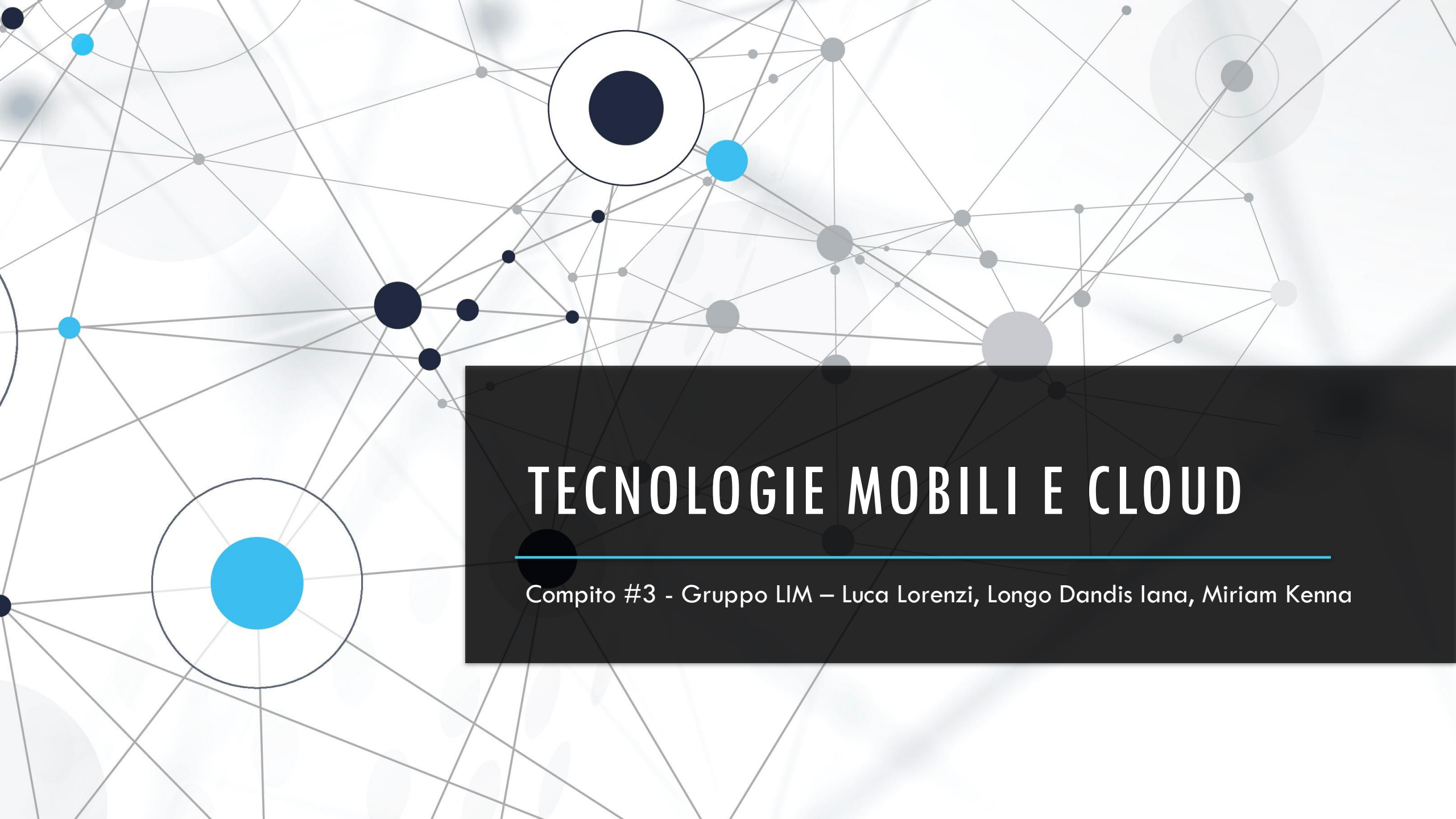

TECNOLOGIE MOBILI E CLOUD

Compito #3 - Gruppo LIM – Luca Lorenzi, Longo Dandis Iana, Miriam Kenna

PRESENTAZIONE HOMEWORK #3

OBIETTIVI

SOLUZIONI

PROBLEMI

OBIETTIVI

OBIETTIVI PER IL HOMEWORK #3

1. Creazione dell'endpoint `/**register_race**` per il passaggio dei dati di interesse e il ritorno del **race_id** e del **token** per il successivo l'upload dei file

2. Creazione dell'endpoint `/**uploadxml**` (POST) con il `**token**`, per il caricamento attraverso body della richiesta del'XML in formato standard

3. Implementazione dell'API e della funzione per poter effettuare la chiamata GET a `/**list_races**` che ritorna una lista JSON degli eventi

4. Implementazione dell'API e della funzione per poter effettuare la chiamata GET a `/**list_classes?id=X**` e ottenere l'elenco delle categorie presenti all'evento X

OBIETTIVI

OBIETTIVI PER IL HOMEWORK #3

5. Implementazione dell'API e della funzione per poter ottenere la classifica attuale della categoria Y dell'evento X con una chiamata GET **a `/results?id=X&class=Y`**

6. Implementazione dell'API e della funzione per poter ottenereil file XML dei risultati della gara di Utente A, con una chiamata GET a **`/downloadxml?id=X`**

SOLUZIONI

OBIETTIVO #1

DESCRIZIONE:

- Creazione dell'endpoint `/**register_race**`

STEP:

- Creazione di una nuova tabella nel database usando il servizio dynamoDB di AWS, per registrare i nuovi parametri passati alla funzione tramite postman dalgli utenti: race_name, race_date ed email.

- Implementazione della funzione **register_race**: che prende i parametri race_name, race_date ed email, poi attraverso delle funzioni random crea un race_id e un token che salva nella tabella insieme ad altri parametri e ritorna come risposta all'utente questi ultimi (race_id ed un token).

- Creazione e collegamento ad un Gateway API per poter passare i parametri con postman: https://8qzr74zo43.execute-api.us-east-1.amazonaws.com/register_race/register_race

SOLUZIONI

OBIETTIVO #2

DESCRIZIONE:

- Caricamento dei risultati di una gara da parte dell'utente A.

STEP:

- Tramite la funzione di upload l'utente A carica i risultati parziali nella gara corrispondente sfruttando il token, il quale è stato registrato e salvato.

- La funzione quindi va a confrontare il token ricevuto con i token salvati nel dynamoDB se trova la corrispondenza allora prende il contenuto xml caricato dall'utente e lo salva nel bucket S3

- L'utente A aggiorna i risultati grazie all'utilizzo del token legato alla sua utenza.

SOLUZIONI

OBIETTIVO #2

DESCRIZIONE:

- Registrazione dei risultati di una gara da parte di un secondo utente B.

STEP:

- Utilizzando sempre lo stesso l'endpoint **register_race** l'utente B fornisce in input i parametri race_name, race_date ed email, i quali tramite la funzione **register_race** vengono registrati all'interno della tabella che contiene i dati della gara e come risposta ottiene un race_id ed un token, i quali lo contraddistingueranno dall'utente A

SOLUZIONI

OBIETTIVO #3

DESCRIZIONE:

Implementazione dell'API e della funzione per poter effettuare la chiamata GET a /**list_races`** che ritorna una lista JSON degli eventi

STEP:

- Creazione del gateway API che si collega alla funzione **«punto 5»** https://s96his1myh.execute-api.us-east-1.amazonaws.com/list_races/list_races

- Implementazione della funzione: collegamento alla tabella che contiene le informazioni e la lista degli eventi, poi con la funzione **listItems** possiamo accedere ai dati all'interno della tabella poi con il metodo json.stringify convertiamo l'oggetto javascript in una stringa json, dopo di che con JSON.parse() possiamo scorrere gli elementi della stringa e salvare tutti I valori in un array che poi restituiamo all'utente come risposta nel body

SOLUZIONI

OBIETIVO #4

DESCRIZIONE:

Implementazione dell'API e della funzione per poter effettuare la chiamata GET a `/**list_classes?id=X**`e ottenere l'elenco delle categorie presenti all'evento X
https://v18yt6ktlc.execute-api.us-east-1.amazonaws.com/list_classes/list_classes

STEP:

- Creazione e configurazione del gateway API che si collega alla funzione «**punto 6**» in modo che possa ricevere id del evento come parametro passato nell'url dall'utente
- Come per il punto precedente implementiamo la funzione in modo che si colleghi alla tabella che contiene le informazioni e la lista degli eventi scorre gli eventi e ritorna le classi degli eventi di interesse, in particolare confronta l'id ricevuto tramite GET e l'id salvato nella tabella del dynamoDB
- Una volta salvate tutte le classi in una variabile prima di restituirle all'utente gli filtriamo con il metodo **filter** in modo da non restituire ripetizioni all'utente

SOLUZIONI

OBIETIVO #5

DESCRIZIONE:

Implementazione dell'API e della funzione per poter ottenere la classifica attuale della categoria Y dell'evento X con una chiamata GET a `/**results?id=X&class=Y**`

STEP:

- Creazione e configurazione del gateway API che si collega alla funzione «**punto 7**» in modo che possa ricevere id e la classe del evento come parametri passati nell'url dall'utente https://vplc5ekese.execute-api.us-east-1.amazonaws.com/result/result
- Come per il punto precedente implementiamo la funzione in modo che si colleghi alla tabella che contiene le informazioni e la lista degli eventi scorre gli eventi con un ciclo e ritorna gli eventi di interesse, in particolare confronta l'id e la classe ricevuti tramite GET e con quelli salvati nella tabella in dynamoDB
- Per mancanza di tempo non abbiamo ancora implementato il metodo compare per far risultare la clssifica ordinata in modo decrescente but work in progress
- Una volta salvate tutti gli eventi con le relative posizioni in una variabile la restituiamo nel body in modo da far vedere all'utente i risultati della classifica

SOLUZIONI

OBIETIVO #6

DESCRIZIONE:

Implementazione dell'API e della funzione per poter ottenereil file XML dei risultati della gara di Utente A, con una chiamata GET a `/**downloadxml?id=X**`

STEP:

- Implementazione della funzione che riceve tramite GET nell'url l'id dell'utente si collega al dynamoDB trova l'id dell'utente e l'id di tutte le gare che ha caricato

- Una volta ottenuto l'id delle gare si collega alla tabella che contiene le informazioni relative alle gare e le salva in formato json che puoi viene convertito in formato XML e inviato all'utente

PROBLEMI

non ancora completamente implementati

DESCRIZIONE:

- Abbiamo incontrato alcune difficolta per l'implementazione del punto 2 e 8 del homework3 assegnato:

- In particolare per il punto 2 non siamo ancora riusciti ad implementare le funzionalità del sistema che andrebbero ad aggiornare il contenuto di un file caricato in precedenza dall'utente A

- Mentre, per il punto 8 non siamo riusciti ad implementare la funzionalità che dopo aver caricato dinamicamente il contenuto del file xml lo faccia scaricare direttamente al utente