Bus Reservation Management System

The Goal of the Project:

The purpose of this project is to create a bus reservation management system. Thanks to the interface, it is aimed to easily enter passenger information, driver information, bus information, bus information, reservation operations, seat reservation operations and comment making feature into the system and save this information to the database. The entered data may be simply accessible thanks to the interface's facilitation. Information can also be changed and removed. The database is updated simultaneously with these updates. As a result, the data is accurately and dependably captured.

Steps of the Projects:

1. ERDPlus was used to first design the project's relational structure.
2. SQL was used to import the ERDPlus-created schema into the database.
3. Using the C# programming language, an interface was developed.
4. Using the written scripts, the interface generated in C# was connected to the tables established in the database.
5. The interface's adding, removing, updating, and choosing functions were each put to the test.
6. Lastly, it was checked to see if the database updated simultaneously with the data changes made to the interface.
7. The project that passed the testing has been the subject of a report and a presentation.

ER Diagram:

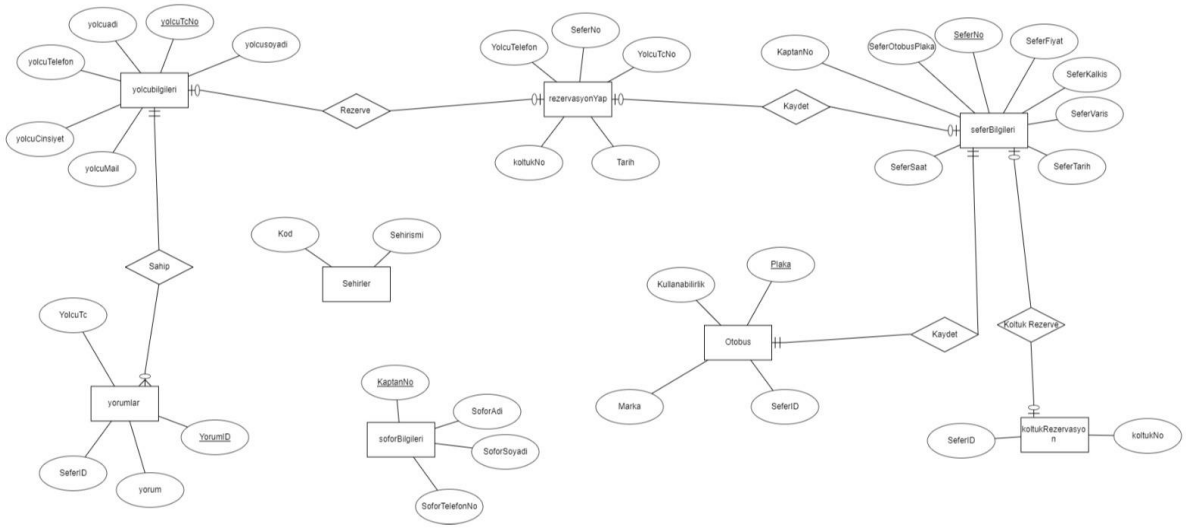

Figure 1: ER Diagram

Relational Schema:

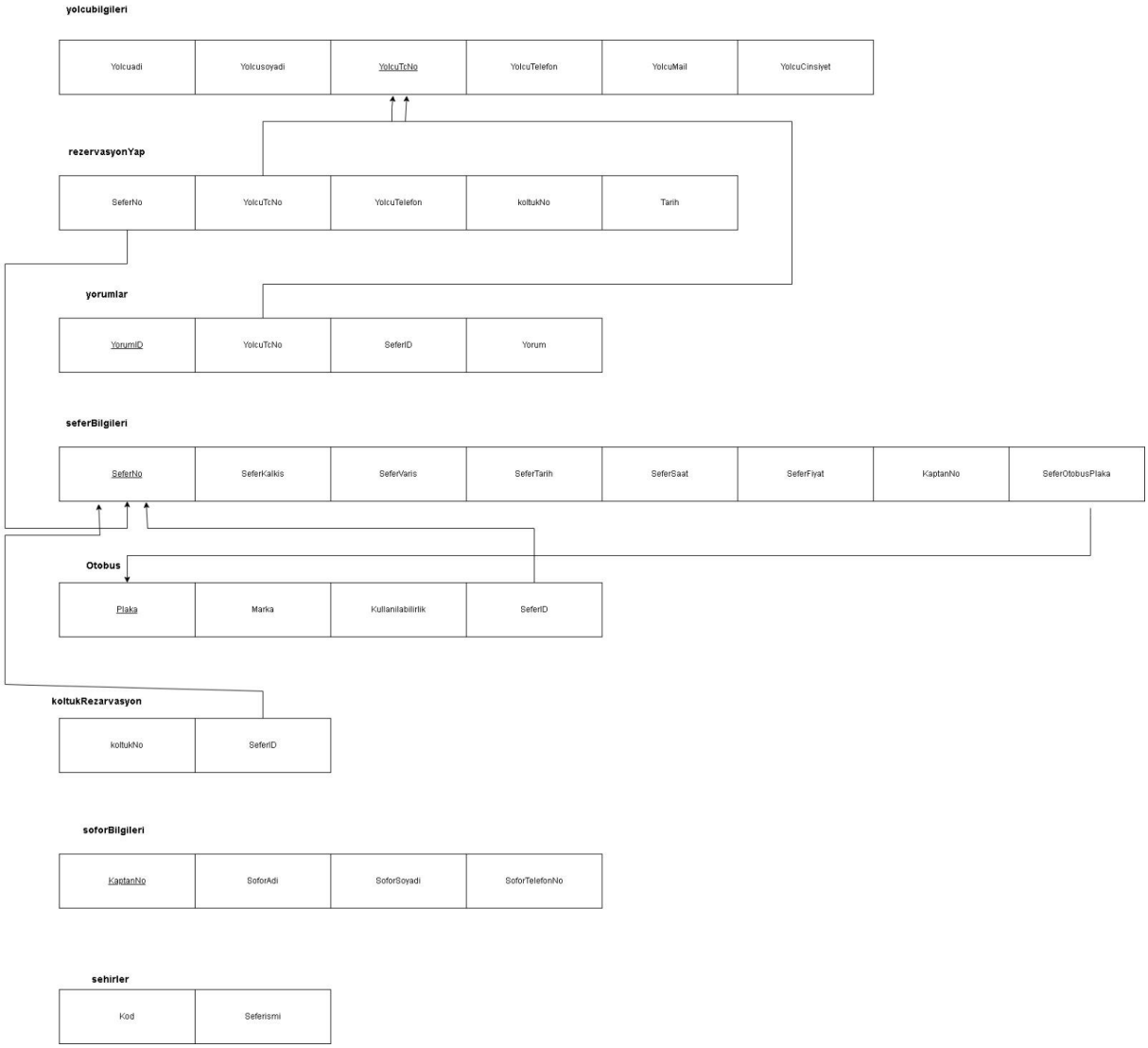

Figure 2: Relational Schema

As seen in the relational schema above, the connection between the tables is as follows;

- yolcubilgileri table and rezervasyonYap linkedby "YolcuTcNo".
- yolcubilgileri table and yorumlar linkedby "YolcuTcNo".
- seferBilgileri table and rezervasyonYap linkedby "SeferNo".
- seferBilgileri table and Otobus linkedby "SeferNo".
- Otobus table and seferBilgileri linkedby "Plaka".

As can be seen from here, there are 6 relationships between the 8 tables.

The screen has seen in Figure 3 when the application is launched.

Figure 3: Home Page

There are operations such as passenger information, driver information, flight information, bus information, reservation operations, seat reservation operations and comment-making features. In addition to these, operations such as adding, updating and deleting can also be performed in these operations.

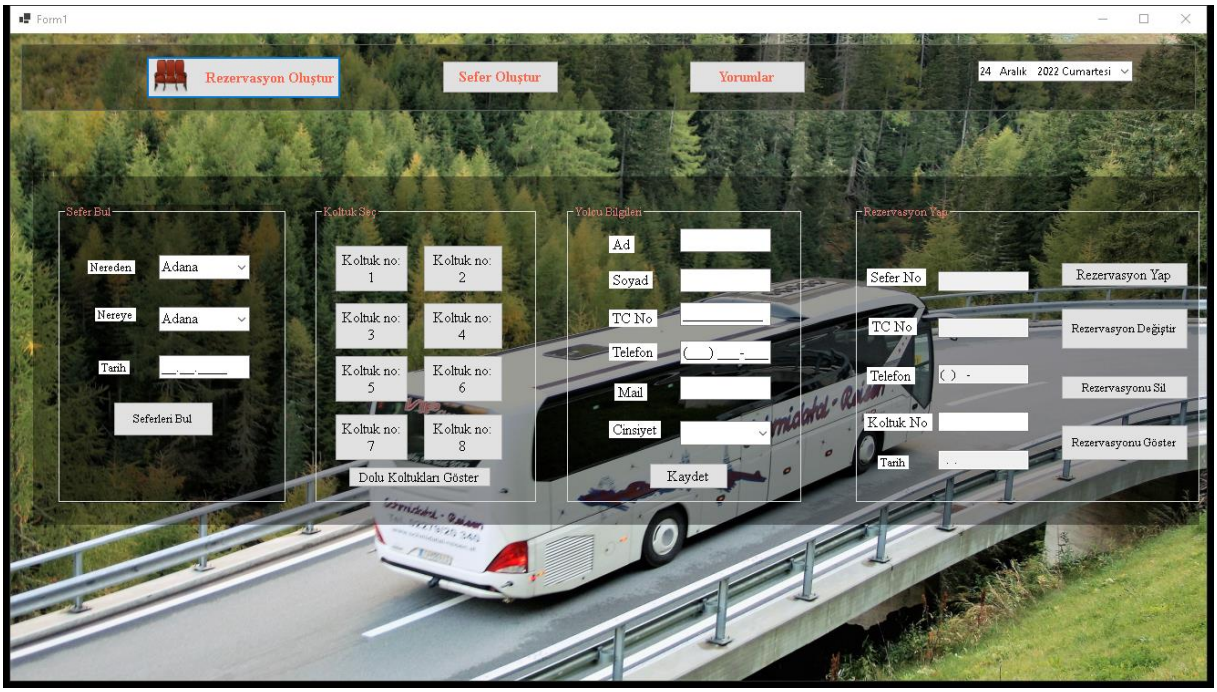

Figure4: Booking Creation Page

The expedition creation page appears in figure 5.

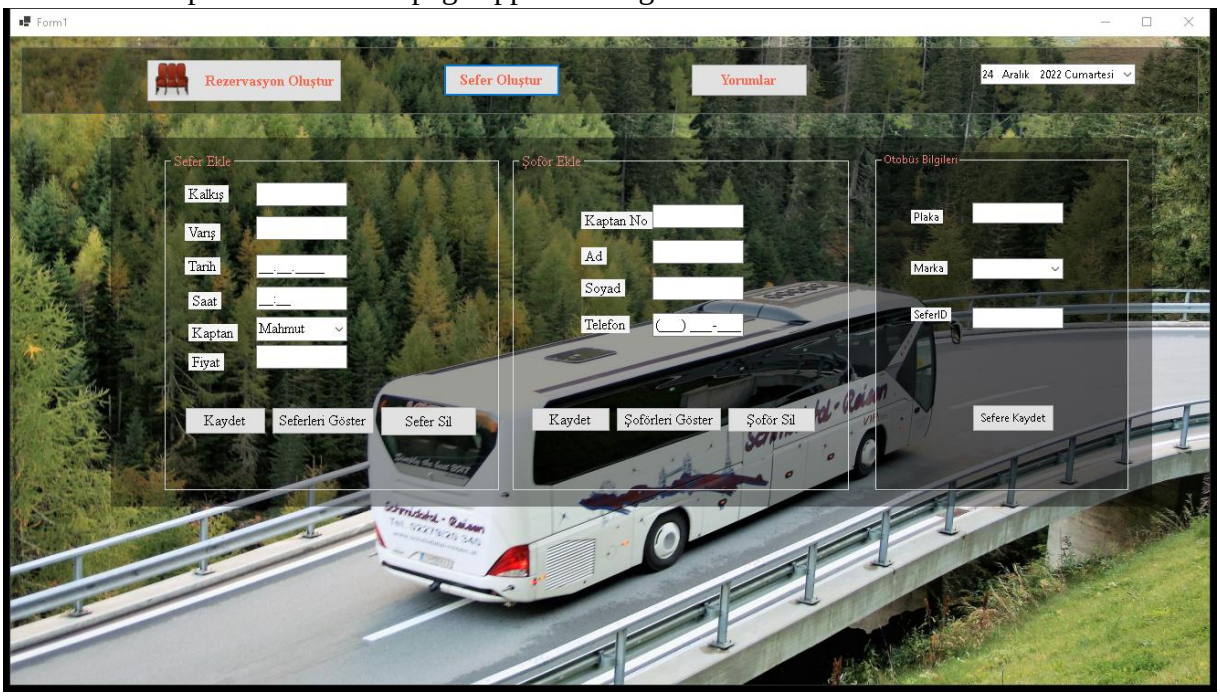

Figure 5: Expedition Creation Page

The comments page appears in figure 6.

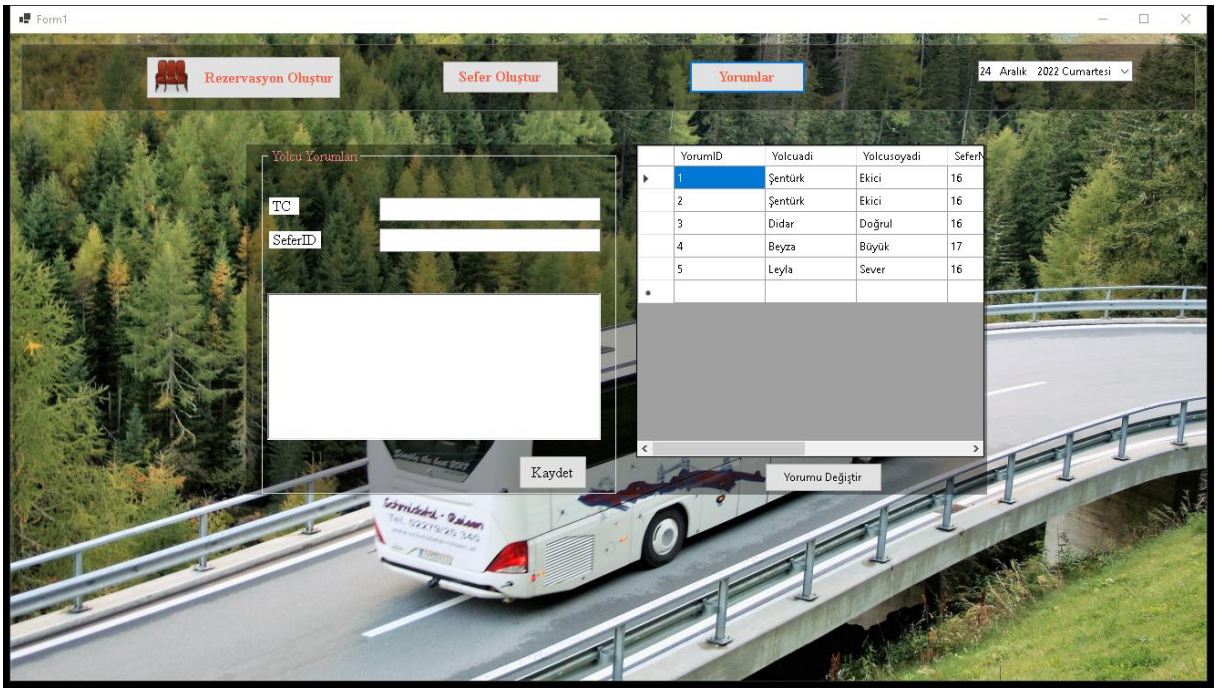

Figure 6: Comments Page

Figure 7 shows the list of available bus services and the form with information about them. There is a clearer view of this list in Figure 8.

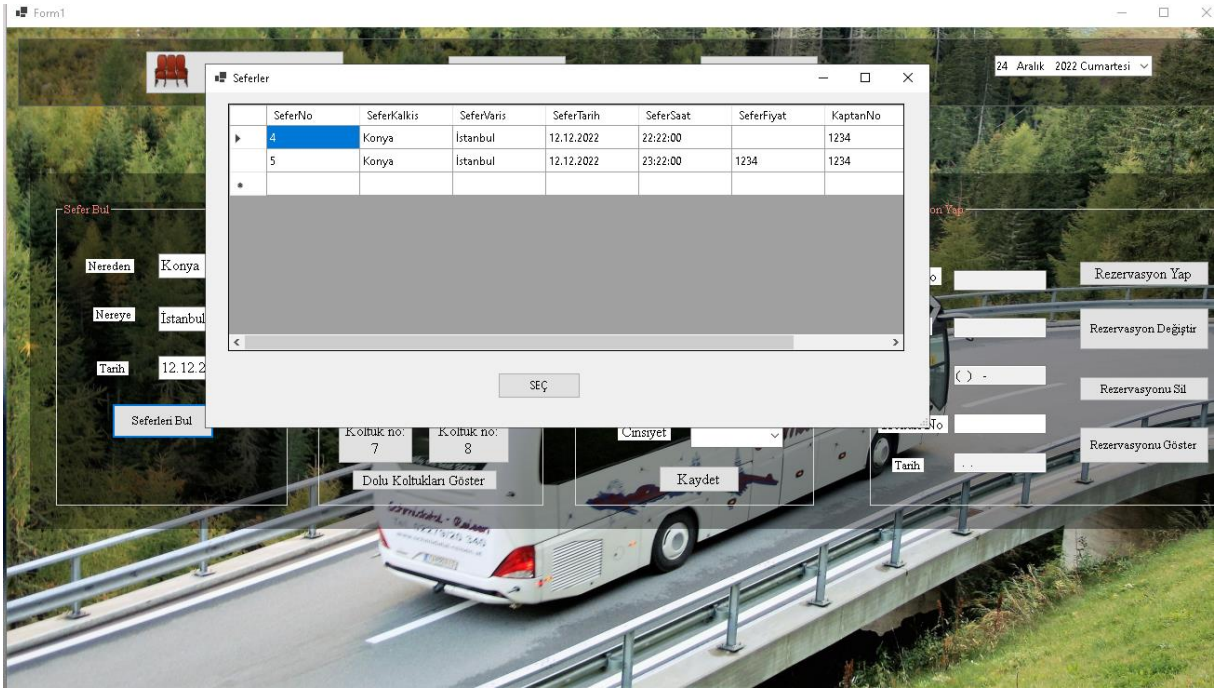

Figure 7: Form of Available Bus Services

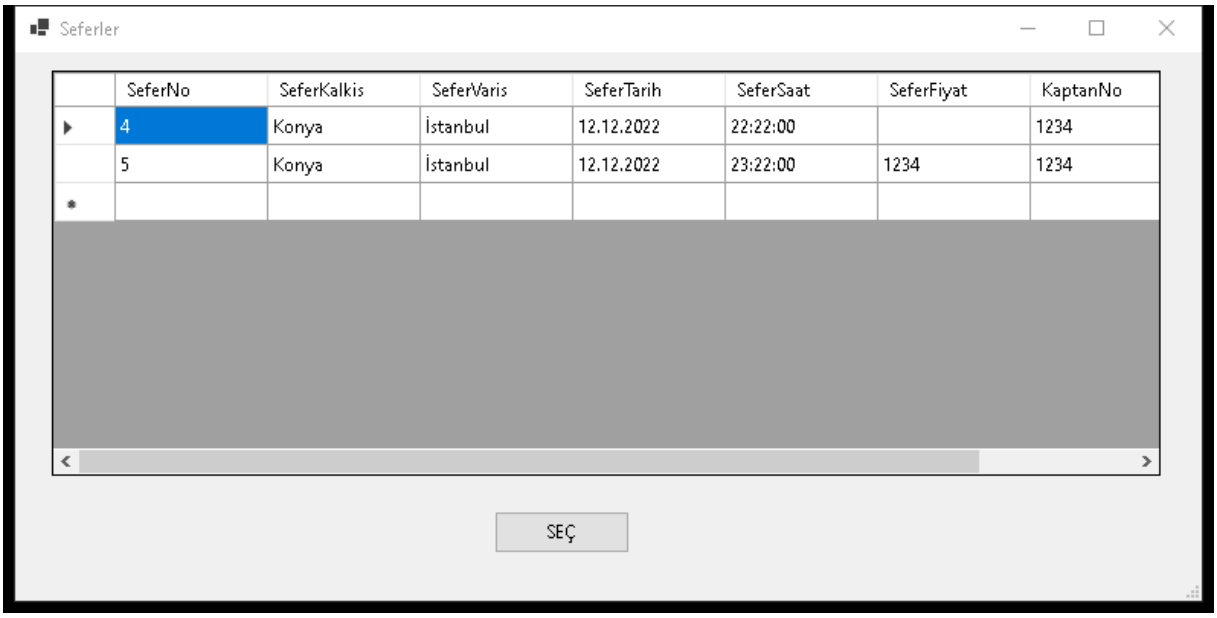

Figure 8: List of Available Bus Services

This happens when we click on the desired bus service in the bus time selection process and then press the select button. When it takes place, an information message is sent by the system stating that the expedition has been selected. These processes are shown in figure 9 and figure 10.

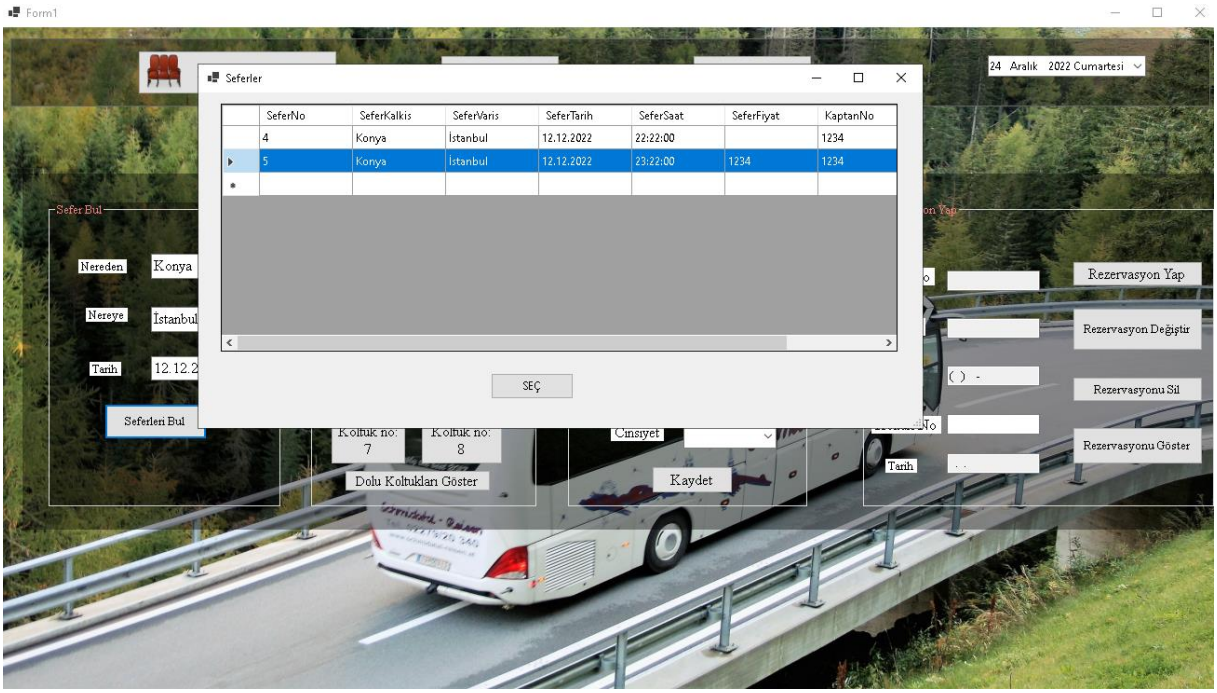

Figure 9: Choosing Bus Services

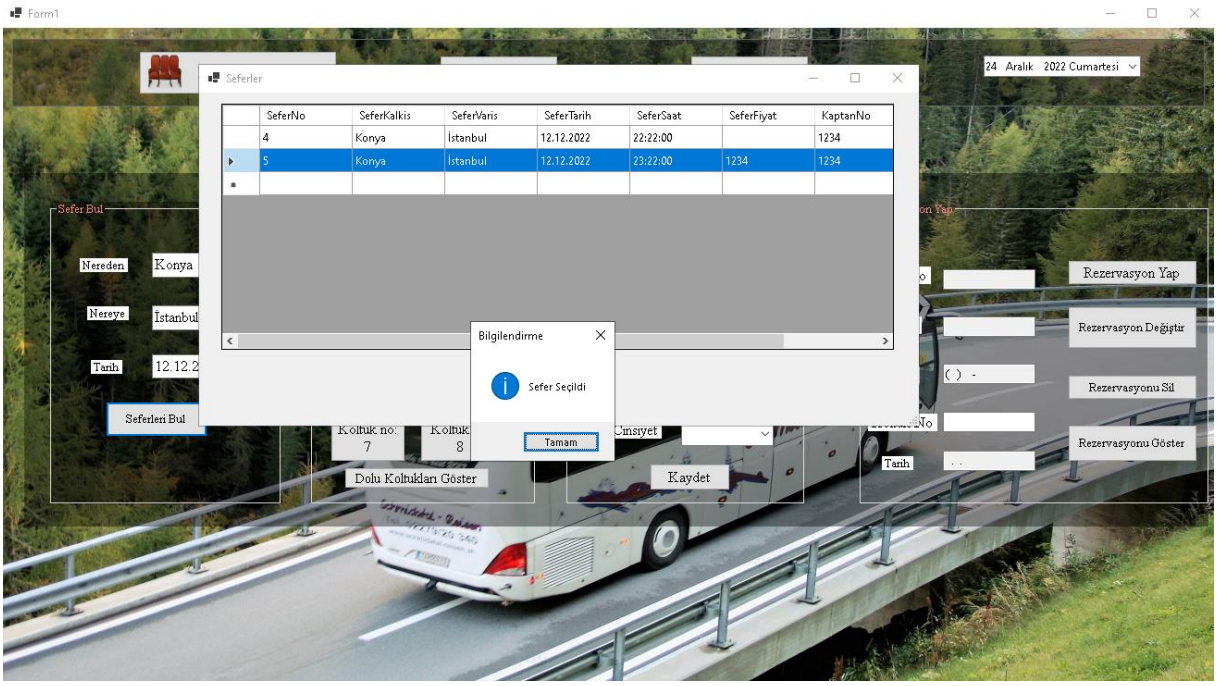

Figure 10: Information Message About The Selection of Bus Services

Figure 11 shows the full seats. After the passenger information is entered in full and the save button is clicked, the system sends an information message that the passenger information has been saved. The visuals of this action are shown figure 12.

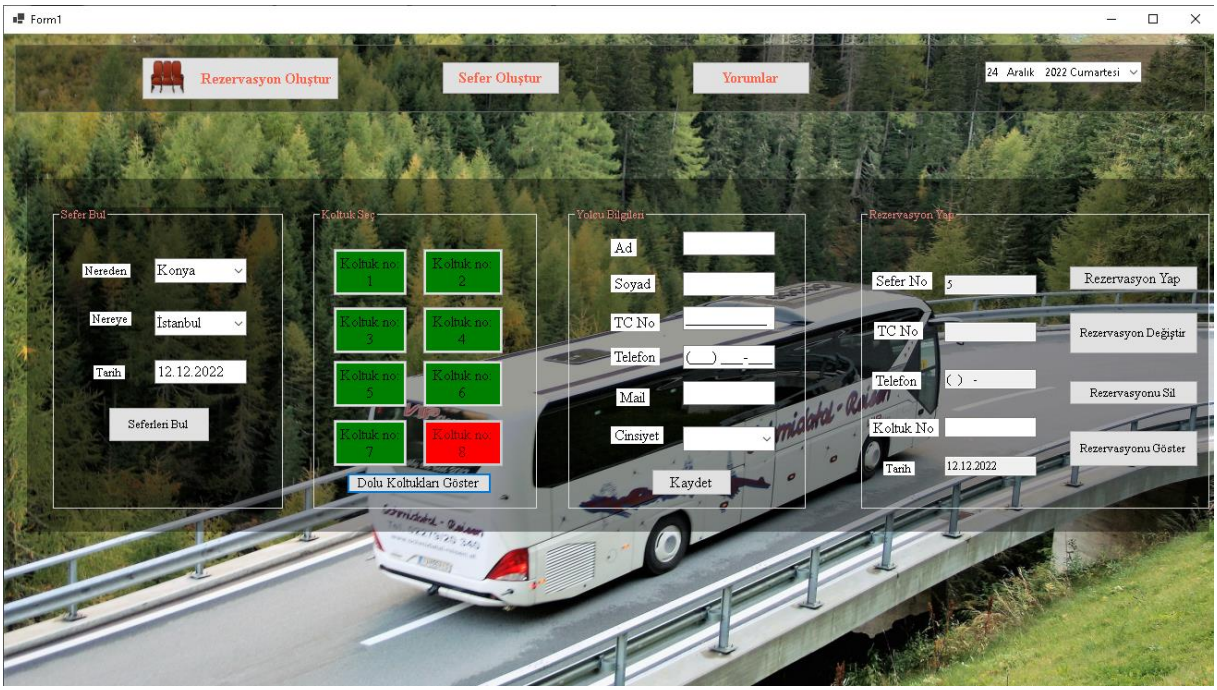

Figure 11: Showing Full Seats

```
create procedure koltukVarMi
@KoltukNumarasi nvarchar(5),
@SeferNumarasi        int
as begin
declare @sonuc int
        If EXISTS(Select * from koltukRezarvasyon where koltukNo=@KoltukNumarasi and
SeferID=@SeferNumarasi )
                begin
                        set @sonuc=1
                end
        else
                begin
                        set @sonuc=0
                end
        return @sonuc
end
```

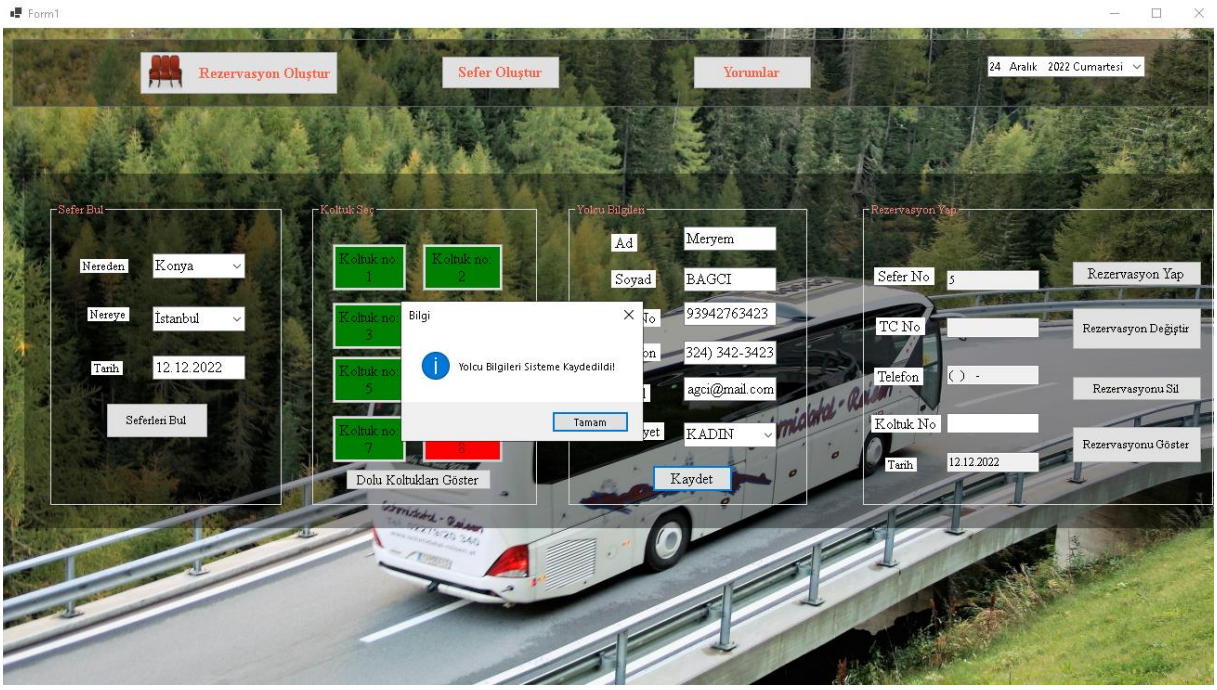

Figure 12: Information Message That Passenger Information Has Been Recorded In The System

To make a reservation, it is necessary to enter the desired data in the Make a reservation panel completely after filling in the select seat and passenger information panel. After filling in these data, when we click on the make a reservation button, our transaction is completed and the system sends us an information message that it has been saved. These processes are clearly seen in figure 13 and figure 14.

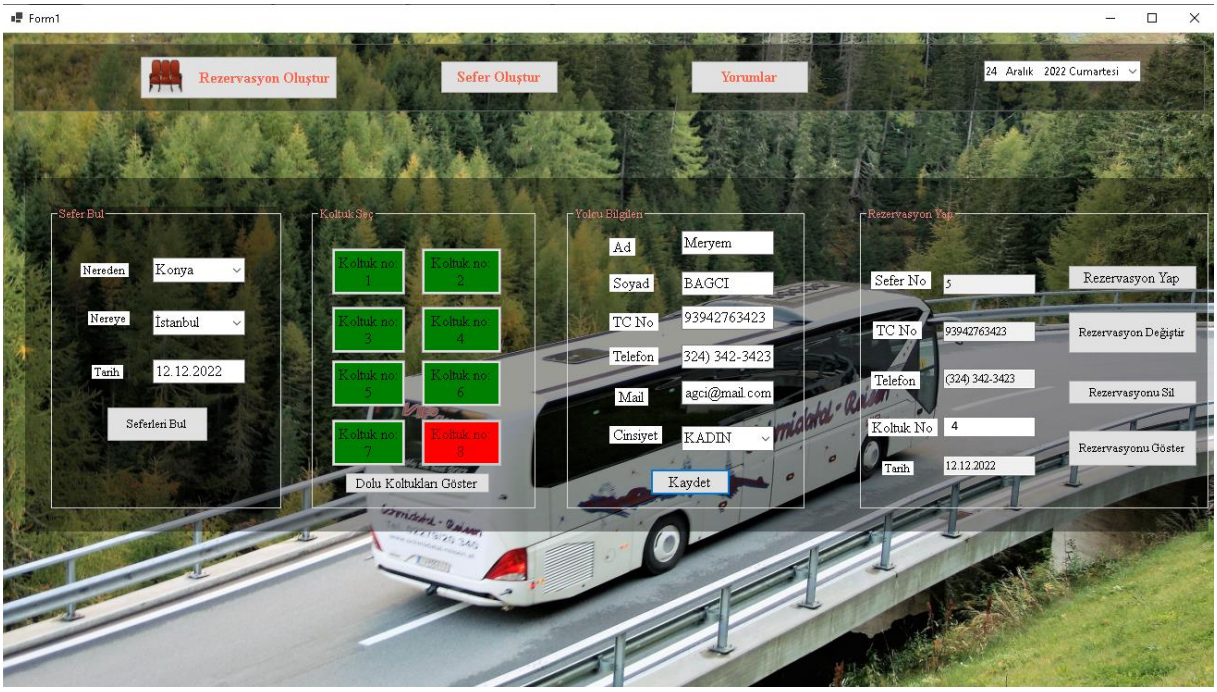

Figure 13: The Appearance of The Passenger Information After It Is Recorded In The System

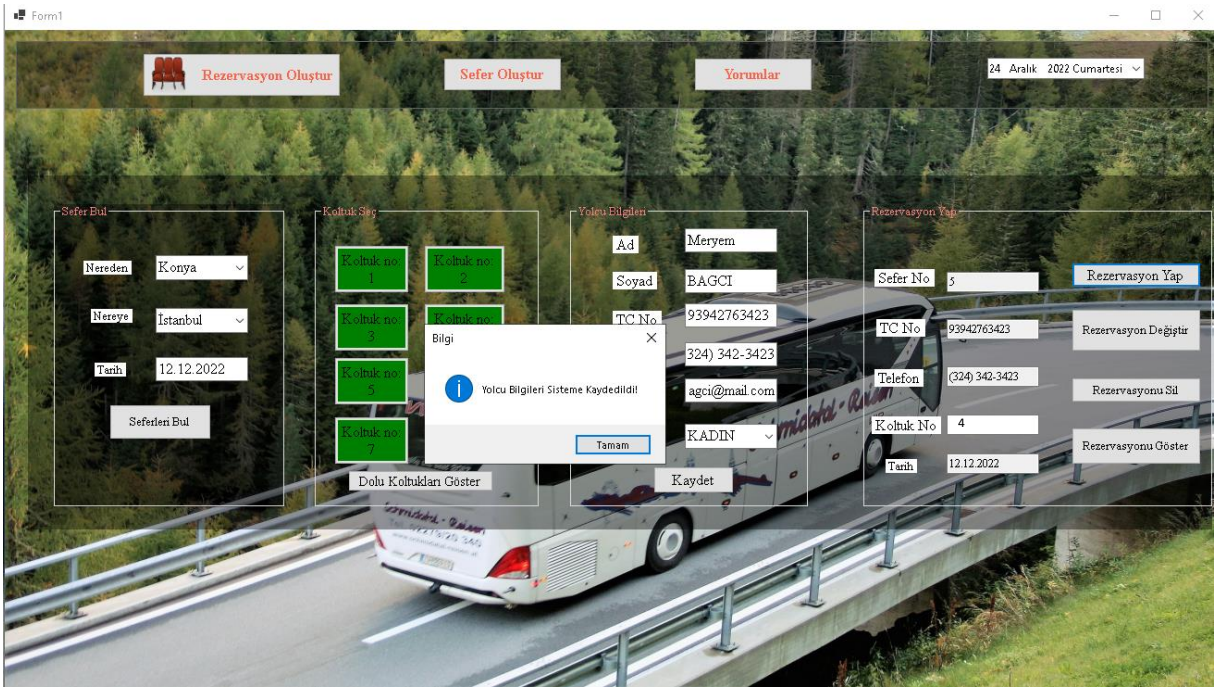

Figure 14: When The Make A Reservation Button Is Clicked

As it appears in figure 15, it can be questioned whether there is a booking record with a passenger ID.

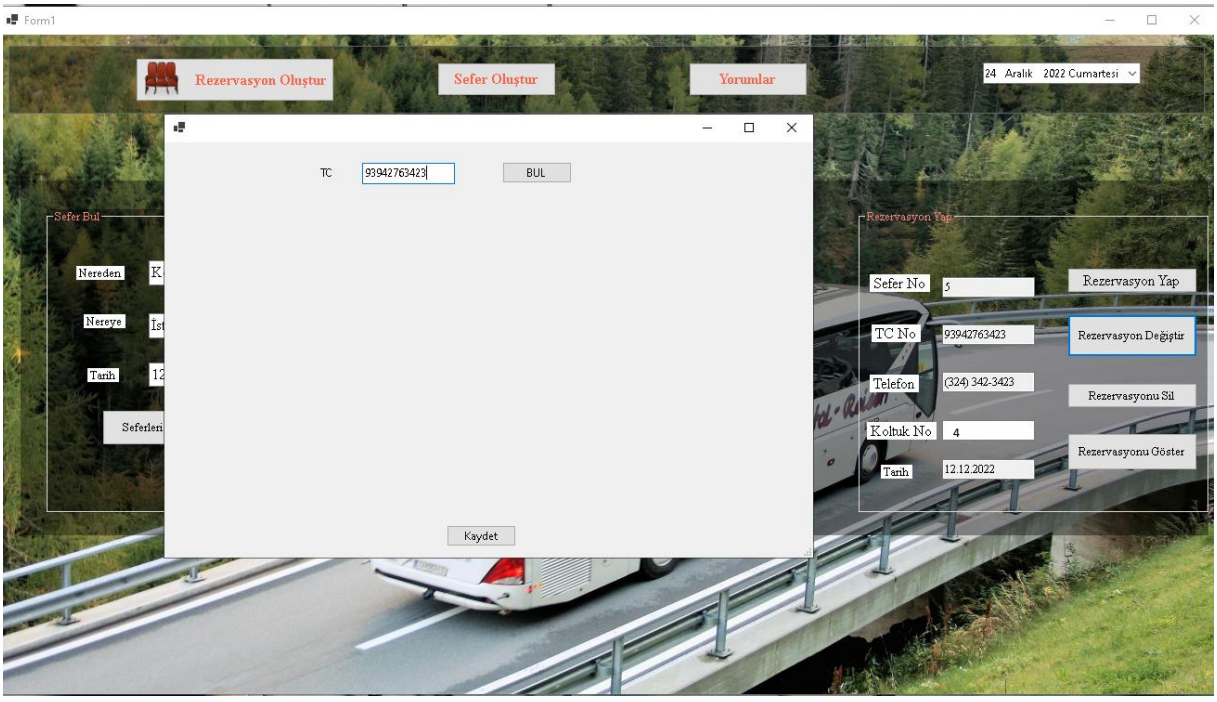

Figure 15: Questioning Whether There Is A Reservation Registration With The Passenger ID

After entering the missing time information you want to find in a form and clicking the find time button, if there is a time, the show time list is opened and the time and information you are looking for are shown to the user by the system. These events are shown in figure 16 and figure 17.

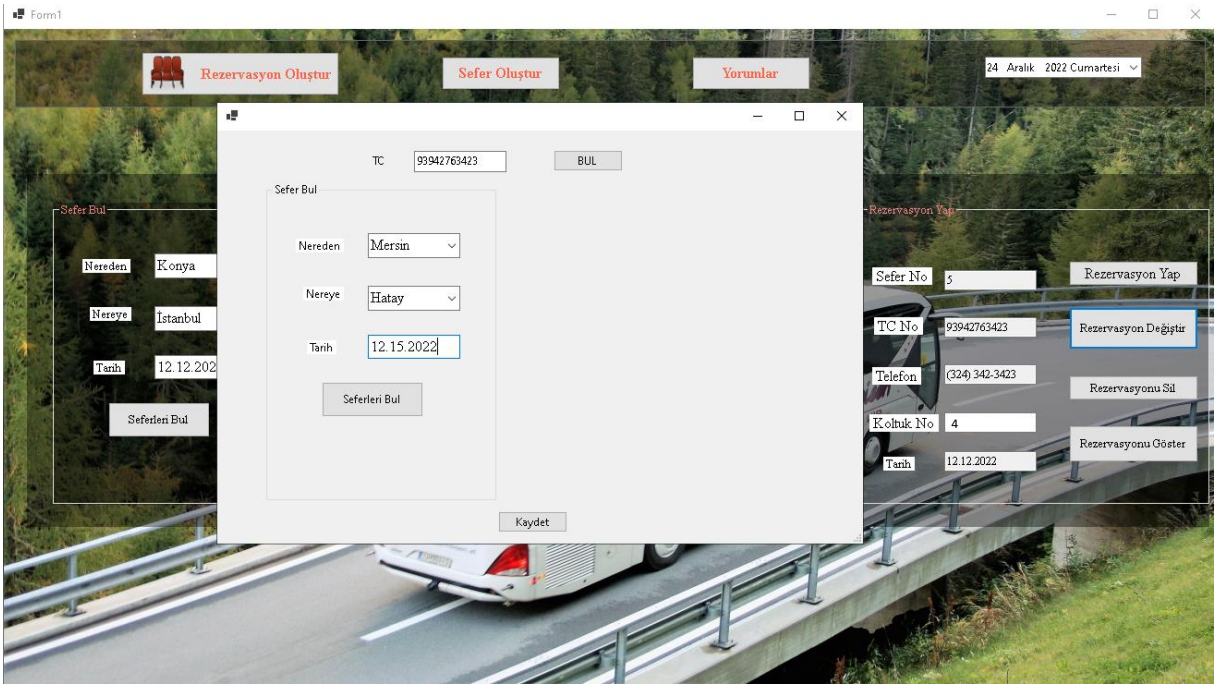

Figure 16: When The Booking Record Is Found

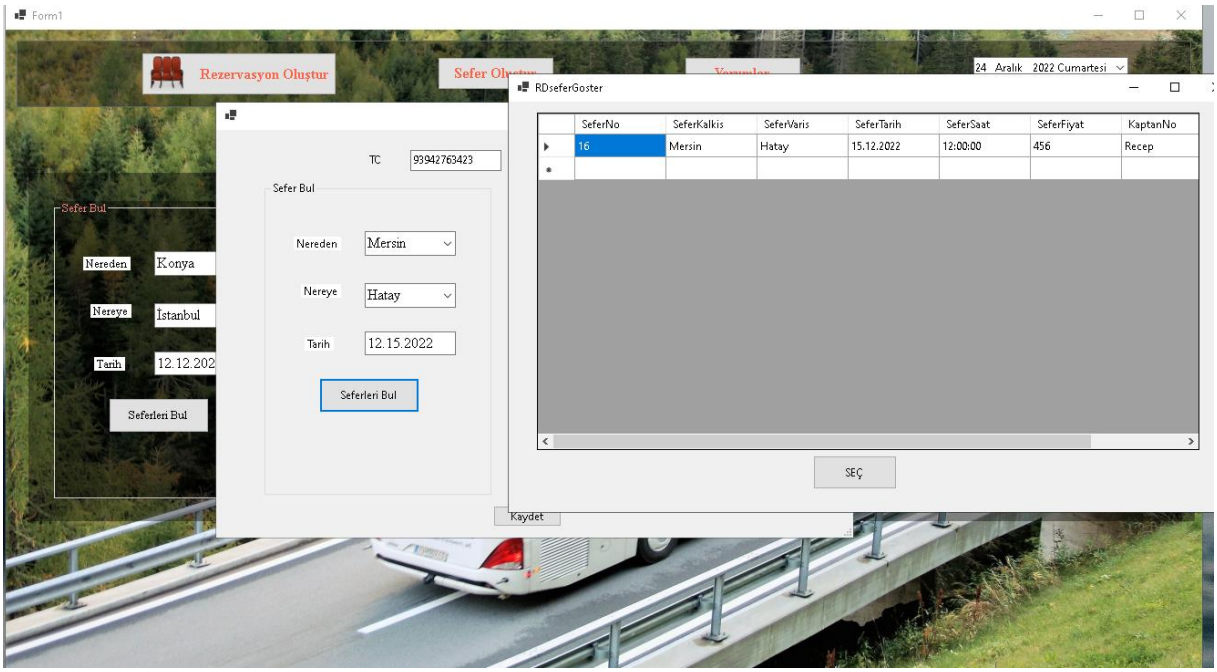

Figure 17: Changing The Booking Of A Bus With A Booking Record

After clicking on the bus service you want to change in the show time panel and clicking the select button, the user receives an information message that the service has been selected. Then the user can change the information in the reservation that he wants to change as he wants. (Provided that the user fills in all the requested information)

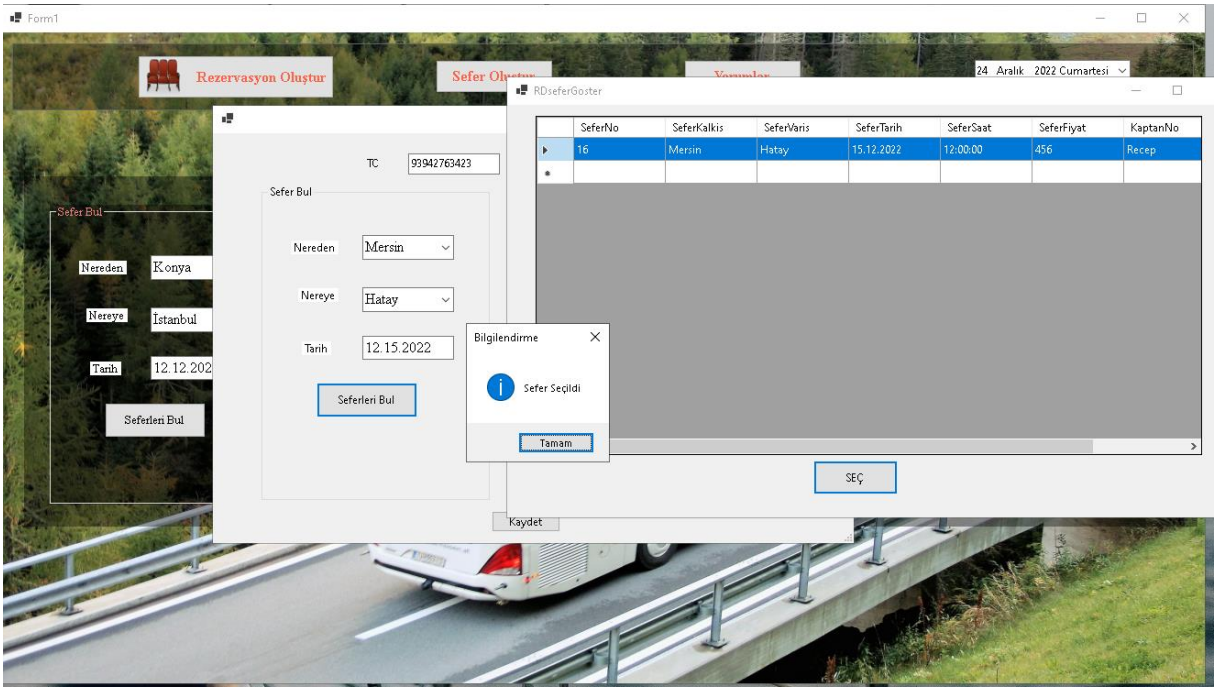

Figure 18: An Information Message When The Reservation Of A Bus With A Reservation Record Is Changed

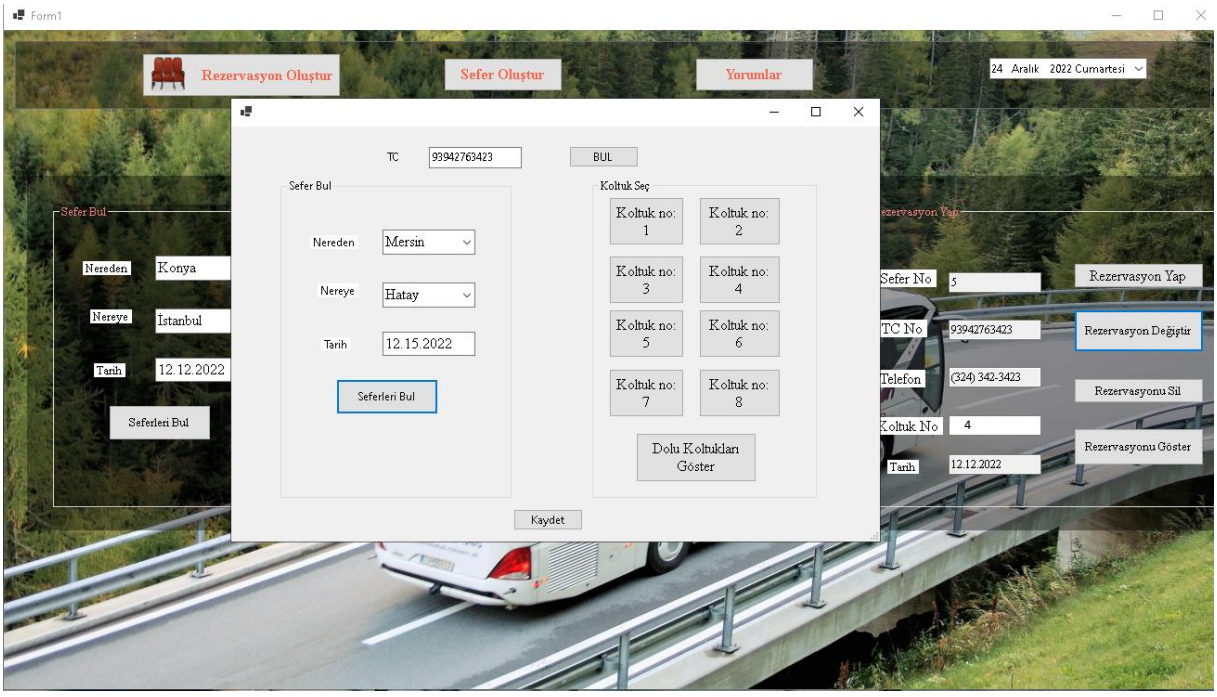

Figure 19: Selecting A New Seat At The Time The Reservation Record Was Changed

Seat selection must be made again in the registration whose reservation has been changed. When choosing jeans, the seats filled by the system are also shown.

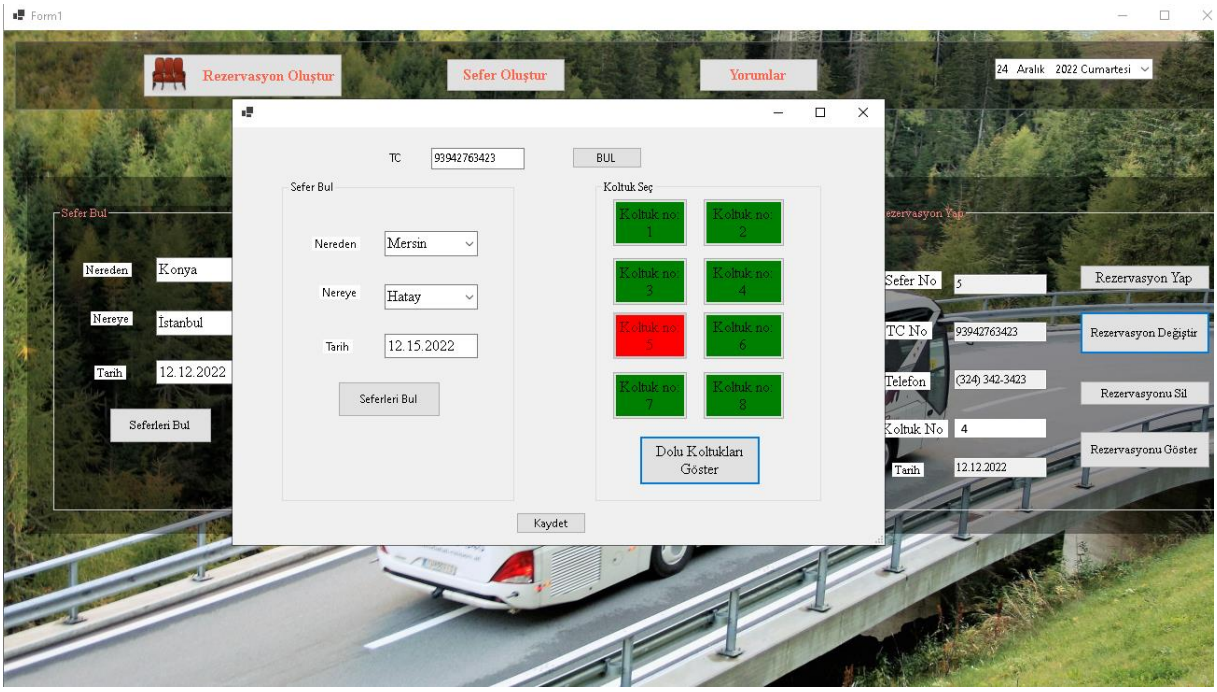

Figure 20: Showing The Full Seats In The Registration Whose Reservation Has Been Changed

The save button must be pressed for the expedition to be updated successfully. Then the system will send an notification message that it has been updated.

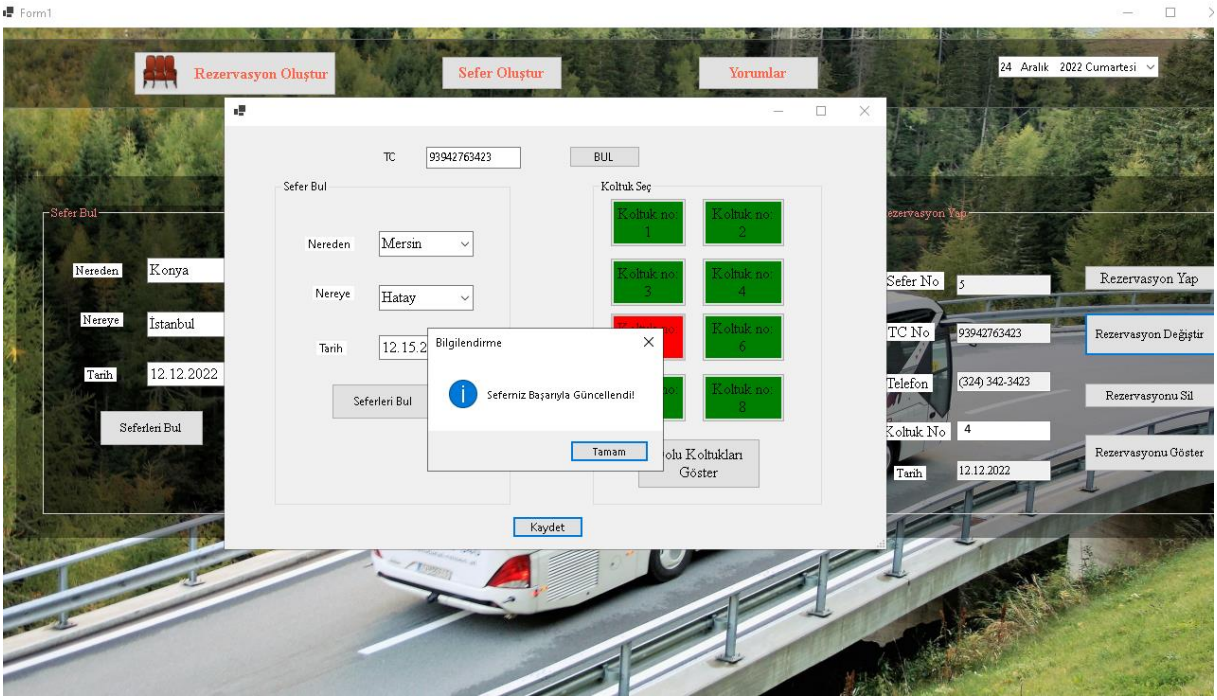

Figure 21: Information Message That The Expedition Has Been Updated

```
create trigger RezarvasyonKoltukDegistir

on rezervasyonYap

after update

as

declare @seferNumarası int

declare @koltukNumarası nvarchar(5)

select @seferNumarası=SeferNo from rezervasyonYap

select @koltukNumarası=koltukNo from rezervasyonYap

update koltukRezarvasyon set koltukNo=@koltukNumarası where
koltukNo=@koltukNumarası and SeferID= @seferNumarası

update koltukRezarvasyon set SeferID=@seferNumarası where SeferID= @seferNumarası
and koltukNo=@koltukNumarası
```

In order to delete the reservation, the user's ID and sefer id are required. When the data is entered completely, when it matches the one in the system, a confirmation message is sent to the user for the last time about whether he wants to delete it. When the user ID and the time ID do not match the one in the system, an information message is sent that the record has not been found, as in figure 25. If the user confirms this message, the reservation in the system will be deleted and an information message will be sent stating that it has been deleted.

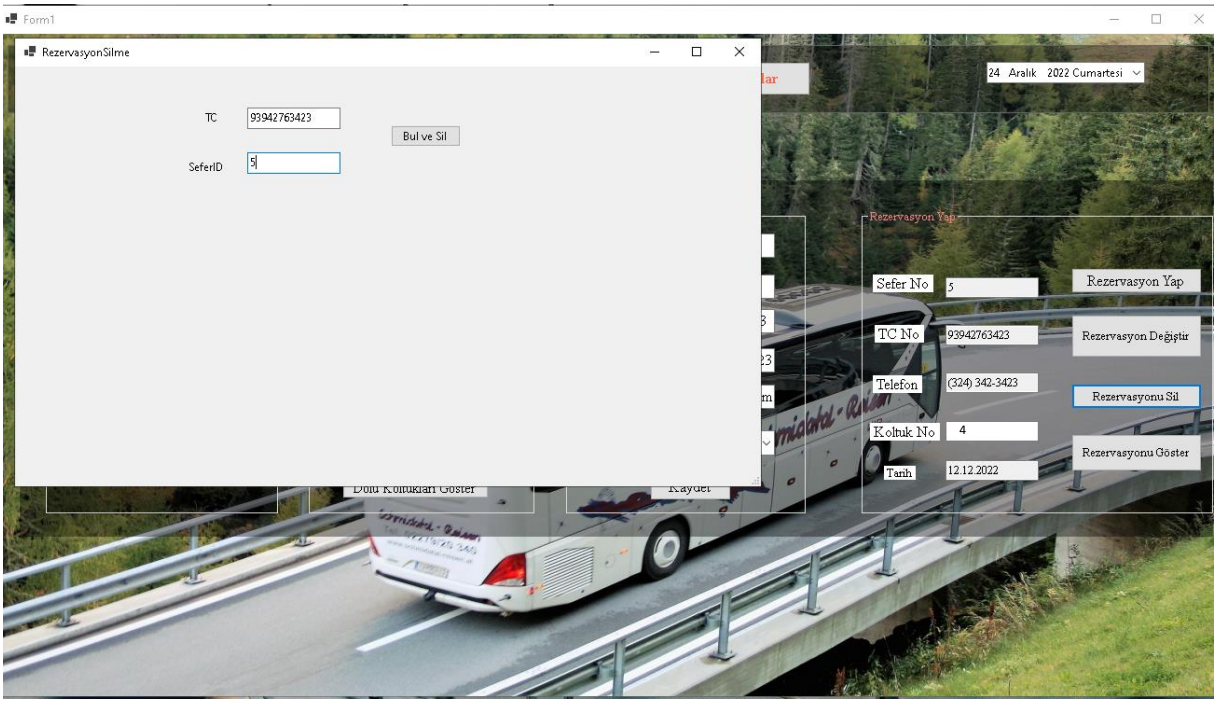

Figure 22: Deleting A Reservation

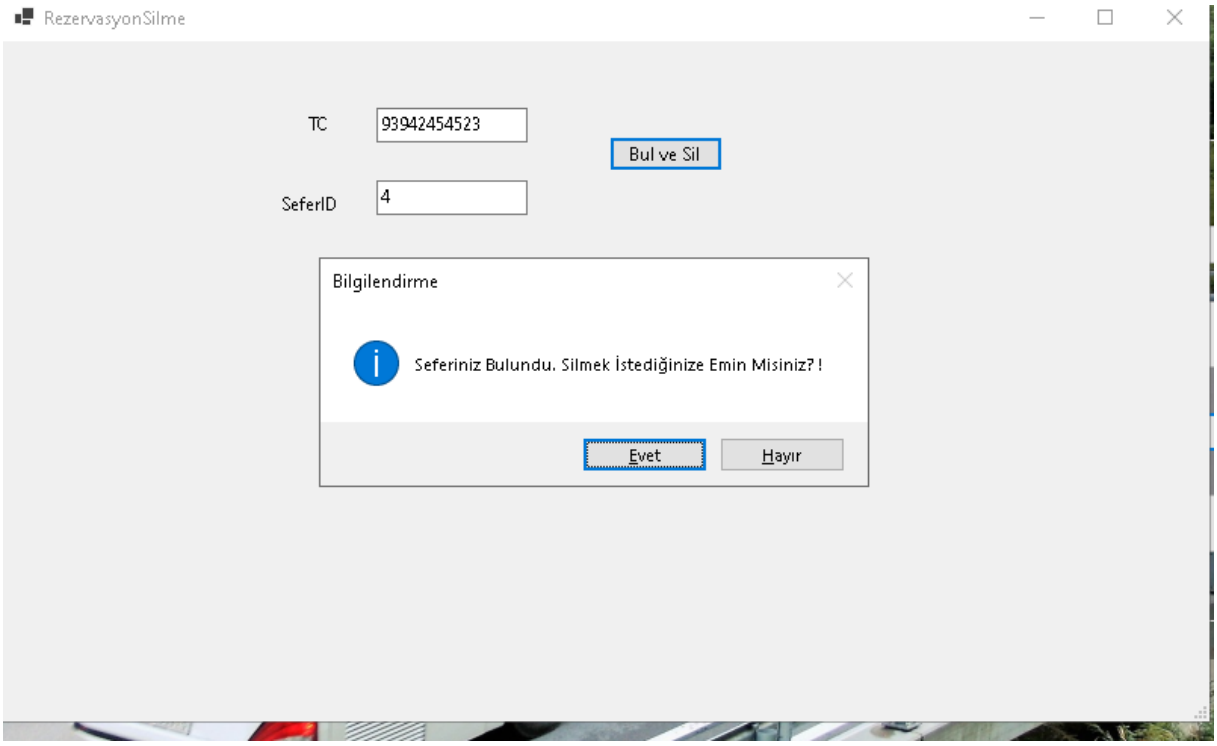

Figure 23: The Last Confirmation Message Before Deletion

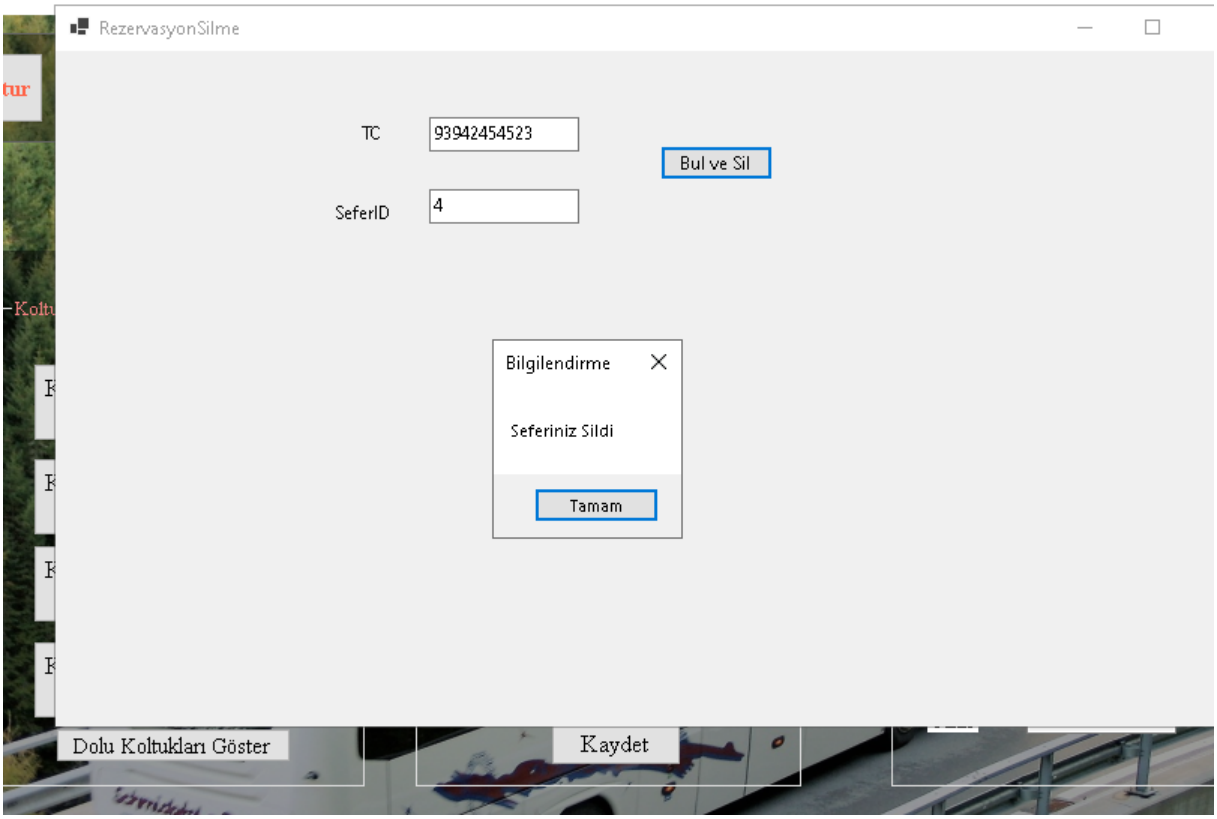

Figure 24: Information Message That The Expedition Has Been Deleted

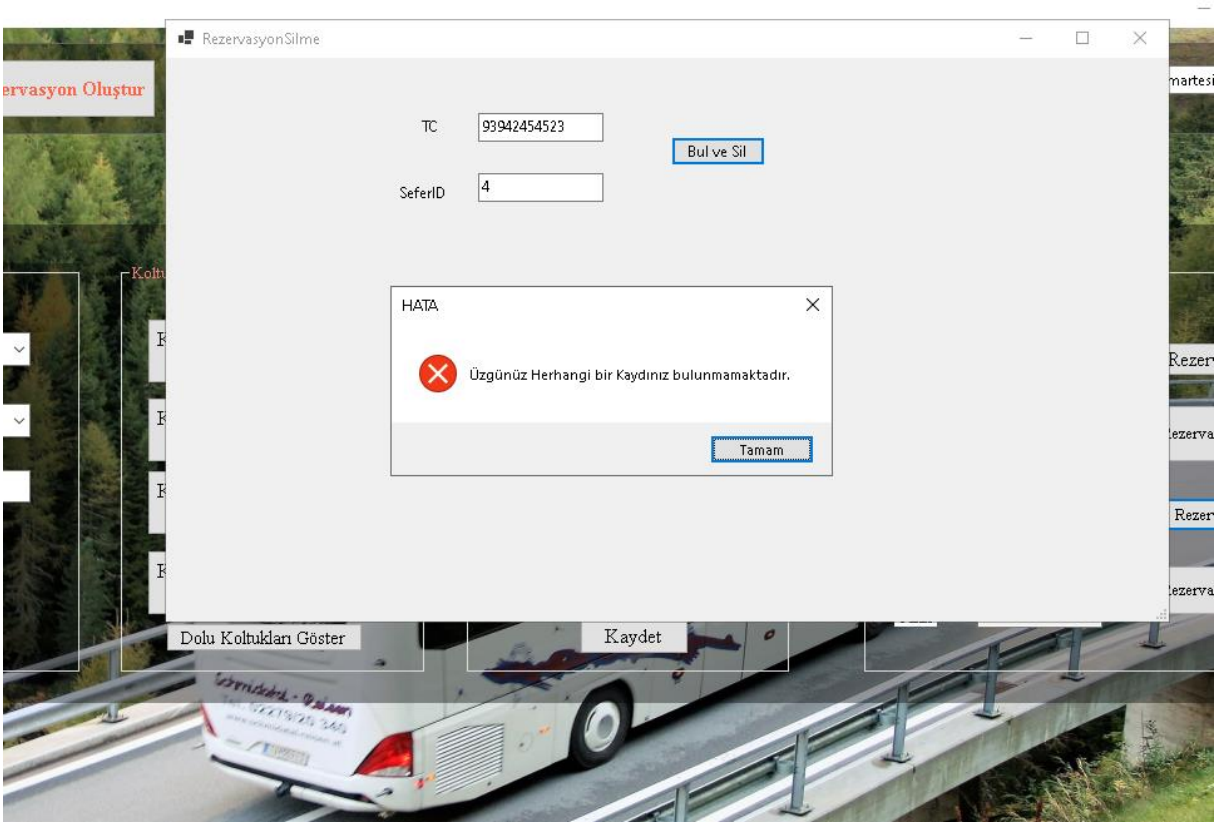

Figure 25: When The Bus Trip Registration And ID Do Not Match

```
create trigger RezarvasyonKoltuklaSil
on rezervasyonYap
after delete
as
declare @seferNumarası int
declare @koltukNumarası nvarchar(5)
select @seferNumarası=SeferNo from rezervasyonYap
select @koltukNumarası=koltukNo from rezervasyonYap
delete from koltukRezarvasyon where koltukNo=@koltukNumarası and
SeferID=@seferNumarası
```

By entering the passenger ID from the reservation show panel, the registered reservations in the system are listed.

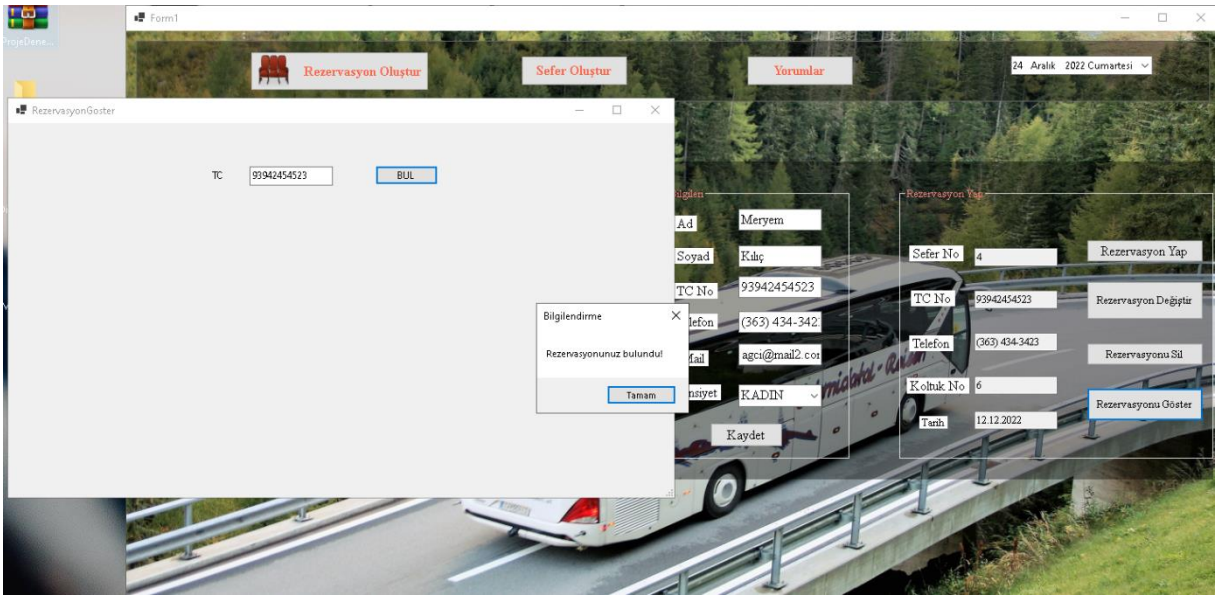

Figure 26: An Information Message When A Reservation Is Found

```
CREATE VIEW viewYolcuRezarvasyon AS

SELECT
SB.SeferNo,Yolcuadi,Yolcusoyadi,YolcuCinsiyet,ry.YolcuTcNo,koltukNo,SeferTarih,SeferS
aat,SeferKalkis,SeferVaris,SeferFiyat

FROM rezervasyonYap as RY inner join seferBilgileri as SB on RY.SeferNo= SB.SeferNo

inner join yolcubilgileri as YB on RY.YolcuTcNo= YB.YolcuTcNo


create  procedure KontrolSeferID

@SeferNo int,

@TC nvarchar(11)

AS

BEGIN

declare @sayac2 int

        If EXISTS(select SeferNo,YolcuTcNo  from rezervasyonYap where
YolcuTcNo=@TC and SeferNo=@SeferNo)

                begin

                        set @sayac2=1

                END

        else

                BEGIN
```

```
                SET  @sayac2=0
            END
        RETURN @sayac2
END
```

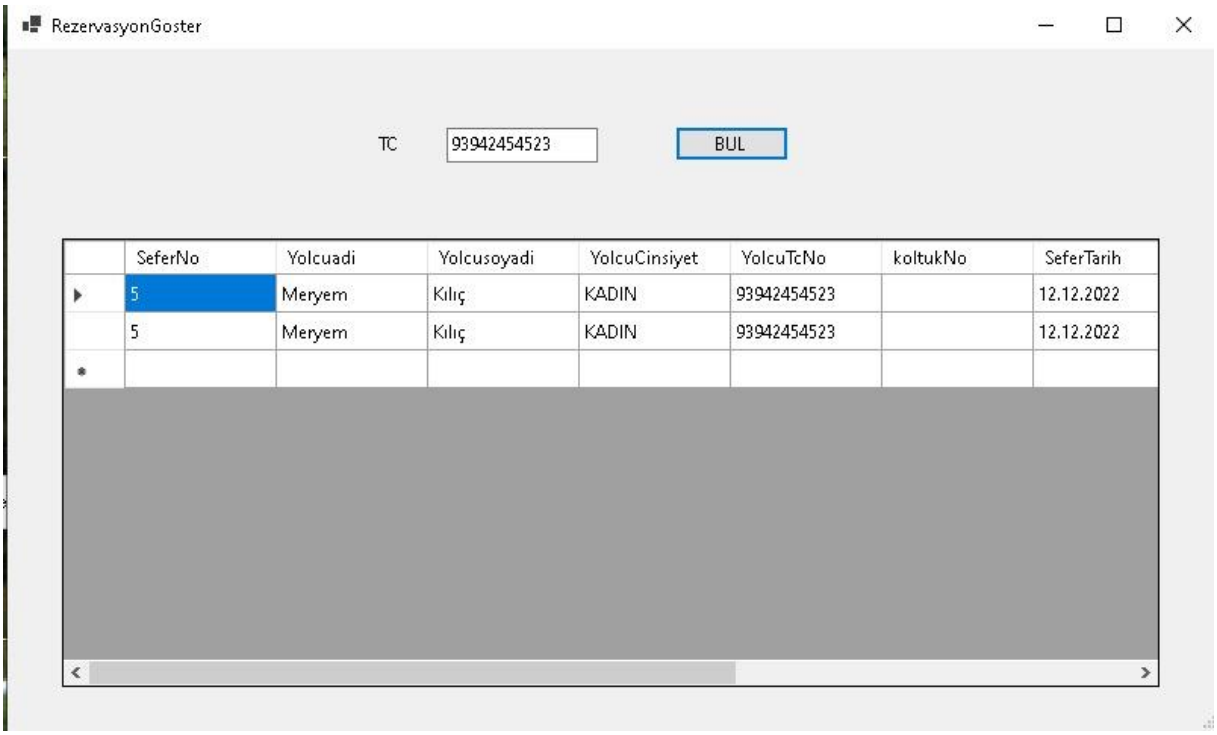

Figure 27: Reservation Show Panel

To add a new bus service, the data in the Add bus service panel must be entered completely. The save button must be pressed to perform the operation. It becomes final when an information message is received that the transaction has been recorded that it has been performed. The added new bus service is automatically added to the list where the buses in the system are shown to the user.

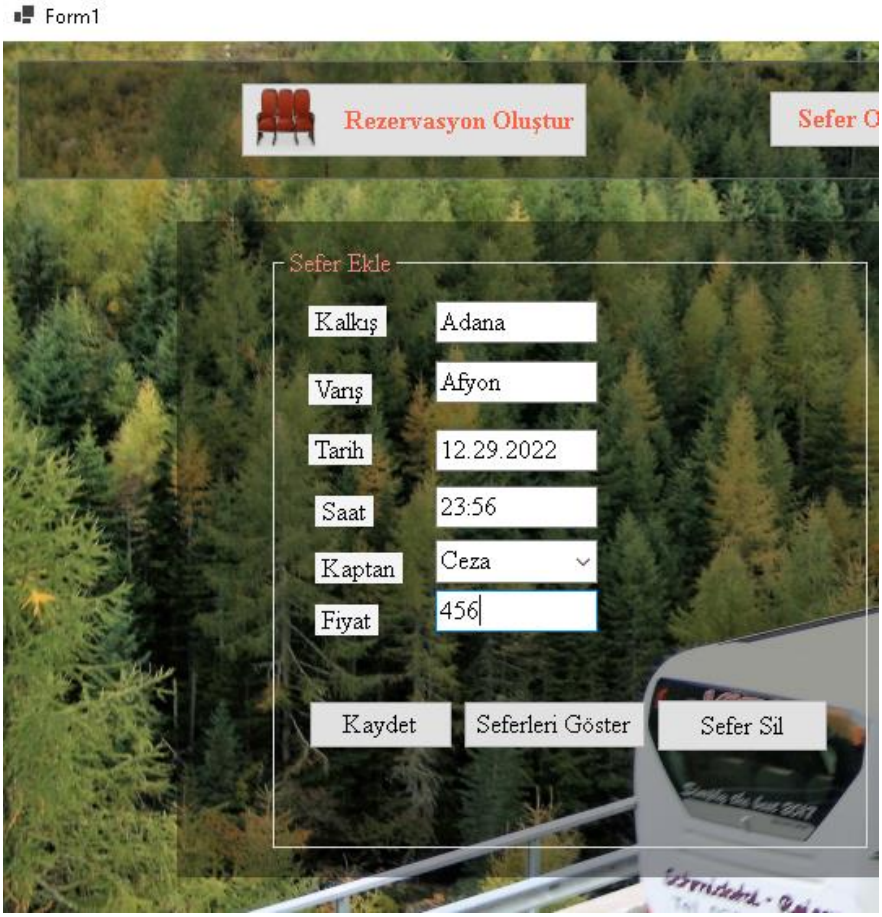

Figure 28: Adding A Bus Service

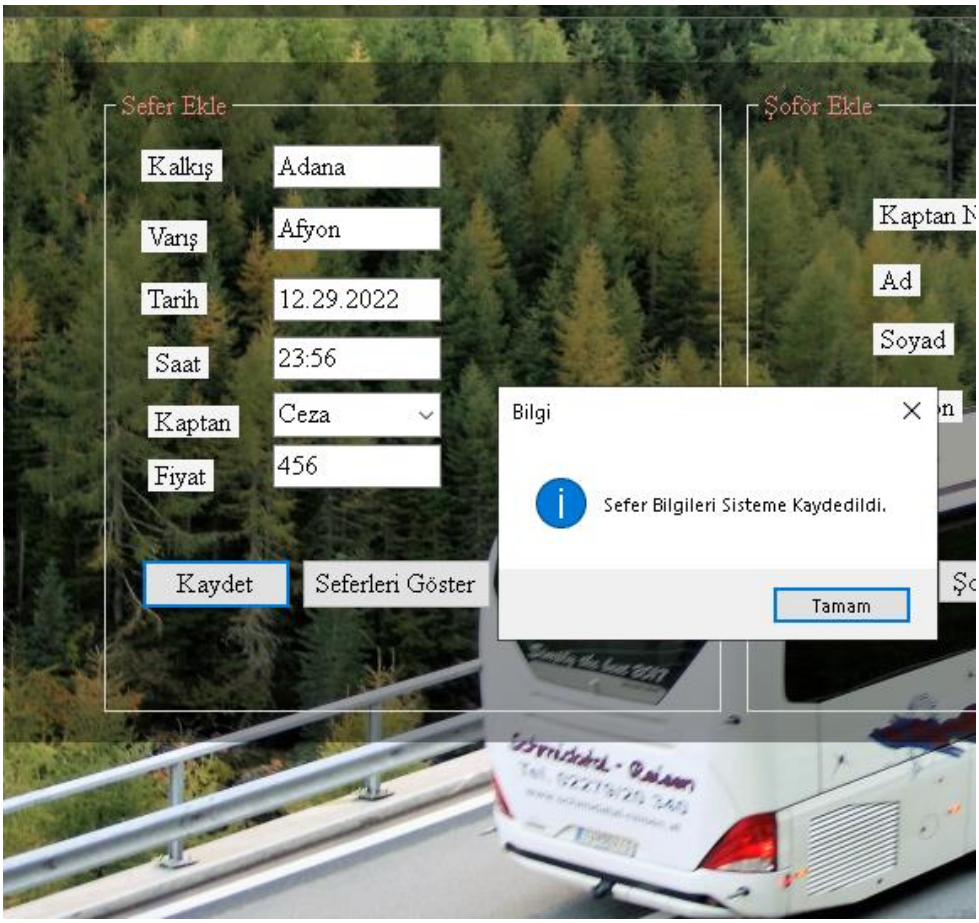

Figure 29: An Information Message About The Addition Of The Expedition

Figure 30: The New Bus Service Added Appears On The List

If the bus service is deleted, the time id is required. After entering the expedition ID and clicking the delete expedition button, if the operation has been successfully performed, an information message is sent by the system stating that it has been deleted. It is automatically deleted from the list of bus services shown to the user.

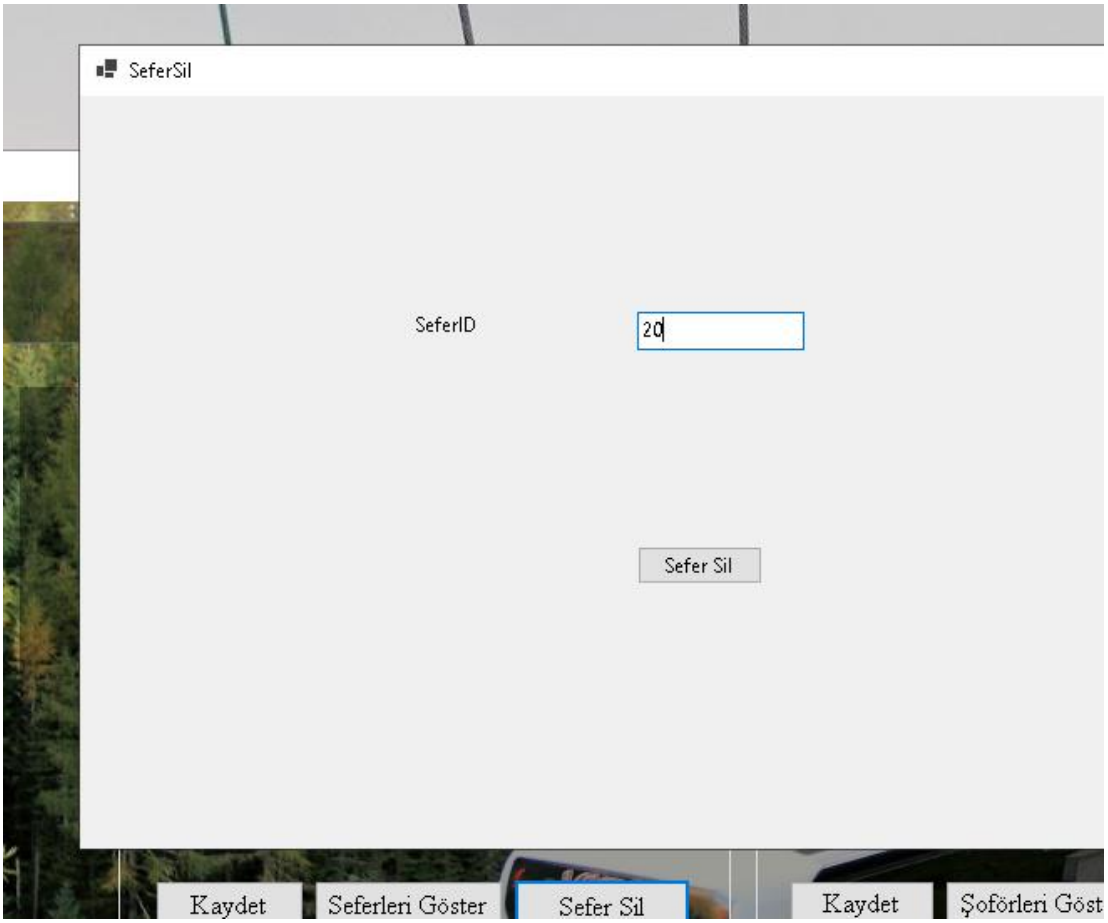

Figure 31: Deleting The Bus Service

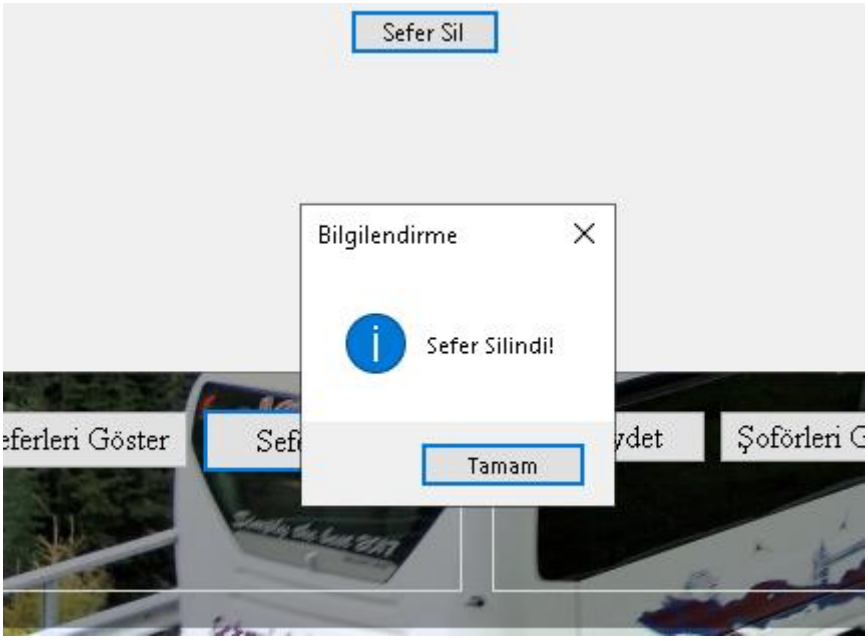

Figure 32: Notification Message That The Bus Service Has Been Deleted

Figure 33: When A Time Is Deleted In The Bus Service List

```
create  procedure KontrolSeferSil
@SeferNo int
AS
BEGIN
declare @sayacım int
        If EXISTS(select SeferNo  from seferBilgileri where SeferNo=@SeferNo)
```

```
        begin
                set @sayacım=1
        END
    else
        BEGIN
                SET  @sayacım=0
        END
    rETURN @sayacım
END
```

To add a driver, the data in the add driver section is entered completely and then adding is performed by pressing the save button. If the operation has been performed successfully, an information message is sent and the driver and information added to the list of drivers shown to the user are automatically added.

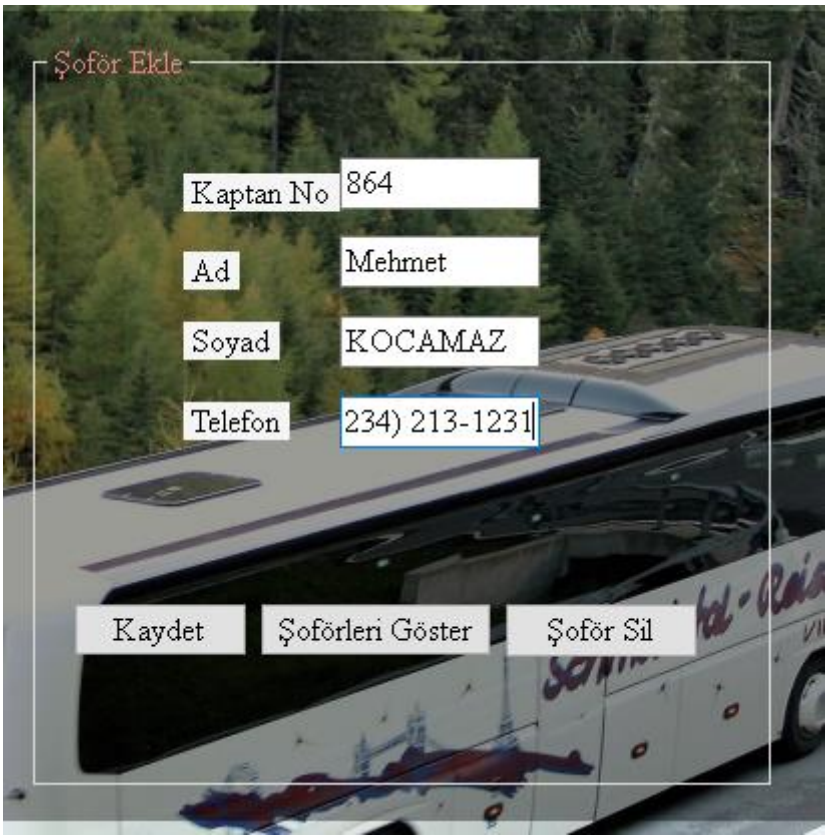

Figure 34: Adding A Driver

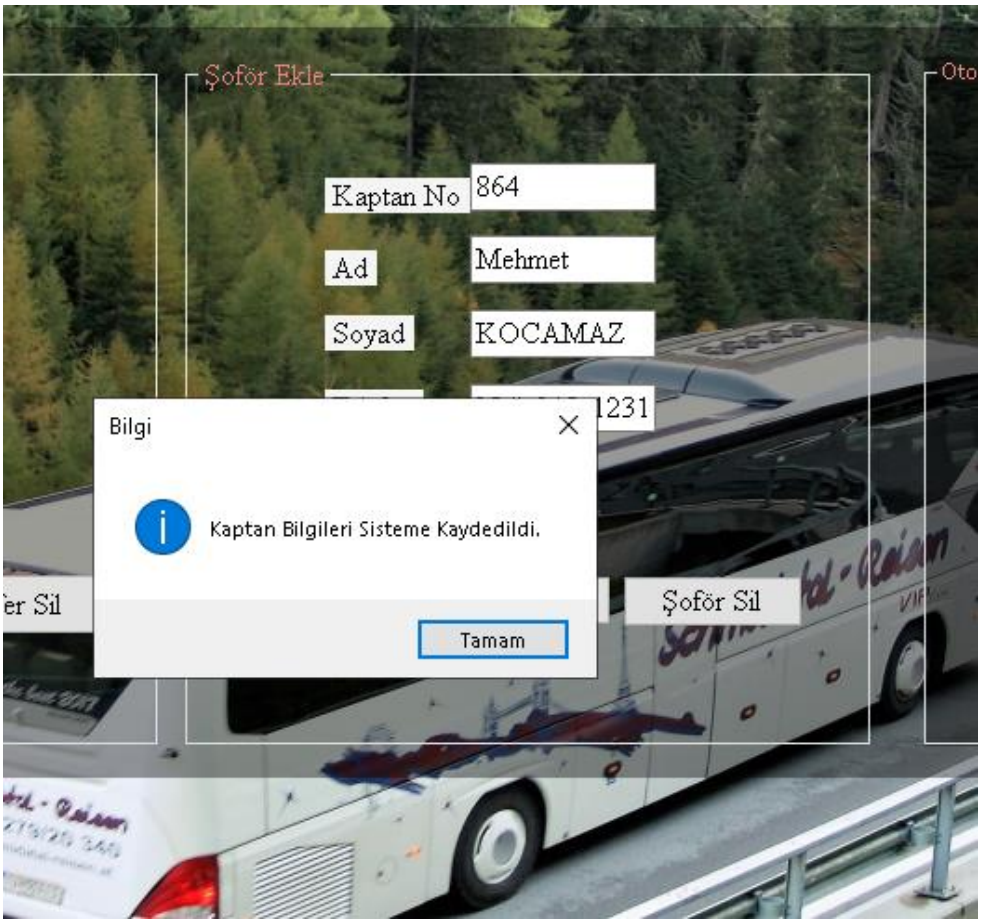

Figure 35: Information Message About The Addition Of A Driver

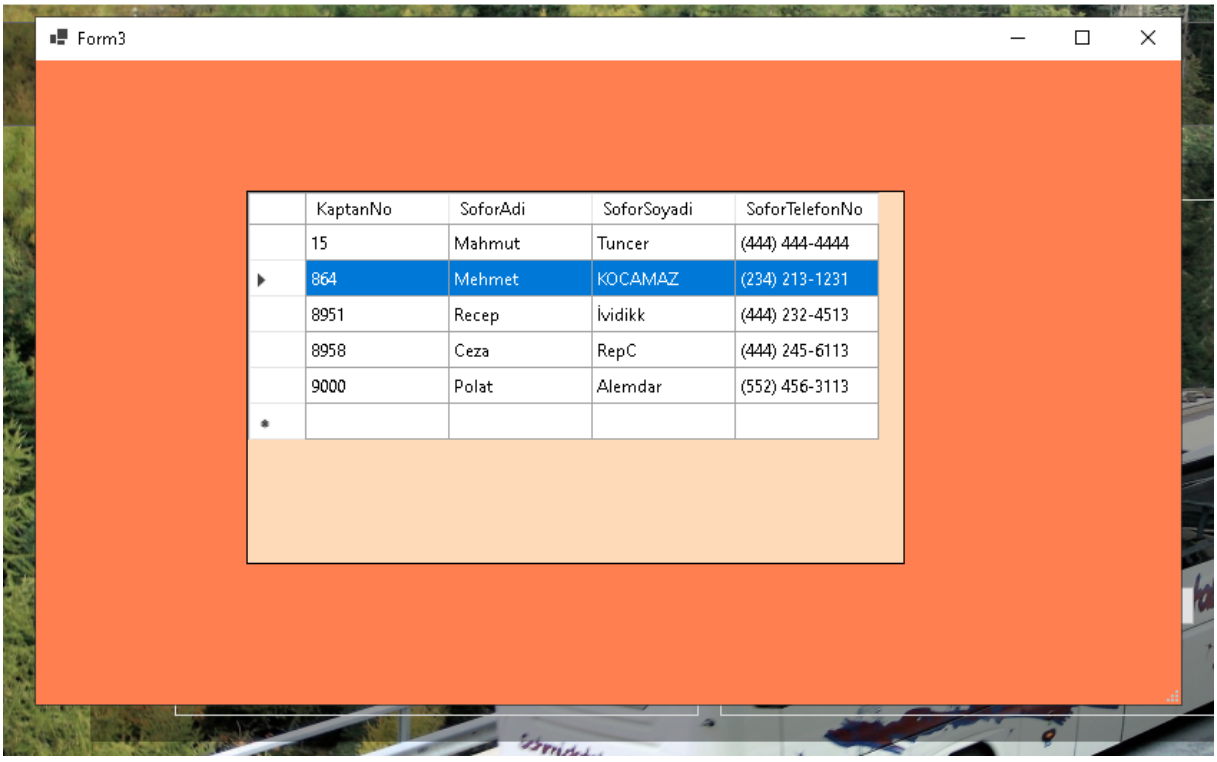

Figure 36: Showing The New Record Recorded In The List Where The Drivers Are Displayed

For the driver deletion process, the driver number is needed. If the operation is performed successfully, an information message is sent by the system and it is automatically deleted from the list of drivers shown to the user.

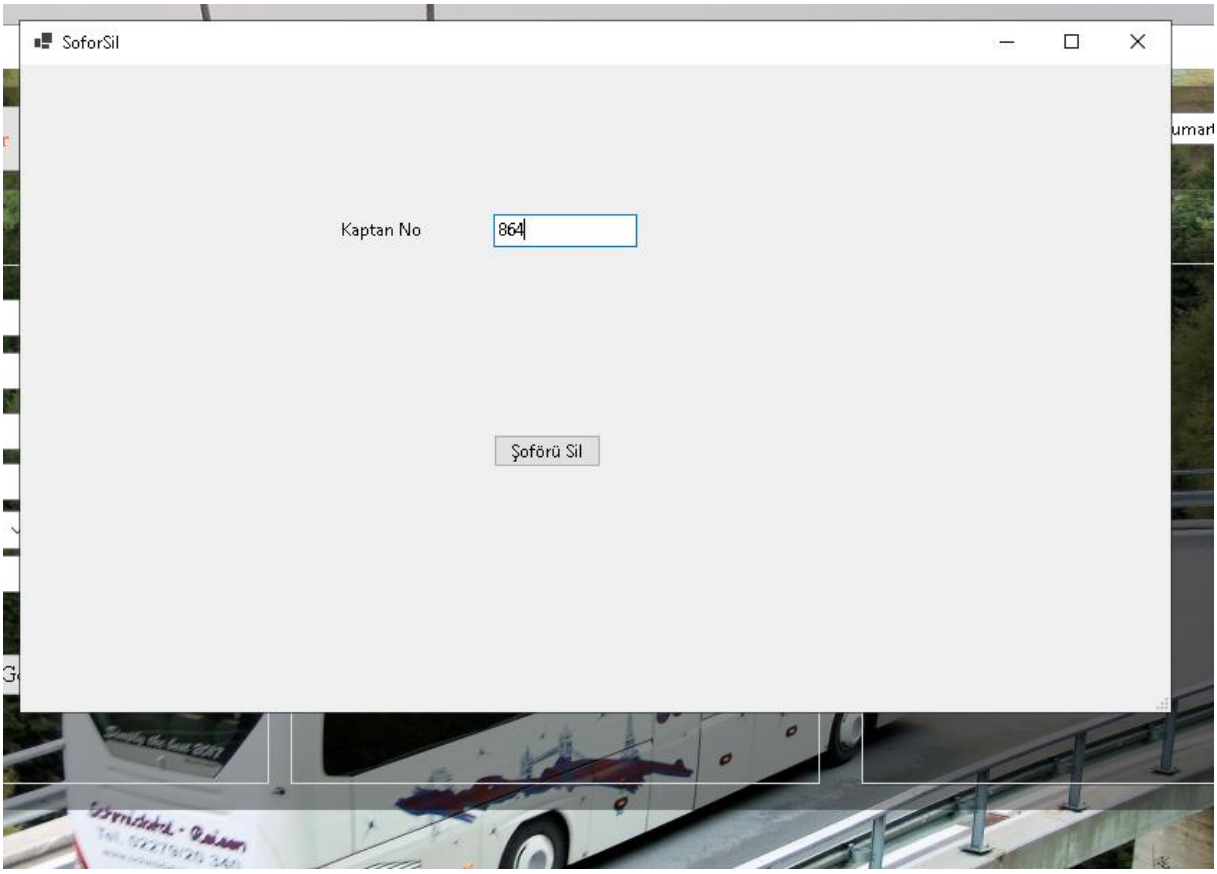

Figure 37: Deleting The Driver Registration

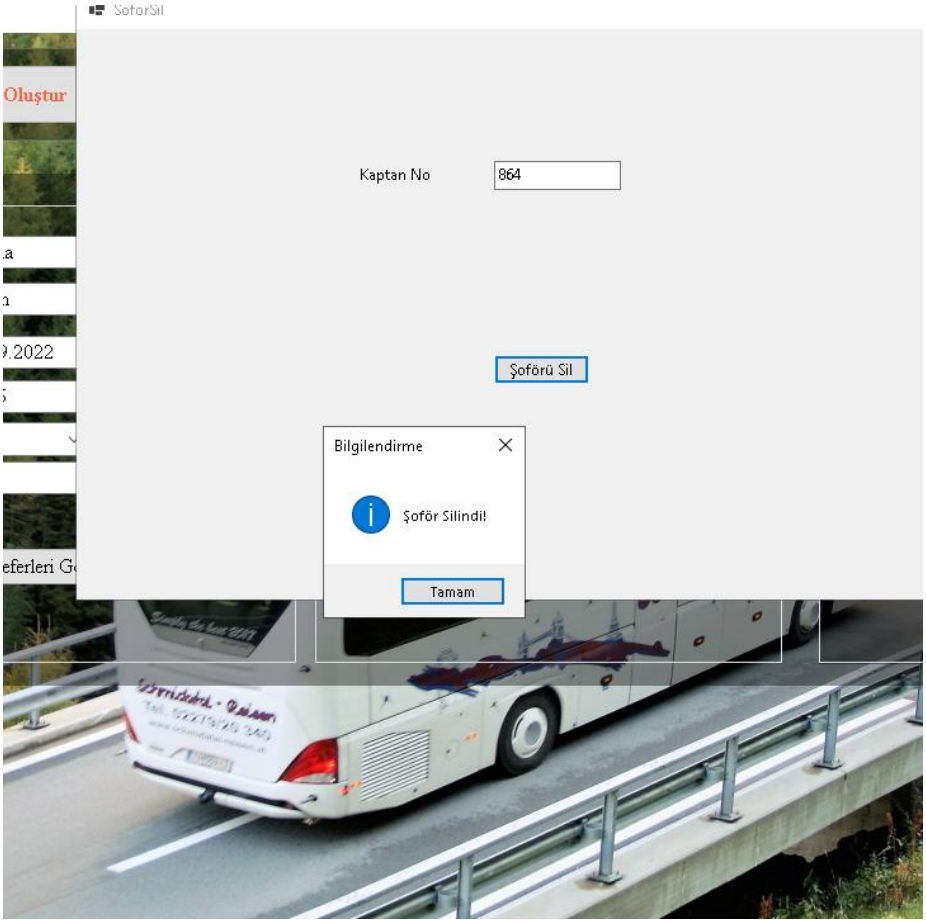

Figure 38: Notification Message That The Driver Has Been Deleted

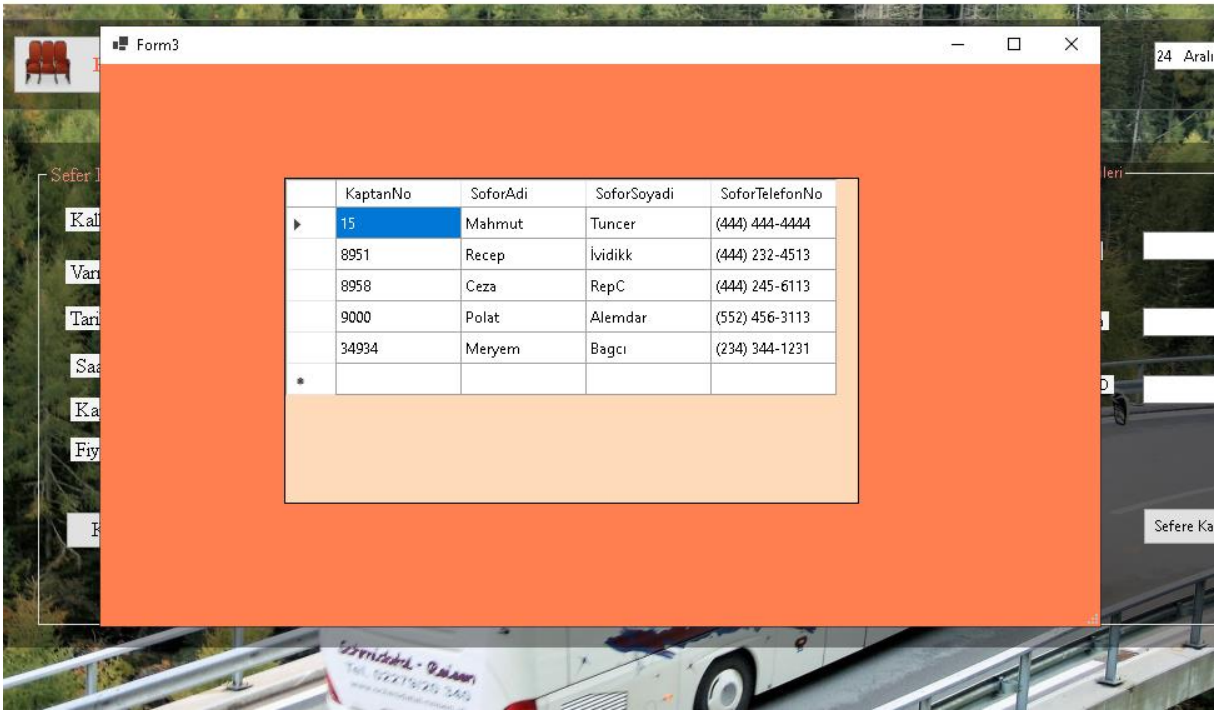

Figure 39: List View When The Driver Registration Is Deleted From The List

```
create  procedure KontrolSoforSil
@kaptanIsmi int
AS
BEGIN
declare @sayacımSofor int
        If EXISTS(select KaptanNo  from soforBilgileri where KaptanNo=@kaptanIsmi)
                begin
                        set @sayacımSofor=1
                END
        else
                BEGIN
                        SET  @sayacımSofor=0
                END
        RETURN @sayacımSofor
END
```

To add buses to the system, the information in the bus information panel is added completely and then it is realized by pressing the save next time button. If the operation has been successfully performed, the user will receive an information message about this.

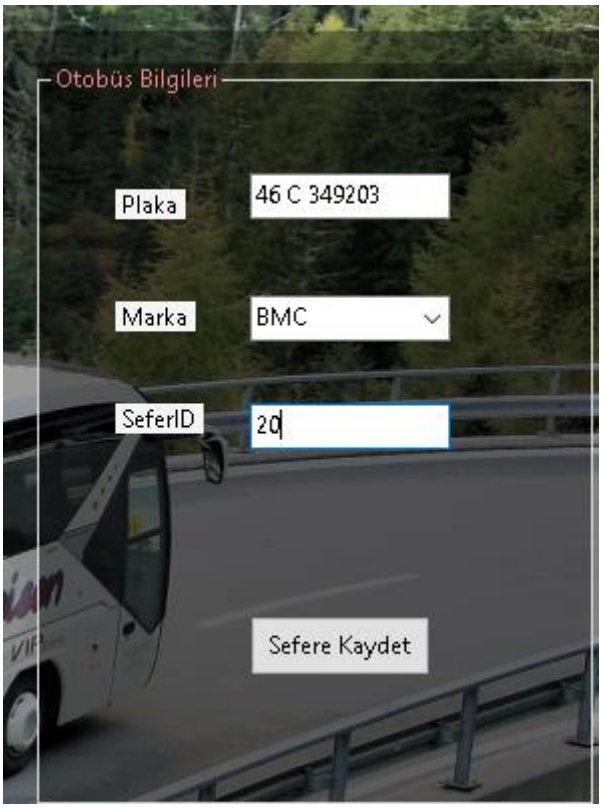

Figure 40: Adding A Bus

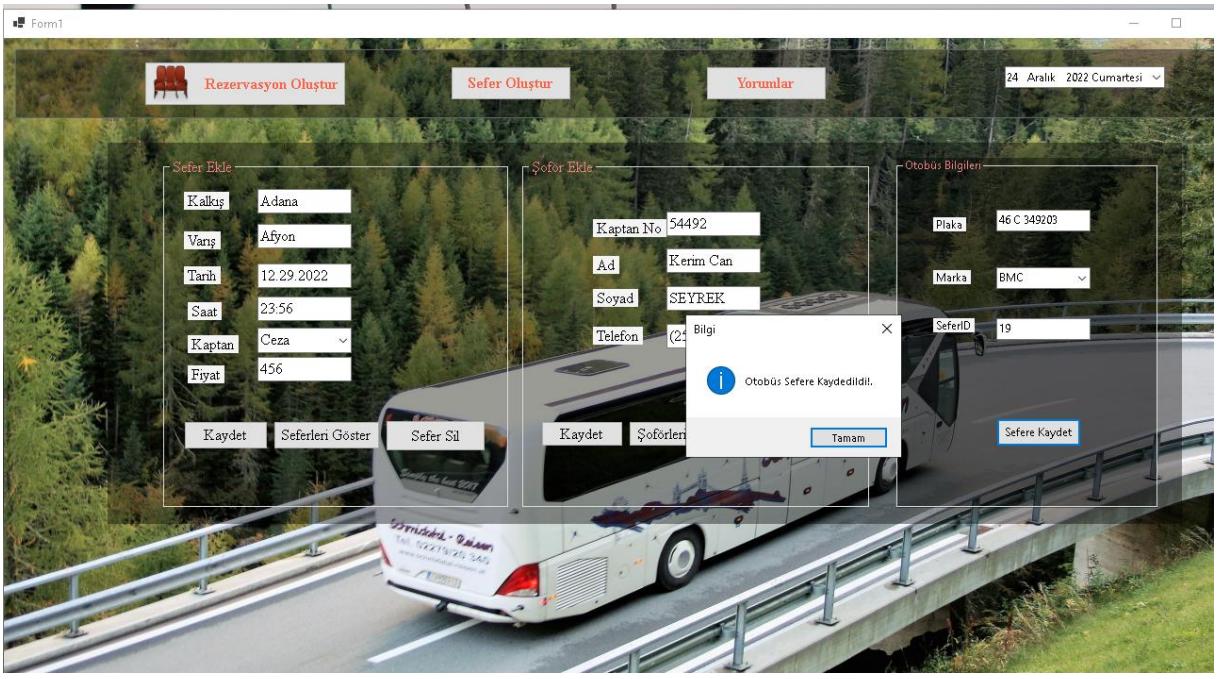

Figure 41: Notification Message That The Bus Has Been Registered

When the user wants to find one of the buses in the expeditions, an error message is sent by the system if there is no match with the data in the system.

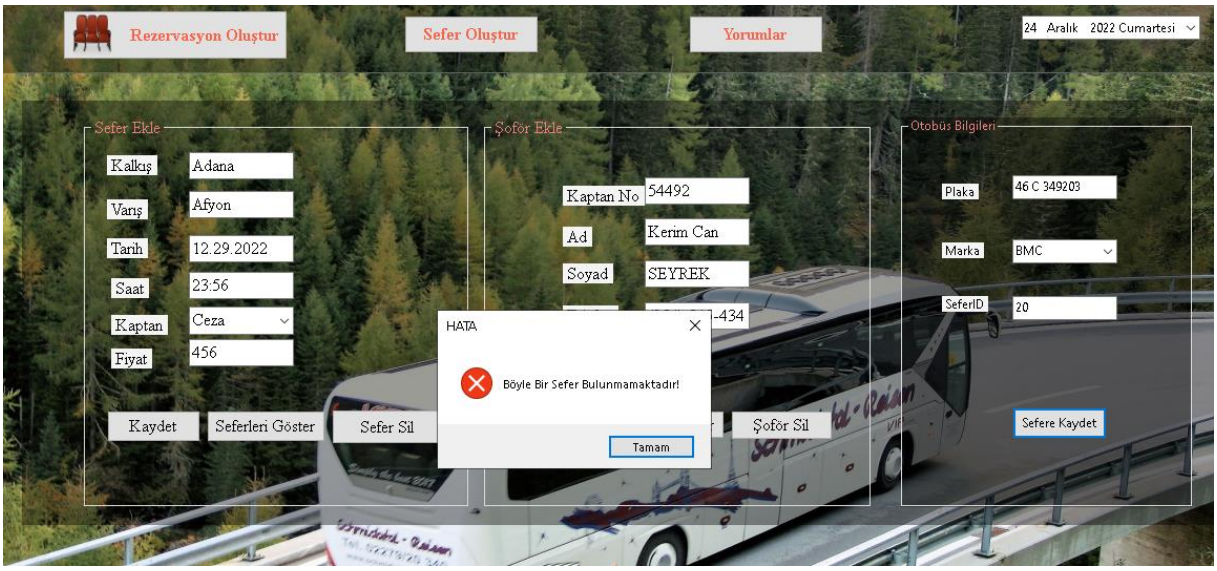

Figure 42: Error Message That The Expedition Was Not Found

```
create procedure SeferBul
        @kalkis nvarchar(30),
        @varis nvarchar(30),
        @tarih date
as
        select*from seferBilgileri where @kalkis=SeferKalkis and @varis=SeferVaris and @tarih=SeferTarih
```

In order to add a comment, the passenger ID and the bus service id must match. After the desired comment is written, it is saved by clicking on the save button. When the operation is successfully performed, an information message is sent by the system and it is automatically added to the list of comments made.

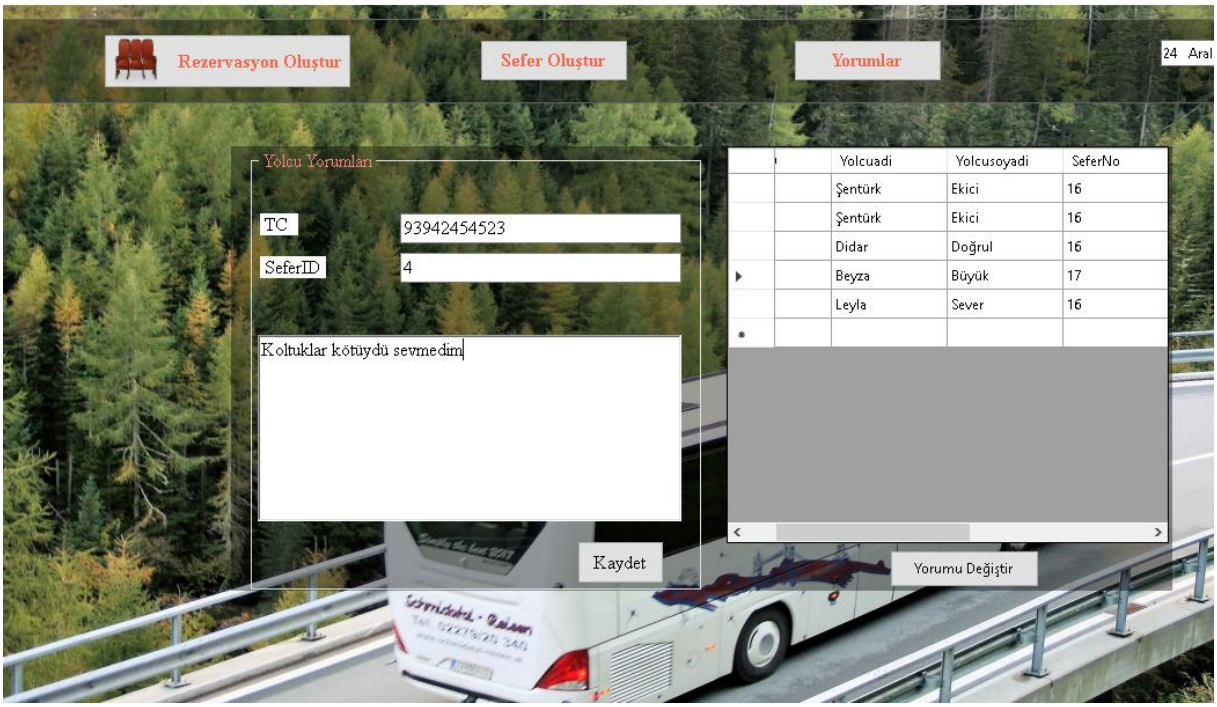

Figure 43: Adding A Comment

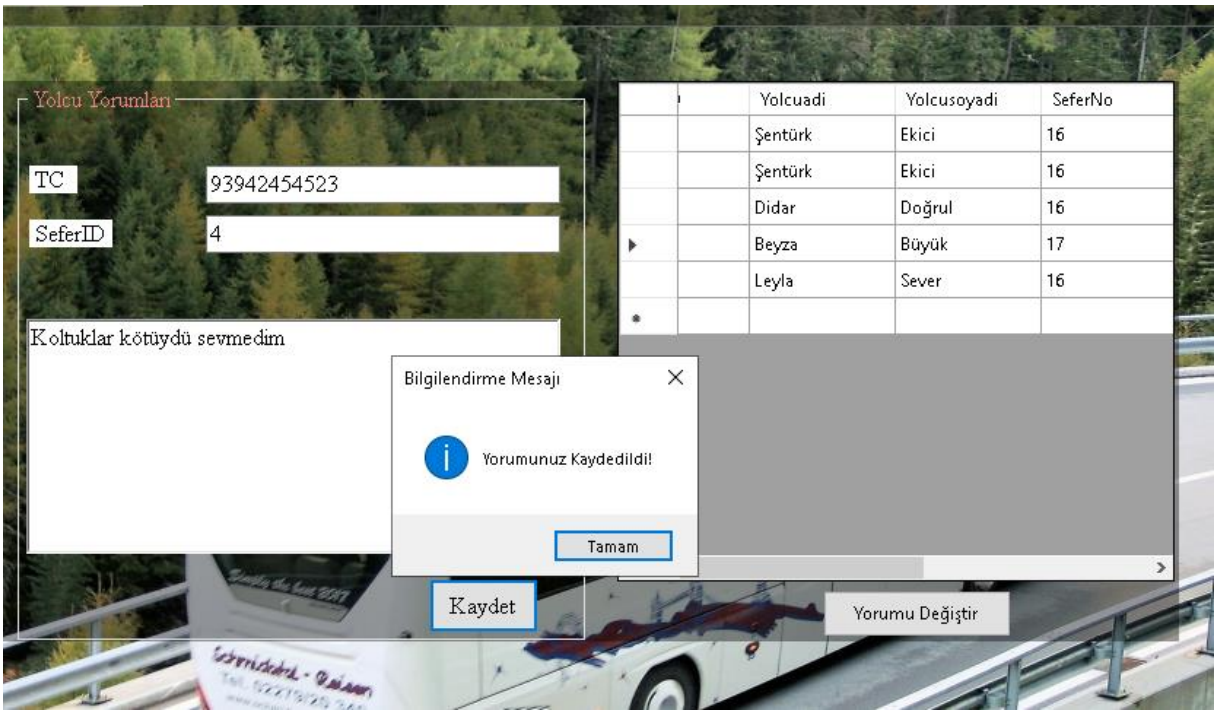

Figure 44: Information Message That The Comment Has Been Recorded

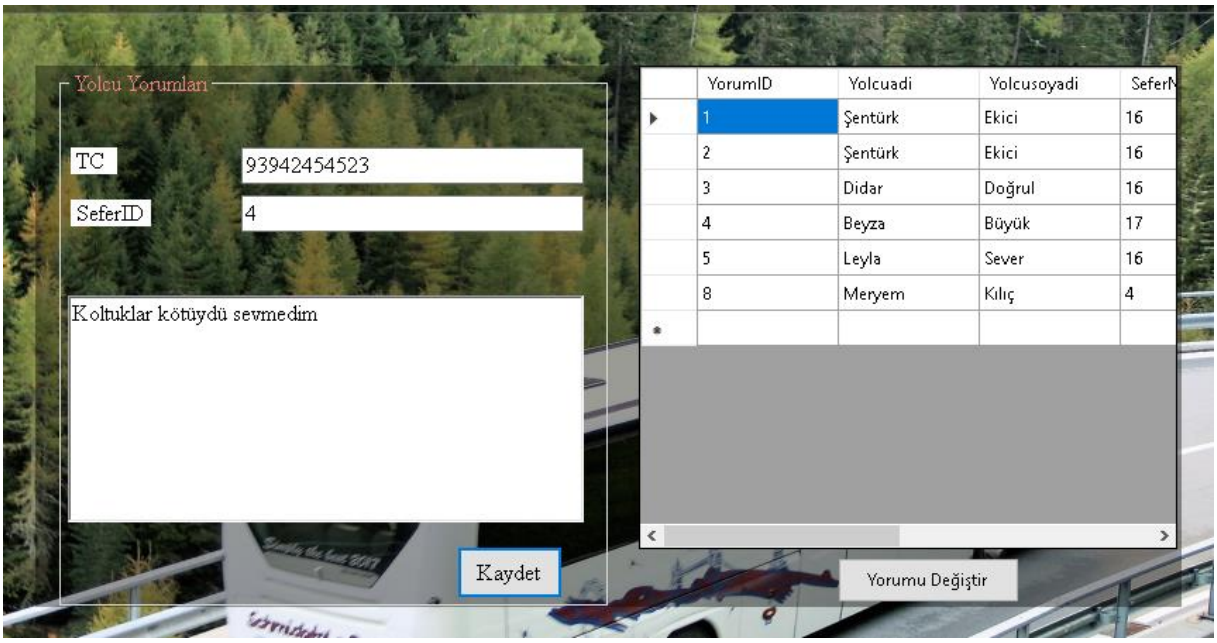

Figure 45: The Added Comment Appears In The Comments Section

```
create  procedure Kontrol
@TC nvarchar(11)
AS
BEGIN
declare @sayac int
        If EXISTS(select YolcuTcNo from yolcubilgileri where YolcuTcNo=@TC)
                begin
                                set @sayac=1
                END
        else
                BEGIN
                        SET  @sayac=0
                END
        RETURN @sayac
END
```

```
create view viewYolcuYorumları as
```

SELECT Yolcuadi,Yolcusoyadi,SeferNo,SeferTarih,SeferOtobusPlaka , yorum FROM yorumlar inner join yolcubilgileri on yorumlar.YolcuTc= yolcubilgileri.YolcuTcNo

inner join seferBilgileri on yorumlar.SeferID=seferBilgileri.SeferNo

Conclusion:

The project mainly includes:

- 8 Table
- 6 Relationships
- 2 Views
- 7 Stored Procedure

Also:

- 11 Select statement
- 6 Insert statement
- 3 Update statement
- 3 Delete Statement
- 3 Triggers