

Implementation of a workbench platform for the management of smart contracts in BlockChain nodes on Azure Cloud

Implementación de una plataforma workbench para la gestión de smart contracts en nodos de BlockChain sobre Azure Cloud

DANIEL-MARTÍNEZ, Wendy†*, SANTANA-VALADEZ, Luis Alejandro, ZAMUDIO-GARCÍA, Víctor Manuel and HERNÁNDEZ-PÉREZ, Faride

Universidad Politécnica Metropolitana de Hidalgo – Ingeniería en Tecnologías de Información

ID 1st Author: Wendy, Daniel-Martínez / ORC ID: 0000-0002-4455-940X, CVU CONACYT ID: 330244

ID 1st Co-author: Luis Alejandro, Santana-Valadez / ORC ID: 0000-0003-1561-020X

ID 2nd Co-author: Víctor Manuel, Zamudio-García / ORC ID: 0000-0002-4660-8025, CVU CONACYT ID: 482212

ID 3rd Co-author: Faride, Hernández-Pérez / ORC ID: 0000-0001-9426-4944, CVU CONACYT ID: 557262

DOI: 10.35429/JIT.2022.27.9.38.50

Received: August 12, 2022; Accepted December 30, 2022

Abstract

Objectives: Analyze, design and build backend components (web API, XML web service and database) and frontend (website) with Microsoft technology that allow the integration of a functional workbench platform to automate the launch of compiled smart contracts in blockchain nodes hosted in Azure Cloud so that they can be used by companies in general. Methodology: The SCRUM methodology was applied for the agile development of the technological products of the backend and frontend, because it adapted very well to the nature of this research, since in each iteration there was always an update of the tools, programming components, testing and configuring services to achieve goals. Contribution: It is shown that the development platform integrated by the solidity language, Azure cloud blockchain, the Visual Studio .Net IDE and the SQL Server database manager allowed to design and build a blockchain workbench platform easily exploitable by companies to validate their processes and generate transparency in the handling of their information.

Blockchain, Backend, Frontend

Resumen

Objetivos: Analizar, diseñar y construir componentes backend (web API, servicio web XML y base de datos) y frontend (sitio web) con tecnología Microsoft que permitan integrar una plataforma workbench funcional para automatizar el lanzamiento de contratos inteligentes compilados en nodos de blockchain alojados Azure Cloud para que puedan ser utilizados por empresas en general. Metodología: Se aplicó la metodología SCRUM para el desarrollo ágil de los productos tecnológicos del backend y frontend, ya que se adaptó muy bien a la naturaleza de esta investigación, ya que en cada iteración siempre hubo una actualización de las herramientas, componentes de programación, pruebas y configuración de servicios para lograr los objetivos. Contribución: Se demuestra que la plataforma de desarrollo integrada por el lenguaje solidity, blockchain de Azure cloud, el IDE de Visual Studio .Net y el manejador de bases de datos SQL Server permitió diseñar y construir una plataforma workbench de blockchain fácilmente explotable por las empresas para validar sus procesos y generar transparencia en el manejo de su información.

Blockchain, Backend, Frontend

Citation: DANIEL-MARTÍNEZ, Wendy, SANTANA-VALADEZ, Luis Alejandro, ZAMUDIO-GARCÍA, Víctor Manuel and HERNÁNDEZ-PÉREZ, Faride. Implementation of a workbench platform for the management of smart contracts in BlockChain nodes on Azure CloudJournal Information Technology. 2022. 9-27: 38-50

* Author's Correspondence: (wdaniel@upmh.edu.mx)

†Researcher contributing as first author.

Introduction

Currently, smart contract technology based on the management of blockchain nodes has increased the benefits generated for the companies that consume them, but there is also a growing need for a better way to design them, validate them and finally submit them to blockchain in order to take advantage of the functionality of transparency in information transactions and monitoring their evolution over time.

Public and private companies in various countries that have different business models (lucrative or not) have already generated their own technologies, but also in some cases already use technologies based on blockchain nodes and smart contracts on custom designs and that regularly, has a high cost of design and development, without contemplating the rent of blockchain services that also depend on a stable technological infrastructure; the above invariably implies an increase in the total cost of the project (Hegedus, 2018).

However, it is currently considered that blockchain and smart contract technologies do not yet have a profound impact on open source or proprietary development environments to be able to integrate a development platform to design and build backend and frontend environments, resulting in a dependency of companies on software factories to manage this work behind closed doors (Nowinski and Kozma, 2017).

This research work aims to demonstrate that a technology project can have a flexible development platform, integrating the following added values over current applications based on blockchain and smart contracts:

- It is based on open access and open consumption technology (http request, web APIs, web services, websites) so that it can be used by any company regardless of its infrastructure.
- It applies security of use on administrator, business and operational users for each company.
- The backend consists of a relational database for smart contract management including a satellite data module to customise the information for each company. Within the web APIs and web services considered in the backend, there is middleware that allows the control of smart contracts, their launch to blockchain and the control of gas consumption.
- Finally, the use of a Microsoft Azure blockchain node is contemplated, as well as the hosting of the web platform and databases of the same owner, in order to publish the backend and frontend included in the project.

The result of generating a comprehensive platform for the management of a blockchain node applying open middleware (web APIs and web services), allows to have two communication channels: 1) direct to http request interfaces and 2) the frontend to be used by companies and their users; however, this objective allowed generating a flexible blockchain and smart contract workbench, with the lowest rental cost of services in the Azure cloud so that it can be customised to each company and this allows them to generate all the smart contracts they need to consume them in their strategic processes that they must register in blockchain for secure, public and permanent tracking.

The first section describes the agile methodology that was used for the development of this project, the second section mentions the analysis, design and development of the backend that integrates the database and its satellite module, as well as the web service and the web API necessary to develop the open middleware for http requests that allow communication with the Azure blockchain node and the database. Finally, the third section describes the design and development of the frontend that will consume in a structured way the web services and http requests through interfaces that allow the management of the processes in a graphical and secure way. Based on the categorisation of MARVID Mexico, the present work corresponds to area VII of Engineering, in the field of Engineering in the discipline of Systems Engineering, to be applied in the sub-discipline of Information Systems.

1. Description of the applied Methodology

For the development of this applied research the agile methodology Scrum was used because it was the one that was best adapted to the way of working and conditions of the technological products that were generated; an important aspect is that the project had to be adapted to the circumstances of the context of blockchain, as it always depended on the research and development options that were available based on the development platform chosen, to apply iteratively design, coding and testing, to finally implement each iteration.



Figure 1 Stages of the Agile Methodologies (Retamosa Santos, 2015)

In order to validate each iteration of the methodology, it was applied in the context of a university, which needs to generate a smart contract to record information in blockchain about the students who have accredited the training courses it offers, in order to generate valid digital certificates that can be consulted at any time by educational institutions or companies.

2. Analysis, design and development of the Backend

The development platform was selected based on the tools and services applied to smart contracts and the blockchain node, mainly because of the openness found to configure the node and the programming components provided at that time by the selected provider: Microsoft. The list of tools was as follows:

Blockchain tools and components

- Solidity Language (compilation of smart contracts on Ethereum Blockchain)
- Ganache - Truffle Suite (local testing platform for Blockchain nodes)
- Azure Transaction Node - Blockchain
- Nethereum.Web3 and NewtonSoft.Json (Libraries for smart contract interaction, wallet creation and json object manipulation)

Backend tools and services

- Microsoft Visual Studio .Net 2017 Developer (IDE for web API and web service development)
- Microsoft SQL Server 2012 service pack 4
- Server language C#
- Communication standards: XML and Json
- Azure Blockchain node
- Hosting for Web server with ASP .Net technology and for SQL Server DB server, with Microsoft as provider (Plesk Onix Web Host was used).

The backend is made up of the following technology products:

- Blockchain Node (Azure Cloud)
- Web API (Visual Studio .Net 2017)
- Support database and satellite data module (SQL Server 2012 service pack 4)
- Web Service (Visual Studio .Net 2017))

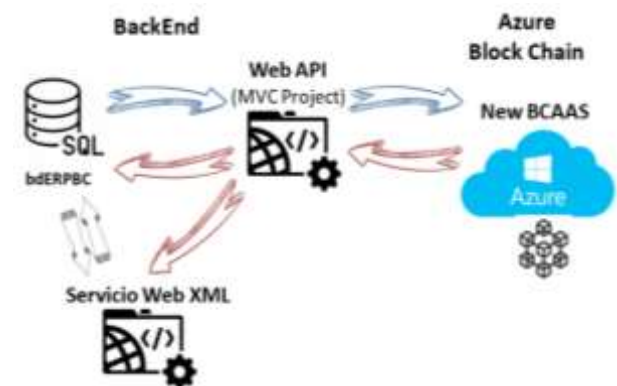


Figure 2 Azure Blockchain functional diagram with backend
Own Design, 2022

The following is a description of the analysis, design and development applied for each backend product.

Blockchain node (Azure Cloud)

The project started with the establishment of the business rules to be able to create the smart contracts, compile them and test launches in local blockchain nodes (Ganache); once the smart contracts were validated and executed in local mode, a real node was generated in Azure Blockchain to test launches of the same contracts.

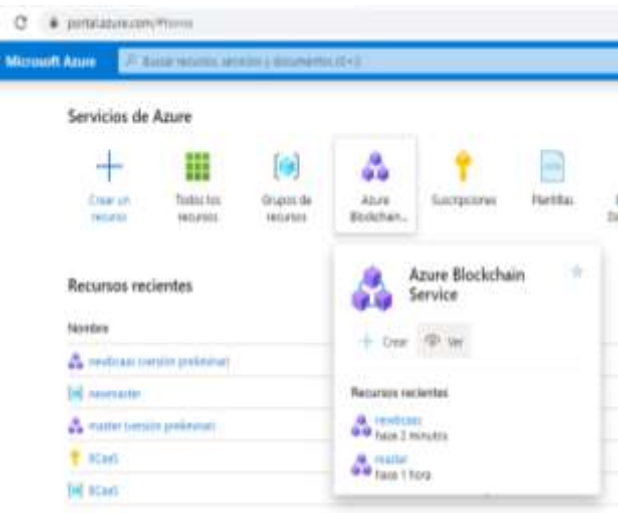


Figure 3 Azure Blockchain Node Configuration
Own design, 2022

The following activities were carried out:

- a. The smart contracts were previously designed and compiled in the solidity language in order to have the base files with the required functions (registration, reading, counts, events). Finally, there will be 3 files that will support the backend:
 - sol file - This will contain the source code of the smart contract.
 - Json file - Will contain the base structure of the smart contract (the interface of functions, events, structures, attributes and their public/private visibility).
 - Json file - Will contain the compiled code of the smart contract (programmed in solidity).

- b. Having controlled the correct versions of the smart contracts in Json files, the next step was to create a blockchain node in Azure, to configure it and publish it on the internet as a licensed service (only for administrator users). We managed to create wallets to later launch the compiled smart contracts both locally (Ganache) and on the Azure blockchain node, being successful both in the contract launches and in the execution of its functions of reading and writing blocks.

Web API (Visual Studio .Net 2017)

Once the process of compiling smart contracts and testing the launches and executions on the actual blockchain node in Azure was secured, we proceeded to generate the web API in Visual Studio .Net 2017 to achieve communication with the Azure Blockchain node and automatically execute the launches and also the methods defined in the smart contract. The rules applied for the design of the web API are as follows:

- a. The Nethereum.Web3 and Newtonsoft.Json libraries were included in the web API configuration to ensure the connection with the Azure Blockchain node and to manipulate the input and output parameters through Json-like structures. The following figure shows the <installed> status of the Nethereum.Web3and Newtonsoft.Json libraries applied in the web API project:

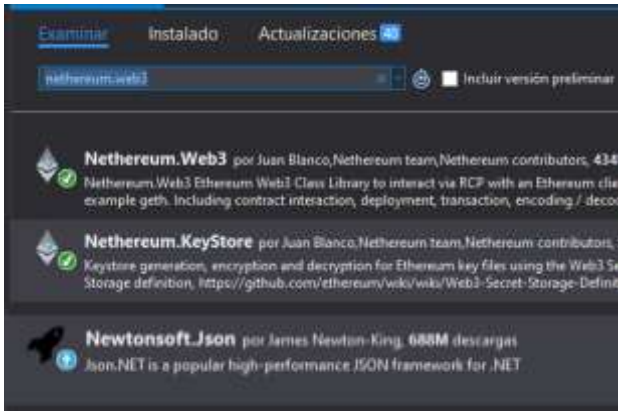


Figure 4 Installation of the Nethereum and Newtonsoft libraries in Visual Studio .Net
Own Design, 2022

The following figure shows the references to the libraries implemented in the configuration web page and in the class files implemented in the developed API.:

```
// librerías referenciadas Nethereum y Newtonsoft
// Controlador de la Web API <SmartContractController.cs>
using Nethereum.Web3;
using Nethereum.Hex.HexTypes;
using Newtonsoft.Json;
using Newtonsoft.Json.Linq;
using System.Threading.Tasks;
using System.Numerics;

using APILanib.Models;

namespace APILanib.Controllers
{

```

Figure 5 Import of the Nethereum and NewtonSoft libraries into the web API driver
Own Design, 2022

- b. The web API methods were created to receive the Json files that integrate the compiled smart contract and the base structure, to validate them through a syntactic and semantic analysis as they must have the same definitions to ensure a successful launch to the Azure blockchain node and finally create a new blockchain based on the sent contract. For this, the connection data to the node must also be configured as follows:

```
// Clase de aplicación global para definir los items para la conexión
// al nodo de Azure Blockchain
public class Global
{
    public static string url = "https://newspace.blockchain.azure.com";
    public static string port = "3000";
    public static string publicKey = "sqooKilx18_Gxw8Tph2pBA";
}

```

Figure 6 Azure blockchain node connection attributes from the web API
Own Design, 2022

- c. The web API methods were also built to send the execution of the methods defined in the smart contract that has been launched, as they can receive parameters or not and can also send varied output results (mapping of the whole blockchain, a single block or specific data of a localised block), thus the following methods were generated:

```
// Definición del método para validar y hacer lanzamiento
// del contrato inteligente
[HttpPost]
[HttpGet]
[Route("api/Contract/Deploy")]
public JObject DeployContract([FromBody] Contrato contract)
{
    System.Net.ServicePointManager.SecurityProtocol = SecurityProtocolType.Tls12;
    int ban = 0;
    JObject res = new JObject();
    JObject datosTrasaccion = new JObject();
}

```

Figure 7 Definition of the DeployContract method in the web API for launching smart contracts
Own Design, 2022

The following is the header of the Json file containing a smart contract successfully compiled with the Solidity language:

```
{
  "contractName": "certificadoDigital",
  "abi": [
    {
      "inputs": [],
      "payable": false,
      "stateMutability": "nonpayable",
      "type": "constructor"
    },

```

Figure 8 Header of a smart contract compiled with solidity in Json format
Own Design, 2022

- d. Tests were designed to validate the connection of the web API in local mode with the Azure blockchain node, which were successful. At this point the release version of the web API was generated so that it could be deployed on a compatible web server. The next step was to mount the web API on the Plesk Onix host, validating the ASP.Net versions and the http request execution configuration to be supported by the server. The execution tests were applied again (creation of wallets, launching of smart contracts to create blockchains and finally execution of methods for block registration, partial and total readings of the blockchains), which were again successful, but now on the host.

```
{
  "ban": "1",
  "-msg": [Array[1]
    0: "La cuenta fue generada exitosamente"
  ],
  "newAccount": "0xd6e8167021b35660d4f40879452f1d0caf65f1e2"
}

```

Figure 9 Message returned by the Web API on execution of the method to create a wallet
Own Design, 2022

The following figure shows the control panel of the Plesk Onix host where the web API was installed and configured (and subsequently the web service and the project database were also installed):

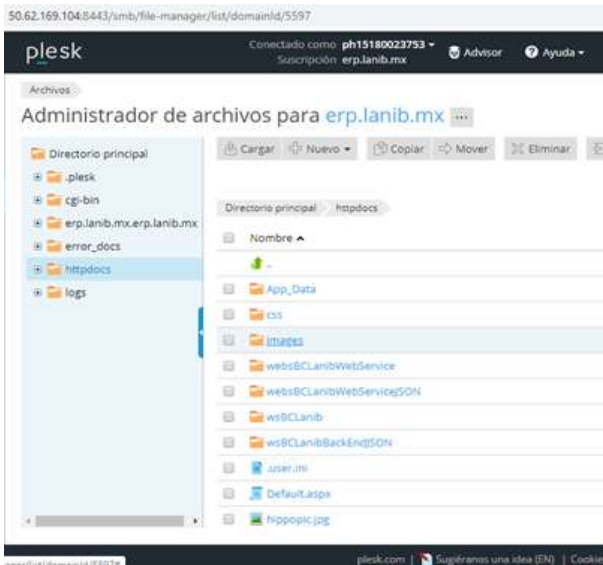


Figure 10 Plesk Onix Host Control Panel
Own Design, 2022

Supporting database and satellite data module (SQL Server 2012 service pack 4)

Having already implemented the Azure blockchain infrastructure and the web API on the host to manage wallets, launch smart contracts and execute blockchain methods, the next step was the generation of the data layer in Microsoft's SQL Server 2012 sp4 database manager, which allowed the management of smart contract information, registration and control of their launches, as well as the execution of the contract methods to manipulate all the blockchains that need to be created; allowing any organisation (regardless of its core business model) to have its own information, users and security in the management of its data. The objective of designing the database to cover the aforementioned requirements was the integration of a module of entities called "satellite", which allowed to store Json formats in its structure so that each company can link additional information to each smart contract launched to the Azure blockchain node and thus achieve to have the official information in blockchains (not modifiable) and also additional data of each block in the satellite database (modifiable).

Finally, stored procedures were generated to manage the database tables and the "satellite" module to encapsulate the data layer and allow the business rules layer to have a secure connection and execution of the processes; finally generating 63 stored procedures.

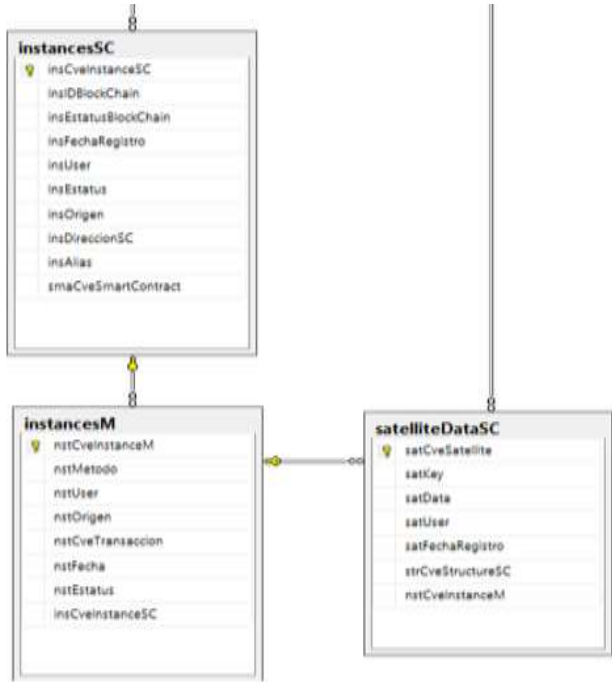


Figure 11 "Satellite" database entities module in-house design, 2022

Having the database script and data objects created locally, we proceeded to its migration on the Plesk Onix host under the same version of the database manager (SQL Server 2012 sp4 from Microsoft) succeeding in the creation of all data objects, finally we created the corresponding connection string to apply it in the web service layer.

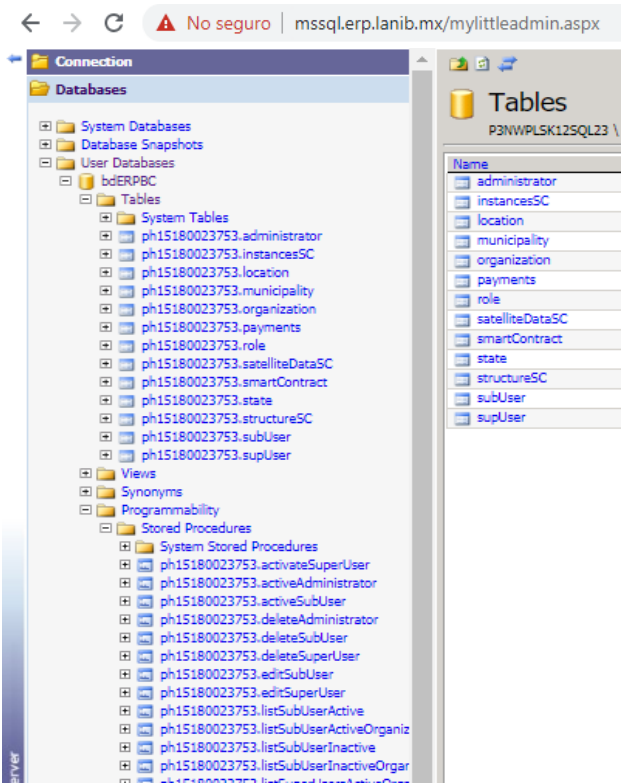


Figure 12 mylittleadmin control panel for Plesk Onix host databases
Own Design, 2022

Web Service (Visual Studio .Net 2017)

Up to this point we have managed to configure and manage the Azure blockchain node through a web API both published on the internet, as well as the design and construction of the database that will have the structure of the information that will be managed by companies, universities or any organisation that requires to launch smart contracts to the Azure blockchain node to generate blockchains on the information they generate and give official validity.

The next step is the last one in the backend and consisted of designing and developing an additional business rules layer to have an alternative connection to the Azure blockchain node and the satellite database but now from websites and mobile platforms: an XML web service to encapsulate the execution of the web API and the web methods needed to manage the execution of the database's stored procedures.

The XML web service aims to centralise the execution of the web API and the database management to support the frontend platform (web and mobile) that will be responsible for giving a complete view of the workbench platform operation to companies, universities or any organisation that needs to implement Azure blockchain in their IT processes.

The construction of the XML web service had two important modules:

- a) Configuration of the necessary references:
 - Reference to the public url of the Plesk Onix host web API.
 - Connection string to the database and its satellite module, located on the Plesk Onix host.
 - Reference to the connection url to the Azure blockchain node.

```
<connectionStrings>
  <add name="cnn" connectionString="data source=184.168.194.77; Initial Catalog=bdERPBC; User Id=ph15180023753; Password=Chintelolo.1" providerName="System.Data.SqlClient" />
</connectionStrings>
```

Figure 13 Plesk Onix host database connection string
Own Design, 2022

- b.-) The encapsulation classes with all the web methods that handle communication with the web API and the database

```
[WebMethod]
public String SmartContractDeploy(string json)
{
    JObject obj = JObject.Parse(json);
    var web3 = new Web3("https://newcas.blockchain.azure.com:3000/-?poolKey=Io Dv4Gf9t2ok4");
    var accountPublicKey = obj["account"].ToString();
    var accountPassword = obj["password"].ToString();
    Task<bool> unlockResult = web3.Personal.UnlockAccount.SendRequestAsync(address:accountPublic
    unlockResult.Wait();
    string abi = obj["abi"].ToString();
    var bytecode = obj["bytecode"].ToString();
    JArray parametros = JArray.Parse(obj["parameters"].ToString());
    JObject validation = validationatos(abi, parametros);
}
```

Figure 14 XML Web Service "SmartContractDeploy" web method
Own Design, 2022

3. Frontend design and development

This section describes the design and functionality applied in the presentation layer built to consume the XML web service of the backend in order to give users of this Azure blockchain workbench platform control and security over the handling of their information; it also covers user types, permissions, tracking of smart contract releases and blockchain creation for their internal processes.

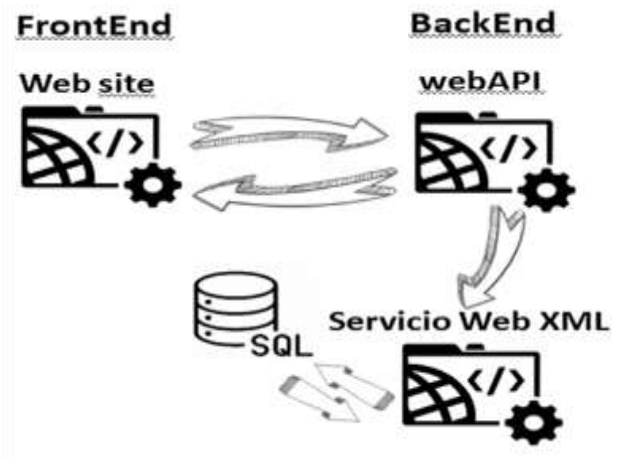


Figure 15 Functional diagram of frontend and backend
Own Design, 2022

The presentation layer is a website that covers the following design and functional characteristics:

- a) Contains the configuration in XML format of the secure connection string for the migrated database on the Plesk Onix host.

- b) Contains the XML web service and web API references in the same configuration file, to be applied in the method encapsulation classes, this ensures the information and execution of the XML web service or web API.

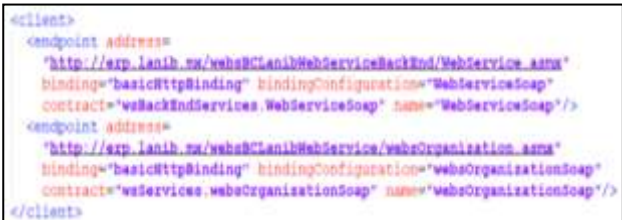


Figure 16 Frontend references for XML web services
Own Design, 2022

- c) The design of the web interfaces is functional, practical and responsive to be deployed on any display dimension (standard PC, tablet or smartphone) as it is based on the integration of the bootstrap framework:



Figure 17 Frontend web design
Own Design, 2022

- d) It manages three types of users, a) Administrator of the workbench platform, b) Superuser for the T.I.C. or administration area of the organisation, and c) Subuser to be assigned to operational employees.

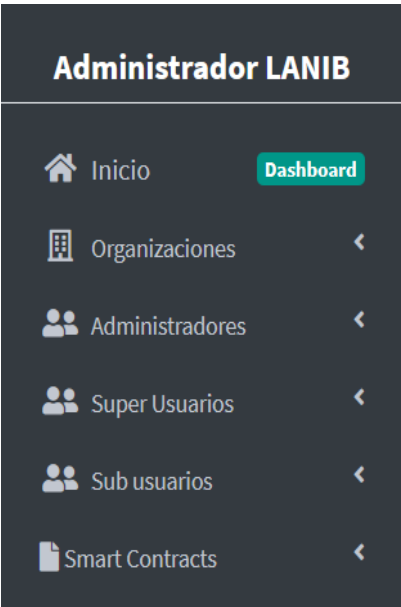


Figure 18 Administrator's options menú
Own Design, 2022

- e) It contains a menu of functions to configure the company and the generation of its password wallet on the Azure blockchain node which will be unique for each company registered on the blockchain workbench platform, finally each smart contract release and blockchain creation will be linked to the enterprise wallet for monitoring.

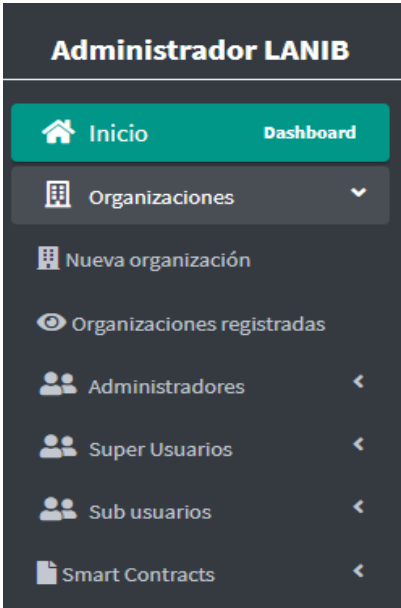


Figure 19 Detailed options for organisations
Own Design, 2022

- f) The "smart contracts" menu options have full tracking over the launch of a smart contract in a company, to be used by superusers or sub-users of each registered company.



Figure 20 Interface for registering and uploading smart contracts
Own Design, 2022

- g) It includes a register and control for each smart contract launched to the Azure blockchain node to generate blockchains and also has a control over the execution of the methods of each blockchain, allowing to apply an online analysis of the execution requirements depending on each defined smart contract (the web interface for methods execution is dynamic and is configured so that the user captures the necessary parameters for validation of data types and correct execution).



Figure 21 Interface for launch tracking and method execution of all smart contracts
Own Design, 2022

Each successful launch of a smart contract to the Azure blockchain node will allow the execution of the methods it contains to generate the required blockchains of the process the company needs to secure. The following figure shows the dynamic interface used for smart contract launches on the frontend:



Figure 22 Interface for configuring the launch of a smart contract
Own Design, 2022

4. Results

Backend analysis, design and development

- a) It was possible to build a programming algorithm to manipulate a smart contract compiled in the solidity language; also allowing the application of parsing in its structure so that a web API can launch it to an Azure blockchain node and thus finally generate blockchains with minimal or no gas consumption.
- b) Nethereum.Web3 and Newtonsoft.Json libraries were successfully applied in the Visual Studio .Net IDE to create web API and web services projects to communicate with Azure blockchain nodes.
- c) Automated the launching of smart contracts to Azure blockchain nodes based on the Json file of the contract previously compiled in solidity language and also the Json format of the base structure of the contract in order to apply the syntactic analysis that validates the content of methods, attributes, events and mappings of the contract.
- d) A web API was built to communicate with the Azure blockchain node to validate the execution of methods via http request with input parameters in Json format; this process allowed the automation of the launching of smart contracts and the execution of methods integrated in the generated blockchains.

- e) A database was created to manage the information of the companies that will consume the services of the workbench platform, however, a module of "satellite" entities was also integrated to ensure that the smart contracts only register the official information that will not change and finally in the "satellite" module, the information related to the blockchain is registered, which can be modified, as it will be stored in Json format.
- f) An XML web service was implemented that allowed the execution of all the methods of the built web API and also integrated the execution of the methods that manipulate the database and the "satellite" module included in the data layer.
- g) Successfully generated and configured an Azure blockchain node in Azure Cloud; to access it with the call and send methods integrated in the execution of smart contract launches.
- h) The web API, XML web service and database were published on the Plesk Onix host using a configuration compatible with the Microsoft development platform that was used to build the technology products of this research.

Frontend design and development

- a) The website was built with the frontend function for the blockchain workbench platform by applying client frameworks such as bootstrap to generate the necessary web interfaces.
- b) Successfully migrated and configured the website on the Plesk Onix host to publish to the internet and allow communication with the web API and XML web service, configured on the same host.

5. Annexes

This section shows evidence of source code and data models that support the design and execution of the backend (web API and web service XML), and also shows evidence of execution on the services built via http request to check the correct validation of the smart contracts before their launch or during the execution of their methods.

The following figure shows the structure of the Json format to send to the web API the structure that the smart contract compiled in solidity should have, this allowed the XML web service to validate the execution interfaces of the methods it contains.:

```
String jsonSmartContract = @"{
  'name' : 'Certificado',
  'consParams' : [ ],
  'metodos' : [
    { 'nombre' : 'creaCert',
      'params' : ['id':'string'],
      'satelliteIn' : [ 'key': ['id'],
        'data' : [ 'nombrealumno' : 'string',
                    'nombrecurso' : 'string',
                    'fechaIngreso' : 'string',
                    'fechaVigencia' : 'string'
                  ]
        }
      ],
    { 'nombre' : 'consultaCert',
      'params' : [ 'id' : 'string'],
      'returns' : 'string'
    }
  ],
  'pubData' : [ ]
}";
```

Figure 23 Json structure of a university's compiled smart contracts for digital certificate control
Own Design, 2022

The following figures show the results of the execution of two methods of the web API via http request, the outputs are handled in Json format in order to send the data resulting from the validations defined in the method; the parameters received from Azure blockchain that integrate the execution status in the node are also integrated: a.-) the first figure indicates the successful registration of a wallet in blockchain and b.-) the second figure shows the successful launch of a smart contract to blockchain

```
{
  "ban": "1",
  "-msg": [Array[1]
    0: "La cuenta fue generada exitosamente"
  ],
  "newAccount": "0xd6e8167021b35660d4f40879452fd0caf65f1e2"
}
```

Figure 24 a.-) Json resulting from the web API for registering a wallet on blockchain
Own Design, 2022

- b) Definitely, blockchain being a new and abstract technology in its conception and operation, there are very few sources of information and reliable documentation from the technological point of view to be able to have reference (Ahram, Sargolzaei, Sargolzaei, Daniels and Amaba, 2017). However, at the end of this research there are no technological products with the characteristics of workbench platform applying blockchain, so this applied research will serve as evidence of the widespread implementation of blockchain nodes with the lowest cost of development and implementation but that can be applied to any company, industry or organisation in general for the processing of its information.
- c) Blockchain is a reliable technology, but at the same time it is new. There are still few vendors, components and documentation in the blockchain field and now with the recent blockchain as a service (BAAS) platform it is imperative to maintain a permanent research on this emerging technology to have the best options for developing and upgrading blockchain workbench platforms in the short and medium term.
- d) The workbench platform built in this project has a wide field of application in the integration with the Internet of Things (IoT) to validate and record information generated in large quantities as a result of sensor data management processes at industrial, business or personal level; It can also be applied in the registration and control of people's identity to process personalised transactions or, as a priority, it can be applied in the registration, monitoring and control of vaccination certificates, both the COVID-19 and the vaccination booklet that all Mexican citizens are entitled to have and use throughout their lives.
- e) The application areas of blockchain nodes and specifically this workbench platform are numerous, as it can be configured to the processes and transactions required by the company applying this technology.
- f) The constructed workbench platform has great opportunities for improvement due to the current development platforms and services provided by the existing compute clouds:
 - Currently there is already a blockchain service (Blockchain As A Service - BAAS) which did not exist at the time of this research. Replacing the Azure blockchain node with one of the 10 current BAAS providers on the internet (AWS, IBM, Corda, Nodesmith, Dragonchain, among others) will be a migration of great impact, because now it will be possible to manage the growth in the use of nodes and users (companies) that use the workbench platform, this will allow for greater savings in costs and implementation times.
 - As the basis of the backend is the server language and the web development IDE, it is suggested to migrate to the open source languages Python and Java to further decrease the cost of developing and updating the workbench platform (Pilkington, 2016), but it will be necessary to investigate with BAAS providers the release of libraries compatible with the communication towards the BAAS nodes that are built, as the solidity language is still a standard for the development of smart contracts.
 - The above suggestion implies considering now open source based web servers: apache, tomcat and derived versions of these in order to migrate the web API and XML web service to these technologies. It is recommended to continue applying XML and Json standards for communication between the business rules layers of the middleware.
 - Regarding the frontend module, it is also suggested to migrate the design to open source client tools such as: JSP or PHP web pages that are compatible with current interface design frameworks (bootstrap, angular, among others); they also have server-side functionality to interact with the middleware built.

- It is recommended to migrate the Database and the satellite entities module to the mySql database manager, this will open up the possibilities of selecting a hosting with lower and more accessible rental costs, since both the backend and frontend would be migrating to open source platforms.
- It is recommended to continue using Ethereum as the base network for enterprise blockchain, but we should not lose sight of the fact that there are currently more enterprise networks whose capacity, reliability, level of security and privacy, as well as transaction transfer speed, must be validated in order to be recognised as a base network for blockchain (Werner, Lawrenz and Rausch, 2020).

Retamosa Santos, A. (2015). *Scrum. Aplicación del método ágil en la gestión de proyectos*. Universidad Carlos III de Madrid. url : <http://hdl.handle.net/10016/26132> Fecha de último acceso: 25 de agosto de 2022

Werner, R., Lawrenz, S., & Rausch, A. (2020). *Blockchain analysis tool of a cryptocurrency*. ACM Digital Library, (pp. 80-84). DOI: 10.1145/3390566.3391671 <https://doi.org/10.1145/3390566.3391671> Fecha de último acceso: 30 de agosto de 2022

9. References

Ahram, T., Sargolzaei, A., Sargolzaei, S., Daniels, J., & Amaba, B. (2017). *Blockchain technology innovations*. IEEE (pp. 137-141). DOI: 10.1109/TEMSCON.2017.7998367 <https://doi.org/10.1109/TEMSCON.2017.7998367> Fecha de último acceso: 28 de agosto de 2022

Hegedus, P. (2018). *Towards analyzing the complexity landscape of solidity based ethereum smart contracts*. ACM Digital Library, (pp. 35-39). DOI: 10.1145/3194113.3194119 <https://doi.org/10.1145/3194113.3194119> Fecha de último acceso: 25 de agosto de 2022

Nowinski, W., & Kozma, M. (2017). *How can blockchain technology disrupt the existing business models?*. Entrepreneurial Business and Economics Review, (pp 173-188). DOI:10.15678/EBER.2017.050309, <https://doi.org/10.15678/eber.2017.050309> Fecha de último acceso: 30 de agosto de 2022

Pilkington, M. (2016). *Blockchain technology: principles and applications*. Edward Elgar Publishing. DOI: 10.4337/9781784717766.00019 <https://doi.org/10.4337/9781784717766.00019> Fecha de último acceso: 30 de agosto de 2022