**Specifiche Esame per superamento del Corso Programmazione Avanzata**

**A.A. 2021/2022**

Gruppo: SI

---------------------------------------------------------------------------

Si chiede di realizzare un back-end utilizzando i seguenti framework / librerie:

* Node.JS
* Express
* Sequelize
* RDBMS a scelta del gruppo (es. Postgres, MySQL, sqlite,…)

Si installi il pacchetto openssl: <https://mirror.firedaemon.com/OpenSSL/openssl-3.0.3.zip>

Usare i file della CA ed il template del file di configurazione sono disponibili al seguente indirizzo:

<https://s3.amazonaws.com/vrai.univpm/PA/2022/06/rootCACert.pem>

<https://s3.amazonaws.com/vrai.univpm/PA/2022/06/rootCAKey.pem>

<https://s3.amazonaws.com/vrai.univpm/PA/2022/06/openssl.cnf>

Per generare le chavi e relativo certificato è necessario eseguire i seguenti comandi:

openssl req -new -config openssl.cnf -keyout my\_private.key -out my.csr

openssl x509 -req -days 365 -in my.csr -CA rootCACert.pem -CAkey rootCAKey.pem -CAcreateserial -out user.crt -extensions user\_crt -extfile openssl.cnf

Per firma singola:

openssl cms -sign -in demo.txt -out demo.txt.p7m -nodetach -signer user.crt -inkey my\_private.key -cades -outform DER -stream -binary

Per firma multipla (es 3 firmatari) :

openssl cms -sign -in demo.txt -out demo.txt.p7m -nodetach -signer user.crt -inkey my\_private.key -cades -outform DER -stream -binary -signer user1.crt -inkey my\_private1.key -signer user2.crt -inkey my\_private2.key

Il file openssl.cnf deve essere personalizzato sulla base dei dati che vengono forniti dall’utente. I dati possono essere derivati da quanto presente nel token JWT e devono essere i seguenti (con relativi esempi):

* commonName=Adriano Mancini
* countryName=IT
* stateOrProvinceName=FM
* localityName=Fermo
* organizationName=ACME
* organizationalUnitName=IT
* emailAddress=demo@mailinator.com
* serialNumber=MNCDRN82T30D542U
* SN=Mancini

dnQualifier assume valore fisso pari a 20175007693.

Dunque, per generare un i certificati e le chiavi deve essere generato l’opportuno file di configurazione e poi lanciare i comandi di cui sopra. Si consiglia di dare come prefisso ai file il codice fiscale dell’utente.

**Descrizione del progetto:**

Si realizzi un sistema che consenta di gestire il processo di creazione di certificati e firma di documenti mediante openssl. In particolare, il back-end deve prevedere che un utente possa effettuare chiamate (payload in JSON) per:

* Creare per un utente se non esiste un certificato (necessaria autenticazione mediante token JWT) prelevando i valori per la creazione del certificato direttamente dal JWT
* Invalidare un certificato associato all’utente (necessaria autenticazione mediante token JWT)
* Richiedere la firma di un documento che viene inviato al back-end (es. docx, PDF,…); in particolare deve essere implementato un sistema di firma che preveda in alternativa:
  + Modalità #1: Firma del singolo utente.
  + Modalità #2: Firma di più utenti; in tal caso è necessario specificare nella richiesta il codice fiscale di un utente (è necessario gestire il caso nel quale si richiede un processo di firma con uno o più utenti che non sono registrati ovvero impedendo il processo di firma).

Il processo di firma richiede che ogni utente, in riferimento al processo di firma al quale appartiene, chieda al back-end un challenging code. Il challenging code è l’insieme di due numeri distinti che variano tra 0 e 15 (es 2 e 4). L’utente, preleva da una sua “tabella” i codici in corrispondenza dei due numeri distinti nell’ordine stabilito ed effettua la richiesta per la conferma. Se sono trascorsi più di N minuti con N impostabile mediante .env è necessario richiedere un nuovo challenging code. La tabella dell’utente è presente anche nel back-end. Il back-end verifica che i due codici forniti siano corretti ed in caso conferma che l’utente ha avviato il processo di firma. La firma viene apposta solo quando tutti gli utenti che partecipano al processo di firma hanno inserito i loro challenging code corretti. La tabella, per ogni utente dunque contiene 16 valori; ogni valore è formato da una stringa di due caratteri che appartengono al seguente insieme di caratteri a-Z0-9 dunque a1 01 Za ZZ sono esempi di sequenze.

* Annullare un dato processo di firma (necessaria autenticazione mediante token JWT); solo chi ha avviato il processo di firma lo può terminare).
* Chiedere lo stato di un processo di firma (necessaria autenticazione mediante token JWT); solo chi ha avviato il processo di firma può chiedere lo stato).
* Ottenere il file firmato (p7m) (necessaria autenticazione mediante token JWT); solo chi partecipa al processo di firma può scaricare il file).
* Per ogni richiesta andata a buon fine viene scalato un credito parti a K token con K uguale al numero di partecipanti al processo di firma (minimo è 1). La richiesta di firma deve essere rifiutata se il credito non è sufficiente. Il credito viene scalato a chi avvia il processo di firma.
* Restituire il credito residuo di un utente (necessaria autenticazione mediante token JWT)

I dati di cui sopra devono essere memorizzati in un database esterno interfacciato con Sequelize. La scelta del DB è a discrezione degli studenti.

Ogni utente autenticato (ovvero con JWT) ha un numero di token (valore iniziale impostato nel seed del database). La tariffazione segue quanto specificato sopra.

Nel caso di token terminati ogni richiesta da parte dello stesso utente deve restituire 401 Unauthorized.

Prevedere una rotta per l’utente con ruolo admin che consenta di effettuare la ricarica per un utente fornendo la mail ed il nuovo “credito” (sempre mediante JWT).

Il numero residuo di token deve essere memorizzato nel db sopra citato. Si deve prevedere degli script di seed per inizializzare il sistema.

Si chiede di utilizzare le funzionalità di middleware.

Si chiede di gestire eventuali errori mediante gli strati middleware sollevando le opportune eccezioni.

Si chiede di commentare opportunamente il codice.

**Note:**

Nello sviluppo del progetto è richiesto l’utilizzo di Design Pattern che dovranno essere documentati opportunamente nel Readme.MD.

I token JWT da usare possono essere generati attraverso il seguente link: <http://jwtbuilder.jamiekurtz.com/>

La chiave privata da usare lato back-end deve essere memorizzata un file .env e caricata mediante la libreria

**Specifiche Repository**

* Il codice deve essere reso disponibile su piattaforma github con repo pubblico
* Nel repository è obbligatario inserire un Readme.md che descriva:
  + Obiettivo del progetto
  + Progettazione
    - diagrammi UML
    - descrizione dei pattern usati motivandone la scelta
  + Come avviare il progetto mediante docker o docker-compose (preferibile) per comporre i servizi richiesti.
  + Test del progetto mediante chiamate effettuate con curl o wget o con Postman
* Il Readme.MD può essere redatto in lingua italiana o inglese (non vi saranno differenziazioni nel processo di valutazione)

**Specifiche Consegna**

* La consegna avviene esclusivamente mediante moodle all’indirizzo di seguito riportato dove dovranno essere indicati:
  + URL del repository pubblico
  + Commit id che verrà usata dal docente per effettuare la valutazione.
  + Data per lo svolgimento dell’esame
* Indirizzo per la consegna: <https://learn.univpm.it/mod/assign/view.php?id=332114>

Buon lavoro 😉

Il docente

Adriano Mancini

{

"commonName": "Lorenzto Tiseni",

"countryName": "IT",

"stateOrProvinceName": "AN",

"localityName": "Porto Recanati",

"organizationName": "ACME",

"organizationalUnitName": "IT",

"emailAddress": "demo2@mailinator.com",

"serialNumber": "TSNLNZ99E06E690J",

"dnQualifier":"20175007693",

"SN": "Tiseni "

}