PROGRAMMA IL TUO FUTURO E-80 SWD

Statuti Alessandro

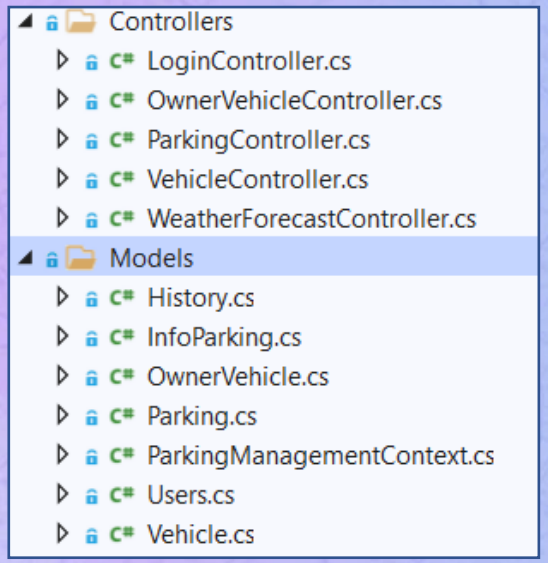
Parking management

GESTIONE DI UN PARCHEGGIO ONLINE TRAMITE AUTENTICAIZONE.

REALIZZATO DA STATUTI, OLEARI E SINGH.

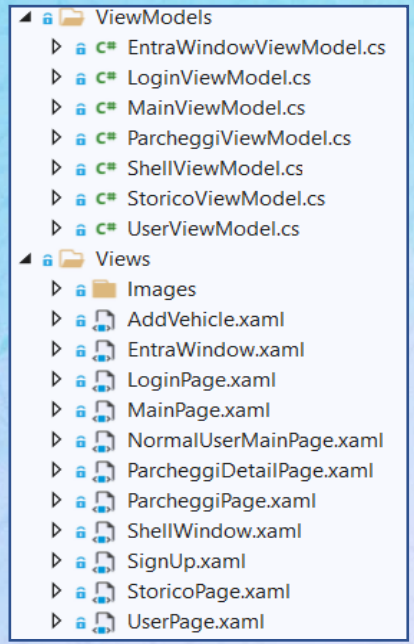
STRUTTURA

L'applicazione è stata realizzata in Dotnet Core 3.1 LTS tramite Windows Tamplate Studio suddividendola in più progetti: API-MVC (WebAPI-Definitivo) e MVVM (WPF-DEFINITIVO).



- Model
- View
- Controller

- Model
- View
- ViewModel



MODEL - DATABASE

Con il comando «dotnet ef» è possibile gestire tutti gli aspetti legati Entity Framework

Vehicle	
PK	VehicleId
FK	OwnerId
	Manufacturer
	Model
	LicencePlate

OwnerVehicle	
PK	OwnerId
FK	UserId
	Name
	Surname
	DateBirth

History	
PK	HistoryId
	PosizioneParcheggio
	ParkingId
	Revenue
	EntryTimeDate
	ExitTimeDate
	VehicleId
	InfoParkId

Parking	
PK	ParkingId
FK	VehicleId
FK	InfoParkId
	Stato
	EntryTimeDate

Users	
PK	UserId
	Username
	Password
	Grado
	LastLogin
	LastLogout

InfoParking	
PK	InfoParkId
	ParkingName
	Ncol
	Nrow

CONTROLLER

Responsabile della logica di controllo per determinare la risposta da restituire ad un utente che effettua una richiesta all'interno di un applicazione MVC ASP.NET.

↓

CRUD operations

Create

```
[Route("api/v1")]
[ApiController]
0 riferimenti
public class LoginController : ControllerBase
{
    [HttpPost("Login")]
    0 riferimenti
    public ActionResult Login([FromBody] Users credentials)
    {
        using (ParkingManagementContext model = new ParkingManagementContext())
        {
            Users candidate = model.Users.FirstOrDefault(q =>
            q.Username == credentials.Username && q.Password == credentials.Password);
            if (candidate == null) return NotFound("Username o password errati");
            var tokenHandler = new JwtSecurityTokenHandler();
            var tokenDescriptor = new SecurityTokenDescriptor
            {
                SigningCredentials = new SigningCredentials
                (
                    SecurityKeyGenerator.GetSecurityKey(candidate),
                    SecurityAlgorithms.HmacSha256Signature
                ),
                Expires = DateTime.Now.AddDays(1),
                Subject = new ClaimsIdentity(
                    new Claim[]
                    {
                        new Claim("Username", candidate.Username.ToString()) ,
                        new Claim("Grado", candidate.Grado.ToString()) ,     //privilegio
                        new Claim("Id", candidate.Id.ToString())
                    })
            };
            SecurityToken token = tokenHandler.CreateToken(tokenDescriptor);
            candidate.LastLogin = DateTime.Now;
            model.SaveChanges();
            return Ok(tokenHandler.WriteToken(token));
        }
    }

    [HttpPost("GetToken")]
    0 riferimenti
```

Read

```
[Authorize]
[HttpGet("GetUsers")]
0 riferimenti
public ActionResult GetUsers([FromBody] Users credentials)
{
    using (ParkingManagementContext model = new ParkingManagementContext())
    {
        List <Users> candidate = model.Users.ToList();
        return Ok(candidate);
    }
}
```

CONTROLLER

Responsabile della logica di controllo per determinare la risposta da restituire ad un utente che effettua una richiesta all'interno di un applicazione MVC ASP.NET.

↓

CRUD operations

Update

```
[Authorize]
[HttpPut("parcheggio/{nomeParcheggio}/{nomePosto}")]
//Reimposto i valori nella tabella parking
0 riferimenti
public ActionResult UpdateParking(string nomeParcheggio, string nomePosto)
{
    try
    {
        using (ParkingManagementContext model = new ParkingManagementContext())
        {
            //trovo park id
            var infoParkId = model.InfoParking.Where(w => w.NamePark == nomeParcheggio).FirstOrDefault();


            //Update Parking Record
            Parking parking = model.Parking.Where(w => w.ParkingId == nomePosto && w.InfoParkId == infoParkId.InfoParkId).FirstOrDefault();

            parking.Stato = false;
            parking.VehicleId = null;
            parking.EntryTimeDate = null;
            model.SaveChanges();

            return Ok("Update riuscito");
        }
    }
    catch (Exception)
    {
        return Problem();
    }
}
```

Delete

```
[Authorize]
[HttpDelete("DeletePark/{name}")]
0 riferimenti
public ActionResult GetIncassiByID(string name)
{
    try
    {
        using (ParkingManagementContext model = new ParkingManagementContext())
        {
            long id = model.InfoParking.Where(w => w.NamePark == name).Select(s => s.InfoParkId).FirstOrDefault();
            List<Parking> parks = model.Parking.Where(w => w.InfoParkId == id).ToList();

            foreach(var a in parks)
            {
                model.Parking.Remove(a);
            }
            model.SaveChanges();

            List<History> history = model.History.Where(w => w.InfoParkId == id).ToList();
            foreach(var a in history)
            {
                model.History.Remove(a);
            }
            model.SaveChanges();

            InfoParking info = model.InfoParking.Where(w => w.NamePark == name).FirstOrDefault();
            model.InfoParking.Remove(info);
            model.SaveChanges();
            return Ok("Parcheggio eliminato correttamente");
        }
    }
    catch (Exception)
    {

        return Problem();
    }
}
```

VIEW MODEL

Ha il compito di effettuare i binding e le chiamate HTTP al API-Controller.

Esempio:

```
public async Task GetParkings()
{
    parkingsTemp.Clear();
    if (NavigationLoginToLogout.isLoggedIn)
    {
        using (var client = new HttpClient())
        {
            // Richiesta
            client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", NavigationLoginToLogout.Token);
            var response = await client.GetAsync("http://localhost:13636/api/v1/ParkingList");
            var list = await response.Content.ReadAsStringAsync();

            if (response.IsSuccessStatusCode)
            {
                // Creo lista di oggetti
                ParkingObject = JsonConvert.DeserializeObject<ObservableCollection<InfoParking>>(list);


                InfoParking i = new InfoParking("Nuovo-Parcheggio", riga, colonna);
                parkingsTemp.Add(i);
                foreach (var a in ParkingObject)
                {
                    parkingsTemp.Add(a);
                }

                foreach (var a in parkingsTemp)
                {
                    bool trovato = false;
                    foreach (var b in parkings)
                    {
                        if (a.NamePark == b)
                            trovato = true;
                    }
                    // Aggiungo gli elementi trovati a parkings
                    if(!trovato)
                        parkings.Add(a.NamePark.ToString());
                }


            }
            else
            {
                System.Windows.MessageBox.Show("Nessuno parcheggio trovato", "Error", MessageBoxButton.OK, MessageBoxImage.Error);
            }
        }
    }
}
```

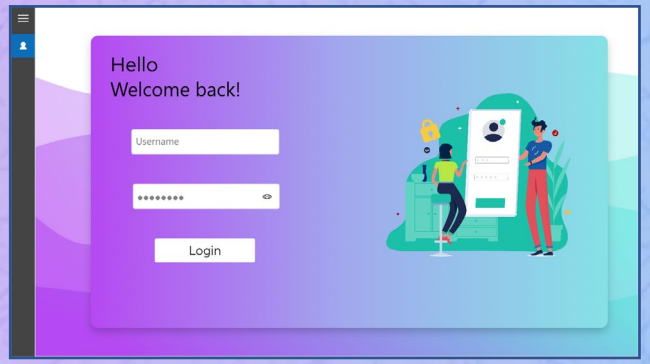
→

Attraverso la chiamata HTTP si ottiene la lista di tutti i parcheggi presenti nel database

LOGIN-LOGOUT

Strutturazione dell'applicazione userfriendly per un utilizzo più intuitivo tramite autenticazione.

Una prima pagina di Login permette di entrare nell'app tramite l'inserimento di un username e password.



Una volta Loggati avremo la di effettuare il Logout o cambiare le nostre credenziali di accesso, aggiornando il database.

MAIN PAGE

Strutturazione dell'applicazione userfriendly per un utilizzo più intuitivo tramite autenticazione.

Incasso
52€

Permette di vedere i parcheggi attualmente disponibili con i relativi incassi e possibilità di eliminazione.

Parcheggi disponibili
52

Permette di visualizzare i posti attualmente liberi per ogni parcheggio con le relative informazioni.

Veicoli Entrati
30

Permette di visualizzare tutti i veicoli registrati ed entrati nel parking online.

PARCHEGGIO ATTUALE

SPECIFICHE

- Implementazione della classe Parcheggio che rappresenti graficamente una griglia composta da righe e colonne.

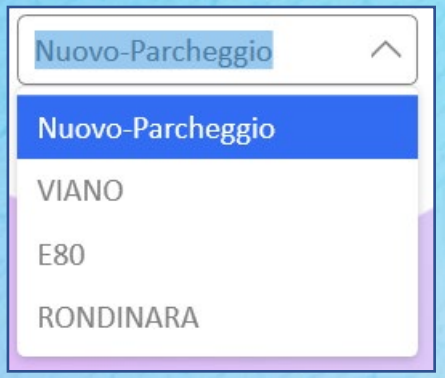
```
namespace WebAPI_Definitivo.Models
{
    32 riferimenti
    public partial class Parking
    {
        0 riferimenti
        public Parking(string parkingId, bool stato, DateTime entryTimeDate, long vehicleId, int infoParkId)
        {
            ParkingId = parkingId;
            Stato = stato;
            EntryTimeDate = entryTimeDate;
            VehicleId = vehicleId;
            InfoParkId = infoParkId;
        }

        1 riferimento
        public Parking()
        {

        }

        2 riferimenti
        public long Id { get; set; }
        12 riferimenti
        public string ParkingId { get; set; }
        7 riferimenti
        public bool Stato { get; set; }
        8 riferimenti
        public DateTime? EntryTimeDate { get; set; }
        7 riferimenti
        public long? VehicleId { get; set; }
        9 riferimenti
        public long InfoParkId { get; set; }

        2 riferimenti
        public virtual InfoParking InfoPark { get; set; }
        1 riferimento
        public virtual Vehicle Vehicle { get; set; }
    }
}
```

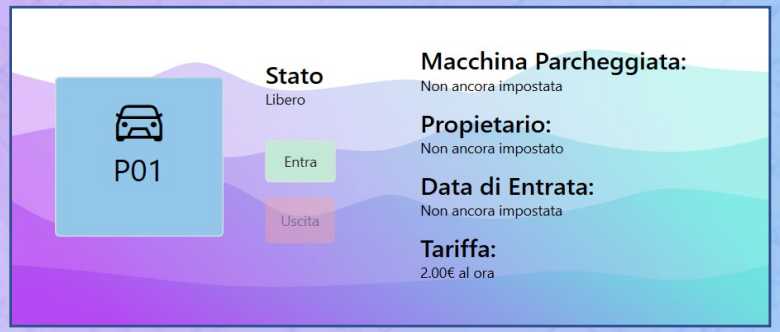


PARCHEGGIO ATTUALE

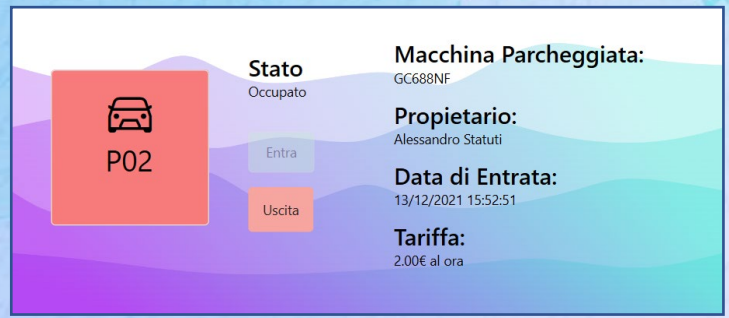
SPECIFICHE

- Implementazione dei metodi entra ed esci per l'uscita e l'entrata dei veicoli.

Il metodo entra permette di inserire le specifiche del veicolo in ingresso e il suo proprietario, aggiornando automaticamente il database.



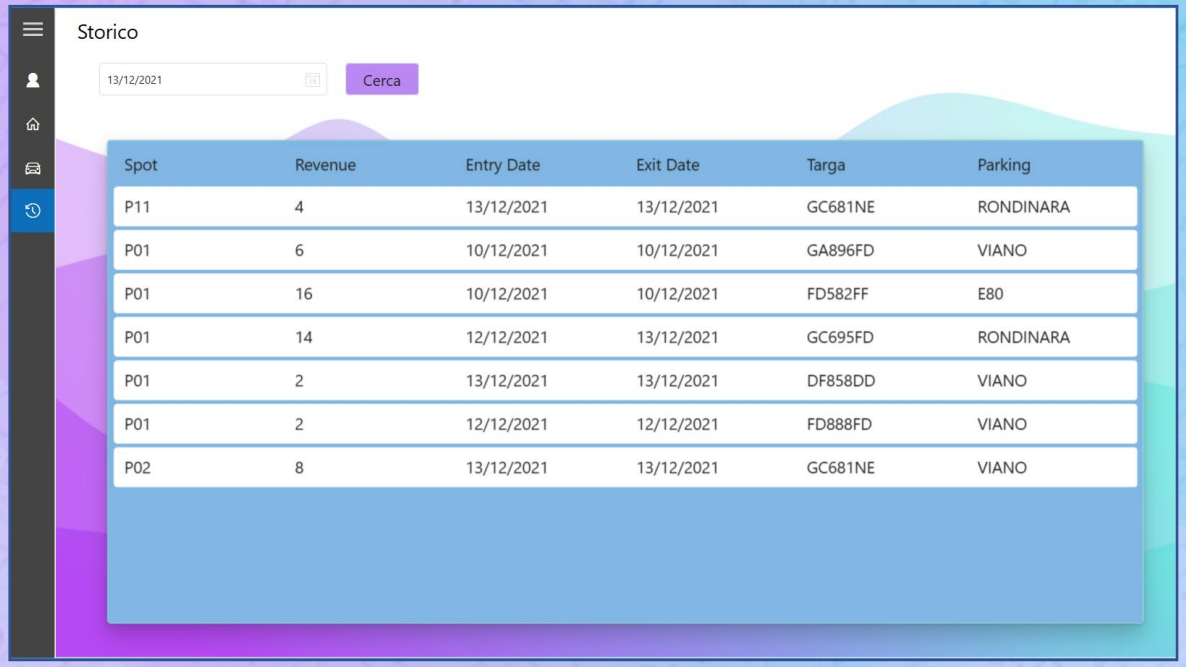
Il metodo esci permette di l'uscita del veicolo parcheggiato, aggiornando la tabella History e calcolando la tariffa.



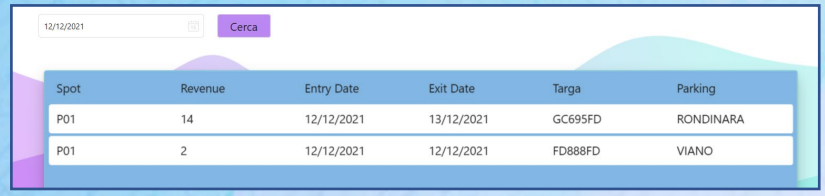
PARCHEGGIO STORICO

SPECIFICHE

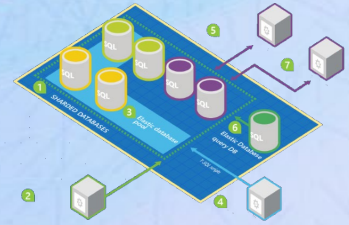
Implementazione di un parcheggio storico contenente tutti i veicoli usciti da tutti i parcheggi con possibilià di ricerca per data.



Spot	Revenue	Entry Date	Exit Date	Targa	Parking
P11	4	13/12/2021	13/12/2021	GC681NE	RONDINARA
P01	6	10/12/2021	10/12/2021	GA896FD	VIANO
P01	16	10/12/2021	10/12/2021	FD582FF	E80
P01	14	12/12/2021	13/12/2021	GC695FD	RONDINARA
P01	2	13/12/2021	13/12/2021	DF858DD	VIANO
P01	2	12/12/2021	12/12/2021	FD888FD	VIANO
P02	8	13/12/2021	13/12/2021	GC681NE	VIANO



Spot	Revenue	Entry Date	Exit Date	Targa	Parking
P01	14	12/12/2021	13/12/2021	GC695FD	RONDINARA
P01	2	12/12/2021	12/12/2021	FD888FD	VIANO



Ora proviamo sul campo l'applicazione