Tiedosto sisältää jokaisen backendin endpointin ja kaikki mahdolliset palautukset.

## Login:

Ottaa vastaan name,password parametrit

Jos tietokannasta löytyy peli jossa name ja password ovat samat, palauttaa seuraavan:

{”status”: peli tiedot dictionary muodossa), 200

JOS peliä EI löydy:

{”error”: No game found with given name and password”}, 400

JOS name tai password parametrit ovat väärin täytetty (min 4 kirjainta molemmat)

{”error”: ” password or name is invalid”}, 400

## Register:

Ottaa vastaan nimen, ja salasanan

Luo pelin ja palauttaa pelin idn takaisin muodossa:  
{game\_id: 1}, 200

Missä game\_id on dictionary ja 200 on html koodi (200 = kaikki kunnossa ja 400 virhe)

Esimerkki käyttö:

127.0.0.1:5000/api/register?name=Jani&password=Pirkka

JOS name tai password parametrit ovat väärin täytetty (min 4 kirjainta molemmat)

{”error”: ” password or name is invalid”}, 400

## Save\_game\_details:

Ottaa vastaan game\_id, ja kaksi tuplea ja tallentaa tiedot tietokantaan.

Esimerkki käyttö:

127.0.0.1:5000/api/save\_game\_details?game\_id=1

&to\_update=money,fuel,fuel\_used

&information=500, 500,1500

Palauttaa

{“error”: “The lengths of 'to\_update' and 'information' must match.”}, 400

TAI

{“status”: “Success”}, 200

Vähän selitystä tälle tuplejen käytölle.

Endpointille annetaan siis 2 tuplea.

Ensimmäinen eli to\_update sisältää informaatio kentät mitä päivitetään tietokannassa esim fuel, money

Ja informaatio sisältää samassa järjestyksessä, informaatiota joka halutaan päivittää kyseiselle kentälle.

Eli, jos to\_update = money, fuel

Niin information pitäisi olla 500, 750,34

Tyylillä.

## Open\_lootbox:

Ottaa vastaan 2 parametria:

Game\_id numero (int)

Open\_type string (money, fuel)

Esimerkki käyttö:

127.0.0.1:5000/api/open\_lootbox?game\_id=1&open\_type=money

Avaa lootboxin joko rahalla tai bensalla perustuen open\_type parametriin.

Palauttaa seuraavan:

{”error”: ”mitä meni pieleen” }, 400

TAI

{”status”: ”palaute tekstiä perustuen että mitä tuli lootboxista.” }, 200

## Refuel:

Ottaa vastaan 2 parametria:

game\_id int

amount int

amount on litra määrä bensaa mitä halutaan ostaa.

Esimerkki käyttö:

127.0.0.1:5000/api/refuel?game\_id=1&amount=500

Jos game\_id tai amount on väärin täytetty palauttaa:

{"error": "Missing 'game\_id' or 'amount'"}, 400

Onnistuneessa kutsussa palauttaa:

{"status": {“fuel”: fuel\_määrä, “money”: rahamäärä } } 200

JOS kutsu EI onnistu:

{“error”: “Not enough money to buy that much fuel.”}, 400

## Get\_airport\_information:

Ottaa vastaan kentän icao\_koodin ja peli idn

Esimerkki:

127.0.0.1:5000/api/get\_airport\_information?icao=EFHK&game\_id=1

Jos icao koodi on väärin, palauttaa:

{"error": "Invalid ICAO-code"}, 400

Jos kentän informaatiota ei saatu kyseiselle pelille, palautetaan:

{"error": f"Unable to get information for said airport for game {game\_id}"}, 400

Jos kutsu onnistui:

{“status”: kentän informaatio dictionary muodossa}, 200