

# Intro REST API

REST API  
&  
Node.js

# Vad är REST?

- 2000 myntade Roy Fielding uttrycket REST i sin avhandling
  - **RE**presentational **S**tate **T**ransfer
- Varför?
  - Aktuellt teknik baserad på SOAP/XML var komplext, långsamt och tungt
    - Använde ett eget protokoll för kommunikation SOAP(Simple Object Access Protocol)
    - XML för att definiera meddelande strukturer och datatyper samt endpoint adresser
  - Istället tyckte Roy Fielding att det vore bättre att utnyttja befintlig infrastruktur för att att kommunicera över internet
    - Det vill säga genom att utnyttja protokollen HTTP/HTTPS
    - Samt att utnyttja de inbyggda sätten att skicka anrop vi http verb
      - GET, POST, DELETE, PUT, PATCH osv...

# Vad är REST forts...?

- REST är en arkitektur som definierar en uppsättning regler för skapandet av webb tjänster.
- REST API är ett sätt att anropa webb tjänster på ett förenklat och flexibelt sätt
- REST grundprinciper
  1. Enhetligt gränssnitt
  2. Client-Server
  3. Stateless
  4. Cacheable
  5. Layered System
  6. Code on Demand

# SOAP och REST

- SOAP bygger på att meddelanden som transporteras är xml baserade
- XML är en beskrivning av data som kan bli stort räknat i kb och mb som ska transporteras över internet
  - Det är inte helt enkelt att komprimera ner till rimliga storlekar
  - Vilket ofta ledde till att SOAP baserade applikationer upplevdes långsamma
- SOAP är även metod/operation baserade
  - Anropen sker direkt mot definierade metoder i server applikationer
  - För att skapa server applikationer som kunde ta emot dessa metod anrop
    - Krävdes att applikationer var skapade utifrån två ytterligare dokument

# SOAP och REST

- REST är resurs orienterat
  - Betyder att en url beskriver till vilken server och vilken resurs som vi vill kommunicera med
    - Till exempel `http://localhost:3000/api/v1/vehicles`
  - Vad vi vill göra med resursen definieras av vilket http verb som används vi anropet
    - GET, POST, PUT, DELETE, PATCH osv...
  - Strukturen av ett meddelande som skickas mellan anrop och svar är oftast Json
    - JavaScript Object Notation
    - Exempel: 

```
{  
  "vehicleId": 17,  
  "manufacturer": "Volvo",  
  "model": "XC90",  
  "modelYear": "2020"  
}
```

# Resurs orienterat?

URL: `http://localhost:3000/api/v1/vehicles/`

Protokoll, server samt port

Resurs

Endpoint/Slutpunkt

Hur vet servern vad som ska utföras med resursen





# HTTP Request

- HTTP definierar vilken metod som används vid anrop
  - Webbservern kontrollerar resursnamnet och dirigerar anropet till rätt resurs
  - Webbservern undersöker HTTP verbet som användes för anropet för att avgöra vilken typ av förfrågan det är
  - Baserat på "verket" kan webbservern avgöra vilken metod/funktion som ska anropas

# HTTP Verb

<b>GET</b>	Används för att hämta data som levereras i HTTP paketets "body"
<b>HEAD</b>	Samma som GET men returnerar inget data i HTTP paketets "body"
<b>POST</b>	Används för att skicka ett objekt(json) till resursen
<b>PUT</b>	Används för att uppdatera(ersätta) allt data i objektet på resursen
<b>DELETE</b>	Används för att radera en specifik resurs
<b>PATCH</b>	Används för att uppdatera delar av ett objekt på resursen
<b>OPTIONS</b>	Beskriver kommunikations alternativ för resursen, t ex CORS



# HTTP Response(svar)

- När webbservern returnerar information så används status koder för att indikera vad som returneras
- Det är väldigt viktigt att returnera rätt status kod baserat på vilken information som returneras
  - Är det data som returneras
  - Är det ett server fel
  - Är det ett klient fel
  - Är innehållet tomt, osv ...

# HTTP Status koder

- Status koder är indelade i 5 huvudgrupper
  - Informational responses (100 - 199)
  - Successful responses (200 - 299)
  - Redirect messages (300 - 399)
  - Client Error Messages (400 - 499)
  - Server Error Messages (500 - 599)

# Vanliga status koder

<b>200 (OK)</b>	Indikerar att allt är OK och att anropet lyckades.
<b>201 (Created)</b>	Indikerar att en ny resurs är skapa på server sidan
<b>204 (No Content)</b>	Indikerar att allt gick bra men det finns inget att returnera. Används framförallt tillsammans med PUT, PATCH och DELETE
<b>400 (Bad Request)</b>	Indikerar att ett fel har inträffat, används för att tala om att information ifrån klienten är felaktig.
<b>401 (Unauthorized)</b>	Klienten är inte auktoriserad, med andra ord inte inloggad
<b>403 (Forbidden)</b>	Klienten har inte åtkomst till resursen
<b>404 (Not Found)</b>	Servern kan inte hitta resursen som efterfrågas
<b>405 (Method Not Allowed)</b>	Innebär att metoden som används inte är tillåten
<b>415 (Unsupported Media Type)</b>	Betyder att HTTP anropet inte skickar data i korrekt format

# Request & Response cykeln

