A

INGEGNERIA DEL SOFTWARE II - ESAME DEL 26 GENNAIO 2018 - PARTE PRATICA *A*

NOME, COGNOME, STUDENT ID: _____

LEGGETE FINO IN FONDO PRIMA DI COMINCIARE

Fase 0: Preparazione.

1. Create un repository su github (vi consigliamo di crearlo **vuoto**, senza aggiungere il file gitignore o il readme suggeriti da github, ma come preferite)
2. Create applicazione vuota su heroku.com
3. Compilate il file di consegna, a **https://tinyurl.com/tse2-20180126**

Procedete con il resto solo DOPO avere compilato la fase di preparazione

Fase 1:

1. Clonate il repo a: https://github.com/trentose2/20182601.git Il repo contiene due branches, master e checker

Ricordate che quando fate clone, vi trovate esplicitamente creata solo la branch di default (di solito master). Per avere anche la branch checker, digitate `Git branch checker origin/checker`

Il repo contiene:

- **Su branch master:** File app.js, un node server. Per ora il server ha solo una API (get), */count*, che ritorna un oggetto fisso (in questa branch, ritorna {**count**: 3})
- **Su branch checker:** File checker.js, che per ora contiene solo una funzione che confronta se un oggetto *actual* corrisponde ai requisiti dell'oggetto *expected*, ovvero, se per ogni coppia <attributo,valore> di expected c'e' la stessa coppia <attributo,valore> in actual. Il file contiene anche una funzione "vuota", *check*, che dovrete implementare

2. Create i "git remote" per fare push sul vostro repository e provate a fare push delle due branches sul vostro repo. Vi consigliamo anche di provare a fare deploy su heroku della vostra master branch, cosi' verificate subito se andando a [herokuapp]/count vi ritorna {**count**: 3}. Per questo dovrete aggiungere il procfile corretto, oltre a express (e node-fetch, se lo usate) al package.json

Fase 2: ** Lavorando sulla branch checker** , implementate la funzione *check*, che riceve in input i parametri:

- url (a string)
- invocationParameters (an object),
- expectedResultData (an object)
- expectedResultStatus (an integer)

E fa questo: fa una chiamata **http GET** allo url specificato, passando i parametri in *invocationParameters* nella query string.

Per esempio, se la chiamata e' `check("https://example.com", {lato1: 3, lato2: 5}, {area: 15}, 200)`

A

La vostra funzione fara' una chiamata a `https://example.com?lato1=3&lato2=5`

Poi, riceve la risposta (in **json**) e verifica che
i) la risposta contenga tutti gli attributi che sono nell'expected results, e che i valori siano gli stessi (la risposta puo' anche contenere altri attributi). Per questo potete usare la funzione *compareResults* fornita
ii) lo status della risposta (http status code) sia quello richiesto da expectedResultStatus

Con la risposta riempiamo questo oggetto:

```
{ // this is the object you need to set
  urlChecked: url,
  resultData: null, // qui mettete l'oggetto che vi ritorna la get
  resultStatus: null,// qui mettete lo status della risposta che ottenete dalla get
  statusTestPassed: null, // qui mettete true se lo status e' quello atteso, false altrimenti
  resultDataAsExpected: null // qui mettete true se l'oggetto ritornato e' quello atteso, false altrimenti
}
```

Suggerimento: per ottenere dinamicamente le properties di un oggetto in forma di array usate Object.keys(*invocationParameters*), come peraltro e' mostrato nella funzione che gia' trovate
Ad esempio, Object.keys(`{lato1: 3, lato2: 5}`) vi ritorna ["lato1","lato2"]
Una volta implementata, fate commit e push **della branch checker** sul vostro github repo creato sopra.

Fase 3: vogliamo esporre la nostra funzione *check* su heroku, in modo che sia raggiungibile con una chiamata allo URI `[vostra app]/check`, da invocare con POST che accetta json e restituisce json.
Il json accettato è del tipo:

```
{
  "url": "https://localhost:5000/count",
  "invocationParameters": {lato1:5, lato2:7},
  "expectedResultData": {"count": 3},
  "expectedResultStatus": 200
}
```

(non state li' a gestire i casi non validi, basta che processiate correttamente una chiamata in questo formato)

Per questo, sempre lavorando sulla branch *checker*, aggiungete quindi il metodo che riceve la chiamata http, invoca la funzione check, e ritorna al chiamante il risultato della funzione check, in notazione json. *Hint: occhio che probabilmente implementerete la funzione check come funzione asincrona…. Dovete quindi aspettare il risultato prima di rispondere al chiamante*

Fase 4: a questo punto vogliamo mettere il tutto su master e pubblicare su heroku. Siccome vogliamo tenere una storia lineare e semplice, raggruppiamo **tutti** i commit della branch *checker* in **un solo** commit, e facciamo poi merge su master - che a questo punto sara' una fast forward merge che sposta master avanti di UN commit.
Facciamo push di master sul nostro repo github, e pubblichiamo il tutto su heroku (non serve fare push di checker)

[A questo punto per bug fixing eventuale lavorate pure su master]