

Tehnička dokumentacija - Predmetni projekat

Projekat sadrži pet glavnih celina, i dve pomoćne.

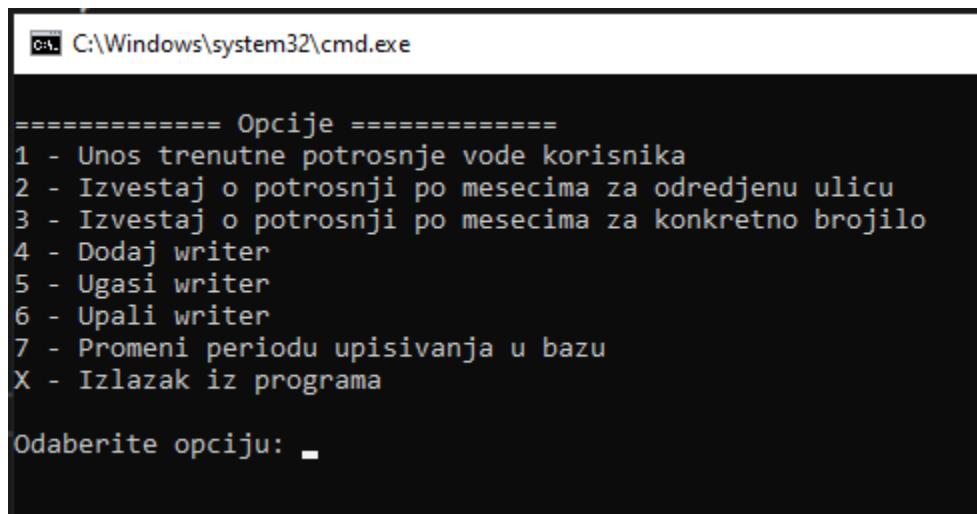
Glavne celine su:

- UI Interface
- Writer
- Replicator
 - Sender
 - Receiver
- Reader
- Reports

Pomoćne celine su:

- Connection Utils
- Data models

UI Interface

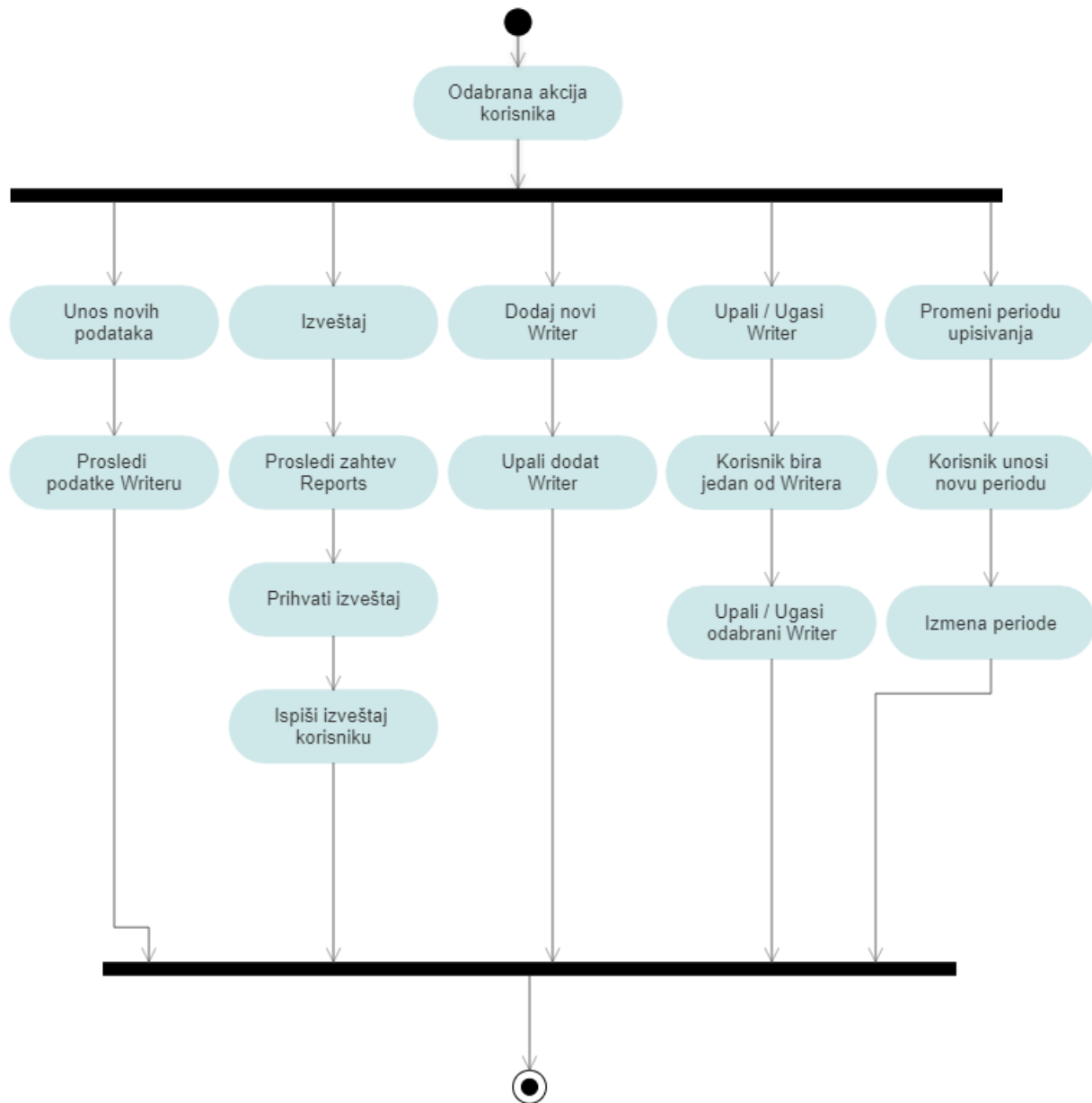


```
C:\Windows\system32\cmd.exe

===== Opcije =====
1 - Unos trenutne potrošnje vode korisnika
2 - Izvestaj o potrošnji po mesecima za odredjenu ulicu
3 - Izvestaj o potrošnji po mesecima za konkretno brojilo
4 - Dodaj writer
5 - Ugasi writer
6 - Upali writer
7 - Promeni periodu upisivanja u bazu
X - Izlazak iz programa

Odaberite opciju: _
```

UML Activity Diagram: UI Interface



Konzolni korisnički interfejs poseduje opcije kao na slici iznad.

Prva opcija omogućava unos nove potrošnje za jedno brojilo pozivom funkcije UnosPotrosnje():

```

1 reference
public static void UnosPotrosnje()
{
    int id;
    double potrosnja;
    Console.WriteLine("Unesite id potrosnje:");
    int.TryParse(Console.ReadLine(), out id);
    Console.WriteLine("Unesite potrosnju:");
    double.TryParse(Console.ReadLine(), out potrosnja);
    int index = pronadjiUkljucenWriter();
    try
    {
        writers[index].AcceptPotrosnja(id, potrosnja);
        Console.WriteLine("Podaci prosledjeni slobodnom writeru");
    }
    catch(Exception e)
    {
        Console.WriteLine("[ERROR] " + e.Message);
    }
}

```

Metoda unešene podatke prosleđuje nekom od Writera za dalju obradu, pozivanje metode AcceptPotrosnja(int id, double potrosnja).

Druga opcija prikazuje izveštaj o potrošnji po mesecima za određenu ulicu, pozivanjem funkcije IzvestajUlica().

```

public static void IzvestajUlica()
{
    Console.WriteLine("Unesite ulicu za pretragu:");
    string ulica = Console.ReadLine();
    try
    {
        List<Potrosnja> potrosnja = reports.PotrosnjaPoUlici(ulica).ToList();
        if (potrosnja.Count == 0)
        {
            Console.WriteLine("Nema podataka za datu ulicu!");
        }
        for (int x = 0; x < meseci.Length; x++)
        {
            Console.WriteLine("\n===== " + meseci[x] + " =====\n");
            Console.WriteLine(Potrosnja.GetFormattedHeader());
            foreach (var p in potrosnja)
            {
                if (p.Mesec - 1 == x)
                {
                    Console.WriteLine(p.ToString());
                }
            }
            Console.WriteLine("=====\\n");
        }
    }
    catch(Exception e)
    {
        Console.WriteLine("[ERROR] " + e.Message);
    }
}

```

Prikuplja se izveštaj od Reports komponente i uredno ispisuje korisniku.

Treća opcija prikazuje izveštaj o potrošnji po mesecima za određeno brojilo, pozivanjem funkcije IzvestajBrojilo().

```
public static void IzvestajBrojilo()
{
    Console.WriteLine("Uneiste id brojila za pretragu:");
    int id;
    int.TryParse(Console.ReadLine(), out id);
    try
    {
        List<Potrosnja> potrosnja = reports.PotrosnjaPoBrojilu(id).ToList();
        if (potrosnja.Count == 0)
        {
            Console.WriteLine("Nema podataka za dato brojilo!");
        }
        for (int x = 0; x < meseci.Length; x++)
        {
            Console.WriteLine("\n===== " + meseci[x] + " =====\n");
            Console.WriteLine(Potrosnja.GetFormattedHeader());
            foreach (var p in potrosnja)
            {
                if (p.Mesec - 1 == x)
                {
                    Console.WriteLine(p.ToString());
                }
            }
            Console.WriteLine("=====\n");
        }
    }
    catch (Exception e)
    {
        Console.WriteLine("[ERROR] " + e.Message);
    }
}
```

Prikuplja se izveštaj od Reports komponente i uredno ispisuje korisniku.

Četvrta opcija dodaj novi Writer, pozivom funkcije DodajWriter().

```
public static void DodajWriter()
{
    try
    {
        Writer.Writer writer = new Writer.Writer(sender);
        writer.Ukljuci();
        writers.Add(writer);
        Console.WriteLine("Writer uspesno dodat!");
    }
    catch (Exception e)
    {
        Console.WriteLine("[ERROR] " + e.Message);
    }
}
```

Funkcije dodaje novi Writer i uključuje isti pozivom metode Writer.Ukljucii().

Peta opcija omogućava gašenje jednog od Writera po indeksu, pozivom funkcije UgasiWriter().

```

public static void UgasiWriter()
{
    int i=0;
    foreach (var w in writers)
    {
        Console.WriteLine("Writer[{0}]:{1}",i,w.Stanje ? "Ukljucen" : "Iskljucen");
        i++;
    }
    Console.Write("Unesite indeks:");
    int unos;
    int.TryParse(Console.ReadLine(),out unos);
    if(unos>=0 && unos<writers.Count)
    {
        writers[unos].Iskljuci();
        Console.WriteLine("Writer sa indeksom {0} je uspesno ugasen!",unos);
    }
    else
    {
        Console.WriteLine("Unet je indeks van opsega!");
    }
}

```

Funkcija za uneti indeks gasi odgovarajući Writer pozivom metode Writer.Iskljuci().

Šesta opcija omogućava paljenje jednog od Writera na osnovu indeksa, pozivom funkcije UpaliWriter().

```

public static void UpaliWriter()
{
    int i = 0;
    foreach (var w in writers)
    {
        Console.WriteLine("Writer[{0}]:{1}", i, w.Stanje ? "Ukljucen" : "Iskljucen");
        i++;
    }
    Console.Write("Unesite indeks:");
    int unos;
    int.TryParse(Console.ReadLine(), out unos);
    if (unos >= 0 && unos < writers.Count)
    {
        writers[unos].Ukljuci();
        Console.WriteLine("Writer sa indeksom {0} je uspesno ugasen!", unos);
    }
    else
    {
        Console.WriteLine("Unet je indeks van opsega!");
    }
}

```

Funkcija za uneti indeks pali odgovarajući Writer pozivom metode Writer.Ukljuci().

Sedma opcija omogućuje promenu perioda upisivanja podataka u bazu , pozivom funkcije PromeniPeriodu().

```
private static void PromeniPeriodu()
{
    Console.Write("Unesite novu periodu:");
    int perioda;
    int.TryParse(Console.ReadLine(), out perioda);
    try
    {
        receiver.SetPerioda(perioda);
    }
    catch (ArgumentException ae)
    {
        Console.WriteLine("[ERROR] " + ae.Message);
    }
}
```

Funkcija menja periodu upisivanja pozivom metode Receiver.SetPerioda(int perioda).
 Prosleđena perioda je reda sekunde.

Pomoćna funkcija PronadjiUkljucenWriter() služi sa pronalazak prvog upaljenog Writera kome se mogu uneti podaci proslediti. Vraća indeks u listi Writera. U slučaju da ne postoji slobodan Writer, poziva se funkcija DodajWriter().

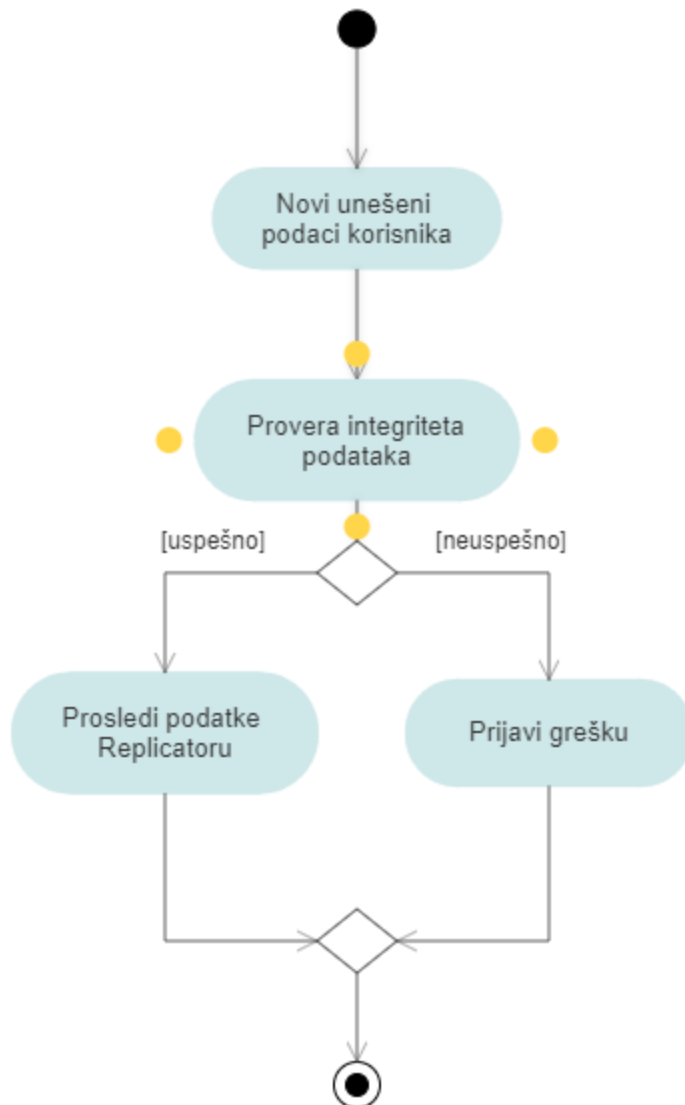
```
private static int pronadjiUkljucenWriter()
{
    int i=0;
    foreach(var w in writers)
    {
        if(w.Stanje)
        {
            return i;
        }
        i++;
    }

    DodajWriter();
    return writers.Count - 1;
}
```

Writer

Writer u sebi sadrži metode Ukljuci(), Iskljuci(), AcceptPotrosnja() kao i konstruktor Klase.

UML Activity Diagram: Writer



```
public void Ukljuci()
{
    stanje = true;
}
```

3 references

```
public void Iskljuci()
{
    stanje = false;
}
```

Metode Ukljuci() i Iskljuci() menjaju tekuće stanje Writera.

```
8 references
public Writer(IReplicatorSender sender)
{
    if (sender == null)
    {
        throw new ArgumentNullException("Sender ne sme biti null.");
    }
    else
    {
        stanje = false;
        this.sender = sender;
    }
}
```

Konstruktor od parametara očekuje IReplicatorSender, kome se podaci pristigli od UI interfejsa prosleđuju.

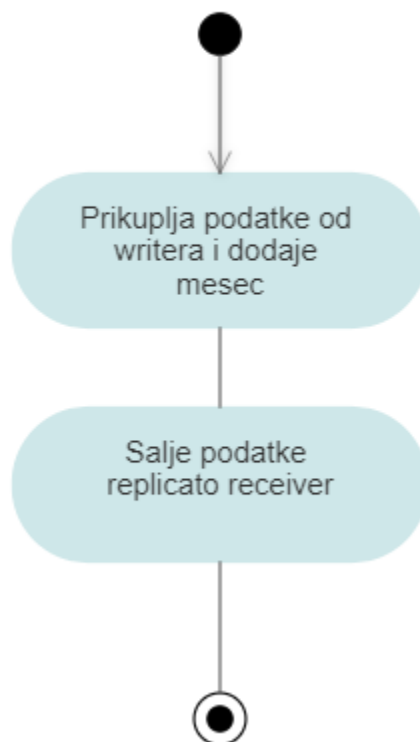
```
5 references
public void AcceptPotrosnja(int id, double potrosnja)
{
    if (!stanje)
    {
        throw new InvalidOperationException("Writer trenutno nije ukljucen. Ukljucite pre koriscenja.");
    }
    if (id <= 0)
    {
        throw new ArgumentException("Id nije validan.");
    }
    if (potrosnja < 0)
    {
        throw new ArgumentException("Potrosnja nije validna.");
    }
    sender.GetData(id, potrosnja);
}
```


Metoda AcceptPotrosnja() prihvata podatke pristigle od UI interfejsa, proverava njihov integritet i iste proslezuje IReplicatorSenderu.

Replicator Sender

Komponenta koja pristigle podatke od Writera pakuje u objekat clase Potrosnja i isti prosleđuje Receiveru. Od metoda sadrži konstruktor sa parametrima, i metodu GetData(int id, double potrosnja).

UML Activity Diagram: Replicator Sender



```

3 references
public ReplicatorSender(IReplicatorReceiver receiver)
{
    if (receiver == null)
    {
        throw new ArgumentNullException("Reciver ne sme biti null!");
    }
    else
    {
        recv = receiver;
    }
}

```

Konstruktor od parametara očekuje IReplicatorReceiver, kome se podaci pristigli od UI interfejsa prosleđuju.

```

4 references
public void GetData(int id, double potrosnja)
{
    if (id <= 0)
    {
        throw new ArgumentException("ID brojila ne sme biti manji od nule!");
    }
    if (potrosnja < 0)
    {
        throw new ArgumentException("Potrosnja ne sme biti manja od nule!");
    }
    else
    {
        IPotrosnja potr = new Potrosnja(id, potrosnja, DateTime.Now.Month);
        recv.AddPotrosnja(potr);
    }
}

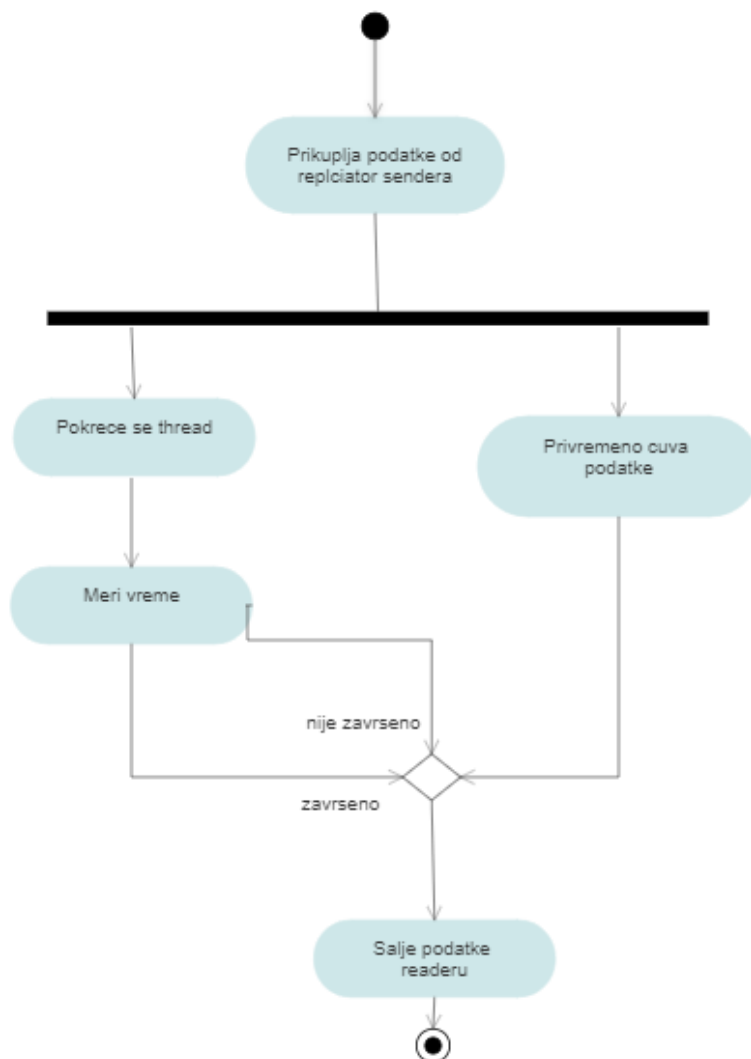
```

Metoda GetData() podatke pristigle od Sendera pakuje u objekat klase Potrosnja i isti prosleđuje Receiveru.

Receiver

Komponenta koja podatke pristigle od Sendera na predefinisane intervale prosleđuje nekom od Readera. Sadrži konstruktor sa parametrima, metode AddPotrosnja(), SavePotrosnja(), MeriVreme(), StopThread(), GetPeriod(), SetPerioda().

UML Activity Diagram: Replicator Receiver



```

public ReplicatorReceiver(int numOfReaders, int second)
{
    potrosnje = new List<IPotrosnja>();
    readers = new List<IReader>();
    if (second < 0)
    {
        throw new ArgumentException("Perioda ne sme biti manja od nule!");
    }
    period = second;
    if (numOfReaders <= 0)
    {
        throw new ArgumentException("Broj reader-a ne sme biti manji od jedan!");
    }
    for (int i = 0; i < numOfReaders; i++)
    {
        readers.Add(new Reader.Reader());
    }
    thread = new Thread(new ThreadStart(MeriVreme));
    thread.Start();
}

```

Konstruktor kao parametre očekuje broj Readera kao i periodu slanja podataka. Inicijalizuje listu Readera, kao i listu Potrosnji gde se podaci privremeno čuvaju. Takođe pokreće nezavisnu nit koja vrši prosleđivanje podataka na intervale.

```

public void AddPotrosnja(IPotrosnja potrosnja)
{
    if (potrosnja == null)
    {
        throw new ArgumentNullException("Potrosnja ne sme biti null");
    }
    else
    {
        potrosnje.Add(potrosnja);
    }
}

```

Metoda AddPotrosnja() očekuje novi objekat Potrosnja i njega upisuje u privremenu listu.

```

2 references
public void SavePotrosnja()
{
    if (potrosnje.Count > 0)
    {
        Random random = new Random();
        int i = random.Next(0, readers.Count);
        try
        {
            readers[i].SavePotrosnja(potrosnje[0]);
        }
        catch (ArgumentNullException ane)
        {
            Console.WriteLine("[ERROR] " + ane.Message);
        }
        potrosnje.RemoveAt(0);
        Console.WriteLine("\n(periodicno upisivanje podataka izvrшено)\n");
    }
}

```

Metoda SavePotrosnja() prosleđuje najstariji zapis Readeru za upis u bazu podataka.

```

public void MeriVreme()
{
    while (true)
    {
        SavePotrosnja();
        Thread.Sleep(1000 * period);
        // Console.WriteLine("Radim!"); // For debugging purposes
    }
}

3 references
public void StopThread()
{
    thread.Abort();
}

5 references

```

Metoda MeriVreme() koju nit izvršava je zaslužna za periodu upisivanja i poziva metodu SavePotrosnja().
Metoda StopThread() zaustavlja nezavisnu nit.

```
public int GetPerioda()
{
    return period;
}

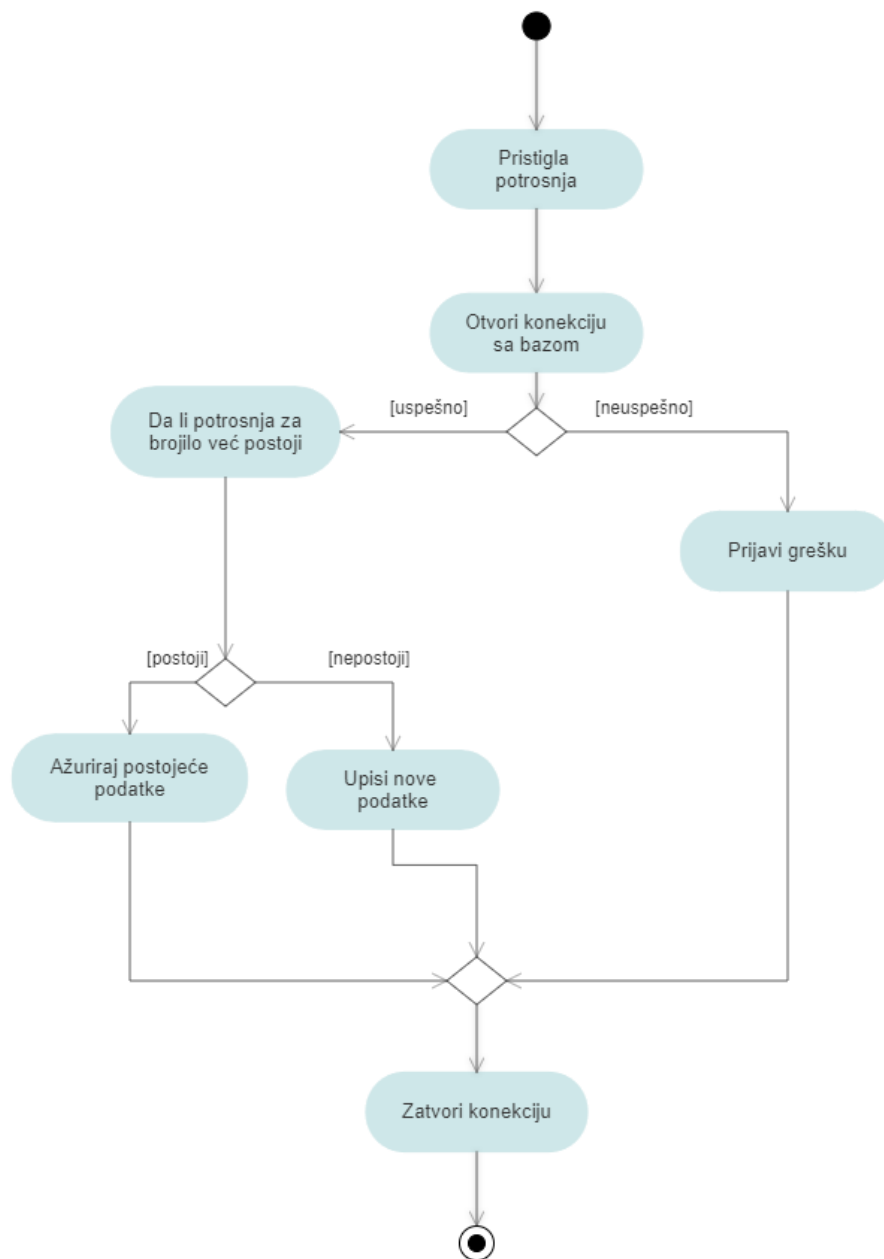
4 references
public void SetPerioda(int perioda)
{
    if (perioda < 0)
    {
        throw new ArgumentException("Perioda ne sme biti manja od nule!");
    }
    else
    {
        period = perioda;
        Console.WriteLine("Perioda upisa uspesno promenjena!");
    }
}
```

Metode GetPerioda() i SetPerioda() su getter i setter polja period.

Reader

Komponenta koja komunicira sa bazom i vrši upisivanje i čitanje podataka. Od metoda ima ReadPotrosnjaBrojila(), SavePotrosnjaBrojila(), ExistsById() .

UML Activity Diagram: Reader



```

public IEnumerable<Potrosnja> ReadPotrosnjaBrojila(int idp)
{
    if (idp <= 0)
    {
        throw new ArgumentException("ID mora biti veci od 0");
    }
    string query = "select * from Potrosnja where IDB = @id";
    List<Potrosnja> potrosnje = new List<Potrosnja>();
    using (IDbConnection conn = ConnectionParameters.GetConnection())
    {
        conn.Open();
        using (IDbCommand comm = conn.CreateCommand())
        {
            comm.CommandText = query;
            ParameterUtil.AddParameter(comm, "id", DbType.Int32);
            comm.Prepare();
            ParameterUtil.SetParameterValue(comm, "id", idp);
            using (IDataReader reader = comm.ExecuteReader())
            {
                while (reader.Read())
                {
                    try
                    {
                        Potrosnja p = new Potrosnja();
                        p.IDB = reader.GetInt32(0);
                        p.PotrosnjaB = reader.GetDouble(1);
                        p.Mesec = reader.GetInt32(2);
                        potrosnje.Add(p);
                    }
                    catch (Exception e)
                    {
                        Console.WriteLine("[ERROR] " + e.Message);
                    }
                }
            }
            return potrosnje;
        }
    }
}

```

Metoda ReadPotronjaBrojila() čita kompletnu potrošnju brojila za zadati indeks iz baze podataka, i vraća listu potrošnji za isti.


```

public int SavePotrosnja(IPotrosnja potrosnja)
{
    if(potrosnja == null)
    {
        throw new ArgumentNullException("Potrosnja ne sme biti null!");
    }
    string insert_query = "insert into Potrosnja(Potrosnja,Mesec,IDB) values ( @potrosnja, @mesec, @id)";
    string update_query = "update Potrosnja set Potrosnja=@potrosnja where Mesec=@mesec and IDB = @id ";
    using (IDbConnection conn = ConnectionParameters.GetConnection())
    {
        conn.Open();
        using (IDbCommand comm = conn.CreateCommand())
        {
            comm.CommandText = ExistsById(potrosnja.IDB,potrosnja.Mesec,conn) == 1 ? update_query : insert_query;
            ParameterUtil.AddParameter(comm, "potrosnja", DbType.Double);
            ParameterUtil.AddParameter(comm, "mesec", DbType.Int32);
            ParameterUtil.AddParameter(comm, "id", DbType.Int32);
            comm.Prepare();
            ParameterUtil.SetParameterValue(comm,"potrosnja", potrosnja.PotrosnjaB);
            ParameterUtil.SetParameterValue(comm,"mesec", potrosnja.Mesec);
            ParameterUtil.SetParameterValue(comm, "id", potrosnja.IDB);
            return comm.ExecuteNonQuery();
        }
    }
}

```

Metoda SavePotrosnja() očekuje parametar tipa Potrosnja, za koji proverava da li već postoji u bazi pozivom funkcije ExistsById(). Ukoliko postoji ažurira stare podatke, u suprotnom unosi novu potrošnju brojila.

```

public int ExistsById(int id,int mesec,IDbConnection conn)
{
    if(id <= 0)
    {
        throw new ArgumentException("ID mora biti veci od 0");
    }
    if(mesec <= 0 || mesec > 12)
    {
        throw new ArgumentOutOfRangeException("Mesec je van opsega");
    }
    if(conn == null)
    {
        throw new ArgumentNullException("Konekcija ne sme biti null!");
    }
    string query = "select count(IDB) from Potrosnja where IDB = @id and Mesec = @mesec";
    using (IDbCommand comm = conn.CreateCommand())
    {
        comm.CommandText = query;
        ParameterUtil.AddParameter(comm, "id", DbType.Int32);
        ParameterUtil.AddParameter(comm,"mesec",DbType.Int32);
        comm.Prepare();
        ParameterUtil.SetParameterValue(comm, "id", id);
        ParameterUtil.SetParameterValue(comm, "mesec", mesec);
        return (int)comm.ExecuteScalar();
    }
}

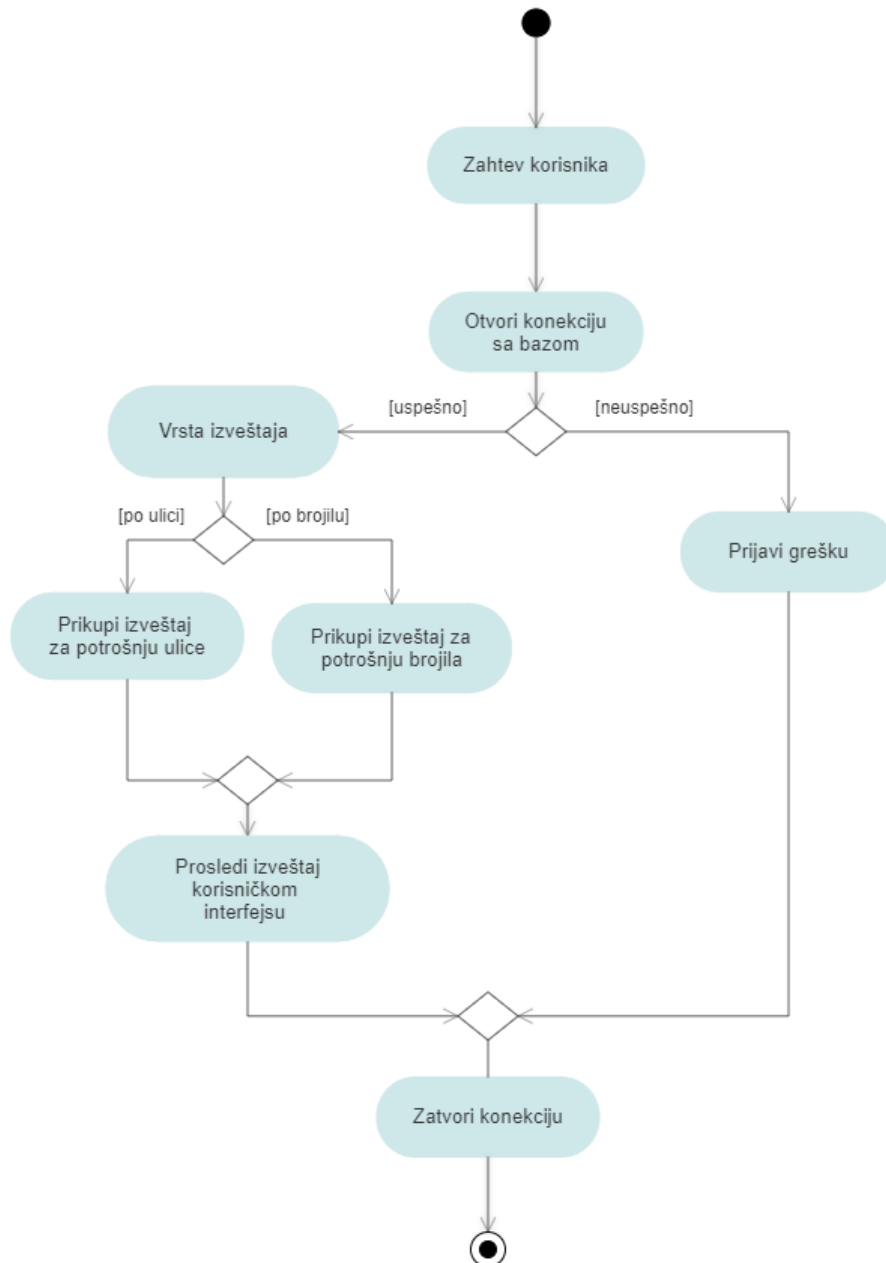
```

Metoda ExistById() proverava da li postoji vec uneti podata za dato brojilo i dati mesec.

Reports

Komponenta Reports je zaslužna za prikupljanje odgovarajućih podataka za izveštaje. Od metoda sadrži PotrošnjaPoUlicu() i PotrošnjaPoBrojilu().

UML Activity Diagram: Reports



```

5 references | 4/7 passing
public IEnumerable<Potrosnja> PotrosnjaPoUlici(string ulica)
{
    if (ulica == null)
    {
        throw new ArgumentNullException("Ulica ne sme biti null!");
    }
    ulica = ulica.Trim();
    if(ulica == "")
    {
        throw new ArgumentException("Ulica ne sme biti prazna!");
    }
    string query = "select * from Potrosnja as p join Brojilo as b on p.IDB = b.ID where Ulica = @ulica";
    List<Potrosnja> potrosnje = new List<Potrosnja>();
    using (IDbConnection conn = ConnectionParameters.GetConnection())
    {
        conn.Open();
        using (IDbCommand comm = conn.CreateCommand())
        {
            comm.CommandText = query;
            ParameterUtil.AddParameter(comm, "ulica", DbType.String, 30);
            comm.Prepare();
            ParameterUtil.SetParameterValue(comm, "ulica", ulica);
            using (IDataReader reader = comm.ExecuteReader())
            {
                while (reader.Read())
                {
                    try
                    {
                        Potrosnja p = new Potrosnja();
                        p.IDB = reader.GetInt32(0);
                        p.PotrosnjaB = reader.GetDouble(1);
                        p.Mesec = reader.GetInt32(2);
                        potrosnje.Add(p);
                    }
                    catch (Exception e)
                    {
                        Console.WriteLine("[ERROR] " + e.Message);
                    }
                }
            }
            return potrosnje;
        }
    }
}

```

Metoda PotrosnjaPoUlici() prikuplja sve potrošnje za zadatu ulicu iz baze podataka i vraća listu potrošnji kao rezultat.

```

public IEnumerable<Potrosnja> PotrosnjaPoBrojilu(int idb)
{
    if(idb <= 0)
    {
        throw new ArgumentException("ID mora biti veci od 0");
    }
    string query = "select * from Potrosnja where IDB = @id";
    List<Potrosnja> potrosnje = new List<Potrosnja>();
    using (IDbConnection conn = ConnectionParameters.GetConnection())
    {
        conn.Open();
        using (IDbCommand comm = conn.CreateCommand())
        {
            comm.CommandText = query;
            ParameterUtil.AddParameter(comm, "id", DbType.Int32);
            comm.Prepare();
            ParameterUtil.SetParameterValue(comm, "id", idb);
            using (IDataReader reader = comm.ExecuteReader())
            {
                while (reader.Read())
                {
                    try
                    {
                        Potrosnja p = new Potrosnja();
                        p.IDB = reader.GetInt32(0);
                        p.PotrosnjaB = reader.GetDouble(1);
                        p.Mesec = reader.GetInt32(2);
                        potrosnje.Add(p);
                    }
                    catch (Exception e)
                    {
                        Console.WriteLine("[ERROR] " + e.Message);
                    }
                }
            }
            return potrosnje;
        }
    }
}

```

Metoda PotrosnjaPoBrojilu() prikuplja podatke iz baze podataka za zadati id brojila i vraća listu potrošnji kao rezultat.

ConnectionUtils (Pomoćna komponenta)

Komponenta sa pomoćnim klasama za komunikaciju sa bazom podataka.

ConnectionParameters

```
0 references
public void Dispose()
{
    if(instance != null)
    {
        instance.Close();
        instance.Dispose();
    }
}

4 references
public static SqlConnection GetConnection()
{
    try
    {
        instance = new SqlConnection(connectionString);
        return instance;
    }
    catch(Exception e)
    {
        Console.WriteLine("Greska prilikom povezivanja sa bazom!\nError stack:" + e.Message);
    }
    return instance;
}
```

ParameterUtil

```
7 references
public static void AddParameter(IDbCommand command, string name, DbType type)
{
    IDataParameter parameter = command.CreateParameter();
    parameter.ParameterName = name;
    parameter.DbType = type;
    command.Parameters.Add(parameter);
}

0 references
public static void AddParameter(IDbCommand command, string name, DbType type, ParameterDirection direction)
{
    IDataParameter parameter = command.CreateParameter();
    parameter.ParameterName = name;
    parameter.DbType = type;
    parameter.Direction = direction;
    command.Parameters.Add(parameter);
}

1 reference
public static void AddParameter(IDbCommand command, string name, DbType type, int size)
{
    IDataParameter parameter = command.CreateParameter();
    parameter.ParameterName = name;
    parameter.DbType = type;
    parameter.Size = size;
    command.Parameters.Add(parameter);
}
```

```

0 references
public static void SetParameterValue(IDbCommand command, string name, Object value)
{
    DbParameter parameter = (DbParameter)command.Parameters[name];
    parameter.Value = value;
}

0 references
public static object GetParameterValue(IDbCommand command, string name)
{
    DbParameter parameter = (DbParameter)command.Parameters[name];
    return parameter.Value;
}

```

Napomena:

ConnectionString=Data Source=spasic.co.rs,25565;Initial
Catalog=ERS_projekat;User ID=ers_projekat;Password=radimo123.