UNIVERZITET U NOVOM SADU

**FAKULTET TEHNIČKIH NAUKA**

# VIRTUELIZACIJA PROCESA

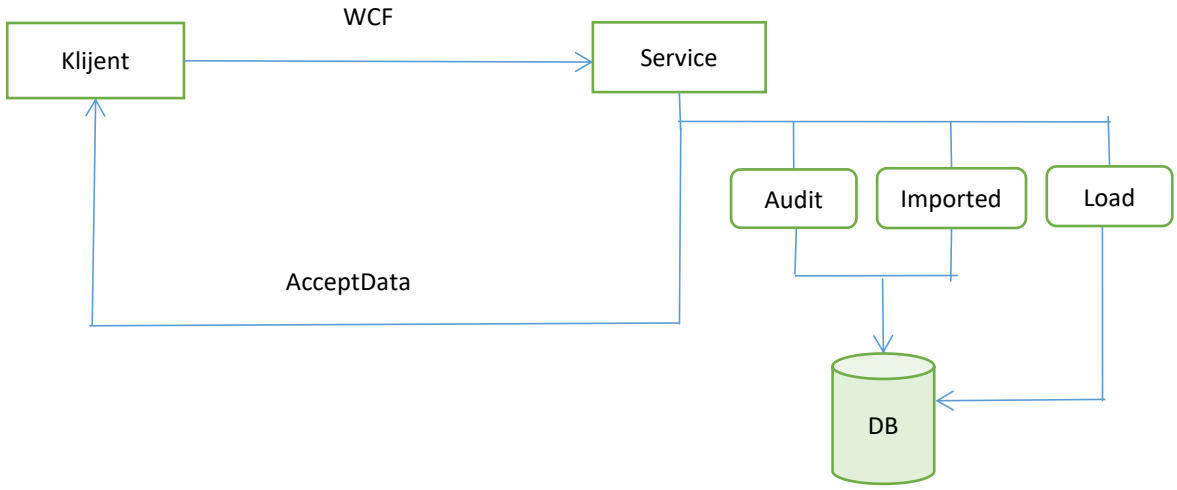
## Kreiranje CSV datoteka na osnovu XML baze podataka


Luka Ivković PR133/2020
Petar Milanović PR135/2020
Marija Ostović PR107/2020
Nikola Radović PR143/2020

**Sadržaj:**

1. Opis projektnog zadatka
2. Arhitektura (dijagram) projekta sa tokom podataka
3. Opis interfejsa sa opisom osnovnih funkcionalnosti
4. Opis korišćenih tehnologija i alata
5. Zaključak

**Arhitektura (dijagram) projekta sa tokom podataka**



## Opis projektnog zadatka

Cilj aplikacije je kreiranje CSV datoteka na osnovu podataka iz XML datoteke, koja sadrži informacije o prognoziranoj i ostvarenoj potrošnji električne energije po satu.

Aplikacija se sastoji od servisa i klijentske aplikacije koje komuniciraju putem WCF-a. Servis i klijentska aplikacija koriste *WCF (Windows Communication Foundation)* za razmenu podataka i pozive metoda.

Potrebne podatke o prognozama i ostvarenjima potrošnje električne energije dobijamo na osnovu prosleđene XML datoteke.

```xml
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<rows>
	<row>
		<TIME_STAMP>2023-01-17 00:00</TIME_STAMP>
		<FORECAST_VALUE>6297.66</FORECAST_VALUE>
		<MEASURED_VALUE>6234</MEASURED_VALUE>
	</row>
	<row>
		<TIME_STAMP>2023-01-17 01:00</TIME_STAMP>
		<FORECAST_VALUE>5892.64</FORECAST_VALUE>
		<MEASURED_VALUE>5886</MEASURED_VALUE>
	</row>
	<row>
		<TIME_STAMP>2023-01-17 02:00</TIME_STAMP>
		<FORECAST_VALUE>5475.54</FORECAST_VALUE>
		<MEASURED_VALUE>5483</MEASURED_VALUE>
	</row>
	<row>
		<TIME_STAMP>2023-01-17 03:00</TIME_STAMP>
		<FORECAST_VALUE>5229.33</FORECAST_VALUE>
		<MEASURED_VALUE>5188</MEASURED_VALUE>
	</row>
```

```xml
<appSettings>
    <add key="uploadPath" value="C:\Users\User\OneDrive\Desktop"></add>
</appSettings>
```

Klijentska aplikacija prati definisanu putanju i naziv XML datoteke. Kada se na toj putanji kreira datoteka sa nazivom ili se promeni atribut "Changed" postojeće datoteke, ona se šalje servisu. Servis koristi pristiglu datoteku da bi kreirao objekte klase Load, koji predstavljaju podatke o potrošnji po satu. Ti objekti se upisuju u In-Memory bazu podataka.

*In-Memory baza podataka* je vrsta baze podataka koja čuva podatke u glavnoj memoriji (RAM) radi bržeg pristupa i obrade umesto na trajnom skladištu. In-Memory baze podataka omogućavaju visoku brzinu i performanse jer se podaci ne moraju čitati i pisati sa fizičkih diskova. Oni su privremeni i postoje samo dok je servis pokrenut, što znači da se podaci gube prilikom gašenja ili restartovanja sistema. Ova tehnologija se često koristi kada je potrebno brzo čitanje, upisivanje i manipulacija podacima u aplikacijama koje zahtevaju visoku efikasnost i reaktivnost.

Veoma je bitno da osiguramo tačnost i integritet informacija koje se unose u bazu podataka, potrebno je izvršiti provjeru koja nam pomaže da identifikujemo eventualne greške ili neusklađenosti u podacima, kao što su nevalidni brojevi, nedostajući podaci ili problemi sa strukturom XML datoteke. Kroz ovu provjeru osiguravamo kvalitet podataka i smanjujemo mogućnost grešaka prilikom daljnje obrade.

Nakon kreiranja posljednjeg objekta Load iz XML datoteke, aktivira se događaj koji može rezultirati jednom ili više CSV datoteka. Broj generisanih datoteka se podešava u App.config datoteci servisa. Ova funkcionalnost se postiže korištenjem Event-a i Delegate-a.

*Delegati* predstavljaju pokazivače na metode koje imaju iste argumente i povratnu vrednost kao sam delegat. Delegati se koriste za prosleđivanje metoda kao argumenata drugim metodama. Bilo koja metoda iz bilo koje dostupne klase ili strukture koja odgovara tipu delegata može biti dodeljena delegatu. *Događaji (events)* sluze da se objekat neke klase obavesti da se desilo nesto od interesa za taj objekat. Događaji mogu da budu različite sistemski notifikacije na koje aplikacija treba da odreaguje. Mehanizam događaja koristi *publisher-subscriber* razvojni uzorak (design pattern).

*Publisher* je objekat koji sadrži definiciju događaja i delegat koji implementira rukovaoca događajem (event handler). *Subscriber* je objekat koji prihvata događaj i vrši njegovu implementaciju.

| | A | B | C | D |
|---|---|---|---|---|
| 1 | DATE | TIME | FORECAST_VALUE | MEASURED_VALUE |
| 2 | 2023-01-20 | 0:00 | 6034.91 | 6169 |
| 3 | 2023-01-20 | 1:00 | 5687.9 | 5768 |
| 4 | 2023-01-20 | 2:00 | 5287.6 | 5384 |
| 5 | 2023-01-20 | 3:00 | 5023.27 | 5096 |
| 6 | 2023-01-20 | 4:00 | 4911.19 | 4961 |
| 7 | 2023-01-20 | 5:00 | 4973.92 | 4974 |
| 8 | 2023-01-20 | 6:00 | 5234.84 | 5167 |
| 9 | 2023-01-20 | 7:00 | 5562.69 | 5528 |

Nakon obrade XML datoteke, kreira se objekat ImportedFile koji se upisuje u bazu podataka. Servis šalje jednu ili više CSV datoteka kao rezultat klijentskoj aplikaciji. Klijentska aplikacija čuva primljene CSV datoteke na posebnoj putanji definisanoj u App.config datoteci, ispisuje pun naziv datoteke u konzoli za svaku primljenu datoteku.

```
importedFiles.Add(new ImportedFile(importedFiles.Count, name));
ImportedFile file = importedFiles[importedFiles.Count - 1];
DataBase.AddData("ImportedFiles", file);
```

```
<appSettings>
    <add key="NumberCsvFIles" value="5"></add>
</appSettings>
```

```
List<MemoryStream> file= proxy.AcceptData(xmlString, e.Name);
string path = ConfigurationManager.AppSettings["uploadCsv"];

for (int i = 0; i < file.Count; i++)
{
    string paths = path + "data" + Convert.ToString(i) + ".csv";
    Console.WriteLine(paths);
    byte[] fileBytes;
    using (var memoryStream = new MemoryStream())
    {
        file[i].CopyTo(memoryStream);
        fileBytes = memoryStream.ToArray();
    }
```

Takođe,ono što je značajno u ovom projektnom zadatku je voditi računa o održavanju memorije prilikom rada sa datotekama. To se postiže korišćenjem Dispose paterna, koji omogućava oslobađanje resursa koje datoteke koriste.
*Dispose patern* se primjenjuje na objekte koji implementiraju IDisposable interfejs. Na taj način se efikasno upravlja memorijom i sprečava curenje resursa. Kod radnje sa unmanaged resursima, inženjer treba voditi brigu o njihovom oslobađanju i upravljanju memorijom. *GC (Garbage Collector)* ne oslobađa ove resurse automatski.
Koristeći vezu Using bloka i Dispose metode u našem projektnom zadatku obezbijedili tačno i kontrolisano oslobađanje resursa.

## Opis interfejsa sa opisom osnovnih funkiconalnosti

Aplikacija se sastoji od servisa i klijentske aplikacije koje komuniciraju putem WCF-a. Servis i klijentska aplikacija koriste WCF (Windows Communication Foundation) za razmenu podataka i pozive metoda.
Klijentska aplikacija pruža konfiguracijske informacije, kao što su adresa servisa, vrsta kanala i postavke sigurnosti. Na osnovu tih informacija, ChannelFactory kreira kanal koji će se koristiti za komunikaciju. Da bi klijentska aplikacija mogla da koristi servis, potrebno je da generiše proxy koja predstavlja klijentski ugovor. Proxy omogućava klijentu da poziva metode servisa kao da se nalazi lokalno.

```
namespace Common.Interfaces
{
    [ServiceContract]
    1 reference
    public interface IFileHandling
    {
        [OperationContract]
        1 reference
        void AcceptData(string document);
    }
}
```

*Kako bismo pokrenuli neki servis prethodno moramo definisati interfejs koji ćemo koristiti kao Contract. Da bi se jedan interfejs koristio kao contract neophodno ga je dekorisati atributom ServiceContract, a metode koje želimo da izlaže sa OperationContract atributom (nalaze se u System.ServiceModel biblioteci)*

Interfejs IFileHandling sadrži metodu *AcceptData* koja prima XML dokument kao string. U klasi EventArgsCsv, metoda AcceptData poziva se s prosleđenim XML dokumentom.

```
public class EventArgsCsv
{
    public Dictionary<string, List<Load>> Files = new Dictionary<string, List<Load>>();

    1 reference
    public static void SendTo(string xml)
    {
        FileHandler.DataSet += OnDataSet;
        FileHandler.Provjera(xml);
    }

    1 reference
    public static void OnDataSet(object obj, EventArgsCsv args)
    {
        Dictionary<string, List<Load>> data = args.Files;
        Console.WriteLine("Ulaz u Event");
        CsvDataHandler.DataToCSV("C:/Users/Nikola/Desktop/Datoteka.csv", data);
        PosaljiMajstoru.Posalji();
    }
}
```

```
1 reference
public static  void DataToCSV(List<Load> fajl, Dictionary<string, List<Load>> lista)
{
    if (FileHandler.promjena)
    {
        string path = "data.csv";
        int counter = 0;

        var csv_number = ConfigurationManager.AppSettings["NumberCsvFIles"];
        int csv_num = Convert.ToInt32(csv_number);
        if (csv_num <= 1)


            ...
            ...
            ...
            ...
            ...
            ...
```

DataToCsv poziva GetCsvFiles koji vraca datoteku I poziva metodu iz FileHandler-a koje smjesta podatke u globalnu promjenjivu u FileHandleru I AcceptData je vraca kao povratnu vrijednost.

```
public List<MemoryStream> AcceptData(string document, string name)
{


    EventArgsCsv.SendTo(document, name);

    return file;
}
```

## Opis korišćenih tehnologija i alata

1. Microsoft Visual Studio - Integrisano programsko okruženje, programirano od strane kompanije Microsoft.
2. C# -moćan programski jezik koji pruža bogate funkcionalnosti koji smo koristili za pisanje logike aplikacije. Kroz korišćenje C# jezika, možemo definisati klase, metode, svojstva i druge konstrukte koji su potrebni za obradu XML datoteka i komunikaciju sa servisom.
3. Windows Communication Foundation (WCF)- framework koji služi za pravljenje servisno-orijentisanih aplikacija. To je Microsoft-ova tehnologija koja omogućava razvoj distribuiranih aplikacija i servisa.Klijenti mogu komunicirati sa WCF servisom putem generisanih klijentskih proxy-ja.
4. App.Config datoteka omogućava fleksibilnost i konfigurabilnost aplikacije bez potrebe za promenom izvornog koda. Ovo je posebno korisno kada se aplikacija mora prilagoditi različitim okruženjima. . App.Config datoteka sadrži parametre i postavke koje se koriste u aplikaciji.

Uz pomoć Visual Studio, .NET Frameworka, C# jezika i App.Config datoteke, možemo razviti aplikaciju koja efikasno komunicira sa servisom putem WCF-a, koristi konfiguracione parametre iz App.Config datoteke i obavlja zadatke poput manipulacije XML datoteka i generisanja CSV datoteka na osnovu servisnih podešavanja.

## Zaključak

Kroz ovaj projekat, uspostavlja se efikasna komunikacija između klijenta i servera putem WCF-a, omogućavajući prenos XML datoteka i generisanje CSV datoteka.
Klijentska aplikacija omogućava odabir i slanje XML datoteke serveru, koji zatim obrađuje tu datoteku i kreira odgovarajuću CSV datoteku, zatim se šalje serveru putem WCF komunikacije.
Na serveru, XML datoteka se analizira i obrađuje. Na osnovu njenog sadržaja, generišu se podaci koji se zatim konvertuju i upisuju u CSV format. Generisana CSV datoteka sadrži podatke o prognoziranoj i ostvarenoj potrošnji električne energije.
Buduća istraživanja i proširenja projekta mogu se fokusirati na dodavanje novih funkcionalnosti, poboljšanje bezbednosti, automatizaciju procesa, implementaciju trajne baze podataka i proširenje validacija i obrade grešaka, podrška za više formata datoteka.
Uvesti mehanizme zaštite podataka i komunikacije kako bi se osigurala sigurnost prilikom razmene podataka između klijenta i servisa.
Pored toga, uvođenje sigurnosnih mehanizama zaštite podataka i komunikacije između klijenta i servisa je važan aspekt koji treba razmotriti.
Projekat pruža osnovu za dalji razvoj i proširenje funkcionalnosti u skladu sa potrebama i zahtevima klijenata.