]. The question is, **how to efficiently verify such claims?**

[1] https://www.elespectador.com/noticias/bogota/el-tal-doctorado-de-penalosa-no-existe-articulo-625911
[2] https://www.washingtonpost.com/news/politics/wp/2018/08/14/florida-candidate-admits-she-faked-her-college-diploma
[3] https://www.theguardian.com/commentisfree/2015/may/26/mps-lies-sliding-scale-alistair-carmichael
[4] https://www.businessinsider.com/9-people-who-were-publicly-shamed-for-lying-on-their-resumes-2012-5
[5] https://www.nytimes.com/2018/08/13/us/melissa-howard-diploma-florida-gop.html

This question has motivated the development of systems capable of verifying these claims while being protected from attacks and fraudulent operations to make them to appear true [6], as it happens to be our case [7].

Blockchain, then, arises as an attractive option to record the evidences, through automated processes. However blockchain by itself will not achieve the whole user experience and it must be combined with other technologies to ease querying certificates.

This work attempts to analyse considerations on software architecture that includes blockchain as part of a more elaborated solution for this specific case.

2 Related work

The very essence of a certification lies on issuing a statement from certain party to any other else that a set of facts are actually veracious. In a normal certification process there is a **claim** emited by an **issuer** based on **evidences** about a **recipient** (i.e., the entity that is being certified) by means of a **certificate** as a document that attest all of them: the issuer, the recipient, the claims upon evidences as needed [2].

There are three processes involved on certifying [2]:

1. **Issuing**: the process where the issuer emits claims based on evidences about a recipient in form of the certificate itself.
2. **Sharing**: the process where the recipient shares their certificate with any third-party.
3. **Verification**: the process where any third-party verifies the authenticity of a certificate.

However there must be guarantee of having certifications issued by significant trusted parties. Thus, identification of the parties involved gets more importance because third-parties must be able to corroborate by any means such identities [2].

Among the advantages of using blockchain for digital certification there are [2]:

- Verification can be done by anyone with access to the blockchain.
- Certifications will be still valid even if the original issuer no longer exists.
- The only way to lose the certificate information is having all the nodes of the blockchain to delete the network client.
- There are mechanisms to share the signature of the document without sharing the document itself

As an emerging technology, blockchain is still on early stages of development. There are no well defined design patterns to integrate it into formal developments

[6] http://certificates.media.mit.edu/
[7] http://certificados.bogota.unal.edu.co/

yet. By replacing central databases with blockchain, it has been demostrated that data structuration on SmartContracts can affect systems' resilience [7].

In [7] a restructured architecture comprised of four layers (i.e., UI; management; data, or off-chain; and, blockchain, or on-chain) was proposed in order to reflect the traceability process of their use case. The key element is to identify where specific data must be placed (i.e., off-chain or on-chain), for that specific case, sensitive information was placed on-chain (e.g., hashes of documents, small and sensitive information, permission control information), whereas off-chain was left with SmartContracts' addresses, and heavy information (e.g., photos, certificates, even SmartContracts' code).

An esential feature in [7] is the concept of the *factory contract*, that creates, upon certain parameters, different instances of new SmartContracts on-chain. It also avoids modifications on SmartContracts by unauthorized entities.

Due to several configurations and variants systems can adopt depending on the chosen Blockchain technology, [8] proposes a Blockchain Taxonomy in order to classify and compare different blockchain technologies as a support tool to design and assess their impact on software architectures.

Other natural factors of current blockchains (e.g., decentralization, execution calls delays) as well as external factors such as the way users interact with those blockchains, poses high challenges on integrating blockchain to software developments. By means of surveying several Đapps [8], [6] examines their architecture in order to find reocurring architectural patterns.

BSeIn is presented as a conceptual framework [4] to implement reconfigurable smart factories. This framework aims to enforce fine-grained access control policies to secure several authentications. The design of the framework includes Initialization (where the factory manager or a suitable authority initializes the system upon system paramenters), Request Issuance (proving user can publish a reasonable request, generating key pair, encrypting and signing info, broadcasting transaction to the blockchain), Chain Transaction, State Delivery (transactions are broadcasted to the cloud and industrial networks), and Permission Update.

A comparison for computational and storage cost between blockchains and databases was done in [5], by, first, capturing the cost models for both technologies, and second, implementing and measuring the cost of business process execution on both for a specific use case. As result of this investigation, they have conclude that using blockchain affects non-functional qualities of a system with two orders of magnitude of difference between both aproaches. However, this research only focuses on monetary costs losing the decentralization purpose of blockchain.

Other aproaches explore architectures to share information on ledgers with the government as a way to ensure public safety and security. The purpose on [1]

[8] Đ is a glyph used in Old English, Middle English, Faroese, Icelandic and Elfdalian (https://en.wikipedia.org/wiki/Eth). Its pronuntiation would be something like *eth*. In the Etherum context it has been used to refer decentralized applications or "Ethapps", but many people simply pronounce it like "Dapps".

is to devise the requirements and architecture for information sharing between corporations and goverment so it is acceptable to businesses, by implementing a blockchain solution to store events and rules for information shared by companies by cryptographic means. As key elements some considerations have to be ensured, due to communication between Business and Goverment should be a must, as public safety and security. This article attemps to provide a modern solution by desingning and building an architecture such that:

- keeps information confidential between parties
- its Reliable
- its Secure
- allows Goverment to keep track for "suspicious" goods

Possible outcomes or expected results of this implementations are related mainly with reduction of smuggling and detect whether goods need inspection or not [1].

Current software development paradigms have to deal with centralization and their associated problems (i.e., flexibility, efficiency, availability, and security). By integrating blockchain to come up with a P2P network with higher security, scalability and a well-structured cloud system, a system comprised of five layers is presented: resource (physical manufacturing resources), perception (senses physical manufacturing resources), manufacturing (service provider, data hashing service, conection provider), infrastructure (other layers infrastructure supporter) and application (API[9] provider) [3].

3 Case of study: Electronic Certification System

3.1 Background

The Engineering Faculty of Universidad Nacional de Colombia proposed a proof of concept project, through the IEI (Instituto de Extensión e Investigación)[10] associated to the same Faculty, to check the viability of applying the blockchain technology to the certification process.

The first stage of the project considers having a system able to:

- generate the certificates on PDF including a QRCode, thus enabling any third-party to verify the metadata in the associated site.
- enable any third-party to verify the authenticity of a copy of the PDF in the associated site.
- allow University's designed officials to upload a list of students to certify with the according course information.
- notify the students to their personal email addresses attaching the PDF certificate, a link to their information, the QRCode, and the PDF certificate hash.

[9] Application Programming Interface
[10] Research and Extension Institute

The office in charge of certifying the students, La Unidad de Educación Continua y Permanente[11], has several formats and types of courses. The idea was to focus on a single type of course with its respective certification format, and prove the concept in one of the Unit courses. The decision was finally to certify the course 'Blockchain - Creación de Contratos Inteligentes'[12]. This course has a duration of 45 hours distributed on sessions of three hours per day, three sessions per week, for a total of 5 weeks. The course is approved with the 80% of attendance, it does mean 12 of 15 sessions.

3.2 Global architecture

The resulting architecture is shown in Figure 1. There is a service component of the solution provided by *Infura*. It might be considered as a Decentralized Gateway as a Service, because Infura[13] provides access to the IPFS[14] public gateways and Ethereum APIs. Through Infura, it is possible to access all the available Ethereum networks (i.e., MainNet, and Testing networks), the Figure 1 only shows Ropsten, because the SmartContract was deployed there.

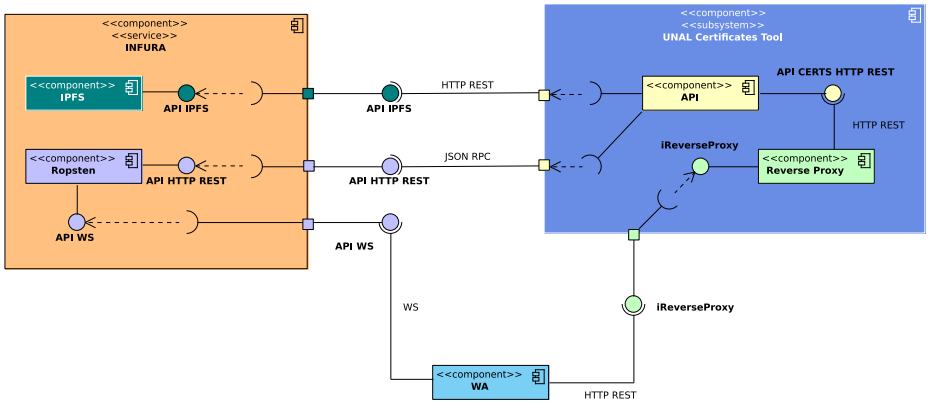

Fig. 1. Microservices architecture for the certifying system.

The solution is encapsulated into the *UNAL Certificates Tool* subsystem, comprised of the backbone API and a Reverse Proxy. The API is the main component in the architecture because it has the business logic and configurations to interact with the Infura service. The Reverse Proxy uses Nginx in a special configuration, so as to filter certain HTTP requests.

[11] Permanent and Continuous Education Unit
[12] Blockchain - Creation of Smart Contracts
[13] https://infura.io/about
[14] InterPlanetary File System https://docs.ipfs.io/introduction/overview/

Finally, there is a client *WA* (Web Application) that connects to the API through the Reverse Proxy, and the Ropsten network using the WebSocket protocol in order to track certain transactions on the Blockchain. This is the client used by any third-party to verify certificates' authenticity. It also allows the designated University's officials to fulfill metadata of the course and upload the list of students to be certified.

WA The WA was developed using VueJS (https://vuejs.org/) as framework. It got the axios (https://github.com/axios/axios) library to perform HTTP requests to the API through the Reverse Proxy. It also got the web3.js (https://web3js.readthedocs.io/en/1.0/) library collection to communicate against Ropsten using the IPC communication protocol.

Reverse Proxy A Nginx configuration script running on a Docker container.

API The API is the most important component in the whole architecture scheme. It has the business logic and certain configurations. It was written in JavaScript with Babel (https://babeljs.io/) integrating Swagger (https://swagger.io/) in order to expose routes as documentation. For communication effects against the Ethereum's Ropsten network, it got the web3.js (https://web3js.readthedocs.io/en/1.0/) library collection.

It has three main modules as it can be seen on Figure 2, the *Sessions* for session handling, *Mailer* for mail construction and transport, and *Certificates* that does all the heavy work on certifying students, the last one uses a series of helpers, such as the *IPFS* which handles all the IPFS operations, *SmartContracts* which interacts through the SmartContract's ABI[15] with the deployed SmartContract on the Ropsten network, *PDFBuilder* which makes the actual PDF file on JS Buffers, and *QRCodeBuilder* that makes QRCodes upon a given text.

This API was built using ExpressJS (https://expressjs.com/) as base middleware, allowing to build the solution with a n-tier architecture as shown on Figure 3. The API has a sequence for valid requests where *Controllers* deal with the initial request, validating authorized operations, the flow goes to the *Services*, where the business logic is coded. The service can request actions from the *Repositories*, where specific configurations are coded for the IPFS and Ropsten networks. The *SmartContract Repository* can access the *Contracts* layer in order to read the compiled SmartContract, specifically its ABI.

3.3 Sequences

Certifying students The Figure 4 shows the flow for certifying a set of students. The University's designated official, prepares a CSV[16] file containing all

[15] Application Binary Interface
[16] Comma Separated Values

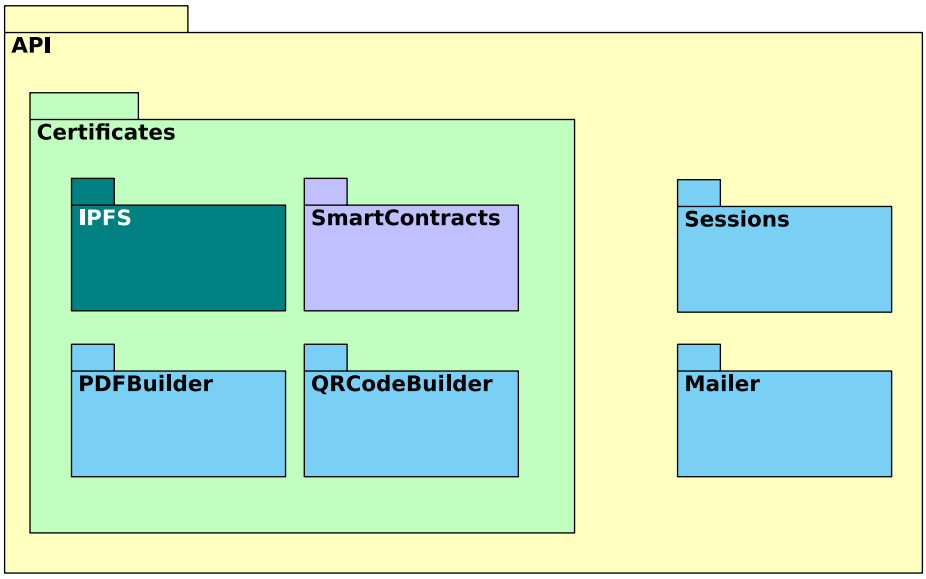

Fig. 2. API decomposition architectural view

the students to be certified in certain course. By fulfilling a form (see Figure 5), the official completes information of the course (i.e. course name, course code, course starting and finishing dates, and hour intensity).

The WA iterates over students, sending a request for certification for each student.

The Reverse Proxy transfer the request to the API, where the data is sanitized, then a unique QRCode is generated in a PNG JS Buffer upon data and a secret salt text. Having the QRCode, the PDF JS Buffer is created upon data and sent to the IPFS network, where the file is going to be hosted, returning its respective hash. The API sends a self signed transaction to the Ropsten network comprised of all the metadata, the IPFS hash and the self generated hash used to create the QRCode. Once the transaction is confirmed, an email is sent to the student with the PDF, its IPFS hash, QRCode and link as attachments.

Parallelly, the WA is listening to the transaction confirmation, once it gets it, starts the process for the next student until the list gets empty.

Verifying certifications by QRCode Verification by QRCode is thought to quick verification with cell phones. It uses the responsive feature of the WA. It's a specific case of sub-sub-section 3.3 (Verifying certifications by link).

Verifying certifications by link One of the forms to verify the veraciousness of the certificate, is accessing via a link (see Figure 6). The link has a code that is queryied by the WA to the API through the Reverse Proxy. The API sends the

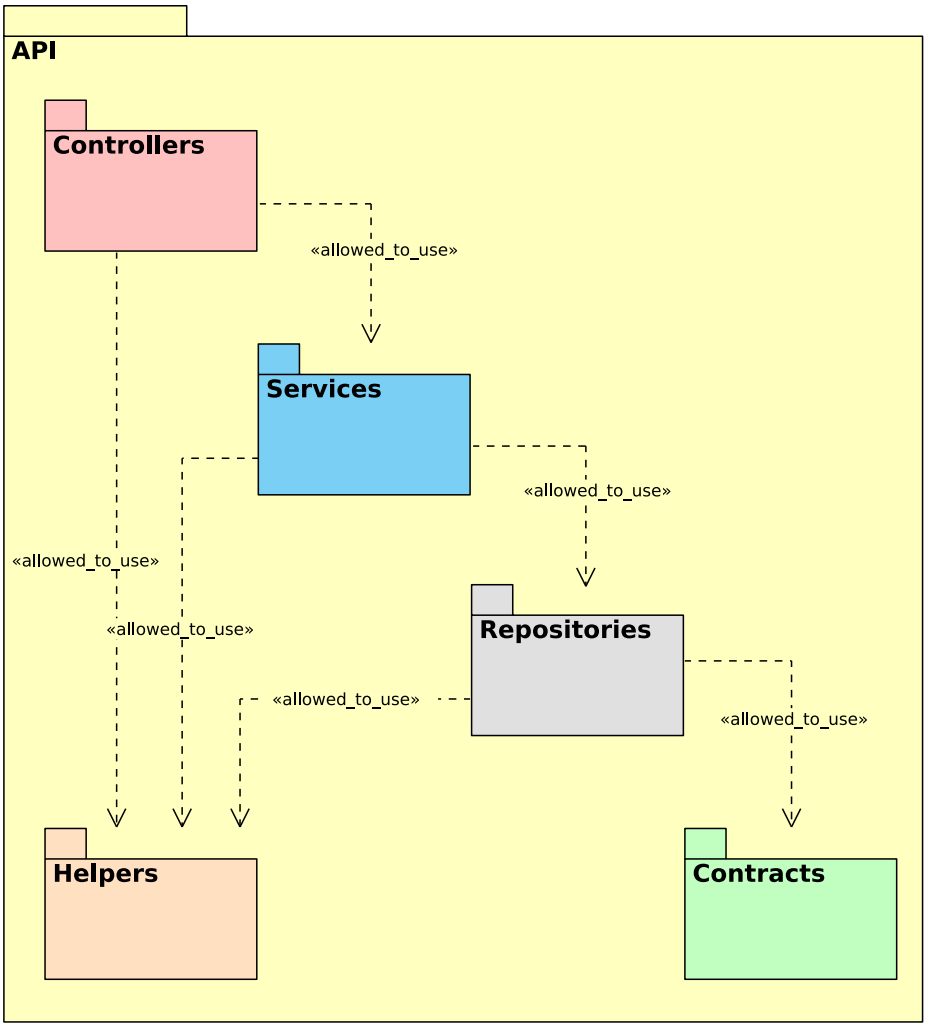

Fig. 3. API n-tier layered architectural view

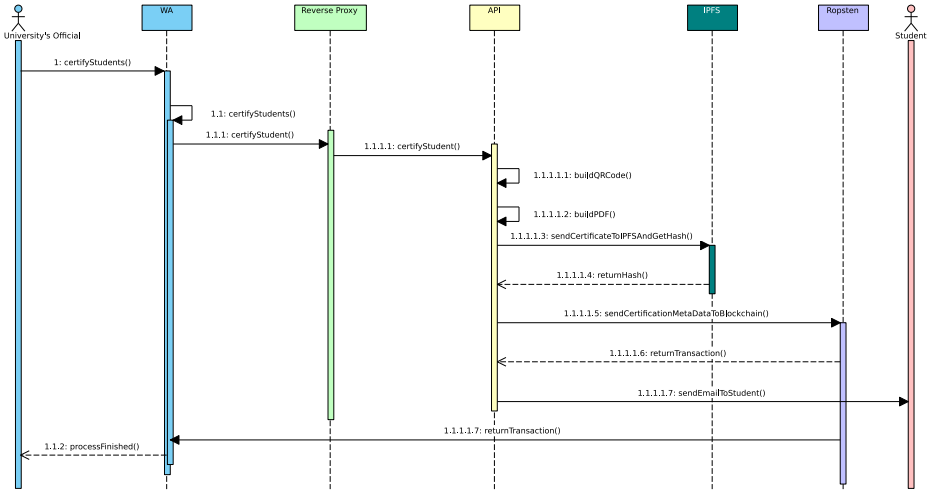

Fig. 4. Certifying students sequence diagram.

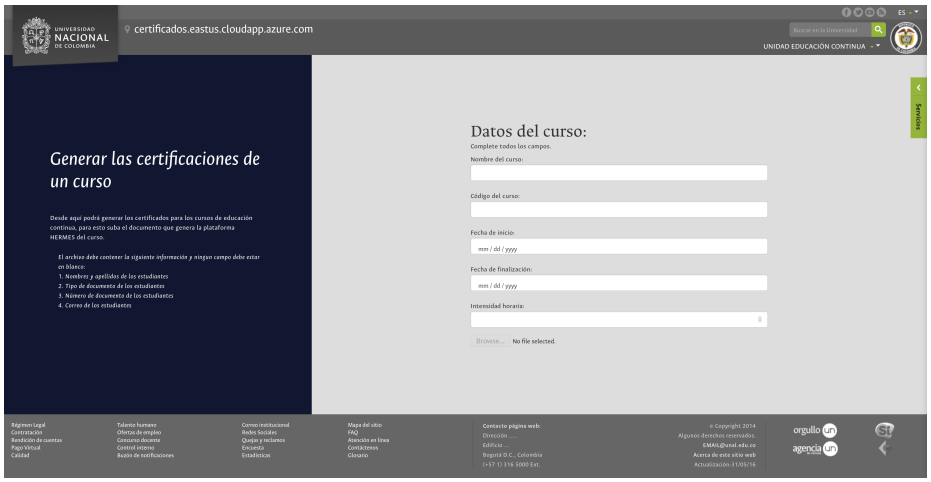

Fig. 5. WA students loader interface.

code to the SmartContract. The SmartContract uses an internal structure to save this link code and binds it to the certificate's IPFS hash (see sub-section 3.4).

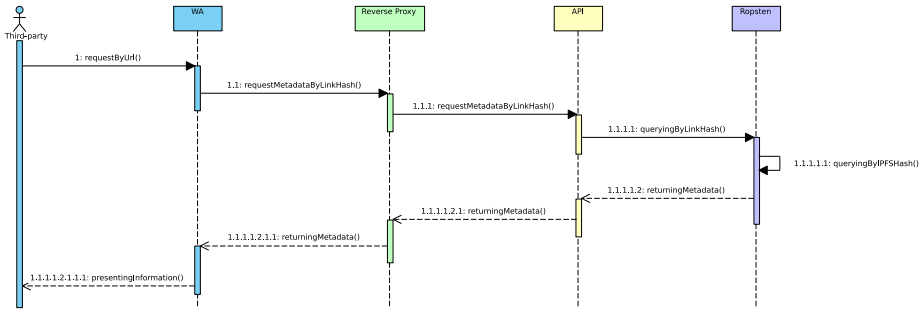

Fig. 6. Verifying certificate information via link.

Verifying certifications by PDF This is the most appealing way to verify authenticity of certificates, because the student can share a copy of their certificate and any third-party can check it through the site.

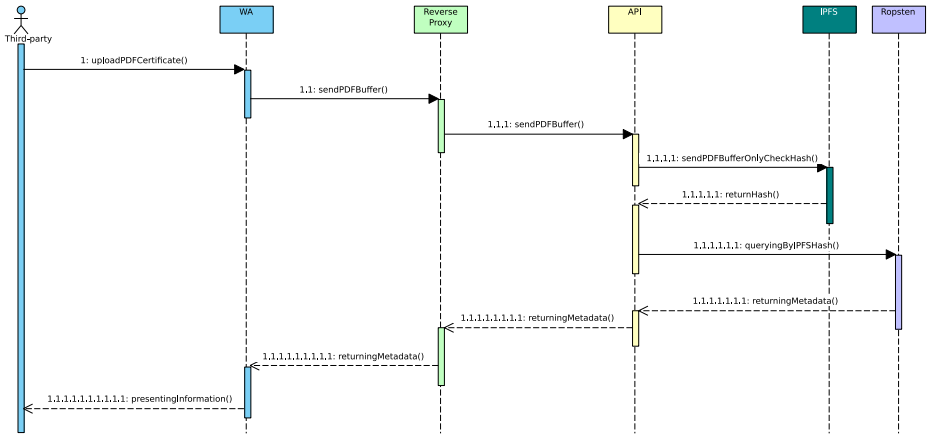

Fig. 7. Verifying certificate information via PDF file.

The third-party will go to https://certificados.bogota.unal.edu.co/ as shown in the Figure 8, will upload the PDF file (see Figure 7), it will go to the API through the Reverse Proxy. The API will send the PDF buffer to IPFS but not to add it, but just to get its hash, once it gets the hash it queries the SmartContract with that hash, returning the metadata associated to it, and presented through the WA.

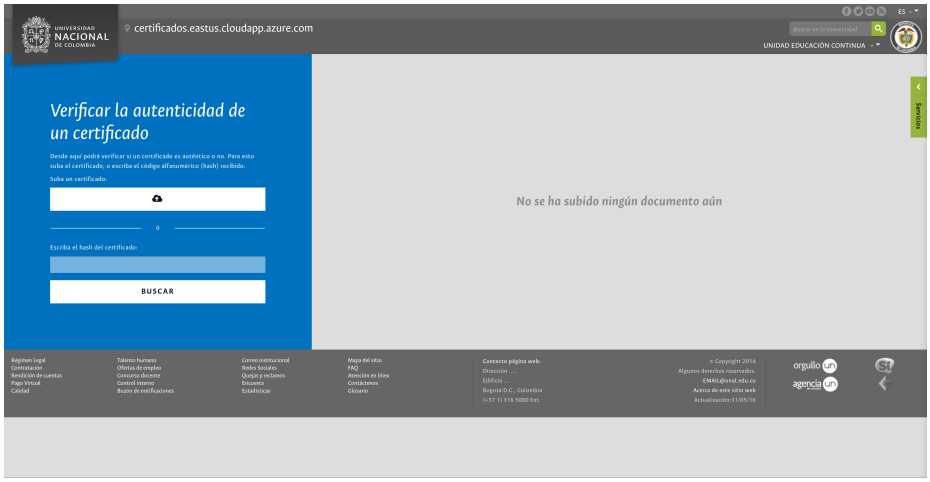

Fig. 8. Universidad Nacional de Colombia's certificates home.

Verifying certificate by hash This is a simple way to verify the integrity by the PDF file's hash. This hash is delivered to the student in the notification email. They can provide the hash text to any third-party, who will paste it in the hash section as seen in the Figure 8. This simply is a simplification of the process shown in the sub-sub-section 3.3 (Verifying certifications by PDF).

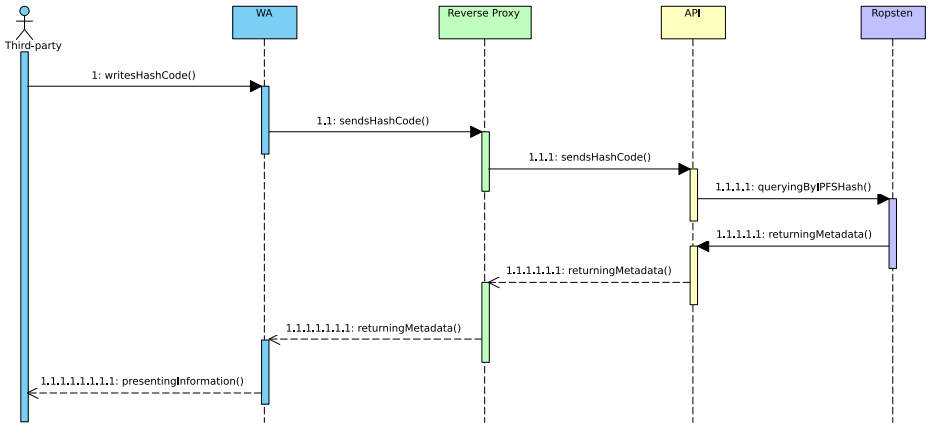

Fig. 9. Verifying certificate by hash code.

Once the third-party enters the hash code, this is delivered to the API through the Reverse Proxy, since this is the very hash of the PDF file, there is no need to go to IPFS, the API can query the SmartContract directly.

3.4 SmartContract

SmartContracts are code that encapsulates logic that allows to persist information into the blockchain. They can also read data from the blockchain at the latest known state (other processess might be necessary in order to trace changes in certain data structures as the blockchain constantly changes its final state).

Nevertheless, there is no formal UML representations for what a SmartContract is, but for the sake of explaining the key elements for this project, the Class diagram will fit to show the SmartContract's data structure (see Figure 10).

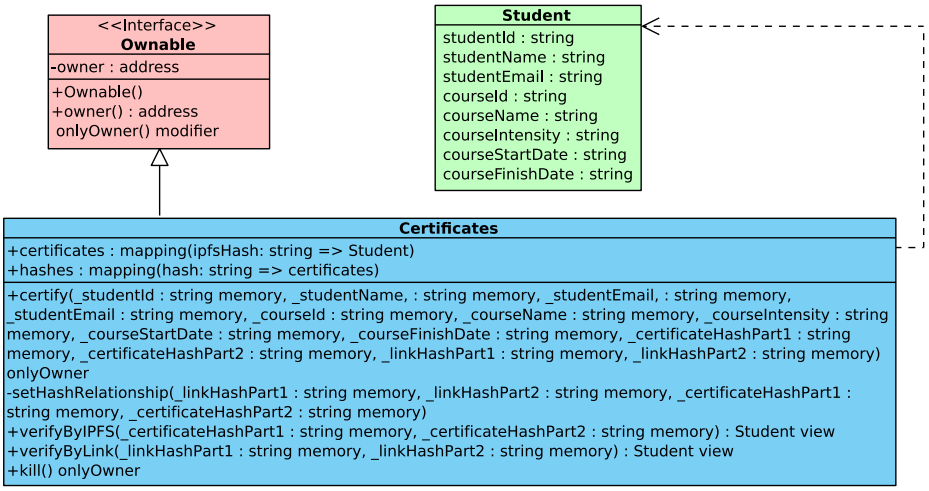

Fig. 10. SmartContract Class diagram

The SmartContract has a struct called *Student* as seen on Code 3.4 (Students struct), to save all the certificate metadata. Such structure is used on the contract in a mapping as shown in the Code 3.4 (SmartContract's mappings).

Students struct

```
1  struct Student {
2    string studentId;
3    string studentName;
4    string studentEmail;
5    string courseId;
6    string courseName;
7    string courseIntensity;
8    string courseStartDate;
9    string courseFinishDate;
10 }
```

SmartContract's mappings

```
mapping(bytes32 => mapping(bytes32 => Student)) certificates;
mapping(bytes32 => mapping(bytes32 => string[2])) hashes;
```

The reason to save students within a nested mapping is due to strings in Solidity: they are considered as dynamic-size arrays, and they poses problems not only as key for mappings, but also in response tuples for functions[17]. The IPFS hashes are string of 46 characters length, so a bytes32 data type was not able to hold the hash, making the API to send the IPFS hash in two chunks.

As it was explained before on sub-sub-sections 3.3 (Verifying certifications by QRCode) and 3.3 (Verifying certifications by link), it is possible to retrieve the information from the SmartContract using a code. That code reads from the *hashes* mapping, taking the information from the *certificates* mapping as the Code 3.4 (Verification Methods) shows.

Verification Methods

```
function verifyByIPFS(
    string memory _certificateHashPart1,
    string memory _certificateHashPart2
  ) public view returns (
      string memory,
      string memory,
      string memory,
      string memory,
      string memory,
      string memory
    ) {
    bytes32 part1 = _certificateHashPart1.stringToBytes32();
    bytes32 part2 = _certificateHashPart2.stringToBytes32();
    Student memory student = certificates[part1][part2];
    return (
    student.studentId,
    student.studentName,
    student.courseName,
    student.courseIntensity,
    student.courseStartDate,
    student.courseFinishDate
    );
  }

  function verifyByLink(
    string memory _linkHashPart1,
    string memory _linkHashPart2
```

[17] This contract was compiled with the pragma 0.4.24. As the documentation states for version 0.5.7, it is now possible to use string as key for mappings (https://solidity.readthedocs.io/en/v0.5.7/types.html#mappings)

```
  ) public view returns (
      string memory,
      string memory,
      string memory,
      string memory,
      string memory,
      string memory
    ) {
    bytes32 part1 = _linkHashPart1.stringToBytes32();
    bytes32 part2 = _linkHashPart2.stringToBytes32();
    string[2] memory IPFSHashes = hashes[part1][part2];
    return verifyByIPFS(IPFSHashes[0], IPFSHashes[1]);
  }
```

The way to bind both mappings is shown in the Code 3.4 (Mappings binding).

Mappings binding

```
function setHashRelationship(
    string memory _linkHashPart1,
    string memory _linkHashPart2,
    string memory _certificateHashPart1,
    string memory _certificateHashPart2
  ) private {
    hashes[_linkHashPart1.stringToBytes32()]
      [_linkHashPart2.stringToBytes32()] =
      [_certificateHashPart1, _certificateHashPart2];
  }
```

Finally, in order to properly query all the certificates of a student an additional information is stored alongside the metadata in the transaction: an event that allows to query by the student id number as the Code 3.4 shows (Student certified event).

Student certified event

```
event studentCertified(
    bytes32 indexed _studentId,
    bytes32 _part1,
    bytes32 _part2
  );
```

4 Future work

Coming work for this project will focus on improving the SmartContract by integrating features on newer versions of Solidity. The contract will be killed after migrating transactions into a Production network.

References