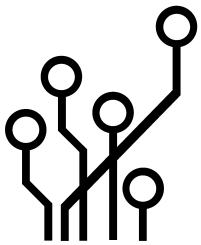


PROJECT ORIGIN

projectorigin2023@gmail.com

Specifica Tecnica

Versione	0.1.0
Responsabile	
Redattori	Corbu Teodor Mihail
Verificatori	
Uso	Esterno
Destinatari	*ProjectOrigin* Prof. Vardanega Tullio Prof. Cardin Riccardo

Descrizione
Nel presente documento si fornisce una visione approfondita dell'architettura, del design e delle specifiche tecniche del progetto *Personal Identity Wallet*

Registro delle modifiche

Vers.	Data	Autore	Ruolo	Descrizione
0.1.1	2023-08-24	Corbu Teodor	Analista	Aggiunte parti Introduzione (capitolo Architettura), Data Scrapper, React Material UI
0.1.0	2023-08-25	Andreetto Alessio	Verificatore	Verifica$_g$del documento
0.0.1	2023-08-24	Corbu Teodor	Analista	Creazione struttura documento, aggiunta Introduzione e spiegazione dei database

Indice

Elenco delle figure

Elenco delle tabelle

1 Introduzione

1.1 Scopo del documento

La Specifica Tecnica si pone come obiettivo di descrivere in modo esaustivo l'organizzazione della struttura del software, delle tecnologie adottate e delle scelte architetturali compiute dal gruppo durante le fasi di progettazione e di codifica del prodotto.
All'interno del documento si possono trovare gli schemi delle classi per delineare l'architettura e le funzionalità chiave del prodotto, con l'obiettivo di fornire una comprensione completa e chiara del sistema e delle interazioni interne.
Il documento contiene anche una sezione per i requisiti che vengono soddisfatti dal prodotto; questo permette al gruppo di valutare il progresso del lavoro e di tener traccia degli obiettivi imposti.

1.2 Scopo del prodotto

Lo scopo del prodotto è quello di creare una versione semplificata di un applicativo per implementare e rilasciare un "portafoglio di identità digitale" conforme a un insieme di standard, in modo che possa essere utilizzato con qualsiasi servizio, che adotti tale struttura, in qualsiasi paese dell'UE.
In particolare, si dovrà realizzare una web app$_g$ avendo queste componenti architetturali:

- Un componente back-office per consentire al dipendente dell'organizzazione emittente di verificare$_g$ manualmente la richiesta di credenziali e autorizzarne l'emissione;
- Un componente di interazione con l'utente dimostrativo per consentire all'utente (titolare) di navigare e richiedere specifiche credenziali da un emittente (ad esempio, il sito di una demo universitaria);
- Un componente di interazione con l'utente dimostrativo per consentire all'utente (titolare) di navigare un sito verificatore$_g$ e fornire le credenziali richieste;
- Un'app front-end per l'utente per archiviare e gestire le proprie credenziali;
- Un componente di comunicazione per consentire lo scambio di credenziali/presentazioni secondo un protocollo standard - il componente di comunicazione sarà implementato tre volte nei tre contesti (lato emittente, lato titolare, lato verificatore).

1.3 Note Esplicative

Alcuni termini utilizzati nel documento possono avere significati ambigui a seconda del contesto. Al fine di evitare equivoci, è stato creato un *Glossario v.1.0.0* contenente tali termini e il loro significato specifico. Per segnalare che un termine è presente nel *Glossario v.1.0.0* , sarà aggiunta una "g" a pedice accanto al termine corrispondente nel testo.

1.4 Riferimenti

1. Normativi:

- ***Norme di Progetto v.1.0.0*** : contengono le norme e gli strumenti per gli analisti;
- **Capitolato d'appalto C3**: https://www.math.unipd.it/~tullio/IS-1/2022/Progetto/C3.pdf;
- **Regolamento del progetto didattico:**: https://www.math.unipd.it/~tullio/IS-1/2022/Dispense/PD02.pdf.

2. Informativi:

- **Analisi dei Requisiti v1.0.0**;

- **Qualità di prodotto – slide T8 di Ingegneria del Software:** : https://www.math.unipd.it/~tullio/IS-1/2022/Dispense/T08.pdf;
- **Qualità di processo – slide T9 di Ingegneria del Software:** : https://www.math.unipd.it/~tullio/IS-1/2022/Dispense/T09.pdf;
- **Verifica e Validazione: introduzione – slide T10 di Ingeneria del Software:**: https://www.math.unipd.it/~tullio/IS-1/2022/Dispense/T10.pdf;
- **Verifica e Validazione: introduzione – slide T11 di Ingeneria del Software:**: https://www.math.unipd.it/~tullio/IS-1/2022/Dispense/T11.pdf;
- **Verifica e Validazione: introduzione – slide T12 di Ingeneria del Software:**: https://www.math.unipd.it/~tullio/IS-1/2022/Dispense/T12.pdf.

2 Architettura

2.1 Introduzione

In questo capitolo verranno descritte brevemente le parti di back-end e di front-end delle 3 webapp. La descrizione dettagliata sarà presente nei capitoli seguenti.

- **originIssuer**: originIssuer è la parte front-end della webApp "issuerApp". Si occupa di comunicare con l'utilizzatore dell'issuerApp, ricevendo i dati in input e restituendo i dati calcolati dal back-end (mediante l'intermediario chiamato Controller). L'interfaccia permette all'utilizzatore di:
  - effettuare il login;
  - effettuare la registrazione;
  - permette di vedere le richieste di credenziale da parte degli Account User (soltanto se si effettua il login con l'account Admin);
  - permette di vedere (tramite una pagina web) la parte di verifica e di validazione delle richieste di credenziale degli account User (soltanto se si effettua il login con l'account Admin);
- **originIssuerApi**: originIssuerApi è la parte back-end della webApp "issuerApp". Si occupa di calcolare tutti i possibili risultati, che poi verrano poi visualizzati nel front-end della webapp. Il back-end comunica con il controller.Il controller, prima di comunicare con il back-end dopo un input fornito dall'utilizzatore della webApp, effettua alcuni controlli, come per esempio di format (es. se un campo accetta soltanto una variabile numerica, non accetterà altri tipi di variabili e l'operazione si arresta ancor prima di mettere in esecuzione il back-end),alcuni controlli logici,ecc.
- **originVerifier**: originVerifier è la parte front-end della webApp "verifierApp". Si occupa di comunicare con l'utilizzatore del verifierApp, ricevendo i dati in input e restituendo i dati calcolati dal back-end (mediante l'intermediario chiamato Controller). L'interfaccia permette all'utilizzatore di:
  - effettuare la connessione al proprio wallet (comunicando in questo caso anche con walletApp) per successivamente effettuare l'autenticazione alla piattaforma;
  - una volta collegato il proprio wallet alla piattaforma, accedere ai servizi offerti da verifierApp;
  - ci sarà un verificatore (non necessariamente umano, potrebbe essere in alcuni casi anche un controllo automatico) che verifica le richieste di servizi effettuate dall'utente;
- **originVerifierApi**: originVerifierApi è la parte back-end della webApp "verifierApp". Si occupa di effettuare controlli e di verificare che la credenziale presentata sia valida. Per esempio, il verifier potrebbe controllare che: il documento presentato per accedere al servizio non sia scaduto, che il nome presente nel documento coincida con il nome dell'utente, ecc. In generale, nel back-end si verificano controlli di compatibilità fra i dati scritti nel documento presentato e i dati effettivi che possiede l'utente (data di nascita, luogo di nascita, nome e cognome,ecc.)
- **originWallet**: originWallet è la parte front-end della webApp "walletApp". L'interfaccia permette all'utilizzatore di:
  - aggiungere i documenti digitali al proprio wallet;
  - vedere i documenti che sono già conservati all'interno del proprio wallet;
  - avere una funzionalità (pagina dedicata) per permettere all'utente di cancellare i propri documenti che non desidera più conservare;
- **originWalletApi**: originWalletApi è la parte back-end della webApp "walletApp". Si occupa di conservare ed organizzare i documenti digitali dell'utente presenti, appunto, nel suo portafoglio digitale (wallet). Quindi si occupa di:
  - conservare i documenti dell'utente;

- verificare, mediante dei semplici controlli, di verificare che i documenti siano validi (per esempio, controllando la loro data di scadenza);
- cancellare dal wallet i documenti scaduti/non validi;
- permettere all'utente in possesso del proprio wallet di cancellare i documenti conservati desiderati;

2.2 Parte di back-end

(PARTE MANCANTE, PARTE ROSSA DRIVE)

2.3 Pattern architetturali e di design

(INTRODUZIONE ANCORA DA SISTEMARE SU DRIVE)

- **DATA SCRAPPER**: la classe "DataScrapper" contiene delle funzioni per accedere e modificare i dati nel database. DataScrapper è una classe di appoggio e verrà successivamente utilizzata dalla classe DatabaseStrategy. Le principali funzioni all'interno di questa classe sono:
  - **setStrategy(str)**: imposta la strategia di accesso ai dati (Pattern Data Strategy);
  - **insertAccount(email, hashed_pass, salt)**: inserisce un nuovo account nel database con l'indirizzo email, la password hash e il salt fornito;
  - **getAccountByEmail(email)**: restituisce un account dal database in base all'indirizzo email ;
  - **getAccountById(id)**: restituisce un account dal database in base all'ID fornito;
  - **insertSys_admin(accountId, role)**: inserisce un nuovo sys_admin (amministratore di sistema) nel database con l'ID dell'account e il ruolo specificato;
  - **insertSys_adminAccount(email, hashed_pass, salt, role)**: inserisce un nuovo account di amministratore di sistema nel database con l'indirizzo email, la password hash, il salt e il ruolo forniti;
  - **getSys_adminById(id)**: restituisce un amministratore di sistema dal database in base all'ID fornito;
  - **insertUser(accountId)**: inserisce un nuovo utente nel database con l'ID dell'account fornito;
  - **insertUserAccount(email, hashed_pass, salt)**: inserisce un nuovo account utente nel database con l'indirizzo email, la password hash e il salt forniti;
  - **getUserById(id)**: restituisce un utente dal database in base all'ID fornito;
  - **getVCSRequestById(id)**: restituisce una richiesta VCS dal database in base all'ID fornito;
  - **getVCSRequestsMarByUserId(id)**: restituisce le richieste VCS di stato civile associate a un utente in base all'ID utente fornito;
  - **getVCSRequestMarById(id)**: restituisce una richiesta VCS di stato civile dal database in base all'ID fornito;
  - **insertVCSRequestMar(applicantId, status, personalIdentifier)**: inserisce una nuova richiesta VCS di stato civile nel database con l'ID dell'applicante, lo stato e l'identificatore personale forniti;
  - **getVCSRequestsPidByUserId(id)**: restituisce le richieste VCS PID (Personal Identifier) associate a un utente in base all'ID utente fornito;
  - **getVCSRequestPidById(id)**: restituisce una richiesta VCS PID dal database in base all'ID fornito;
  - **insertVCSRequestPid(applicantId, currentAddress, dateOfBirth, familyName, firstName, gender, nameAndFamilyNameAtBirth, personalIdentifier, placeOfBirth)**: inserisce una nuova richiesta VCS PID nel database con i dettagli specificati;

- **getVCSRequestVerification(id)**: restituisce una verifica della richiesta VCS dal database in base all'ID fornito;
- **updateVCSRequestReleased(vcs_requestId, released)**: aggiorna lo stato di rilascio di una richiesta VCS specificata nel database;
- **updateVCSRequestVerificationStatus(vcs_requestId, status)**: aggiorna lo stato di verifica di una richiesta VCS specificata nel database;
- **getVCSRequestsPending()**: restituisce le richieste VCS con lo stato "in sospeso" dal database;
- **getVCSRequestsNotPending()**: restituisce le richieste VCS non "in sospeso" dal database;
- **insertVCSRequestVerification(vcs_request, admin_verifier, status)**: inserisce una nuova verifica di richiesta VCS nel database con i dettagli specificati.

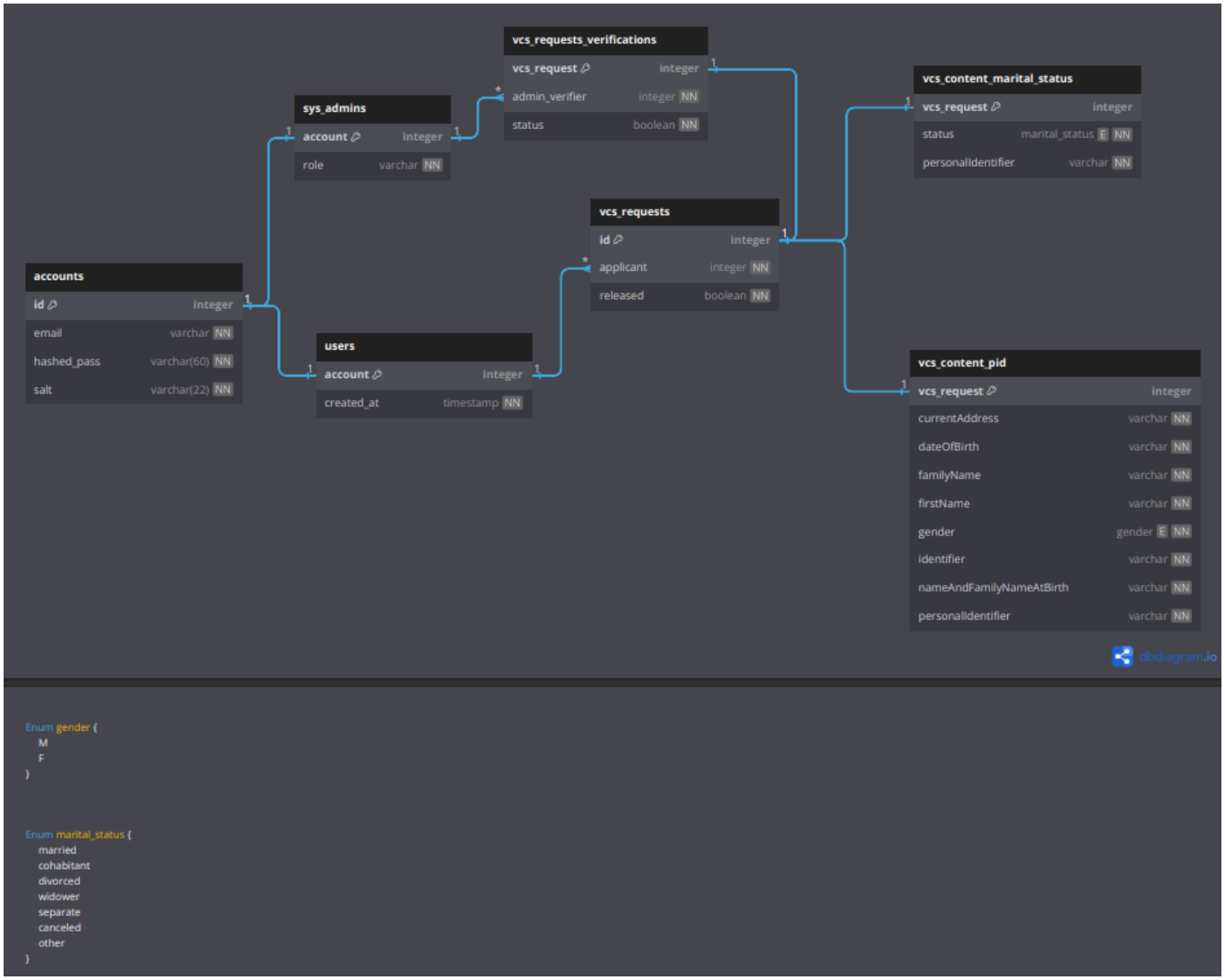
- **DATABASE STRATEGY**: (ASPETTO CONFERMA SE POSSO STENDERE)
- **ROUTING**: (DA SISTEMARE)

2.3.1 Database

2.3.1.1 Introduzione

Tenendo conto dei requisiti e dell'architettura da rispettare del capitolato, si sono sviluppati 2 database: "issuerdb" per l'Issuer e "walletdb" per lo User (cioè l'utente in possesso del proprio portafoglio digitale).

2.3.1.2 Issuerdb



L'immagine sopra riportata descrive il database "issuerdb" implementato mediante un grafico entità-relazioni (schema ER).
Issuerdb è stato pensato per gestire e conservare le informazioni legate agli utenti, alle richieste di certificati digitali (VCS requests) e alle verifiche dei certificati stessi. Per quanto richiesto dal capitolato, e per rispettare la logica dietro tutto il meccanismo dell'Issuer, si distinguono 2 tipi diversi di account: gli account "sys_admins" e gli account "users".

- "sys_admins" sono gli account utilizzati dagli amministratori di sistema, cioè quelle entità che si occupano di approvare (o meglio, verificare) le richieste degli "users" (VCS requests).

- "users", invece, sono gli account utilizzati dai semplici utilizzatori del servizio. Si occupano semplicemente di effettuare delle richieste di certificati di loro interesse alle entità che si occupano di verificare i certificati.

Il contenuto della richiesta approvata e rilasciata può essere di 2 tipi soltanto: un "vcs_content_marital_status" (contenuto riferito allo stato di matrimonio di un utente) o un "vcs_content_pid" (contenuto riferito ad un documento PID di un utente).
Più in dettaglio:

Tipi enumerati: sono definite due tabelle ENUM per rappresentare:

- il genere maschile/femminile ("gender", che può avere come valori "M" o "F");
- lo stato civile ("marital_status", che può avere come valori "married", "cohabitant", "divorced","widower", "separate", "canceled", "other").

Queste tabelle verranno utilizzate per definire, rispettivamente, gli attributi "gender" della tabella "vcs_content_pid" e "marital_status" della tabella "vcs_content_marital_status".

Tabella "accounts":

- "id": identificativo univoco per l'account, chiave primaria;
- "email": indirizzo email dell'account, varchar not null;
- "hashed_pass": password hash dell'account, varchar massimo 60 caratteri not null;
- "salt": salt usato per la sicurezza nella creazione dell'hash della password dell'account dell'utente, varchar massimo 22 caratteri not null.

Tabella "users":

- "account": riferimento all'id della tabella "accounts", identificativo univoco per l'account degli utilizzatori, chiave primaria;
- "created_at": data di creazione dell'account, formato timestamp not null.

Tabella "sys_admins":

- "account": riferimento all'id della tabella "accounts", identificativo univoco per l'account degli amministratori, chiave primaria;
- "role": ruolo dell'amministratore di sistema, varchar not null.

Tabella "vcs_requests":

- "id": identificativo univoco della richiesta del certificato;
- "applicant": riferimento all'account dell'utente nella tabella "users", integer not null;
- "released": flag per indicare se il certificato è stato rilasciato, variabile booleana not null.

Tabella "vcs_requests_verifications":

- "vcs_request": riferimento all'id di una richiesta nella tabella "vcs_requests";
- "admin_verifier": riferimento all'account dell'amministratore di sistema;
- "status": stato della verifica, variabile booleana not null.

Tabella "vcs_content_pid":

- "vcs_request": riferimento all'identificativo di una richiesta nella tabella "vcs_requests" e "vcs_requests_verifications";
- "currentAddress": l'indirizzo corrente scritto nel documento PID, varchar not null;

- "dateOfBirth": data di nascita scritta nel documento PID, varchar not null;
- "familyName": cognome scritto nel documento PID, varchar not null;
- "firstName": nome scritto nel documento PID, varchar not null;
- "gender": genere scritto nel documento PID, enum con valori "M"/"F" not null;
- "identifier": identificatore scritto nel documento PID, varchar not null;
- "nameAndFamilyNameAtBirth": nome e cognome di famiglia scritto nel documento PID, varchar not null;
- "personalIdentifier": identificatore personale scritto nel documento PID, varchar not null.

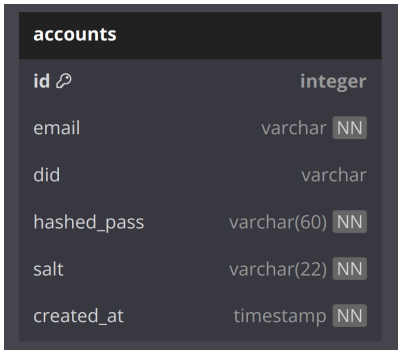
Tabella "vcs_content_marital_status":

- "vcs_request": riferimento all'identificativo di una richiesta nella tabella "vcs_requests" e "vcs_requests_verifications";
- "status": stato civile = stato di matrimonio, enum corrispondente ai valori della tabella "marital_status", not null;
- "personalIdentifier": identificatore personale per es. codice fiscale, varchar not null.

RELAZIONI FRA LE TABELLE

- **"accounts" – "sys_admins" e "users"**: "accounts" – "sys_admins" e "accounts" – "users" relazione 1-1. Un account appartiene a un solo utente, un utente può avere soltanto un account. Un account appartiene a un solo amministratore, un amministratore può avere solo un account. Un account è un account "sys_admins" oppure un account "users", non può essere entrambi. La relazione deve essere unica, e questo limite si controlla nel back-end (poiché nel database non c'è modo di controllare l'unicità della relazione). Nel back-end si controllerà anche se un account è sia un account amministratore che un account user (con conseguente errore).
- **"sys_admins" – "vcs_requests_verifications"**: relazione 1-N. Una verifica di richiesta appartiene al massimo ad un amministratore. Un amministratore può verificare molteplici richieste.
- **"users" – "vcs_requests"**: relazione 1-N. Una richiesta di credenziale può appartenere al massimo ad un user. Un user può fare molteplici richieste.
- **"vcs_requests_verifications" e "vcs_requests" – "vcs_content_marital_status" e "vcs_content_pid"**: similmente a prima, il contenuto del VCS o è un contenuto relativo al "marital status", oppure un contenuto relativo al "PID" (non entrambi, ma deve essere obbligatoriamente un contenuto di marital status oppure un contenuto di PID). Questa condizione sarà controllata nel back-end. La richiesta sarà rilasciata DOPO che la richiesta sarà approvata dal verificatore. Per essere rilasciata (dopo essere approvata), l'utente riceve la richiesta come credenziale (con conseguente stoccaggio nel "walletdb"). Una volta ricevuta, la richiesta verrà contrassegnata come "true" (nel campo "released" in "vcs_requests"). Invece, il campo "status" in "vcs_requests_verifications" indica se la richiesta è stata approvata dal verificatore. Se lo "status" sarà "true", allora la richiesta sarà pronta per il rilascio (quindi lo status "true" dell'attributo "released" di "vcs_requests" dipende dallo status "true" dell'attributo "status" di "vcs_requests_verifications"). Una richiesta può essere verificata una sola volta. Una verifica appartiene ad una e una sola richiesta. Se la richiesta non è stata ancora approvata non si può richiedere una riapprovazione. Se la richiesta è stata verificata con esito status="false" soltanto allora si può fare un altro tentativo di richiesta. Per questo motivo la relazione fra "vcs_requests" e "vcs_requests_verifications" è 1-1.

2.3.1.3 Walletdb



L'immagine sopra riportata descrive il database "walletdb" implementato mediante un grafico entità-relazioni (schema ER). Walletdb è stato pensato per gestire e conservare (fare lo "storing") le informazioni legate alle credenziali degli utenti, come espresso da capitolato.
Più in dettaglio:

Tabella "accounts":

- "id": identificativo univoco per l'account degli utenti, chiave primaria,
- "email": indirizzo email associato all'account, varchar not null,
- "did": documento did associato all'account dell'utente, varchar,
- "hashed_pass": password hash dell'account dell'utente, varchar massimo 60 caratteri not null,
- "salt": salt usato per la sicurezza nella creazione dell'hash della password dell'account dell'utente, varchar massimo 22 caratteri not null,
- "created_at": data di creazione dell'account, timestamp not null.

2.4 Parte di front-end

(PARTE MANCANTE, PARTE ROSSA DRIVE)

2.4.1 React Material-UI

Material-UI è una libreria di componenti UI (User Interface) per React fatto da Google. Material-UI è una libreria che viene utilizzata principalmente per realizzare interfacce utente dinamiche e personalizzate. Segue il design system "Material Design", che si occupa di fornire linee guida di design per la realizzazione di applicazioni e siti web. Alcune caratteristiche che sono state di gradimento per la scelta di questa libreria all'interno del progetto Personal Identity Wallet sono state:

- la libreria offre una vasta gamma di componenti UI predefiniti (pre-implementati). Questo permette di facilitare lo sviluppo dell'interfaccia grafica, con un notevole accorciamento delle tempistiche;
- i componenti sono inoltre facilmente personalizzabili, il ché permette di integrarsi facilmente all'interno del progetto e di soddisfare i requisiti imposti dal capitolato;
- è una libreria ben documentata: in caso di dubbi si può consultare la documentazione ufficiale. Inoltre nella community si possono trovare numerosi esempi pratici per ottenere familiarità (oppure cercare risposte ai vari problemi di implementazione) con Material-UI;
- è stata realizzata specificamente per React. Questo significa che si integra perfettamente con il framework;
- inoltre, è compatibile con una buona parte delle altre librerie di React;

2.5 Componente di API

2.6 Design pattern