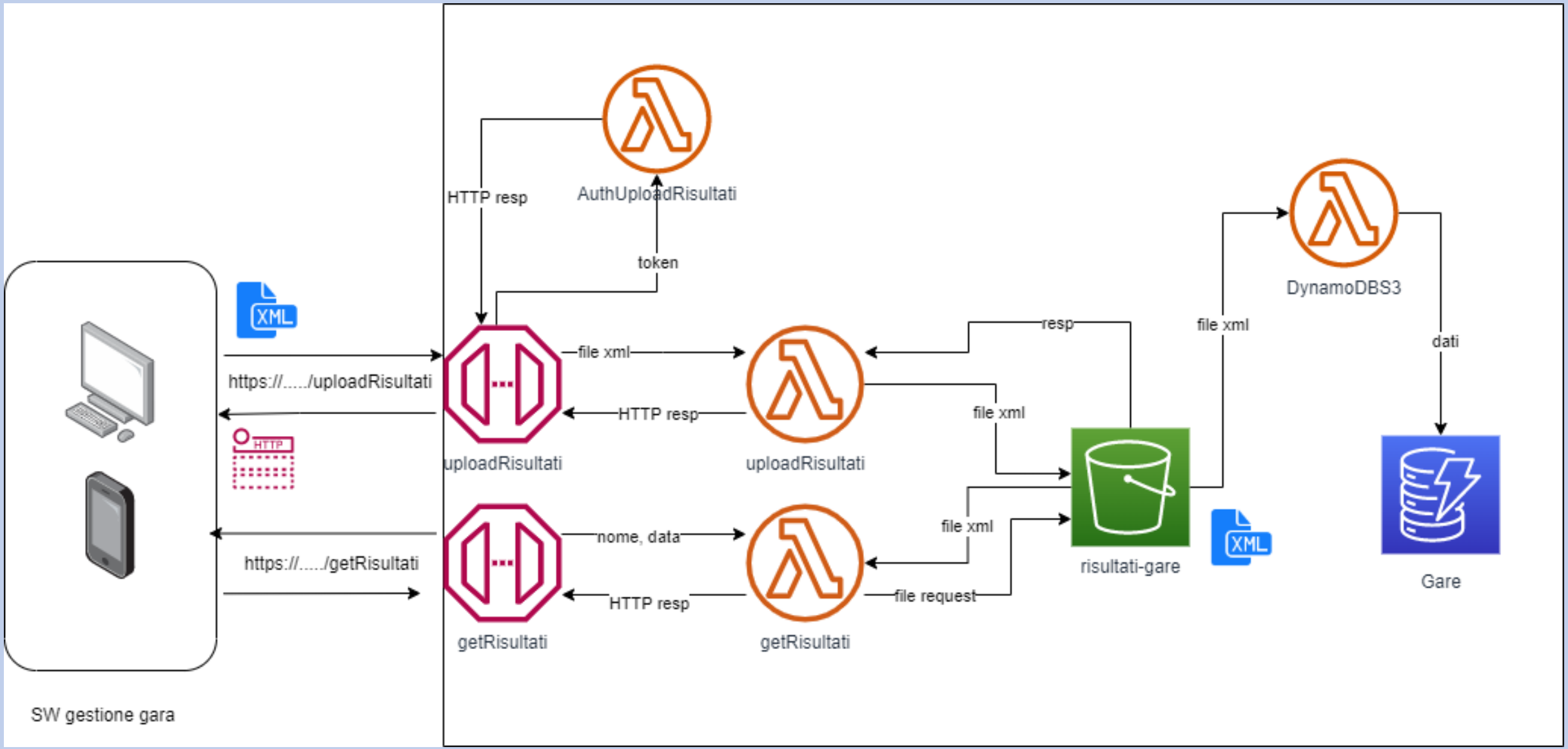
TCM - Homework 2

OSLO
Fagiani Lorenzo – 1068955
Gambirasio Cristiano – 1066866
Villa Valentina – 1067719


AuthUploadRisultati
HTTP resp
token
XML
https://...../uploadRisultati
HTTP
uploadRisultati
file xml
HTTP resp
uploadRisultati
resp
file xml
DynamoDBS3
file xml
dati
risultati-gare
XML
Gare
https://...../getRisultati
getRisultati
nome, data
HTTP resp
getRisultati
file xml
file request
SW gestione gara

1. Completamento dello scheletro

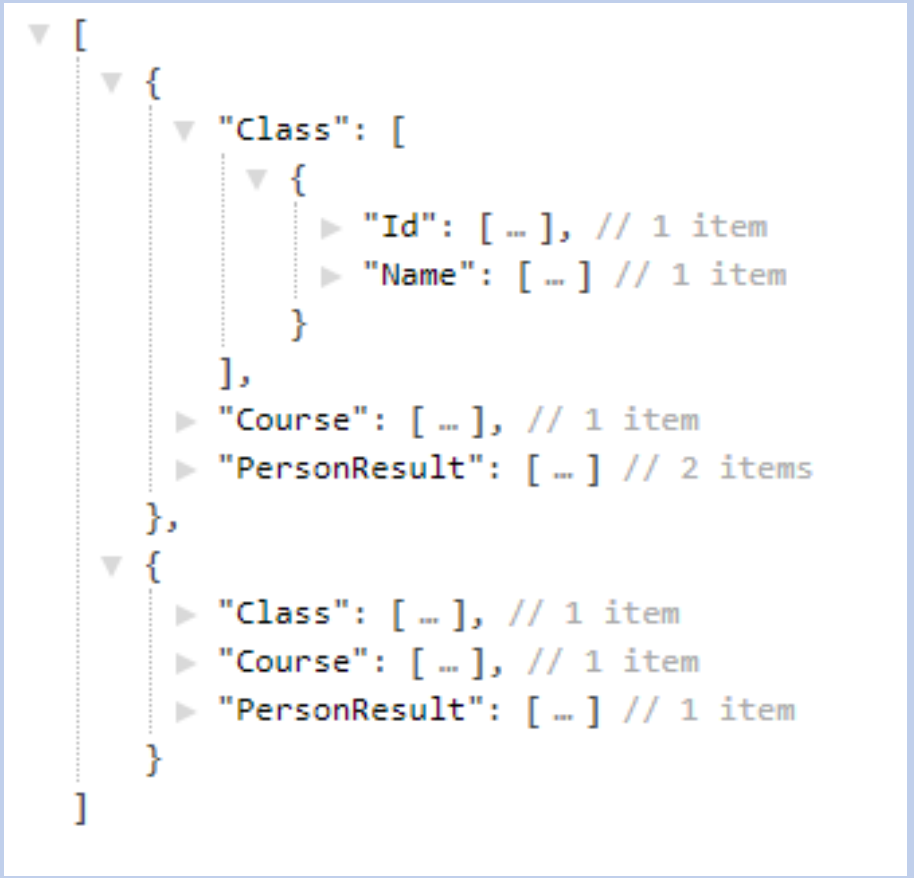
Il completamento dello scheletro è stato ottenuto creando due funzioni lambda, denominate ‘getRisultati’ per rendere disponibili i risultati delle gare (metodo GET), e, ’uploadRisultati’ per il caricamento del file xml nel Bucket S3 (metodo POST).

Entrambe le funzioni sono connesse allo stesso API Gateway, ma con due instradamenti differenti: uno POST e uno GET.

Dall’immagine si può osservare che viene restituito un documento JSON contenente una lista di categorie che a loro volta conterranno i risultati personali.

2. Come gestire più file

Per poter gestire più file è stato deciso di identificare ogni file non solo dal nome dell’evento ma anche dal giorno dello svolgimento della gara. Unendo questi due attributi abbiamo definito la chiave univoca per identificare i file nel Bucket S3.



3. Come evitare che utenti non autorizzati carichino file

Per evitare che utenti non autorizzati carichino file abbiamo implementato una funzione lambda ‘AuthUploadRisultati’, specificando all’interno il token necessario (‘OSLO’) perché venga accettata la richiesta di POST.

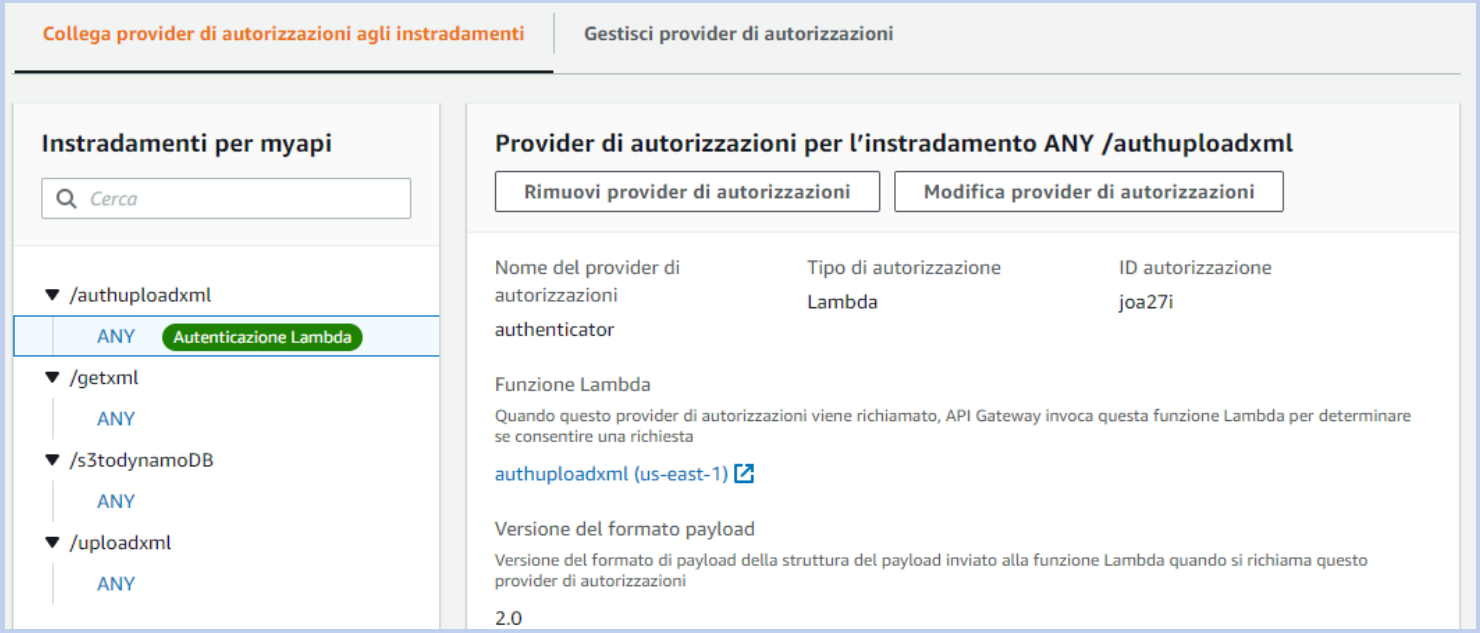
Anche questa funzione è collegata all’API Gateway iniziale ma abilitando un provider di autorizzazioni dalle impostazioni. In particolare abbiamo utilizzato una modalità di risposta: semplice, e non JWT.

index.js

```
exports.handler = async (event) => {
    let response = {
        "isAuthorized": false,
    };

    if (event.headers.authorization == "OSLO") {
        response = {
            "isAuthorized": true,
        };
    }
    return response;
};
```

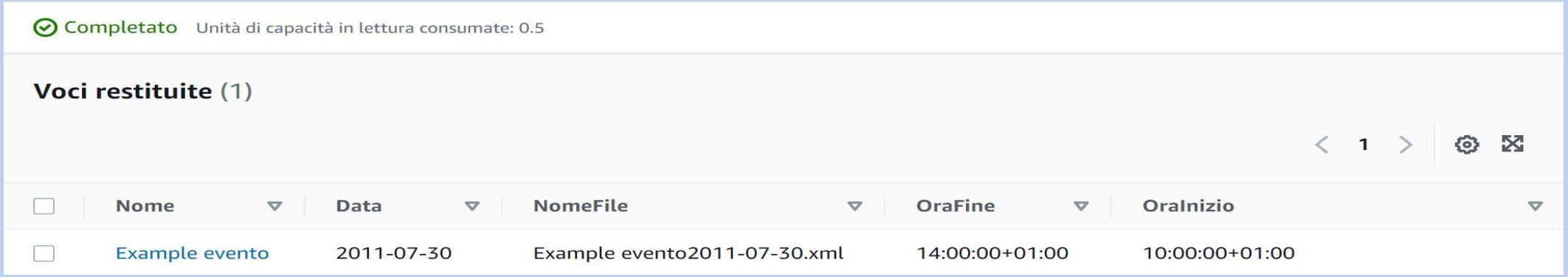
Codice della funzione AuthUploadRisultati



4. Integrazione di AWS DynamoDB

Sfruttando la chiave univoca (nome + data), abbiamo creato una tabella. Tramite la funzione lambda ‘DynamoS3’ e utilizzando come trigger il caricamento di un file sul Bucket S3, salviamo le informazioni necessarie anche nella tabella del Database.

Le informazioni contenute nel Databse potranno essere utilizzate poi nella funzione ‘getRisultati’ per verificare che i parametri inseriti corrispondano a un file esistente.



5. Software locale che simula una gara (vedi allegato)

Il software locale funziona nel seguente modo:

-> Dalla linea di comando:

node index.js ‘nome_file_input’ ‘n_minuti_passati’

-> Il software prende il file in input (da mettere nella stessa cartella del codice) e verrà modificato come di seguito:

- se un partecipante non è arrivato dopo i minuti passati come input, verranno eliminati i campi: FinishTime, Time, TimeBehind, Position, Status, in quanto sono tutti campi che dipendono dall’arrivo. Inoltre vengono considerati solo gli split time precedenti al minuto indicato in input.