Specifiche Esame per superamento del Corso Programmazione Avanzata

A.A. 2023/2024 Gruppo: SI (gruppo)

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,…)
- Tesseract OCR https://www.npmjs.com/package/node-tesseract-ocr

Descrizione del progetto:

Si realizzi un sistema che consenta di gestire il calcolo di eventuali multe a seguito del passaggio di autoveicoli con classi differenti tra diversi varchi autostradali (es. sistema Tutor). Dovranno essere modellati le tipologie di veicolo che hanno limiti differenti. Dovranno inoltre essere modellati i varchi che hanno una posizione geografica nota. Dovranno essere inseribili i transiti impostando data e ora del passaggio e targa del veicolo lungo una tratta che ha un varco di inizio, un varco di fine ed una distanza. Un veicolo in un giorno può attraversare diversi varchi / tratte. L'inserimento del transito deve riportare anche le condizioni metereologiche del varco ovvero se era presente o meno pioggia in modo da ridurre il limite di velocità. Il sistema deve anche provvedere a generare delle infrazioni per superamento della velocità media tra due varchi limitrofi. Il sistema deve prevedere 3 diversi ruoli:

- Operatore
- Varco
- Automobilista

Devono essere predisposte le seguenti rotte:

- [U] CRUD per la gestione dei varchi (utente operatore)
- [U] CRUD per la gestione delle tratte (utente operatore); per tratta si intende l'associazione tra un varco di ingresso ed un varco di uscita con la relativa distanza.
- [U] CRUD per la gestione dei veicoli (utente operatore)
- [U] CRUD per inserimento transiti:
    - POST (inserimento) operatore OR varco
    - Nel caso di un varco dare la possibilità di fornire (devono essere implementate tutte e due le funzionalità:
        - JSON con dati del veicolo, data e ora.
        - Immagine con testo da riconoscere mediante tesseract. Usare una espressione regolare per validare la targa; in caso di errato riconoscimento memorizzare il transito come "illeggibile". Per semplicità la data e ora in questo caso si riferiscono alla data di ricezione lato server.
    - GET (ottenere specifico varco) solo operatore
    - DELETE e UPDATE solo operatore
    - PUT/PATCH solo operatore per aggiornare un transito (es. interpretando una immagine "dubbia"/"illeggibile" per l'OCR
- [U] GET che consente di ottenere la lista di transiti "illeggibili" anche filtrando per varco (utente operatore) dando la possibilità di visionare l'immagine.
- All'atto dell'inserimento di un transito valutare se è necessario creare in automatica una "multa".
- [U] Creare una rotta che data/e una o più targhe in ingresso ed un periodo temporale fornisca le multe con i dettagli del caso (tratta: varco in, varco out, velocità media, delta rispetto al limite, condizioni ambientali). L'utente può specificare il formato in uscita che è: JSON. Utente operatore non ha limiti mentre automobilista può vedere solo i veicoli ad esso associati.
- [U] Creare una rotta che consenta di scaricare un bollettino di pagamento in formato PDF che contenga targa, importo ed un QR-code che riporti la stringa <uuid pagamento>;<multa id>;<targa>;<importo>. <uuid pagamento> è un uuid generato in automatico associato alla multa.

- *Il gruppo dovrà dunque allegare anche delle immagini di esempio che consentano di effettuare delle prove.*

Si chiede di sviluppare il codice utilizzando typescript.

[U] corrisponde ad una rotta autenticata mediante 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.

Le richieste devono essere validate.

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.

Implementazione in typescript.

I token JWT da usare possono essere generati attraverso il seguente link: https://jwt.io/ (token JWT non deve contenere il payload della richiesta, ma solo i dati strettamente necessari per autenticare ed autorizzare le richieste).

La chiavi da usare lato back-end devono essere memorizzata un file .env

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 (casi d'uso, diagrammi delle sequenze)
        - descrizione dei pattern usati motivandone la scelta
    - Come avviare il progetto mediante docker-compose per comporre i servizi richiesti (fornire tutti i file neceessari).
    - Test del progetto mediante chiamate effettuate Postman / Newman (fornire collection)
- 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=531167

Buon lavoro 😉

Il docente, Adriano Mancini