A black and white logo

Description automatically generatedSVEUČILIŠTE/UNIVERZITET „VITEZ“

FAKULTET INFORMACIONIH TEHNOLOGIJA

STUDIJ I CIKLUSA; GODINA STUDIJA III

SMJER: INFORMACIJSKE TEHNOLOGIJE

**Zulka Musić**

**STOMATOLOŠKA ORDINACIJA – C# KONZOLNA APLIKACIJA**

**SEMINARSKI RAD**

Travnik, 24.12.2024. godine

A black and white logo

Description automatically generatedSVEUČILIŠTE/UNIVERZITET „VITEZ“

FAKULTET INFORMACIONIH TEHNOLOGIJA

STUDIJ I CIKLUSA; GODINA STUDIJA III

SMJER: INFORMACIJSKE TEHNOLOGIJE

**STOMATOLOŠKA ORDINACIJA – C# KONZOLNA APLIKACIJA**

**SEMINARSKI RAD**

IZJAVA: Ja **Zulka Musić**, student Sveučilišta/Univerziteta „VITEZ“, Indeks broj: **390-24/RIIT** odgovorno i uz moralnu i akademsku odgovornost izjavljujem da sam ovaj rad izradio potpuno samostalno uz korištenje citirane literature i pomoć predmetnog profesora.

STUDENT: Zulka Musić

PREDMET: Viši programski jezici

MENTOR: prof.dr.sc. Selver Pepić

ASISTENT: Admir Sivro

**SADRŽAJ**

[1. UVOD 1](#_Toc185887999)

[1.1. PROBLEM, PREDMET I OBJEKT ISTRAŽIVANJA 1](#_Toc185888000)

[1.2. SVRHA I CILJEVI ISTRAŽIVANJA 1](#_Toc185888001)

[1.3. RADNA HIPOTEZA I POMOĆNE HIPOTEZE 2](#_Toc185888002)

[1.4. NAUČNE METODE 2](#_Toc185888003)

[1.5. STRUKTURA RADA 3](#_Toc185888004)

[2. STOMATOLOŠKA ORDINACIJA – APLIKACIJA 4](#_Toc185888005)

[2.1. FUNKCIONALNOSTI APLIKACIJE 4](#_Toc185888006)

[2.2. KORIŠTENJE PROGRAMA 4](#_Toc185888007)

[2.3. DETALJI IMPLEMENTACIJE 5](#_Toc185888008)

[3. FAZE PROGRAMIRANJA 7](#_Toc185888009)

[3.1. ANALIZA ZAHTJEVA 7](#_Toc185888010)

[3.2. DIZAJN 8](#_Toc185888011)

[3.3. IMPLEMENTACIJA 9](#_Toc185888012)

[3.4. TESTIRANJE 10](#_Toc185888013)

[3.5. ODRŽAVANJE 10](#_Toc185888014)

[4. DETALJNO OBJAŠNJENJE KLJUČNIH DIJELOVA 11](#_Toc185888015)

[5. KOMPLETAN CODE PROGRAMA 16](#_Toc185888016)

[6. OUTPUT PROGRAMA 25](#_Toc185888017)

[7. PROGRAMSKE KONSTRUKCIJE I ELEMENTI 29](#_Toc185888018)

[8. ZAKLJUČAK 31](#_Toc185888019)

# UVOD

## PROBLEM, PREDMET I OBJEKT ISTRAŽIVANJA

U današnjem vremenu, organizacija i upravljanje terminima u stomatološkim ordinacijama predstavlja značajan izazov. Tradicionalni pristupi vođenju evidencije često uključuju papirne kartone ili jednostavne elektronske tabele koje ne pružaju dovoljnu fleksibilnost, niti su dovoljno intuitivne za osoblje i pacijente. Problem koji se istražuje odnosi se na kreiranje softverskog rješenja koje omogućava efikasno upravljanje uslugama i terminima u stomatološkim ordinacijama.

Predmet istraživanja ovog rada je razvoj aplikacije za upravljanje terminima u stomatološkim ordinacijama, dok je objekt istraživanja proces unapređenja organizacije usluga u zdravstvenom sektoru kroz primjenu informacionih tehnologija.

Code programa je postavljen na GitHub-u:

<https://github.com/Zulka12/Stomatoloska_ordinacija/blob/main/Stomatoloska_ordinacija.cs>

## SVRHA I CILJEVI ISTRAŽIVANJA

Svrha ovog istraživanja je unapređenje svakodnevnih operacija u stomatološkim ordinacijama kroz razvoj softverskog alata koji omogućava jednostavno i brzo zakazivanje termina, pregled dostupnih usluga, kao i upravljanje postojećim evidencijama.

Ciljevi istraživanja su:

1. Dizajn i implementacija funkcionalnog sistema za upravljanje uslugama i terminima.
2. Omogućavanje personalizovanog pristupa pacijentima putem specifičnih opcija kao što su pregled i otkazivanje termina.
3. Unapređenje efikasnosti osoblja kroz automatizaciju procesa dodavanja usluga i brisanja termina.
4. Testiranje sistema kako bi se osigurala njegova tačnost, pouzdanost i intuitivnost.

## RADNA HIPOTEZA I POMOĆNE HIPOTEZE

Radna hipoteza: Uvođenje softverskog sistema za upravljanje uslugama i terminima u stomatološkoj ordinaciji poboljšale efikasnost organizacije rada i zadovoljstvo korisnika.

Pomoćne hipoteze:

* Pomoću automatizovanog sistema, osoblje će uštediti vrijeme prilikom vođenja evidencije.
* Pacijenti će smanjiti broj grešaka prilikom zakazivanja i otkazivanja termina.
* Lista usluga i cijena biće jasno dostupna, što će povećati transparentnost usluga ordinacije.
* Korisnici će smatrati interfejs aplikacije jednostavnim i intuitivnim za korištenje.

## NAUČNE METODE

istraživanju se koriste sljedeće metode:

* Analitička metoda: Analizirani su postojeći sistemi za upravljanje terminima kako bi se identifikovale prednosti i nedostaci.
* Konstruktivna metoda: Razvijen je model aplikacije zasnovan na identifikovanim potrebama korisnika.
* Eksperimentalna metoda: Program je testiran u kontrolisanim uslovima sa simulacijama scenarija stvarne upotrebe.
* Komparativna metoda: Upoređeni su rezultati prije i poslije implementacije aplikacije.

## STRUKTURA RADA

Struktura seminarskog rada je usklađena sa Uputstvom za pisanje seminarskog rada na prvom ciklusu studija kao i temi seminarskog rada. On sadrži osam poglavlja.

* Prvo poglavlje, Uvod, sadrži pet pod poglavlja:
* Problem, predmet i objekt isprašivanja,
* Svrha i ciljevi istraživanja,
* Radna hipoteza i pomoćne hipoteze,
* Naučne metode,
* Struktura rada.
* Drugo poglavlje, Stomatološka ordinacija – aplikacija, sadrži tri pod poglavlja:
* Funkcionalnosti aplikacije,
* Korištenje programa,
* Detalji implementacije.
* Treće poglavlje, Faze programiranja, sadrži pet pod poglavlja:
* Analiza zahtjeva,
* Dizajn,
* Implementacija,
* Testiranje,
* Održavanje.
* Četvrto poglavlje, Detaljno objašnjenje ključnih dijelova;
* Peto poglavlje, Kompletan code programa;
* Šesto poglavlje, Output programa;
* Sedmo poglavlje, Programske konstrukcije i elementi;
* Osmo poglavlje, Zaključak, daje odgovore na postavljene hipoteze.

# STOMATOLOŠKA ORDINACIJA – APLIKACIJA

## FUNKCIONALNOSTI APLIKACIJE

Aplikacija predstavlja digitalnu platformu za upravljanje terminima i uslugama stomatološke ordinacije. Njene glavne funkcionalnosti uključuju:

1. Pregled ponude usluga – Korisnici mogu pregledati dostupne stomatološke usluge i njihove cijene putem intuitivnog sučelja.
2. Zakazivanje termina – Omogućeno je jednostavno zakazivanje termina, pri čemu korisnik unosi svoje ime, odabire željenu uslugu, te datum i vrijeme termina.
3. Pregled zakazanih termina – Korisnici i osoblje mogu pregledati listu svih zakazanih termina, uključujući detalje poput imena pacijenta, odabrane usluge i vremena termina.
4. Otkazivanje ili brisanje termina – Korisnici mogu otkazati vlastite termine, dok osoblje ima mogućnost brisanja termina prema potrebi.
5. Dodavanje novih usluga – Osoblje može proširiti ponudu usluga dodavanjem novih stavki uz specifikaciju cijene.

## KORIŠTENJE PROGRAMA

Za pacijente

* Nakon pokretanja aplikacije, pacijent bira opciju za pregled ponude usluga ili zakazivanje termina.
* Za zakazivanje, potrebno je unijeti lične podatke, odabrati uslugu te unijeti željeni datum i vrijeme termina.
* Ako je potrebno, pacijent može otkazati već zakazan termin unosom relevantnih informacija.

Za osoblje

* Osoblje ima pristup dodatnim opcijama kao što su dodavanje novih usluga, pregled svih zakazanih termina, te brisanje termina u slučaju potrebe.
* Sve promjene koje osoblje napravi odmah se reflektiraju u sistemu kako bi informacije bile ažurne.

## DETALJI IMPLEMENTACIJE

1. Tehnološki okvir – Aplikacija je implementirana na programskom jeziku C# koristeći .NET okvir. Grafički interfejs nije implementiran, ali se aplikacija koristi putem konzole.
2. Struktura koda

* Kod je podijeljen u klase za bolje organiziranje i održavanje:
* StomatoloskaOrdinacija: Upravljanje uslugama i terminima.
* Termin: Model koji opisuje pojedinačni termin sa svojstvima za ime pacijenta, uslugu i vrijeme.
* Program: Ulazna tačka aplikacije s logikom za upravljanje korisničkim unosima.

1. Funkcionalnosti koda

* Dodavanje usluga: Implementirano pomoću metode koja dodaje uslugu u rječnik samo ako ne postoji duplikat.
* Zakazivanje termina: Lista objekata klase Termin služi za pohranu zakazanih termina. Metoda provjerava validnost unosa prije dodavanja termina.
* Pregled i brisanje termina: Implementirano pomoću iteracije kroz listu termina, čime se osigurava brz i intuitivan pristup informacijama.

1. Korisnički unos i validacija – Za svaku funkcionalnost koristi se provjera korisničkih unosa kako bi se spriječile greške, poput unosa pogrešnog datuma ili vremena.
2. Fleksibilnost i proširivost – Sistem je dizajniran tako da omogućava lahko dodavanje novih funkcionalnosti i održavanje postojećih, uz minimalne izmjene u kodu.

# FAZE PROGRAMIRANJA

Razumijevanje i objašnjavanje programa kroz sve faze programiranja može pomoći da se bolje razumije kako se program razvija i funkcioniše. Program je razvijen u nekoliko faza:

* Analiza zahtjeva
* Dizajn
* Implementacija
* Testiranje
* Održavanje

## ANALIZA ZAHTJEVA

Ova faza se fokusira na prikupljanje i razumjevanje potreba korisnika i definisanje šta program treba da radi.

Ciljevi:

* Kreirati aplikaciju za stomatološku ordinaciju.
* Omogućiti pacijentima:
  + Pregled dostupnih usluga i njihovih cijena.
  + Zakazivanje termina za određenu uslugu.
  + Pregled zakazanih termina.
  + Otkazivanje termina.
* Omogućiti osoblju:
  + Dodavanje novih usluga i njihovih cijena.
  + Pregled zakazanih termina svih pacijenata.
  + Brisanje termina pacijenata.
  + Zakazivanje termina za pacijente.
* Napraviti program koji je intuitivan i jednostavan za korištenje.

Funkcionalni zahtjevi:

* Sistem treba podržavati unos i upravljanje terminima.
* Lista usluga treba da bude dostupna i lako proširiva.
* Mogućnost interakcije kroz tekstualni meni.

## DIZAJN

U ovoj fazi se pravi plan kako će program biti implementiran. Ovo uključuje strukturu programa, strukture podataka, interfejs i interakcije.

Arhitektura:

* Klase:
  + StomatoloskaOrdinacija: Glavna klasa za upravljanje podacima o uslugama i terminima.
  + Termin: Model klase za čuvanje informacija o terminima.
  + Program: Glavna klasa koja pruža korisnički interfejs kroz tekstualni meni.
* Strukture podataka:
  + Dictionary<string, double> za čuvanje usluga i njihovih cijena.
  + List<Termin> za čuvanje zakazanih termina.

Interakcije:

* Korisnici biraju svoju ulogu (pacijent ili osoblje).
* Na osnovu izbora, prikazuju se relevantne opcije (zakazivanje termina, pregled, dodavanje usluga itd.).

Dizajn metoda:

* Za pacijente:
  + Pregled usluga (IspisiUsluge).
  + Zakazivanje termina (ZakaziTermin).
  + Pregled zakazanih termina (PregledajTermine).
  + Otkazivanje termina (OtkaziTermin).
* Za osoblje:
  + Dodavanje novih usluga (DodajUslugu).
  + Brisanje termina (ObrisiTermin).

## IMPLEMENTACIJA

Program se razvija prema planu iz faze dizajna.

* Definicija klasa: Kreirane su klase StomatoloskaOrdinacija i Termin.
* Inicijalizacija podataka: Inicijalizuje se lista usluga sa njihovim cijenama.
* Metode: Svaka funkcionalnost je razdvojena u metode koje su jednostavne za razumjevanje i ponovnu upotrebu.
* Korisnički interfejs: Implementiran kroz tekstualni meni sa unosom brojeva.

## TESTIRANJE

U ovoj fazi se program provjerava kako bi se osiguralo da ispravno funkcioniše. Testiraju se funkcionalnosti: pregleda usluga, zakazivanja termina, otkazivanja termina, dodavanje usluga i testiranje izuzetaka.

## ODRŽAVANJE

Faza održavanja softverskog sistema predstavlja jednu od ključnih faza životnog ciklusa razvoja softvera. Nakon što je softver uspješno implementiran i pušten u rad, njegova upotreba često otkriva nove zahtjeve, greške ili prilike za unapređenje. Održavanje osigurava da softverski sistem ostane funkcionalan, pouzdan i prilagođen potrebama korisnika tokom vremena.

# DETALJNO OBJAŠNJENJE KLJUČNIH DIJELOVA

Definicija klase StomatoloskaOrdinacija

****

* Dictionary<string, double> usluge: Sadrži nazive usluga kao ključeve i njihove cijene kao vrijednosti.
* List<Termin> termini: Lista objekata Termin koja čuva zakazane termine.

A computer screen with text on it

Description automatically generated

* Konstruktor inicijalizuje listu usluga sa njihovim cijenama i praznu listu termina.
* Automatski se poziva kada se kreira nova instanca klase.
* Zadatak mu je da inicijalizuje dvije kolekcije usluge i termini.
* Konstruktor osigurava da ordinacija u startu ima definisane usluge i da je spremna za zakazivanje termina.

Dodavanje usluga

A black screen with white text

Description automatically generated

* Provjerava da li usluga već postoji u rječniku usluge pomoću if uslova.
* Ako ne postoji, dodaje uslugu u kolekciju i ispisuje poruku o uspjehu.
* Ako postoji, ispisuje poruku da je usluga već dodata.
* Metoda omogućava fleksibilnost i prilagođavanje usluga koje ordinacija nudi, bez potrebe za promjenom koda.

Ispis svih usluga

A black screen with text

Description automatically generated

* Ispisuje sve usluge sa njihovim cijenama.

Zakazivanje termina

A screen shot of a computer code

Description automatically generated

* Provjerava da li tražena usluga postoji u rječniku pomoću if uslova.
* Ako ne postoji ispisuje grešku i prekida izvršavanje sa return.
* Ako postoji, kreira se novi objekat Termin i dodaje se u listu termini i ispisuje potvrdu o zakazivanju.
* Omogućava pacijentima i osoblju da efikasno upravljaju terminima.

Pregled termina

A computer screen with text

Description automatically generated

* Ako nema termina, ispisuje poruku da nema zakazanih termina, pomoću if uslova.
* Ako postoji termin prolazi kroz svaki termin u listi termini pomoću foreach petlje i ispisuje detalje.
* Omogućava korisnicima pregled zakazanih termina na organizovan način.
* Foreach petlja prolazi kroz svaki element liste termini i omogućava pristup svakom terminu zasebno.

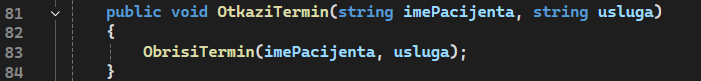
Brisanje termina

A screen shot of a computer code

Description automatically generated

* Prvo traži termin koji odgovara pacijentu i usluzi pomoću metode Find.
* Find traži prvi element u listi koji zadovoljava uslov.
* Ako termin postoji kroz if uslov se uklanja iz liste i ispisuje se potvrda.
* Ako ne postoji ispisuje se poruka o grešci.
* Omogućava lahko upravljanje greškama u rasporedu i otkazivanje termina.

Otkazivanje termina



* Jednostavno poziva metodu ObrisiTermin.

Definicija klase Termin

A screen shot of a computer code

Description automatically generated

* Model za čuvanje informacija o terminu.
* Predstavlja jedan zakazani termin.
* Omogućava jednostavno organizovanje i rukovanje podacima o zakazanim terminima.

Klasa Program

* Centralna tačka za interakciju s korisnikom.

Metoda Main

* Glavni meni:

A screen shot of a computer

Description automatically generated

* Poziva metode za upravljanje ulogama (pacijenti ili osoblje).

Unos izbora

A screen shot of a computer code

Description automatically generated

* Osigurava da korisnik unosi validan broj.

# KOMPLETAN CODE PROGRAMA

using System.Globalization;

class StomatoloskaOrdinacija

{

private Dictionary<string, double> usluge;

private List<Termin> termini;

public StomatoloskaOrdinacija()

{

usluge = new Dictionary<string, double>

{

{ "preventivni pregled", 30.00 },

{ "punjenje zuba", 50.00 },

{ "vadjenje zuba", 70.00 },

{ "izbjeljivanje zuba", 100.00 }

};

termini = new List<Termin>();

}

// Dodavanje novih usluga

public void DodajUslugu(string naziv, double cijena)

{

if (!usluge.ContainsKey(naziv))

{

usluge[naziv] = cijena;

Console.WriteLine($"Usluga '{naziv}' je uspješno dodata sa cijenom {cijena.ToString("F2", CultureInfo.InvariantCulture)} KM.");

}

else

{

Console.WriteLine($"Usluga '{naziv}' već postoji u ponudi.");

}

}

// Ispis svih usluga

public void IspisiUsluge()

{

Console.WriteLine("\nPonuda usluga i cijena:");

foreach (var usluga in usluge)

{

Console.WriteLine($"{usluga.Key} - {usluga.Value.ToString("F2", CultureInfo.InvariantCulture)} KM");

}

}

// Zakazivanje termina

public void ZakaziTermin(string imePacijenta, string usluga, DateTime datumVrijeme)

{

if (!usluge.ContainsKey(usluga))

{

Console.WriteLine($"Usluga '{usluga}' nije pronađena.");

return;

}

termini.Add(new Termin(imePacijenta, usluga, datumVrijeme));

Console.WriteLine($"Termin za uslugu '{usluga}' je zakazan za {imePacijenta} na datum {datumVrijeme}.");

}

// Pregled termina

public void PregledajTermine()

{

if (termini.Count == 0)

{

Console.WriteLine("Nema zakazanih termina.");

}

else

{

Console.WriteLine("\nZakazani termini:");

foreach (var termin in termini)

{

Console.WriteLine($"Pacijent: {termin.ImePacijenta}, Usluga: {termin.Usluga}, Datum i vrijeme: {termin.DatumVrijeme}");

}

}

}

// Brisanje termina

public void ObrisiTermin(string imePacijenta, string usluga)

{

Termin terminZaBrisanje = termini.Find(t => t.ImePacijenta == imePacijenta && t.Usluga == usluga);

if (terminZaBrisanje != null)

{

termini.Remove(terminZaBrisanje);

Console.WriteLine($"Termin za uslugu '{usluga}' za pacijenta '{imePacijenta}' je obrisan.");

}

else

{

Console.WriteLine($"Termin za uslugu '{usluga}' za pacijenta '{imePacijenta}' nije pronađen.");

}

}

// Otkazivanje termina (za pacijente)

public void OtkaziTermin(string imePacijenta, string usluga)

{

ObrisiTermin(imePacijenta, usluga);

}

}

class Termin

{

public string ImePacijenta { get; }

public string Usluga { get; }

public DateTime DatumVrijeme { get; }

public Termin(string imePacijenta, string usluga, DateTime datumVrijeme)

{

ImePacijenta = imePacijenta;

Usluga = usluga;

DatumVrijeme = datumVrijeme;

}

}

class Program

{

static void Main()

{

StomatoloskaOrdinacija ordinacija = new StomatoloskaOrdinacija();

while (true)

{

Console.WriteLine("\nDobrodošli u stomatološku ordinaciju!");

Console.WriteLine("Odaberite svoju ulogu:");

Console.WriteLine("1. Pacijent");

Console.WriteLine("2. Osoblje");

Console.WriteLine("0. Izlaz");

int izbor = UnosIzbora();

if (izbor == 0) break;

switch (izbor)

{

case 1:

IzborPacijenta(ordinacija);

break;

case 2:

IzborOsoblja(ordinacija);

break;

default:

Console.WriteLine("Nepoznat izbor. Molimo pokušajte ponovo.");

break;

}

}

Console.WriteLine("Hvala na korištenju stomatološke ordinacije. Doviđenja!");

}

static int UnosIzbora()

{

while (true)

{

try

{

return int.Parse(Console.ReadLine());

}

catch (FormatException)

{

Console.WriteLine("Pogrešan unos. Molimo unesite broj.");

}

}

}

static void IzborPacijenta(StomatoloskaOrdinacija ordinacija)

{

Console.WriteLine("\nOpcije za pacijenta:");

Console.WriteLine("1. Pregled usluga i cijena");

Console.WriteLine("2. Zakazivanje termina");

Console.WriteLine("3. Pregled zakazanih termina");

Console.WriteLine("4. Otkazivanje termina");

Console.WriteLine("0. Povratak na glavni meni");

int izbor = UnosIzbora();

switch (izbor)

{

case 1:

ordinacija.IspisiUsluge();

break;

case 2:

ZakaziTerminPacijenta(ordinacija);

break;

case 3:

ordinacija.PregledajTermine();

break;

case 4:

OtkaziTerminPacijenta(ordinacija);

break;

case 0:

break;

default:

Console.WriteLine("Nepoznat izbor.");

break;

}

}

static void IzborOsoblja(StomatoloskaOrdinacija ordinacija)

{

Console.WriteLine("\nOpcije za osoblje:");

Console.WriteLine("1. Dodavanje novih usluga");

Console.WriteLine("2. Pregled zakazanih termina");

Console.WriteLine("3. Zakazivanje termina");

Console.WriteLine("4. Brisanje termina");

Console.WriteLine("0. Povratak na glavni meni");

int izbor = UnosIzbora();

switch (izbor)

{

case 1:

DodajUslugu(ordinacija);

break;

case 2:

ordinacija.PregledajTermine();

break;

case 3:

ZakaziTerminOsoblja(ordinacija);

break;

case 4:

ObrisiTerminOsoblja(ordinacija);

break;

case 0:

break;

default:

Console.WriteLine("Nepoznat izbor.");

break;

}

}

static void ZakaziTerminPacijenta(StomatoloskaOrdinacija ordinacija)

{

Console.Write("Unesite svoje ime: ");

string imePacijenta = Console.ReadLine();

Console.Write("Unesite naziv usluge: ");

string usluga = Console.ReadLine();

Console.Write("Unesite datum termina (yyyy-MM-dd): ");

DateTime datum = DateTime.Parse(Console.ReadLine());

Console.Write("Unesite vrijeme termina (HH:mm): ");

TimeSpan vrijeme = TimeSpan.Parse(Console.ReadLine());

DateTime datumVrijeme = datum.Add(vrijeme);

ordinacija.ZakaziTermin(imePacijenta, usluga, datumVrijeme);

}

static void OtkaziTerminPacijenta(StomatoloskaOrdinacija ordinacija)

{

Console.Write("Unesite svoje ime: ");

string imePacijenta = Console.ReadLine();

Console.Write("Unesite naziv usluge za otkazivanje: ");

string usluga = Console.ReadLine();

ordinacija.OtkaziTermin(imePacijenta, usluga);

}

static void DodajUslugu(StomatoloskaOrdinacija ordinacija)

{

Console.Write("Unesite naziv usluge: ");

string naziv = Console.ReadLine();

Console.Write("Unesite cijenu usluge: ");

double cijena = double.Parse(Console.ReadLine(), CultureInfo.InvariantCulture);

ordinacija.DodajUslugu(naziv, cijena);

}

static void ZakaziTerminOsoblja(StomatoloskaOrdinacija ordinacija)

{

ZakaziTerminPacijenta(ordinacija);

}

static void ObrisiTerminOsoblja(StomatoloskaOrdinacija ordinacija)

{

Console.Write("Unesite ime pacijenta: ");

string imePacijenta = Console.ReadLine();

Console.Write("Unesite naziv usluge: ");

string usluga = Console.ReadLine();

ordinacija.ObrisiTermin(imePacijenta, usluga);

}

}

# **OUTPUT PROGRAMA**

Prikaz output-a za pacijenta:

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generatedA screenshot of a computer program

Description automatically generated

Prikaz output-a za osoblje:

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

# PROGRAMSKE KONSTRUKCIJE I ELEMENTI

Aplikacija koristi razne programske konstrukcije i elemente koji omogućavaju njenu funkcionalnost.

1. Namespaces – Namespaces (prostor imena) organizuju kod u logičke cjeline i sprečavaju konflikt imena između različitih dijelova koda.

* Koristimo System, System.Collections.Generic, i System.Globalization. S tim što kod mene u Visual Studio kad pokrenem kod System i System.Collections.Generic skloni i smatra kao da to podrazumijevamo dok kod pišemo.
* Na primjer, System.Globalization omogućava rad sa različitim kulturama i formatima.
* System.Collections.Generic omogućava rad sa generičkim kolekcijama poput Dictionary i List.

1. Klase – Klase su osnovni građevinski blokovi objektno orijentisanog programiranja. One objedinjuju podatke (polja) i ponašanje (metode) u jednu strukturu.

* StomatoloskaOrdinacija: Glavna klasa koja sadrži funkcionalnosti za upravljanje uslugama i terminima.
* Termin: Klasa koja predstavlja model jednog termina sa svojstvima za ime pacijenta, uslugu i datum/vrijeme.
* Program: Sadrži ulaznu tačku aplikacije (Main metoda) i logiku korisničkog interfejsa.

1. Kolekcije – Kolekcije su strukture podataka koje omogućavaju skladištenje više elemenata.

* Dictionary<string, double>: Koristi se za skladištenje usluga i njihovih cijena, gdje je ključ naziv usluge, a vrijednost je cijena.
* List<Termin>: Lista objekata klase Termin koja omogućava praćenje zakazanih termina.

1. Metode – Metode su skup instrukcija koje obavljaju određenu funkciju.

* DodajUslugu: Dodaje novu uslugu u ponudu.
* ZakaziTermin: Zakazuje termin za pacijenta.
* ObrisiTermin: Briše termin prema zadatim kriterijumima.
* PregledajTermine: Prikazuje sve zakazane termine.

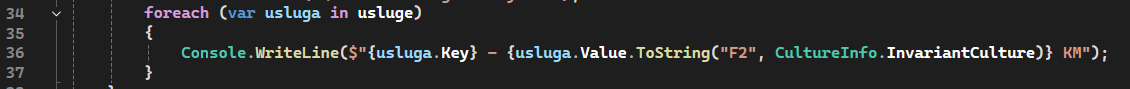
1. Petlje – Petlje omogućavaju ponavljanje skupa instrukcija dok je određeni uslov ispunjen.

* foreach petlja: Koristi se za iteraciju kroz kolekcije, poput ispisa svih usluga ili termina.
* while petlja: Koristi se za stalno prikazivanje menija dok korisnik ne izabere izlaz.

1. Izuzeci (Exceptions) – Izuzeci su mehanizam za upravljanje greškama tokom izvršavanja programa.

* Metoda UnosIzbora koristi try-catch blok za obradu grešaka prilikom unosa korisnika, poput pogrešnog formata broja.

1. String interpolacija – String interpolacija omogućava umetanje vrijednosti varijabli direktno unutar stringova koristeći {}.



1. Kultura (CultureInfo) – Kultura određuje kako se formati brojeva, datuma i vremena prikazuju.

Koristi se CultureInfo.InvariantCulture kako bi se osiguralo da se decimalni brojevi prikazuju sa tačkom (.) kao separatorom.



# ZAKLJUČAK

Razvoj aplikacije za upravljanje terminima i uslugama u stomatološkim ordinacijama predstavlja značajan doprinos modernizaciji zdravstvenih usluga. Cilj projekta bio je unaprijeđenje efikasnosti rada i pružanje bolje korisničke usluge kroz intuitivno rješenje koje omogućava jednostavno zakazivanje termina, pregled usluga i transparentno informisanje pacijenata.

Iako je aplikacija trenutno lokalno rješenje, budući razvoj može uključivati mrežnu podršku, integraciju s cloud platformama, personalizovane opcije za pacijente i unapređenje korisničkog interfejsa. Također, dodavanje analitičkih funkcionalnosti pružilo bi dodatnu vrijednost za upravljanje poslovanjem.

Ovaj projekat potvrđuje kako digitalizacija može značajno unaprijediti rad stomatoloških ordinacija, poboljšati dostupnost usluga i podići standard kvaliteta zdravstvene zaštite. Dalji razvoj aplikacije ima potencijal da postane ključni alat za modernizaciju malih zdravstvenih ustanova.

Hipoteze postavljene na samom početku su dokazane i potvrđene.