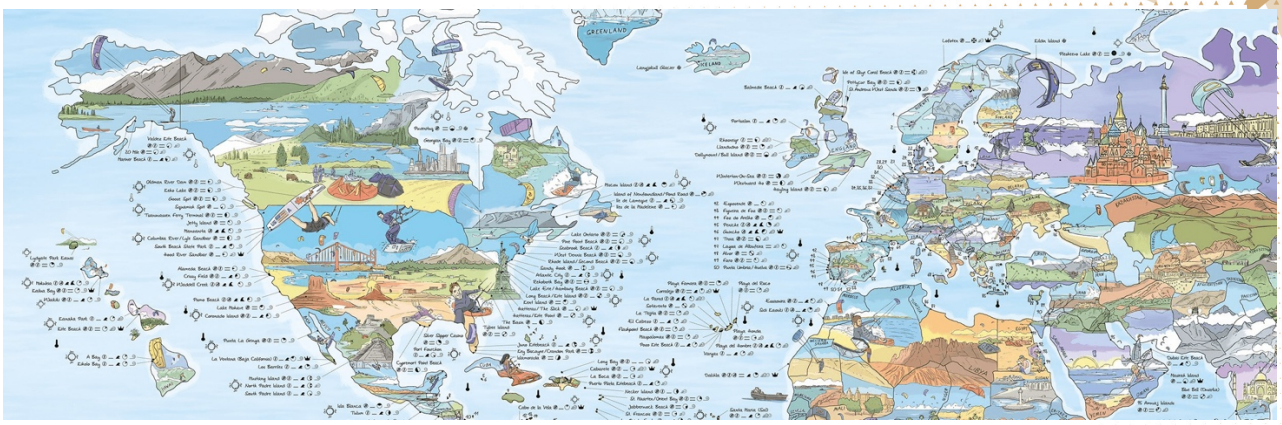
Problema Backend Engineer

Descriere

Pe langa dezvoltare de produse software, echipa noastra are multe alte pasiuni, mai ales legate de sport. Una dintre pasiunile noastre este kitesurfing-ul. Practicantii sunt trasi pe apa de un zmeu, avand in picioare o placa. Viteza cu care te dai este de 30-50km/h, iar partea cea mai tare este ca poti zbura deasupra apei la inaltimi de cativa metri pentru cateva secunde.

Am adunat de-a lungul timpului o baza de date cu o multitudine de destinatii (spot-uri) de practicat kitesurfing din toata lumea. Pentru a veni in ajutorul altor practicanti de kitesurfing, dorim sa dezvoltam un API prin intermediul caruia sa facem aceste informatii disponibile altor dezvoltatori de aplicatii mobile sau web.



Avem nevoie de un **super inginer, ca tine,** care sa ne ajute sa dezvoltam acest API.

Detalii de implementare

Ne dorim sa rezolvi aceasta problema in folosind Spring Framework. Poti sa faci un stub de proiect foarte repede folosind generatorul de aici: https://start.spring.io.

In arhiva care vine impreuna cu enuntul problemei, vei gasi baza de date in format csv: internship-backend-data.csv cu cele mai faine spoturi de kite din lume.

Dorim sa sa structurezi aceaste informatii intr-o baza de date (SQL, noSQL, sau ce stii mai bine). Structura bazei de date o lasam la latitudinea ta. Daca nu iti ia foarte mult timp, poti sa te gandesti sa faci si importul initial al datelor. Daca ai mai lucrat cu Hibernate/JPA ti-am sugera sa folosesti aceste tehnologii la lucrul cu baza de date.

Dorim sa construim un API (detaliat mai jos) care va avea 2 functionalitati:

1. Va oferi informatii despre spoturile de kitesurfing
2. Va permite sa adaugi/scoti anumite spots in/din o lista de favorite (poti sa specifici utilizatorul caruia ii salvezi spotul ca favorit ca parametru la API daca nu implementezi partea de autentificare)

API-ul va fi folosit de alti dezvoltatori pentru a crea aplicatii mobile si web.

Nice to have:

1. filtrare pe call-ul de api/spots dupa country si wind probability
2. paginare pentru call-ul de api/spots
3. User management (sign up, sign in, sign out) + autentificare/securizare API -- nu e foarte usor de facut, dar iti garantam ca vei face ++ la skill daca implementezi si acest nice to have

Vrem sa scrii cod de care esti foarte mandra/mandru si nu crezi ca mai poate fi imbunatatit :)

Te rugam sa lucrezi sub un repository private de git (bitbucket, gitlab, github), sa faci commit-uri granulare si dese. Cand ai terminat, vom trimite emailul unuia dintre colegii nostri care va face review la rezolvarea ta.

Ce vom urmari:

1. Claritatea codului: cat de ordonat este scris, cat de usor e de inteles
2. Cum aplici principiile OOP
3. Cum tratezi erorile

Structura API recomandata

Pe baza alegerilor tale poate sa difere (eg. daca nu folosesti token si faci autentificare cu user/pass pentru endpointuri).

Endpoint-uri:

- **/api/signup**
- **/api/login** - raspunsul contine informatii despre userul logat(in caz de succes), inclusiv token-ul de autentificare
- **/api/users/me/forgot-password**
- **/api/users/me -** raspuns asemanator cu cel de la **/api/login**
- **/api/spots -** Listare spot-uri de kiting - informatii generale. Endpoint-ul prezinta si optiuni de filtrare ale datelor.
    - optional parameters:
        - country
        - windProbability (constrangere ca valoarea sa fie in intervalul [0...100]
- **/api/spots/{spotId} -** Listare detalii despre spot-ul de kiting ales (detalii: latitudine, longitudine, when to go).
- **/api/spots/countries -** Listare tari disponibile in lista de spot-uri de kiting.
- **/api/favorites/spots -** Adauga un spot de kiting la lista de favorite.
- **/api/favorites/spots/{spotId} -** Sterge un spot de kiting din lista de favorite.

In caz de eroare, backend-ul poate raspunde cu urmatorul format (exemplu de la **/api/spots**):

```
{
  "error": {
    "message": "Wind probability filter provided is not a number. Please consult the documentation file and try again.",
    "code": "windProbabilityIsNaN"
  }
}
```