



**UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH**

**Escola Tècnica Superior d'Enginyeria
de Telecomunicació de Barcelona**

TEACHING SELF-SOVEREIGN IDENTITY

A Master's Thesis

Submitted to the Faculty of the

Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

by

Laia Rus Bordas

In partial fulfilment

of the requirements for the degree of

MASTER IN TELECOMMUNICATIONS ENGINEERING

Advisor: Juan Bautista Hernández Serrano

Barcelona, June 2022



UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH



telecos
BCN



Title of the thesis: TEACHING SELF-SOVEREIGN IDENTITY

Author: Laia Rus Bordas

Advisor: Juan Bautista Hernández Serrano

Keywords: Self-Sovereign Identity, Decentralized Identifier, Verifiable Credential, Selective Disclosure, Verifiable Presentation, Veramo, W3C

1. Abstract

For service providers, secure and reliable identification of users is essential to provide its services.

From a user perspective, traditional identifiers are currently solved by centralized entities who have the capacity to control not only the creation of the identifier, but also the withdrawal. Moreover, in most cases more personal information is being provided than needs to be demonstrated.

A blockchain-based Self-Sovereign Identity (SSI) provides a secure and reliable identification method for service providers, gives the user self-control of the identifier, and enables a way to provide just the essential information that is needed to get the service.

This paper aims to make two practical documents; the first one being an introductory practice to get started with this topic and the second one that consists of developing a simple SSI login system for web services offered to university students.



2. Acknowledgements

I would like to thank Juan Bautista Hernández Serrano for his insightful reviews and fruitful discussions in the creation of this work.

3. Revision History and Approval Record

Revision	Date	Purpose
0	14/05/2022	Document creation
1	01/06/2022	Document revision
2	24/06/2022	Document finalization
3	29/06/2022	Document approval

Written by:		Reviewed and approved by:	
Date	24/06/2022	Date	29/06/2022
Name	Laia Rus	Name	Juan Hernández
Position	Project Author	Position	Project Supervisor

4. Table of Contents

1. Abstract	1
2. Acknowledgements	2
3. Revision History and Approval Record	3
4. Table of Contents	4
5. List of Figures	5
6. Introduction.....	6
6.1. Objectives	7
6.2. Requirements and Specifications	7
6.3. Work Plan.....	8
7. State-of-the-Art of Self-Sovereign Identity.....	10
7.1. Distributed Identifiers.....	10
7.2. Verifiable Credentials	12
7.3. Zero Knowledge	14
7.4. SSI Technologies	15
8. Methodology and Used Technologies.....	16
8.1. Veramo.....	17
8.2. Blockchain Setup.....	19
9. Practical Approaches.....	20
9.1. Manage DID in a Blockchain Environment.....	20
9.1.1. Initial Design.....	20
9.1.2. Implemented Application	21
9.2. Login with SSI	24
9.2.1. Initial Design.....	24
9.2.2. Implemented Application	30
10. Conclusions and future development	38
11. References	39
12. Glossary.....	40

5. List of Figures

<i>Figure 1 - A simple example of a DID</i>	<i>11</i>
<i>Figure 2 - Ecosystem for VCs.....</i>	<i>13</i>
<i>Figure 3 - Plugins Architecture.....</i>	<i>17</i>
<i>Figure 4 - Ecosystem of the second practical work.....</i>	<i>25</i>
<i>Figure 5 - Sequence diagram of the second practical work.....</i>	<i>26</i>
<i>Figure 6 - Monsters University login page.....</i>	<i>35</i>
<i>Figure 7 - Monsters University's web page</i>	<i>35</i>
<i>Figure 8 - Monsters University web page. Red alert</i>	<i>36</i>
<i>Figure 9 - Monsters University web page. Green alert.....</i>	<i>36</i>
<i>Figure 10 - Monsters Gym web page.....</i>	<i>37</i>

6. Introduction

Today's login systems have some inconvenience for both, service providers and end-users.

From a service provider point of view, it is needed to store users' credentials if they log in the service with usernames and passwords. It is true that saving salted hashed passwords prevents an attacker getting the password in raw if there is a data leakage. However, in many cases people use very simple passwords, which follow very common patterns and are, therefore, very easy to guess. This poses a risk to the company that stores the passwords, and it must also take the necessary steps to comply with the GDPR.

From a user point of view, they should use a secure password for every service they use, but it is infeasible. That is why OAuth and OpenID Connect appeared; to facilitate this task to the user.

However, after all, users do not control their account. It is the service provider the one who manages its creation and its withdrawal. For example, Google Account is used to sign into other applications or services. That means that if Google decides to invalid someone's Google Account, this person ends up without an identifier. Another possible situation is that for some reason Google stops working. In that case, many people lose their identifier, too.

A solution to prevent the system from being centralized is to use non-controlled identities in a blockchain, which works as a decentralized cloud service that stores addresses and related information for user identification.

The concept of Self-Sovereign Identity (SSI) has emerged with the aim that the user is the only one to have the full control over their own identity.

Apart from having more control over their own identity, SSI allow users accessing many different services using the same anonymous credential, so this solution also facilitates this task to users, since they do not need to remember the credentials for every service provider.

It is also interesting for the end user to provide only the essential information needed to get the desired service. In most cases more personal information is being provided than needs to be demonstrated. For example, if a person goes to a nightclub and is asked for his national identity document to prove that he is of legal age, he is not only giving the data of the day he was born and, therefore, the age he currently has, but also informs about where he lives, where he was born, what is his name and what is the name of his parents. With SSI this person could have an identifier associated with a verifiable credential whose single claim is being more than 18 years old.

This solution not only has more benefits for users because they achieve more privacy, but it is also beneficial for the company because it deals with the minimum possible data, and this meets a requirement of the GDPR called Data Minimization.

In SSI, a digital identity is built upon a public-private key pair that is unique to that identity. The public key is used to create the public identity, and the private one to prove ownership of that identity. The key pair is created and managed by the users themselves usually using what is called a Wallet application.

All in all, SSI provides a secure and reliable identification method for service providers, allows end users to disclose just the needed information to get the service, and retains control over their identifiers back to end users.

However, it should be noted that users having control over their identifiers implies that if they lose their secret keys, there will be no other alternative than to create new ones, since no one will be able to recover them.

6.1. Objectives

The aim of this thesis is to make teaching practices for the explanation and understanding of SSI, based on the use of blockchain Distributed Identifiers (DIDs) and Verifiable Credentials (VCs), which are explained in more detail in section [State-of-the-Art of Self-Sovereign Identity](#).

Specifically, the goal is to make two practical documents:

- The first one to develop a program capable of creating, listing, modifying, and deleting DIDs in a blockchain environment.
- And the second one to make a website of a university in which logged in students can access a service following the SSI protocol.

6.2. Requirements and Specifications

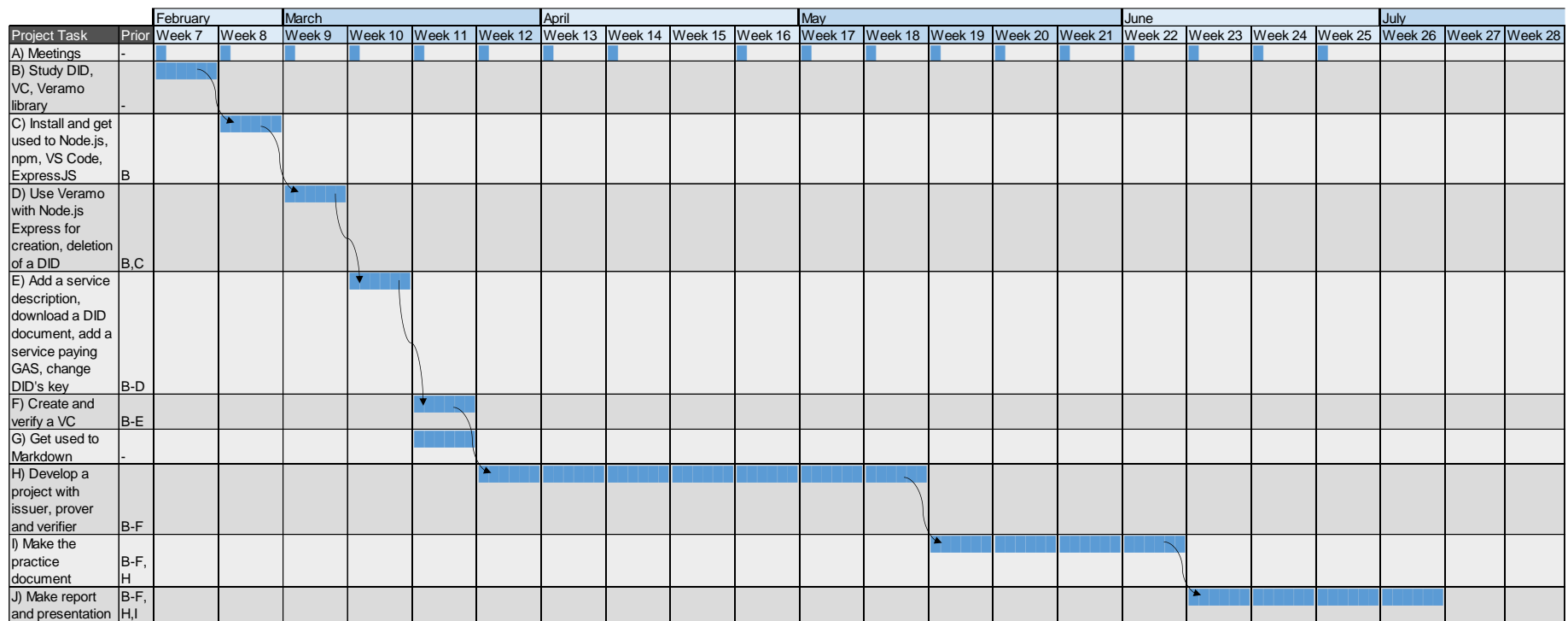
Regarding the **practical documents**, they must introduce the topic, so that students get familiar with it, and the statement must be understandable to make it easy for them to comprehend the task they must do.

With respect to the **practical work**, the requirements are the following:

- Basic cyber-security knowledge is needed
- It is assumed that the student has previously completed the Express with Passport practice of the subject (see [Annex I](#)).
- The working environment can be their host computer or a Virtual Machine (VM).
- The programming language must be [JavaScript](#) and [TypeScript](#) for both the frontend and the backend.
- [React framework](#) must be used for the frontend
- [Node.js](#) with [Express](#) and [Passport](#) must be used for the backend.
- The software [npm](#) must be used to manage the JavaScript packages.
- The [W3C](#)'s SSI protocol must be followed to access the website of the gym service.

6.3. Work Plan

The tasks have been planned and done as shown in the following Gantt chart:



Each task is explained below:

- A) **Meetings.** Once a week, advisor and student meet online to assess the work done so far and to plan the following tasks.
- B) **Study DID, VC, Veramo library.** As the student is not initially familiar with these concepts, she needs to spend some time studying them to understand them correctly.
- C) **Install and get used to Node.js, npm, VS Code, ExpressJS.** Before implementing the final goals, it is necessary to have completed the Express with Passport practice of **Annex I**.
- D) **Use Veramo with Node.js Express for creation, deletion of a DID.** It refers to the development of the first practical work mentioned in [section 6.1. Objectives](#); that of creating, modifying, and deleting DIDs in a blockchain environment.
- E) **Add a service description, download a DID document, add a service paying GAS, change DID's key.** This consists of finishing the first practical work.
- F) **Create and verify a VC.** This means implementing an Express server capable of creating and verifying a VC to get some practice before implementing the second practical work mentioned in [section 6.1. Objectives](#).
- G) **Get used to Markdown.** Since the statement of the practice will be written in [Markdown](#), it is necessary to know how to use it
- H) **Develop a project with issuer, prover, and verifier.** It refers to the implementation of the second practical work mentioned in [section 1.1. Objectives](#).
- I) **Make the practice document.** This task consists of writing the statement of the two practices: DID Management (**Annex II**) and Self-Sovereign Identity Login System (**Annex III**).
- J) **Make report and presentation.** It means writing this thesis and preparing the final presentation.

7. State-of-the-Art of Self-Sovereign Identity

7.1. Distributed Identifiers

Data exchange is likely to require verification of the involved peers' identities, or at least of some claims proving them. However, the traditional authentication strategy based on usernames and passwords does not scale properly. Users must use different passwords for each individual service to provide enough security to the system. This is inconvenient, hard to manage, and prone to security risks. Moreover, from the server point of view, the management of customers' identities comes with a lot of associated risks in terms of data leaks and compliance with the GDPR.

That is why during the last years there has been a natural evolution towards the use of external identity providers, such as Google or Facebook, to prove ownership of a given identity (or pseudo-identity). De facto standards on the Internet today are OAuth2 RFC6749 and, above all, OpenID Connect Saki14. However, users still depend on these external services to create and manage their identities, and providers are the ones having full control of the users alter egos.

In contraposition, **SSI** gives the control of their own identities back to the users. The need for trusted identity providers can now be removed since users themselves oversee the creation and the management of their own identities.

In SSI, a digital identity is built upon a public-private key pair that is unique to that identity. The public key is used to create the public identity, and the private one to prove ownership of that identity. The key pair is created and managed by the users themselves usually using what is called a Wallet application.

Once a user has an identity, different entities can issue verifiable credentials for that identity. A verifiable credential, which can represent the same information as a traditional physical credential, is a tamper-evident credential whose authorship can be cryptographically verified with digital signatures.

Although the approach guarantees that users are in control of their identities, it comes with a huge risk: if a private key is lost, compromised, or just needs to be updated, the user will lose control on the associated identity, and all the verifiable credentials issued for it.

As a result, state-of-the-art SSI approaches create immutable identifiers that are just pointers to identities - that is to say, public keys or addresses - in a way that one can update the identity keys without changing the identity's identifier. This is the main idea behind the use of **DIDs**, which point to **DID Documents**. DID Documents are sets of data that contain the public key for the corresponding DID, any other public credentials the identity owner wishes to disclose, and the network addresses for interaction. The identity owner controls the DID Document with the associated private key.

DIDs are the first globally unique verifiable identifiers that require no registration authority. They are a globally resolvable, and a cryptographically verifiable **open standard** proposed by the World Wide Web Consortium (W3C); the main international standards organization for the World Wide Web.

A DID is a simple text string consisting of three parts: the did URI scheme identifier, the identifier for the DID method, and the DID method-specific identifier, as shown in *Figure 1*:



Figure 1 - A simple example of a DID

Source: <https://www.w3.org/TR/did-core/#a-simple-example>

The example DID above resolves to a DID document. A DID document contains information associated with the DID, such as ways to cryptographically authenticate a DID controller.

A DID Document is a JSON-LD object that contains information about an identity, such as credentials and about how to contact it.

JSON-LD (semantic web) allows to understand the structure of an object to easily interpret the data.

Below there is an example of a simple DID Document:

```
{
  "@context": [
    "https://www.w3.org/ns/did/v1",
    "https://w3id.org/security/suites/ed25519-2020/v1"
  ],
  "id": "did:example:123456789abcdefghi",
  "authentication": [{
    "id": "did:example:123456789abcdefghi#keys-1",
    "type": "Ed25519VerificationKey2020",
    "controller": "did:example:123456789abcdefghi",
    "publicKeyMultibase": "zH3C2AVvLMv6gmMnam3uVAjZpfkcJCwDwnZn6z3wXmqPV"
  }]
}
```

The fact that the DID Document is resolved using a network that is not controlled by anyone, allows the user to be the sole owner of the identity and have control over it. That is why a public blockchain is used to store the DID Documents.

A public blockchain is not controllable by any entity (governments, consortia nor CAs) and it can be used as a technology for anchoring the identity registry, since it provides fully decentralized, cryptographically verifiable, and publicly available registries. It uses a **consensus algorithm** operating over many different machines and replicated by many different entities.

With a public blockchain for DIDs, anyone can issue a digitally signed credential, and anyone else can verify it.

7.2. Verifiable Credentials

Trust for DIDs is based on **VCs** that other entities endorse. VCs define credentials or claims issued for DIDs and they can be considered as the core technologies for SSI today.

VCs are expected to be useful in an ecosystem composed of the following roles:

The **issuer**, which is an entity that oversees the following operations:

1. Asserting claims about one or more subjects
2. Creating a VC from these claims
3. Transmitting the VC to a holder.

Example issuers include corporations, non-profit organizations, trade associations, governments, and individuals.

The **holder**, who possesses one or more VCs and generates VPs from them. Example holders include students, employees, and customers.

The **subject** is an entity about which claims are made. Example subjects include human beings, animals, and things. In many cases the holder of a VC is the subject, but in certain cases it is not. For example, a parent (the holder) might hold the verifiable credentials of a child (the subject), or a pet owner (the holder) might hold the verifiable credentials of their pet (the subject).

The **verifier** is an entity that receives one or more VCs, optionally inside a **Verifiable Presentation (VP)**, for processing. Example verifiers include employers, security personnel, and websites.

A VP is data derived from one or more verifiable credentials and issued by one or more issuers. This data is shared with a specific verifier, who processes it using cryptography.

The **verifiable data registry** mediates the creation and verification of identifiers, keys, and other relevant data, such as verifiable credential schemas, revocation registries, issuer public keys, and so on, which might be required to use verifiable credentials. Example verifiable data registries include trusted databases, decentralized databases, government ID databases, and distributed ledgers.

Figure 2 illustrates the relationship between those roles:

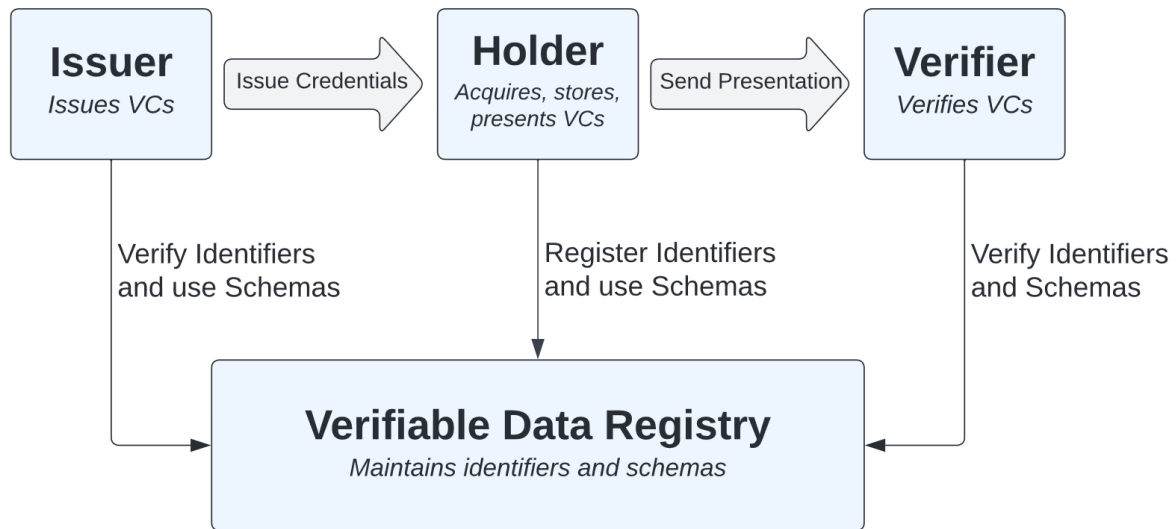


Figure 2 - Ecosystem for VCs

Based on the one at: <https://www.w3.org/TR/vc-data-model/#ecosystem-overview>

A VC is also a JSON-LD object, and it follows a standard data model and representation format for cryptographically VCs. It is a JSON Web Signature (JWS), and it is commonly sent with a structure composed by three fields: the credential metadata, one or more claims and one or more proofs.

VCS are designed to be compatible with a variety of proof formats, most of which can only reveal all (or none) of their attributes.

7.3. Zero Knowledge

Zero-Knowledge Proofs (ZKPs) are a cryptographic technique or a proof format that enables data-minimization features in **VPs**, such as **selective disclosure (SD)** and **predicate proofs**. Moreover, ZKP's VPs do not contain, the original VC.

SD means that a holder can decide which claims to reveal to prove a specific statement or request and predicate proofs are a cryptographic solution to proving something about an item of data without revealing the data itself.

For example, people go to a bar, but instead of showing their ID, which reveals their name, the day they were born and more, they only prove that they are old enough to get in. In this specific case, with SD holders reveal the specific claim that says they are of legal age, and with predicate proofs they prove this claim without revealing any more personal information.

When ZKP is not used in SSI, users need to provide the entire VC. That is why it is convenient to create credentials with a single claim when no cryptographic techniques are used.

To the contrary, when ZKP is used, users can have a VC with many different claims and proof with predicate proofs that they comply only a specific claim.

Zk-Snarks, that stands for *Zero-Knowledge Succinct Non-Interactive Argument of Knowledge*, is an implementation of ZKP that allows one party to prove it possesses certain information without revealing that information and it is used as part of the protocol for a cryptocurrency called **Zcash**.

Zk-Snarks was developed because some lack of privacy was perceived in Bitcoin, since it is proved that it is relatively easy to re-identify people who gives pseudonymous data to multiple sources. This study was published in 2019 in an article called ***Estimating the success of re-identifications in incomplete datasets using generative models***.

7.4. SSI Technologies

In this section it is described some of the most popular already available SSI technologies.

There is a protocol for SSI called **Sovrin** that was created by Sovrin Foundation, a non-profit organization supported by a team of dedicated volunteers and experts from around the world. Sovrin wants to meet high standards of privacy using ZKP working on a blockchain infrastructure. Blockchain works as a decentralized self-service registry for public keys, and ZKP becomes the standard for all interactions between identity owners.

Evernym is a company that builds and deploys self-sovereign identity solutions. It is the creator of Sovrin Foundation and its public service utility. Evernym has developed a flagship credential exchange platform, a web application for verifiable credential exchange, a digital wallet app for storing, managing, and sharing digital credentials and a Software Development Kit (SDK) to integrate verifiable credentials into any iOS or Android app.

Trinsic is a full stack platform built for Ethereum blockchain-based Smart Contracts (SCs) that offers an infrastructure for sending verifiable data between digital identity wallets, an ecosystem to authenticate users and share data safely, an easy-to-use dashboard for organizations and a wallet SDK.

uPort project began in 2015. It was an SSI solution based on the Ethereum blockchain and they provided JavaScript libraries so that developers could integrate uPort's functionalities to their application. Later, uPort was divided into two new projects: Serto and Veramo.

Serto is an ecosystem that provides decentralized identity and connected data solutions for enterprises. It is compatible with Ethereum, easy-to-use and no code is needed.

Veramo are modular APIs for Verifiable Data and SSI backed by TypeScript. It is compatible with Node, React and React Native, and it is the technology used in this thesis.

Civic is a company that offers an integrated permissioning tool for businesses to control the access to their decentralised applications (dApps). Civic works on the Solana blockchain and it uses **identity**'s open-source and blockchain-based ecosystem to verify credentials.

Validated ID is a company that, among other services, offers an SSI service based on blockchain to provide people control over their identity and facilitate secure user access to online services.

ID_Alustria is a theoretical SSI model for digital identity based on Ethereum that will allow transactions on the Alustria network to be legally valid and comply with Spanish and European regulations. Currently, they have a Minimum Viable Product (MVP) which is available for consultation and use in **Alustria's GitHub**.

Gataca is a company that has developed a credential issuance tool for trusted authorities, a credential verification tool for service providers and an identity wallet to store encrypted identity credentials. Gataca offers a set of APIs for multiple blockchain networks, and it currently supports Ethereum public network and private networks based on Hyperledger Fabric, Hyperledger Besu or Quorum.

8. Methodology and Used Technologies

The development of this project has been done progressively every day.

The working environment is a computer running Windows 11 and all programming has been performed in Visual Studio Code.

Both the frontend and the backend are programmed in **JavaScript** and **TypeScript**. Specifically, a framework called **React** has been used for the frontend (everything displayed by the browser), and it has been combined with **Material UI (MUI)** for its design. As for the backend (everything that runs in the APIs), **Node.js** with **Express** and **Passport** has been used. Before implementing the backend, the Passport practice of **Annex I** has been followed.

React is a JavaScript-based User Interface (UI) development library that can be used as a base in the development of single-page application and mobile application (React Native).

MUI is an open-source component library that follows Google's guidelines for rendering components.

Node.js is an open-source server environment running on the V8 JavaScript engine, the same one used by Chromium-based web browsers (Chrome, Opera, Edge, Brave, etc.).

Express and Passport are two popular packages for Node.js today. The former is a web development framework for Node.js, and the de-facto standard for the majority of Node.js applications; and the latter is an authentication middleware whose sole purpose is to authenticate requests, and that perfectly integrates with Express.

The packets of the project are managed with **npm**, which is Node.js's default package manager, which facilitates the management of the software dependencies of the project.

Veramo is used as a JavaScript library to generate VCs, DIDs and to connect to the blockchain; being the chosen blockchain the **Ropsten** public testnet.

8.1. Veramo

W3c descriptions have been implemented with Veramo, a flexible, modular, and scalable JavaScript/TypeScript framework that makes it easy for developers to use cryptographically verifiable data in their applications.

In this thesis Veramo is used to manage DIDs, to create VCs and to connect to a testnet through Infura.

To do all these actions, it is needed to design an agent that works as the entry point into the Veramo framework. This agent can be customized by using their **core plugins**.

There are some relationships between the core agent, interfaces, plugins, and external protocols, as shown in *Figure 3*:

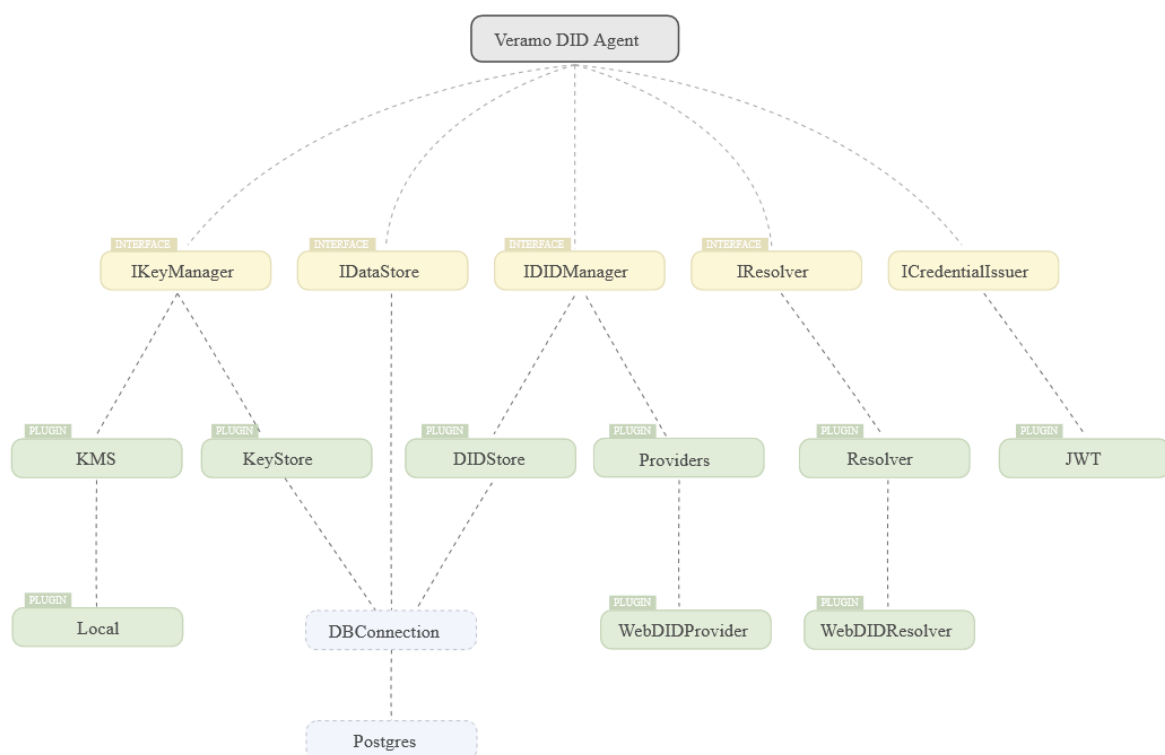


Figure 3 - Plugins Architecture

Source: https://veramo.io/docs/veramo_agent/plugins

IKeyManager is an interface that has a set of methods that allow making actions with the keys of the identifiers. For example, it grants signing an Ethereum transaction, signing a JWT, deleting a key, and others.

IDataStore is another interface that allows storing and getting information about verifiable credentials and presentations to/from a database.

IDIDManager interface has multiple methods to perform different actions to identifiers, such as creating, deleting, listing, and modifying them.

IResolver allows getting the DID Document of the specified DID.

ICredentialIssuer has two methods: one for generating VCs and the other to generate VPs.

W3C presents a **large list of different possible DID methods**. DID methods are used to resolve the corresponding DID Document and Veramo core plugins support three of them: *did:ethr*, *did:key*, and *did:web*.

did:ethr allows any Ethereum key pair to become an identity and it gives any Ethereum address the ability to collect on-chain and off-chain data. This DID method relies on the *ethr-did-registry*, a SC that facilitates public key resolution for on-chain and off-chain authentication, key rotation, delegate assignment and revocation to allow third party signers, as well as setting and revoking off-chain attribute data. The identifier is the SC itself.

As the target system is the Ethereum network, where the ERC1056 is deployed, blockchains like Mainnet, Ropsten, Rinkeby, and Kovan can be used.

did:key is a very light-weight self-certifying DID method that does not require any external utility such as a blockchain. A Key-DID is created by generating a cryptographic key pair and it always resolves the same DID Document. The DID Document is immutable, so is not possible to add service endpoints and other keys.

did:web allows the owner of a web origin to turn it into a DID. After creating the DID Document, it is hosted under an URL, and it must be available through HTTPS GET every time the DID is resolved. As no blockchain is required to create a DID, it must be considered that the security of a Web-DID is rooted in the existing Internet PKI by enforcing TLS. The URL where the DID Document is hosted can be either a web domain, like www.example.com/.well-known/did.json, or a specific sub-path. Using sub-paths allows hosting multiple DIDs under one web domain. This approach can also be used to allow a web application to create Web-DIDs for their users, e.g., www.example.com/users/username/.well-known/did.json.

8.2. Blockchain Setup

There are different Distributed Ledgers Technologies (DLTs), like [Bitcoin](#), [Ethereum](#), [HyperLedger Fabrik](#), [Polygon](#), and others.

Among all these technologies Ethereum has been chosen.

Ethereum is a decentralized blockchain platform that establishes a peer-to-peer network. Its cryptocurrency is called Ether (ETH) and transactions are paid with a fee called Gas. Small fractions of the cryptocurrency ETH are referred to as Gwei or Nanoeth.

The building blocks of Ethereum applications are the SC and they carry out the transactions over the network.

The operation of a DLT requires running the technology in a real infrastructure; a cost that is translated to users usually in the form of fees per operation, which require paying with real cryptocurrencies. However, there are as well public testnets allowing testing of developed technologies for free, before moving to a mainnet.

There are many testnets based on the Ethereum technology, like [Ropsten](#), [Rinkeby](#) and [Kovan](#). In this project Ropsten is the one being used because of the convenience of its [Faucet](#) service, which allows getting Ropsten Ethers for free.

Changing an Ethereum-based project from a testnet to a mainnet only implies changing the configuration to the network and to assume that payments will be done with real currencies.

A cryptocurrency wallet is a device, physical medium, program, or a service that stores public and/or private keys for cryptocurrency transactions and it usually offers the functionality of encrypting and/or signing information. [Electrum](#), [Mycelium](#), [Exodus](#), and [Metamask](#) are examples of cryptocurrency wallets.

If the product of this thesis were to be marketed, a cryptocurrency wallet would be used. As it is a teaching practice, for the sake of simplicity, keys will be stored in the browser's [localStorage](#), and cryptographic operations will be performed using browser JavaScript.

[SQLite](#) is used to store the public and private keys for the DID management in the server-side.

There are websites like Etherscan that let you check the balance and the transaction made in Ethereum networks. The [Ropsten Etherscan](#) service has been used for the first practical work presented in section [1.1.1. Objectives](#).

In order to operate with a blockchain testnet, a network node is needed to connect the application to it. The node can be created manually or using an API. In this project, an API called [Infura](#) is used because it facilitates the connection to the Ropsten testnet

9. Practical Approaches

In this section it is explained the two practical works developed for this thesis.

The first one is called **DID Management** and it can be found in [this GitHub repository](#). It is an introductory program that let the student understand and get used to most of the technologies needed for the next practical work.

The second one is called **Self-Sovereign-Identity Login System** and it is located in [this GitHub repository](#). It is the main practical work, and it allows the student to deeply understand the concept and the utility of Self-Sovereign-Identity in a use case that could be extrapolated to a real-world application.

9.1. Manage DIDs in a Blockchain Environment

Before implementing the actual work, a design is needed to plan and consider how it should be.

In this section it is presented the proposed design and the resulting implementation.

9.1.1. Initial Design

The idea is to create a TypeScript program capable of managing DIDs in a blockchain environment. Specifically, the program must be able to **create, list, modify and delete DIDs**. Modifying DIDs implies **adding a service and/or a key**. To do all this actions, the program must be able to store the created DIDs and SQLite will be the database to do so.

It is enough to interact with the program through a console, so no graphical interface is needed.

Ropsten is going to be the testnet used for this project because it is free (it uses Ethers as digital coins, but they are not real), and many people around the world participate in the network.

In a mainnet, making changes on the blockchain has a cost. To avoid paying some fee every time that someone creates a DID, it was defined a base DID document called Minimal DID Document, that can be generated without writing on the blockchain. This document is basically a public key that is stored nowhere in the blockchain and that can be read by a specific SC. The *did:ethr* method is solved using a SC deployed in Ropsten and follows this strategy.

Regarding the functionalities of this project, creating, listing, and deleting DIDs do not make a transaction in the blockchain (they are free).

A DID is a pointer or link to a DID Document, so when users want to modify it, *did:ethr* method contacts with the SC that manages the respective DID Document and tells it what it needs to change. This change does imply making a transaction in the blockchain, so some fee (Gas, in this case) must be paid. Therefore, adding a service and/or a key do require making a transaction in the blockchain.

Other actions that do require paying Gas, but that are not implemented in this project are revoking it or changing its owner.

9.1.2. Implemented Application

Each functionality (creating, deleting, and modifying DIDs) has been implemented in a different TypeScript document to make the structure of the project easier to understand.

Six different commands have been programmed to execute the functionalities of this application and they are explained below:

Creating a DID: the program can create a DID associated with an alias that allows a person to identify it easier. Every created DID is stored in the SQLite database. This functionality is executed with the following command:

```
$ npm run id:create --alias <alias>
```

Creating a DID does not require making any connection with the blockchain, since they are saved locally and what is generated is a DID Document by default.

```
/* @example
 * ```typescript
 * const identifier = await agent.didManagerCreate({
 *   alias: 'charlie',
 *   provider: 'did:ethr:rinkeby',
 *   kms: 'local'
 * })
 * ```
 */

didManagerCreate(args: IDIDManagerCreateArgs, context:
IAgentContext<IKeyManager>): Promise<IIdentifier>
```

Listing all DID Documents: this functionality lists all created (and not already deleted) DIDs with its associated alias, service, keys and DID Document:

```
$ npm run id:list
```

Listing identifiers does not require making a transaction in the blockchain, either.

```
/* @example
 * ```typescript
 * const rinkebyIdentifiers = await agent.didManagerFind({
 *   provider: 'did:ethr:rinkeby'
 * })
 * ```
 */

didManagerFind(args: IDIDManagerFindArgs): Promise<Array<IIdentifier>>
```


Deleting a specific DID: the alias is the argument used to refer to a specific identifier:

```
$ npm run id:delete --alias <alias>
```

Deleting all DIDs: it is also possible to delete all identifiers in one command:

```
$ npm run id:delete-all
```

Deleting identifiers in another action that does not require a connection with the blockchain.

```
/**  
 * Deletes identifier  
 */  
didManagerDelete(args: IDIDManagerDeleteArgs, context:  
  IAgentContext<IKeyManager>): Promise<boolean>
```

Adding a service to a DID: the service is the following JSON, and it is an implementation of **Veramo's IService**:

```
{  
  "id": "did:web:veramo.dev#msg",  
  "type": "Messaging",  
  "serviceEndpoint": "https://veramo.dev/messaging",  
  "description": "You can contact me by Telegram. My username is @Eu*****ta."  
}
```

This service is added to a DID Document with the following command:

```
$ npm run id:add-service --alias <alias>
```

As the code below shows, when a transaction is generated, it is returned an identifier of this transaction, which can be a hash:

```
/**  
 * Adds a service to a DID Document  
 * @returns identifier provider specific response. Can be txHash, etc,  
 */  
didManagerAddService(args: IDIDManagerAddServiceArgs, context:  
  IAgentContext<IKeyManager>): Promise<any> //txHash?
```


Adding a key to a DID: this functionality adds a random public key to the DID Document of the DID specified with an alias.

```
$ npm run id:add-key --alias <alias>
```

This action also generates a transaction in the blockchain. Therefore, anything that means modifying the DID Document (adding a service or a key, in this case), requires paying some Gas.

```
/**  
 * Adds a service to a DID Document  
 * @returns identifier provider specific response. Can be txHash, etc,  
 */  
didManagerAddKey(args: IDIDManagerAddKeyArgs, context:  
IAgentContext<IKeyManager>): Promise<any>
```

9.2. Login with SSI

In this section it is explained the initial design of the second practical work and the result after implementing it.

9.2.1. Initial Design

The objective of this practical work is to create a very simple SSI system in which a user with a DID can request a VC to guarantee an SD.

To put it in context, the system deals with a student who has a DID and who asks a university to issue student credentials to be able to access external services, such as the gym, the library, and the café.

An ecosystem composed by different roles is needed to manage VCs. Those roles are the ones in *Figure 2* from section [2. State-of-the-Art of Self-Sovereign Identity](#).

The **issuer** is the role in charge of declaring a claim or a set of claims and transmitting a VC created from these claims to a holder. In this practical work, the issuer is the **server of the university**.

The **student** is both, the **holder**, and the **subject**. The former possesses one or more VCs and generates VPs from them, and the latter refers to the entity about which claims are made.

The **verifier** is the entity that receives one or more VCs, optionally inside a verifiable presentation, for processing. The **university's gym service** is the verifier in this practice.

The **verifiable data registry** mediates the creation and verification of identifiers, keys, and other relevant data, such as verifiable credential schemas, revocation registries, issuer public keys, and so on. In this practical work the verifiable data registry is a public distributed ledger (blockchain); specifically, testnet Ropsten, which is free. In a marketable product, transactions would be handled by a cryptocurrency wallet, but browser's local storage will be used instead to keep the practice simple.

A **Selective Disclosure Request (SDR)** is implemented in Veramo as a message type that is created and signed by a DID. It contains a request for specific Verifiable Credential claims and can specify the issuer(s) of those credentials along with other criteria. The subject of the request can respond by creating and signing a Verifiable Presentation to include the requested claims.

The scenario of the practical work is composed by the following subjects:

- The university is called **Monsters University**. As it represents the trusted issuer, it is responsible for the delivery of an anonymous credential (a VC) for every logged in student that requests for it. It declares the following **three claims**:
 - That the person who owns the VC is a student
 - That the university where this student studies is called Monsters University
 - And that it has not expired yet, since it becomes obsolete in 2022-08-01.
- **Mike** is an enrolled student in Monsters University. His username is *mike* and his password is *ILoveCelia*. He is the one generating his VP.

- **Monsters Gym** is a service provided by Monsters University. Enrolled students have the right to access this website if their VP has been verified by Monsters Gym's server.

The ecosystem of the project (*Figure 4*) follows the structure of the appropriate ecosystem for VCs presented in *Figure 2* from section **8.2. Verifiable Credentials**.

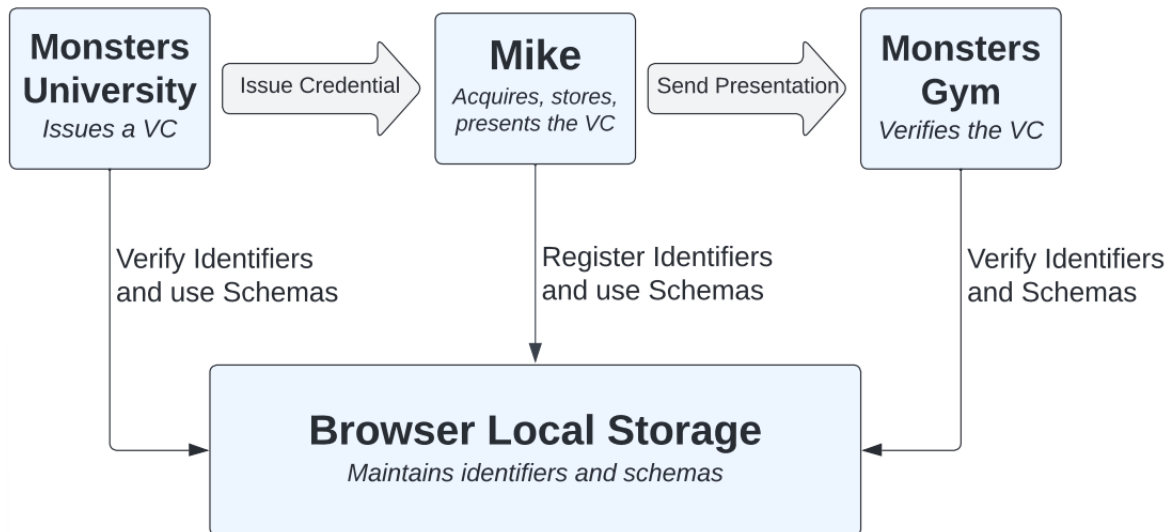


Figure 4 - Ecosystem of the second practical work

The sequence diagram of *Figure 5* describes the flow of the application:

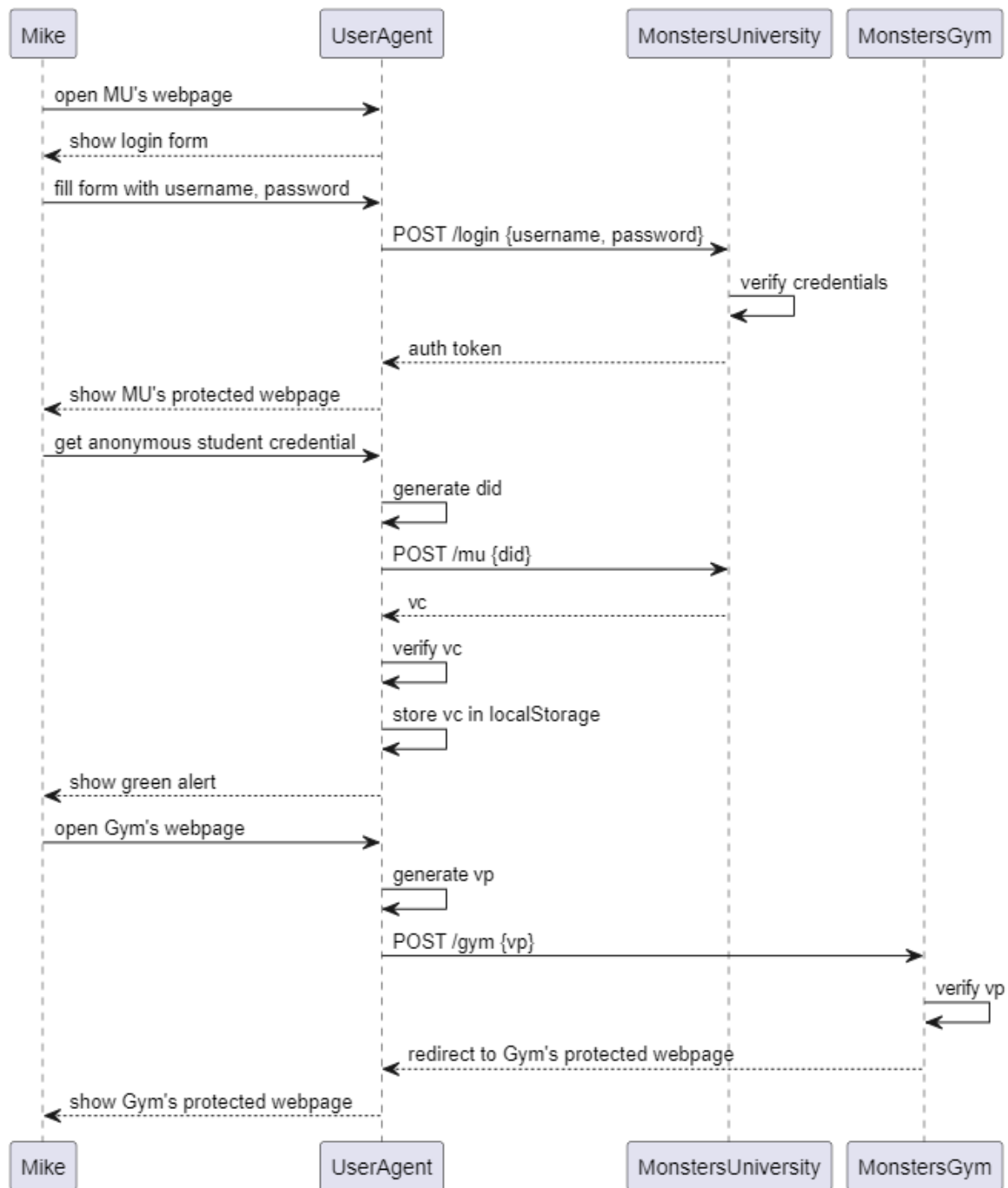


Figure 5 - Sequence diagram of the second practical work

For Mike to be able to access Gym service, he must demonstrate that he is a student at Monsters University. As all students at this university have an account for its webpage, Mike logs in there with his username and password. This part will be developed following the Express with Passport practice of **Annex I**.

Monsters University's server already has a Selective Disclosure whose claims are that the user that asks for a VC is currently a student at this university.

Mike does not notice it, but when he asks for a VC by clicking on a button, the client-site of the application generates a DID automatically for him and stores it in the local storage of his browser (not only the DID value is stored, but also its public and private keys. For security, this information is encrypted).

After receiving the VC, the client's JavaScript automatically verifies that the VC is correct by checking the following:

- That the issuer of the VC equals to Monsters University's DID
- That the subject of the VC is Mike's DID
- That it claims that Mike is a student
- And that the VC has not expired yet

The VP stores the same three claims as Monsters University's Selective Disclosure, and it is sent to Monsters Gym's server.

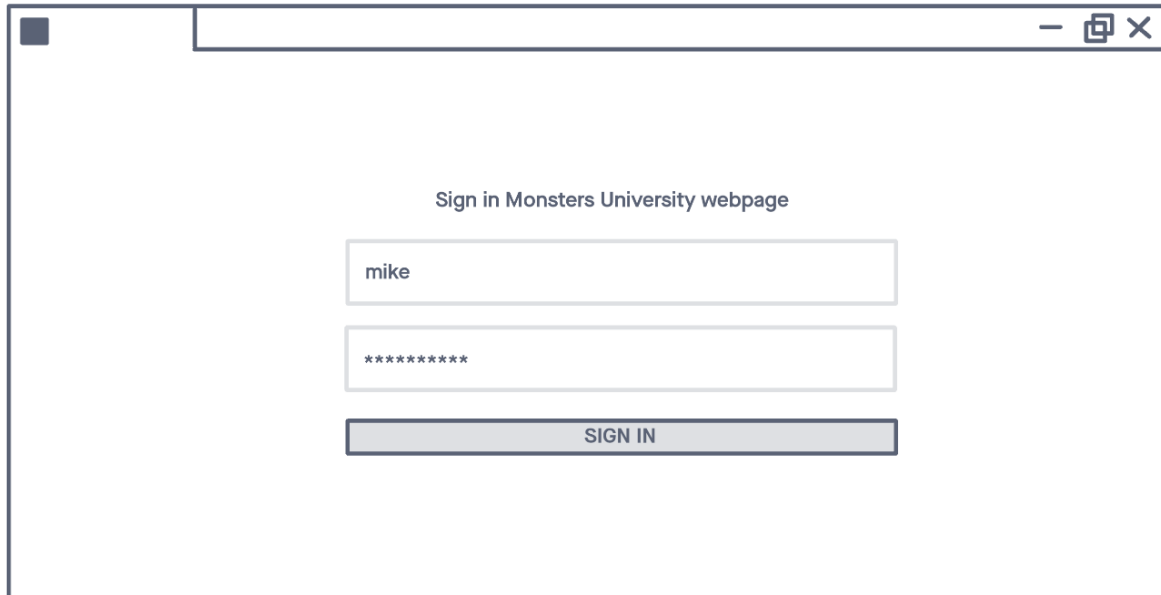
Then, Monsters Gym's server verifies the following:

- That Mike's DID equals to the subject of the VP
- That Mike is a student
- That the university where Mike studies is called Monsters University
- That Mike's credential has not expired yet

If everything is alright, Monsters Gym's server redirects Mike to its web service.

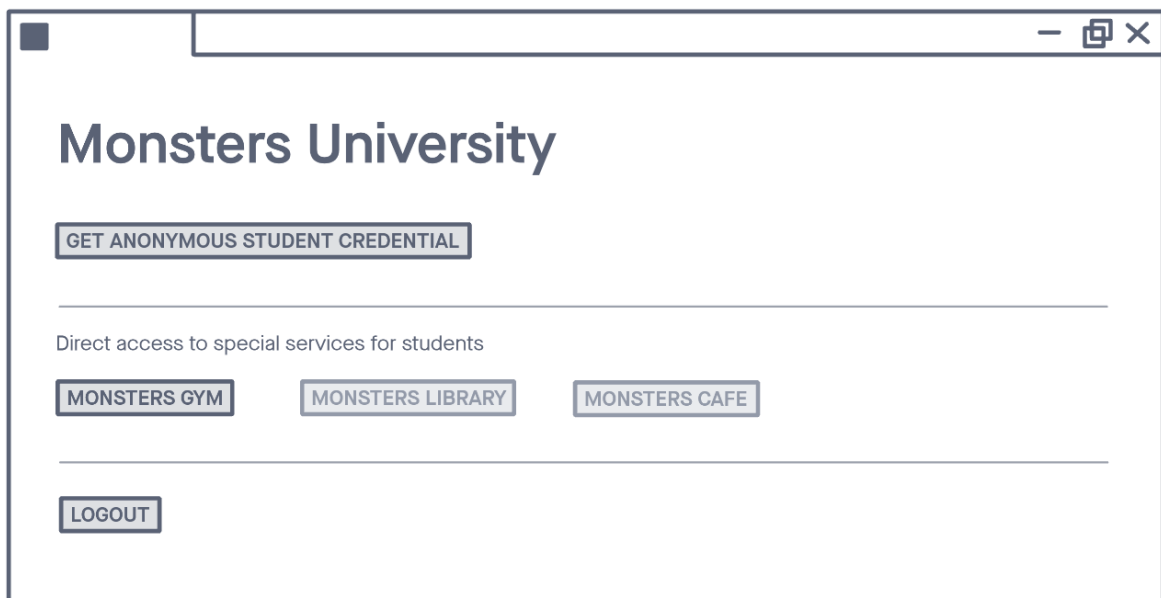
Next, the flow of the project is explained with mock-ups:

Mike goes in Monsters University's login page and enters his credentials:



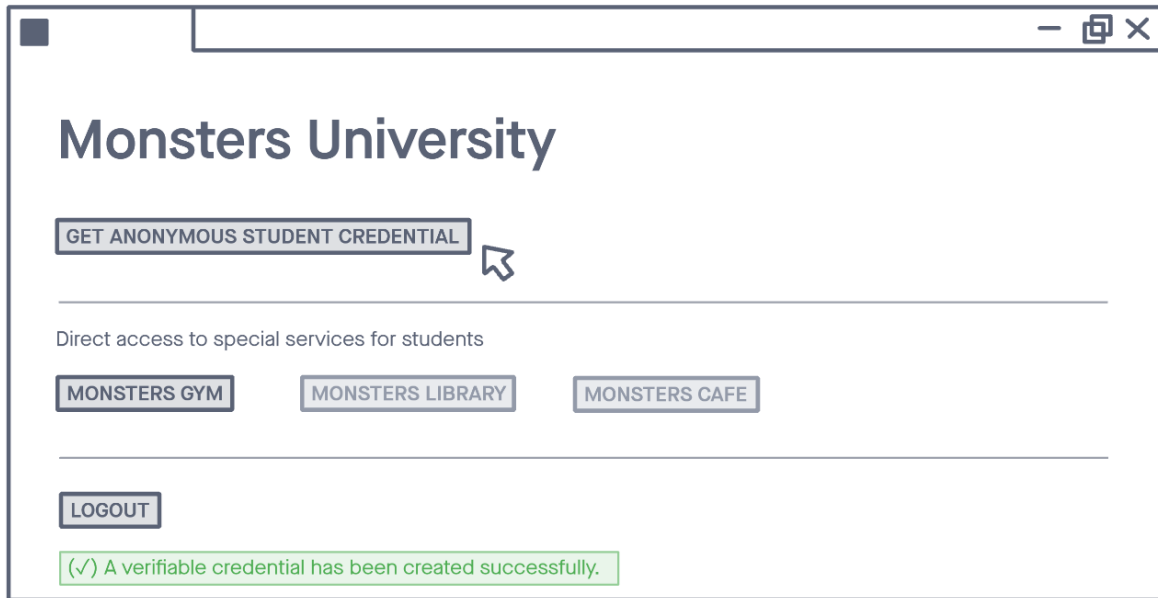
A browser window mock-up showing a login page. The title bar has a close button and a maximize button. The page content is centered and includes the text "Sign in Monsters University webpage". Below this text are two input fields: the first contains the text "mike" and the second contains a series of asterisks "*****". Below the input fields is a button labeled "SIGN IN".

As he introduced his credentials correctly, he is allowed to access Monsters University's webpage, whose design is the following:

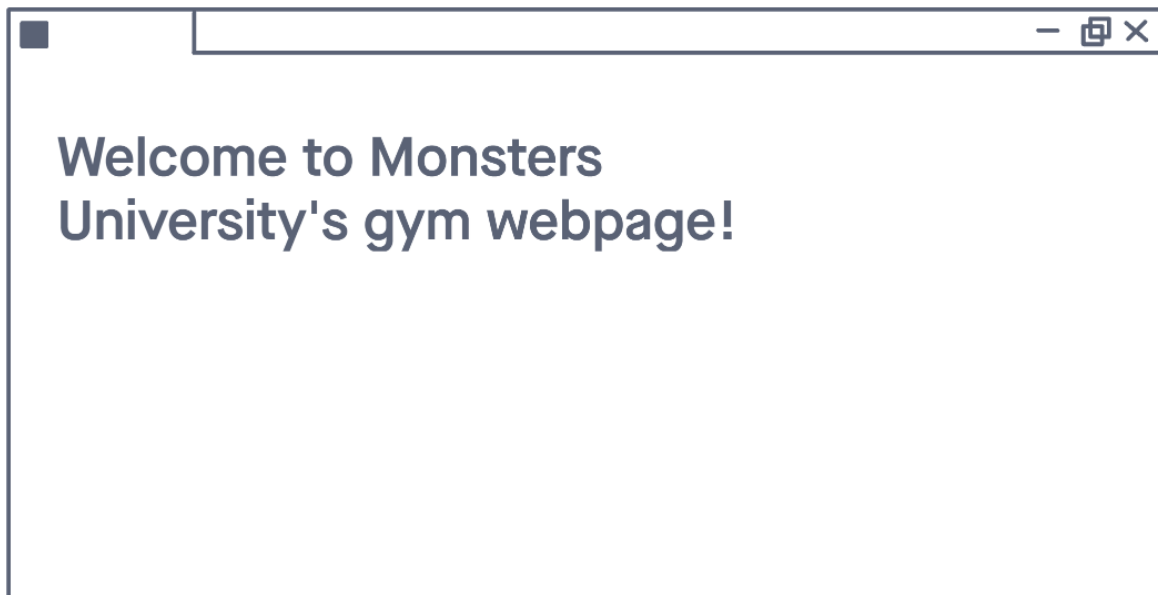


A browser window mock-up showing the Monsters University dashboard. The title bar has a close button and a maximize button. The page content includes the heading "Monsters University" in a large, bold font. Below the heading is a button labeled "GET ANONYMOUS STUDENT CREDENTIAL". A horizontal line separates this from the text "Direct access to special services for students". Below this text are three buttons: "MONSTERS GYM", "MONSTERS LIBRARY", and "MONSTERS CAFE". Another horizontal line separates these from a button labeled "LOGOUT".

Mike asks for the anonymous credential (the VC) by clicking on the first button:



After clicking *MONSTERS GYM* button, he is automatically redirected to Monsters University's Gym webpage. The SSI protocol is executed to access there.



9.2.2.Implemented Application

The system consists of three projects: one for the React application, one for the University's Express API and one for the Gym's Express API.

The **React application** runs the frontend of the login page, the University's page, and the Gym's page.

- The **login page** asks for the student's username and password, and it has a button to log in. When the login is successful, a JWT is written in a HTTP cookie (**Express with Passport practice**) and it sends the credentials to University's Express API.
- **University's webpage** has a button to get an anonymous credential or VC, another one to go to Gym's webpage and a last one to logout and be redirected to the login page.
 - After clicking on the button to get the VC, Mike's DID is stored in the local storage of the browser, together with the corresponding public and private keys. Just for security, it is encrypted with a library called **crypto-js**. This **DID management** has been **implemented manually** by the student of this thesis, since Veramo can only manage the storage on Node or React Native (and not React). This code will be provided in the practical document's statement.
 - In case that Mike tries to access Gym's webpage without previously getting the anonymous credential, it appears an alert to inform about it.
 - There are two other disabled buttons called *MONSTERS LIBRARY* and *MONSTERS CAFÉ* that just represent other possible external services that require a VC from Monsters University to login.
- **Gym's webpage** shows a welcome sentence to highlight that the student has been redirected there successfully.

University's Express API performs the following tasks:

- It stores the traditional login credentials in a JSON file and passwords are encrypted with a strong Key Derivation Function (KDF) called **scrypt-pbkdf**.
- It verifies the JWT
- It sends by POST the VC when the button *GET ANONYMOUS STUDENT CREDENTIAL* is clicked by Mike.

Gym's Express API verifies Mike's VP.

The actual data that is shared in this system is specified below:

Monsters University's DID:

```
did:ethr:ropsten:0x03d8fc8ec731cdc17f4046edae7ad519f4c6bf2c3c1339ffd  
119b020f4a870788
```

Mike's DID:

```
did:ethr:ropsten:0x03de15fbcc72382b54d554421b561dc054fac932a60eacae0e  
e072892b8da112f4
```

Verifiable Credential:

```
{  
  "credentialSubject": {  
    "claims": {  
      "universityName": "Monsters University",  
      "student": true,  
      "expDate": "2022-08-01T00:00:00.000Z"  
    },  
    "id": {  
      "did":  
"did:ethr:ropsten:0x03de15fbcc72382b54d554421b561dc054fac932a60eacae0ee072892b8da112f  
4",  
      "controllerKeyId":  
"04de15fbcc72382b54d554421b561dc054fac932a60eacae0ee072892b8da112f46c622f7616ab4f9c9f  
3270ef36fab479e4f8a3a1630c4231e5bf28e3c4152923",  
      "keys": [  
        {  
          "type": "Secp256k1",  
          "kid":  
"04de15fbcc72382b54d554421b561dc054fac932a60eacae0ee072892b8da112f46c622f7616ab4f9c9f  
3270ef36fab479e4f8a3a1630c4231e5bf28e3c4152923",  
          "publicKeyHex":  
"04de15fbcc72382b54d554421b561dc054fac932a60eacae0ee072892b8da112f46c622f7616ab4f9c9f  
3270ef36fab479e4f8a3a1630c4231e5bf28e3c4152923",  
          "meta": {  
            "algorithms": [  
              "ES256K",  
              "ES256K-R",  
              "eth_signTransaction",  
              "eth_signTypedData",  
              "eth_signMessage"  
            ]  
          },  
          "kms": "local"  
        }  
      ]  
    }  
  }  
}
```

[illegible]

The resulting website is shown in the screenshots below:

Monsters University login page, where Mike introduces his credentials (*Figure 6*):

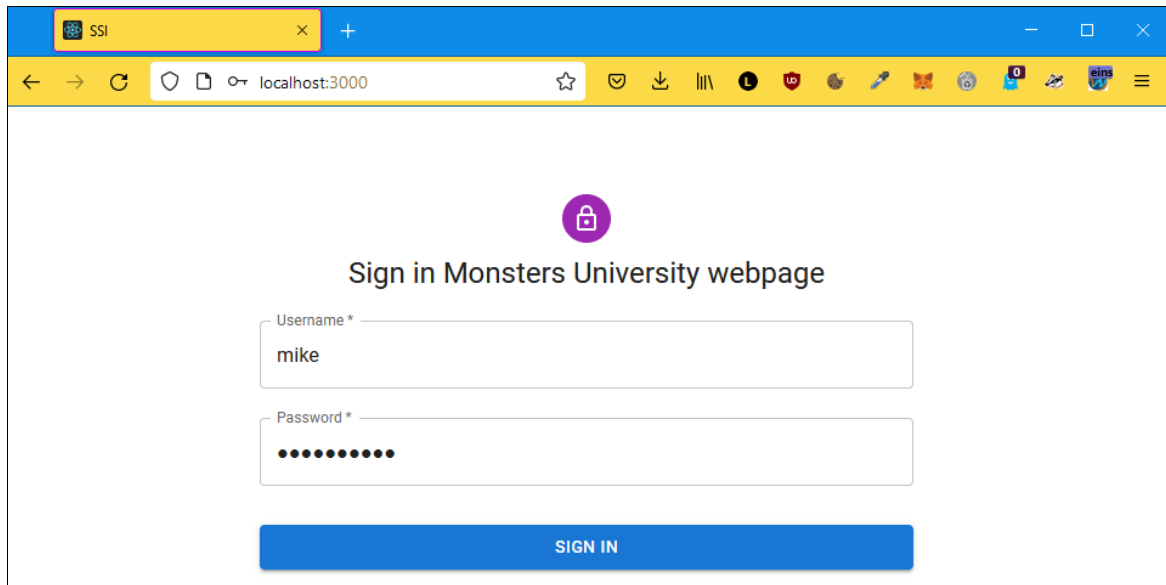


Figure 6 - Monsters University login page

Monsters University's web page, where Mike can access special services after getting an anonymous credential from the trusted issuer (*Figure 7*):

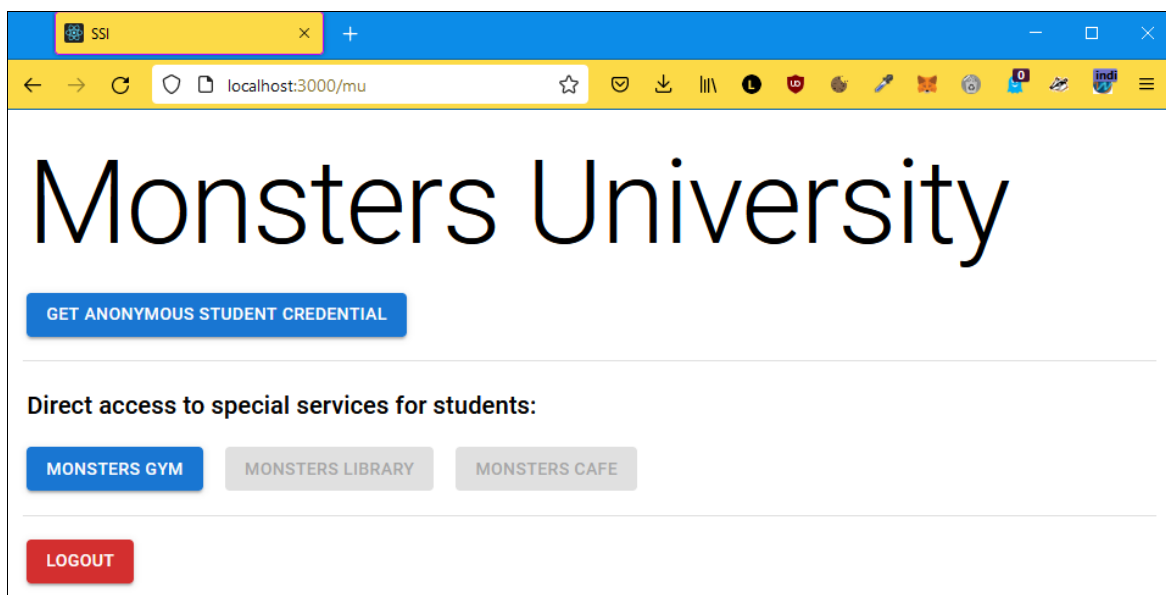


Figure 7 - Monsters University's web page

If Mike tries to access to Monsters Gym without an anonymous credential, an alert appears to inform about it (*Figure 8*):

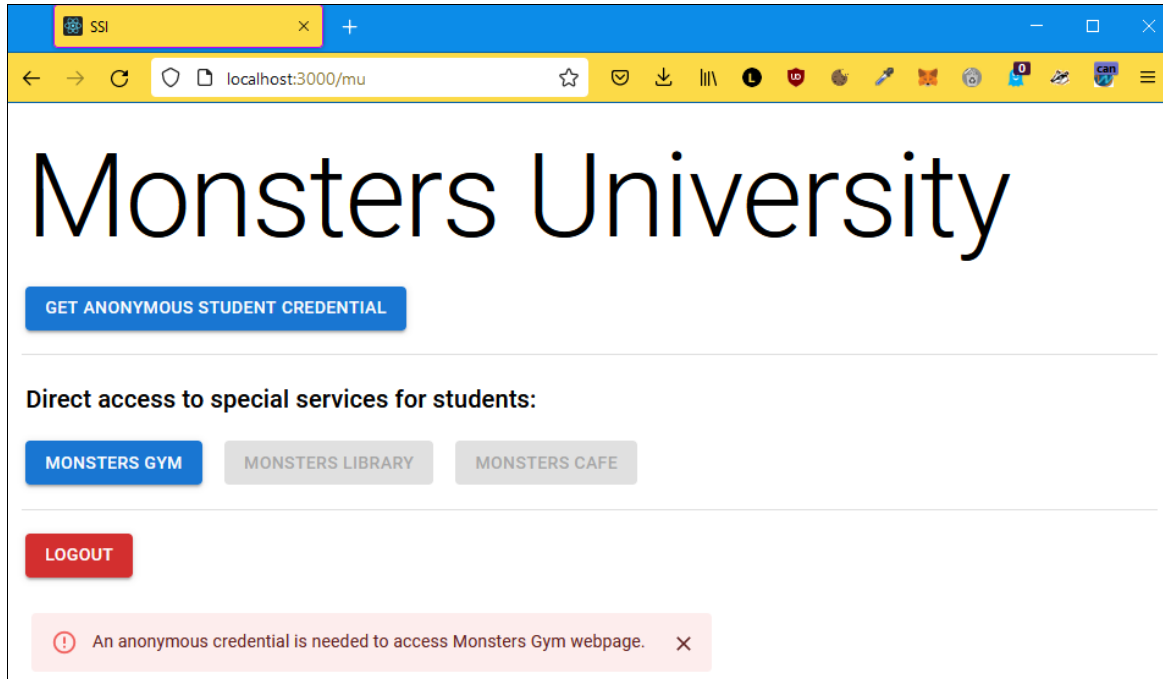


Figure 8 - Monsters University web page. Red alert

A green alert appears when Mike obtains the VC successfully (*Figure 9*):

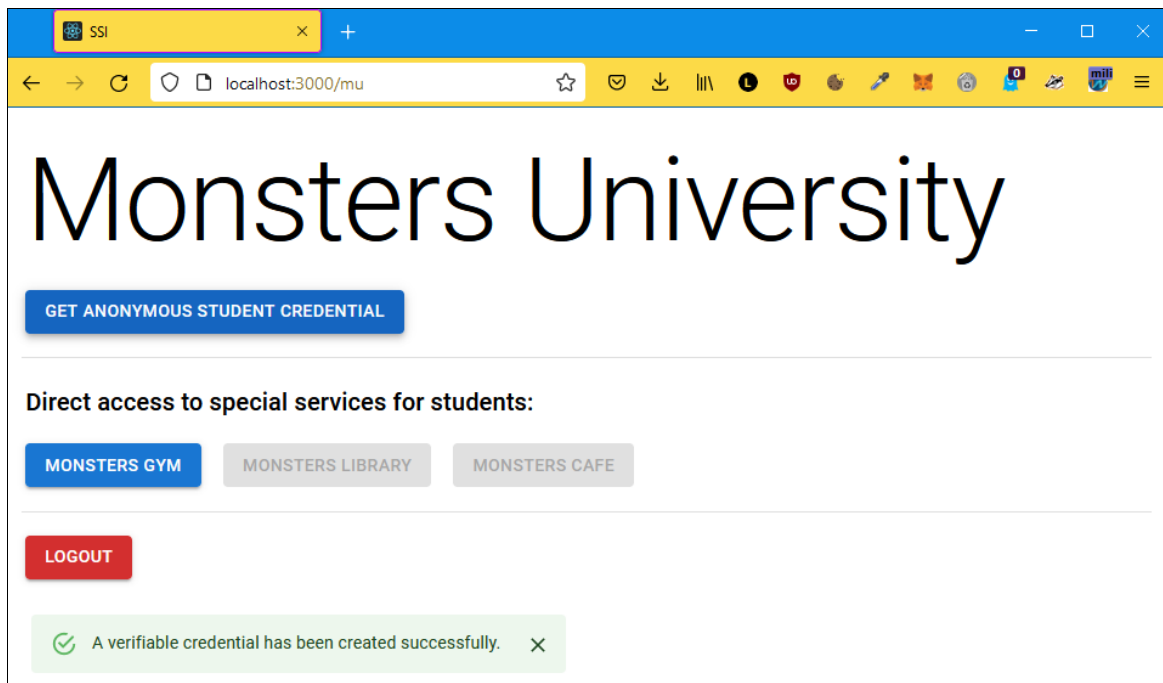


Figure 9 - Monsters University web page. Green alert

After Monsters Gym's server verifies Mike's VP, a welcome sentence is shown (*Figure 10*):

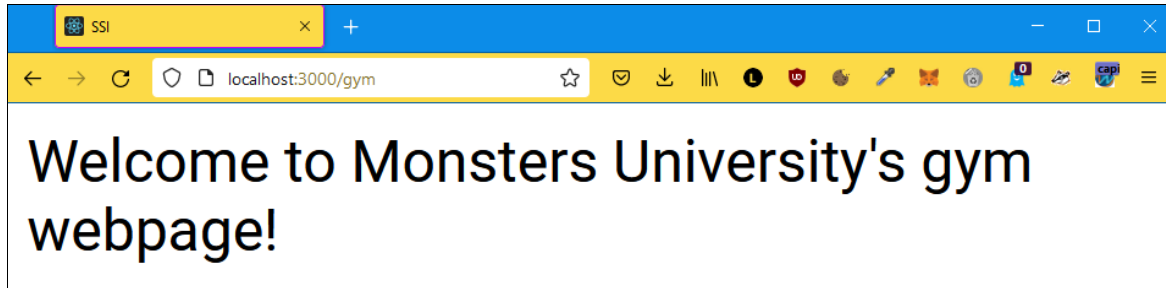


Figure 10 - Monsters Gym web page

10. Conclusions and future development

Two teaching practices have been created successfully. The first one is meant to let the student understand the core architecture, the data model, and the representations of Distributed Identifiers proposed by W3C and get used to basic management using Veramo APIs.

The second one consists of a more complex work that applies the SSI protocol in a system that could be deployed in a real-life use case. A VC is emitted to users authenticated in the webpage of a university and they can use it to log in other systems.

To keep the second practical work simple, all information about DIDs is stored in the browser local storage, but ideally it should be used a cryptographic wallet instead.

In the future it could be developed a project in which users can efficiently manage different identities with cryptographic wallets. A new identity would be created every time they get a VC.

Another interesting idea would be to integrate Zero Knowledge, so that users can provide specific information derived from their claims. An example could be developing a system in which users have a claim about the day they were born, and they can proof they are of legal age without revealing any more information.

11. References

- [1] Ethr DID. A Scalable Identity Method for Ethereum Addresses. [Online] Available: <https://developer.uport.me/ethr-did/docs/index>
- [2] Dr. Christian Lundkvist, Rouven Heck, Joel Torstensson, Zac Mitton, Michael Sena. "UPORT: A PLATFORM FOR SELF-SOVEREIGN IDENTITY". DRAFT VERSION, February 2017. [Online] Available: https://whitepaper.uport.me/uPort_whitepaper_DRAFT20170221.pdf 22/06
- [3] GatacaID. "Gataca DID Method Specification", April 2020 [Online] Available: <https://github.com/gataca-io/gataca-did-method>
- [4] W3C. "Decentralized Identifiers (DIDs) v1.0", August 2021 [Online] Available: <https://www.w3.org/TR/did-core/>
- [5] W3C. "Verifiable Credentials Data Model v1.1", March 2022 [Online] Available: <https://www.w3.org/TR/vc-data-model/>
- [6] Veramo (v3.1.2). [Online] Available: <https://veramo.io/>
- [7] InvisionApp. [Online] Available: <https://www.invisionapp.com/>
- [8] Mircea Nistor, Peter Grassberger, Zachary Carlin. "ETHR DID Method Specification", June 2022 [Online] Available: <https://github.com/decentralized-identity/ethr-did-resolver/blob/master/doc/did-method-spec.md>

12. Glossary

- Self-Sovereign Identity (SSI)
- Distributed Identifier (DID)
- Verifiable Credential (VC)
- Virtual Machine (VM)
- World Wide Web Consortium (W3C)
- Verifiable Presentation (VP)
- JSON Web Signature (JWS)
- Zero-Knowledge Proofs (ZKPs)
- Software Development Kit (SDK)
- Decentralised Applications (dApps)
- Minimum Viable Product (MVP)
- User Interface (UI)
- Smart Contract (SC)
- Distributed Ledgers Technologies (DLTs)
- Ether (ETH)
- Selective Disclosure Request (SDR)
- Key Derivation Function (KDF)