1. Uvod u mikroservisne aplikacije

Tema ovih časova je upoznavanje studenata sa planom kursa i obavezama na kursu, kao i razvijanje jednostavnog mikroservisa sa MongoDB SUBP.

1. O kursu

- Sajt kursa: http://rs2.matf.bg.ac.rs/
- Prezentacija: http://rs2.matf.bg.ac.rs/vezbe/o-kursu.pdf
- O seminarskim radovima: http://rs2.matf.bg.ac.rs/seminarski-radovi/
- Neophodni alati:
    - Visual Studio (Windows), Visual Studio for Mac (OSX), Visual Studio Code (Windows, OSX, Linux), Jetbrains Rider (Windows, OSX, Linux)
    - .NET 5 (Windows, OSX, Linux)
    - Docker Desktop (Windows, OSX), Docker Server (Linux)

2. O mikroservisnim aplikacijama

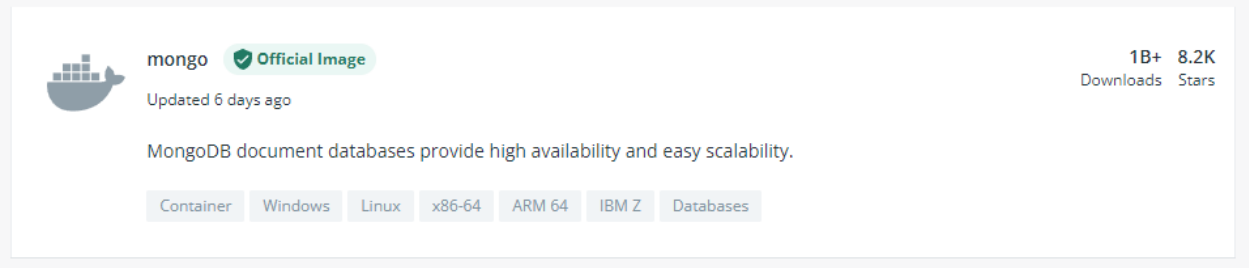
- Ukratko proći kroz tekst: http://rs2.matf.bg.ac.rs/vezbe/ukratko-o-mikroservisima.pdf

3. Razvoj Catalog.API mikroservisa

- Priprema MongoDB kontejnera
- Kreiranje projekta i instalacija paketa
- Razvoj mikroservisa
- Pokretanje i debagiranje aplikacije

Priprema MongoDB kontejnera

Na stranici https://hub.docker.com/ uneti „mongo“ u polje za pretragu. Otvoriti sledeću stranicu:



Osnovni MongoDB pojmovi: baza dokumenata, dokumenti, jedinstveni ključevi (_id), kolekcije, indeksi (nad _id).

Pokretanje mongo kontejnera:

- Otvoriti Powershell
- **docker run --name mongo_catalog -p 27017:27017 -d mongo**

Ove naredbe ispisuju identifikator kontejnera u kojem je pokrenut MongoDB SUBP. Korisne docker naredbe:

- Ispisuje sve pokrenute kontejnere
    - **docker ps**
- Ispisuje sve kontejnere
    - **docker ps -a**
- Ispisuje samo identifikatore pokrenutih kontejnera

    - **docker ps -q**
- Pokreće ugašeni kontejner
    - **docker start IDENTIFIKATOR_KONTEJNERA**
- Zaustavlja pokrenuti kontejner
    - **docker stop IDENTIFIKATOR_KONTEJNERA**
- Uklanja kontejner
    - **docker rm IDENTIFIKATOR_KONTEJNERA**
- Uklanja sve kontejnere sa sistema
    - **docker rm $(docker ps -aq)**
- Kreira novi kontejner na osnovu slike čiji je naziv **IME_SLIKE** na hub.docker.com i pokreće ga
    - **docker run IME_SLIKE**
- Kreira novi kontejner na osnovu slike i pokreće ga, pri čemu mu daje ime **IME_KONTEJNERA**
    - **docker run --name IME_KONTEJNERA IME_SLIKE**
- Kreira novi kontejner na osnovu slike i pokreće ga, pri čemu se vrši preslikavanje portova, tako što se **UNUTRAŠNJI_PORT** preslikava na **SPOLJNI_PORT** u localhost-u
    - **docker run -p SPOLJNI_PORT:UNUTRAŠNJI_PORT IME_SLIKE**
- Kreira novi kontejner na osnovu slike i pokreće ga, ali u pozadini (terminal se ne blokira)
    - **docker run -d IME_SLIKE**
- Čita (i prati, ako se navede opcija **-f**) sve logove za pokrenuti kontejner
    - **docker logs -f IME_KONTEJNERA**
- Pokreće interaktivni terminal u kontejneru. Ovo je korisno za izvršavanje proizvoljnih naredba
    - **docker exec -it IME_KONTEJNERA /bin/bash**

Kada se prikačimo za mongo kontejner, dostupan nam je CLI alat **mongo** kojim možemo izvršavati proizvoljne naredbe za upravljanje mongo bazom. Neke osnovne komande ovog alata su:

- Prikazuje sve baze podataka
    - **show dbs**
- Bira bazu podataka **BAZA_PODATAKA** za koju će se odnositi sve dalje naredbe (odabrana BP biće dostupna kroz objekat **db**)
    - **use BAZA_PODATAKA**
- Čitanje svih dokumenata iz kolekcije **IME_KOLEKCIJE**
    - **db.IME_KOLEKCIJE.find({})**
- Unošenje novog dokumenta u kolekciju **IME_KOLEKCIJE**
    - **db.IME_KOLEKCIJE.insertOne({ name: 'Pera', prezime: 'Perić' })**
- Ažuriranje
    - **db.IME_KOLEKCIJE.updateOne({ name: 'Pera' }, { $set: { izmenjen: true } })**
- Brisanje
    - **db.IME_KOLEKCIJE.deleteOne({ name: 'Pera' })**

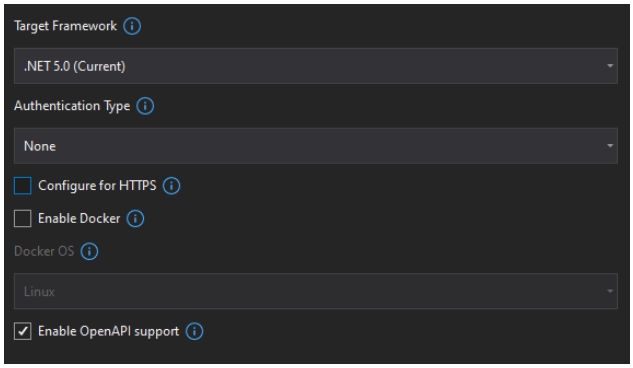
Kreiranje projekta i instaliranje paketa

S obzirom da započinjemo razvoj „od nule“, potrebno je prvo da napravimo jedan *Solution* pre nego što kreiramo bilo koji projekat:

- File > New > Project
- Blank Solution
- Popuniti neophodnim podacima:
    - Solution name: **Webstore**
    - Location: **LOKACIJA_LOKALNOG_REPOZITORIJUMA**
    - Solution: **Create new solution**
- Create

Sada možemo da kreiramo nove projekte:

- Desni klik na ime *Solution-a*
- Add > New Project
- ASP.NET Core Web API
- Popuniti neophodnim podacima:
  - Project name: **Catalog.API**
  - Location: **LOKACIJA_LOKALNOG_REPOZITORIJUMA\Services\Catalog**
- Next
- Odabrati opcije kao na slici pored:
- Create



Pre nego što krenemo sa razvojem, potrebno je da instaliramo neophodne pakete. Otvoriti *NuGet Package Manager* sledećim koracima:

- Desni klik na naziv projekta
- Manage NuGet Packages

Prvo ćemo ažurirati sve pakete koji su do sada instalirani:

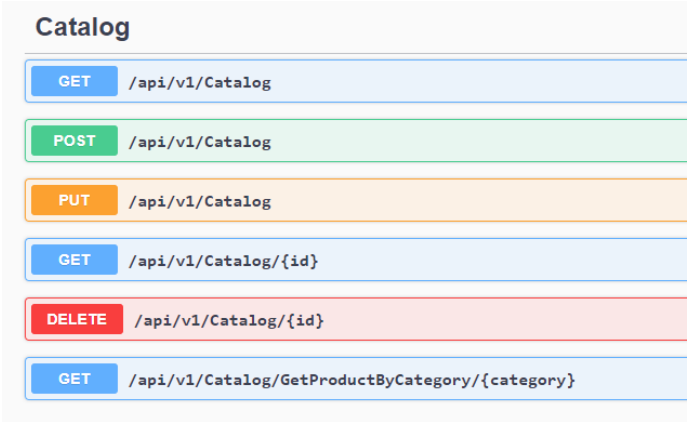
- Updates
- Select all packages
- Update

Zatim je potrebno otvoriti tab „Browse“ i tu pronaći i instalirati sledeće pakete:

- MongoDB.Driver

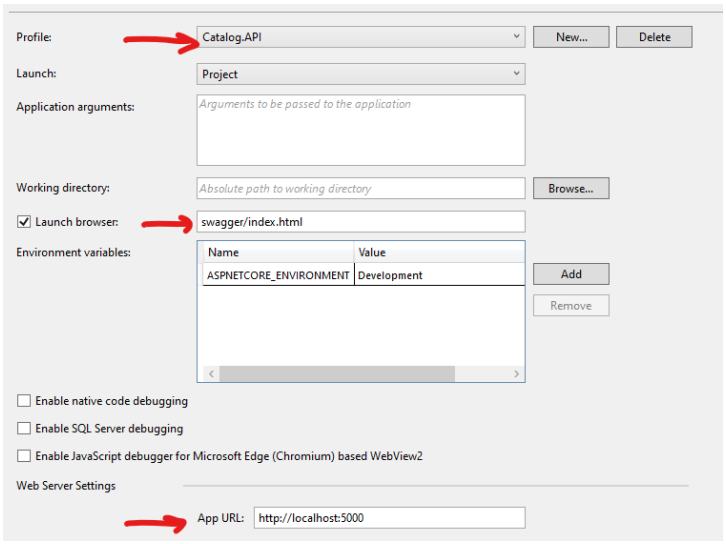
Razvoj mikroservisa

API mikroservisa je opisan slikom pored.



Prvo podešavamo opcije za pokretanje aplikacije:

- Desni klik na naziv projekta
- Properties
- Debug
- Podesiti opcije sa slike ispod.



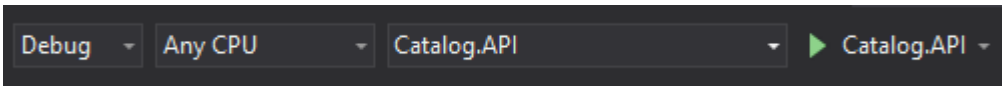
Zatim prelazimo na kodiranje, prema narednom redosledu:

- Entities
  - Product.cs
- Data
  - ICatalogContext.cs
  - CatalogContext.cs
  - CatalogContextSeed.cs
- Repositories
  - IProductRepository.cs
  - ProductRepository.cs
- Controllers
  - CatalogController.cs

Sada je preostalo da dodamo ubrizgavanje zavisnosti, što se nalazi u **Startup.cs** datoteci.

Pokretanje i debagiranje aplikacije

Iz gornjeg menija odabrati naredne opcije i pokrenuti aplikaciju:



Postaviti u nekom zahtevu tačku prekida i prolaziti kroz kod. Prikazivati Visual Studio okruženje za debagiranje.

2. Višestruki mikroservisi. Kontejnerizacija aplikacije.

Tema ovih časova je rad sa Redis, distrubuiranom keš memorijom i kontejnerizacija projekata pomoću Docker i Docker Compose alata.

1. Razvoj Basket.API mikroservisa

Priprema Redis kontejnera

Na stranici https://hub.docker.com/ uneti „redis“ u polje za pretragu. Otvoriti sledeću stranicu:

Osnovni Redis pojmovi: baza ključ-vrednost, keš memorija, prednosti i ograničenja (https://redis.io/topics/faq).

Pokretanje redis kontejnera:

- Otvoriti Powershell
- **docker run -d -p 6379:6379 --name redis_basket redis**

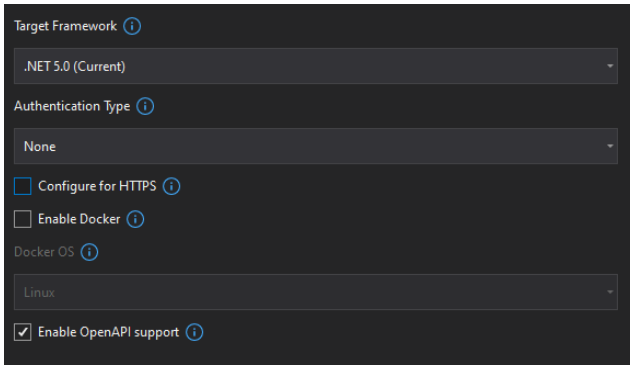
Kada se prikačimo za redis kontejner, dostupan nam je alat **redis-cli** kojim možemo izvršavati proizvoljne naredbe za upravljanje redis bazom. Neke osnovne komande ovog alata su:

- Provera da li je baza spremna (očekuje se odgovor PONG)
  - ping
- Postavljanje vrednosti
  - set **KLJUČ VREDNOST**
- Čitanje vrednosti
  - get **KLJUČ**

Kreiranje projekta i instaliranje paketa

Dodajemo novi projekat u okviru već napravljenog *Solution-a*:

- Desni klik na ime *Solution-a*
- Add > New Project
- ASP.NET Core Web API
- Popuniti neophodnim podacima:
  - Project name: **Basket.API**
  - Location: **LOKACIJA_LOKALNOG_REPOZITORIJUMA\Services\Basket**
- Next
- Odabrati opcije kao na slici pored.
- Create



Pre nego što krenemo sa razvojem, potrebno je da instaliramo neophodne pakete. Otvoriti *NuGet Package Manager* sledećim koracima:

- Desni klik na naziv projekta
- Manage NuGet Packages

Prvo ćemo ažurirati sve pakete koji su do sada instalirani:

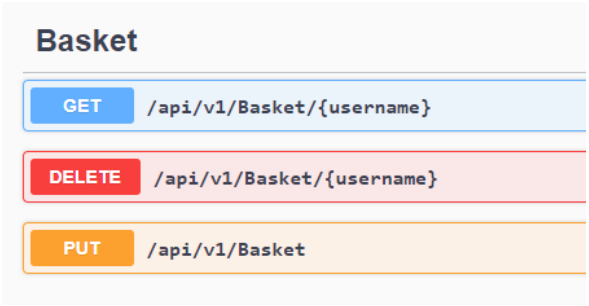
- Updates
- Select all packages
- Update

Zatim je potrebno otvoriti tab „Browse“ i tu pronaći i instalirati sledeće pakete:

- Newtonsoft.Json
- Microsoft.Extensions.Caching.StackExchangeRedis

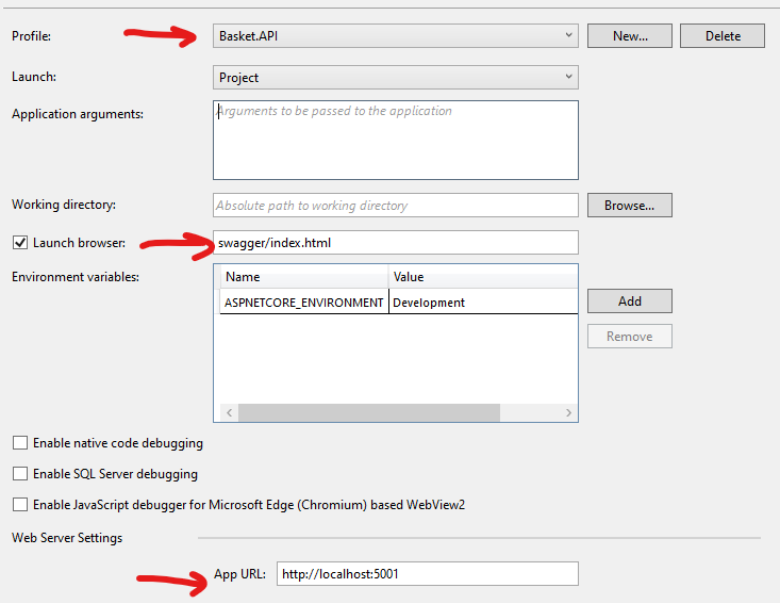
Razvoj mikroservisa

API mikroservisa je opisan slikom pored.



Prvo podešavamo opcije za pokretanje aplikacije:

- Desni klik na naziv projekta
- Properties
- Debug
- Podesiti opcije sa slike ispod.



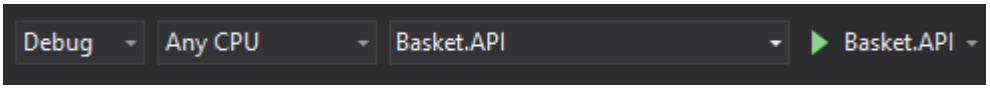
Zatim prelazimo na kodiranje, prema narednom redosledu:

- Entities
  - ShoppingCartItem.cs
  - ShoppingCart.cs
- Repositories
  - IBasketRepository.cs
  - BasketRepository.cs
- Controllers
  - BasketController.cs

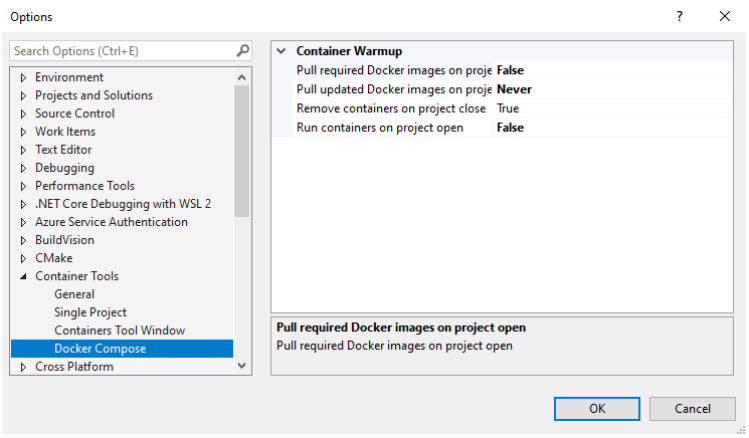
Sada je preostalo da dodamo ubrizgavanje zavisnosti, što se nalazi u **Startup.cs** datoteci.

Pokretanje i debagiranje aplikacije

Iz gornjeg menija odabrati naredne opcije i pokrenuti aplikaciju:



2. Kontejnerizacija projekata



Pre nego što bilo šta uradimo, preporuka je da postavimo naredne opcije u Visual Studio alatu, kako bismo izbegli neka suvišna pokretanja Docker Compose alata:

- Tools > Options
- Otvoriti Container Tools grupu opcija
- Odabrati opciju Docker Compose
- Odabrati opcije sa naredne slike:

Dodavanje podrške za Docker Compose projektu:

- Desni klik na naziv projekta
- Add > Container Orchestrator Support
- Docker Compose
- Ok
- Linux
- Ok

Proći kroz generisani **Dockerfile** i objasniti neke najvažnije elemente, a zatim objasniti **docker-compose.yml** i **docker-compose.override.yml** datoteke.

Dodati naredne resurse u **docker-compose.yml** datoteku:

```
services:
 catalogdb:
   image: mongo
 basketdb:
   image: redis:alpine

volumes:
 mongo_data:
```

Dodati naredne resurse u **docker-compose.override.yml** datoteku:

```
services:
 catalogdb:
   container_name: catalogdb
   restart: always
   ports:
     - "27017:27017"
   volumes:
     - mongo_data:/data/db

 basketdb:
   container_name: basketdb
   restart: always
```

```
  ports:
    - "6379:6379"

  catalog.api:
    container_name: catalog.api
    environment:
      - ASPNETCORE_ENVIRONMENT=Development
      - "DatabaseSettings:ConnectionString=mongodb://catalogdb:27017"
    depends_on:
      - catalogdb
    ports:
      - "8000:80"

  basket.api:
    container_name: basket.api
    environment:
      - ASPNETCORE_ENVIRONMENT=Development
      - "CacheSettings:ConnectionString=basketdb:6379"
    depends_on:
      - basketdb
    ports:
      - "8001:80"
```

Pokretanje projekta iz komandne linije

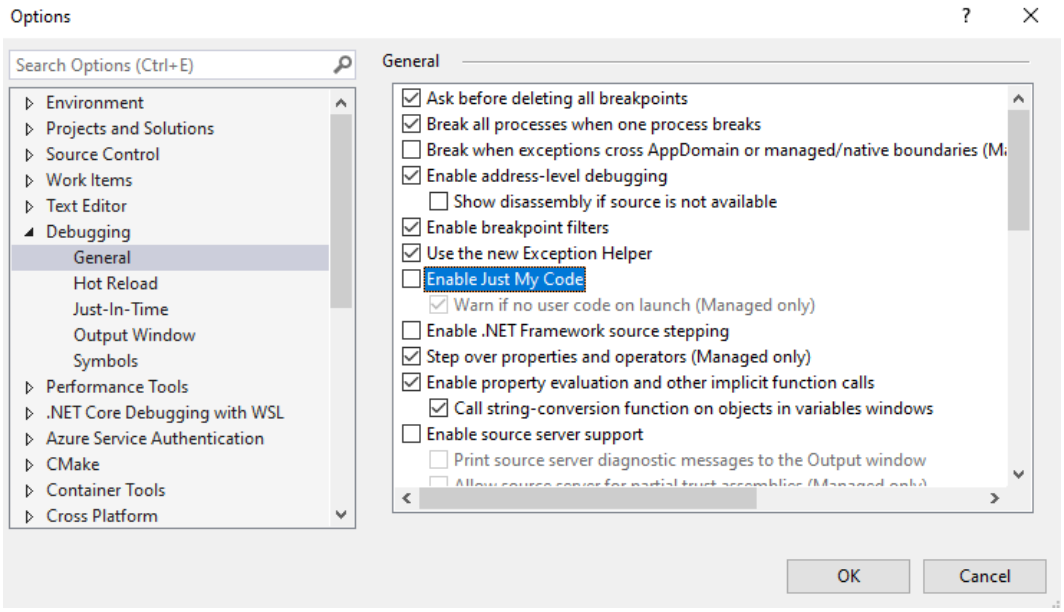
Pokretanje kontejnera iz komandne linije se vrši alatom **docker-compose** koja ima nekoliko važnih komandi (opcija **-d** označava da će se proces nastaviti u pozadini, kako se ne bi blokirao terminal):

- Izgradnja svih kontejnera
  - **docker-compose build**
- Podizanje svih kontejnera
  - **docker-compose up -d**
- Izgradnja i podizanje svih kontejnera
  - **docker-compose up --build -d**
- Zaustavljanje svih kontejnera
  - **docker-compose stop**
- Zaustavljanje i uklanjanje svih kontejnera
  - **docker-compose down**
- Specifikovanje datoteka koje se koriste za podizanje/spuštanje svih kontejnera
  - **docker-compose -f docker-compose.yml -f docker-compose.override.yml up --build -d**
  - **docker-compose -f docker-compose.yml -f docker-compose.override.yml**

Debagiranje projekata u kontejneru

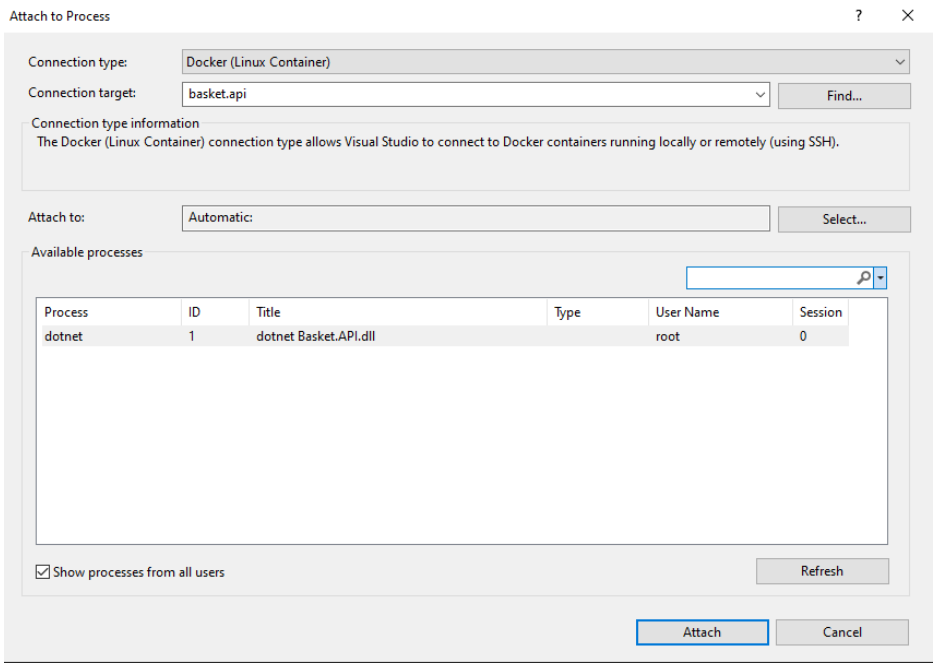
Kako bismo omogućili debagiranje projekata koji se pokreću u kontejneru, potrebno je da ručno *zakačimo* debager za proces koji se izvršava u kontejneru. Pre toga, potrebno je isključiti opciju „Enable Just My Code“ u podešavanjima, kako bismo instruisali Visual Studio da debagira i kod koji je izgrađen u kontejnerima:

- Tools > Options
- Debugging > General
- Isključiti opciju „Enable Just My Code“, kao na slici ispod



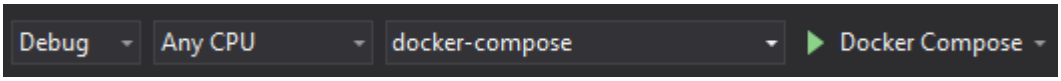
Sada možemo zakačiti debager:

- Debug > Attach to Process
- Za „Connection type“ odabrati Docker (Linux Container)
- Odabrati dugme Find
- Nakon nekoliko sekundi bi trebalo da se pojavi spisak svih podignutih kontejnera. Odabrati, na primer, basket.api projekat, pa dugme Ok
- U tabeli „Available processes“ bi trebalo da se pojavi „dotnet“ proces, kao na slici ispod
- Odabrati taj proces, pa dugme Attach
- Odabrati opciju „Managed (.NET Core for Unix)“, pa dugme Ok
- Posle nekoliko sekundi, debager će biti *zakačen* za proces



Pokretanje projekta iz Visual Studio alata

Nakon što smo napravili **docker-compose** projekat, potrebno je da odaberemo naredne opcije u glavnom meniju:



Klikom na pokretanje se vrše naredne akcije:

- Izgrađuje se kod iz *Solution-a*
- Izgrađuju se kontejneri
- Pokreću se kontejneri
- Pokreće se debager i automatski se zakači za izvršni kod

Neke napomene

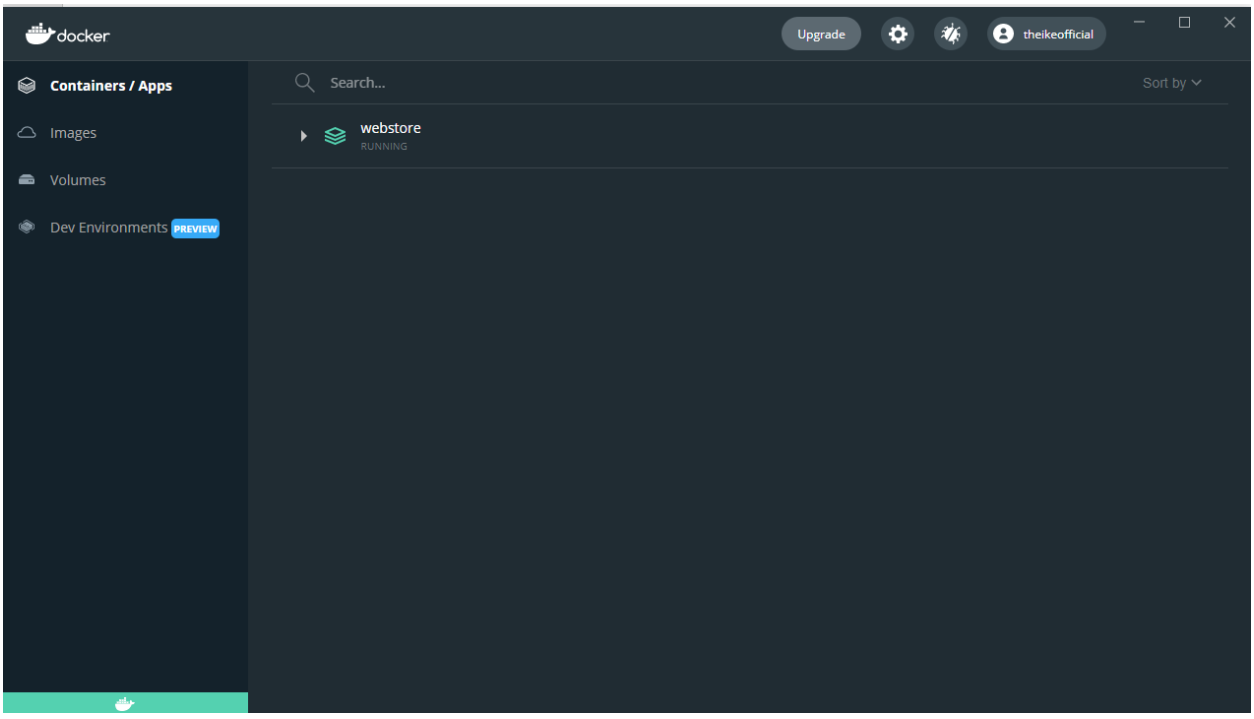
- Prednosti korišćenja su automatsko podizanje i automatsko debagiranje.
- Gašenjem projekta (bilo iz Visual Studio alata, gašenjem pregledača ili gašenjem terminala) se ne gase kontejneri, ali se ne mogu ugasiti iz **Docker Desktopa**, pa je neophodno uraditi **Build > Clean solution** iz glavnog menija – obavezno pre zatvaranja Visual Studio alata.
- Treba imati u vidu da će se konfiguracija prvo pročitati iz **appsettings.json** i **appsettings.Development.json** datoteka nego iz **docker-compose.yml** datoteka. Preporuka je da se prepišu sve promenljive okruženja iz Yaml datoteka u json pre pokretanja. Naravno, ovime se onemogućava pojedinačno pokretanje projekata iz Visual Studio alata.
- Debagiranje u kontejnerima je nešto sporije u odnosu na debagiranje aplikacija koje su pokrenute na host računaru, ali funkcioniše identično.

Docker Desktop

Alat Docker Desktop nam služi za upravljanje kontejnerima iz grafičke korisničke aplikacije. Na narednoj slici možemo videti prikaz nakon pokretanja. Sa strane možemo birati neke od tabova koji nam daju uvid u naredne elemente za rad sa kontejnerima:

- **Containers/Apps** prikazuje pregled svih kontejnera koji postoje na sistemu
- **Images** prikazuje pregled svih slika koji su dovučeni na sistemu
- **Volumes** prikazuje pregled svih „diskova" koje kontejneri koriste za trajno skladištenje datoteka
- **Dev Environments** prikazuje pregled okruženja za razvoj za jednostavnu kolaboraciju razvijalaca softvera u timu

Nama će najznačajniji biti prvi pregled.



Kontejneri mogu biti pokrenuti pojedinačno, ili kao deo neke mreže, tj. orkestra kontejnera. Na slici ispod je prikazana orkestrizacija celokupne aplikacija *Webstore* koja se, u ovom trenutku, sastoji od četiri kontejnera. Sa

desne strane možemo videti zajedničke dnevnike, a klikom na konkretan kontejner, biće nam prikazan dnevnik samo za taj kontejner.