T1.1 Ykköstehtävä: Rajapinnat – yleistä

Mitä rajapinnat ovat?

Rajapinta tarkoittaa kahden laitteen tai ohjelman välistä kommunikointia. Rajapinta määrittelee säännöt ja ohjeet joiden mukaan viestintä tapahtuu, mutta ei itse toteuta viestien sisältöä tai toimintaa. Rajapinta mahdollistaa siis yhteyden ja tiedonvaihdon, lisäksi rajapinta ei saa tietoon, mitä viesteissä on.

Miksi niitä tarvitaan?

Rajapinta mahdollistaa sen, että eri laitteet ja ohjelmat voivat kommunikoida toistensa kanssa ilman, että niiden tarvitsee tietää toistensa tarkkaa toteutusta. Esimerkiksi ohjelmoinnissa rajapinta määrittelee selkeästi, mitä eri ohjelma osien tulisi toteuttaa. Kuten eläin rajapinta määrittelee eläimen ominaisuuksia, kuten äänteleekö, silloin kaikkien eläinten kuuluu äänellä omalla tavallaan. Rajapinnat myös tärkeänä osana yhteistyötä ohjelmointitiimeissä, koska ne määrittelevät selkeästi, mitä jonkin osan pitää tehdä, jolloin eri ihmiset voivat työskennellä projektin kanssa samanaikaisesti.

Mihin niitä käytetään?

Rajapintoja käytetään helpottamaan eri laitteiden ja ohjelmistojen välistä kommunikointia. Esimerkiksi ohjelmoinnissa rajapinnat selkeyttävät koodia ja tekevät siitä helpommin ymmärrettävää ja ylläpidettävää. Rajapintojen avulla voidaan myös ”yhdistää” eri ohjelmistoja toisiinsa. Esimerkiksi, jos sovellus tarvitsee säätietoja, se voi hakea ne toisen palvelun tarjoaman rajapinnan kautta ilman, että sen tarvitsee tietää tarkasti, miten tuo palvelu toimii tarkalleen koodista asti.

Mitä koet oppineesi tässä vaiheessa?

Koen oppineeni jo todella paljon siitä, mitä rajapinnat tarkoittavat, miten ne toimivat ja miten niitä pystyisi hyödyntämään. Yritän jatkossa aina omissa ohjelmointi projekteissa luoda rajapintoja ja näin tehdä koodista selkeää ja helposti ymmärrettävää.

B) Miksi valitsin Routing API:n?

Valitsin Digitransitilta Routing API:n, koska se on erittäin kiinnostava rajapinta, jota voidaan käyttää julkisen liikenteen reittien ja aikataulujen hallintaan. Julkinen liikenne on tärkeä osa monen kaupungin toimimista ja Digitransit tarjoaa avoimen rajapinnan, joka tekee siitä helposti käytettävän ja laajennettavan.

Mihin sitä käytetään?

Digitransit Routing API:tä käytetään reittien suunnitteluun ja julkisen liikenteen tietojen hakemiseen. Se mahdollistaa reittien suunnittelun ja tarkistamisen julkisessa liikenteessä. Antaa mahdollisuuden ennustaa matkojen kestoa tietyillä reiteillä ja aikaväleillä.

Kuinka laaja rajapinta on, mitä kaikkea sillä voi tehdä?

Routing API tarjoaa aikatauluja ja mahdollistaa matka-aikojen laskemisen, mikä on hyödyllistä matkustajille, jotka haluavat tietää milloin kulkuväline on saapumassa.

Voisiko tämä olla valitsemasi harjoitustyön aihe?

Kyllä, tämä voisi ehkä olla valitsemani harjoitustyön aihe. Minusta tämä on todella mielenkiintoinen rajapinta. Ainut ongelma tulee siinä, että voi vaatia liian suurta osaamista näin aloittelijana ja käyttötarkoitusta ohjelmalle, jotta voisin käyttää sitä itse ei ole. Itse suosisin jotain sääapplikaatiota, jolloin oppisit rajapinnan hyödyntäminen ennen syvään päätyyn menemistä.

Ota selvää onko olemassa suomalaisia tietolähteitä, joihin on määritelty rajapinta? (Mahdollisia aiheita: Kartta/Aikataulut/Sää/Tilastot/...)

Kyllä on, esimerkiksi Jyväskylän vihreä liikenne käyttää digitransit rajapintaa. Digitransit on HSL:n, Fintrafficin ja Waltti Solutions Oy:n tarjoama palvelualusta. Digitransit tarjoaa erillaisia Rajapintoja, kuten Routing Api, Routing Data Api, Geocoding Api, Real-Time Api – Vehicle Positions.

Monet linja-auto yritykset käyttävät näitä rajapintoja hyödyksi reittioppaissa ja aikataulusovelluksissa.

T1.2 Käsitetehtävä (Määritelmiä/Apua/Ohjeita/Tietoa)

Mikä on API?

API eli Application Programming Interface on rajapinta, joka auttaa erilaisia ohjelmistoja muodostamaan yhteyden ja viestimään keskenään. API ottaa käskyn ja kertoo järjestelmälle, mitä haluat tehdä ja järjestelmä palauttaa tiedon sinulle.

Mikä on API URL?

API URL on osoite, jonka avulla voit käyttää Apia ja sen eri ominaisuuksia. Perus url osoite on ikään kuin käyttämäsi tietyn apin perusosoite. Perus url ei kuitenkaan tee paljoa ilman endpointia eli päätepistettä.

Esimerkki perus-url:sta (https://www.jypliiga.fi/)

Mitä ovat parameters?

Paremetrit ovat ”lisätietoja”, jotka voidaan liittää mukaan api pyyntöön, jotta palvelin osaa antaa tarkemmin rajattua tai suodatettua tietoa. Ne tulee yleensä osaksi url-osoitetta ja tämä vaikuttaa siihen, että millainen vastaus käyttäjälle annetaan. Parametrit kirjoitetaan aina url osoitteen loppuun ? merkin jälkeen.

Käytännössä muoto on: avain=arvo

Esimerkiksi: www.sivustosi.fi/pelaajat/?myparam1=123&myparam2=abc&myparam2=xyz

Avaimet: myparam1 ja myparam2

Arvot: 123, abc, xyz

Mikä on endpoint?

Endpoint eli päätepiste tarkoittaa API-pyynnön sisällä olevaa päätepistettä, joka on tarkkapaikka sivulla, mikä näytetään.

Esimerkiksi perus url-osoitteessa: https://www.jypliiga.fi/ottelut Päätepiste on /ottelut.

Mikä on API key/token?

API-avain on ainutlaatuinen tunniste, jonka avulla voidaan vahvistaa käyttäjän henkilöllisyys. Sen avulla varmistetaan, että juuri sinulla on oikeus käyttää kyseistä apia. Kun

API tunnistaa sinut lailliseksi käyttäjäksi avaimen perusteella, se sallii pääsyn tietoihin ja palveluihin. Ilman voimassa olevaa api-avainta pääsy apiin ei ole mahdollista.

Mitkä ovat headers?

Api-headers on lisätietoja, jotka lähetetään api-pyynnön mukana, mutta eivät näy url-osoitteessa. Ne auttavat apia ymmärtämään, kuka pyynnön lähettää, millaista tietoa halutaan, ja miten se käsitellään. Headerit kulkevat "taustalla" pyynnön mukana ja auttavat apia ymmärtämään, miten vastata pyyntöön oikein.

Mikä on GET request?

Get-pyyntö on tapa hakea tietoa palvelimelta internetin kautta. Kun lähetät GET-pyynnön apille, kerrot sille, että haluat saada tiettyä dataa. Palvelin käsittelee pyyntösi ja palauttaa pyydetyt tiedot selkeässä ja järjestellyssä muodossa. Get-pyynnöt eivät muuta palvelimen tietoja – ne ainoastaan hakevat olemassa olevaa dataa.

Mikä on POST request?

Get pyyntö oli tiedot hakemista ilman palvelimen tietojen lisäämistä niin Post-pyyntö on taas nimensä mukaisesti data lähettämistä ja tallentamista internetin avulla. Esimerkiksi uusi käyttäjä rekisteröityy, tämä tieto lähetetään post-pyyntöä käyttäen. Lisäksi post pyynnössä ei näy tiedot url osoitteessa, jolloin voidaan käsitellä salaista tietoa.

T1.3 JSON-tehtävä

JSON eli **J**ava**S**cript **O**bject **N**otation on tapa tallentaa ja siirtää dataa sellaisessa muodossa, joka on helppo ihmisen lukea ja tietokoneiden helppo ymmärtää, joka helpottaa todella paljon rajapinnan kehittämistä ja virheiden selvittämistä. JSON esittää tiedostossa tiedot avain ja arvo pareina. JSON on pelkkää teksti, joka on kirjoitettu JavaScript-objektimerkinnöillä.

Esim: { “Name”: “Matti”, “Ikä”: 30, “Harrastukset”: [“Juokseminen”, “Uiminen”] }

Esimerkissä nähdään hyvin, kuinka dataa tallennetaan. JSON-tiedoston tiedot formatoidaan (“avain”: “arvo”).

Avaimia tuossa äskeisessä esimerkissä oli: (Nimi, Ikä, Harrastukset).

Arvoja taas oli (Matti, 30, Juokseminen ja uiminen).

JSON-tiedoston toiminta toimii käytännössä samalla tavalla kuin, mikä tahansa ohjelmointikieli. Tiedon alussa on aaltosulku ja lopussa, jolloin se päättää objektin. JSON-tiedostossa jokaista aaltosulkeiden sisällä olevaa dataa kutsutaan “Objekteiksi”. Sen lisäksi huomataan, kuinka tallensin Harrastukset avaimeen 2 arvoa käyttäen taulukkoa. Taulukkoa voi käyttää JSON-tiedostossa laittamalla tiedot hakasulkeiden sisälle ja erottamalla pilkulla.

JSON on todella suosittu tapa tallentaa tietoa. Syy tähän on se, että JSON on todella helppolukuista aloittelijoillekin, tämän lisäksi sitä on nopea kirjoittaa ja lukea. Se sopii monen eri ohjelmointi kielen kanssa ja laaja tuki erilaisille API-rajapinnoille. JSON on sen lisäksi todella kevyt tiedostomuoto verrattuna muihin formaatteihin, jolloin tiedonsiirrosta tulee tehokkaampaa ja nopeampaa.

JSON toimii rajapintojen tiedonsiirtomuotona, ja se mahdollistaa helpon ja tehokkaan tiedonsiirron palvelimien ja ohjelmiston välillä. Rajapintojen avulla sovellukset voivat pyytää tietoja (esimerkiksi tietokannasta) ja saada ne takaisin JSON-muodossa. Rajapinnoissa JSON tiedostolla voidaan lähettää dataa palvelimelle (esimerkiksi käyttäjätiedot). Rajapinnoissa tiedon vastaanottamisessa yleensä käytetään JSON-muotoista dataa. Data voisi tässä tilanteessa olla hakutuloksia tai käyttäjän profiilitietoja.

JSON tukee erinomaisesti C# ohjelmointi kieltä. JSON-tiedoston lukemisessa ja kirjoittamisessa C# tarvitaan, kuitenkin erilaisia kirjastoja (esimerkiksi System.Text.JSON ja Newtonsoft.JSON (tunnetaan myös nimellä JSON.NET)). JSON-tiedoston käsittely voidaan jakaa kahteen päätoimintoon:

- **Serialisointi** eli C# objektin muuntaminen JSON-muotoon.
- **Deserialisointi**: JSON-datan muuntaminen C#-objektiksi.

Serialisointi koodiesimerkki C#-ohjelmointikielessä:

```
using System.Text.JSON;
        string JSON = JSONSerializer.Serialize(objekti);
        Console.WriteLine(JSON);
```

Deserialisointi koodiesimerkki C#-ohjelmointikielessä:

```
using System.Text.JSON;
string JSON = filename.JSON;
        Henkilö henkilö = JSONSerializer.Deserialize<Henkilö>(JSON);
        Console.WriteLine($"Nimi: {henkilö.Nimi}, Ikä: {henkilö.Ikä}");
```