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,…)

Descrizione del progetto:

Si realizzi un sistema che consenta di gestire il calcolo del costo dei parcheggi a seguito del passaggio di autoveicoli con classi differenti tra un varco di ingresso e di uscita. Un parcheggio può avere diversi punti di ingresso e diversi punti di uscita. Dovranno essere modellati le tipologie di veicolo che hanno poi costi differenti. Dovranno essere inseribili i transiti impostando data e ora del passaggio e targa del veicolo lungo un varco specifico di un dato parcheggio; la richiesta di inserimento deve essere rifiutata se non ci sono più posti disponibili all'interno del parcheggio. Un veicolo in un giorno può parcheggiare in diversi parcheggi. Il sistema deve anche provvedere a calcolare il costo del parcheggio in funzione della permanenza effettiva all'interno dello stesso (si lascia discrezione al gruppo per la gestione delle tariffe che devono comunque prevedere una differenziazione per tipologia del veicolo, fascia oraria e giorno della settimana es. festivo / feriale).

- Operatore
- Varco

Devono essere predisposte le seguenti rotte:

- [U] CRUD per la gestione dei parcheggi e dei relativi varchi (utente operatore; varchi possono essere anche di tipo bi-direzionale)
- [U] CRUD per la gestione delle tariffe per parcheggio (utente operatore) .
- [U] CRUD per inserimento transiti (ingresso o uscita):
    - POST (inserimento) operatore OR varco
    - GET (ottenere specifico varco) solo operatore
    - DELETE e UPDATE solo operatore
- All'atto dell'inserimento di un transito valutare se è necessario creare in automatica la fattura del parcheggio che viene associata all'utente (sistema di pagamento non deve essere creato)
- [U] Creare una rotta che data/e una o più targhe in ingresso ed un periodo temporale fornisca lo stato dei transiti con i dettagli del caso (varco in, varco out, tipologia veicolo, costo se disponibile). L'utente può specificare il formato in uscita che è: JSON, PDF (tutti e due i formati devono essere implementati. Utente operatore non ha limiti mentre automobilista può vedere solo i veicoli ad esso associati.
- [U] creare una rotta fruibile dall'operatore che fornisca delle statistiche (operatore può filtrare per range temporale); l'utente può scegliere il formato in uscita se JSON o PDF:
    - Fatturato di ciascun parcheggio
    - Numero medio di posti liberi per parcheggio distinguendo per fascia oraria.
- [U] creare una rotta fruibile dall'operatore che fornisca delle statistiche per un determinato parcheggio (operatore può filtrare per range temporale):
    - Numero totale di transiti distinti anche per tipologia di veicolo e per fascia oraria
    - Fatturato

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.

Ogni utente autenticato (ovvero con JWT) ha un numero di token (valore iniziale impostato nel seed del database).

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