Jarno Sihvo

Nettiauton REST-rajapinnan dokumentointi

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tietotekniikan koulutusohjelma

Insinöörityö

27.10.2016

Metropolia

Tekijä(t) Otsikko Sivumäärä Aika	Jarno Sihvo Nettiauton REST-rajapinnan dokumentointi 25 sivua + 1 liite 27.10.2016
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Tietotekniikan koulutusohjelma
Suuntautumisvaihtoehto	Ohjelmistotekniikka
Ohjaaja(t)	Lehtori Kimmo Sauren Lehtori Jussi Alhorinne

Nykyajan sovellukset käyttävät usein rajapintoja laitteiden väliseen kommunikointiin ja tiedonvälitykseen. Tässä työssä käydään läpi sovellusten välisiä rajapintoja yleisesti, miten ja missä niitä käytetään, sekä esitellään erilaisia mahdollisia yritysten rajapintastrategioita. Yrityksillä voi olla erilaisia lähtökohtia rajapintojen rakentamiseen. Niistä voidaan esimerkiksi tehdä julkisia kaikkien käytettäväksi tai rajoitettuja kumppaniyritysten käytettäväksi.

Työssä esitellään lyhyesti REST-rajapintamallia, joka on yleistyvä, kevyt rajapintojen arkkitehtuurimalli. REST-rajapinnoista tutustutaan Twitterin REST API:in ja testataan sen tarjoamaa rajapinnan interaktiivista testaustyökalua.

Käytännön osuutena työssä dokumentoidaan Nettiauton.comin uutta REST-rajapintaa Swagger-kehystä hyödyntäen. Lopputuloksena on valmis interaktiivinen dokumentaatio, josta selviää rajapinnan resurssit ja käyttömahdollisuudet. Tätä dokumentaatiota hyödyntävät rajapinnan mahdolliset käyttöönottajat.

Avainsanat	Ohjelmointirajapinta, API, Swagger, REST-API

Author(s) Title Number of Pages Date	Jarno Sihvo Nettiauto REST-API documentation 25 pages + 1 appendix 27th October 2016
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Software Engineering
Instructor(s)	Kimmo Sauren, Senior Lecturer Jussi Alhorinne, Senior Lecturer

Nowadays almost every web service and mobile device application uses some kind of Application Programming Interface (API). This thesis aims to discuss APIs and define what they are, where they are used and how a company could benefit from different API strategies.

The thesis covers shortly the REST API model, which is popular, and the light API architecture used in many web services. Secondly, the thesis covers Twitter's REST API which gives an opportunity to test the API via interactive documentation and a testing tool provided by Twitter.

The thesis also documents a case study on the new REST-API of Nettiauto.com by using Swagger, an API documentation framework, to represent RESTful APIs. As result, a documentation of Nettiauto REST API was compiled, and it is displayed in the Swagger framework. The documentation covers the resources of the REST API and the use possibilities. The documentation will be used by the future users of the REST API and it is aimed to be as clear as possible to read and use.

Keywords	API, Swagger, REST-API

Sisällys

Lyhenteet ja käsitteet

Sisällysluettelo

Lyhenteet ja käsitteet

API	Application Programming Interface eli ohjelmointirajapinta.
REST	Representational State Transfer, arkkitehtuurityyli verkkosovellusten rakentamisessa.
HTTP	Hypertext Transfer Protocol. Tiedonsiirtoon käytettävä kommunikaatioprotokolla.
JSON	JavaScript Object Notation. Standardi tiedostomuoto tiedonvälittämiseen.
HTML	Hypertext Markup Language. Standardoitu kuvauskieli. Yleisesti internetsivujen kirjoittamiseen.
Swagger	Swagger on yksinkertainen tapa esittää REST-rajapintoja. Swagger sisältää useita työkaluja dokumentaation luomiseen.

1 Johdanto

Nykyajan teknologisessa maailmassa tekniikkaa ja sovelluksia käytetään helpottamaan ihmisten elämää. Erilaiset laitteet ja ohjelmistot keskustelevat keskenään jakaakseen tietoa, prosessoidakseen sitä sekä antaakseen tuloksia. Sanotaan, että toisella on tietoa keräävä ohjelmisto ja toisella on idea, miten se tulisi näyttää luettavassa muodossa esimerkiksi puhelinsovelluksen avulla. Jotta ei tarvitse itse luoda alusta asti ohjelmaa, joka kerää saman tiedon, tarjoaa toinen toiselle paikan, josta voi hakea tiedon sovellusta varten. Tämä paikka on rajapinta, jossa nämä kaksi ohjelmaa voivat keskustella yksinkertaisella tavalla toisistaan sen enempää tietämättä.

Tämän insinöörityön aiheena on kertoa rajapinnoista yleisesti, miten ja missä niitä käytetään sekä esitellä erilaisia yritysten mahdollisia rajapintastrategioita. Yrityksillä voi olla erilaisia lähtökohtia rajapintojen rakentamiseen. Niistä voidaan esimerkiksi tehdä julkisia kaikkien käytettäväksi tai rajoitettuja kumppaniyritysten käytettäväksi.

Työssä esitellään lyhyesti REST-rajapintamallia, joka on yleistyvä, kevyt rajapintojen arkkitehtuurimalli. REST-rajapinnoista tutustutaan Twitterin REST API:iin.

Nettiauto.com on Suomen suurin ajoneuvojen ilmoittelusivusto. Yli 80 000 ilmoitusta kattava sivusto on tarkoitettu sekä yksityisille että kauppiaille. Kauppiaiden kymmenet autot tuodaan sivuille joko käsin syöttämällä tai varastonhallintajärjestelmistä massoittain luettuna. Tällaisen eräsiirron tilalle ollaan luomassa rajapintaa, joka hoitaa sekä ilmoitusten jättämisen että niiden lukemisen sivuilta koneellisesti. Insinöörityössäni esittelen dokumentoinnin kyseisestä rajapinnasta Swagger-kehystä käyttäen.

2 Rajapinnat yleisesti

Luvussa kerrotaan, mitä ovat rajapinnat ja mihin niitä käytetään. Lisäksi pohditaan, miksi yritysten tulisi panostaa omaan rajapintastrategiaansa ja millaisia strategioita siihen on olemassa.

2.1 Mikä on API?

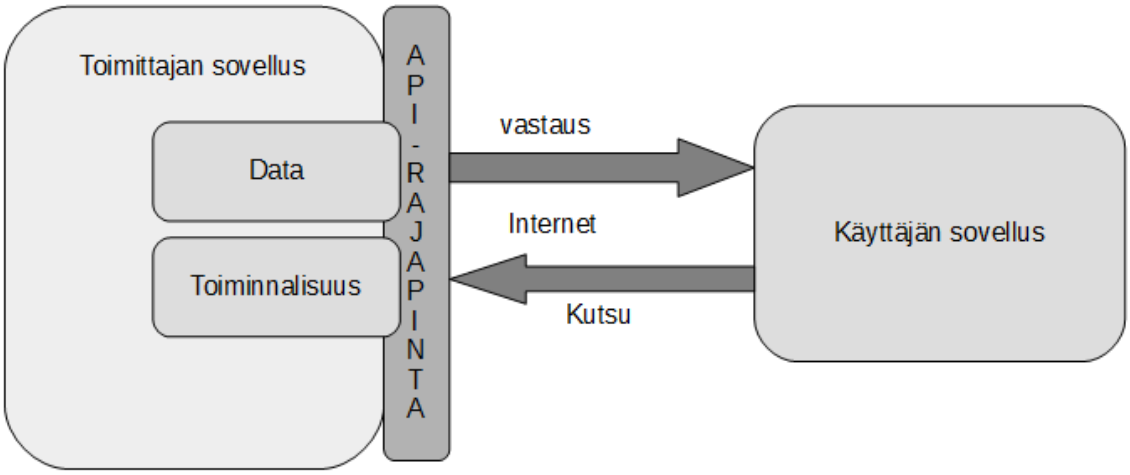
API eli Application Programming Interface on ohjelmoinnissa käytettävä rajapinta sovellusten ja ohjelmistojen rakentamiseen. Rajapinnat voivat koostua erilaisista protokollista, rutiineista ja työkaluista, joita sovellukset ja ohjelmistot käyttävät. Rajapinnat on usein rakennettu helpottamaan sovellusten välistä keskustelua ja yhteistyötä sekä luomaan yksinkertaistetun kerroksen ja piilottamaan monimutkaisuutta. Ohjelmisto voi antaa toiselle pääsyn dataan tai prosesseihin, joita kyseisessä ohjelmistossa ajetaan ilman, että toisen tarvitsee tietää, miten se toimii.

2.2 Missä ohjelmointirajapintoja käytetään?

Rajapintoja voi olla esimerkiksi erilaisissa verkkopalveluissa, käyttöjärjestelmissä, tietokannoissa. Ne tarjoavat tarvittavat toiminnot, datat ja työkalut esimerkiksi sovelluksille käytettäviksi.

Ohjelmiston kehittäjä voi tarjota ohjelmointirajapintaansa muille ohjelmistokehittäjille joko avoimesti tai luvalla käytettäväksi integraatioissa, joissa tarjoajan ohjelmasta olisi hyötyä toiselle. Esimerkiksi Helsingin Seudun Liikenne tarjoaa omaa avointa Reittiopas API:aan kehittäjille, jotka ovat kiinnostuneita luomaan sovelluksia, joissa tarvitaan HSL:n tietoa. Esimerkiksi erilaiset reitti- ja aikatauluoppaat hakevat HSL:n aikataulutietoja rajapintaa hyödyntäen. [1.]

Rajapintaa voidaan hyödyntää myös yrityksen sisällä. Esimerkiksi eri ohjelmistoja ei tarvitse rakentaa alusta asti uudelleen, vaan ne voivat kaikki käyttää yhteistä rajapintaa, josta tietoa voidaan kutsua. Näitä tietoja voisivat olla esimerkiksi käyttäjätiedot, joita ohjelma kutsuu.



Kuva 1: Rajapinnan käyttö sovellusten välillä

2.3 Miksi yrityksen tulisi rakentaa oma rajapinta?

Kuten johdannossa todettiin, Nykyään teknologia on jo suuri osa ihmisen arkipäivää, ja tulevaisuudessa se tulee olemaan sitä vielä enemmän. Uusia innovaatioita syntyy kovaa vauhtia, ja yritysten kilpailu markkinoilla kiristyy. Yritysten tavoitteena on parantaa tehokkuutta sekä lisätä kannattavuutta ja tuottavuutta. Sharifin ja Paganon mukaan parhaiten pärjäävät ne yritykset, jotka pystyvät kehittämään ketteryyttä, hyödyntämään uusia kanavia ja kasvattamaan brändiään. Teknologia ei ole niin tärkeää kuin ymmärtää, että rajapinnat ovat niitä, jotka vahvistavat sovelluksia ja prosesseja. Siksi rajapintastrategia onkin tärkeä osa yrityksen kasvua. [2, s.4.].

Yrityksen tuotteen laajentamiseen ja kehittämiseen tarvitaan usein yhteistyötä muiden yritysten kanssa. Ennen se onnistui tuomalla molempien yritysten tekniset tiimit yhteen

miettimään, kuinka yhteistyö teknisesti sovellusten tai tuotteiden välillä luodaan. Rajapinnat poistavat nämä raskaat yksittäiset prosessit yritysten välillä luoden tavan, jota hyödyntämällä yritykset ketteröittävät yhteistyötään ja monistavat kanavansa.

Ennen rajapintastrategian luomista on kuitenkin hyvä ottaa yhteyttä mahdollisiin yhteistyökumppaneihin ja selvittää, mitä toiminallisuuksia ja työkaluja heidän yrityksensä voisivat rajapinnan avulla hyödyntää. Tämä antaa suuntaa sille, millainen rajapinta tulisi kehittää ja varmistaa sen, että sille on kysyntää.

Strategian alkaessa hahmottua tulisi asettaa tavoitteet ja miettiä, kuinka ne saavutetaan. Api Academyn API Design 101 -artikkelin mukaan näiden tavoitteiden olisi hyvä heijastaa yrityksen liiketoiminnan tavoitteita. Hyvä suunnittelu hyödyntää yrityksen vahvuuksia teknisessä osaamisessa ja infrastruktuurissa, mutta ottaa myös huomioon heikkoudet ja rajoitteet, kuten budjetin.

Apigee-sivuston blogissa Brian Mulloy kirjoittaa yritysten rajapinta-strategioista jakaen sen kolmeen osaan: sisäiset-, yhteistyökumppani- ja avoimet-rajapinnat. [3.]

2.3.1 Tehokkuuden lisääminen sisäisesti rajapintojen avulla

Yritys on usein jaettu eri yksiköihin ja työryhmiin, joilla on eri johtajat ja budjetit. Tämä voi johtaa brändin murtumiseen, mikä näkyy työntekijöiden, yhteistyökumppaneiden ja asiakkaiden huonoina kokemuksina. Rajapinnat voidaan tuoda mukaan tehostamaan sisäistä toimintaa, mikä vaikuttaa kaikkiin edellä mainittuihin positiivisesti. [2, s. 12.]

Tehokkuutta voidaan lisätä sisäisesti rakentamalla rajapintoja näiden yksiköiden käyttämien prosessien välille. Yrityksen työntekijöillä voi esimerkiksi olla eri kirjautumistunnukset kaikkiin työssä käytettäviin järjestelmiin. Olisi järkevää rakentaa kirjautumisrajapinta, jota järjestelmiin kirjautuessa kutsutaan, jolloin työntekijä pärjää yhdellä tunnuksella. Tämän lisäksi uutta järjestelmää käyttöönottaessa ei tarvitse luoda uutta käyttäjäkantaa, vaan se voidaan integroida käyttämään kirjautumisrajapintaa.

Rajapintojen rakentaminen ja järjestelmien integroiminen voi olla iso ja kallis työstää, mutta sen tuoma tehokkuus jatkossa on sen arvoista. Uuden tuominen mukaan helpottuu ja nopeutuu rajapintojen myötä, kun samaa asiaa ei tarvitse tehdä uudelleen.

Yritysten huolenaiheena on myös tietoturva, jota pystytään lisäämään rajapintojen avulla. Osastojen välistä tiedonsiirtoa, järjestelmien ja palveluiden käyttöä pystytään suojaamaan ja suoraviivaistamaan rajapinnalla päästämättä ketään suoraan käsiksi taustajärjestelmiin. [3.]

2.3.2 Rajapinnat yhteistyökumppaneiden kanssa

Kasvattaakseen brändiään yritykset usein tekevät työtä yhteistyökumppaneiden kanssa. Yhteistyökumppani voi olla joku, joka tekee sovelluksia, lisäosia tai integraatioita ja tarvitsee niihin yrityksen dataa ja tietoja. Yrityksen sisäisen rajapinnan rakentamisen jälkeen onkin hyvä miettiä, olisiko tarpeellista rakentaa rajapinta, jota yhteistyökumppanit voisivat hyödyntää.

Hyvänä esimerkkinä olisi yhteistyökumppani, joka rakentaa yritykselle sovelluksen. Sovelluksen rakentaminen alusta loppuun olisi raskas ja pitkä työ, mutta valmiin rajapinnan ollessa olemassa voi sovelluskehittäjä keskittyä itse applikaation ominaisuuksiin. Tällöin myös vältytään koodin uudelleen kirjoittamiselta, jos esimerkiksi tehdään toinen sovellus tai verkkopalvelu, joka käyttää samoja tietoja.

Yhteistyökumppaneiden käyttö myös auttaa rajapinnan kehittämisessä, sillä se tulee tällöin hyvin testattua myös organisaation ulkopuolella. Samalla rajapinnan kehittäjät oppivat tuesta, dokumentoinnista sekä käyttäjien todennuksesta. [3.]

Rajapintojen avaaminen tuo uusia etuja yrityksen liiketoiminnalle, kuten avata uusia kanavia tuotteelle. Esimerkiksi verkossa toimivasta palvelusta voidaan rakentaa sovellus useisiin eri kanaviin kuten työpöytä-, mobiili- tai jopa tv-sovellus. Hyvänä esimerkkinä näistä ovat elokuvien ja sarjojen suoratoistopalvelut Netflix, Viaplay ja Ruutu+.

Apigeen blogissa Brian Mulloy myös kirjoittaa, että yhteistyökumppaneille rajapintojen avaaminen voi parantaa ja täydentää omaa päätuotetta. Kumppani voi tuoda tuotteeseen jonkun lisäosan tai parannuksen, joka houkuttaa asiakkaita tekemään ostopäätöksen.[3.]

Seuraava askel yritykselle olisi helpottaa rajapintojen automatisointia, jotta yhä useampi yhteistyökumppani voitaisiin ottaa mukaan ja tehdä tuotteesta entistä halutumpi ja tehokkaampi.

2.3.3 Yrityksen avoimet rajapinnat

Kun yritys on päättänyt tehdä suuren harppauksen digitalisoimisessa ja edellisissä luvuissa esitellyt sisäiset ja yhteistyökumppaneiden kanssa tehdyt rajapinnat ovat käytössä, on yrityksellä jo paljon tietotaitoa rajapinnoista kartutettuna. Monet yritykset ovatkin avanneet rajapintansa avoimeen käyttöön kehittäjille ympäri maailman. Tämä tarkoittaa sitä, että kenellä tahansa on vapaa pääsy rajapintaan sekä mahdollisuus rakentaa siitä jotain, mitä ei vielä ole keksitty. Avoimet rajapinnat avaavat ovet innovaatioille, joista saattaa syntyä esimerkiksi suosittuja sovelluksia.

Rajapinnan käyttäjien uudet sovellukset saattavat avata yritykselle uusia markkinarakoja. Näille markkinoille yrityksillä voi olla kiinnostusta päästä, mutta resurssit ja budjetti eivät välttämättä riitä. Avoimen rajapinnan kautta rakennettu sovellus tuo siis arvoa, niin sen tekijällekin kuin rajapinnan tarjoajalle.

Tietoa ja taitoa löytyy kehittäjiltä ympäri maailman, joten miksi sitä ei kannattaisi hyödyntää. Yksi tapa yritykselle on järjestää ns. Hackathon eli ohjelmointimaratoni, jossa tarkoituksena on jotain uutta yrityksen rajapintaa hyödyntäen. Esimerkiksi Euroopan suurimmassa startup-tapahtumassa Slushissa järjestetään Euroopan suurin hackathon-kilpailu Ultrahack, jossa kilpailijat innovoivat ja rakentavat sovelluksia käyttäen avointa dataa ja rajapintoja. [4.] Vuonna 2015 voittajat Butterfly Effect kehittivät sovelluksen, jolla kansalaiset voivat helposti tehdä uusia aloitteita kehittääkseen kaupunkiympäristöä. Sovellus hyödynsi mm. Instagramin rajapintaa ja kaupungin avointa dataa. [5.]

3 REST-rajapinnat

Ohjelmointirajapinta on laaja käsite. Käyttötarkoituksia on useita ja sitä myöten myös rajapintamalleja on syntynyt monia. Tässä luvussa esitellään REST-rajapinnan ominaisuuksia sekä esimerkki Twitterin REST-API:a.

3.1 REST-arkkitehtuuri

REST on lyhenne sanoista Representational State Transfer eli suoraan suomennettuna kuvaavan tilan siirto. REST on kommunikaatioprotokolla, joka on tilaton, tallennettava ja yleensä myös ohjelman ja palvelimen välisessä kommunikoinnissa käytetty. Se on yksinkertaistettu tapa koneiden kommunikointiin HTTP-protokollaa kutsuissa käyttäen. Rest-ohjelmistot käyttävät HTTP-kutsuja tiedon lähettämiseen, lukemiseen, päivittämiseen ja poistamiseen eli ns. CRUD-mallia käyttäen. [6.]

Tilattoman REST-arkkitehtuurista tekee se, että jokaista kutsua käsitellään yksilöllisesti. Mikään edellisistä kutsuista ei vaikuta uuteen kutsuun, vaan jokainen kutsu muodostuu omasta pyynnöstä ja sille palautettavasta vastauksesta.

Kutsujen palauttamat vastaukset ovat talletettavissa lokaalisti, joka vähentää ohjelmiston vastausaikaa ja kaistan käyttöä, jos ohjelmisto toistaa samaa kutsua. Käytännössä siis vähennetään kutsujen määrää, jos kutsun vastaus on jo tallennettu lokaalina.

Yksinkertaisimmillaan REST-rajapinta kutsu voisi olla http://api.com/tilaajat/56, jossa URL on koko GET-kutsun sisältö. Vastauksena saataisiin HTTP-vastaus sisältäen datan, jota pyydettiin. Vastauksen dataa voidaan käyttää suoraan sellaisenaan omaan tarkoitukseensa.[7.]

3.2 Twitter REST-APIt esimerkki

Twitter REST-APIt ovat hyvä esimerkki yrityksen tarjoamasta avoimesta rajapinnasta. Rajapintojen avulla kehittäjät voivat rakentaa omia ohjelmistoja tai lisäosia, joilla esimerkiksi voidaan luoda uusia twiittejä, lukea niitä tai poistaa. Google Play -kaupasta löytyy useita twitter-sovelluksia, jotka käyttävät näitä rajapintoja.

Twitterin API-dokumentaatiosta löytyy kohta GET users/lookup -kutsu, joka palauttaa user-objektin, joka sisältää käyttäjän tietoja kuten Id:n, lokaation, käyttäjätunnuksen ja käyttäjän kuvauksen. Kutsun URL on https://api.twitter.com/1.1/users/lookup.json , jossa 1.1

kertoo rajapinnan version, users on resurssia kuvaava osa polkua ja lookup.json kyseisen kutsun nimi. [8.]

Esimerkkinä on kutsu, jossa käytetään twitterin omaa rajapinnan testaamiseen tarkoitettua konsolia https://dev.twitter.com/rest/tools/console. Kutsu kokonaisuudessaan on GET https://api.twitter.com/1.1/users/lookup.json?screen_name=jarsih , jonka vastauksessa halutaan käyttäjätunnuksen jarsih tietoja. Vastauksena rajapinta palauttaa JSON-objektin, joka sisältää tietoja käyttäjätilistä jarsih. Kuvasta 2 näkee osan palautetusta JSON-objektista, jossa esimerkiksi tieto description kertoo käyttäjän itse määrittelemän kuvauksen tilistään.


```
[
  {
    "id": 399941493,
    "id_str": "399941493",
    "name": "Jarno Sihvo",
    "screen_name": "JarSih",
    "location": "Finland",
    "description": "Ultimate frisbee fanatic.",
    "url": null,
    "entities": {
      "description": {
        "urls": []
      }
    },
```


Kuva 2:Twitterin REST-rajapinna vastaus users/lookup.json kyselylle

Kutsusta saatua JSON-objektia voidaan käyttää esimerkiksi näyttämään käyttäjän tiedot omalla sivullaan parsimalla eri tiedot JSON-objektista ja esittämällä ne esimerkiksi HTML-kieltä käyttäen.

4 Nettiauton rajapintojen ja eräsiirtojen korvaus REST-rajapinnalla

Nettix Oy:n tavoitteena on korvata nykyiset ilmoitusten eräsiirrot sekä rajapinnat yhdellä REST-arkkitehtuurimallia mukailevalla rajapinnalla, jonka kautta ilmoituksia voidaan syöttää Nettiauto.com-palveluun. Rajapintaan tehdään kutsuja HTTP-protokollaa käyttäen. Rajapinnan toteuttaa erikseen ja sen dokumentoinnin toteutus kuvataan tässä työssä.

4.1 Nettiauton eräsiirtojen ja rajapintojen nykytila

Nettiauto.com on käytettyjen autojen kauppapaikka verkossa sekä yksityisille myyjille että yrityksille. Noin kaksi kolmasosaa sivustolla olevista autoista on autoliikkeiden ilmoituksia ja palvelussa on yli 1700 yritystä. [10]

Autoliikkeet markkinoivat myynnissä olevia autojaan useissa palveluissa internetissä, mutta niiden käsin syöttäminen erikseen jokaiseen palveluun on työlästä. Monilla liikkeillä on käytössään varastonhallintajärjestelmä, jolla he pitävät kirjaa myynnissä olevista autoistaan ja siirtävät niitä massoina ei palveluihin.

4.1.1 Eräsiirrot Nettiautossa

Nettiautossa olevilla autoliikkeillä on mahdollisuus tuoda ilmoituksensa massoina käyttämästään varastonhallintajärjestelmästä. Nykyisessä toteutuksessa suurin osa siirroista tehdään XML-eräsiirtoina, joissa liikkeiden ilmoitukset luetaan kerralla ja päivitetään vanhojen ilmoitusten päälle. Kolmannen osapuolen järjestelmä päivittää XML-tiedostoonsa liikkeen autoilmoitukset, minkä jälkeen Nettiauton järjestelmä käy lukemassa ilmoitukset ja syöttää ne palveluun kyseisen kauppiaan käyttäjätilille. Jos vanhoihin ilmoituksiin on tullut muutoksia, ne päivitetään, tai jos ilmoitusta ei löydy tiedostosta, se merkitään myydyksi. Eräsiirtojen ongelmana on useiden toimijoiden eri tyyliset XML-tiedostot ja tietojen esitystavat. Näiden pohjalta on jouduttu tekemään muutoksia niin tiedon esitykseen kuin lukemiseenkin.

4.1.2 Nettiauton Datapipe

Datapipe on Nettiauton rajapinta kauppiaiden ilmoitusten lukemiseen. Tämä XML-muotoinen rajapinta on kolmiportainen sisältäen listan kauppiaista, jotka on siihen kytketty, listan heidän autoilmoituksistaan sekä yksittäisten autoilmoitusten tiedot. Liikkeiden ilmoitusten tiedot voidaan näin helposti antaa kolmannen osapuolen käyttöön. Esimerkiksi kolmas osapuoli toteuttaa autoliikkeen kotisivut ja voi tuoda ilmoitusten tiedot sinne koneellisesti. Rajapinnan avaaminen kolmannelle osapuolelle tapahtuu autoliikkeen luvalla.

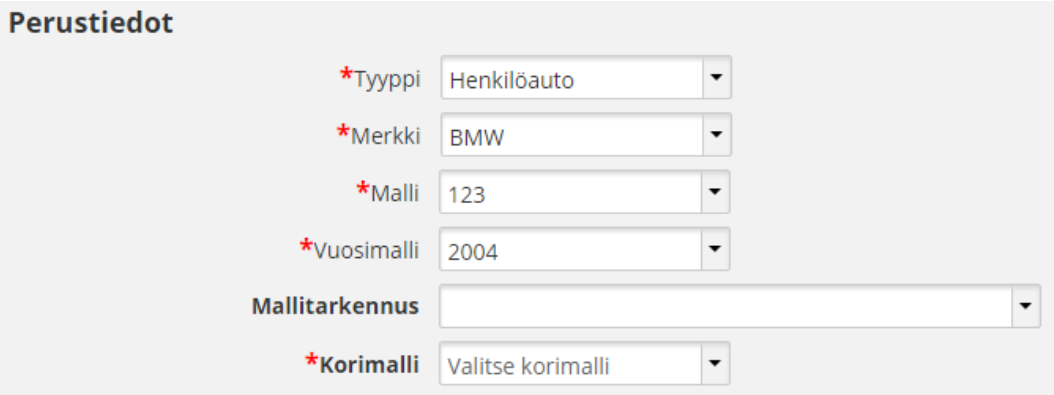
4.1.3 Autosolution rajapinta

Autosolution on Nettix Oy:n omistama ohjelmisto, joka toimii toiminnanohjausjärjestelmänä autokauppiaille. Ohjelmisto on integroitu Nettiautoon rajapinnan kautta, mikä mahdollistaa reaaliaikaisen ilmoitusten syöttämisen suoraan ohjelmistosta. Ohjelmisto käyttää samoja valintoja ja vaihtoehtoja kuin Nettiauton ilmoituksen jättö, joten marginaali virheille on pieni.

4.2 Uuden rajapinnan määrittely

Uusi rajapinta tulee korvaamaan edellä mainitut eräsiirrot sekä rajapinnat, joiden kautta ilmoituksia lisätään ja luetaan. Tämä halutaan toteuttaa siksi, että kaikki toiminnot voidaan hoitaa kerralla samasta rajapinnasta. Muutoksien tekeminen tapahtuu tällöin vain yhteen paikkaan.

Nettiautossa ilmoituksen jättäminen tapahtuu syöttämällä pyydetyt tiedot myytävästä autosta mahdollisimman tarkasti. Tietoihin kuuluvat auton perustiedot kuten tyyppi, merkki, malli ja vuosimalli sekä tekniset tiedot kuten mittarilukema, polttoaine ja vetotapa. Myös ilmoitusten jättäminen verkkosivuilla on tarkoitus laittaa uuden rajapinnan päälle. Kuvassa 3 on osa Nettiauton ilmoituksen jätön perustietojen valitsemisen käyttöliittymästä.



Kuva 3: Nettiauton ilmoituksen jätössä on pudotusvalikoita, ruksittavia valintaruutuja ja vapaa tekstikenttiä

Osa tiedoista on vapaata tekstiä, kuten hintapyyntö ja osa pudotusvalikoita, joissa vaihtoehdot valinnalle, kuten vaihteisto (manuaalinen tai automaattinen). Näiden lisäksi ilmoituksen jätöstä löytyy myös ruksittavia valintaruutuja, jotka kertovat, löytyykö kyseinen ominaisuus autosta. Valintaruutuja löytyy esimerkiksi varusteista. Vapaista kentistä osaan syötetään numeroita ja osaan tekstiä.

Uuden rajapinnan yksi käyttötarkoituksista on antaa käyttäjälle mahdollisuus hakea vaihtoehdot, joita ilmoituksen jätössä oleviin kenttiin on sallittua syöttää. Esimerkiksi kaikki mahdolliset merkit voisi hakea tämän rajapinnan kautta. Toimija voisi näitä tietoja hyödyntäen rakentaa omaan palveluunsa valikkoja siten, että niistä saa vaihtoehdoiksi vain Nettiautossa sallittuja tietoja.

Yritykset voivat antaa kolmansille osapuolille luvan hakea heidän ilmoituksensa Nettiautosta ja käyttää niitä esimerkiksi kotisivujensa sisällön luomiseen. Tällöin Nettiautoon lisätyt ilmoitukset saadaan rajapinnan kautta vietyä kauppiaan omille sivuille. Ilmoitusta hakiessa rajapinnan tulisi palauttaa samat tiedot, joita kysytään ilmoitusta jättäessä.

5 Nettiauto-rajapinnan dokumentointi Swaggerilla

Nettiauton uusi rajapinta, joka korvaa ja yhdistää olemassa olevat, kuvataan ja dokumentoidaan Swagger-kehikolla, joka on tarkoitettu kuvamaan REST-rajapintoja selkeästi ja interaktiivisesti. Tämä helpottaa mahdollisen käyttäjän rajapinnan käyttöönottoa. Tämä dokumentaation luominen ja testaus suoritettiin lokaalisti omalla tietokoneella, sillä rajapinta ei ollut valmis dokumentaatiota tuotettaessa. Dokumentoinnissa on mukana rajapinnan osat, jotka on tarkoitettu ilmoituksen jättöön sekä ilmoituksen jätössä käytettävien vaihtoehtojen hakuun. Tässä dokumentoinnissa on jätetty pois osio, jolla ilmoituksia voidaan lukea rajapinnan kautta. Rajapintojen dokumentointi Swaggerilla on aloitettu jo ennen rajapintojen valmistumista, jolloin niiden suunnittelu ja tarkka läpikäynti mahdollistaa rajapinnan muutokset lennosta.

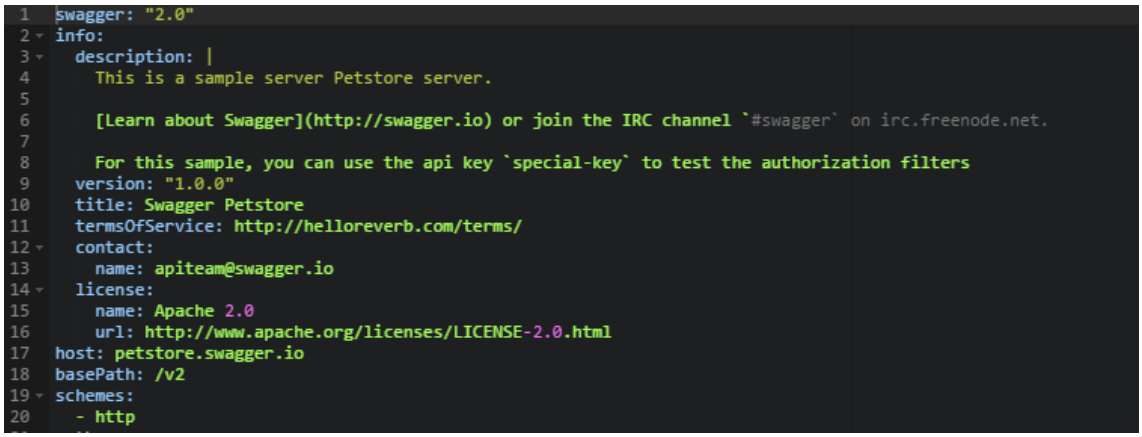
5.1 Swagger kehitystyökalut, käyttöönotto ja määrittelytiedostot.

Swagger-kehikon voi ottaa käyttöön lataamalla Swagger-UI:n GitHubista, jolloin saa kaikki Swagger-frameworkin tarvittavat tiedostot omalle koneellensa. Tiedostot voi ladata omalle palvelimelleen, jolloin interaktiivisen dokumentaation testaaminen voidaan tehdä suoraan Swagger-UI-käyttöliittymästä valmiiseen rajapintaan. Swagger käyttöliittymänäkymän saa myös näkymään omalla koneellensa selaimella käyttämällä esimerkiksi avoimen lähdekoodin Xampp-ohjelmistolla pyörivää Apache-palvelinta lokaalisti.

Swagger luo Swagger-UI-nimisen käyttöliittymän dokumentaatiolle REST-rajapinnan määrittelytiedostoista, joissa rajapinta esitetään joko JSON-objekteina, jotka käyttävät tunnettuja JSON-standardeja, tai YAML-kielellä kuvattuna. [10.] Määrittelytiedoston muodon ja sen sisältämät tiedot ovat määritelty Swagger Open API -määrittelyssä, jota seuraamalla ja mukailemalla Nettiauton rajapinta-dokumentaatio on luotu.

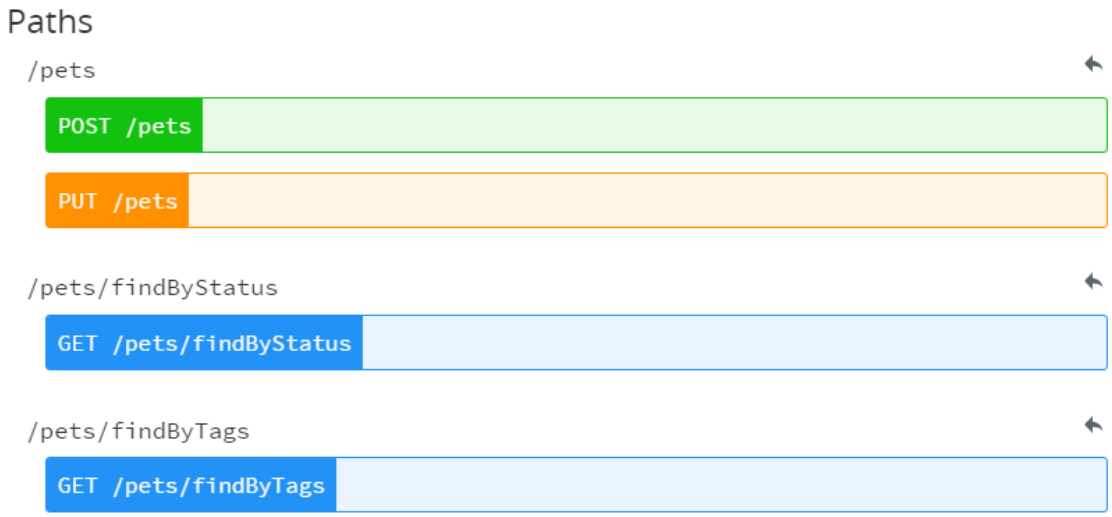
Swagger Editor mahdollistaa määrittelytiedoston muokkaamisen omassa selaimessa YAML-kielellä tai käännettynä JSON-objekteista. Editorissa on reaaliaikainen virheiden

käsittely ja virheilmoitukset. Myös muutokset määrittelytiedostossa näkyvät dokumentaatiossa reaaliajassa. [11.]



Kuva 4:Swagger Editor -työkalu selaimessa

Kuvassa 4 Swagger Editorin esimerkki Pet Store -rajapinnan määrittelytiedoston muokkaustyökalusta. Editori tarkistaa syntaksin ja merkkaa virheet. Kuvassa 5 on määrittelytiedoston muodostama esikatselu käyttöliittymälle. Käyttöliittymän esikatselu on myös interaktiivinen, joten siinä olevia painikkeita voi käyttää.



Kuva 5:Käyttöliittymän esikatselu Swagger Editorissa

5.2 Resurssien dokumentointi

Nettiauton rajapinta ohjelmoidaan PHP-kielellä. Kutsut lähetetään rajapintaan HTTP-kutsuina, joiden sisällä mahdolliset parametrit JSON-objekteina. Rajapinta palauttaa aina JSON-objekteja HTTP-vastauksen sisällä. Kaikkien polkujen kanta on /rest/car.

Taulukko 1: Nettiauto-rajapinnan resurssit

Kutsun toiminto	Metodi	Kutsun URL-osoite	Kutsun parametrit	Vastauksen parametrit
Hae vaihtestotyypit	GET	/options/gearType		id,fi,en
Hae maat	GET	/options/country		id,fi,en
Hae polttoaineet	GET	/options/fuelType		id,fi,en
Hae merkit	GET	/options/make		id,fi,en
Hae tietyn merkin mallit	GET	/options/make/{idMake}	IdMake – id merkille, jonka malleja haetaan	Id,fi,en
Hae maakunnat	GET	/options/domicile		Id,fi,en
Hae maakunnan kaupungit	GET	/options/domicileTown/{idDomicile}	IdDomicile – id maakunnalle, jonka kaupunkeja haetaan	Id,fi,en
Hae vetotavat	GET	/options/driveType		Id,fi,en
Hae ajoneuvotyypit	GET	/options/vehicleType		Id,fi,en
Hae värit	GET	/options/color		Id, fi, en
Hae värityypit	GET	/options/colorType		Id,fi,en
Hae moottorintilavuudet	GET	/options/engineSize		Id,fi,en
Ilmoituksen jättö	POST	/ad	ks. Liite 1	Id

Swaggerin määritystiedostoihin resurssit määritellään polkuina Path-osioon. Kuvassa 6 on esimerkki gearType-kutsun dokumentoinnin luomisesta määrittelytiedostossa. Alkuun kerrotaan polku /options/gearType, joka on URL-osoite, josta rajapintaa kutsutaan.

```
paths:
  /options/gearType:
    get:
      tags: [ Options ]
      summary: List all allowed gear types test
      description: Get list of all allowed gear types used in nettiauto.com.
      responses:
        200:
          description: An array of gear types
          schema:
            type: array
            title: Options
            items:
              $ref: '#/definitions/Option'
        400:
          description: Unexpected error
          schema:
            type: array
            title: Error
            items:
              $ref: '#/definitions/Error'
```

Kuva 6: Polkujen luominen

Tämän jälkeen ilmoitetaan metodi. Tässä tapauksessa GET eli kyseinen kutsu lukee rajapinnasta. Tags-osioon lisätään nimi osiolle, johon kyseinen kutsu kuuluu. Samaan osioon kirjatut kutsut järjestetään Swagger UI -dokumentaatiossa samaan paikkaan. Summary kertoo lyhyesti, mitä kutsu tekee, ja description-objektiin voi tarkentaa tietoja kyseisestä resurssista.

Responses-objektiin annetaan mahdollisten vastausten määrittely. Kuvassa 6 vaihteistotyyppien haussa saatavat vastaukset ovat ”200” eli onnistunut sekä ”400” eli ns. Error. Onnistunut vastaus palauttaa jonon objekteja, jotka mukailevat Option-objektia. Option objektin on määritelty tiedostossa myöhemmin ja siihen voidaan viitata seuraavasti: $ref: '#/definitions/Option'.

Kuvan 6 mukainen määrittely luo Swagger-UI -näkymään kuvassa 7 näkyvän dokumentoinnin. Responce Class (Status 200) näyttää onnistuneen vastauksen palauttaman JSON-objektin, jossa mukana yhden vaihteistotyypin id, suomenkielinen nimi ja englanninkielinen nimi.

Kuva 7: Swagger-UI dokumentaation resurssi vaihteistotyyppien hakuun.

Kuvassa 7 näkyy myös Error-vastaus, jonka HTTP Status -koodina on 400 ja syyksi määritelty ”Unexpected error”. Tälle virhemallille ei ole määritelty vastauksen JSON-muotoa, joten Response model -sarake on tyhjä.

5.3 Objektien määrittäminen

Resurssien eli polkujen määrittelyn lisäksi määrittelytiedostosta löytyy Definitions-objekti, jonka sisälle voidaan tehdä määrittelyjä eri objekteista. Näihin objekteihin voidaan esimerkiksi viitata resurssien määrittelyjä tehtäessä. Tämä vähentää uudelleenkirjoittamista, jos objekti toistuu usein määrittelyissä.

Taulukko 2: Määrittelytiedostossa määritetyt objektit

Objekti	Selitys
Options	Jokainen options-polun resurssi palauttaa onnistuessaan tämän muotoisen JSON-objektin.
Error	Error-objekti esiintyy jokaisessa resurssissa. Palauttaa virhekoodin
New Car	Uuden autoilmoituksen luomiseen vaadittavat ominaisuudet ovat määritetty tähän objektiin.
Car	Uutta ilmoitusta luodessa rajapinta palauttaa tämän JSON-objektin, joka kertoo ilmoituksen tiedot.

Options-objekti esiintyy jokaisen options-kutsun onnistuneessa vastauksessa. Esimerkiksi vaihteistotyyppejä haettaessa kutsu palauttaa JSON-objektin, joka näkyy kuvassa 8. Vastauksessa esiintyä ”id” on vaihteistotyypin id, ”fi” sen suomenkielinen nimi ja ”en” englanninkielinen nimi, jos se on saatavilla.

Response Body


```
[
  {
    "id": "3",
    "fi": "Automaatti",
    "en": "Automatic"
  },
  {
    "id": "1",
    "fi": "Ei saatavilla",
    "en": "Ei saatavilla"
  },
  {
    "id": "2",
    "fi": "Manuaali",
    "en": "Manual"
  }
]
```


Kuva 8: GET /options/gearType palauttama JSON-objekti

Error-objekti esiintyy epäonnistuneissa kutsuissa, ja se sisältää kohdat ”Code”, joka kertoo virhekoodin, ja ”Message”, joka kertoo virhekoodin syyn.

New Car -objektia käytetään luodessa uutta ilmoitusta, ja se sisältää kaikki tarvittavat tiedot ilmoituksen jättämiseen. Osa tiedoista on määritetty pakollisiksi ja niiden tulee esiintyä kutsua lähetettäessä. Car-objektin määrittelee onnistuneen ilmoituksen lisäys-kutsun vastauksessa esiintyvän JSON-objektin. Vastauksessa on kaikkien ilmoituksen tietojen lisäksi sille määritetty ilmoituksen id.

```
NewCar:
  required:
    - idMake
    - idModel
  properties:
    idVehicleType:
      description: >-
        ID for defining type of vehicle. List of vehicle type id's found in
        /car/ad/options
      type: integer
      example: '3'
    idMake:
      description: >-
        Make ID tells the make of the vehicle. List of ID's for makes
        /car/options/make
      type: integer
      format: int32
      example: '6'
    idModel:
      description: 'Car Model ID. This ID is found from car/options/model/{idMake}'
      type: integer
      format: int32
      example: '13'
```

Kuva 9: Osa YAML-kielellä määriteltyä NewCar-objektia

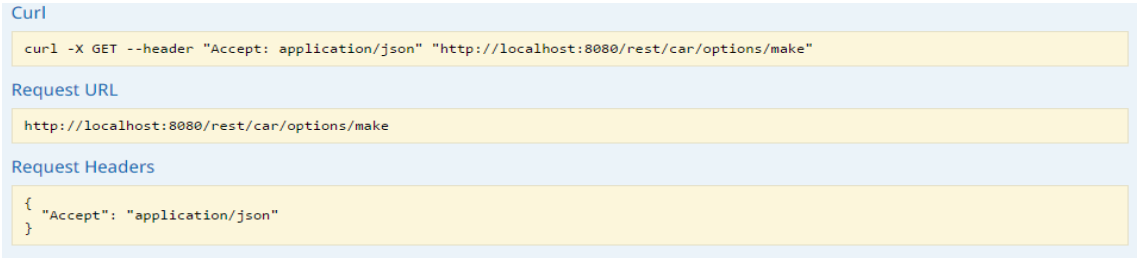
Kuvassa 9 on esimerkkinä osa NewCar-objektin määrittelyä. Required-osio kertoo, mitkä tiedot ovat pakollisia tässä objektissa. Properties-osassa määritellään objektin sisältämät tiedot ja niiden tarkennukset. Esimerkiksi idVehivleType on määritelty tyypiksi integer ja sille on annettu esimerkki arvo ”3”. Description-osassa voidaan kertoa lisätietoja kyseisestä ominaisuudesta. Nämä lisätiedot ovat esillä dokumentaatiossa vastauksen mallissa, josta käyttäjät voivat katsoa tarvittavat tiedot ilmoituksen lisäämiseen.

5.4 Rajapinnan testaaminen Swagger-dokumentaatiossa

Rajapinnan toimivuutta voidaan testata Swagger-UI-käyttöliittymässä. Testatessa valmista dokumentaatiota rajapinnan kanssa, on koneelle asennettava MySQL tai vastaava tietokannan käsittelyjärjestelmä. Sinne lisätään tarvittavat tietokannat oikeiden vastausten saamiseksi rajapinnasta. Koko Nettiauton tietokantaa ei tarvinnut syöttää, vaan riitti,

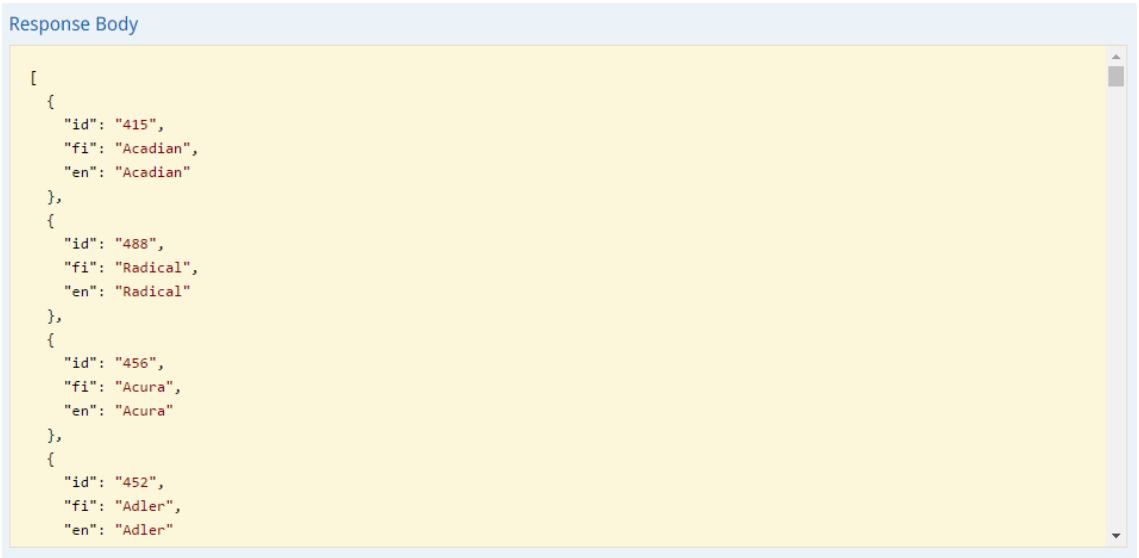
jos tietokannassa on rajapinnan käyttämät kannat. Kun määrittelyt on tehty oikein ja tietokannat on asennettu, voi Swagger-UI:ssa avata jonkin resursseista ja painaa ”Try it out”-nappia, joka tekee kyseisen kutsun rajapintaan.

Kuvassa 10 Swagger-UI:ssa tehdystä rajapinta kutsusta /options/make-polkuun, jonka tulisi palauttaa tietokannasta kaikki vaihtoehdot, joita Nettiautossa voi valita auton merkiksi. Rajapinnan koodeja ja Swagger-määrittelytiedostoja pidettiin testivaiheessa lokaalisti omalla koneella, ei palvelimella.



Kuva 10: Esimerkki rajapinta kutsusta Swagger-UI:ssa

Swagger lähettää curl-komennolle pyynnön rajapinnan osoitteeseen, joka on ”http://localhost:8080/rest/car/options/make”. Kutsun tunnisteessa se kertoo, minkälaista resurssia odotetaan vastauksena eli ”application/json”.



Kuva 11: Kutsun palauttama JSON-objekti

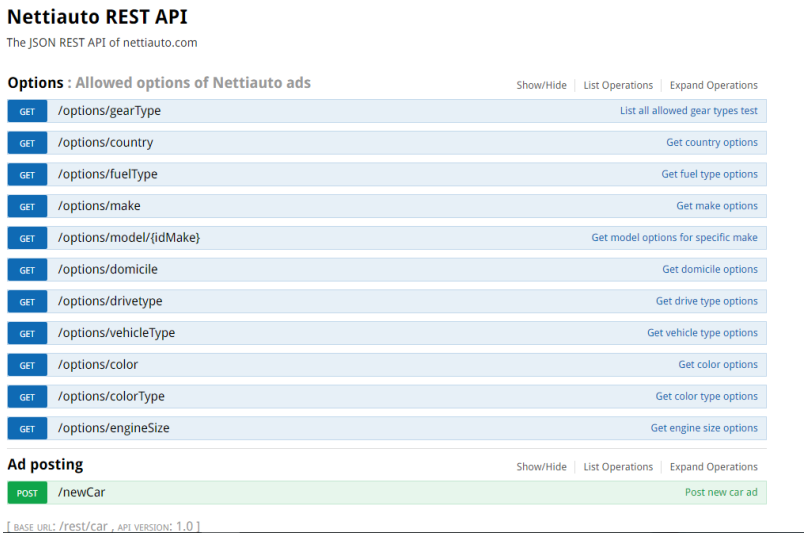
Kuvassa 11 on onnistuneen rajapintakutsun palauttama vastaus, joka sisältää JSON-objektissa kaikkien merkkien id:t, suomen- ja englanninkieliset nimet. Tätä objektia käyttäjä voi parsia koneellisesti listatakseen merkit itsellensä.

Kutsun testaamisen yhteydessä Swagger myös listaa vastauksen koodit sekä tunnisteen tiedot. Kuvassa 12 testatun kutsun Response Code ”200” ja tunnisteen tiedot, kuten sisällön pituus ja sisällön tyyppi.

Kuva 12: Kutsun vastauskoodi ja tunnistetiedot

5.5 Valmiin dokumentaation käyttöliittymä

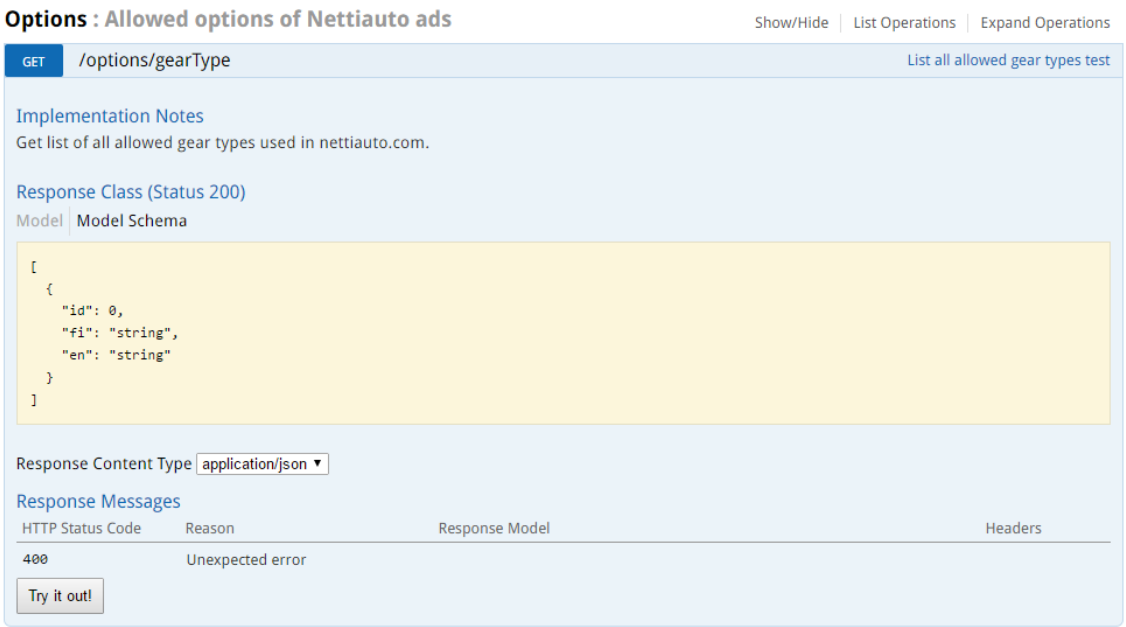
Tässä luvussa esitellään valmiin dokumentaation käyttöliittymää. Dokumentaatio on selkeälukuinen ja sen tarkoituksena on opastaa mahdollista rajapinnan käyttöönottajaa olettaen, että käyttäjällä on tietoa rajapinnoista ja niiden käyttöönotosta.



Kuva 13: Dokumentaation käyttöliittymän yleiskuva

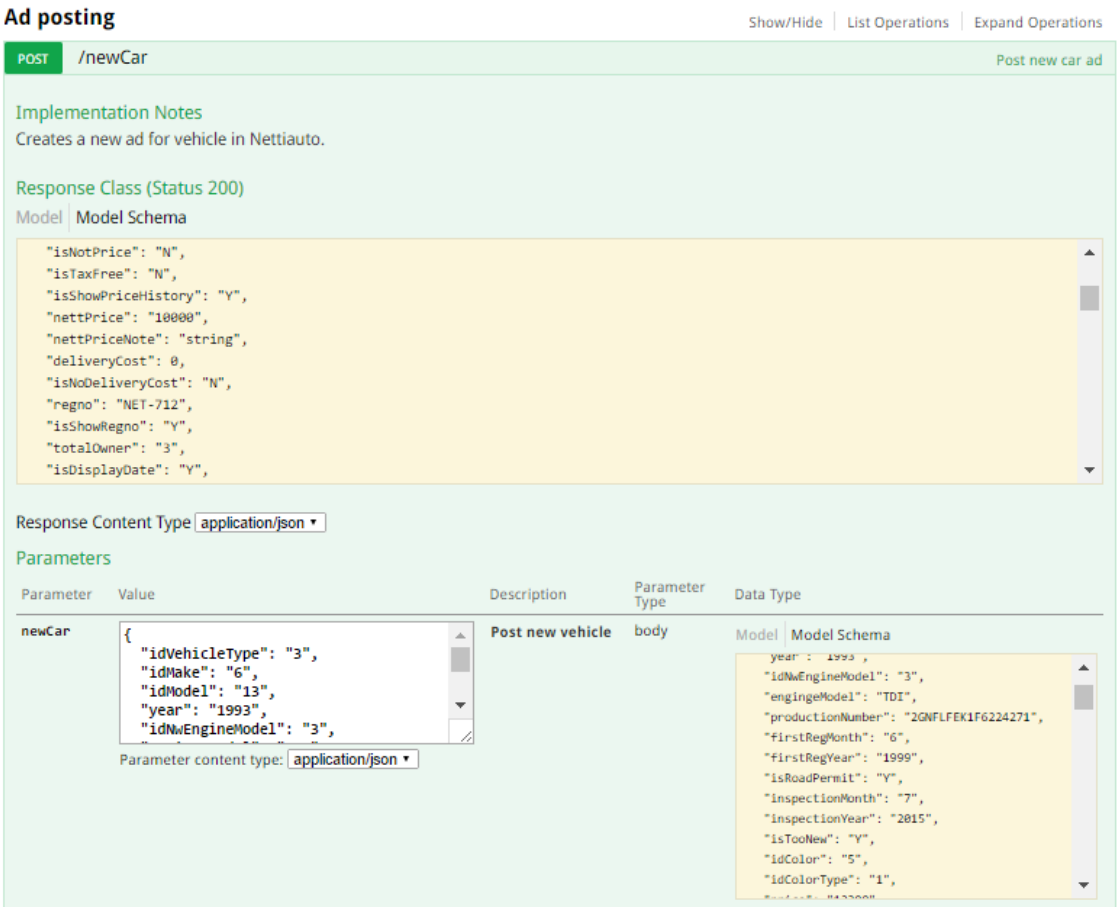
Kuvassa 13 on rajapintadokumentaation yleisnäkymä Swagger-UI:ssa. Dokumentaation alussa kerrotaan rajapinnan nimi sekä lisätietoa rajapinnan tyypistä. Resurssit on jaoteltu kahteen osioon: Optionsiin, jossa GET-kutsuilla haettavat vaihtoehdot mahdollisille tiedoille, ja Ad postingiin, jossa polku ilmoituksen jättöön käytettävään kutsuun. Get-kutsut ovat sinisellä ja Post-kutsut vihreällä. Resurssien lyhyet selitykset löytyvät samalta riviltä polun jälkeen oikealta laidalta.

Jokaisen resurssin voi laajentaa nähdäkseen tarkemman sisällön. Laajennus avaa näkymän, jossa kerrotaan, mitä kutsu tekee ja mitä se palauttaa. Model-välilehdessä esitetään vastauksen sisältö ja osat sekä niiden lisätiedot. Model Schema -välilehdessä esitetään vastauksen palauttama sisältö JSON-objektina ja esimerkki tiedoilla. Kaikkien resurssien esityksen perässä on ”Try it out” -nappi, jolla kutsua rajapintaan voi testata. Kuvassa 14 näkyy gearType-kutsun dokumentointi, jossa esillä on Model Schema.



Kuva 14: gearType-kutsun dokumentointi

Kuvassa 15 newCar POST -kutsun esitystapa dokumentaatiossa. Parametreiksi sille on määritetty ainoastaan newCar-niminen body-typpinen parametri, jonka sisällä kerrotaan JSON-objektin muodossa kaikki autoilmoituksen tiedot. Datatype-sarakkeessa on määritetty esimerkki, johon on lisätty tarvittavat tiedot ilmoituksen jättämiseen. Tätä klikkaamalla Swagger-UI kopioi esimerkki bodyn suoraan kenttään, johon kyseinen JSON-objekti tulisi sijoittaa kutsua tehtäessä. Tällöin voidaan testata suoraan valmiilla esimerkkitiedoilla täytettyä kutsua.



Kuva 15: newCar-kutsun esitystapa dokumentaatiossa

6 Yhteenveto

Työn tarkoituksena oli selvittää rajapintojen toimintaa ja käyttötarkoituksia sekä tutkia yritysten mahdollisia rajapinta strategioita. Työssä myös dokumentoitiin Nettiauto.comin uusi REST-rajapinta Swagger frameworkia käyttäen.

Nettiauton rajapinnan dokumentoinnista haluttiin tehdä mahdollisimman selkeä, jotta sen mahdollisten käyttöönottajien olisi helppo, dokumentaation avulla, implementoida rajapinta omiin käyttötarkoituksiinsa. Rajapinnan dokumentointi toteutettiin Swagger-dokumentaatiotyökaluilla, joilla saadaan selkeä kuvaus rajapinnasta ja sen resursseista. Swagger-työkaluilla rajapinnan ominaisuudet tulee käytyä läpi tarkasti, joten se auttaa myös syventämään omaa ymmärrystä rajapinnasta.

Swaggerin interaktiivisella dokumentaation käyttöliittymällä rajapintaa voidaan halutessaan testata omalta selaimelta. Rajapinnan vielä ollessa työn alla, testaaminen toteutettiin pystyttämällä tuotantoa vastaavat resurssit, kuten tietokanta ja rajapinnan koodipohjat.

Jatkokehityksenä dokumentaatiolle olisi lisätä siihen myöhemmin kehitettäviä kutsuja, tarkoituksena koota kaikki samaan paikkaan. Lisäksi käyttöliittymän ulkoasua olisi mahdollista muuttaa HTML- ja CSS-muutoksilla, esimerkiksi Nettiauton värimaailmaa mukaillen.

Lähteet

1 Reittiopas API developers guide, Verkkodokumentti. HSL <http://developer.reittiopas.fi/pages/en/home.php>, Luettu 01.03.2016.

2 Nijim, Sharif & Pagano, Brian. 2014. APIs For Dummies, Apigee Special Edition. Hoboken, NJ: John Wiley & Sons, Inc.

3 The Why and How of APIs, Verkkodokumentti. Apigee <http://apigee.com/about/blog/digital-business/why-and-how-apis-internal-api-model>, Luettu 30.3. 2016.

4 Slushin yhteyteen Euroopan suurin Hackathon, Verkkodokumentti. Teknologiateollisuus. <http://teknologiateollisuus.fi/fi/ajankohtaista/uutiset/slushin-yhteyteen-euroopan-suurin-hackathon> Luettu 28.7.2016.

5 Striking developer weekend at Ultrahack: Butterfly Effect and Routes Helsinki pocket the main prizes, Verkkodokumentti. Helsinki Region Infoshare. <http://www.hri.fi/en/news/striking-developer-weekend-at-ultrahack-butterfly-effect-and-routes-helsinki-pocket-the-main-prizes> Luettu 28.7.2016.

6 What exactly is restful programming?, Verkkodokumentti. Stackoverflow. <http://stackoverflow.com/questions/671118/what-exactly-is-restful-programming> Luettu 24.10.2016.

7 How simple is REST?, Verkkodokumentti. Learn REST: a Tutorial http://rest.elkstein.org/2008/02/how-simple-is-rest.html Luettu 24.10.2016.

8 Twitter Developer Documentation, Verkkodokumentti. Twitter. <https://dev.twitter.com/rest/reference/get/users/lookup> Luettu 24.10.2016.

9 Nettiauto, Verkkodokumentti. Nettix. <http://www.nettix.fi/nettiauto> Luettu 19.10.2016.

10 Swagger Specification, Verkkodokumentti. Swagger. <http://swagger.io/specification> Luettu 19.10.2016.

11 Swagger Editor, Verkkodokumentti. Swagger. <http://swagger.io/swagger-editor> Luettu 19.10.2016.

Uuden ilmoituksen vaadittavat parametrit ja selitykset

idVehicleType(integer):
ID for defining type of vehicle. List of vehicle type id's found in /car/ad/options

idMake(integer):
Make ID tells the make of the vehicle. List of ID's for makes /car/options/make

idModel(integer):
Car Model ID. This ID is found from car/options/model/{idMake}

year(integer):
Year of manufacture for the vehicle. Values starting from **1901** to **current year**
idNwEngineModel(integer):
ID Engine model of the vehicle. Options for engine model depend on the make, model and manufacture year of vehicle
engingeModel(string):
If engine model not found for certain make, model, year it can be free text on this parameter
productionNumber(string):
Vehicle Identification Number is unique proudction number for vehicle.
firstRegMonth(integer):
Number of the month when the vehicle was first registerd. Can be found from registration document of vehicle
firstRegYear(integer):
Year of registration for the vehicle. Can be found from registration document of vehicle
isRoadPermit(string):
This parameter tells if the vehicle is road worthy or not. If vehicle has flaws that prevent driving or vehicle is not inspected it is not roadworthy. **Y** means vehicle has road permits and **N** that vehicle is not roadworthy
=['Y', 'N'],
inspectionMonth (integer):
Number of the month when the vehicle was inspected last time. Can be found from registration document of vehicle
inspectionYear(integer):
Year when the vehicle was inspected last time. Can be found from registration document of vehicle
isTooNew(string):
Tells if the car is so new that it hasn't been inspected yet and the first inspection is coming.
=['Y', 'N'],
idColor(integer):
Id for the color of the car. List of id's for color options can be found /car/options/color
idColorType(integer):

Id for the type ofcolor of the car. List of id's for color type options can be found /car/options/colortype
price(integer):
Asking price for vehicle in euros. Minimum value is 50. Only characters 0-9 and max. 10 characters. Price is mandatory unelse vehicle is posted as "not priced".
isNotPrice(string):
Value tells if vehicle is set as "not priced" and posted without asking price.
=['Y', 'N'],
isTaxFree(string):
Value tells if the vehicle is tax free which means that the VAT and Car Tax is yet to pay.
=['Y', 'N'],
isShowPriceHistory(string):
Dealer has option to choose if they want to show the price history of vehicle
=['Y', 'N'],
nettPrice(integer):
Dealers can set nett price for their vehicles. Nett price is shown only in dealer exchange section in Nettiauto where dealers sell cars to other dealers
nettPriceNote(string):
Note field for Nett price. Additional information about the price. Max 200 characters, Max word length40 characters.
deliveryCost(integer):
Cost of delivery for the vehicle
isNoDeliveryCost(string):
Tells if dealer has no delivery cost for the vehicle
=['Y', 'N'],
regno(string):
The register number of the vehicle. E.g abc-123
isShowRegno(string):
Tells if dealer wants to show register number on car detail page
=['Y', 'N'],
totalOwner(string):
Tells the amount of owners of the vehicle. Value range 1-20
isDisplayDate(string):
Tells weather dealer wants to show the date ad was posted
=['Y', 'N'],
milage(integer):
Mileage that the vehicle has travelled in kilometers.
isNoMilage(string):
Tells if the mileage is not known
=['Y', 'N'],
idFuelType(integer):
Type of fuel the vehicle uses. Id's for fuel type options can be found from /car/options
engineSize(integer):
Size of the engine used by vehicle in litres. From 0.0 to 10.0 in decimal intervals

idDriveType(integer):
Drive type of the vehicle. Id's for drive type options can be found from /car/options/driveType
idGearType (integer):
Gear type of the vehicle. Id's for drive type options can be found from /car/options/gearType
seats(integer):
Number of persons the car can carry. Range is 1-12
doors(integer):
Number of doors the car has. Range is 2-7
driversPosition(string):
Tells which side the steering wheel is on.
=['L', 'R'],
power(integer):
Amount of power the engine can produce
powerUnit(string):
Unit of the power, kilowatts or horsepower
=['kw', 'hp'],
batteryCapacity(integer):
How much capacity on battery if the fueltype of the vehicle is electric or hybrid unit: kWh
co2Emission(integer):
amount of C20-Emission of the car. Unit is g/km
topSpeed(integer):
How fast can the car go. Unit is kilometers per hour
acceleration(integer):
How fast can the car reach from 0 to 100 km/h
consumeUrban(integer):
Fuel consumption when driving in city. Unit is litres per 100 kilometres
consumeRoad(integer):
Fuel consumption when driving the road. Unit is litres per 100 kilometres
consumeCombined (integer,):
Consumption of the fuel combining road and city consumption. Unit is litres per 100 kilometres
,
curbWeight (integer,):
Curb weight of the vehicle in kilograms
,
grossWeight (integer,):
Total weight of the vehicle in kilograms
,
towWeightBrake (integer,):
Tow weight with brakes in kilograms
,
towWeightNoBrake (integer,):
Tow weight without brakes kilograms
,
locationDealerId (integer,):

Id for dealer's location for the car
,
idCountry (integer,):
Id for the country where the vehicle is located. Id's can be found from /car/options
,
idDomicile (integer,):
Id for the Domicile where the vehicle is located. Id's can be found from /car/options
,
idTown (integer,):
Id for the Town where the vehicle is located. Id's can be found from /car/options
,
isShowExactLocation (string):
For sellers if they want to show the exact location of the car in the vehicle detail page
= ['Y', 'N'],
streetAddress (string):
Street address of the exact location of the vehicle. Mandatory if isShowExactLocation=Y
,
accessories (accessoriesList,),
adExchange (adExchange,),
accessoriesList (accessories,),
carId (integer)
}
accessoriesList {
}
adExchange {
isAdExchange(string):
**Are you interested to exchange cars?(Seller login only)
=['Y', 'N'],
exchangeMailFreq(string):
**Email me about new ads that match my interests. D=Daily, W=Weekly, N=Never(Seller login only)
=['D', 'W', 'N'],
exchangeModel[45](string):
**Car Exchange model value in array. Pass value for set, otherwise blank.(Seller login only)
=['1'],
exchangeModel[2920](string):
**Car Exchange model value in array. Pass value for set, otherwise blank.(Seller login only)
=['1'],
exchangeSubType[2](string):
**Car Exchange subtype value in array. Pass value for set, otherwise blank.(Seller login only)
=['1'],
exchangeSubType[51](string):
**Car Exchange subtype value in array. Pass value for set, otherwise blank.(Seller login only)
=['1'],

exchangePriceFrom(integer):
**Price From
exchangePriceTo(integer):
**Price To
exchangeYearFrom(integer):
**Year From
exchangeYearTo(integer):
**Year To
exchangeMileageFrom(integer):
**Mileage From (km)
=['1', '5000', '10000', '15000', '20000', '25000', '30000', '35000', '40000', '45000', '50000', '55000', '60000', '65000', '70000', '75000', '80000', '85000', '90000', '95000', '100000', '105000', '115000', '125000', '135000', '145000', '155000', '165000', '175000', '185000', '195000', '205000', '215000', '225000', '235000', '245000', '255000', '265000', '275000', '285000', '295000', '320000', '350000', '400000', '450000', '500000', '550000', '600000'],
exchangeMileageTo(integer):
**Mileage To (km)
=['1', '5000', '10000', '15000', '20000', '25000', '30000', '35000', '40000', '45000', '50000', '55000', '60000', '65000', '70000', '75000', '80000', '85000', '90000', '95000', '100000', '105000', '115000', '125000', '135000', '145000', '155000', '165000', '175000', '185000', '195000', '205000', '215000', '225000', '235000', '245000', '255000', '265000', '275000', '285000', '295000', '320000', '350000', '400000', '450000', '500000', '550000', '600000'],
exchangeEngineSizeFrom(integer):
**Engine size From (l)
=['0.1', '0.2', '0.3', '0.1', '0.2', '0.3', '0.4', '0.5', '0.6', '0.7', '0.8', '0.9', '1.0', '1.1', '1.2', '1.3', '1.4', '1.5', '1.6', '1.7', '1.8', '1.9', '2.0', '2.1', '2.2', '2.3', '2.1', '2.2', '2.3', '2.4', '2.5', '2.6', '2.7', '2.8', '2.9', '3.0', '3.1', '3.2', '3.3', '3.1', '3.2', '3.3', '3.4', '3.5', '3.6', '3.7', '3.8', '3.9', '4.0', '4.1', '4.2', '4.3', '4.1', '4.2', '4.3', '4.4', '4.5', '4.6', '4.7', '4.8', '4.9', '5.0', '5.1', '5.2', '5.3', '5.1', '5.2', '5.3', '5.4', '5.5', '5.6', '5.7', '5.8', '5.9', '5.0', '6.1', '6.2', '6.3', '6.1', '6.2', '6.3', '6.4', '6.5', '6.6', '6.7', '6.8', '6.9', '7.0', '7.1', '7.2', '7.3', '7.1', '7.2', '7.3', '7.4', '7.5', '7.6', '7.7', '7.8', '7.9', '8.0', '8.1', '8.2', '8.3', '8.1', '8.2', '8.3', '8.4', '8.5', '8.6', '8.7', '8.8', '8.9', '9.0', '9.1', '9.2', '9.3', '9.1', '9.2', '9.3', '9.4', '9.5', '9.6', '9.7', '9.8', '9.9', '10.0', '10.1', '10.2', '10.3', '10.1', '10.2', '10.3', '10.4', '10.5', '10.6', '10.7', '10.8', '10.9', '0.0'],
exchangeEngineSizeTo(integer):
**Engine size To (l)
=['0.1', '0.2', '0.3', '0.1', '0.2', '0.3', '0.4', '0.5', '0.6', '0.7', '0.8', '0.9', '1.0', '1.1', '1.2', '1.3', '1.4', '1.5', '1.6', '1.7', '1.8', '1.9', '2.0', '2.1', '2.2', '2.3', '2.1', '2.2', '2.3', '2.4', '2.5', '2.6', '2.7', '2.8', '2.9', '3.0', '3.1', '3.2', '3.3', '3.1', '3.2', '3.3', '3.4', '3.5', '3.6', '3.7', '3.8', '3.9', '4.0', '4.1', '4.2', '4.3', '4.1', '4.2', '4.3', '4.4', '4.5', '4.6', '4.7', '4.8', '4.9', '5.0', '5.1', '5.2', '5.3', '5.1', '5.2', '5.3', '5.4', '5.5', '5.6', '5.7', '5.8', '5.9', '5.0', '6.1', '6.2', '6.3', '6.1', '6.2', '6.3', '6.4', '6.5', '6.6', '6.7', '6.8', '6.9', '7.0', '7.1', '7.2', '7.3', '7.1', '7.2', '7.3', '7.4', '7.5', '7.6', '7.7', '7.8', '7.9', '8.0', '8.1', '8.2', '8.3', '8.1', '8.2', '8.3', '8.4', '8.5', '8.6', '8.7', '8.8', '8.9', '9.0', '9.1', '9.2', '9.3', '9.1', '9.2', '9.3', '9.4', '9.5', '9.6', '9.7', '9.8', '9.9', '10.0', '10.1', '10.2', '10.3', '10.1', '10.2', '10.3', '10.4', '10.5', '10.6', '10.7', '10.8', '10.9', '0.0'],
exchangeFuelType[2](string):
**Car Exchange fuel Type value in array. Pass value for set, otherwise blank.(Seller login only)

=['1'],
exchangeFuelType[5](string):
**Car Exchange fuel Type value in array. Pass value for set, otherwise blank.(Seller login only)
=['1'],
exchangeDriveType[2](string):
**Car Exchange drive Type value in array. Pass value for set, otherwise blank.(Seller login only)
=['1'],
exchangeDriveType[3](string):
Car Exchange drive Type value in array. Pass value for set, otherwise blank.(Seller login only)
=['1']
}
accessories {
fogLights(boolean):
Fog lights / Sumuvalot
absBreaks(boolean):
ABS Breaks / ABS Jarrut
soundSystem(info):
If the vehicle has any sound system / Auton äänentoistojärjestelmä
invalidAccessories(boolean):
Accessories for invalids / Invalidivarustus
airbag(boolean):
Airbag / Turvatyynyt
airconditioning(airconditioning)
Alarm system(boolean):
Alarm system on the car / Auton hälytysjärjestelmä
Alloy wheels(boolean):
Alloy wheels / Erikoisvanteet
Audio input jack(boolean):
Vehicle has audio input jack in sound system / Audioliitäntä äänentoistojärjestelmässä
Auto hold(boolean):
auto hold / Käynnistyksenesto
Cargo safety net(boolean):
Cargo safety net / Tavaratilan turvaverkko
Central locking(Remote Central locking):
Central locking / keskuslukitus
Cruise control(Adaptive cruise control):
Cruise control / Vakionopeudensäädin
Curve lights(boolean):
Curve lights / Kaarrevalot
Driving assistant(boolean):
Driving assistant / Ajonvakautusjärjestelmä
Electric mirrors(boolean):
Electric mirrors / Sähköpeilit

Electric seats(Electric seats with memory):
Electric seats / Sähkösäätöiset istuimet
Electrically operated tailgate(boolean):
Electrically operated tailgate / Sähkötoiminen takaluukku
Fuel heater(boolean):
Fuel heater / Polttoainekäyttöinen lisälämmitin
Heated windshield(boolean):
Heated windshield / Lämmitettävä tuulilasi
Immobilizer(boolean):
Immobilizer / käynnistyksenesto
Internal socket (boolean):
Internal socket / Sisätilanpistoke
Isofix-ready(boolean):
Isofix-ready / Isofix-valmius
Keyless start(boolean):
Keyless start / Avaimeton käynnistys
Leather upholstery(boolean):
Leather upholstery / Nahkasisusta
LED-headlights(boolean):
LED-headlights / LED-ajovalot
Mobilephone equipment(boolean):
Mobilephone equipment / Matkapuhelinvarustus
Motor heater (boolean):
Motor heater / Moottorinlämmitin
Multiactive Steering Wheel(boolean):
Multiactive steering wheel / Monitoimiohjauspyörä
On board computer(boolean):
On board computer / Ajotietokone
Parking Assistant(boolean):
Parking Assistant / Pysäköintiavustin
Parking camera(boolean):
Parking camera / Pysäköintikamera
Parking sensors(boolean):
Parking sensor / Pysäköintitutka
Power steering(boolean):
Power steering / Ohjaustehostin
Power windows(boolean):
Power windows / Sähkökäyttöiset ikkunat
Rain Sensor(boolean,):
Rain Sensor / Sadetunnistin
Roof Opening Mechanism(Automatic Roof Opening Mechanism):
Roof Opening Mechanism / Katon avausmekanismi
Satellite navigator(boolean):
Satellite navigator / Navigointijärjestelmä

Seat heaters(boolean):
Seat heaters / Penkinlämmittimet
Service manual(boolean):
Service manual / Huoltokirja
Ski hatch(boolean):
Ski hatch / Suksiluukku
(boolean):
Rain Sensor / Sadetunnistin
Sport seats(boolean):
Sport seats / Urheiluistuimet
Sport base(boolean):
Sport base / Urheilualusta
Sun hatch(boolean):
Sun hatch / Kattoluukku
Sunroof(boolean):
Sunroof / Lasikatto
Tow bar(boolean):
Tow bar / vetokoukku
Traction control(boolean):
Traction control / Luistonesto
Turbo(boolean):
Turbo
Two sets of tyres(boolean,):
Two sets of tyres / Kahdet renkaat
Xenon lights(boolean):
Xenon lights / Xenon valot
}
info {
info(string):
Lisätietoa stereoista / additional information for stereo
}
airconditioning {
Automatic(boolean):
Tells if the air conditioning is automatic / Automaattinen ilmastointi
}
Remote Central locking {
remote(boolean):
Remote central locking / kaukosäätöinen keskuslukitus
}
Adaptive cruise control {
adaptive(boolean):
Adaptive cruise control / Adaptiivinen vakionopeudensäädin
}
Electric seats with memory {

adaptive(boolean):
Electric seats with memory / Sähkösäätöiset istuimet muistilla
}
Automatic Roof Opening Mechanism {
automatic(boolean):
Automatic Roof Opening Mechanism / Automaattinen katon avausmekanismi
}