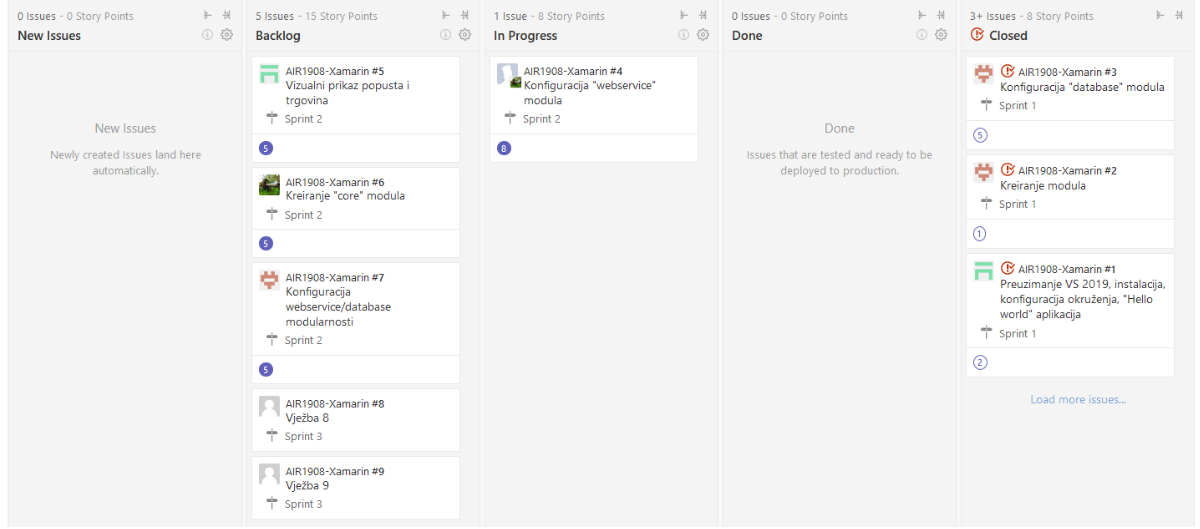
***Sažetak***

U ovoj vježbi konfigurirat ćemo webservice modul. U webservice modul dodat ćemo nekoliko klasa koje će nam služiti za dohvaćanje podataka s webservisa. U ovoj vježbi nećemo imati demonstraciju već ćemo se fokusirati na rad s web servisom. Web servis koji ćemo koristiti nalazi se na sljedećoj adresi: <http://cortex.foi.hr/mtl/courses/air/>.

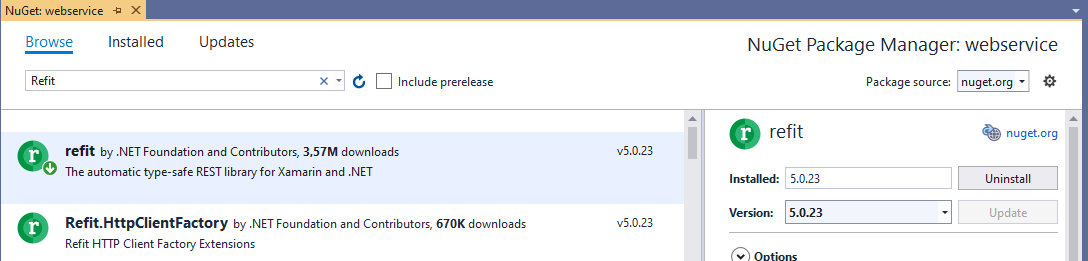
Zadnja verzija aplikacije: Commit 5491509 [<https://github.com/AIR-FOI-HR/AIR1908-Xamarin>]

***Što radimo?***

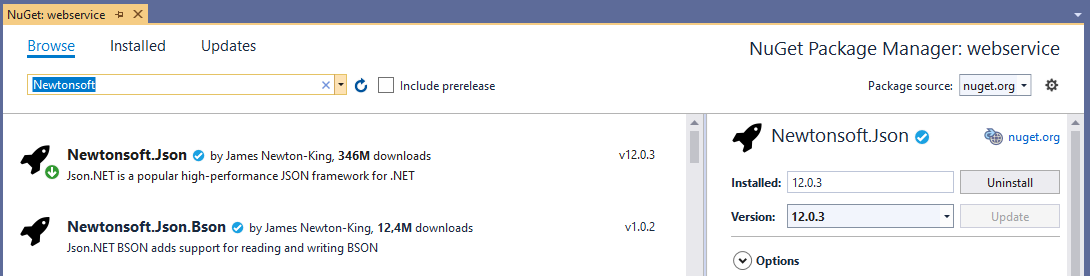


Za početak ćemo dodati dva nova modula: core i webservice. Desni klik na Solution 'DiscountLocator19' > Add > New project > Class Library (.NET Standard). Pod naziv stavit ćemo 'webservice'. Nakon toga kliknemo na 'Create' i stvorit će nam se novi modul. Na isti način kreiramo i core modul.

Nakon što smo kreirali ova dva modula, potrebno je dodati pakete za rad s web servisima. Desni klik na webservice modul > Managet NuGet Packages… Otvorit će nam se NuGet Package Manager za webservice modul. Upišemo 'Refit' u tražilicu i prikazat će nam se željeni paket.

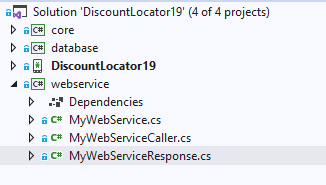


Instaliramo 'refit by .NET Foundation and Contributors. Zatim ćemo u tražilicu upisati 'Newtonsoft'. Prikazat će nam se željeni paket.



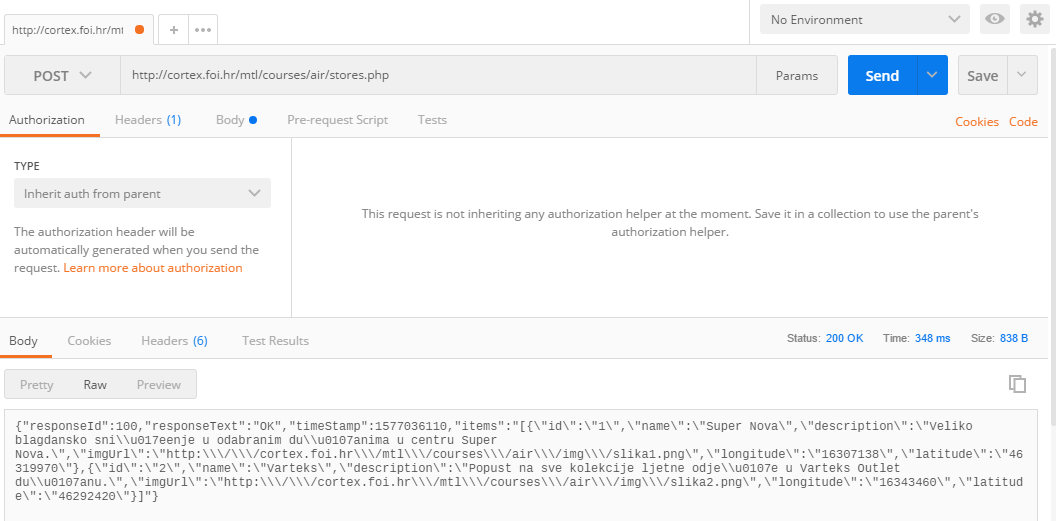
Instaliramo 'Newtonsoft.Json by James Newton-King'. Sada možemo krenuti raditi s web servisima. Dodat ćemo novi interface u webservice modul (desni klik na webservice modul > Add > Class… > Interface > Name: MyWebService).

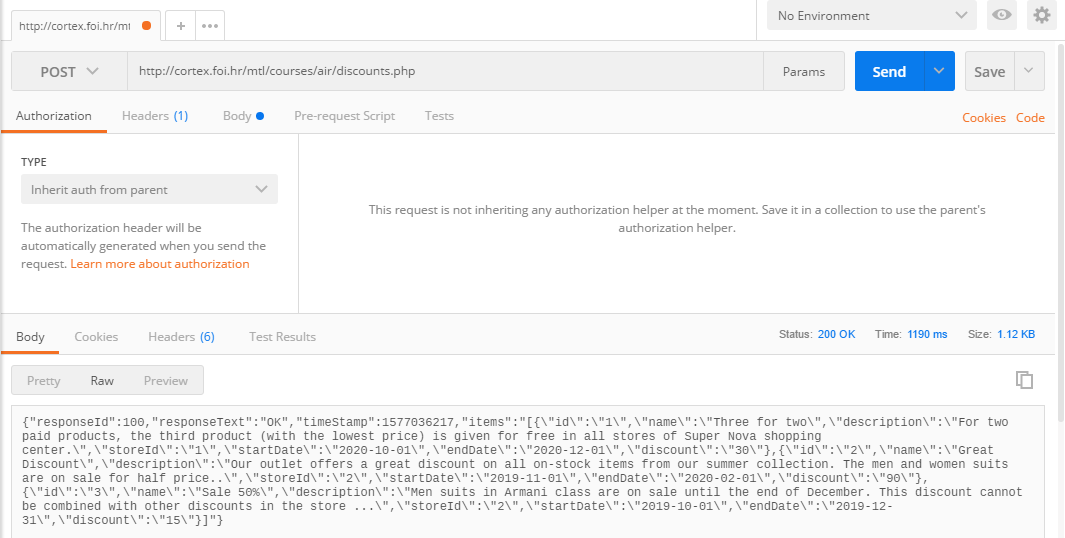
Zatim ćemo dodati dvije nove klase u webservice modul: MyWebServiceCaller i MyWebServiceResponse (desni klik na webservice modul > Add > Class… > Class > Name: …)



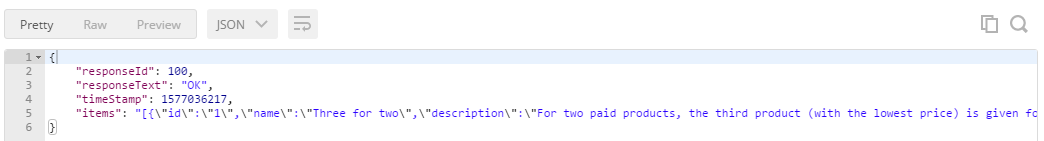
Prije nego što krenemo implementirati klase, testirat ćemo naš web servis korištenjem alata Postman. Postman je alat koji omogućuje testiranje RESTful API-ja na jednostavan i brz način bez potrebe za pisanjem automatskih testova. Alat možete skinuti na sljedećoj adresi: <https://www.getpostman.com/downloads/>.

U prozoru alata unosimo adresu web servisa i metodu dohvaćanja podataka (u našem slučaju koristi ćemo POST metodu). POST metodom podaci se šalju na server (mi ćemo slati jedan tekstualni podatak, a server će nam poslati odgovor). Poslat ćemo dva zahtjeva.

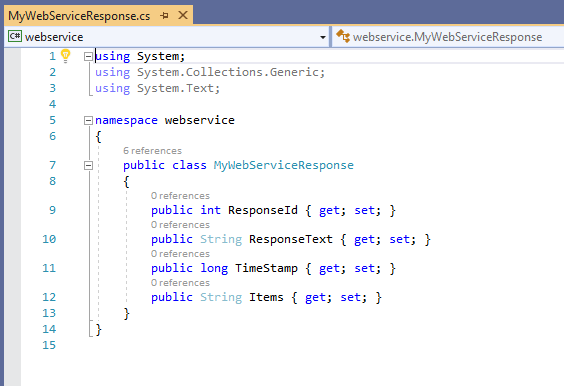




U oba smo slučaja uspješno dobili odgovor od servera. U prvom smo slučaju web servisu slali zahtjev na adresu [http://cortex.foi.hr/mtl/courses/air/**stores.php**](http://cortex.foi.hr/mtl/courses/air/stores.php)**.** U drugom slučaju zahtjev smo slali na adresu: [http://cortex.foi.hr/mtl/courses/air/**discounts.php**](http://cortex.foi.hr/mtl/courses/air/discounts.php). U oba smo slučaja dobili odgovor u JSON formatu. Možemo vidjeti kako se odgovor sastoji od nekoliko dijelova ako u alatu odaberemo 'Pretty' pregled odgovora.

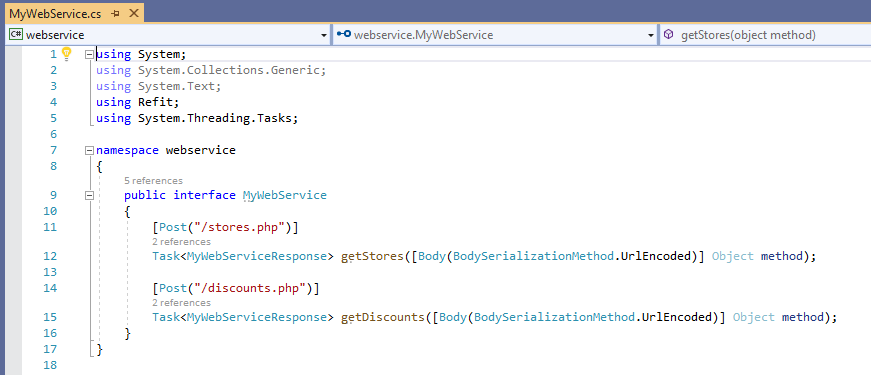


Ovaj će nam pregled služiti za kreiranje atributa klase MyWebServiceResponse.



Od svega što smo dobili u odgovoru od web servisa koristit ćemo samo *items*. Tu su spremljeni podaci koji su nama potrebni, dućani i popusti sa svim atributima.

Sada je potrebno u našoj aplikaciji dodati metode koje će služiti za dohvat podataka. To ćemo pisati u interface-u MyWebService.



Definiramo dvije metode: getStores i getDiscounts. Prva će medota POST metodom dohvatiti dućane, a druga će dohvatiti popuste. U obje funkcije prosljeđujemo parametar method kojeg šaljemo serveru.

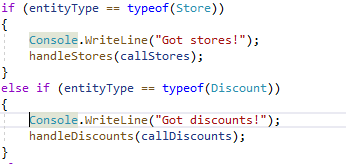
Sada ćemo u klasi MyWebServiceCaller dodati metodu getAll. Ona prima dva parametra, jedan je tipa Dictionary, a drugi je tipa Type. Ova će se metoda pozivati kad budemo htjeli ostvariti komunikaciju s web servisom, a tražit ćemo ili dućane ili popuste. U ovoj klasi definirat ćemo i jednu konstantnu varijablu tipa String koja će sadržavati baznu adresu web servisa.

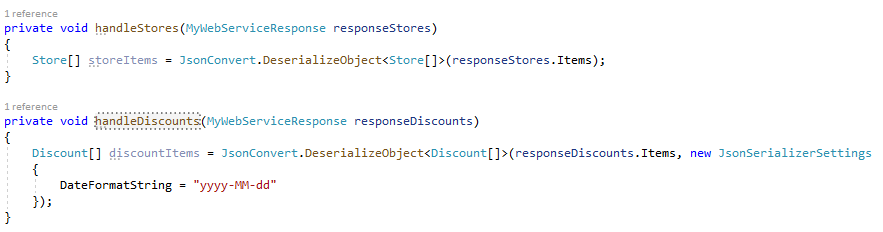


Sada slijedi implementacija naše getAll metode. Metoda For iz Refit biblioteke prima našu baznu adresu web servisa. Metode getStores i getDiscounts pozivaju se asinkrono (ključna riječ await) jer aplikacija ne smije čekati odgovor web servisa, ona mora nastaviti sa izvođenjem neovisno o tome mogu li se dohvatiti podaci ili ne. Nakon što dobijemo odgovor od web servisa, provjeravamo kakvog je tipa varijabla entityType koju smo poslali. Na temelju toga, u konzolu ispisujemo ili 'Got stores!' ili 'Got discounts'.



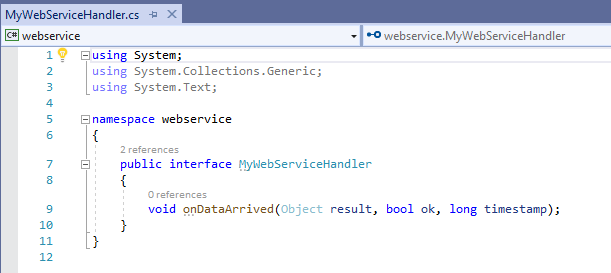
Na redu je deserijalizacija JSON odgovora. U MyWebServiceCaller klasi dodat ćemo dvije metode: handleStores i handleDiscounts. U svakoj ćemo deserijalizirati odgovor koji dobijemo. U našoj getAll metodi, nakon provjere tipa varijable entityType, pozvat ćemo ove dvije metode. Metode primaju parametar tipa MyWebServiceResponse.





Na gore prikazanoj slici možemo vidjeti implementaciju obiju metoda. Metoda handleStores deserijalizirani odgovor web servisa sprema u listu tipa Store (u primjeru s alatom Postman vidjeli smo da se u *items* dijelu odgovora nalaze potrebni podaci). Metoda handleDiscounts deserijalizirani odgovor sprema u listu tipa Discount.

Sada ćemo dodati novi interface. Desni klik na webservice modul > Add > Class… > Interface. Nazvat ćemo ga MyWebServiceHandler. Sadržavat će jednu metodu onDataArrived koja prima tri parametra.



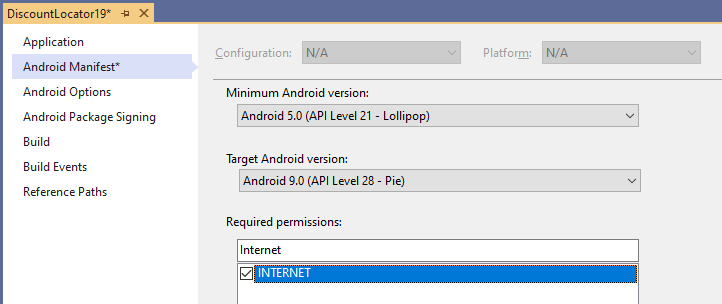
U MyWebServiceCaller klasu dodat ćemo novi konstruktor koji prima MyWebServiceHandler kao parametar.



U istoj klasi, u metodama handleStores i handleDiscounts, dodat ćemo pozive onDataArrived metode iz interface-a MyWebServiceHandler. Proslijedit ćemo liste dućana i popusta, 'true' vrijednost i timeStamp iz odgovora od web servisa. Pozivanjem te metode postavljamo događaj koji će proslijediti podatke u parametrima.



Za kraj, prije testiranja, potrebno je dodati Internet konekciju. Desni klik na DiscountLocator19 > Properties > Android Manifest > Required permissions: Internet.



Nakon što smo ovo napravili, potrebno je napraviti Rebuild projekta. Kartica Build > Rebuild Solution. U sljedećoj vježbi konfigurirat ćemo core modul.