Informacije:
Indeks: 16533
Student: Milos Denic
Kontakt: dmilos@elfak.rs

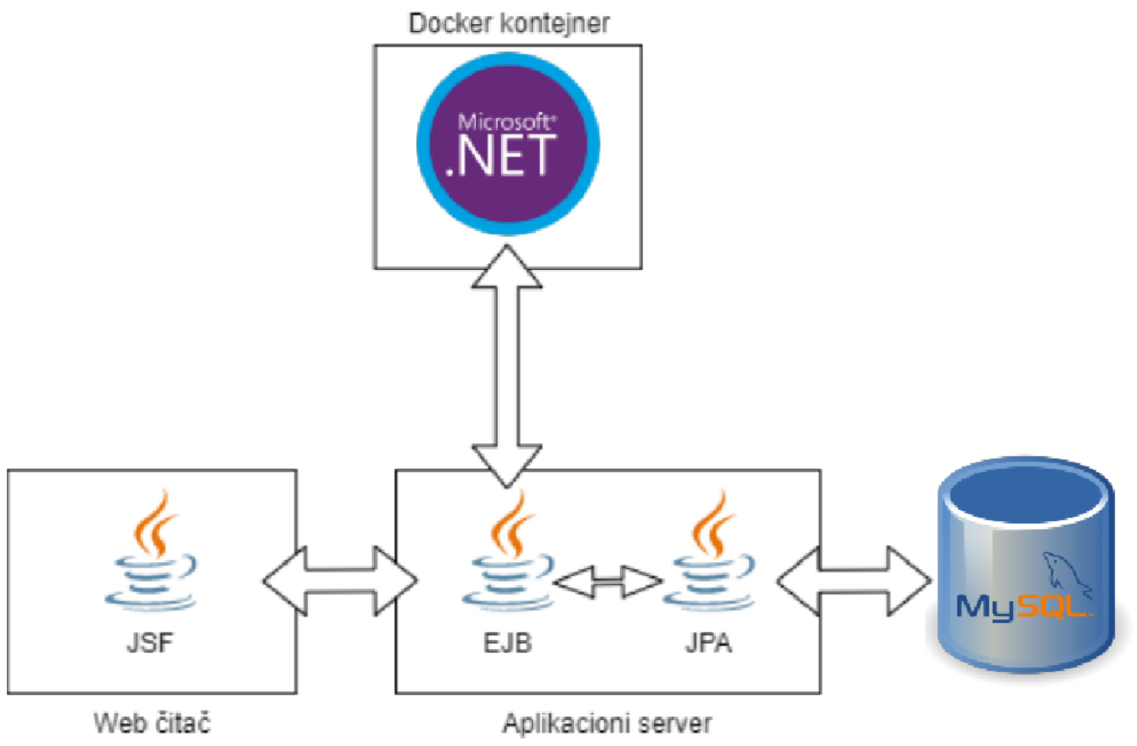
Zadatak

zad-31.txt

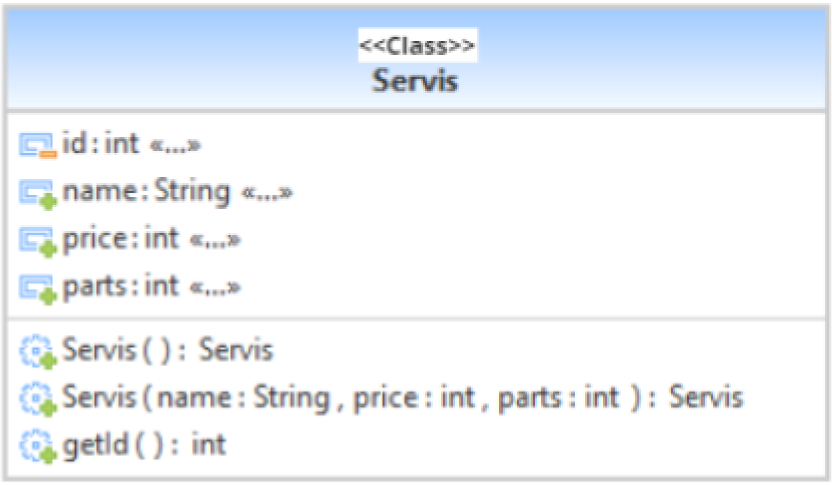
Tekst: 14. Informacioni sistem servisa za automobile. Cuva se evidencija automobila primljenih na servis, dostupnih usluga i delova (sa cenam), ali dijagnosticih informacija za automobile. Za svaki automobil se cuva informacija o modelu, boji, broj licne karte vlasnika, datum prijema. Dijagnostika se odnosi na uocene probleme i njihov opis. Nakon dolaska u servis, zavisno od uocenih problema, primenjuju se servisne usluge (popravke i slicno), ali može biti potrebna i nabavka novih delova. Omoguciti generisanje obacuna usluge (broj licne karte, datum kada je završena popravka i ukupna cena) na osnovu dostavljenih usluga i cena, koji se takode trajno cuva pored prethodno navedenih evidencija.

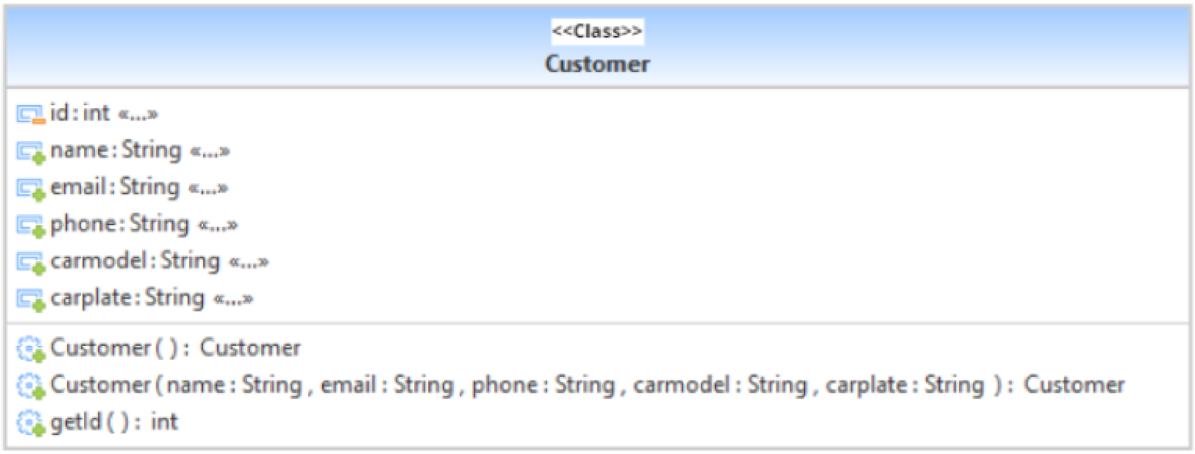
Obrazac: dekompozicija poslovnog cilja

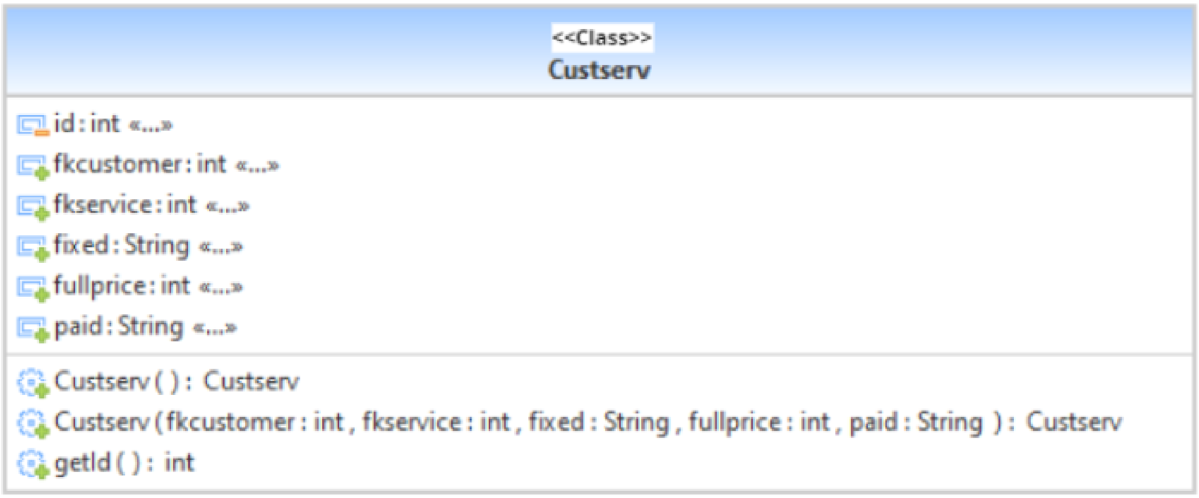
Arhitektura projekta

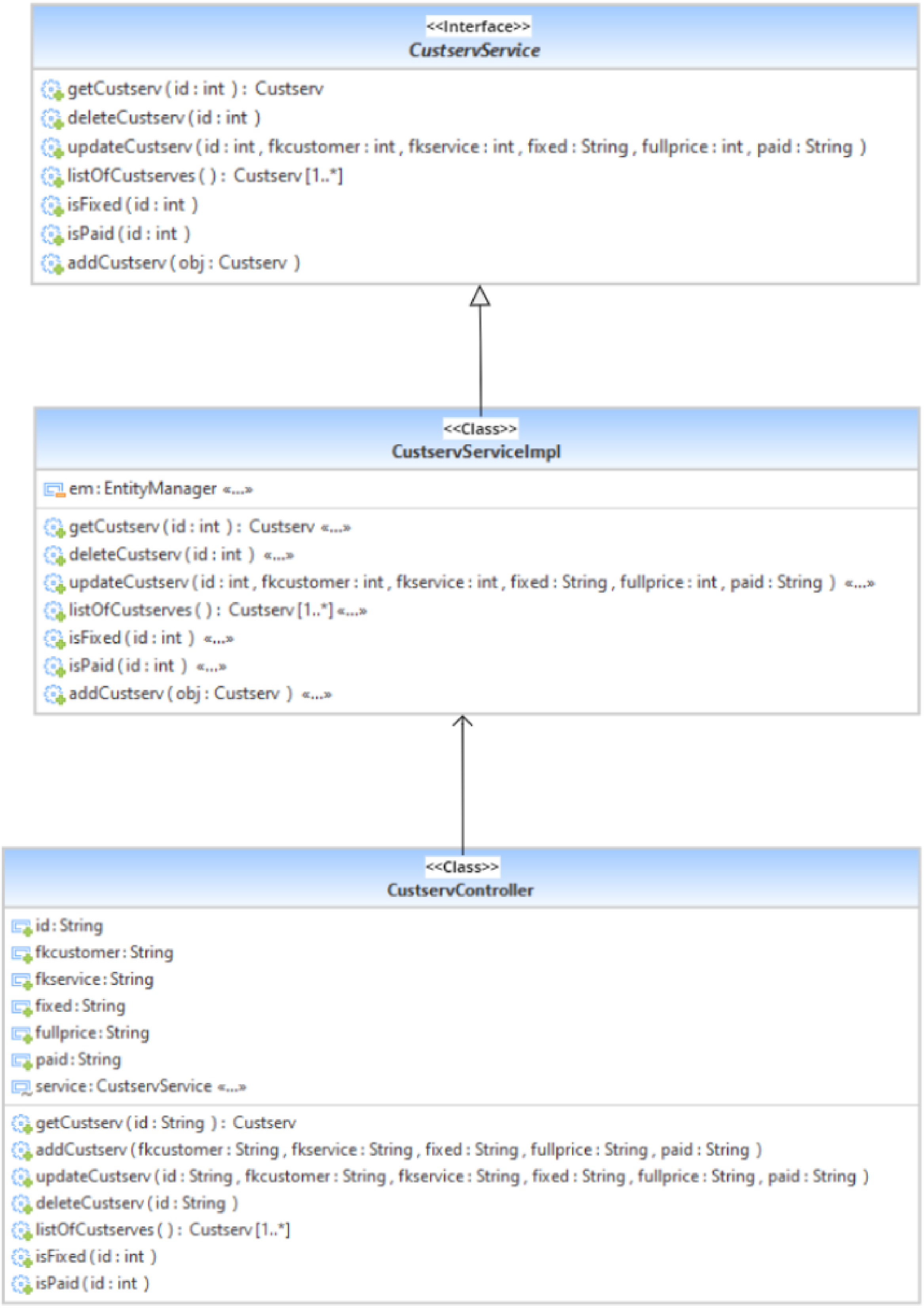


Class dijagram

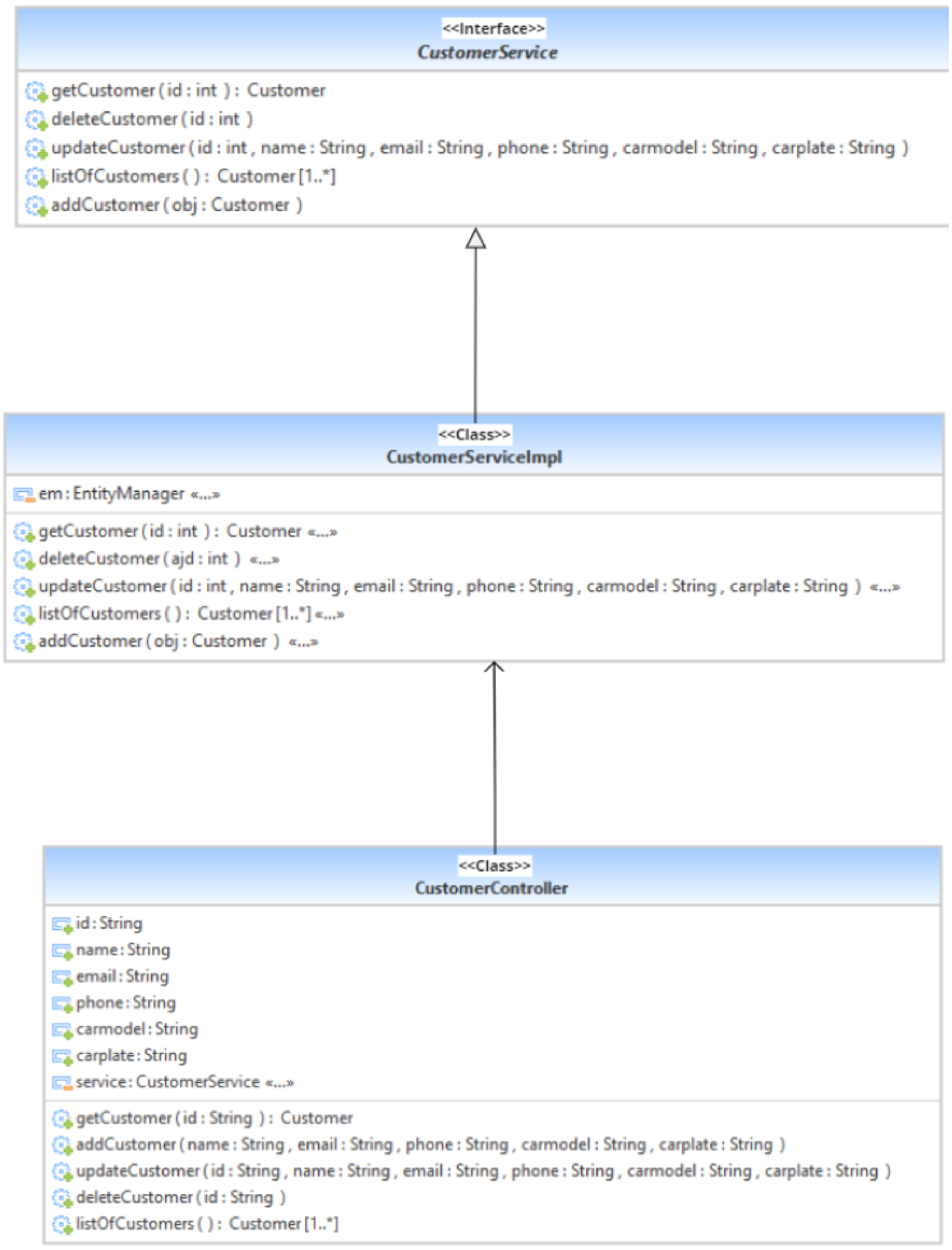




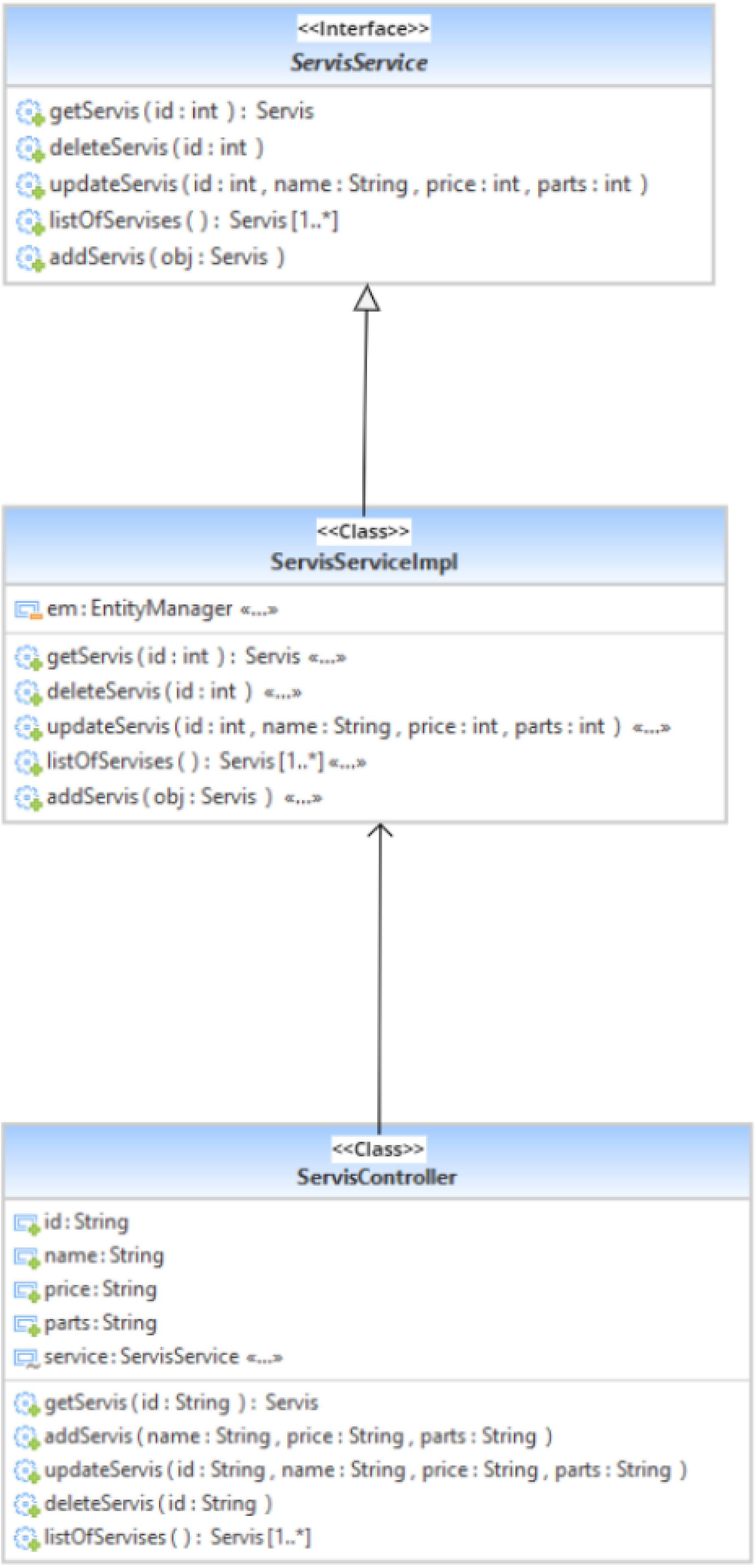




<<Interface>>
CustservService
getCustserv (id : int) : Custserv
deleteCustserv (id : int)
updateCustserv (id : int , fkcustomer : int , fkservice : int , fixed : String , fullprice : int , paid : String)
listOfCustserves ( ) : Custserv [1..*]
isFixed (id : int)
isPaid (id : int)
addCustserv (obj : Custserv)
<<Class>>
CustservServiceImpl
em : EntityManager «...»
getCustserv (id : int) : Custserv «...»
deleteCustserv (id : int) «...»
updateCustserv (id : int , fkcustomer : int , fkservice : int , fixed : String , fullprice : int , paid : String) «...»
listOfCustserves ( ) : Custserv [1..*] «...»
isFixed (id : int) «...»
isPaid (id : int) «...»
addCustserv (obj : Custserv) «...»
<<Class>>
CustservController
id : String
fkcustomer : String
fkservice : String
fixed : String
fullprice : String
paid : String
service : CustservService «...»
getCustserv (id : String) : Custserv
addCustserv (fkcustomer : String , fkservice : String , fixed : String , fullprice : String , paid : String)
updateCustserv (id : String , fkcustomer : String , fkservice : String , fixed : String , fullprice : String , paid : String)
deleteCustserv (id : String)
listOfCustserves ( ) : Custserv [1..*]
isFixed (id : int)
isPaid (id : int)

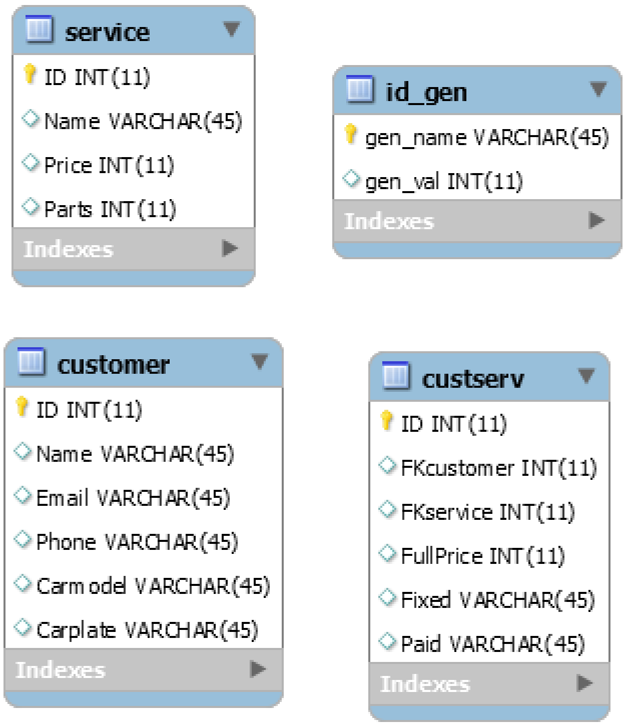


<<Interface>>
CustomerService
getCustomer (id : int) : Customer
deleteCustomer (id : int)
updateCustomer (id : int , name : String , email : String , phone : String , carmodel : String , carplate : String)
listOfCustomers ( ) : Customer [1..*]
addCustomer (obj : Customer)
<<Class>>
CustomerServiceImpl
em : EntityManager «...»
getCustomer (id : int) : Customer «...»
deleteCustomer (ajd : int) «...»
updateCustomer (id : int , name : String , email : String , phone : String , carmodel : String , carplate : String) «...»
listOfCustomers ( ) : Customer [1..*] «...»
addCustomer (obj : Customer) «...»
<<Class>>
CustomerController
id : String
name : String
email : String
phone : String
carmodel : String
carplate : String
service : CustomerService «...»
getCustomer (id : String) : Customer
addCustomer (name : String , email : String , phone : String , carmodel : String , carplate : String)
updateCustomer (id : String , name : String , email : String , phone : String , carmodel : String , carplate : String)
deleteCustomer (id : String)
listOfCustomers ( ) : Customer [1..*]



<<Interface>>
ServisService
getServis (id : int) : Servis
deleteServis (id : int)
updateServis (id : int , name : String , price : int , parts : int)
listOfServises ( ) : Servis [1..*]
addServis (obj : Servis)
<<Class>>
ServisServiceImpl
em : EntityManager «...»
getServis (id : int) : Servis «...»
deleteServis (id : int) «...»
updateServis (id : int , name : String , price : int , parts : int) «...»
listOfServises ( ) : Servis [1..*] «...»
addServis (obj : Servis) «...»
<<Class>>
ServisController
id : String
name : String
price : String
parts : String
service : ServisService «...»
getServis (id : String) : Servis
addServis (name : String , price : String , parts : String)
updateServis (id : String , name : String , price : String , parts : String)
deleteServis (id : String)
listOfServises ( ) : Servis [1..*]

Database dijagram

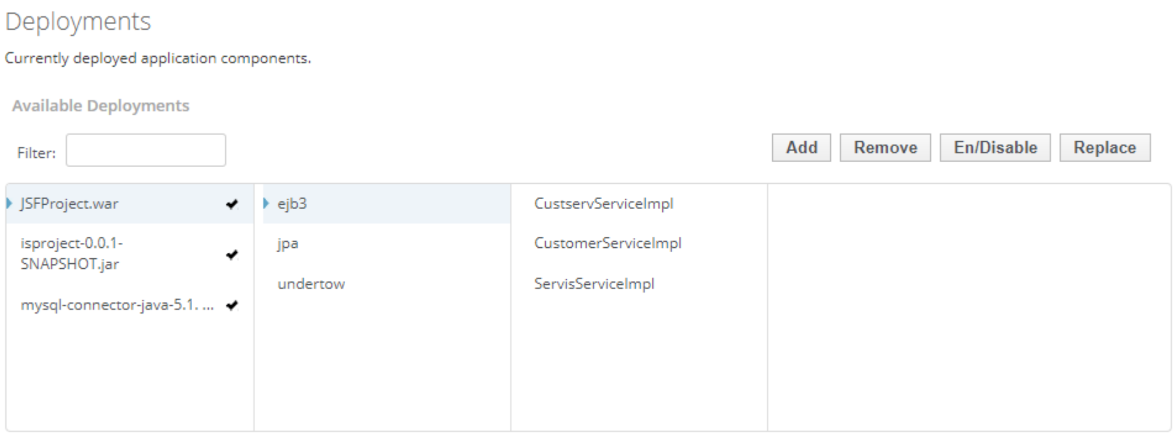


Deployment

Za depoyment aplikacije potrebno je:

- Pokrenuti MySQL server na portu 3306
- Pokrenuti WildFly server, zatim: • Desni klik na projekat.
- Run as > Run on Server.
- Finish.
- Pokrenuti docker.

Nasa JSF aplikacija se pokrenula i mozemo pristupiti serveru linkom http://127.0.0.1:9990

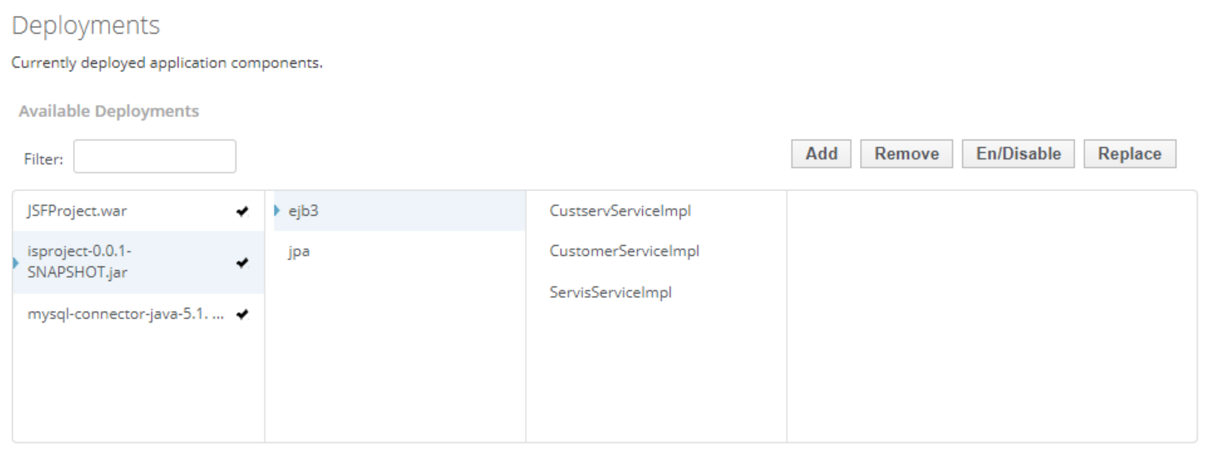

Deployments

Currently deployed application components.

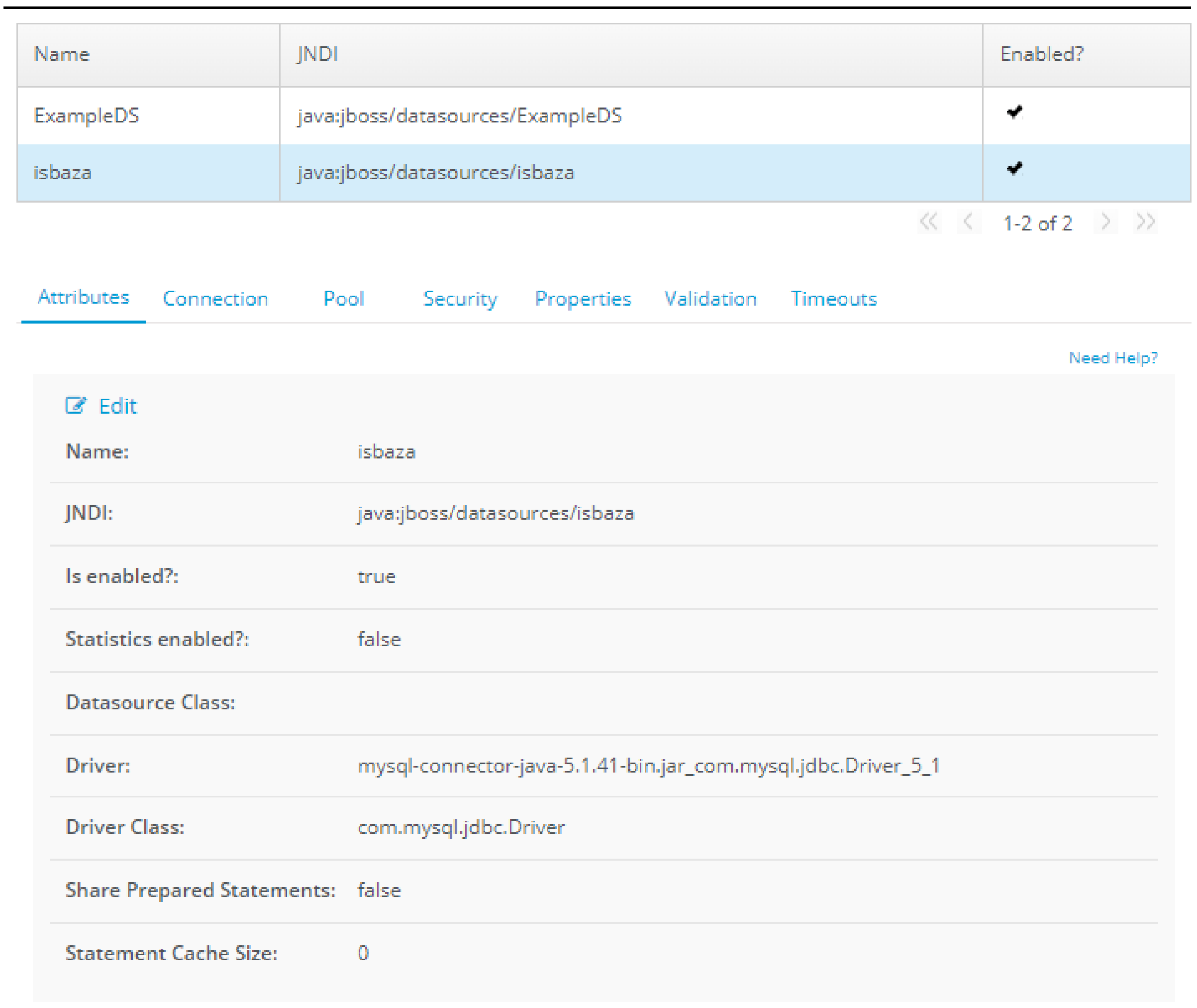
Available Deployments

Filter:

JSFProject.war ✔	ejb3	CustservServiceImpl
isproject-0.0.1-SNAPSHOT.jar ✔	jpa	CustomerServiceImpl
mysql-connector-java-5.1. ... ✔	undertow	ServisServiceImpl

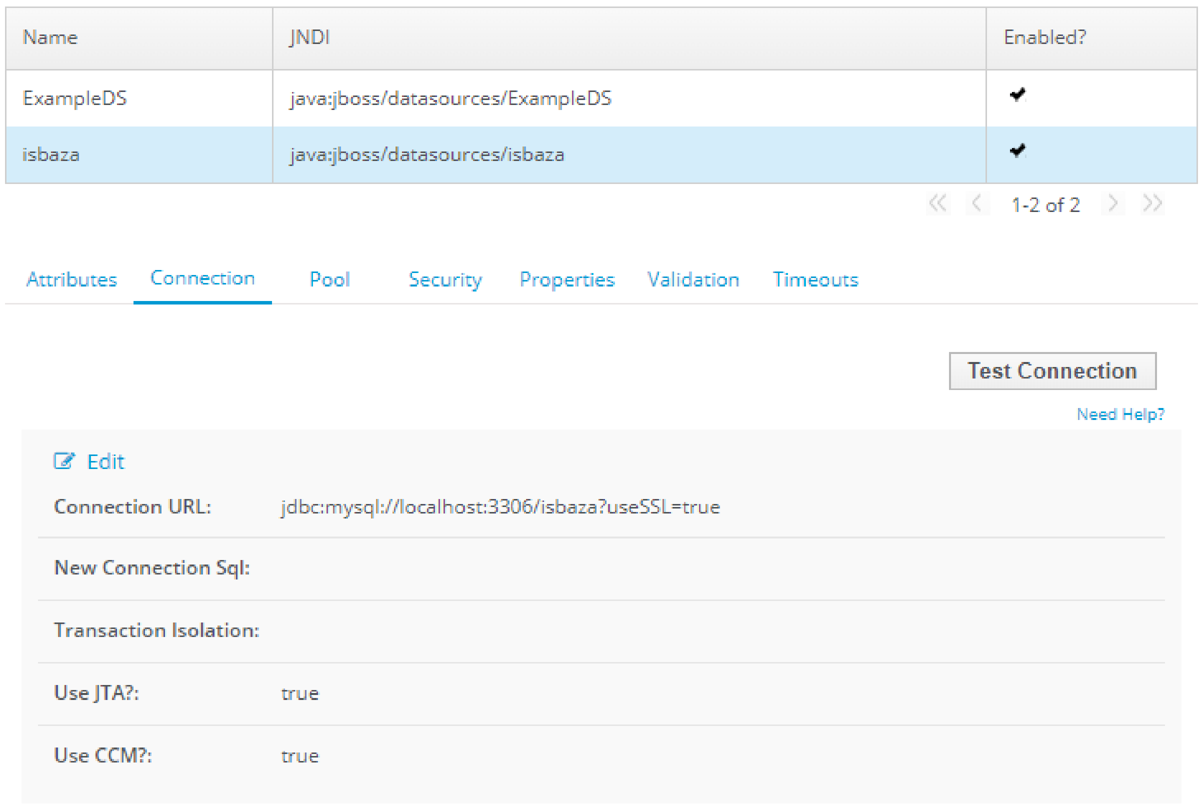


JSFProject.war ✓	ejb3	CustservServiceImpl
isproject-0.0.1-SNAPSHOT.jar ✓	jpa	CustomerServiceImpl
mysql-connector-java-5.1. ... ✓		ServisServiceImpl



Name	JNDI	Enabled?
ExampleDS	java:jboss/datasources/ExampleDS	✓
isbaza	java:jboss/datasources/isbaza	✓

Name:	isbaza
JNDI:	java:jboss/datasources/isbaza
Is enabled?:	true
Statistics enabled?:	false
Datasource Class:	
Driver:	mysql-connector-java-5.1.41-bin.jar_com.mysql.jdbc.Driver_5_1
Driver Class:	com.mysql.jdbc.Driver
Share Prepared Statements:	false
Statement Cache Size:	0



Name	JNDI	Enabled?
ExampleDS	java:jboss/datasources/ExampleDS	✓
isbaza	java:jboss/datasources/isbaza	✓

Connection URL:	jdbc:mysql://localhost:3306/isbaza?useSSL=true
New Connection Sql:	
Transaction Isolation:	
Use JTA?:	true
Use CCM?:	true

Ukoliko je sve korektno odrađeno možemo pristupiti aplikaciji na linku: http://localhost:8080/JSFProject

Unit testovi

1. Prvi unit test proverava inicijalizaciju CustomerServiceImpl
    - Metoda: new CustomerServiceImpl();
    - Preduslov: Da klasa postoji.
    - Koraci:
        - Instanciranje Customer objekta sa ovim funkcijom

2. i 5.. Drugi i peti unit test proverava da li funkcija vraca korisnika sa zadatim ID-jem
    - Metoda: getCustomer(int id)
    - Preduslov: Nema.
    - Koraci:
        - Trazi u bazi korisnika sa zadatim ID-jem.
        - (2) Ako ne moze, vraca NULL
        - (5) Ako moze, vraca korisnika

3. Treći unit test proverava dodavanje korisnika u bazi.
    - Metoda: addCustomer(Customer obj)
    - Preduslov: Da je prosledjen objekat tipa Customer.
    - Koraci:
        - Dodavanje objekta kao argument
        - Dodaje u bazu.

        - Proverava da uplata nije ostala u bazi.

4. Četvrti unit test proverava azuriranje korisnika
    - Metoda: updateCustomer()
    - Preduslov: Da korisnik vec postoji
    - Koraci:
        - Poslati novi objekat tipa Customer sa istim ID-jem
        - Naci u bazi korisnika sa istim ID-jem
        - Izvrsiti izmene

Docker demployment (.net 6)

Kreiramo novi Web API

```
dotnet new webapi -ISDocker
```

U folderu ISDocker kreiramo Dockerfile i u njemu upisemo sledeci code

```
FROM mcr.microsoft.com/dotnet/sdk:6.0 AS build-env
WORKDIR /app

COPY *.csproj ./
RUN dotnet restore

COPY ./ ./
RUN dotnet publish -c Release -o out

FROM mcr.microsoft.com/dotnet/aspnet:6.0
WORKDIR /app
COPY --from=build-env /app/out .
ENTRYPOINT ["dotnet", "ISDocker.dll"]
```

Mozemo da dodamo .dockerignore u ISDocker folderu kako bi image bio sto manji.

```
bin/
obj/
```

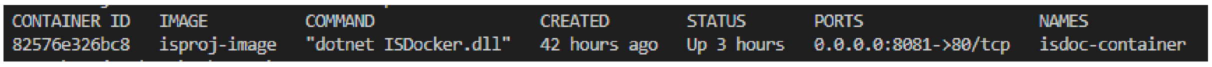
Pokrenemo command terminal u folder ISDocker i preko njega pozovemo komandu:

```
docker build -t isproj-image -f Dockerfile .
```

U isto command terminal zatim pozovemo sledecu komandu za kreiranje container-a i njegovo pokretanje:

```
docker run -d -p 8081:80 --name isdoc-container isproject-image
```

Ukoliko je sve odrađeno tačno komanda `docker ps` bi trebala da pokaže:



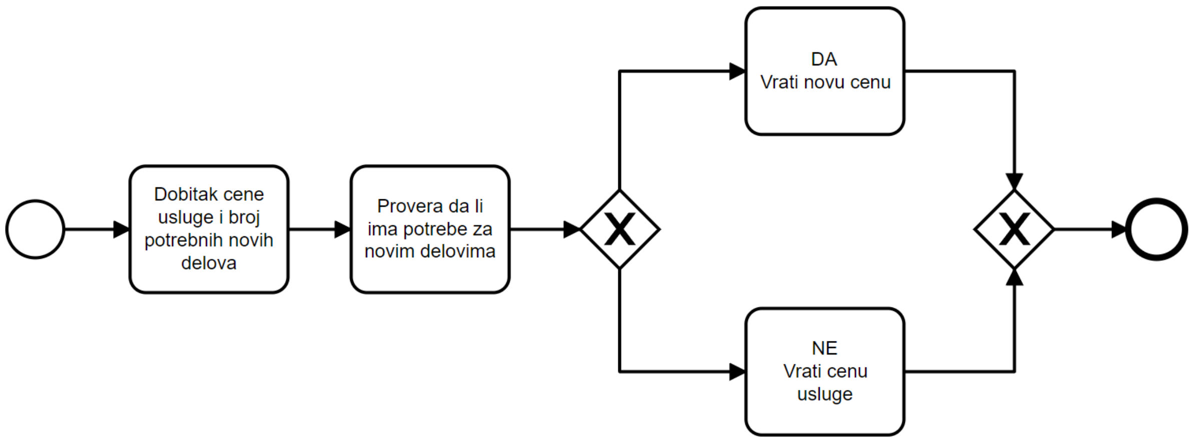
```
CONTAINER ID   IMAGE          COMMAND                CREATED        STATUS       PORTS                  NAMES
82576e326bc8   isproj-image   "dotnet ISDocker.dll"  42 hours ago   Up 3 hours   0.0.0.0:8081->80/tcp   isdoc-container
```

Kontroleru prosledjujemo 2 broja, prvi predstavlja cenu usluge, dok drugi predstavlja koliko novih delova je bilo potrebno pribaviti da se usluga izvrsi.
Ako ima dodatnih delova onda dodati cenu tih delova na cenu usluge.

```
using Microsoft.AspNetCore.Mvc;

namespace ISDocker.Controllers;

[ApiController]
[Route("[controller]")]
public class MoneyController : ControllerBase
{
    public MoneyController() {}

    [Route("{price}/{hasParts}")]
    [HttpGet]
    public async Task<ActionResult> Get(int price, int hasParts)
    {
        if(hasParts > 0) {
          Random objRnd = new Random();
          int hasPartsPrice = objRnd.Next(hasParts*10, hasParts*1000);
          return await Task.FromResult(Ok(hasPartsPrice + price));
        }else
          return await Task.FromResult(Ok(price));
    }
}
```

BPMN

Dijagram predstavlja obrađivanje zahteva koji vrši .net web aplikacija u docker kontejneru, proverava da li ima novih delova potrebnih za izvrsenje usluge i na osnovu toga vraca cenu.


Dobitak cene usluge i broj potrebnih novih delova
Provera da li ima potrebe za novim delovima
DA
Vrati novu cenu
NE
Vrati cenu usluge

Pristup: http://localhost:8081/Money/**{broj1}**/**{broj2}**