Djungle Contest API v2.4

Introduzione

Creare un progetto django per implementare l'estrazione pseudo casuale in un concorso a premi.

Il sistema deve prevedere che possano essere attivi più concorsi contemporaneamente. [S1]

Ogni concorso:

- ha un premio che può essere vinto più volte al giorno (*perday*) anche dallo stesso giocatore [S2]
- ha un intervallo di date di validità, gli estremi sono compresi [S3]

Ad esempio i concorsi potrebbero presentarsi in questa forma:

```
contest = {
  "code": "C0001"
  "name": "Vinci uno sconto",
  "start": "2020-02-01",
  "end": "2020-02-29",
  "prize": {
    "code": "five-percent-discount",
    "perday": 45,
    "name": "Sconto del 5%"
  }
}
```

Il concorso C0001 è valido per tutto il mese di febbraio 2020 ed ha come premio lo sconto del 5% che può essere vinto al massimo 45 volte al giorno.

API

Ogni chiamata API e' una giocata ed il sistema deve rispondere se si ha vinto o no.

Request

```
GET /play/?contest={code}
```

Esempio di una chiamata:

```
GET /play/?contest=C0001
```

Responses

HTTP 400 - concorso mancante [R1]

```
{
  "error": {
    "status": "400",
    "title":  "Contest code missing",
    "detail": "Specify contest code parameter."
  }
}
```

HTTP 404 - concorso non trovato [R2]

```
{
  "error": {
    "status": "404",
    "title":  "Contest not found",
    "detail": "Contest code {code} not found."
  }
}
```

HTTP 422 - il concorso non e' attivo (fuori dal range di date) [R3]

```
{
  "error": {
    "status": "422",
    "title":  "Contest in not active",
    "detail": "The contest with code {code} is not active."
  }
}
```

HTTP 200 - risposta in caso di vincita [R4]

```
{
  "data": {
    "winner": true,
    "prize": {
      "code": "five-percent-discount",
      "name": "Sconto del 5%"
    }
  }
}
```

HTTP 200 - risposta in caso di non vincita [R5]

```
{
  "data": {
    "winner": false,
    "prize": null
  }
}
```

Requisiti

Il sistema deve essere implementato in modo che:

- per ogni concorso tutti i premi siano vinti ogni giorno [S4]
- le vittorie devono essere distribuite nel modo più uniforme possibile nell'arco delle 24 ore [S5]

Si può ipotizzare che ogni giorno arrivino richieste di gioco in modo uniforme durante tutte le 24 ore in numero di due ordini di grandezza superiori rispetto al numero di premi in palio per quel giorno. [S6]

Inoltre:

- utilizzare un db relazionale per la permanenza dei dati [S7]
- è richiesto usare docker-compose per avviare eventuali servizi necessari al corretto funzionamento del progetto in locale [S8]
- sono richiesti Unit/Integration test [S9]
- il sorgente va rilasciato su repo git privato a scelta corredato di un minimo di documentazione per spiegare le scelte progettuali e per provarlo in locale [S10]

Bonus 1 [facoltativo]

Implementare un sistema di autenticazione delle API. Un utente autenticato e' un cliente di queste API e può effettuare chiamate solo sui concorsi per cui e' abilitato. [S11]

Definire il meccanismo di autenticazione e la configurazione cliente - concorso. [S12]

La API non cambia, aggiungere le response:

401 Unauthorized (richiesta non autenticata) [R6]
403 forbidden (non puoi giocare a questo concorso) [R7]

Bonus 2 [facoltativo]

Implementare il sistema in modo che un utente possa vincere solo WMAX volte al giorno (WMAX > 0) per ogni concorso. [S13]

La richiesta API diventa:

GET /play/?contest={code}&user={user_id}

aggiungere la response:

420 Enhance your calm (utente ha raggiunto il limite di premi vinti) [R8]