TCM - Homework 3

OSLO

Fagiani Lorenzo – 1068955

Gambirasio Cristiano – 1066866

Villa Valentina – 1067719

1. /register_race

- Per passare i dati al Database sfruttiamo la funzione dynamoS3 che si attiva alla creazione di un oggetto S3.
- Nella funzione creiamo un file xml contenente i parametri di input e un token univoco generato tramite la libreria ‘uuid’, questi vengono caricati sul Bucket.
- Alla creazione di un nuovo file xml sul bucket si attiva la funzione dynamoS3, creata nei passaggi precedenti, che registra i dati nella tabella di DynamoDB.

2. /uploadxml

- Questa funzione Lambda prende il contenuto passato nella richiesta post, lo trasforma in formato JSON.
- Sfrutta il token passato come parametro nella richiesta per ricercare nel DynamoDB l'id univoco (nome + gara).
- Carichiamo quindi il file in questione nel file S3 corrispondente al token.

3. /uploadxml

- Utilizzando token diversi è possibile selezionare file diversi nel Bucket.

4. /uploadxml (bonus)

- All'interno di questa funzione è stato implementato un controllo del file xml, utilizzando la libreria 'xmllint'. Questa libreria permette di controllare un file xml tramite il file xsd contenente lo standard.
- Controlliamo inoltre che nome e data contenuti nel file xml siano gli stessi registrati all'interno della tabella del DynamoDB.
- È stato necessario aumentare il tempo di time-out e la memoria a disposizione della funzione lambda a causa della complessità del calcolo di controllo.

5. /list_races

- Interroga DynamoDB per ottenere la lista degli eventi registrati, in particolare restituisce un file JSON contenente: nomeGara, dataGara, ID.

6. /list_classes?id=X

- Tramite l'id passato in input estraiamo il file xml dal Bucket che verrà poi convertito in JSON.
- Il file JSON viene scansionato per stampare tutti i nomi delle categorie presenti.

7. /results?id=X&class=Y

- Nello stesso modo del punto precedente ricaviamo la gara con l'id corrispondente.
- Dividiamo i risultati validi (status=OK) e quelli non validi.
- Ordiniamo i risultati validi.
- Restituiamo la combinazione dei risultati validi ordinati e in coda i risultai non validi.
- I risultati sono un array di file JSON, è stato scelto questo formato perché più conforme a Flutter.

8. /downloadxml?id=X

- Tramite l'id passato come parametro troviamo il file che ci interessa.
- Convertiamo il file in base64.
- Completiamo il download con il seguente codice:

```
//Risposta (e download)
const response = {
        statusCode: 200,
        body: bodyRes,
        headers:{
            "Content-Type" : "application/xml",
            "Content-Disposition": "attachment; filename=download.xml"
        },
        isBase64Encoded: true
    };
return response;
```

in questo modo la richiesta GET scaricherà il file invece di mostrarlo come testo.

9. /results?id=X&organisation=Z

- Ricaviamo la gara con l'id corrispondente con lo stesso procedimento del punto 6.
- Controlliamo che la richiesta contenga il parametro 'organisation' e non 'class'.
- Con due cicli forEach controlliamo tutte le persone appartenenti all'organizzazione del dato evento, e le restituiamo