jamk.fi

NHL-API

.Net Harjoitustyö

Matti Järvensivu

Harjoitustyö
Joulukuu 2015

Ohjelmistotekniikan koulutusohjelma
Tieto –ja viestintätekniikka

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

jamk.fi

Sisällysluettelo

1 Asennus

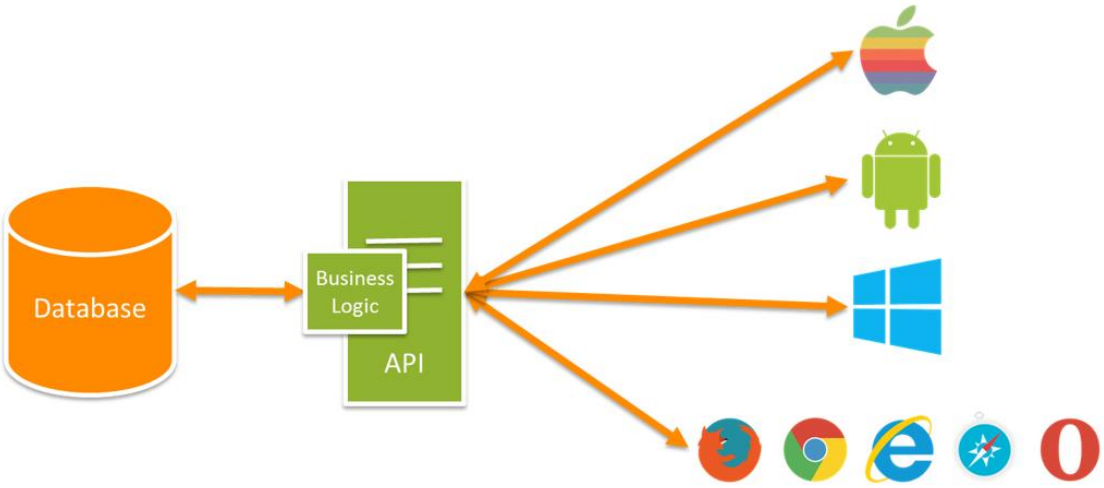
Api olisi hyvä olla julkaistuna jollain palvelimella, jotta apiin olisi helppoa ottaa yhteyttä, tässä tapauksessa ideana oli julkaista se smarterasp.net sivustolle

Pystyttäessä apia täytyy tietää mihin tietokantaan api on yhdistetty. Tässä tapauksessa tietokanta on yhdistetty smarterasp.net sivustolla olevaan kantaan.

Mikäli haluaa muuttaa tietokantayhteyttä täytyy H4102_3Entities yhteyttä muuttaa. Yhteyden pystyy muuttamaan muuttamlla web.config tiedostosta connection stringiä H4102_3Entities.

2 Tietoa

Työ tarkoituksen oli luoda Restfull*(Representional state tranfer)* web api eli ohjelmistorajapinta, joka palauttaa NHL joukkueiden ja pelaajien tietoja. Api on yhteydessä tietokantaan entitien avulla, jonka tietoja voi muuttaa, lisätä ja poistaa. Api hyväksi käyttää HTTP kutsuja esimerkiksi GET, POST, DELETE ja PUT



Julkaistuna Apiin voi ottaa yhteyttä monissa eri laitteissa ja sitä on helppo käyttää, koska logiikka on itse apissa eikä joka laitteessa erikseen.

Web api koostui entity modeleista jotka ovat yhteydessä tietokantaan. Entity modeleilla on myös controllerit jotka ohjaavat HTTP kutsuja ja päättävät mitä niillä tehdään.

Visual studiossa voi generoida valmiiksi oikein hyvän pohjan web apia varten. Modelit saa generoitua suoraan entitien avulla tietokannasta ja controllerit voi luoda modeleista. Niitä täytyy tosin muokata jonkan verran haluamallaan tavalla. Web apissa juotuu todennäköisesti käyttämään Data Transfer Objecteja (DTOs), ristiriitaisten referenssien poistamiseksi.

Esimerkiksi Joukkue koostuu Pelaajista ja pelaaja kuuluu johonkin joukkueseen. Mikäli hakee kaikkia pelaajia api hakee kaikki pelaajat ja joukkueet johon ne kuuluvat ja koska joukkue koostuu pelaajista se hakee ne uudelleen ja näin syntyy niin sanottu referenssi ristiriita ja api ei pysty hakemaan mitään.

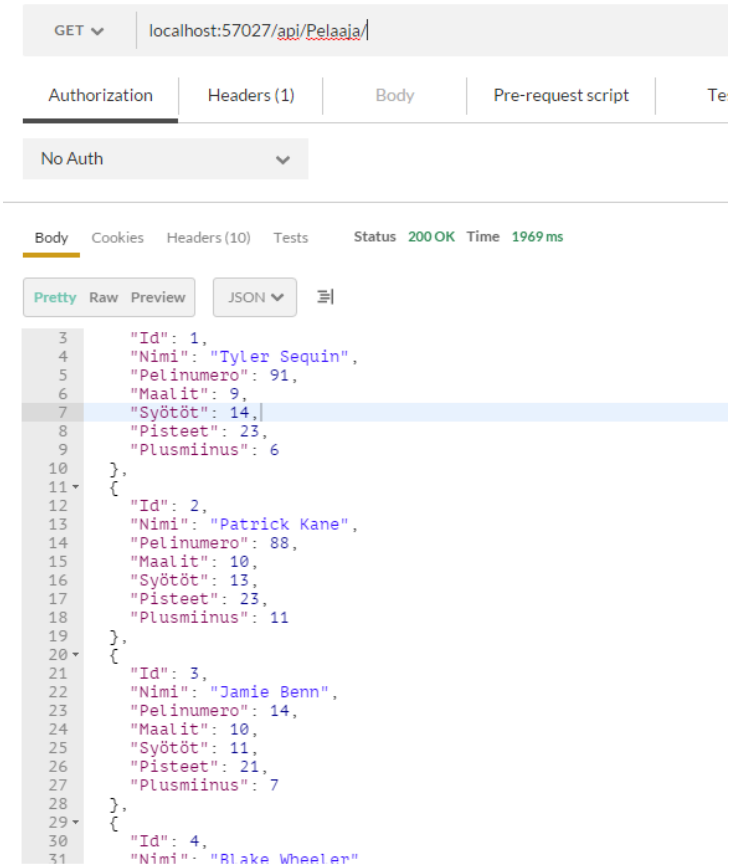
DTO:ita käytetään yksinkertaisesti. Luodaan DTO modellit, jotka suodattavat näytettyä dataa. Kun kutsuu GET lausetta DTO modeliin tallennetaan halutut tiedot ja DTO model palautetaan.

DTOilla voi myös lisätä näytettyä dataa esimerkiksi tässä NHL apin tapauksessa se laskee pisteet. Pisteet siis ei ole tietokannassa vaan se laskee yhteen esimerkiksi maalit ja syötöt ja palauttaa myös sen.

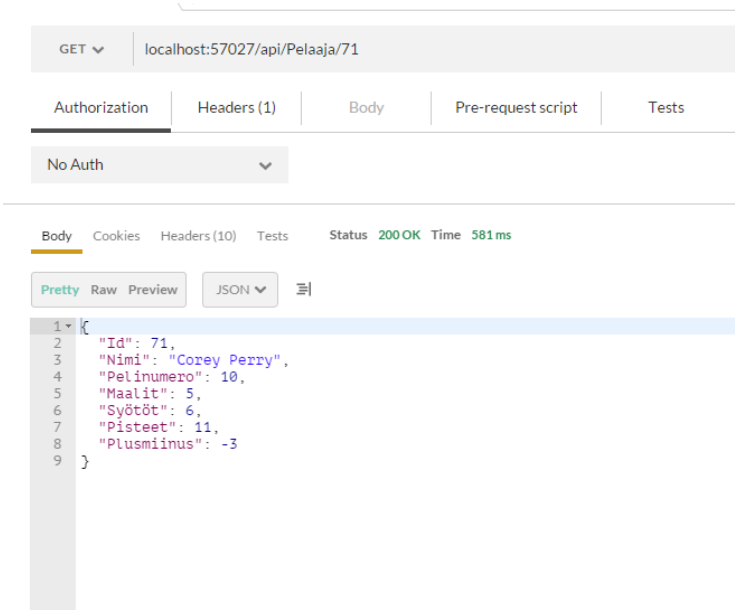
3 Kuvankaappaukset

Käytin apin testaamiseen Google Chromen Postmania.

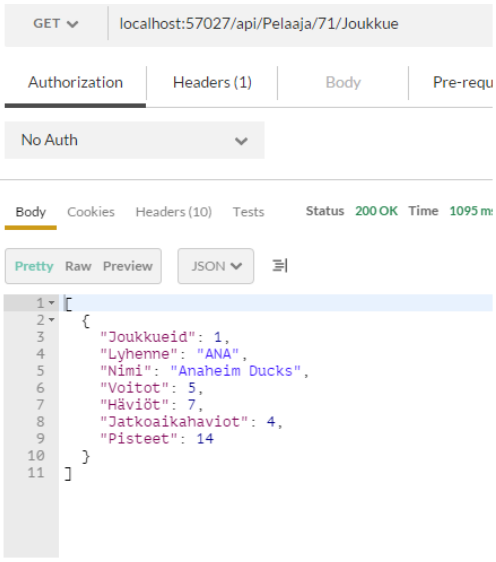
Ohessa kuvankaappauksia apista hakemiseen, lisäämiseen ja poistamiseen.



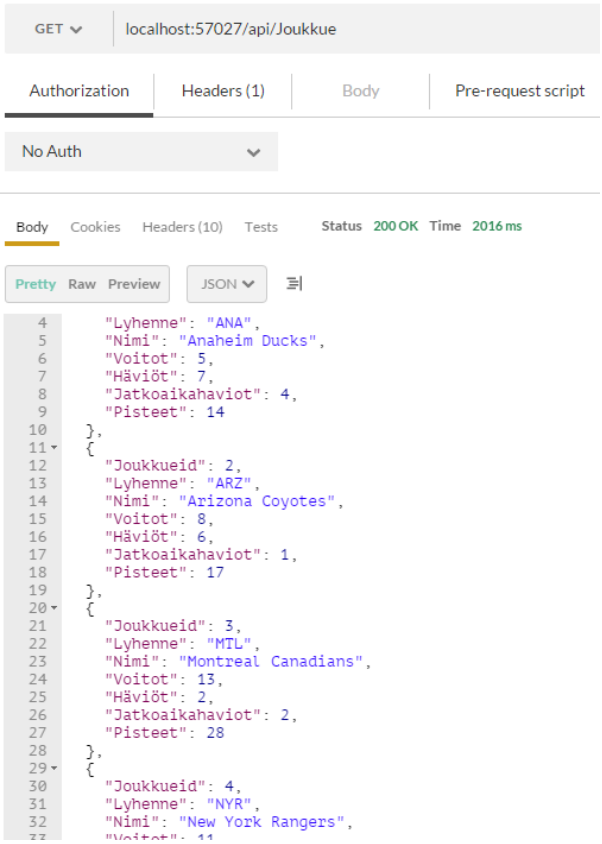
Apissa voi hakea GET:llä kaikki pelaajat /api/Pelaaja haulla



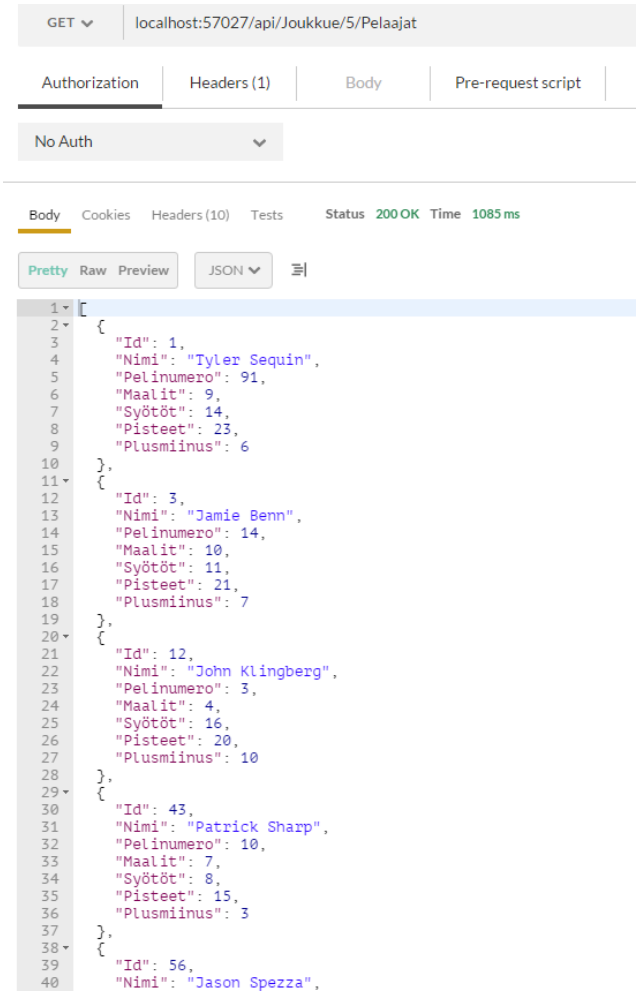
Apissa voi hakea myös id:n perusteella yksittäistä pelaajaa api/Pelaaja/{id}



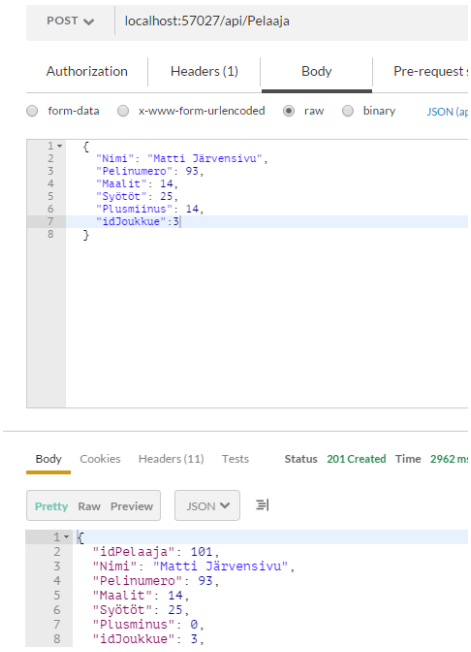
On mahdollista hakea myös pelaajan joukkue syöttämällä /api/Pelaaja/{id}/Joukkue



Samalla tavalla voi hakea kaikki joukkueet komennolla /api/Joukkue



Yksittäisen joukkueen saa sytömmällä api/Joukkue/{id} ja kaikki sen pelaajat syöttämällä api/Joukkue/{id}/Pelaajat



Apiin voi lisätä dataa POST komennolla syöttämällä ensin sijainti joka on /api/Pelaaja ja laittamalla tarvittavat tiedot JSON formaattiin. Data validointia tein sen verran että pelinumeron on oltava väliltä 1-99 ja nimi on pakko syöttää. validointi tapahtuu Modelissa asettamalla tarvittava data [Required] ja jos datan oltava joltain väliltä niin [Range(1,99)]. Muut voivat olla NULL tai oletuksena 0. Id:tä ei tarvi syöttää. Sen tietokanta luo automaattisesti.

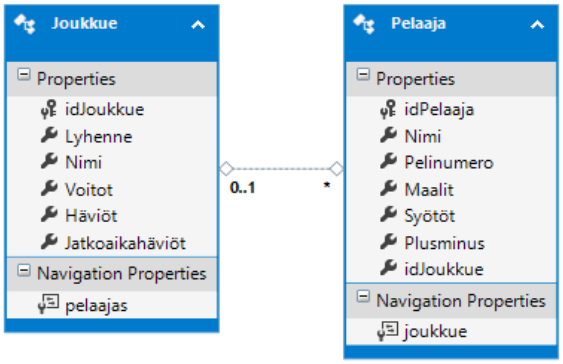
Delete komento toimii samalla tavalla. URLiin /api/Pelaaja/{haluaman pelaajan id} ja se poistaa sen kannasta.

Joukkueiden POST, PUT ja DELETE toimii samalla tavalla

4 Tietokanta

Tietokanta koostuu kohdesta taulusta Joukkue ja Pelaaja. Joukkueella on id, nimi, lyhenne, voitot, häviöt ja jatkoaikahäviöt. Pelaaja tauluun kuuluu id, nimi, pelinumero, maalit, syötöt, plusmiinus ja joukkueid. Joukkuid toimii pelaajataulussa foreign keynä ja referoi Joukkue taulun joukkueid:tä

Apin kanta jota se käyttää löytyy tällä hetkellä Smarterasp.net sivulta. Kanta löytyy myös mysql.labranet palvelimelta, sekä eight palvelimelta



5 Jatkokehitys

Jatkokehittäessä tietokantataulujen, datan ja erilaisten hakujen määrää voi lisätä runsaasti. Esimerkiksi dataan voisi lisätä jäähyt ja vaikka pelaajien kuvat. En saanut julkaistua api palvelimelle jonka kanssa henkilökohtaisesti voin parantaa ja jatkokehittää omaa osaamistani.

6 Opittu

Opin projektissa runsaasti entiteistä, sekä restfull apin toiminnasta ja luonnista. Opin paljon myös HTTP kutsuista ja apien mahdollisuuksista. Opin paljon mitä kaikkea .Net Frameworkin sisällä voikaan tehdä. Harmittamaan jäi kun en saanut apia palvelimelle pystyyn, jonka takia apin demoaminen ei oikein onnistu. Aion vielä kehittää osaamistani tällä kyseisellä osaamisalueella.

7 Työmäärä

Työmääräksi arviosin noin 60 tuntia. Todella paljon arviota enemmän aikaa meni itse api pohjan pystyttämiseen. Tuli todella paljon aikaa vieviä odottamattomia ongelmia. Suurin osa oli sen takia että web apit oli aivan uusi aihe itselle. Eli opettelemiseen meni runsaasti aikaa. Ongelmia tuotti myös että labranettiin ei pystynyt tietyntyyppisiä tietokantayhteyksiä luomaan visual studiolla ja labranetin palvelimen

oikeudet on rajalliset. Näitten odottamattomien ongelmien takia työden tekeminen jäi hieman viime tinkaan.

Olen kuitenkin tyytäväinen kaikkeen mitä opin projektissa vaikka apin pystyttäminen palvelimelle ei onnistunutkaan.