**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,…)
* <https://plotly.com/nodejs/>

**Descrizione del progetto:**

Si realizzi un sistema che consenta di gestire il processo di compravendita di energia “locale”. Esistono N produttori e M consumatori. Un produttore può mettere a disposizione di un acquirente in una fascia oraria di 1h un certo quantitativo di energia. Il processo di compravendita ha un orizzonte temporale di 1 giorno, ovvero si compra oggi uno o più “slot” per domani.

I produttori mettono possono mettere a disposizione degli slot temporali della durata di 1h con un valore pari a x kWh. Ogni produttore ha una capacità massima di produzione oraria che non deve essere superata (questo valore è diverso per ogni produttore e per ogni fascia oraria).

Ogni produttore specifica anche la fonte con la quale genera energia e si identificano le seguenti possibilità: Fossile; Eolico; Fotovoltaico. Per ogni fascia oraria il produttore indica il quantitativo che può erogare che deve essere minore o uguale al quantitativo che può fornire.

Un compratore può acquistare energia da uno o più venditori per un giorno. Per una fascia oraria un compratore può comprare solo da un produttore. Un compratore non può comprare per una fascia oraria un quantitativo superiore a quello massimo erogabile da quel produttore.

I consumatori possono opzionare gli slot entro le 24h. (es. slot 7 Luglio 2022 ore 15:00 è prenotabile fino alle ore 14:59 del 6 Luglio 2022).

Se un produttore riceve per una fascia oraria più richieste, allora devono essere verificati i seguenti casi:

* ~~Se la somma delle richieste è inferiore o uguale alla capacità erogabile per quella fascia oraria allora non vi sono particolari azioni da svolgere.~~
* ~~Se la somma delle richieste è superiore o uguale alla capacità erogabile per quella fascia oraria allora il produttore potrà decidere se accettare le richieste effettuando un taglio lineare a quanto richiesto dai vari consumatori. Il taglio è proporzionale al quantitativo richiesto.~~

Si chiede dunque di creare delle API che consentano di:

* ~~Creare un nuovo produttore specificando quanto necessario. I seguenti campi sono obbligatori.~~
  + ~~Tipologia energia erogata~~
  + ~~Quantitativo massimo erogabile per ogni fascia oraria~~
  + ~~Costo energia per ogni kWh (token; non necessariamente di tipo intero)~~
  + ~~Emissione di CO~~~~2~~in termini di g CO2/kWh (es.[~~https://www.isprambiente.gov.it/files2020/pubblicazioni/rapporti/Rapporto317\_2020.pdf~~](https://www.isprambiente.gov.it/files2020/pubblicazioni/rapporti/Rapporto317_2020.pdf) ~~pag. 28)~~
* ~~Creare un nuovo consumatore. I campi sono a scelta degli studenti, ma deve essere previsto un valore inziale di credito.~~
* ~~Aggiornare i quantitativi erogabili per ogni fascia oraria da un produttore.~~
* ~~Aggiornare i costi dell’energia associati a una o più fasce orarie del produttore.~~
* ~~Dare la possibilità ad un consumatore di riservare uno slot per il giorno seguente in una fascia oraria. L’acquisto minimo è di 0.1kWh~~
* **~~Dare la possibilità ad un consumatore di modificare (anche cancellare, ovvero imponendo una quantità parti a zero) i quantitativi richiesti per uno o più slot. Se la cancellazione avviene prima delle 24h allora non vi sono costi; se avviene in un periodo temporale inferiore o uguale alle 24 viene addebitato l’intero costo.~~**
* ~~Dare ad un produttore la possibilità di verificare le richieste per il giorno seguente; dare la possibilità di filtrare per fasce orarie (es. 10:00 – 17:00). Tale rotta deve tornare per ogni fascia oraria la~~ **~~% di occupazione~~** ~~rispetto alla capacità erogabile in quella fascia oraria~~
* ~~Scalare il credito di un utente al momento della “prenotazione” dello slot. Restituire il credito se viene effettuata la cancellazione entro i tempi stabiliti~~
* ~~Dare la possibilità ad un utente di ritornare i vari acquisti effettuati dando la possibilità di filtrare per:~~
  + ~~Produttore~~
  + ~~Tipologia di fonte usata~~
  + ~~Intervallo temporale (data – ora) esempio: 2022-06-07 00:00:00|2022-07-07 23:59:59~~
* ~~Dare ad un consumatore la possibilità di verificare la sua impronta di carbonio in un dato intervallo di tempo sulla base degli acquisti fatti.~~
* Dare ad un produttore la possibilità di calcolare le seguenti statistiche rispetto ad un intervallo temporale per ogni fascia oraria:
  + % minima di energia venduta (venduta / erogabile \* 100)
  + % massima di energia venduta (vedi sopra)
  + % media di energia venduta (vedi sopra)
  + Deviazione standard della (vedi sopra)

Tali statistiche possono essere tornate sotto forma di:

JSON

Immagine (grafico a scelta del gruppo) mediante la libreria **plotly**.

* ~~Dare ad un produttore di calcolare in un intervallo temporale i guadagni~~

~~Le rotte di cui sopra devono essere autentica con JWT. Necessario che nel JWT ci siano i dati minimi associati all’utente ed anche il ruolo ovvero “producer”, “consumer” o “admin”.~~

~~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 il nuovo “credito” (rotta autenticata 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. 🡪 COR

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. (**MVC**, ~~ClientServer~~, **Singleton** per connessione DB, **COR**)

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

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