SVEUČILIŠTE U DUBROVNIKU

ODJEL ZA ELEKTROTEHNIKU I RAČUNARSTVO

Studij Primijenjeno/poslovno računarstvo

**Projekt iz kolegija  
Objektno orijentirano programiranje**

Akademska godina 2020./2021.

**Podsjetnik za uzimanje lijekova**

Dokumentacija

Student:Katarina Musladin

Dubrovnik, 26.05.2021.g.

**Sadržaj**

[1 Uvod 2](#_Toc63021910)

[2 Arhitektura projekta 2](#_Toc63021916)

[3 Realizacija koda 4](#_Toc63021927)

[4 Tehnički detalji 7](#_Toc63021928)

[5 Zaključak 8](#_Toc63021928)

# Uvod

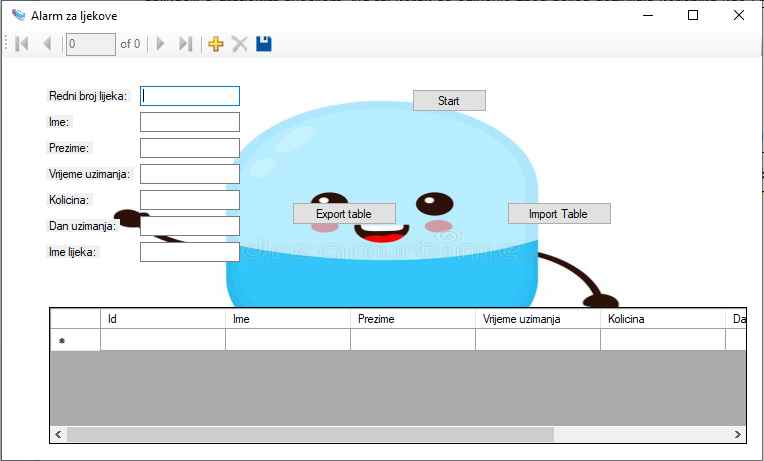
Kroz ovaj projekt prikazati će se kreiranje aplikacije koja služi kao podsjetnik za uzimanje lijekova. Samo rješenje bit će napisano u potpunosti u programskom jeziku C#, koji kao takav spada u skupinu objektno orijentiranih programskih jezika. Također, za realizaciju samog projekta biti će potrebno uključiti i dodatne pakete, koji se za C# programski jezik mogu pronaći u NuGet manageru. S obzirom na to kako aplikacija treba alarmirati informaciju o uzimanju lijekova potrebno je svakako voditi i evidenciju o osobama koje trebaju uzimati određene terapije kao i o vremenu, odnosno danu uzimanja. Za takvo što se koristi baza podataka u koju će biti spremljeni svi ti podaci. Sam projekt zamišljen je kroz upisivanje podataka o osobi u spomenutu bazu podatka koja se nalazi lokalno u projektu u koju se zadaju parametri kao što su vrijeme, dan te osoba koja mora uzimati terapiju. Program nakon pokretanja treba registrirati koje je vrijeme trenutno spremljeno u bazu podataka, te uspoređujući ga s trenutnim vremenom koje je lokalno na računalu odlučiti o tome da li je potrebno alarmirati korisnika.

## 2 Arhitektura projekta

Kako je već ranije rečeno, programski kod napisan je u programskom jeziku C# koji spada u skupinu objektno orijentiranih jezika. Kao takav kod se sastoji od nekoliko formi i klasa, a to su:

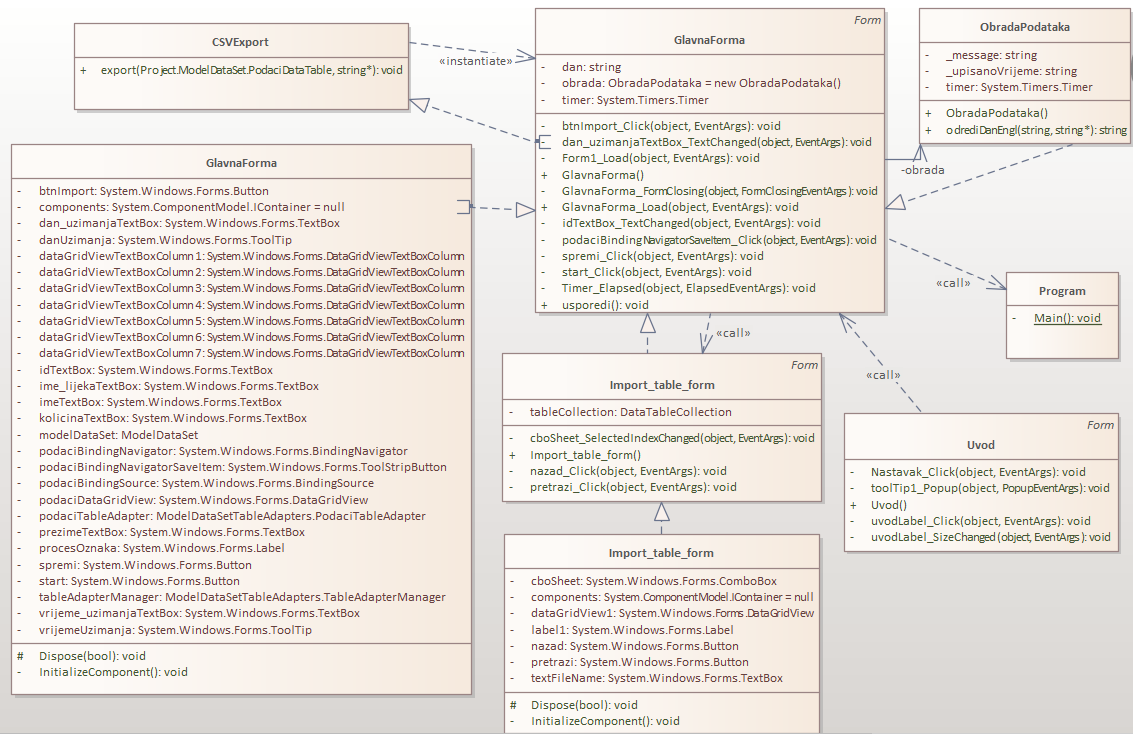
* CSVExport.cs
* ObradaPodataka.cs
* Program.cs
* GlavnaForma.cs
* Import\_table\_form.cs
* Uvod.cs

Prisustvo formi u ovom popisu govori o tome kako se radi o WinForm aplikaciji odnosno windows aplikaciji s grafičkim sučeljem. Na taj korak se odlučuje zbog boljeg doživljaja korisnika kao i praktičnosti korištenja. Izgled glavnog korisničkog sučelja prikazan je na slici 1:



Slika Izgled aplikacije

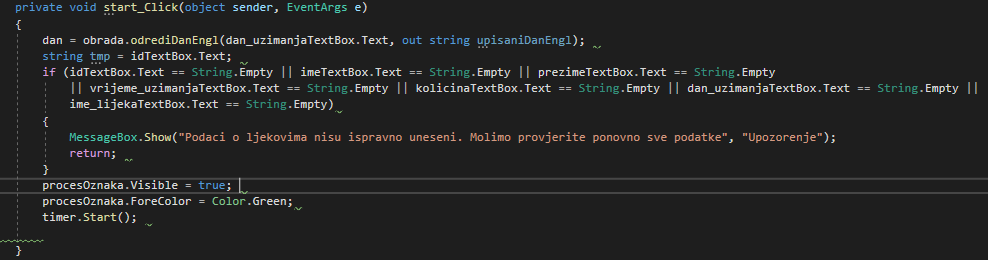
Izgled aplikacije sadržava bazu podataka kao i polja za upis u tu bazu. Osim toga postoje još gumbovi početak rada aplikacije, tj. gumb „Start“ gdje nakon pritiska spomenutog gumba program čita iz baze podataka upisano dan i vrijeme uzimanja terapije te vrši usporedbu s lokalnim danom i vremenom koje se trenutno nalazi na računalu. Jednom kada dođe do izjednačavanja tih vrijednosti uključuje se pisani i zvučni alarm na zaslonu koji korisnika upozorava kako je vrijeme za uzimanje terapije. Kod realizacije toga, svakako je potrebno obratiti pozornost na razliku u jezicima i načinu zapisivanja vremena u pojedinim formatima. Kako je taj problem riješen bit će detaljnije objašnjeno u poglavlju Realizacija koda. Osim spomenutog gumba, nalaze se još dva gumba koja su povezana s importom i exportom podataka iz aplikacije. Ta dva gumba daju mogućnost spremanja trenutnog sadržaja iz baze podataka lokalno na memoriju računala, kao i čitanje s računala. Zbog povremene nejasnosti delimitera kod .csv file-a, u ovom slučaju je odabran .xlsx file, odnosno Excel tablica. Uz sve navedeno postoje još dodatne mogućnosti vezane za bolji korisnički doživljaj u vidu raznih tooltip-ova, potvrde zatvaranja programa i slično o kojima neće biti pretjerano riječi u nastavku. Sama arhitektura kako je prethodno spomenuto sastoji se od nekoliko klasa i formi, ali se može generalizirati na sljedeći način: Sve klase se instanciraju unutar GlavneForme.cs, tj. objekti svih klasa se prave unutar te forme, a ta forma se instancira u formi Uvod.cs koja se pokreće u Program.cs. Podaci sadržani unutar GlavneForme.cs koji se nalaze u bazi podataka prosljeđuju se dalje u metode u pojedinim klasama, gdje one vrše potrebnu obradu podataka i vraćaju željeni parametar. Detalje toga moguće je vidjeti u poglavlju Realizacija koda. Za bolje razumijevanje izgleda te povezanosti klasa i formi prikazana je slika 2. Na toj slici moguće je uočiti veze između pojedinih dijelova.



Slika Grafički prikaz klasa

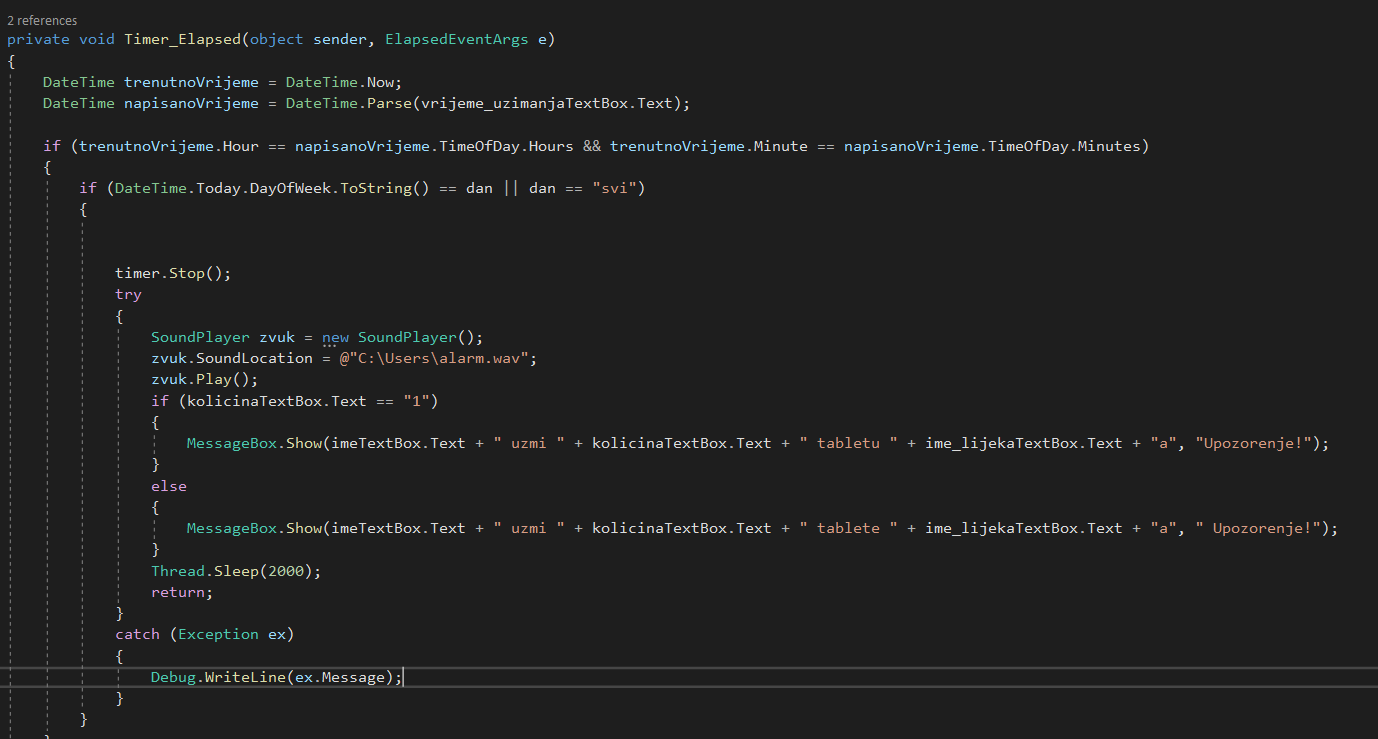
# Programska realizacija

Za početak realizacije ovog projekta potrebno je kreirati bazu podataka. Za ovaj slučaj odabrana je lokalna baza podataka, jer sadrži sve potrebne osobine koje su potrebne kako bi se spomenuti problem uspješno riješio. Zatim je potrebno unutar korisničkog sučelja složiti potrebne dijelove, kako je to već prikazano na Slici 1 te početi pisati kod. Glavnina koda smještena je unutar GlavneForme.cs s obzirom na to da ona služi kao premosnica koja povezuje sve klase, odnosno mjesto gdje se inicijaliziraju objekti svih klasa i pozivaju metode unutar njih. U nastavku na slici 3 prikazan je primjer poziva metode za alarmiranje korisnika koja se triggerira nakon pritiska gumba „Start“.



Slika Pokretanje brojača

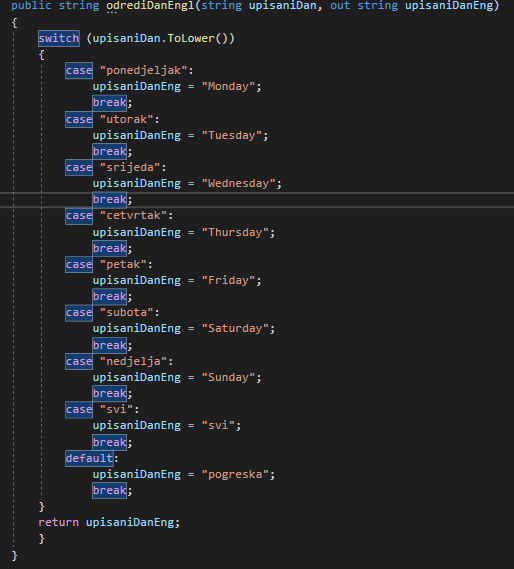
S prethodne slike vidljivo je kako je objekt klase ObradaPodataka.cs već kreiran ranije u kodu, te kako se naziva obrada. Unutar tog objekta poziva se metoda odrediDanEngl koja kao parametar zadaje dan uzimanja lijekova nakon čega se vrši provjera da li su svi podaci ispravno uneseni, i pokreće mjerač vremena. Taj mjerač je od izuzetnog značaja jer u metodi prikazanoj na slici 4 vidljivo je kako se on zaustavlja te je uz pomoć njega, trenutnog vremena na računalu kao i napisanog vremena u za to predviđenom prostoru unutar baze podataka omogućeno zvučno i pisano upozorenje o potrebnom uzimanju lijeka.



Slika Kod za alarmiranje korisnika

Sama ideja algoritma temelji se na sljedećem: Prvo se uzima trenutno vrijeme koje pokazuje računalo u nekom trenutku, zatim se uzima i vrijeme koje je zapisano u bazu podataka i prebacuje se u *DateTime*tip podataka da bi se moglo usporediti, nakon čega se usporedi trenutni broj sati i trenutni broj minuta. U ovom dijelu algoritma potrebno je paziti na načine zapisivanja vremena i datuma koji se međusobno razlikuju među kontinentima, pa čak i državama unutar istog kontinenta. Nakon toga provjerava se koji je dan unesen uz pomoć realizacije metode iz ObradaPodataka.cs (što se može vidjeti na slici 3), te ako su zadovoljeni svi dosadašnji uvjeti kreira se objekt SoundPlayer klase uz pomoć kojeg se reproducira zvuk, te se ispisuje pripadajuća poruka. Dodatni if uvjeti postavljeni su iz gramatičkih razloga, gdje se nastoji napisati pravilan padež prilikom ispisa poruke. Uz pomoć naredbe *Thread.Sleep(2000)*osigurava se da će vrijeme alarmiranja trajati minimalno 2 sekunde.

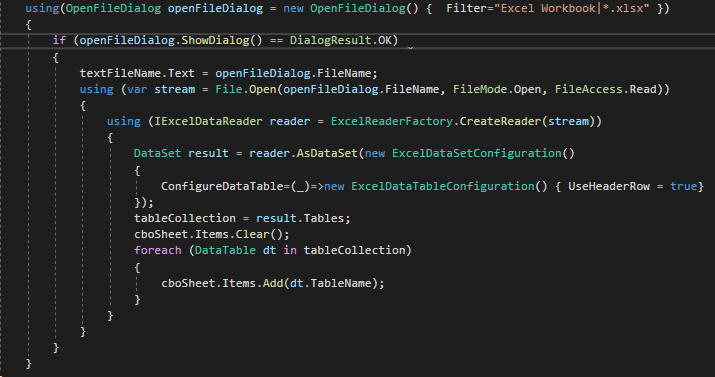
Kako je već ranije spomenuto iz klase ObradaPodataka.cs poziva se metoda odrediDanEngl. Izgled same metode moguće je vidjeti na slici 5:



Slika Određivanje dana na Engleskom jeziku

Algoritam metode je poprilično jednostavan te se ogleda u prijevodu određenog dana koji je napisan na izvornom jeziku u njegov ekvivalent na engleskom jeziku. Realizacija je izvedena uz pomoć switch case petlje, gdje je kao ulazni argument zadan upisani dan iz baze podataka koji je tipa string, a izlazni podatak je njegov engleski ekvivalent također u obliku stringa.

Osim već spomenutog potrebno je naglasiti još dijelove koda koji služe za export i import Excel tablice. Ovdje će biti prikazan samo import čiji kod se nalazi u Import\_table\_form.cs, dok export je ustvari samo njegov reverz smješten u klasi CSVExport.cs. Na slici 6 vidljiv je odsječak koda za import.



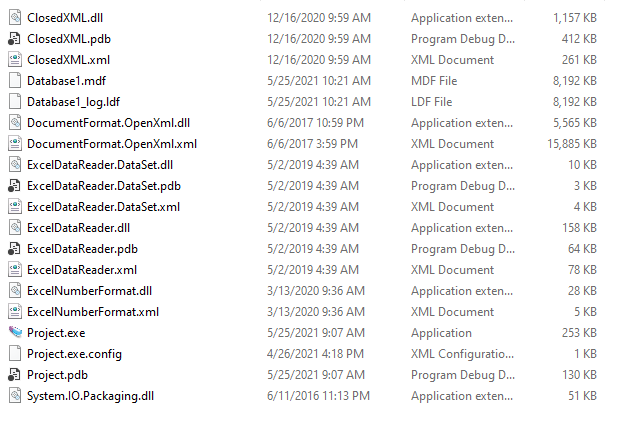
Slika Import excel tablice

Ključna riječ using predstavlja objekt korišten samo unutar tih zagrada, također novootvorenom prozoru nakon kreiranja objekta *openFileDialog*se postavlja filtar o mogućnosti učitavanja samo .xlsx tipa podataka. U nastavku se kreira objekt *reader* koji služi za kreiranje novog DataSet-a koji je nazvan *result* uz pomoć kojeg se unutar foreach petlje prolazi po svakom pojedinom DataTableu i dodaju se novi redci koji su zapisani unutar spomenutog .xlsx dokumenta.

Programska realizacija osim gore navedenih odsječaka koda sadrži još puno različitih implementacija o kojima sada neće biti riječ, ali prethodni su navedeni radi njihovog značaja, te su detaljno objašnjeni algoritmi njihovog funkcioniranja.

# 4 Tehnički detalji

Kako se ovdje radi o studentskom projektu sama instalacija ovog programskog rješenja neće biti omogućena. Umjesto toga ponuđene su dvije alternative u vidu pokretanja koda iz Visual studia, čime se builda aplikacija te ju je nakon toga moguće neometano koristiti. Druga mogućnost je unutar projekta otvoriti bin folder u kojemu se nalazi Debug folder. Unutar Debug foldera moguće je naći veći broj dokumenata među kojima jedan ima ekstenziju .exe. Pokretanjem tog dokumenta otvorit će se program koji je tada moguće koristiti. Prikaz izgleda Debug foldera vidljiv je na slici 7.



Slika Folder iz kojeg se pokreće aplikacija

Važno je napomenuti kako kod prve mogućnosti je bitno isključiti djelovanje antivirusa, jer u nekim slučajevima može dovesti do nepravilnog rada aplikacije, s obzirom na to kako ovu aplikaciju prepoznaje kao njemu nepoznat i sumnjiv kod.

# Zaključak

Kroz prethodnih nekoliko stranica prikazan je način na koji je razvijen projekt od njegovog početka do konačne verzije. Sama ideja projekta je vrlo zanimljiva i korisna, stoga ovaj projekt je svakako nešto što bi se i u budućnosti trebalo nastaviti razvijati, a za što su sada postavljeni temelji. Projekt se je temeljio na izgradnji lokalne baze podataka koja bi u nekoj idealnoj verziji sadržavala podatke o svim korisnicima i njihovim terapijama. Kroz nastavak napisane su dobro strukturirane metode koje upravljaju tim podacima. Metode su pisane u za to predviđenim klasama, te uz pomoć dijagrama klasa, omogućen je lakši uvid u program kao i nastavak nekog njegovog budućeg razvoja. Uz to kao dodatni podsjetnik na sve to služi i ova dokumentacija u kojoj su jasno definirani najvažniji algoritmi kao i dijelovi koda kojima su ti algoritmi realizirani.