PACKAGE RACE

Rapport som del av projektarbete - DA159A Webbtjänster

Chanon Borgström, Sofia Hallberg (Systemutveckling)

Alva Karlborg, Adam Wiktorsson (Informationsarkitekt)

Om tjänsten

Vi har valt att döpa vår tjänst till Package Race. Som namnet antyder fungerar den ungefär som en tävling där man ser hur långt i världen man kan ta sig med reguljärt flyg under den tid det tar att skicka ett paket mellan två punkter med Postnord. Användaren möts vid första anblick av ett formulär att fylla i. Det som måste anges är land, stad och postnummer för avsändare och mottagare. På grund av begränsningar i Postnords API finns endast ett begränsat antal länder att välja mellan. Även API:et för flyg har vissa begränsningar varför samtliga resor avgår från flygplatsen i Madrid. Utan denna begränsning hade Köpenhamn/Kastrup används som utgångspunkt för flygen, då det är den största flygplatsen närmast oss.

Användaren måste också fylla i vilket datum paketet ska skickas, dagens datum rekommenderas och är ifyllt när programmet startas.

Efter att ha skickat formuläret möts användaren av en animation som visar vilka länder vårt API hittat flyg ifrån, när de lyfter och landar och hur mycket väntetid som finns däremellan. Detta gör att användaren får en tydlig bild av hur mycket tid som kan spenderas på resande fot medan denne väntar på sitt paket.

För att förtydliga transporten av paketets tillsammans med flygresorna skapas en storyline med hjälp av values som visar sig så fort användaren skickar formuläret. Exempel på en sådan storyline kan ser ut såhär:

“Resan börjar i Madrid.
Sedan åker du till Paris.
Paketet är framme vid sin slutdestination klockan 18:00 den 2021-01-22.”

Den tekniska lösningen

För frontend har HTML, CSS, Javascript, jQuery och Bootstrap använts, där HTML, CSS och Javascript tar upp störst del. För backend används Java eftersom det är det programmeringsspråk som projektgruppens systemutvecklare är mest bekväma med.

API:et är tänkt att vara pragmatiskt RESTful och för att bygga detta i Java har biblioteket Spark använts. För kommunikation med externa tjänsters API:er har biblioteket Unirest använts.

Tre externa API:er har använts:

1. *PostNords API “Transport Time Calculated Per Service”* används för att ta reda på hur lång tid det tar att skicka ett paket mellan två platser. Detta API kräver land och postnummer för både avsändare och mottagare samt vilket datum paketet ska skickas. I anropssvaret framgår vilket datum och vilken tid paketet levereras. API:et är RESTful, autentisering sker med Oauth2, och respons fås i form av ett JSON-objekt.
2. *Amadeus API “Flight Inspiration Search”* används för att få en flygdestination från en given flygplats vid ett givet datum. API:et kräver flygplatskoden och datum för avgång. I anropssvaret framgår vilka destinationer det finns avgångar till från aktuell flygplats och datum. Autentisering sker med hjälp av en API key och en API Secret som vid ett autentiseringsanrop responderar med ett token som skickas med get-anropet. Respons på get-anropet fås i form av ett JSON-objekt.
3. *Amadeus API “Flight Offers Search”* används för att få reda på avgångstid, ankomsttid och transittid för en flight. API:et kräver flygplatskoder för avgående och ankomst samt avgångsdatum. Autentisering sker med hjälp av en API key och en API Secret som vid ett autentiseringsanrop responderar med ett token som skickas med get-anropet. Respons på get-anropet fås i form av ett JSON-objekt.

Delar av datan som efterfrågas är datum, tidpunkter och tidsintervall. För att kunna göra beräkningar och jämförelser med denna data behöver den göras om från strängar till tids och datum objekt och i vissa fall transformeras vidare till integers och tillbaka till strängar.

Flygplatskoderna översätts till ort och världsdel innan respons skickas till klienten.

Installations- och körinstruktioner

För att kunna köra servern behövs bibliotek för kommunikation och hantering av date-objekt samt någon utvecklingsmiljö för Java, gruppen har använt sig av IntelliJ, och en webbläsare.

Javabibliotek för kommunikation

- Unirest
- Spark
- Gson
- Json-simple

Javabibliotek för hantering av date-objekt

- Joda-Time

För att köra applikationen

- Öppna Javaapplikationen ”PackageRace” i någon utvecklingsmiljö för Java.
- Starta programmet API-runner. När meddelandet “Server is running” visas i terminalfönstret är servern igång.
- I paketet “webbtjanster” finns en HTML-fil som heter “index.html” och som öppnas i valfri webbläsare.
- Mata in korrekta värden i formuläret via webbläsaren. Vid felaktiga värden varnas användaren.
- “Tada magic!” - Johan Holmberg

API-dokumentation

API:et tar emot och skickar data i JSON-format eftersom JSON av projektmedlemmarna upplevs som enkelt att förstå och använda. JSON

lämpar sig också väl som format eftersom kommunikation via API:et sker mellan två applikationer.

Det finns bara en ändpunkt, som nås med URL *http://localhost:5000/v1/getDestinations*, och den används för att ta reda på hur långt det går att flyga reguljärt från det att ett paket skickas iväg till det att paketet når sin mottagare. Eftersom klienten vill hämta något från ändpunkten görs get-anrop till API:et.

Till ett GET-anrop anges som inparametrar landskod- och postnummer varifrån ett paket ska skickas, landskod- och postnummer till mottagaren av paketet samt datum då paketet skickas.

För att tjänsten ska vara så RESTful som möjligt inkluderas inparametrarna i URL:en vid get-anropet.

Returdatan består av en array som representerar platser och tidpunkter för flygavgångar som hinns med innan paketet nått sin mottagare.

Som statuskoder används statuskoderna för HTTP, eftersom dessa redan finns definierade och därför förmodas vara enkla för en användare att tolka.

Den tekniska dokumentationen nås med länken längst ned i dokumentet.

Exempel på respons av en GET-förfrågan /getDestinations:

```
{
    arrivalCities: ["Europe/Paris"]
    arrivalTimes: ["21:05"]
    departureCities: ["Europe/Madrid"]
    departureTimes: ["19:40"]
    packageDeliveryDate: "20210122"
    packageDeliveryTime: "18:00"
    packageRemainingHours: "20"
    waitingTimes: [13]
}
```

https://app.swaggerhub.com/apis/PackageRace/PackageRace/1.0.0