Aplikacija za upravljanje sustavom javne garaže i njenim korisnicima

Novosel, Ivor

VELEUČILIŠTE SUVREMENIH INFORMACIJSKIH TEHNOLOGIJA

STRUČNI PRIJEDIPLOMSKI STUDIJ INFORMACIJSKIH TEHNOLOGIJA

Ivor Novosel

ZAVRŠNI RAD

APLIKACIJA ZA UPRAVLJANJE SUSTAVOM JAVNE GARAŽE I NJENIM KORISNICIMA

Zagreb, listopada 2024.

Veleučilište suvremenih informacijskih tehnologija
10000 Zagreb, Ulica Vjekoslava Klaića 7

Studij:	Stručni prijediplomski studij informacijskih tehnologija smjer programiranje
Student:	**Ivor Novosel**
Matični broj:	2019007

Zadatak završnog rada

Predmet:	Programiranje u C#
Naslov:	**Aplikacija za upravljanje sustavom javne garaže i njenim korisnicima**
Zadatak:	Napraviti aplikaciju koja će pojednostavniti upravljanje javnom garažom. Aplikacija mora automatizirati proces propuštanja vozila i naplatu parkiranja za jednokratne korisnike i za pretplatnike. U okviru rada opisati korištene tehnologije te detaljno opisati procese i moguće probleme vezane za samu aplikaciju.
Mentor:	mr. sc. Julijan Šribar, v. pred.
Zadatak uručen kandidatu:	9.8.2024.
Rok za predaju rada:	17.10.2024.
Rad predan:	_____

Povjerenstvo:

Jurica Đurić, v. pred.	član predsjednik	_____
mr. sc. Julijan Šribar, v. pred.	mentor	_____
Mariza Maini, pred.	član	_____

SADRŽAJ

POPIS SLIKA

POPIS KODOVA

1. UVOD

Rijetko koji vozač se u današnje vrijeme nije susreo s parkiranjem vozila u nekoj od javno dostupnih parkirnih garaža u urbanim sredinama. Nekoć se ovaj proces odvijao isključivo uz kontrolu svakog elementa od strane djelatnika garaže: od dolaska pred ulaznu rampu, izdavanja karte s vremenom ulaska, podizanja ulazne rampe, obavljanja financijskih transakcija, do podizanja izlazne rampe te konačnog izlaska iz garaže.

Uzevši u obzir moderno doba informatizacije, cilj ovoga rada je automatizacija procesa parkiranja u javnoj garaži izradom *Windows Forms* aplikacije kao kontrolnog centra za upravljanje sustavom javne garaže i njezinim korisnicima.

Aplikacija kroz praćenje ulazaka i izlazaka vozila iz garaže zapisuje u bazu podataka sve potrebne podatke te ih obrađuje kako bi cjelokupan proces bio u potpunosti automatiziran. Vozilo dolazi do ulazne rampe i vozač uzima parkirnu kartu iz uređaja u sklopu rampe, a u slučaju mjesečnih pretplatnika, vozač umeće pretplatnu kartu. Sustav bilježi podatke o ulasku (vrijeme, registracijska oznaka, postojeći korisnik), podiže ulaznu rampu i odabire mjesto u garaži.

Nakon korištenja garaže korisnik umeće parkirnu kartu u uređaj za automatsku naplatu i plaća vrijeme koje je njegovo vozilo provelo u garaži. Vozilo dolazi do izlazne rampe i vozač umeće kartu u uređaj u sklopu rampe. Ovisno o podacima o naplati parkirne karte, sustav otvara izlaznu rampu i pušta korisnika da napusti garažu ili ga usmjerava da izvrši uplatu na uređaju za automatsku naplatu karata.

Administracija pretplatnika garaže se odvija u zasebnoj formi aplikacije u kojoj se korisnici i njihove pretplatničke karte mogu dodavati, uređivati i obrisati. Svakom pretplatniku se može dodati vlastita parkirna karta koju će koristiti prilikom ulaska i izlaska iz garaže.

Da bi mogao koristiti pretplatničku kartu, korisnik ju mora zatražiti i platiti kod zaposlenika garaže. Pritom bira vrstu mjesečne karte, ovisno o periodu dana unutar kojeg planira koristiti garažu, i izvrši plaćanje za odabranu vrstu karte za narednih mjesec dana. Djelatnik garaže dodaje korisnika u sustav unoseći njegove osobne podatke i pridodaje korisniku odabranu vrstu pretplatne parkirne karte s unesenim datumom do kojega će karta vrijediti.

Prilikom izrade aplikacije uzeti su u obzir elementi potrebni za autonoman rad garaže kao cjeline: koder parkirnih karata, senzori zauzeća parkirnih mjesta, automatska naplatna jedinica, ručna blagajna te ulazna i izlazna rampa. Navedeni uređaji su implementirani u samoj aplikaciji kao zasebni, modularni entiteti. U slučaju korištenja aplikacije u stvarnom objektu javne garaže, ovi se uređaji mogu jednostavno integrirati kako bi u suradnji s aplikacijom činili homogeni sustav.

U sljedećem poglavlju su opisane tehnologije koje čine razvojno okruženje korišteno za izradu aplikacije. Zatim slijedi poglavlje koje opisuje kako funkcionira i koje probleme donosi sustav javne garaže. Konačno poglavlje objašnjava kako je dizajnirano korisničko sučelje te implementaciju ključnih funkcionalnosti aplikacije.

2. KORIŠTENI ALATI I TEHNOLOGIJE

Za razvoj aplikacije korišteni su alati i tehnologije koji su omogućili efikasno i brzo razvijanje svih funkcionalnosti sustava. U nastavku je detaljno opisan svaki od korištenih elemenata razvojnog okruženja.

2.1. Visual Studio 2022

Visual Studio 2022 bio je ključan alat korišten tijekom cijelog razvojnog procesa. Kao integrirano razvojno okruženje (engl. *Integrated Development Environment, IDE*), Visual Studio pruža funkcionalnosti potrebne za pisanje, testiranje i debugiranje koda.

Napredni alati za debugiranje omogućuju detaljno praćenje izvršavanja koda te brzo otkrivanje i ispravljanje pogrešaka. Visual Studio je također pomogao u razvoju specifičnih funkcionalnosti ove aplikacije kroz svoje napredne dizajnerske alate. Dizajn korisničkog sučelja u Windows Forms okviru bio je značajno olakšan alatom vizualnog dizajna koji omogućuje povlačenje i ispuštanje (engl. *drag and drop*) kontrola na forme, brzo dodavanje rukovatelja događajima (engl. *event handler*) i prilagodbu svojstava kontrola. Ovaj vizualni pristup dizajnu korisničkog sučelja ubrzao je razvoj i osigurao dosljednost kroz cijelu aplikaciju.

Osim toga, Visual Studio omogućuje jednostavno upravljanje projektima i rješenjima (engl. *solution*), omogućujući organizaciju koda u različite projekte unutar jednog rješenja. Ovo je bilo posebno korisno za razdvajanje različitih slojeva aplikacije, kao što su poslovna logika, pristup podacima i korisničko sučelje. Integracija s alatom za kontrolu verzija Git je omogućila efikasno upravljanje verzijama i praćenje promjena u kodu.

Korištenje Visual Studija kao razvojnog okruženja omogućilo je stvaranje funkcionalne i efikasne aplikacije za upravljanje javnom garažom, osiguravajući visoku razinu kvalitete, dosljednosti i pouzdanosti kroz cijeli razvojni ciklus.

2.2. Programski jezik C#

Aplikacija je implementirana u programskom jeziku C#. Kao objektno orijentirani jezik, C# omogućuje modularnost, višekratnu uporabu koda i lakšu održivost aplikacije. C# je naročito prigodan za rad s .NET-om, što ga čini idealnim izborom za razvoj Windows Forms aplikacija.

Jedna od ključnih prednosti korištenja jezika C# je njegova jednostavna i lako razumljiva sintaksa. U kombinaciji s fleksibilnošću i mogućnostima jezika, to omogućuje brzo pisanje koda, bez gubitka na performansama ili funkcionalnosti. U ovom projektu, C# je korišten za implementaciju poslovne logike, manipulaciju podacima, komunikaciju s bazom podataka i interakciju s korisničkim sučeljem.

Za razliku od tradicionalnog pristupa u komunikaciji s bazom podataka putem SQL upita, komunikacija između C# koda i baze podataka uspostavljena je korištenjem Entity Frameworka (Kôd 1).

```
public async Task<List<Garage>> GetAllGaragesAsync()
{
    try
    {
        return await _context.Garages
            .Include(g => g.Levels)
                .ThenInclude(l => l.Spots)
            .Include(g => g.Tickets)
            .Include(g => g.Users)
                .ThenInclude(u => u.UserTicket)
                    .ThenInclude(ut => ut.TicketEvents)
            .AsSplitQuery()
            .ToListAsync();
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, "Error occurred while getting all
garages.");
        throw;
    }
}
```

Kôd 1. Dohvaćanje svih garaža iz baze podataka

Ovaj primjer prikazuje jednostavnu metodu za dohvaćanje svih garaža iz baze podataka. Korištenje Entity Frameworka omogućuje jednostavno i efikasno rukovanje podacima bez potrebe za ručnim pisanjem SQL upita. Na primjeru se također vidi kako C# omogućuje

korištenje ključnih riječi `async/await` za asinkrono izvođenje, te korištenje LINQ (engl. *Language Integrated Query*) sintakse za upite nad kolekcijama podataka.

Korištenje jezika C# u ovom projektu omogućilo je stvaranje robusne i efikasne aplikacije koja zadovoljava sve zahtjeve i omogućuje jednostavno proširenje i održavanje u budućnosti.

2.3. .NET

.NET je razvojni okvir koji pruža sveobuhvatan programski model za izradu aplikacija. U kontekstu ove aplikacije, .NET je bio temelj na kojem je cijeli sustav izgrađen, omogućujući jednostavno korištenje različitih tehnologija i alata koji su neophodni za razvoj funkcionalne aplikacije.

Jedna od ključnih komponenti .NET-a, korištena u ovom projektu, je biblioteka Windows Forms, koja je omogućila razvoj bogatog korisničkog sučelja. Windows Forms pruža niz kontrola i alata za brzo i jednostavno kreiranje vizualnih elemenata aplikacije. Korištenjem alata vizualnog dizajnera u Visual Studiju, kontrole su mogle na jednostavan način biti postavljane na forme, čime se značajno ubrzao proces razvoja sučelja.

Osim Windows Formsa, .NET je osigurao i osnovnu infrastrukturu za rad aplikacije. Common Language Runtime (CLR) omogućuje izvršavanje koda, pružajući osnovne funkcionalnosti kao što su prevođenje koda iz međujezika (engl. *Common Intermediate Language, CIL*) u strojni kod, automatski sustav za upravljanje memorijom (engl. *garbage collector*), upravljanje iznimkama i sigurnost. CLR osigurava da se aplikacija izvršava efikasno i pouzdano, upravljajući resursima sustava na optimalan način.

Ovaj razvojni okvir također podržava integraciju s različitim NuGet paketima koji su bili potrebni zbog svojih specifičnih funkcionalnosti i biblioteka neophodnih za razvoj i održavanje aplikacije. NuGet je upravitelj paketa koji olakšava preuzimanje, instalaciju i upravljanje bibliotekama treće strane (engl. *third party libraries*) direktno u projektu. Integracija NuGet paketa s Visual Studijom omogućila je jednostavno korištenje Entity Frameworka i njegovih funkcionalnosti prilikom izrade aplikacije.

.NET je omogućio stabilnu platformu za razvoj svih aspekata aplikacije, od korisničkog sučelja do poslovne logike i pristupa podacima.

2.4. Microsoft SQL Server

Microsoft SQL Server je upravljački sustav relacijskih baza podataka koji je korišten za ovu aplikaciju kako bi podaci o garažama, parkirnim mjestima, korisnicima i ostale relevantne informacije ostale pohranjene i nakon završetka rada aplikacije. SQL Server pruža pouzdanu platformu za pohranu i upravljanje podacima, što ga čini idealnim izborom za aplikacije koje zahtijevaju robustan sustav za upravljanje podacima.

Za komunikaciju s SQL Serverom korišten je Entity Framework, koji omogućuje preslikavanje (mapiranje) objekata aplikacije na tablice u bazi podataka. Entity Framework olakšava manipulaciju podacima uz pomoć objektno-orijentiranog pristupa, što smanjuje potrebu za ručnim pisanjem SQL upita.

SQL Server je optimiziran za visoke performanse i skalabilnost, što omogućuje aplikaciji da učinkovito upravlja rastućim brojem korisnika i podataka. Indeksi, transakcijska izolacija i optimizacija upita osiguravaju da aplikacija nesmetano radi čak i pod opterećenjem.

Redovite sigurnosne kopije i oporavak podataka ključni su za očuvanje podataka. SQL Server omogućuje automatizaciju sigurnosnih kopija podataka te pruža alate za oporavak podataka u slučaju neželjenih događaja poput kvarova sklopovlja ili korisničkih pogrešaka.

Korištenjem Microsoft SQL Servera kao temeljne baze podataka, osigurana je stabilnost, sigurnost i skalabilnost potrebna za uspješno upravljanje podacima o javnim garažama i korisnicima. Integracija s Entity Frameworkom pojednostavila je razvoj komunikacijskog sloja za pristup podacima, dok su napredne značajke SQL Servera osigurale visoku razinu performansi i sigurnosti aplikacije.

2.5. Entity Framework

Entity Framework (EF) je ORM (engl. *Object-Relational Mapping*) alat koji omogućuje da se s podacima, pomoću objektno-orijentiranog pristupa, radi kao s objektima u .NET aplikaciji, što značajno pojednostavljuje interakciju s bazom podataka.

Umjesto da se SQL upiti pišu direktno, definiraju se modeli podataka kao C# klase (entiteti), koje Entity Framework preslika (mapira) na tablice u bazi podataka. Ovaj pristup omogućuje

jednostavno izvođenje operacija CRUD (engl. *Create, Read, Update, Delete*) nad podacima pomoću objekata.

U ovoj aplikaciji za upravljanje javnom garažom, Entity Framework je korišten za sve operacije vezane uz pristup podacima. Definirane su klase poput `Garage`, `GarageUser`, `ParkingSpot` i `Ticket` koje su mapirane na odgovarajuće tablice u Microsoft SQL Server bazi podataka. Na primjer, klasa `Garage` (Kôd 2) sadrži svojstva kao što su: `Name`, `Levels`, `Users` i `TotalCapacity`, dok klasa `ParkingSpot` sadrži informacije o parkirnom mjestu u garaži.

```
public class Garage
{
    [Key]
    public int Id { get; private set; }
    [Required]
    public string Name { get; private set; }
    public List<Level> Levels { get; private set; }
    public int TotalCapacity { get; private set; }
    public List<GarageUser> Users { get; private set; }
    public List<Ticket> Tickets { get; private set; }
```

Kôd 2. Podatkovni članovi klase Garage

Klasa `Garage` se na jednostavan način preslikava u tablicu u bazi podataka (Kôd 3).

```
private void ConfigureGarageEntity(ModelBuilder modelBuilder)
{
    modelBuilder.Entity<Garage>()
        .HasMany(g => g.Levels)
        .WithOne(l => l.GarageRef)
        .HasForeignKey(l => l.GarageId);

    modelBuilder.Entity<Garage>()
        .HasMany(g => g.Tickets)
        .WithOne(t => t.GarageRef)
        .HasForeignKey(t => t.GarageId);

    modelBuilder.Entity<Garage>()
        .HasMany(g => g.Users)
        .WithOne(u => u.GarageRef)
        .HasForeignKey(u => u.GarageId);
}
```

Kôd 3. Kreiranje tablice u bazi podataka kroz kôd

Prednosti korištenja Entity Frameworka su:

- Pojednostavljene operacije nad podacima: Entity Framework automatski generira SQL upite na temelju LINQ (engl. *Language-Integrated Query*) sintakse, čime se eliminira potreba za ručnim pisanjem SQL upita.
- Manja ovisnost o bazi podataka: zahvaljujući apstrakciji podatkovnog sloja koju omogućuje Entity Framework, programer može raditi s modelima podataka bez detaljnog poznavanja strukture baze podataka.
- Automatsko praćenje promjena (engl. *Change Tracking*): Entity Framework automatski prati promjene na objektima i sinkronizira ih s bazom podataka.

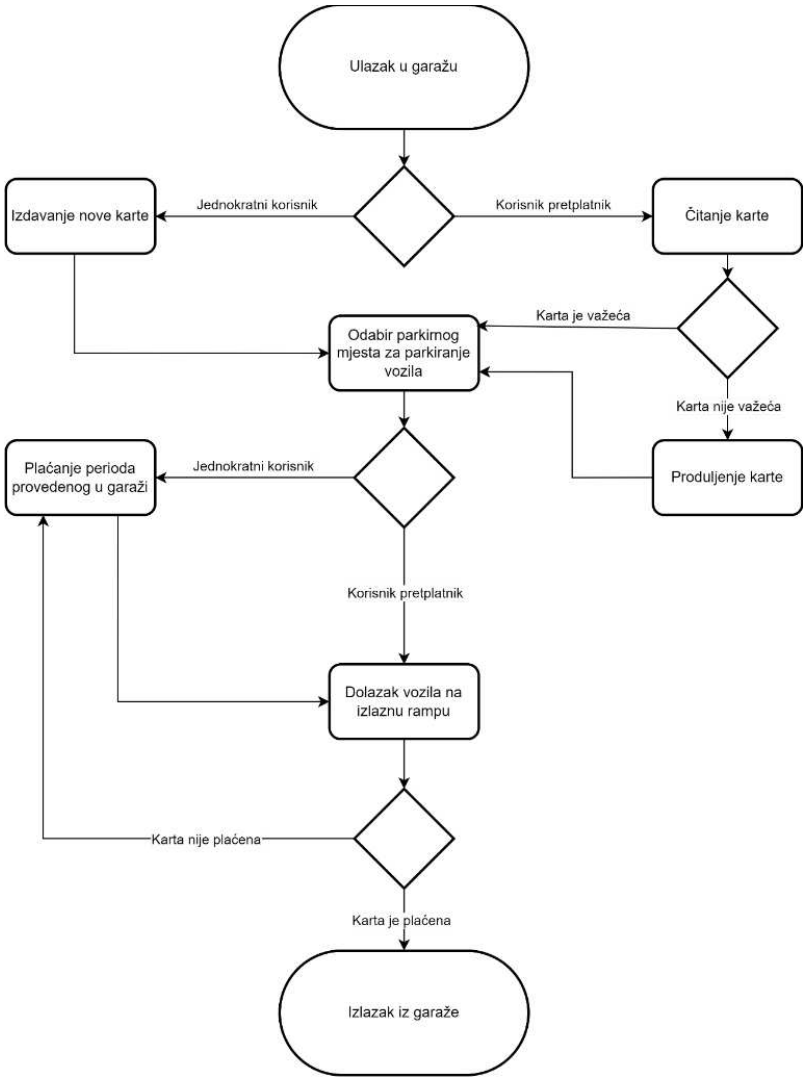
Entity Framework također pruža napredne značajke, poput migracija podataka, kojima se omogućuje automatsko ažuriranje baze podataka s novim verzijama modela podataka, bez potrebe za ručnim intervencijama. Osim toga, podržava razne načine upita i optimizaciju performansi, kao i mogućnost upotrebe spremljenih procedura (engl. *Stored Procedures*).

Korištenjem Entity Frameworka, znatno je pojednostavljena implementacija komunikacijskog sloja u aplikaciji za upravljanje javnom garažom, pružajući visoku razinu apstrakcije nad bazom podataka i optimizirajući produktivnost programera.

3. IZAZOVI SUSTAVA JAVNE GARAŽE I LOGIKA RJEŠENJA

Ovo poglavlje obrađuje problematiku sustava javne garaže, logiku rješenja koju aplikacija implementira, kao i tipične probleme na koje treba obratiti pažnju. Fokus će biti na izazovima kao što su izrada garaže, upravljanje zauzetošću parkirnih mjesta, plaćanje i validacija parkirnih karata te upravljanje pretplatnicima.

Dijagram toka (Slika 1) vizualno prikazuje kako se korisnici i podaci kreću kroz sustav javne garaže, od ulaska u garažu, preko parkiranja, plaćanja, pa sve do izlaska iz garaže. Automatizacija procesa smanjuje potrebu za ljudskom intervencijom, poboljšava učinkovitost i omogućuje korisnicima brži i jednostavniji način korištenja garaže.



Slika 1. Dijagram toka korištenja javne garaže

3.1. Izrada i odabir garaže

Pri otvaranju aplikacije, prvi element s kojim se korisnik aplikacije susreće je izrada nove garaže ili odabir jedne od postojećih garaža iz baze podataka. Svaka garaža ima sljedeća svojstva: naziv, broj etaža i broj parkirnih mjesta na svakoj od etaža.

Na primjeru korištenja (engl. *use case*) (Slika 2) može se vidjeti kako izgleda postupak izrade garaže.

Izrada garaže

Glavni sudionici: korisnik aplikacije, sustav za upravljanje garažom

1. Korisnik odabire izradu nove garaže.
2. Korisnik upisuje naziv i odabire broj etaža garaže.
3. Korisnik upisuje broj parkirnih mjesta za svaku od etaža.
4. Korisnik izrađuje garažu.
5. Sustav otvara aplikaciju za izrađenu garažu.

Odstupanje

1a. Korisnik se odlučuje na otvaranje postojeće garaže.
    1a1. Korisnik odabire jednu od postojećih garaža s popisa.
    1a2. Sustav otvara aplikaciju za odabranu garažu.

Slika 2. Izrada garaže

Bilo da se radi o izradi nove ili odabiru postojeće garaže, svi događaji i promjene u garaži ostvareni tijekom rada aplikacije bit će spremljeni u bazu podataka. Tako podaci ostaju sačuvani i nakon prestanka rada aplikacije kako bi se osigurala konzistentnost podataka i neometano ponovno korištenje.

3.2. Upravljanje zauzetošću parkirnih mjesta

Jedan od glavnih izazova u sustavu javne garaže je učinkovito upravljanje zauzetošću parkirnih mjesta (Slika 3). U velikim garažama s više razina, brzo i precizno praćenje slobodnih i zauzetih mjesta ključno je za optimizaciju protoka vozila i minimiziranje nezadovoljstva korisnika.

Praćenje zauzetosti parkirnih mjesta

Glavni sudionici: korisnik garaže, sustav za upravljanje garažom

1. Korisnik ulazi u garažu prolazeći pored senzora na ulaznoj rampi.
2. Senzor javlja sustavu da je vozilo ušlo u garažu i sustav umanjuje ukupan broj slobodnih mjesta za 1.
3. Korisnik parkira vozilo na slobodno parkirno mjesto.
4. Senzor na parkirnom mjestu detektira prisutnost vozila i sustav ažurira status mjesta kao zauzeto.
5. Korisnik napušta parkirno mjesto.
6. Senzor na parkirnom mjestu detektira odlazak vozila i sustav ažurira status mjesta kao slobodno.
7. Korisnik izlazi iz garaže prolazeći pored senzora na izlaznoj rampi.
8. Senzor javlja sustavu da je vozilo izašlo iz garaže i sustav uvećava ukupan broj slobodnih mjesta za 1.

Slika 3. Praćenje zauzetosti parkirnih mjesta

Kako bi se cijeli proces snalaženja po garaži korisnicima dodatno olakšao i ubrzao, poželjno je da se u garaži postave vizualni pokazatelji slobodnih mjesta. Uređaji poput ekrana s prikazom broja slobodnih mjesta po etažama i svjetlosnih signala iznad svakog parkirnog mjesta osiguravaju da, u slučaju velikog broja parkiranih automobila, sustav korisnika navodi prema slobodnom mjestu, što je mnogo učinkovitije od nasumične potrage.

Za potrebe demonstracije funkcionalnosti aplikacije nakon što vozilo uđe u garažu, nasumično se odabire slobodno parkirno mjesto koje se označava kao zauzeto te se umanjuje broj slobodnih mjesta na odgovarajućoj etaži, kao i ukupan broj slobodnih mjesta u garaži.

3.3. Naplata parkirnih karata

Drugi značajan izazov je sustav naplate parkiranja. Neophodno je osigurati točne i pravovremene naplate te učiniti proces plaćanja jednostavnim za korisnike, što može biti kompleksan zadatak. Provjera valjanosti karata je ključna kako bi se spriječile prevare i osigurala ispravna naplata.

Razlikuju se dva tipa korisnika garaže. Jednokratni korisnici su oni koji javnu garažu namjeravaju koristiti kraći vremenski period (najčešće manje od jednoga dana) te njima, prilikom ulaska u garažu, jedinica ulazne rampe izdaje jednokratnu parkirnu kartu čija cijena se utvrđuje umnoškom vremena provedenog u garaži i jedinične cijene jednog sata korištenja usluge garaže. Drugi tip korisnika su pretplatnici koji svoju kartu kupuju unaprijed kod djelatnika javne garaže, po utvrđenim cijenama. Za razliku od jednokratnih korisnika, pretplatnici prilikom ulasku u garažu svoju pretplatničku kartu umeću u ulaznu rampu.

Da bi se izlazna rampa mogla podići i napustiti garaža, neovisno o tipu korisnika, potrebno je u jedinicu izlazne rampe ubaciti kartu kako bi se moglo provjeriti stanje plaćanja. Zato je neposredno prije napuštanja garaže potrebno izvršiti plaćanje na jedinici za automatsku naplatu (Slika 4), koja se može usporediti s bilo kojom vrstom samoposlužnog automatskog uređaja (npr. aparat s hranom i pićem).

Naplata parkirne karte

Glavni sudionici: korisnik garaže, jedinica za automatsku naplatu
Preduvjet uspjeha: vozilo korisnika se nalazi u garaži

1. Korisnik umeće parkirnu kartu u jedinicu za automatsku naplatu.
2. Jedinica izračunava i na ekranu prikazuje iznos kojeg je potrebno platiti.
3. Korisnik plaća traženi iznos.
4. Jedinica označava parkirnu kartu kao plaćenu i vraća je korisniku.

Slika 4. Naplata parkirne karte

Radi naplate jednokratnih parkirnih karata, aplikacija svakoj karti prilikom izdavanja pridružuje podatak o vremenu ulaska u garažu. Kada korisnik umetne kartu u automatsku jedinicu za

naplatu, sustav izračunava razliku između vremena ulaska i trenutnog vremena. Ta razlika se zatim množi s cijenom parkiranja po satu, čime se dobiva ukupna cijena parkirne karte. Nakon uspješne naplate, sustav ažurira kartu u plaćeno stanje i vozilo je slobodno napustiti garažu.

Pretplatničke karte su podijeljene u različite kategorije (tarife) koje pokrivaju određene periode tijekom dana. Kod ovog tipa karata, s cijenom sata parkiranja se množe samo sati provedeni u garaži koji nisu pokriveni pripadajućom tarifom. Time se osigurava pravedna naplata za sve korisnike, ovisno o vremenu parkiranja koje njihova pretplatnička karta ne pokriva.

3.4. Upravljanje pretplatnicima

Jedan od važnih aspekata sustava javne garaže je upravljanje pretplatnicima i njihovim kartama. Pretplatnici su korisnici koji unaprijed plaćaju za određene periode parkiranja, čime im se osigurava korištenje garaže po povlaštenim cijenama. Sustav mora biti prilagodljiv kako bi zadovoljio potrebe različitih korisnika i omogućio jednostavno upravljanje pretplatama. Učinkovito upravljanje pretplatnicima ključno je za održavanje zadovoljstva korisnika i osiguranje stabilnog prihoda za garažu.

Korisnik garaže koji želi kupiti pretplatničku kartu to može ostvariti tako da se obrati djelatniku zaduženom za upravljanje sustavom garaže. Djelatnik od budućeg pretplatnika uzima osobne podatke koji osiguravaju jedinstvenost i omogućuju potencijalni kontakt s pretplatnikom u slučaju potrebe te ih unosi u zasebnu formu aplikacije zaduženu za upravljanje pretplatnicima. Nakon odabira tarife i vremenskog perioda trajanja pretplate, korisnik plaća traženu kartu i djelatnik mu je izdaje.

Već postojeći pretplatnici koji žele produžiti svoju kartu i eventualno promijeniti tarifu, također moraju pristupiti djelatniku garaže kako bi se produženje karte moglo naplatiti. Nakon uspješnog plaćanja, djelatnik produžuje pretplatničku kartu promjenom njenog datuma isteka.

Ako se ukaže potreba da se pretplatniku onemogući korištenje garaže, pretplatnička karta se može postaviti u blokirano stanje. Blokirana karta će uzrokovati da se prilikom dolaska vozila pretplatnika na ulaznu ili izlaznu rampu, bez obzira na rok valjanosti karte, rampa ne podiže uz odgovarajuću poruku na ekranu uređaja rampe. Pretplatnik u tom slučaju problem mora razriješiti izravnom komunikacijom s djelatnikom garaže.

3.5. Potencijalni problemi i kako ih izbjeći

Korištenje javne parkirne garaže kao i upravljanje pripadajućim sustavom podložno je mnogim rizicima. Međutim, pravilno planiranje i implementacija ključnih komponenti može značajno smanjiti te rizike. Neki od uobičajenih i očekivanih problema su:

- *Garaža je popunjena*: Kada se popune sva parkirna mjesta na svim etažama garaže, sustav će onemogućiti ulazak sljedećem vozilu tako dugo dok se ne oslobodi najmanje jedno mjesto, odnosno dok barem jedno vozilo ne napusti garažu. Time se sprječava ulazak vozila u popunjenu garažu, što može izazvati nezadovoljstvo korisnika zbog nemogućnosti pronalaska parkirnog mjesta. Kako bi korisnici bili obaviješteni zašto im sustav ne izdaje parkirnu kartu i ne pušta ih u garažu, prilikom pritiska tipke za izdavanje karte na ekranu uređaja ulazne rampe prikazat će se odgovarajuća poruka.

- *Korištenje parkirne karte za drugo vozilo*: Kako bi se osiguralo da svaki kratkotrajni korisnik može koristi jedinstvenu parkirnu kartu isključivo za vlastito vozilo, sustav prilikom izrade svakoj karti pridodaje registarske oznake vozila za koje se izdaje karta. Preduvjet ove funkcionalnosti je kamera u sklopu ulazne rampe uz odgovarajuću OCR (engl. *Optical Character Recognition*) programsku podršku za čitanje znakova. Budući da se nakon izlaska vozila iz garaže s plaćenom parkirnom kartom ona označava kao iskorištena i briše iz popisa prisutnih karata, nemoguće ju je koristiti za izlazak više od jednog vozila.

- *Gubitak parkirne karte*: Problemi izazvani gubitkom jednokratne parkirne karte izbjegnuti su zahvaljujući gore opisanoj funkcionalnosti čitanja registracijskih oznaka i njihova pridruživanja jednokratnim parkirnim kartama. Korisnik koji izgubi svoju kartu dužan je obratiti se djelatniku garaže i dati na uvid podatak o registarskim oznakama svog vozila. Djelatnik pretražuje popis prisutnih jednokratnih karata u sustavu kako bi pronašao kartu koja pripada vozilu tih registarskih oznaka i uz pomoć jedinstvenog identifikacijskog broja karte izradio duplikat kako bi ju stranka mogla platiti na automatskoj naplatnoj jedinici i napustiti garažu.

- *Besplatan period korištenja*: Nakon ulaska u garažu, korisnici imaju određeni vremenski period (u ovoj aplikaciji on iznosi 10 minuta) unutar kojeg mogu napustiti

garažu bez plaćanja parkirne karte. Cilj ovog pristupa je smanjiti nezadovoljstvo korisnika koji garažu namjeravaju koristiti bez duljeg zadržavanja. Na primjer: korisnik koji na rampi čeka red za ulazak u garažu se predomislio i ne želi parkirati vozilo, ali se iza njega već oformila kolona drugih vozila i ne može pomaknuti svoje vozilo na drugi način nego da uđe u garažu, pri čemu mora uzeti kartu kao i svi ostali korisnici. Budući da će garažu koristiti samo „u prolazu“, slobodan ju je napustiti bez naknade. Za javne garaže u sklopu objekata posebnih namjena, poput bolnica ili trgovačkih centara, prolazni korisnici mogu biti korisnici koji dolaze osigurati prijevoz osobama bez vozila i stoga nemaju namjeru zadržavati se u garaži.

- *Pretplatnik ne može ući ili napustiti garažu*: Pretplatničke karte imaju tri moguća stanja ovisna o tome gdje se nalaze. S kartom koja je u stanju *unutar garaže* moguće je napustiti garažu, s kartom u stanju *izvan garaže* moguće je ući u garažu, a s kartom u *neutralnom* stanju moguće je i ući i napustiti garažu. Prolaskom kroz ulaznu rampu stanje im se mijenja u *unutar garaže* dok se prolaskom kroz izlaznu rampu ono mijenja u stanje *izvan garaže*. Ova stanja karata osiguravaju da se s jednom pretplatničkom kartom u bilo kojem trenutku samo jedno vozilo može nalaziti u garaži. Kada se pretplatnička karta izrađuje, ona je u *neutralnom* stanju, budući da pretplatnik može kupiti kartu kada se njegovo vozilo nalazi unutar ili izvan garaže. Prvim prolaskom kroz jednu od rampi, pretplatnička karta će se postaviti u odgovarajuće stanje. Kada u sustavu dođe do kvara, npr. zbog nestanka struje, pada baze podataka, prolaska dva ili više vozila tokom jednog podizanja rampe, moguće je da rampe neće registrirati ulazak odnosno izlazak vozila. Uslijed toga će prilikom slijedećeg dolaska do rampe pretplatniku biti onemogućen prolaz zbog neodgovarajućeg stanja karte. Kako bi se pretplatniku osiguralo daljnje korištenje usluge, djelatnik garaže ima mogućnost postavljanja pretplatničke karte u *neutralno* stanje.

4. PRAKTIČNI RAD – DIZAJN I IMPLEMENTACIJA

Arhitektura aplikacije izvedena je kao slojevita, gdje je svaki sloj odgovoran za različite zadatke unutar aplikacije, dok je dizajn korisničkog sučelja uređen da bi bio intuitivan, konzistentan i jednostavan za korištenje.

4.1. Arhitektura

Glavni cilj slojevite arhitekture je podijeliti strukturu aplikacije na slojeve, od kojih svaki ima svoju odgovornost. Time se postiže bolje upravljanje složenošću sustava, lakše održavanje i veća skalabilnost. Svaki sloj ima jasno definiranu funkciju i komunikacija između slojeva odvija se kroz dobro definirana sučelja.

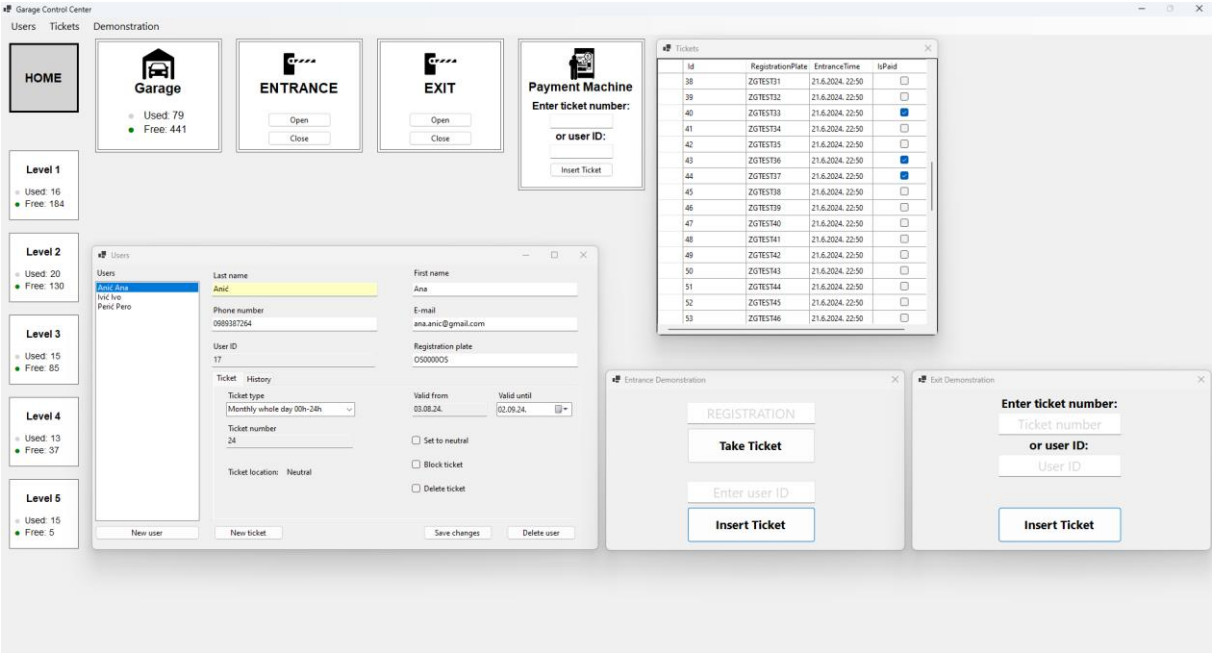
Arhitektura ove aplikacije se sastoji od sljedećih slojeva:

1. Prezentacijski sloj:
    - *Odgovornost*: Prikaz korisničkog sučelja i interakcija s korisnikom.
    - *Primjeri*: Početni zaslon, forma za upravljanje korisnicima, forma za pregled karata sustava, prikaz zauzeća parkirnih mjesta…
    - *Tehnologija*: Windows Forms.
2. Poslovni sloj:
    - *Odgovornost*: Poslovna logika aplikacije koja obavlja sve potrebne proračune, validacije i obradu podataka.
    - *Primjeri*: Validacija korisničkih unosa, izračunavanje popunjenosti garaže, izračunavanje cijene parkiranja…
    - *Tehnologija*: C# klase i metode.
3. Komunikacijski sloj:
    - *Odgovornost*: Upravljanje pristupom i komunikacijom s bazom podataka.
    - *Primjer*: CRUD operacije nad tablicama baze podataka.
    - *Tehnologija*: Entity Framework.
4. Baza podataka:
    - *Odgovornost*: Fizičko spremište podataka.
    - *Primjeri*: Tablica garaža, tablica parkirnih karata, tablica parkirnih mjesta, tablica korisnika…

- *Tehnologija*: Microsoft SQL Server.

4.2. Dizajn korisničkog sučelja

Korisničko sučelje aplikacije (Slika 5) izrađeno je tako da bude jednostavno, intuitivno, konzistentno, pristupačno i responzivno, što čini iskustvo korištenja ugodnim i učinkovitim.



Slika 5. Korisničko sučelje aplikacije

Prilikom dizajna, vodilo se računa o sljedećim načelima:

- *Jednostavnost*: Sve forme aplikacije, kao i glavni izbornik, sadrže samo najvažnije opcije, bez suvišnih elemenata koji bi mogli zbuniti korisnike. Ovakav pristup osigurava korisnicima da brzo pronađu i pristupe ključnim funkcionalnostima, što skraćuje vrijeme potrebno za obavljanje zadataka i smanjuje rizik od pogrešaka.
- *Konzistentnost*: Stilovi teksta, boje, fontovi i tipke su konzistentni u svim elementima aplikacije. Ova uniformiranost omogućuje korisnicima da se lakše snađu, jer znaju što mogu očekivati na svakom dijelu sučelja. Konzistentnost u dizajnu stvara osjećaj povjerenja i pouzdanosti, što je ključno za pozitivno korisničko iskustvo.
- *Povratne informacije*: Nakon što korisnik aplikacije pokuša spremiti neku promjenu, prikazuje se potvrda o uspjehu ili poruka o grešci. Ovakve povratne informacije

omogućuju korisnicima aplikacije da odmah znaju jesu li njihove radnje završile uspješno ili trebaju poduzeti dodatne korake. Brza i jasna povratna informacija smanjuje nesigurnost i frustraciju korisnika aplikacije.

- *Pristupačnost*: Svi tekstovi u aplikaciji su dovoljno veliki za čitanje, poruke jasne, a odabrane boje takve da kontrasti između pozadine i teksta budu dovoljno veliki za korisnike slabijeg vida. To osigurava da svi korisnici, bez obzira na njihove fizičke sposobnosti, mogu koristiti aplikaciju s lakoćom. Također, aplikacija je dizajnirana tako da se u potpunosti može koristiti uz pomoć tipkovnice, što dodatno povećava njezinu pristupačnost bilo kojoj vrsti korisnika.
- *Efikasnost*: Korisnici mogu na jednostavan način, u samo nekoliko koraka, pristupiti svakoj funkcionalnosti aplikacije. Ova efikasnost u dizajnu smanjuje vrijeme potrebno za navigaciju kroz aplikaciju, a time i obavljanje zadatka, te poboljšava ukupno korisničko iskustvo. Jednostavni i brzi procesi omogućuju korisnicima da se fokusiraju na svoje glavne ciljeve, umjesto na navigaciju kroz aplikaciju.

4.3. Simulacija ulaska u garažu

Kako bi se aplikacija mogla testirati i demonstrirati bez uređaja koji se nalaze u javnoj garaži (ulazne/izlazne rampe s čitačem i koderom karata), napravljeni su moduli kojima se simulira proces ulaska vozila u garažu. Korisnik aplikacije ima mogućnost unosa registracijske oznake za jednokratne korisnike i izrade jednokratne parkirne karte ili unosa identifikacijskog broja korisnika pretplatnika pomoću kojeg će se pročitati podaci s pretplatničke karte unesenog pretplatnika.

Nakon unosa u jedno od dva polja za unos teksta, provjerava se postoji li barem jedno slobodno parkirno mjesto na nekoj od etaža u garaži. U slučaju da postoji, izrađuje se nova parkirna karta s ulaznim podacima, podiže se rampa i popunjava se parkirno mjesto (Kôd 4).

```
var availableLevels = myGarage.Levels.Where(level => level.FreeSpots()
> 0).ToList();

if (availableLevels.Count > 0)
{
    string registration = RegistrationTextBox.Text;
    if (registration != "REGISTRATION" && registration != "")
    {
        var ticket = Entrance.IssueTicket(RegistrationTextBox.Text);
        myGarage.Tickets.Add(ticket);
        await garageService.AddTicketAsync(ticket);
        await UpdateParkingSpot(availableLevels);
        Entrance.OpenBarrier();
    }
    else
    {
        MessageBox.Show("Please enter your vehicle's registration.");
    }
}
else
{
    MessageBox.Show("No available spots in the garage.");
}
```

Kôd 4. Ulazak jednokratnog korisnika

Izrada karte je funkcija koju izvršava ulazna rampa, a konstruktor klase `Ticket` stvara novu kartu s podacima o vremenu ulaska i registraciji vozila, te postavlja kartu u stanje „Nije plaćeno“.

Parkirno mjesto koje će se popuniti pri ulasku u garažu se bira funkcijom `UpdateParkingSpot`, kojoj se prosljeđuje popis etaža garaže koje sadrže najmanje jedno slobodno parkirno mjesto. Nasumično se odabire jedna od etaža, unutar koje se nasumično odabire i popunjava jedno slobodno parkirno mjesto, te se podiže događaj `CustomerEntry` kako bi glavna forma aplikacije mogla ažurirati stanje parkirnih mjesta u garaži (Kôd 5).

```
private async Task UpdateParkingSpot(List<Level> availableLevels)
{
    // Randomly select a level from the available levels
    int randomLevelIndex = random.Next(availableLevels.Count);
    var chosenLevel = availableLevels[randomLevelIndex];

    // Filter spots that are not occupied in the chosen level
    var availableSpots = chosenLevel.Spots.Where(spot =>
!spot.IsOccupied).ToList();

    // Select a random spot from the available spots
    int randomSpotIndex = random.Next(availableSpots.Count);
    var chosenSpot = availableSpots[randomSpotIndex];
    chosenSpot.ReserveSpot();

    await CustomerEntry?.Invoke(this, new
CustomerEntryEventArgs(chosenSpot));
}
```

Kôd 5. Algoritam odabira parkirnog mjesta

4.4. Simulacija izlaska iz garaže

Izlazak vozila iz garaže odvija se vrlo slično ulasku vozila. Vozilo dolazi pred izlaznu rampu te senzor prepoznaje prisustvo vozila i aktivira se čitač karata unutar rampe. Za razliku od ulazne rampe, izlazna rampa nema mogućnost izdavanja karata, već samo čitanja. U rampu se ubacuje karta s koje se čitaju podaci o naplati. Ako su svi troškovi plaćeni, rampa se podiže i vozilo izlazi iz garaže.

Funkcionalnost demonstracije izlaska vozila iz garaže implementirana je tako da simulira gore navedeni primjer. Korisnik aplikacije ima na raspolaganju dva polja za unos teksta i jednu tipku kojom se simulira umetanje karte u izlaznu rampu.

Jedno polje za unos teksta očekuje unos broja jednokratne parkirne karte, a drugo unos identifikacijskog broja korisnika pretplatnika. Pritiskom na tipku „Insert Ticket“ uneseni podaci se provjeravaju i ako su podaci ispravni i karta je plaćena, izlazna rampa se podiže.

Budući da jednokratna karta i karta korisnika pretplatnika nisu ista vrsta objekta, demonstracija izlaska iz garaže sadrži dvije funkcije za obradu svake od karata:

```
1. public async Task HandleTicketInserted(int ticketNumber)
2. public async Task HandleUserIdEntered(int userId)
```

Funkcija `HandleTicketInserted` (Kôd 6) prima broj jednokratne parkirne karte kao argument te poziva funkciju izlazne rampe `ReadTicket` koja provjerava je li karta plaćena. U slučaju da je ona plaćena, karta se briše iz baze podataka, poziva se funkcija `UpdateParkingSpot` koja nasumice oslobađa jedno parkirno mjesto (algoritam odabira mjesta je jednak kao i kod demonstracije ulaska u garažu) i rampa se podiže u sklopu funkcije `ReadTicket`.

```
var ticket = myGarage.GetTicket(ticketNumber);

if (ticket != null)
{
    Exit.ReadTicket(ticket);

    if (Exit.IsOpen)
    {
        myGarage.Tickets.Remove(ticket);
        await service.RemoveTicketAsync(ticket);
        await UpdateParkingSpot();
        MessageBox.Show("Thank you for using our garage!", "Barrier
Open", MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
    else
    {
        MessageBox.Show("Ticket not paid. Please pay the ticket on the
automatic payment machine!", "Barrier Closed", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
    }
}
```

Kôd 6. Funkcija izlaska vozila s jednokratnom parkirnom kartom

Kada se radi o korisniku pretplatniku, poziva se funkcija `HandleUserIdEntered` (Kôd 7) s identifikacijskim brojem korisnika kao argumentom. Nakon provjere unosa, poziva se funkcija `ReadUserTicket` kojoj se šalje objekt (parkirna karta korisnika pretplatnika) na provjeru plaćanja.

```
Exit.ReadUserTicket(user.UserTicket);

if (Exit.IsOpen)
{
    user.UserTicket.SetOutside();
    await userService.UpdateUserAsync(user);
    await UpdateParkingSpot();
    MessageBox.Show("Thank you for using our garage!", "Barrier Open",
MessageBoxButtons.OK, MessageBoxIcon.Information);
}
else
{
    MessageBox.Show("Ticket not paid. Please pay the ticket on the
automatic payment machine!", "Barrier Closed", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
}
```

Kôd 7. Funkcija izlaska vozila s pretplatnom kartom

Kako pretplatnici imaju mogućnost kupnje različitih tarifa pretplatnih parkirnih karata koje pokrivaju različite periode u danu, potrebno je provjeriti je li pretplatnik proveo više vremena u garaži nego li što to dozvoljava njegova pretplatna karta (Kôd 8). Ako je svo vrijeme provedeno u garaži pokriveno parkirnom kartom ili je korisnik platio razliku provedenog vremena, izlazna rampa se podiže i vozilo napušta garažu.

```
public void ReadUserTicket(UserTicket ticket)
{
    int uncoveredHours = ticket.GetUncoveredHours();
    if (uncoveredHours == -1)
    {
        ticket.CalculateUncoveredHours();
        uncoveredHours = ticket.GetUncoveredHours();
    }
    if (uncoveredHours == 0)
    {
        OpenBarrier();
    }
}
```

Kôd 8. Provjera provedenog vremena za vozila s pretplatnom kartom

4.5. Naplata parkirne karte

Naplata vremena provedenog u garaži je neophodna da bi vozila mogla napustiti garažu. Funkcije implementirane u klasi `PaymentMachine` koja simulira automatsku naplatnu

jedinicu garaže, izračunavaju i naplaćuju vrijeme korištenja garaže. Budući da kontrola korisničkog sučelja koja predstavlja naplatnu jedinicu ima dva polja za unos, unosi se ili broj jednokratne parkirne karte ili identifikacijski broj korisnika pretplatnika te se prikazuje poruka s iznosom kojeg je potrebno platiti.

U slučajevima jednokratnih parkirnih karata, izračun cijene koju je potrebno platiti vrši funkcija `CalculateTotalPrice` (Kôd 9). Vremenski period proveden u garaži izračunava se usporedbom trenutnog vremena (tj. vremena naplate) s vremenom ulaska u garažu. Razlikom tih vremena dobiva se broj sati za naplatu koji se množi s cijenom jednog sata parkiranja zapisanog kao podatkovni član `Rate` klase `PaymentMachine`.

```
public decimal CalculateTotalPrice(Ticket ticket)
{
    TimeSpan elapsedTime = DateTime.Now - ticket.EntranceTime;
    int totalElapsedMinutes =
(int)Math.Ceiling(elapsedTime.TotalMinutes);

    int totalHours = totalElapsedMinutes / 60;
    int minutesAfterHour = totalElapsedMinutes % 60;

    decimal totalPrice = (minutesAfterHour < GracePeriod) ? totalHours
* Rate : (totalHours + 1) * Rate;
    return totalPrice;
}
```

Kôd 9. Izračun cijene jednokratne parkirne karte

Prilikom izračunavanja vremenskog perioda, izuzima se vrijeme potrebno za pronalazak parkirnog mjesta, napuštanje vozila, plaćanje karte i izlazak iz garaže. Taj period (`GracePeriod`) također je zapisan u klasi kao podatkovni član i posljednji sat parkiranja se naplaćuje kao puni sat jedino ako je korisnik prekoračio `GracePeriod`. U slučaju da je vozilo boravilo u garaži kraće od tog perioda, neće biti iznosa za naplatu i vozilo može besplatno napustiti garažu.

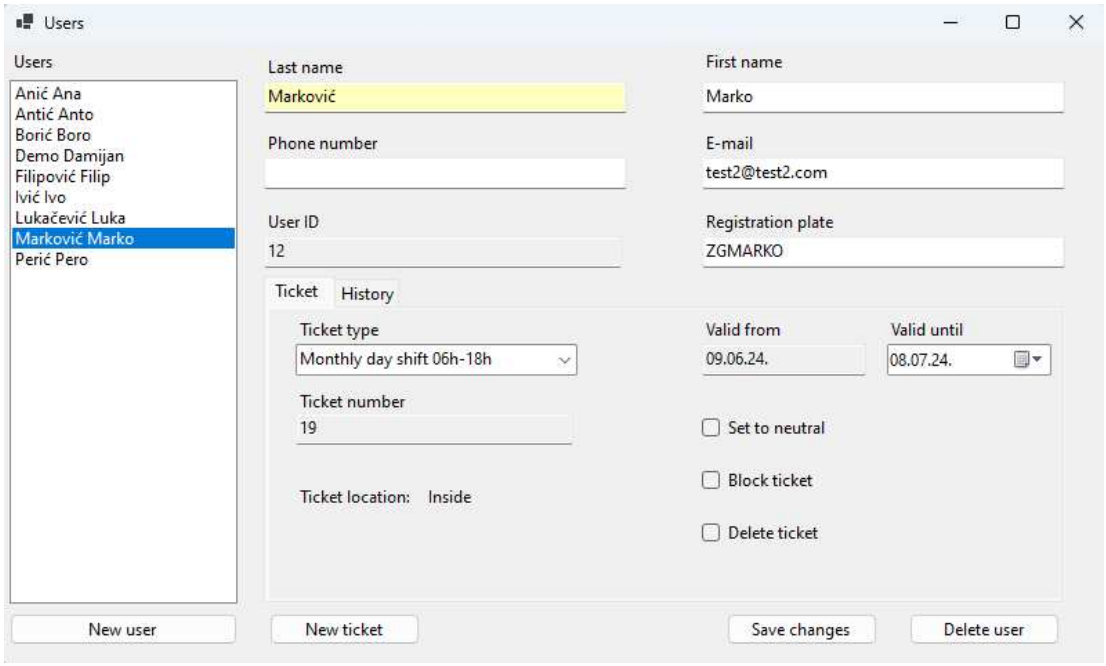
Budući da korisnici pretplatnici imaju mogućnost kupnje različitih tarifa pretplatnih karata, izračun njihove cijene parkiranja je drugačiji. Pretplatnu kartu korisnika predstavlja klasa `UserTicket`, koja sadrži funkciju za izračun sati koji nisu pokriveni tarifom karte (Kôd 10). Od vremena ulaska sve do vremena plaćanja, naplaćuju se samo sati koji su izvan perioda kojeg pokriva tarifa pretplatničke karte.

```
var startTime = lastEntranceEvent.TimeStamp;
var endTime = DateTime.Now;

for (var time = startTime; time < endTime; time = time.AddHours(1))
{
    switch (Type)
    {
        case TicketType.DayShift:
            if (time.Hour < 6 || time.Hour >= 18)
            {
                UncoveredHours++;
            }
            break;

        case TicketType.NightShift:
            if (time.Hour >= 6 && time.Hour < 18)
            {
                UncoveredHours++;
            }
            break;
        default:
            UncoveredHours = 0;
            break;
    }
}
```

Kôd 10. Izračun sati koje ne pokriva pretplatnička karta

4.6. Upravljanje korisnicima

Zasebna forma aplikacije (Slika 6), izrađena isključivo za administraciju pretplatnika javne garaže, omogućuje djelatnicima garaže da kroz korisničko sučelje upravljaju korisnicima, uključujući dodavanje novih, ažuriranje postojećih, te brisanje korisnika i njihovih pretplatnih karata.



Slika 6. Forma za upravljanje pretplatnicima

Uz lijevi rub forme nalazi se `ListBox` kontrola koja sadrži popis imena ili naziva svih već postojećih pretplatnika. Kako bi se osiguralo da kontrola uvijek bude popunjena aktualnim podacima, koriste se klase `BindingList` i `BindingSource` koje povezuju listu `Users` klase `Garage` s kontrolom tako da svaka promjena u listi `Users` bude preslikana u kontrolu.

Odabir nekog od pretplatnika s `ListBox` kontrole podiže događaj `SelectedIndexChanged`, na koji se poziva funkcija `PopulateUserControls` koja popunjava polja forme podacima odabranog pretplatnika.

Prilikom dodavanja novog pretplatnika, sve kontrole forme koje sadrže podatke o korisniku i njegovoj karti se prazne kako bi bile spremne za unos novih podataka. Nakon spremanja, u listi `Users` i u bazi podataka stvara se novi korisnik s unesenim podacima (Kôd 11).

```
else
{
    var newUser = CreateUser();
    UserService.ValidateUser(newUser);
    myGarage.Users.Add(newUser);
    await userService.AddUserAsync(newUser);
    userIdTextBox.Text = newUser.Id.ToString();

    if (newUser.UserTicket != null)
    {
        ticketNumberTextBox.Text = newUser.UserTicket.Id.ToString();
        TicketStateLabel.Text =
existingUser.UserTicket.State.ToString();
    }
    bindingUserList.ResetBindings();
}
```

Kôd 11. Dodavanje novog pretplatnika

U slučaju promjena podataka postojećih korisnika, nakon odabira korisnika s popisa na lijevoj strani forme, u kontrole se unose novi podaci o korisniku ili o njemu pridruženoj parkirnoj karti, te se spremaju u listu i bazu podataka (Kôd 12).

```
if (existingUser != null)
{
    UserService.ValidateUser(new GarageUser(
        lastNameTextBox.Text,
        firstNameTextBox.Text,
        phoneNumberTextBox.Text,
        emailTextBox.Text,
        registrationPlateTextBox.Text,
        myGarage.Id));

    UpdateExistingUser(existingUser);
    await userService.UpdateUserAsync(existingUser);
    if (existingUser.UserTicket != null)
    {
        ticketNumberTextBox.Text =
existingUser.UserTicket.Id.ToString();
        TicketStateLabel.Text =
existingUser.UserTicket.State.ToString();
    }
}
```

Kôd 12. Ažuriranje podataka pretplatnika

Budući da su kontrole s podacima o korisniku polja za unos teksta, potrebno je, prije spremanja promjena podataka, napraviti provjeru unesenog teksta. Servis zadužen za komunikaciju s bazom podataka sadrži funkciju `ValidateUser` (Kôd 13) koja izbacuje iznimku sve dok svi

uneseni podaci ne zadovoljavaju uvjete funkcije. Korisniku aplikacije se pritom prikazuje poruka i upućuje ga na polje za unos čiji sadržaj ne zadovoljava uvjete provjere.

```
public static void ValidateUser(GarageUser user)
{
    if (string.IsNullOrWhiteSpace(user.FirstName))
    {
        throw new ArgumentException("First name is required");
    }
    if (user.FirstName.Length > 50)
    {
        throw new ArgumentException("First name cannot exceed 50
characters");
    }
    if (string.IsNullOrWhiteSpace(user.LastName))
    {
        throw new ArgumentException("Last name is required");
    }
    if (user.LastName.Length > 50)
    {
        throw new ArgumentException("Last name cannot exceed 50
characters");
    }
    if (!string.IsNullOrEmpty(user.PhoneNumber) &&
!Regex.IsMatch(user.PhoneNumber, @"^\d{10}$"))
    {
        throw new ArgumentException("Invalid phone number");
    }
    if (!string.IsNullOrEmpty(user.Email) && !new
EmailAddressAttribute().IsValid(user.Email))
    {
        throw new ArgumentException("Invalid email address");
    }
    if (string.IsNullOrWhiteSpace(user.RegistrationPlate))
    {
        throw new ArgumentException("Registration plate is required");
    }
}
```

Kôd 13. Provjera podataka o pretplatniku

Nakon dodavanja novog pretplatnika, kontrole s podacima o njegovoj karti nisu vidljive jer novi pretplatnici nemaju pridruženu parkirnu kartu, već ju je potrebno izraditi (Kôd 14) i spremiti. U slučaju već postojećeg korisnika, spremaju se izmijenjeni podaci o parkirnoj karti. Ako treba obrisati kartu korisnika, potrebno je postaviti kvačicu u rubriku „Delete Ticket“ prije spremanja promjena i karta odabranog korisnika će biti obrisana.

```
private UserTicket CreateTicket()
{
    var from = DateTime.ParseExact(validFromTextBox.Text, "dd.MM.yy.",
System.Globalization.CultureInfo.InvariantCulture);
    var until = validUntilTextBox.Value;
    var type = ticketTypeComboBox.SelectedIndex switch
    {
        0 => TicketType.WholeDay,
        1 => TicketType.DayShift,
        2 => TicketType.NightShift,
        _ => TicketType.WholeDay,
    };

    return new UserTicket(from, until, type);
}
```

Kôd 14. Izrada pretplatničke karte

Dio forme administracije korisnika zadužen za prikaz podataka o pretplatničkoj parkirnoj karti sastoji se od dvije kartice. Prva kartica „Ticket“ prikazuje podatke o samoj karti, a druga kartica „History“ tablično prikazuje povijest evidentiranih ulazaka i izlazaka iz garaže za dotičnu kartu.

Za ovu funkcionalnost kreirana je klasa `TicketEvent` i enumeracija `TicketEventType`. Klasa `UserTicket`, koja predstavlja parkirnu kartu korisnika pretplatnika, sadrži listu `TicketEvents` kojoj se, prilikom svakog ulaska ili izlaska iz garaže, dodaje novi član tipa `TicketEvent` s podacima o vrsti aktivnosti (radi li se o ulasku ili izlasku) i vremenom događaja (Slika 7). Na ovaj način se unutar kartice „History“ forme za administraciju mogu tablično prikazati svi ulasci i izlasci iz garaže odabranog korisnika, odnosno njegove parkirne karte.

Slika 7. Klasa `UserTicket` i `TicketEvent` tip podataka

5. ZAKLJUČAK

Razvoj aplikacije za upravljanje sustavom javne garaže rezultirao je nizom značajnih spoznaja i postignuća koji su prikazani u ovom radu. Ključna postignuća uključuju učinkovito praćenje zauzetosti parkirnih mjesta, optimizaciju naplate parkiranja te jednostavnost procesa upravljanja pretplatnicima. Ove funkcionalnosti implementirane su kako bi značajno poboljšale operativnu učinkovitost garaže i iskustvo njenih korisnika.

Jedna od ključnih spoznaja proizašla iz ovog projekta je važnost preciznog praćenja zauzetosti parkirnih mjesta. Implementacijom senzora za detekciju vozila omogućeno je praćenje u stvarnom vremenu, što olakšava korisnicima pronalazak slobodnog mjesta i smanjuje vrijeme provedeno u traženju parkirnog mjesta. Vizualni pokazatelji, poput ekrana s prikazom broja slobodnih mjesta po etažama te svjetlosni signali iznad svakog parkirnog mjesta, dodatno unapređuju korisničko iskustvo. Ova rješenja ne samo da povećavaju zadovoljstvo korisnika, već i optimiziraju protočnost vozila kroz garažu, smanjujući zastoje i frustracije korisnika.

Sustav naplate parkiranja ključan je za financijsku održivost garaže. Razlikovanje jednokratnih korisnika od pretplatnika stvara fleksibilniji i pravedniji pristup naplati. Jednokratni korisnici dobivaju parkirne karte prilikom ulaska u garažu, a sustav automatski izračunava trošak na temelju vremena provedenog u garaži. S druge strane, pretplatnici koriste karte s unaprijed definiranim tarifama, što omogućuje precizniju kontrolu troškova i bolje upravljanje resursima garaže. Ovakav sustav naplate je učinkovit, pouzdan i jednostavan za korisnike i ključna je komponenta svake moderne javne garaže.

Upravljanje pretplatnicima također je područje koje je značajno pojednostavljeno. Budući da se pretplatnici ne mogu sami registrirati, već sve promjene podataka i produženja karata obavlja djelatnik garaže, sustav je osmišljen kako bi olakšao rad djelatnicima te osigurao točne i pravovremene informacije o korisnicima. Jednostavnost procesa ažuriranja podataka, izdavanja novih karti te upravljanja postojećim pretplatničkim kartama, smanjuje vjerojatnost pogrešaka i povećava učinkovitost rada što rezultira zadovoljstvom korisnika te boljom organizacijom i transparentnošću poslovanja.

Jedan od važnih aspekata koji su proizašli iz ovog rada je i fleksibilnost sustava za buduće nadogradnje i proširenja. Arhitektura aplikacije omogućuje jednostavnu integraciju novih

funkcionalnosti, što je ključno za dugoročnu održivost i prilagodljivost promjenjivim potrebama tržišta. Mogućnost dodavanja naprednih analitičkih alata za generiranje izvještaja, kupovine parkirnih karata putem web-a, optimizaciju cijena parkiranja te uvođenje novih pretplatničkih artikala poput dnevnih ili tjednih karata na temelju potražnje, omogućit će sustavu da ostane relevantan i učinkovit i u budućnosti.

Zaključno, rezultati ovog rada pokazuju kako se uz korištenje suvremenih tehnologija može značajno unaprijediti upravljanje javnim garažama. Sustav za upravljanje zauzetošću parkirnih mjesta, naplatu parkiranja i upravljanje pretplatnicima, kako je prikazan u ovom radu, predstavlja cjelovito rješenje koje odgovara na trenutne izazove i priprema se za buduće potrebe. Ovaj projekt ne samo da doprinosi boljem korisničkom iskustvu, već i optimizira operativne procese, čime se osigurava dugoročna održivost i uspjeh sustava za upravljanje javnim garažama.

SAŽETAK

Ovaj završni rad opisuje problematiku i postupak izrade aplikacije za upravljanje sustavom javne garaže i njenim korisnicima. U radu je, uz korištene tehnologije i implementaciju rješenja, obuhvaćen proces korištenja garaže od strane kratkotrajnih korisnika i pretplatnika što uključuje ulazak, naplatu i izlazak iz garaže kao i probleme koji se pritom pojavljuju. Kako se objekt javne garaže sastoji od fizički razdvojenih elemenata, aplikacija je osmišljena da ih objedini na informacijskoj razini kako bi se osigurala automatizacija i autonomija rada javne garaže. Ovime se smanjuje potreba za ljudskom intervencijom i pojednostavljuju zadaci djelatnika garaže koji su ciljana skupina korisnika aplikacije.

Ključne riječi: parking, garaža, automobil, korisnik, karta, sustav, naplata, automatizacija

SUMMARY

This final thesis describes the challenges and development process of an application for managing a public garage system and its users. The paper covers the technologies used and the implementation of the solution, as well as the process of garage usage by both short-term users and subscribers, including entry, payment, and exit, along with the issues that arise. Since the public garage consists of physically separated elements, the application is designed to integrate them on an informational level to ensure the automation and autonomy of the garage's operations. This reduces the need for human intervention and simplifies the tasks of the garage staff, who are the target users of the application.

Keywords: parking, garage, car, user, ticket, system, payment, automation