

Univerzitet u Novom Sadu
Fakultet tehničkih nauka
Inženjerstvo informacionih sistema

Aerodrom

Student: Luka Miljanović IT11/2022

Asistent: Dušan Krstić

Specifikacija projektne dokumentacije

1. Opis realnog sistema
2. Zadatak
3. Opis tehnologija
4. UML
 - 4.1. Dijagram slučajeva upotrebe
 - 4.2. Dijagram klasa
 - 4.3. Dijagram sekvenci
5. Baza podataka
 - 5.1. Kreiranje baze podataka
 - 5.2. Upiti i DML naredbe
6. Kreiranje WPF aplikacije
 - 6.1. Dodavanje
 - 6.2. Izmena
 - 6.3. Brisanje
 - 6.4. Izgled aplikacije
7. Testiranje aplikacije

8. Zaključak

1. Opis realnog sistema

Posmatrani sistem predstavlja aerodrom, mesto na kom se obavljaju operacije poletanja i sletanja aviona, utovar i istovar robe, ukrcavanje i iskrcavanje putnika, kontrola leta itd.

Za ovakav sistem je potrebno razviti aplikaciju koja će zaposlenima omogućiti da lakše vode evidenciju o dešavanjima na aerodromu kao i da bez velike muke unesu promene u sistem.

2. Zadatak

Naš zadatak se ogleda u tome da analiziramo učesnike u našem sistemu (aktere), da jasno definišemo njihove osobine, zatim je potrebno razmotriti događaje odnosno slučajeve upotrebe koji se dešavaju u našem sistemu i na koji način akteri učestvuju u njima.

Potrebno je definisati klase koje će se kasnije instancirati u vidu podataka u tabelama (table entry), njihove osobine će predstavljati kolone u tim tabelama i bitan je tip podataka koji će se koristiti.

Zatim analiziramo i pravimo dijagram sekvence odnosno redosled događaja u našem sistemu i kako klase (podaci iz tabele) interaktuju jedni sa drugima.

Sve ovo je praćeno detaljnim opisom i slikama.

Nakon kreiranih okvirnih imaginarnih slika i opisa, potrebno je ovaj (za sad) apstraktan sistem pretvoriti u nešto što zapravo postoji i funkcioniše.

To počinjemo tako što kreiramo bazu podataka i odgovarajuće tabele sa njihovim kolonama. Nakon kreiranja i testiranja baze prelazimo na drugi korak, a to je frontend interface i backend logic.

Nakon završetka front i backend-a mi imamo aplikaciju sa kojom možemo da pristupamo bazi podataka i da bez problema unosimo i brišemo podatke.

Izuzeci:

[Let nije internacionalni] Ukoliko let nije internacionalni odnosno putnik ne napušta državu, nije u dužnosti da prođe kroz graničnu kontrolu.

Uslovi koji moraju biti zadovoljeni nakon izvršavanja: Putnik dobija mogućnost da se ukrca na svoj let.

Slučaj upotrebe: Dolazak na aerodrom

Kratak opis: Dolazak putnika na aerodrom

Učesnici: Fizička lica.

Uslovi koji moraju biti zadovoljeni pre izvršavanja: /

Opis: Da bi putnik mogao da se ukrca na svoj let, mora da dođe na aerodrom sa kog taj let poleće

Izuzeci: /

Uslovi koji moraju biti zadovoljeni nakon izvršavanja: Putnik prolazi kroz graničnu kontrolu.

Slučaj upotrebe: Predaja / preuzimanje prtljaga

Kratak opis: Predaja i preuzimanje prtljaga

Učesnici: Magacioneri, Korisnici

Uslovi koji moraju biti zadovoljeni pre izvršavanja: Korisnik je prošao kontrolu.

Opis: Pre ulaska u avion korisnik mora da preda prtljag kako bi isti mogao da se ubaci u avion.

[Izuzetak: Korisnik nema prtljag].

Izuzeci:

[Korisnik nema prtljag] Ukoliko osoba nema prtljag, ulazi na let bez predaje istog.

Uslovi koji moraju biti zadovoljeni nakon izvršavanja: Putnik se ukrcava.

Slučaj upotrebe: Kupovina karte

Kratak opis: Kupovina karte za odgovarajući let

Učesnici: Avio Kompanija, Korisnici

Uslovi koji moraju biti zadovoljeni pre izvršavanja: Korisnik je uneo podatke.

Opis: Da bi korisnik ušao na let neophodno je da poseduje kartu.

Izuzeci: /

Uslovi koji moraju biti zadovoljeni nakon izvršavanja: Putnik je dobio kartu.

Slučaj upotrebe: Unos podataka

Kratak opis: Unos podataka neophodnih za kupovinu karte

Učesnici: Avio Kompanija, Korisnici

Uslovi koji moraju biti zadovoljeni pre izvršavanja: /

Opis: Da bi korisnik kupio kartu neophodno je da unese validne podatke.

Izuzeci: /

Uslovi koji moraju biti zadovoljeni nakon izvršavanja: Putnik može da kupi kartu.

Slučaj upotrebe: Provera dokumenata I karte

Kratak opis: Provera dokumenata I karte pre ukrcavanja na let

Učesnici: Zaposleni na aerodromu, Korisnici

Uslovi koji moraju biti zadovoljeni pre izvršavanja: Korisnik je kupio kartu

Opis: Da bi korisnik ušao na avion njegovi podaci moraju da se provere, kako u ličnim dokumentima tako I na karti.

Izuzeci: /

Uslovi koji moraju biti zadovoljeni nakon izvršavanja: Putnik može na proveru pasoša.

Slučaj upotrebe: Ukrcavanje / Iskrccavanje putnika

Kratak opis: Ukrcavanje I iskrccavanje putnika.

Učesnici: Zaposleni na aerodromu, Korisnici

Uslovi koji moraju biti zadovoljeni pre izvršavanja: Korisnik je prošao proveru pasoša

Opis: Korisnici se nakon predaje prtljaga (ili ne) I provere dokumenata ukrcavaju odnosno iskrccavaju sa aviona.

Izuzeci: /

Uslovi koji moraju biti zadovoljeni nakon izvršavanja: Putnik je ukrcan / iskrccan.

Slučaj upotrebe: Utovar / istovar robe

Kratak opis: Utovar i istovar robe

Učesnici: Zaposleni na aerodromu

Uslovi koji moraju biti zadovoljeni pre izvršavanja: Korisnik je predao prtljag / robu

Opis: Radnici na aerodromu unose, odnosno iznose robu i prtljag iz trupa aviona.

Izuzeci: /

Uslovi koji moraju biti zadovoljeni nakon izvršavanja: Roba / prtljag je unesen.

Slučaj upotrebe: Parkiranje na odgovarajući gejt

Kratak opis: Parkiranje aviona na odgovarajući gejt

Učesnici: Aviokompanija, Zaposleni na aerodromu

Uslovi koji moraju biti zadovoljeni pre izvršavanja: /

Opis: Piloti zajedno sa osobljem na tlu parkiraju avion na odgovarajući gejt kako bi omogućili ukrcavanje putnika.

Izuzeci: /

Uslovi koji moraju biti zadovoljeni nakon izvršavanja: Avion je parkiran.

Slučaj upotrebe: Otvaranje / zatvaranje gejta

Kratak opis: Otvaranje i zatvaranje gejta

Učesnici: Zaposleni na aerodromu

Uslovi koji moraju biti zadovoljeni pre izvršavanja: Avion je parkiran

Opis: Radnici na aerodromu nakon provere dokumenata i parkiranja aviona na gejt isti mogu da otvore kako bi se putnici ukrcali.

Izuzeci: /

Uslovi koji moraju biti zadovoljeni nakon izvršavanja: Gejt je otvoren.

Slučaj upotrebe: Sipanje goriva

Kratak opis: Sipanje goriva neophodnog za let

Učesnici: Zaposleni na aerodromu, Aviokompanija

Uslovi koji moraju biti zadovoljeni pre izvršavanja: Avion je parkiran

Opis: Osoblje na tlu izvršava sipanje goriva u avion po dokumentima koje je izdala aviokompanija i po dijagramima rasporeda goriva u trupu aviona.

Izuzeci: /

Uslovi koji moraju biti zadovoljeni nakon izvršavanja: Rezervoari aviona su napunjeni.

Slučaj upotrebe: Poletanje / sletanje aviona

Kratak opis: Poletanje / sletanje aviona

Učesnici: Zaposleni na aerodromu, piloti

Uslovi koji moraju biti zadovoljeni pre izvršavanja: Odrađene su provere pre leta

Opis: Nakon provera pre leta, avion izlazi na pistu i uz odgovarajuće dozvole poleće sa iste, nakon leta na drugom (ili istom) aerodromu avion sleće i silazi sa piste.

Izuzeci: /

Uslovi koji moraju biti zadovoljeni nakon izvršavanja: Avion je u letu / na zemlji.

Slučaj upotrebe: Kontrola vazdušnog prostora

Kratak opis: Kontrola aviona u prolazu i letova na sletanju / poletanju

Učesnici: Zaposleni na aerodromu, piloti

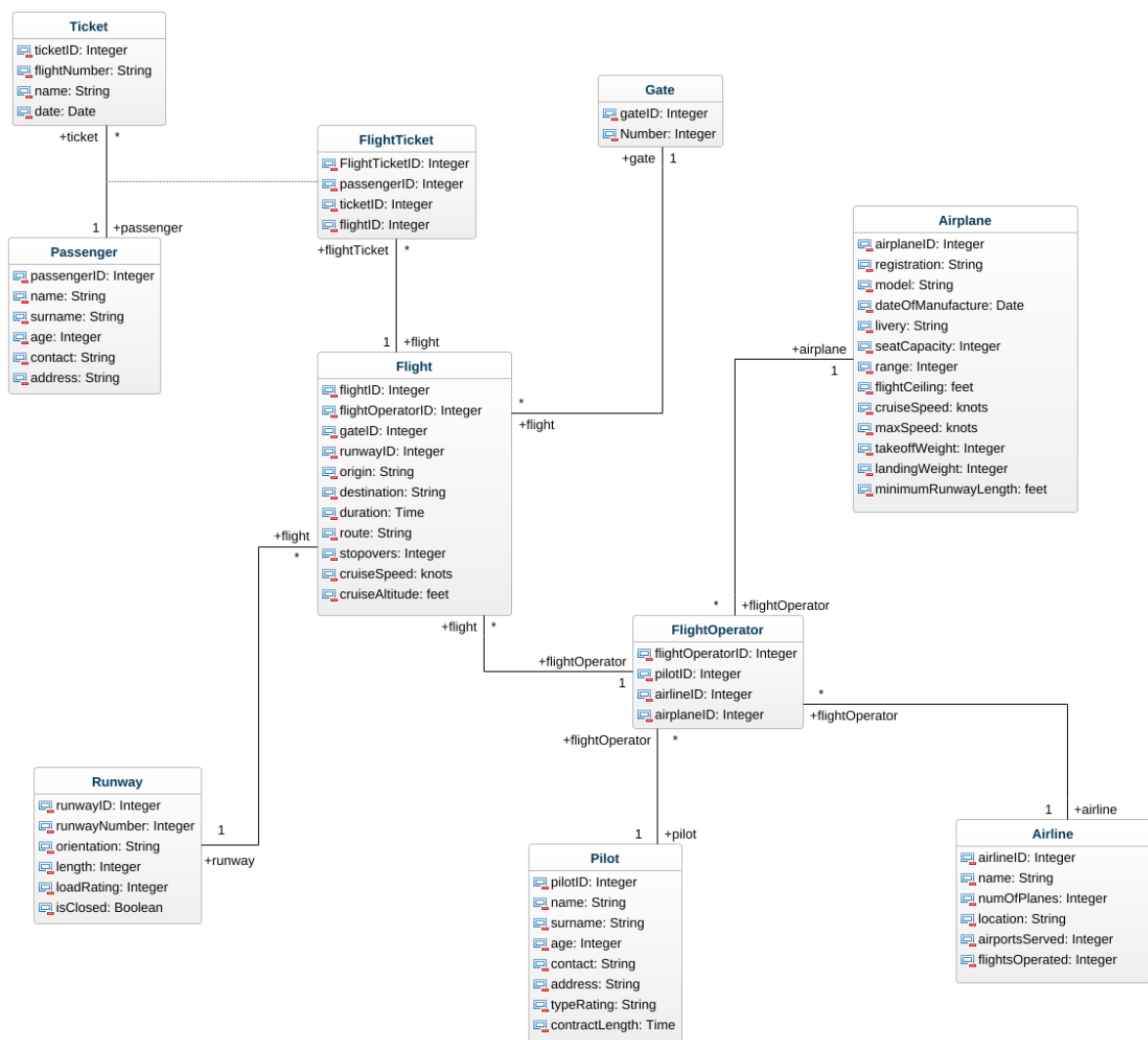
Uslovi koji moraju biti zadovoljeni pre izvršavanja: Avion je na istoj radio frekvenciji kao i kontroleri

Opis: Vazdušni prostor je neophodno kontrolisati i zato su kontroleri i piloti na istoj frekvenciji i razmenjuju informacije.

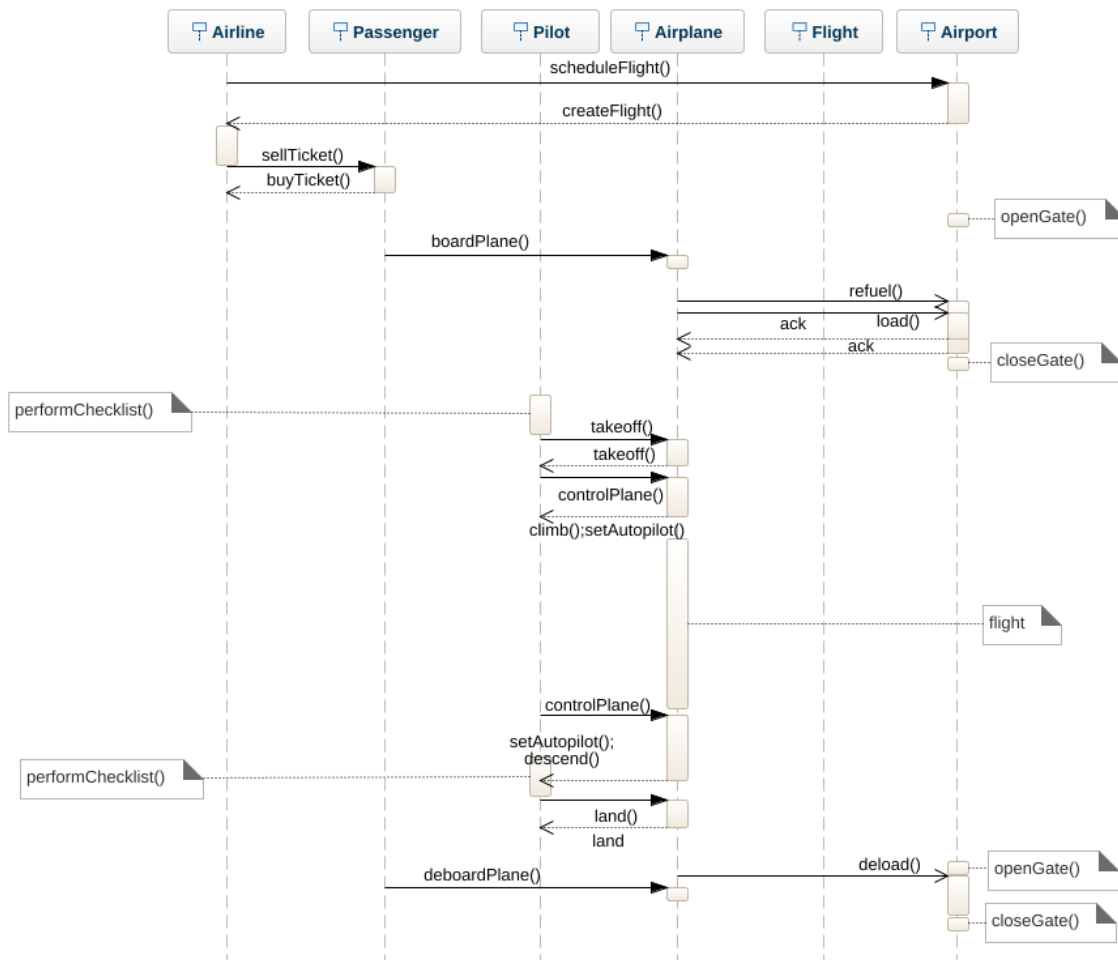
Izuzeci: /

Uslovi koji moraju biti zadovoljeni nakon izvršavanja: Piloti / Kontroleri reaguju na dobijene informacije.

4.2. Diagrama klasa



4.3. Dijagram sekvenci

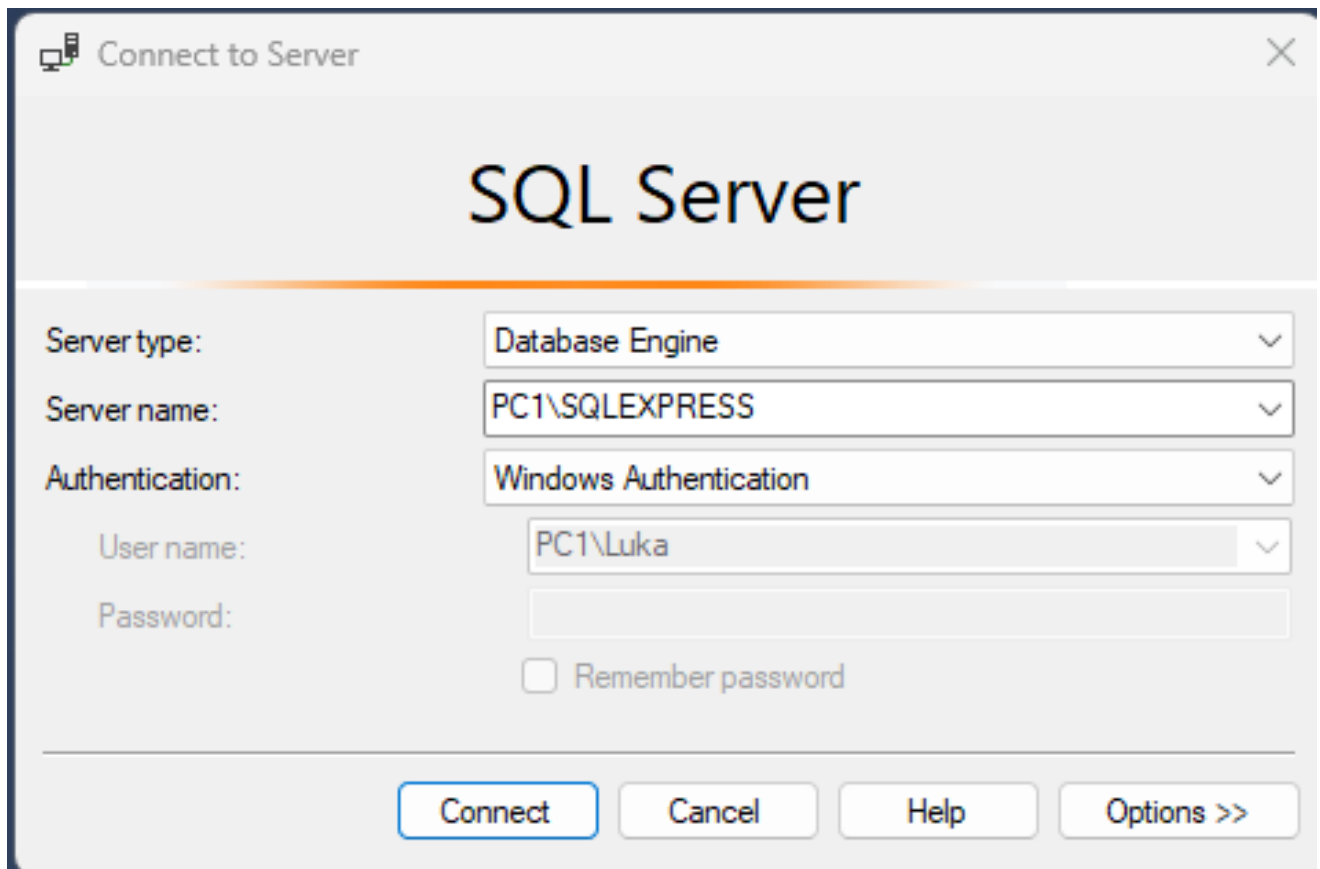


5. Baze podataka

5.1. Kreiranje baze podataka

Da bi smo kreirali bazu podataka neophodan nam je alat za takvu operaciju, mi ćemo u ovom projektu koristiti Microsoft SQL Server Management Studio, koji upravlja pozadinskim SQL Server-om 2022. Listu servera kao i trenutnu konfiguraciju možemo videti u alatu SQL Server 2022 Configuration Manager.

Kada upalimo Microsoft SQL Management Studio dočekani smo sa prozorom koji od nas očekuje podatke neophodne za povezivanje sa nekim serverom.



The screenshot shows the 'Connect to Server' dialog box in Microsoft SQL Server Management Studio. The dialog has a title bar with a server icon and the text 'Connect to Server'. The main content area has 'SQL Server' in large text. Below this, there are several input fields and a checkbox:

- Server type:** A dropdown menu showing 'Database Engine'.
- Server name:** A dropdown menu showing 'PC1\SQLEXPRESS'.
- Authentication:** A dropdown menu showing 'Windows Authentication'.
- User name:** A dropdown menu showing 'PC1\Luka'.
- Password:** A text input field.
- Remember password:** An unchecked checkbox.

At the bottom of the dialog, there are four buttons: 'Connect', 'Cancel', 'Help', and 'Options >>'.

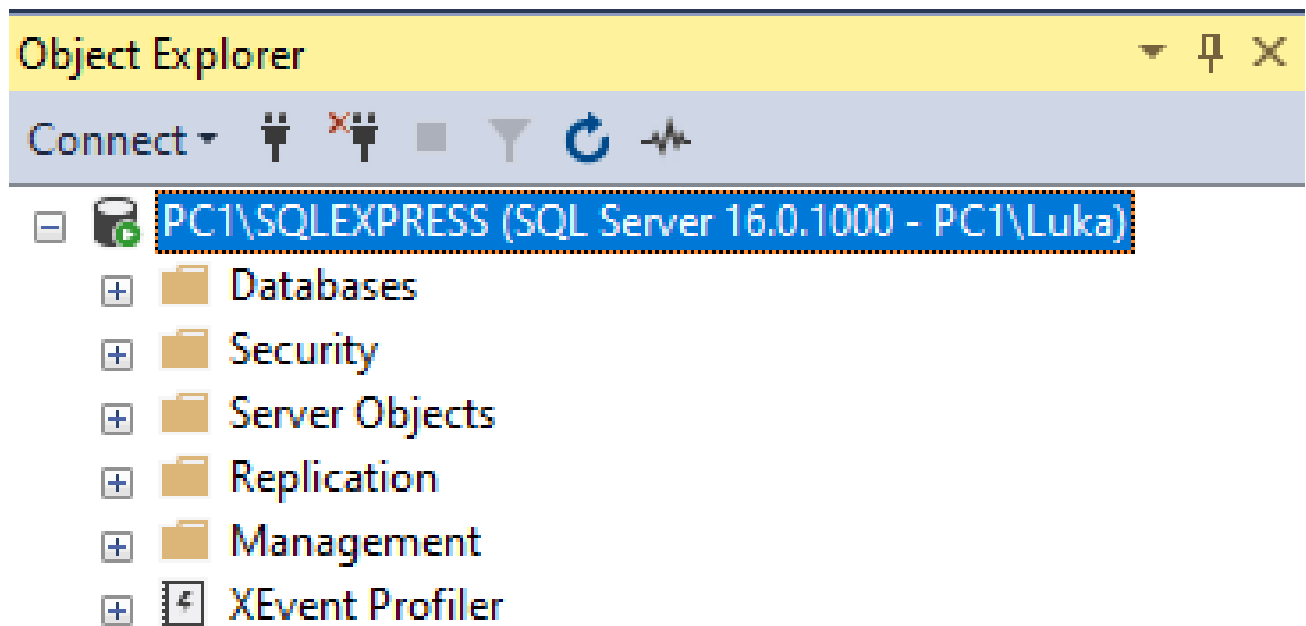
U našem slučaju je to lokalna mašina (dodatno se SQL Server i Management Studio nalaze na istom računaru), pa su nam parametri sledeći:

- Server type: Database Engine
- Server name: PC1\SQLEXPRESS [servername]\[sqlserver]
- Authenticaion: Windows Authentication, odnosno na koji način će nam se proveravati ovlašćenja i prava prilikom upravljanja bazom podataka.

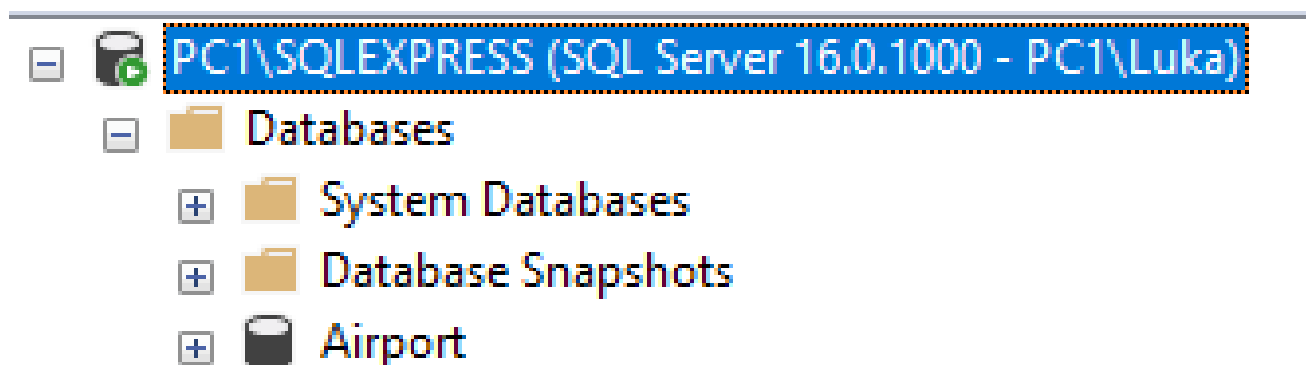
Nakon unesenih podataka klikenmo Connect.

Nakon nekog vremena će nam se sa leve strane pojaviti hijerarhija foldera, podataka i tabela.

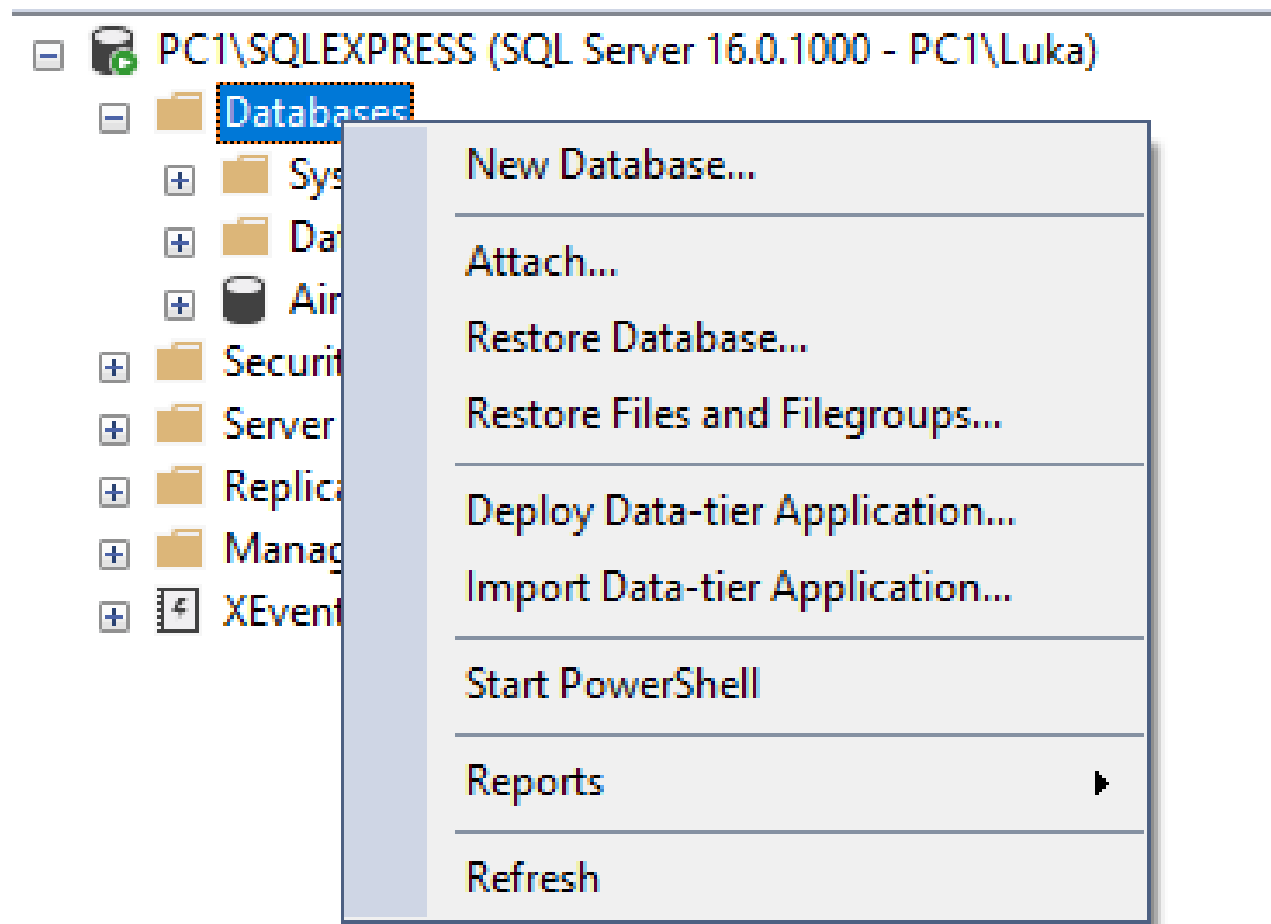
Nas interesuje “Databases”



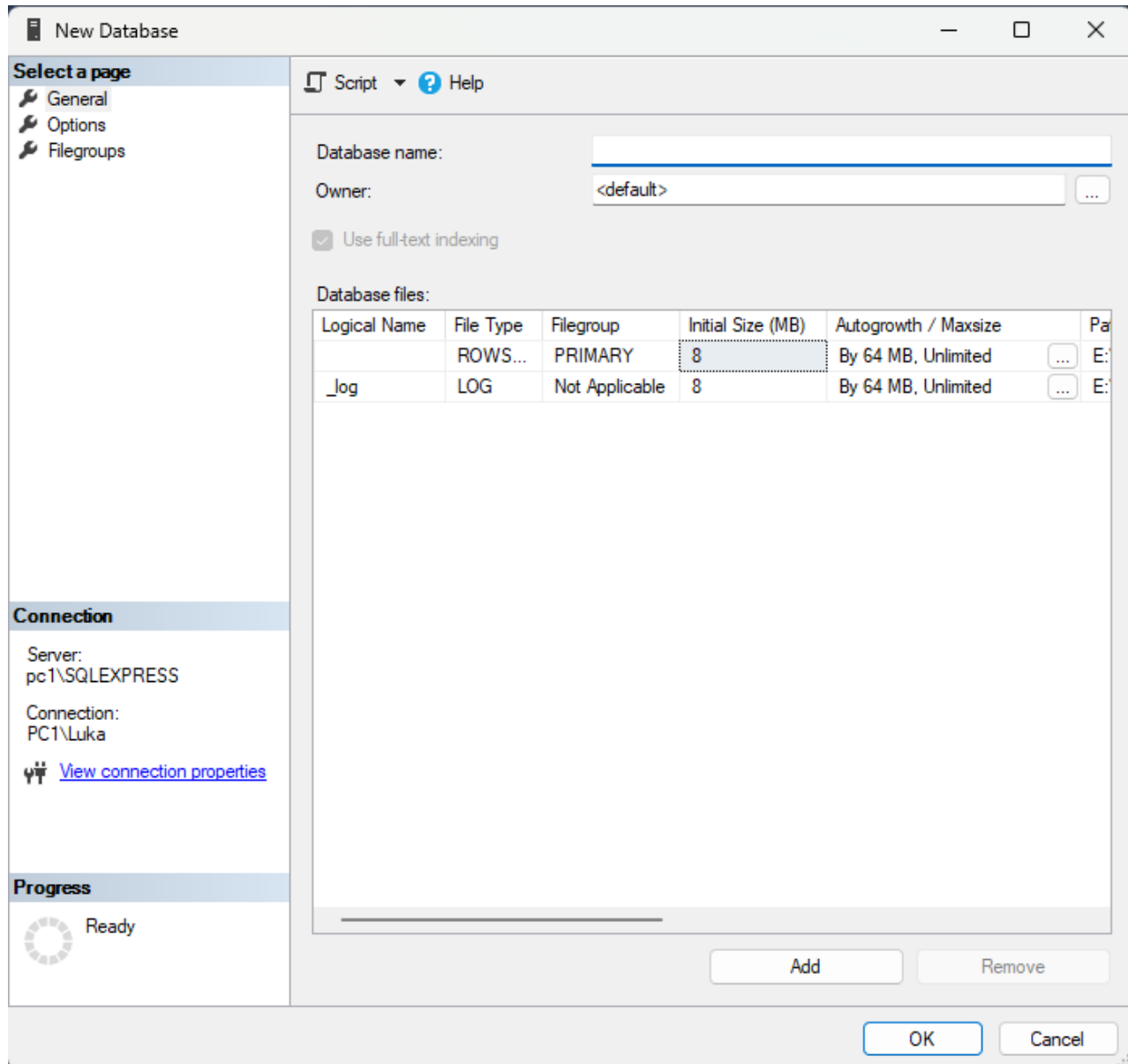
Klikom na “+” proširujemo hijerarhiju i vidimo baze koje postoje na našem serveru.



U slučaju novog servera, videćemo samo “System Databases”, i “Database Snapshots”. Desnim klikom na “Databases” dobijamo opciju da napravimo novu bazu podataka.

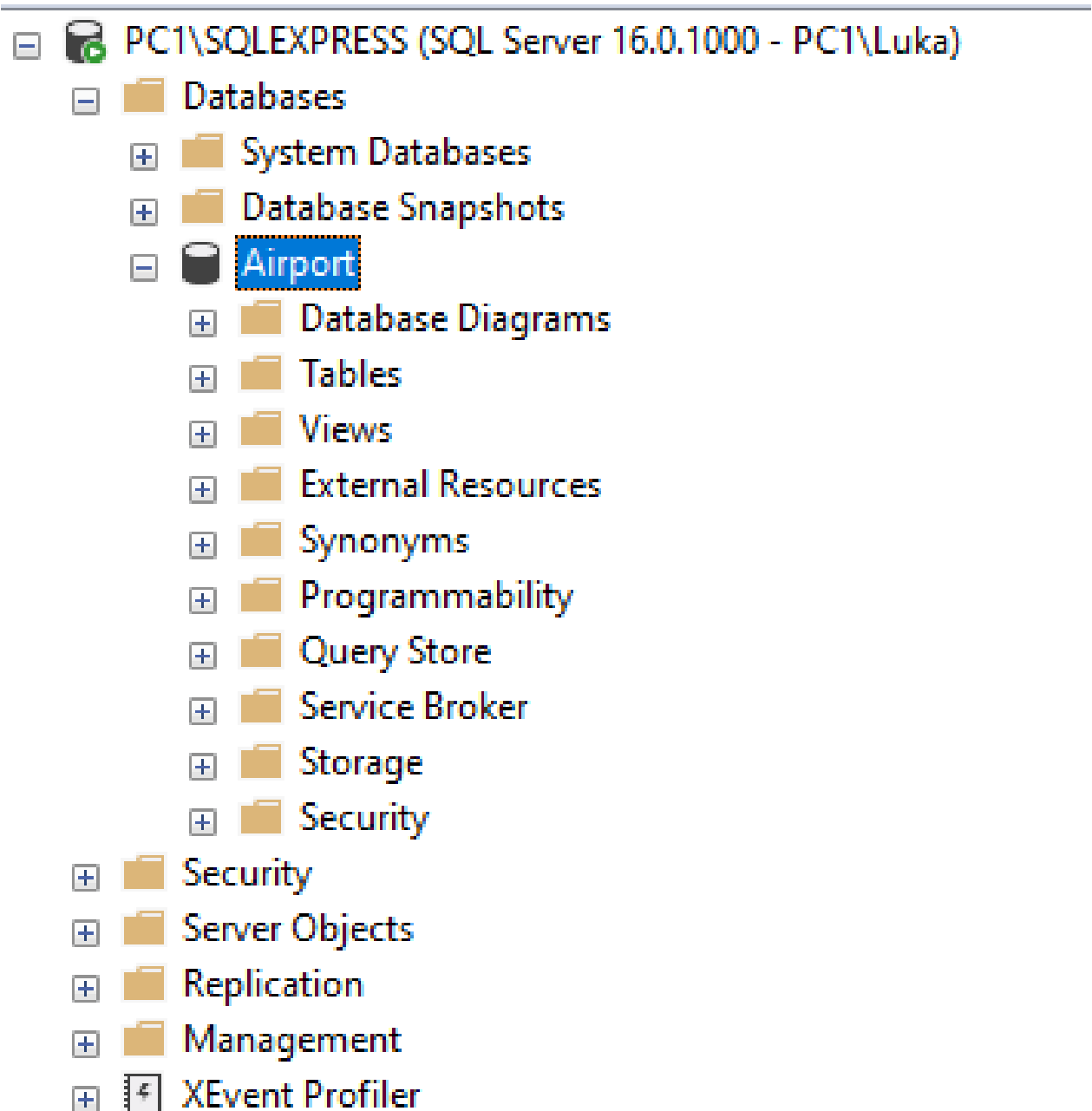


Klikom na “New Database” otvara nam se novi prozor:





































U ovom prozoru nećemo ništa menjati iako imamo opciju za tako nešto. Nas će u ovom slučaju samo interesovati ime baze, a to će biti “Airport”. Nakon unesenog imena i klika na “OK”, naša baza podataka će biti kreirana i neće imati nijednu tabelu.

Kada nam se u *Object Explorer*-u pojavi baza možemo kliknuti na “+” kako bi proširili odnosno prikazali hijerarhiju.

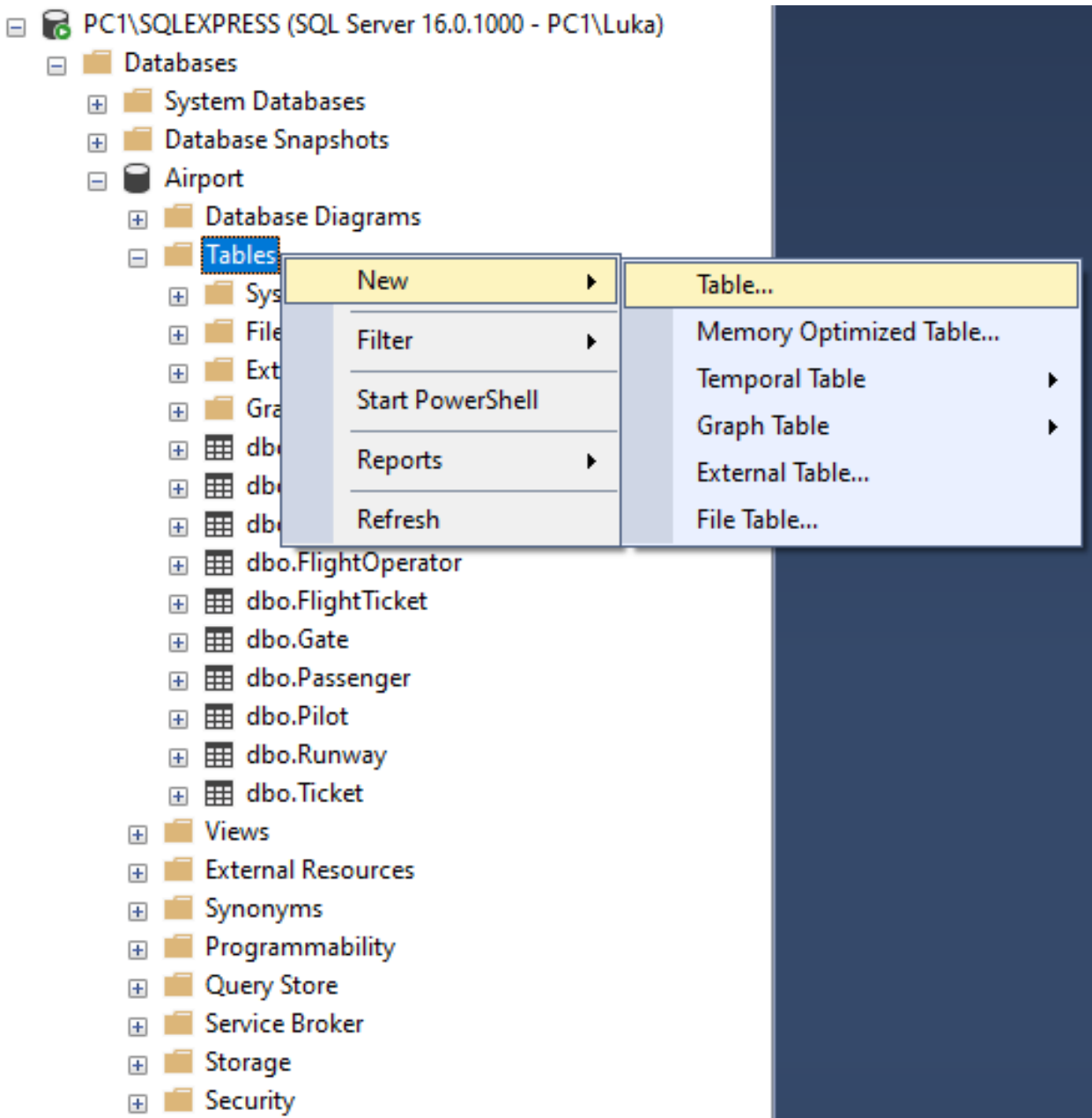


Ovde imamo priliku da vidimo dosta *foldera* koji obslužuju našu bazu, nas interesuje “Tables” kako će nam se tu nalaziti tabele sa podacima. Daljim klikom na “+” na opciji “Tables”

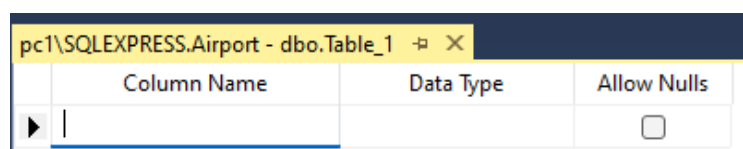
prikazujemo hijerarhiju i vidimo “System Tables”, “FileTables”, “External Tables”, i “Graph Tables”, nakon kreiranja nove baze podataka ovde nećemo videti nijednu tabelu.

- [-]  PC1\SQLEXPRESS (SQL Server 16.0.1000 - PC1\Luka)
 - [-]  Databases
 - [+]  System Databases
 - [+]  Database Snapshots
 - [-]  Airport
 - [+]  Database Diagrams
 - [-]  Tables
 - [+]  System Tables
 - [+]  FileTables
 - [+]  External Tables
 - [+]  Graph Tables
 - [+]  dbo.Airline
 - [+]  dbo.Airplane
 - [+]  dbo.Flight
 - [+]  dbo.FlightOperator
 - [+]  dbo.FlightTicket
 - [+]  dbo.Gate
 - [+]  dbo.Passenger
 - [+]  dbo.Pilot
 - [+]  dbo.Runway
 - [+]  dbo.Ticket
 - [+]  Views
 - [+]  External Resources
 - [+]  Synonyms
 - [+]  Programmability
 - [+]  Query Store
 - [+]  Service Broker
 - [+]  Storage
 - [+]  Security
 - [+]  Security
 - [+]  Server Objects
 - [+]  Replication
 - [+]  Management
 - [+]  XEvent Profiler

Desnim klikom na "Tables" dobijamo opciju da napravimo tabelu u našoj bazi podataka.




Otvora nam se nova prazna tabela u kojoj možemo da počnemo da zadajemo kolone.



Kao primer ćemo uzeti tabelu *dbo.Airline* u nju unosimo:

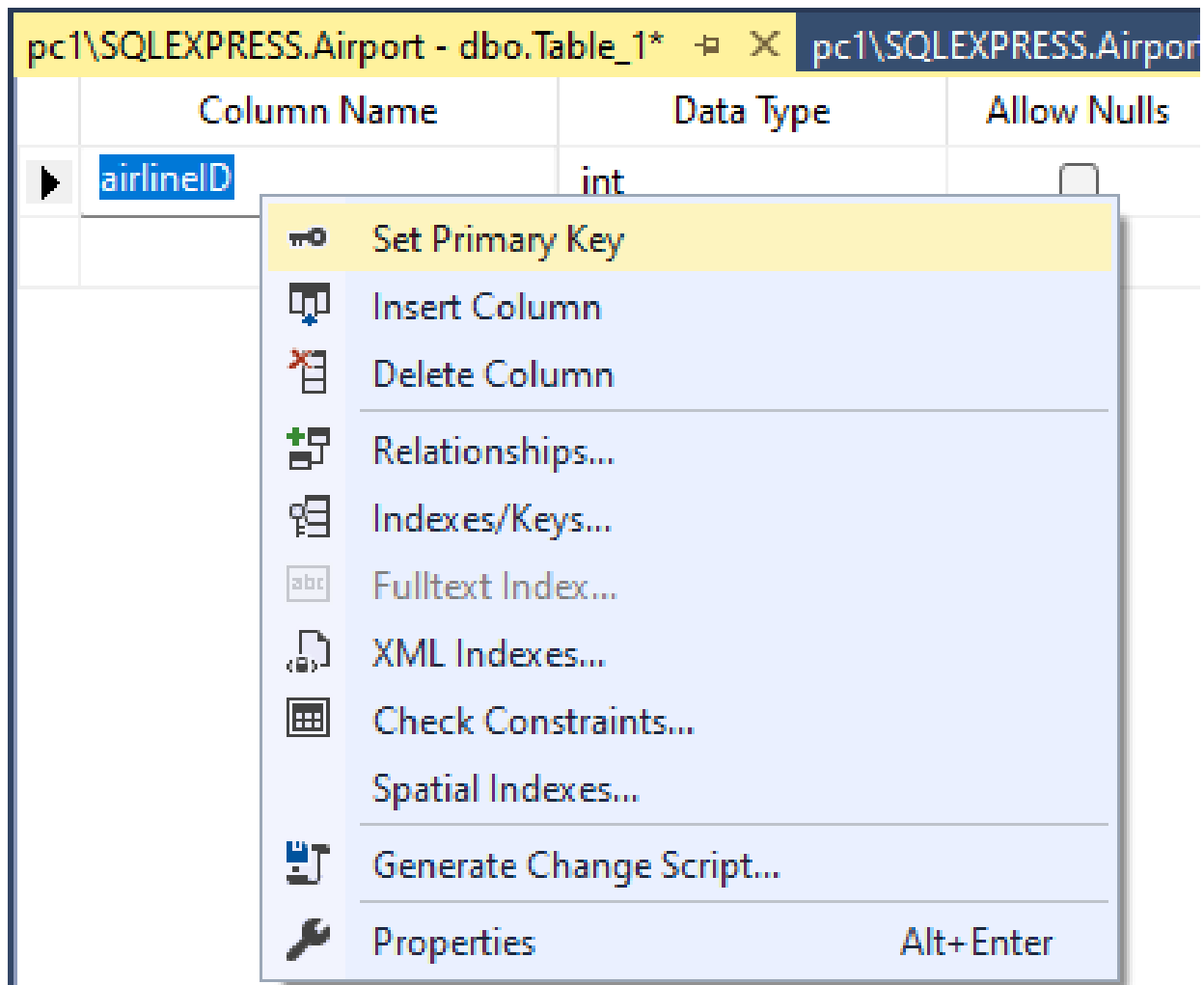
- airlineID (ID aviokompanije)
- name (ime aviokompanije)
- numOfPlanes (broj aviona u floti)
- location (lokacija *headquarters*)
- airportsServed (broj opsluženih aerodroma)
- flightsOperated (broj letova)

pc1\SQLEXPRESS.Airport - dbo.Airline			
	Column Name	Data Type	Allow Nulls
	airlineID	int	<input type="checkbox"/>
	name	nvarchar(30)	<input checked="" type="checkbox"/>
	numOfPlanes	int	<input checked="" type="checkbox"/>
	location	nvarchar(50)	<input checked="" type="checkbox"/>
	airportsServed	int	<input checked="" type="checkbox"/>
	flightsOperated	int	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

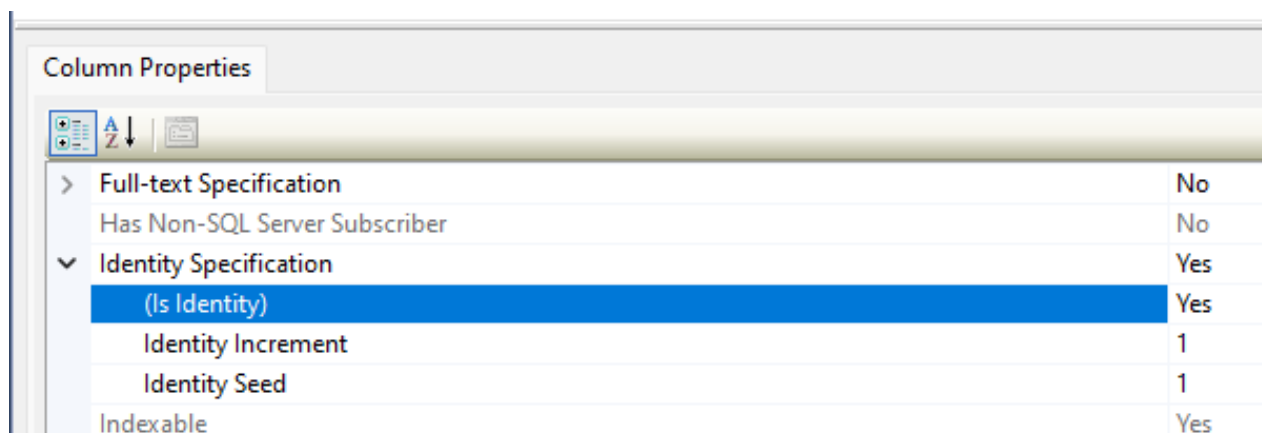
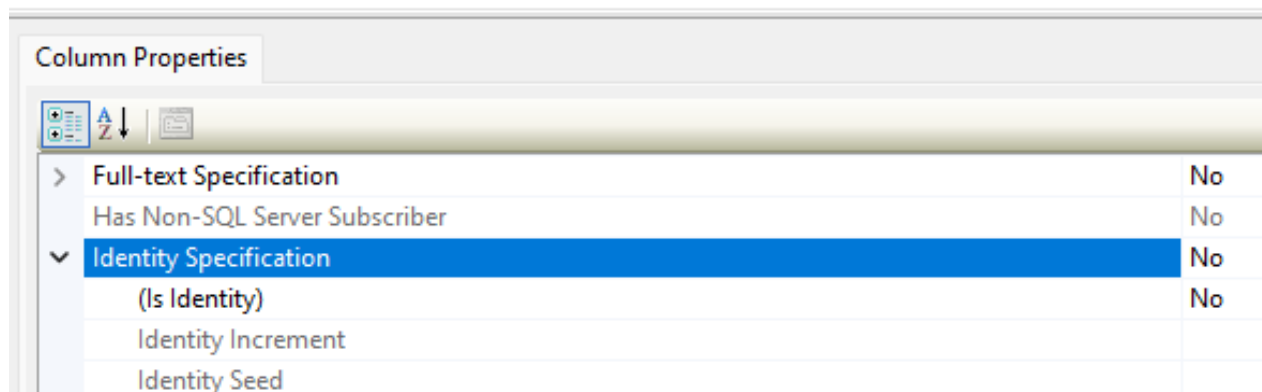
Bitan je tip podatka koji koristimo i to smo već definisali u dijagramu klasa. Sa strane imamo dodatnu opciju da dozvolimo da nema zapisa za neki podatak, odnosno da dozvoljavamo prazna polja u tabeli.

Dobra je praksa čekirati sve osim primarnog ključa (prikazano ikonicom ključa), zato što je moguće da ne znamo svaki podatak u trenutku kreiranja zapisa u tabeli. Primarni ključ ne može da bude *null* zato što je to identifikator tog zapisa i na osnovu toga uparujemo podatke iz tabele!

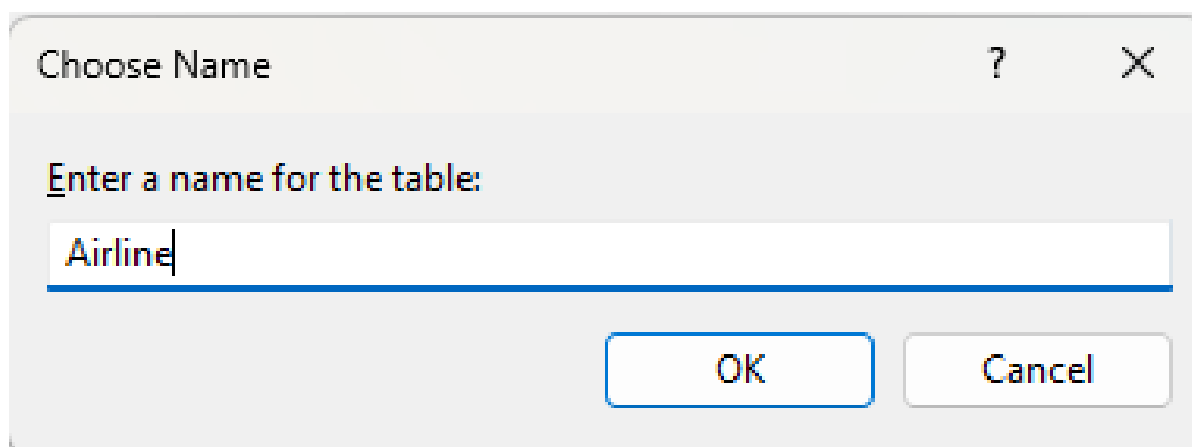
Desnim klikom na kolonu koju želimo da proglasimo za primarni ključ dobijamo tu opciju.



Nakon proglašenja primarnog ključa, dole u svojstvima kolone naći ćemo opciju "Identity Specification", i ispod nje "Is Identity". Duplim klikom na polje "Is Identity" mi ovu kolonu postavljamo kao identitet i automatski se i Identity Specification postavlja kao "Yes", nakon toga nam se prikazuje i za koliko će se primarni ključ uvećavati sa svakim novim unosom, kao i broj od koga kreće "seed".







Kada smo uneli sve kolone, klikom na ikonicu za čuvanje, klikom na *File* pa *Save* ili čak skraćenicom "Ctrl+S", iskače nam prozor u kojem trebamo da unesemo ime tabele.






Nakon unesenog imena i klika na "OK" naša tabela je sačuvana sa tim imenom. Postupak se sad ponavlja za svaku tabelu u bazi. Ono što ćemo pratiti jeste da se primarni ključ nalazi na početku odnosno kao prva kolona tabele, iza nje idu sekundarni ključevi odnosno primarni ključevi drugih tabela, a zatim obične kolone za smeštanje podataka.

Primer za svaku tabelu:

pc1\SQLEXPRESS.Airport - dbo.Airline			
	Column Name	Data Type	Allow Nulls
	airlineID	int	<input type="checkbox"/>
	name	nvarchar(30)	<input checked="" type="checkbox"/>
	numOfPlanes	int	<input checked="" type="checkbox"/>
	location	nvarchar(50)	<input checked="" type="checkbox"/>
	airportsServed	int	<input checked="" type="checkbox"/>
	flightsOperated	int	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

pc1\SQLEXPRESS.Airport - dbo.Airplane			
	Column Name	Data Type	Allow Nulls
	airplaneID	int	<input type="checkbox"/>
	registration	nchar(10)	<input checked="" type="checkbox"/>
	model	nchar(20)	<input checked="" type="checkbox"/>
	dateOfManufacture	date	<input checked="" type="checkbox"/>
	livery	nchar(10)	<input checked="" type="checkbox"/>
	seatCapacity	int	<input checked="" type="checkbox"/>
	range	int	<input checked="" type="checkbox"/>
	flightCeiling	int	<input checked="" type="checkbox"/>
	cruiseSpeed	int	<input checked="" type="checkbox"/>
	maxSpeed	int	<input checked="" type="checkbox"/>
	takeoffWeight	int	<input checked="" type="checkbox"/>
	landingWeight	int	<input checked="" type="checkbox"/>
	minimumRunwayLength	int	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

pc1\SQLEXPRESS.Airport - dbo.Flight			
	Column Name	Data Type	Allow Nulls
	flightID	int	<input type="checkbox"/>
	flightOperatorID	int	<input checked="" type="checkbox"/>
	gateID	int	<input checked="" type="checkbox"/>
	runwayID	int	<input checked="" type="checkbox"/>
	origin	nvarchar(50)	<input checked="" type="checkbox"/>
	destination	nvarchar(50)	<input checked="" type="checkbox"/>
	duration	int	<input checked="" type="checkbox"/>
	route	nvarchar(20)	<input checked="" type="checkbox"/>
	stopovers	int	<input checked="" type="checkbox"/>
	cruiseSpeed	int	<input checked="" type="checkbox"/>
	cruiseAltitude	int	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

pc1\SQLEXPRESS.Ai...dbo.FlightOperator			
	Column Name	Data Type	Allow Nulls
	flightOperatorID	int	<input type="checkbox"/>
	pilotID	int	<input checked="" type="checkbox"/>
	airlineID	int	<input checked="" type="checkbox"/>
	airplaneID	int	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

pc1\SQLEXPRESS.Air...- dbo.FlightTicket ➤ ✕


	Column Name	Data Type	Allow Nulls
▶ 🔑	flightTicketID	int	<input type="checkbox"/>
	passengerID	int	<input type="checkbox"/>
	ticketID	int	<input type="checkbox"/>
	flightID	int	<input type="checkbox"/>
			<input type="checkbox"/>


pc1\SQLEXPRESS.Airport - dbo.Gate ➤ ✕


	Column Name	Data Type	Allow Nulls
▶ 🔑	gateID	int	<input type="checkbox"/>
	number	int	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

pc1\SQLEXPRESS.Air...rt - dbo.Passenger ➤ ✕

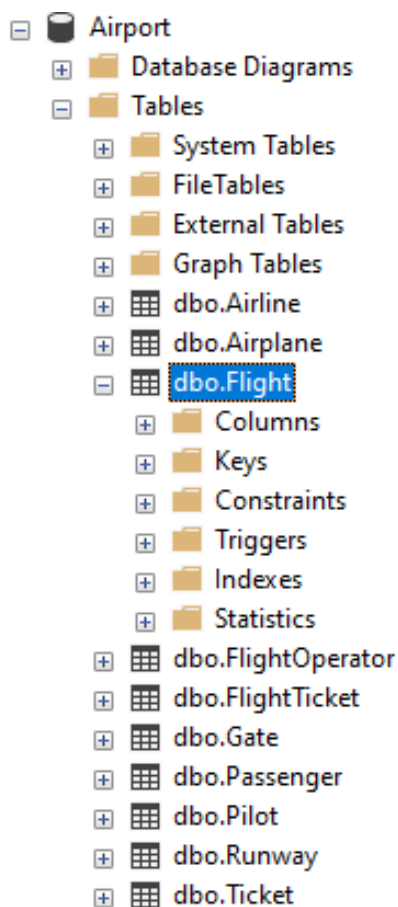
	Column Name	Data Type	Allow Nulls
▶ 🔑	passengerID	int	<input type="checkbox"/>
	name	nvarchar(50)	<input checked="" type="checkbox"/>
	surname	nvarchar(50)	<input checked="" type="checkbox"/>
	age	int	<input checked="" type="checkbox"/>
	contact	nvarchar(50)	<input checked="" type="checkbox"/>
	address	nvarchar(50)	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

pc1\SQLEXPRESS.Airport - dbo.Pilot ↗ ✕			
	Column Name	Data Type	Allow Nulls
	pilotID	int	<input type="checkbox"/>
	name	nvarchar(20)	<input checked="" type="checkbox"/>
	surname	nvarchar(30)	<input checked="" type="checkbox"/>
	age	int	<input checked="" type="checkbox"/>
	contact	nvarchar(50)	<input checked="" type="checkbox"/>
	address	nvarchar(50)	<input checked="" type="checkbox"/>
	typeRating	nchar(20)	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

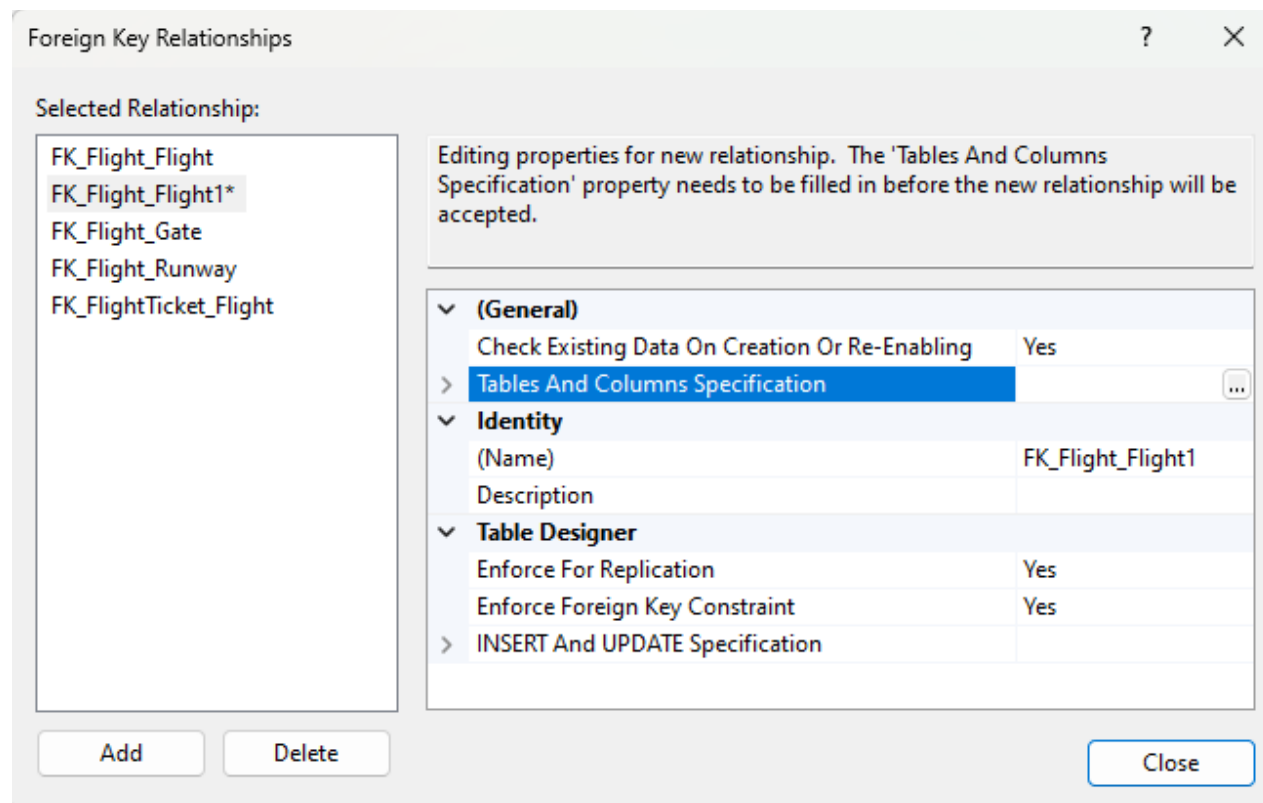
pc1\SQLEXPRESS.Airport - dbo.Runway ↗ ✕			
	Column Name	Data Type	Allow Nulls
	runwayID	int	<input type="checkbox"/>
	runwayNumber	int	<input checked="" type="checkbox"/>
	orientation	nchar(1)	<input checked="" type="checkbox"/>
	length	int	<input checked="" type="checkbox"/>
	loadRating	int	<input checked="" type="checkbox"/>
	closed	bit	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

pc1\SQLEXPRESS.Airport - dbo.Ticket			
	Column Name	Data Type	Allow Nulls
	ticketID	int	<input type="checkbox"/>
	flightNumber	nchar(20)	<input checked="" type="checkbox"/>
	name	nvarchar(50)	<input checked="" type="checkbox"/>
	date	date	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

Nakon kreiranja svih tabela i beleženja njihovih kolona, neophodno je kreirati vezu između primarnih i sekundarnih ključeva. Klikom na “+” pored nekih od tabela koje sadrže strane ključeve proširujemo hijerarhiju i vidimo sledeće:



Desnim klikom na “Keys” dobijamo opciju da “dodamo” novi strani ključ, mi smo strane ključeve zapravo već dodali, sada dodajemo njihovu vezu. Otvara nam se prozor sa vezama stranih ključeva i on izgleda ovako:



Interesuje nas ime ove veze, radi lakše administracije nazvaćemo ga nečim što ima smisla, ili ćemo ostaviti podrazumevano ime. Dalje klikom na “Tables And Columns Specification” nam se pojavljuju tri tačke kao unos, i na to klikćemo.

Tables and Columns



Relationship name:

FK_Flight_Flight1

Primary key table:

Flight



Foreign key table:

Flight

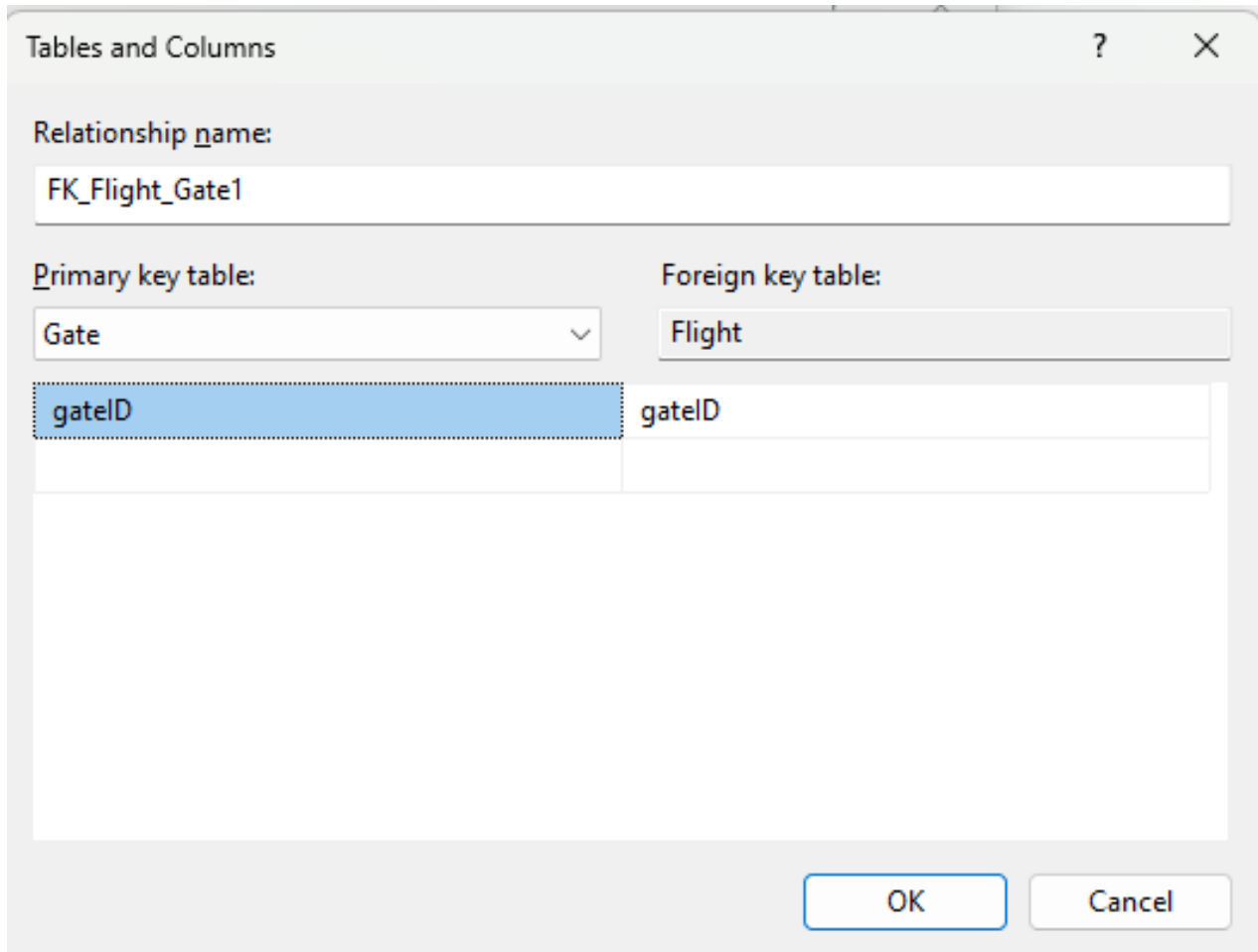
flightID

flightID

OK

Cancel

Tu nam opet izlazi ime veze ali i ono nama bitnije, primarne tabele i strane tabele. Za primarnu stavljamo onu tabelu u kojoj se nalazi taj strani ključ, stranu tabelu ne diramo, a ključeve uparujemo iz primarne sa sekundarnom.



Tables and Columns

Relationship name:
FK_Flight_Gate1

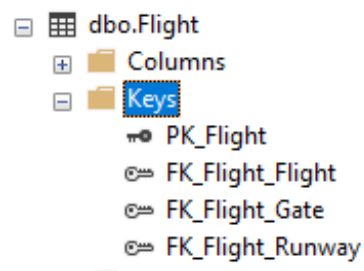
Primary key table:
Gate

Foreign key table:
Flight

Primary key column	Foreign key column
gateID	gateID

OK Cancel

Ovaj postupak ponavljamo za svaku tabelu u kojoj imamo neke strane ključeve onoliko puta koliko imamo stranih ključeva. Na kraju se u *folderu* "Keys" nalaze jedan primarni i jedan ili više stranih ključeva. Primer:



Ovim smo pokrili osnove kreiranja baze podataka, prilikom svake izmene tabela potrebno je iste sačuvati i bazu podataka osvežiti ukoliko se promene ne prikazuju odmah.

5.2. Upiti i DML naredbe

Da bi smo podatke uneli u našu bazu podataka moramo koristiti upite odnosno komande koje upisuju podatke u tabele. Pored unosa, pregledanje, izmena, i brisanje se sve vrši preko upita.

Da bismo nešto prikazali, neophodno je da unesemo šta prikazujemo i odakle ga prikazujemo.

Da bismo nešto uneli, koristimo obeležja:

- Ime tabele
- Ime kolona (ukoliko ne unosimo podatke za sve kolone u onom redosledu u kom se nalaze u bazi)
- Vrednosti (podaci)

Da bismo nešto ažurirali, koristimo obeležja:

- Ime tabele
- Ime kolone i vrednost koju ažuriramo
- Podatak po kom identifikujemo gde izmena treba da se dogodi (ID).

Da bismo nešto obrisali, koristimo obeležja:

- Ime tabele
- Podatak identifikacije po kom se briše unos iz tabele (ID).
 - Bitno je napomenuti da prilikom brisanja nismo u mogućnosti da obrišemo onaj podatak koji se koristi u drugoj tabeli kao sekundarni.

Primer insert-a, select-a, update-a, i delete-a možete videti ovde:

```

insert into Airline
values (N'Air Serbia', '25', N'Belgrade, Serbia', '24', '41');
insert into Airline
values (N'Ryan Air', '430', N'Dublin, Ireland', '120', '186');
insert into Airline
values (N'Aeroflot', '336', N'Moscow, Russia', '122', '189');
insert into Airline
values (N'SAS', '134', N'Stockholm, Sweden', '99', '124');
insert into Airline
values (N'LOT Polish Airlines', '91', N'Warsaw, Poland', '84', '120');

insert into Pilot
values (N'Marko', N'Pavlović', '32', N'marko.pavlo@airserbia.rs', N'Ustanička 68, Beograd', 'CL');
insert into Pilot
values (N'Paul', N'Haunders', '58', N'paulhaunders@dublinairport.ie', N'7th Street 45, Dublin', 'CL');
insert into Pilot
values (N'Mihail', N'Nauchenko', '44', N'mihailfixstuff@gmail.com', N'Ovchinnikovskaya Naberezhnaya 54, Moscow', null);
insert into Pilot
values (N'Ann-Charlott', N'Gunnarsson', '41', null, N'Fannys vag 5, Stockholm', 'A0');
insert into Pilot
values (N'Maks', N'Kowalczyk', null, N'kowalskianalysis@lotarilines.pl', N'Pabianicka 28, Lodz', 'NL');

insert into Runway (runwayNumber, length)
values('54', '3200');
insert into Runway (runwayNumber, length)
values('45', '3200');
insert into Runway (runwayNumber, length)
values('13', '2980');
insert into Runway (runwayNumber, length)
values('14', '2920');
insert into Runway (runwayNumber, length)
values('10', '2700');

insert into Airplane (registration, model, seatCapacity, range)
values('YU-234', 'A319', '70', '13000');
insert into Airplane (registration, model, seatCapacity, range)
values('IE-R41', 'B777-400', '320', '16000');
insert into Airplane (registration, model, seatCapacity, range)
values('RU-R555', 'A340', '280', '12200');
insert into Airplane (registration, model, seatCapacity, range)
values('SW-786', 'B747-800', '380', '12000');
insert into Airplane (registration, model, seatCapacity, range)
values('PL-230', 'A380', '480', '14000');

```

```

INSERT INTO FlightOperator (pilotID, airlineID, airplaneID)
SELECT
    (SELECT pilotID FROM Pilot WHERE name = 'Marko'),
    (SELECT airlineID FROM Airline WHERE name = 'Air Serbia'),
    (SELECT airplaneID FROM Airplane WHERE model = 'A319');
INSERT INTO FlightOperator (pilotID, airlineID, airplaneID)
SELECT
    (SELECT pilotID FROM Pilot WHERE name = 'Marko'),
    (SELECT airlineID FROM Airline WHERE name = 'Aeroflot'),
    (SELECT airplaneID FROM Airplane WHERE model = 'A340');
INSERT INTO FlightOperator (pilotID, airlineID, airplaneID)
SELECT
    (SELECT pilotID FROM Pilot WHERE name = 'Paul'),
    (SELECT airlineID FROM Airline WHERE name = 'Aeroflot'),
    (SELECT airplaneID FROM Airplane WHERE model = 'A380');
INSERT INTO FlightOperator (pilotID, airlineID, airplaneID)
SELECT
    (SELECT pilotID FROM Pilot WHERE name = 'Maks'),
    (SELECT airlineID FROM Airline WHERE name = 'SAS'),
    (SELECT airplaneID FROM Airplane WHERE model = 'B777-400');
INSERT INTO FlightOperator (pilotID, airlineID, airplaneID)
SELECT
    (SELECT pilotID FROM Pilot WHERE name = 'Maks'),
    (SELECT airlineID FROM Airline WHERE name = 'Ryan Air'),
    (SELECT airplaneID FROM Airplane WHERE model = 'A380');

```

```

update Airline
Set numOfPlanes='350'
where name='Ryan Air';
update Airplane
Set livery='black12'
where airplaneID='5';
update Flight
Set stopovers='2'
where destination='New York';
update FlightOperator
Set airplaneID='3'
where pilotID='3';

```

```

delete from Airplane
where airplaneID='4';
delete from Pilot
where pilotID='3';

```

```

select
    Passenger.name as 'Putnik',
    Ticket.name as 'Route',
    Flight.duration as 'Duration(hours)',
    Ticket.date as 'Date'
from FlightTicket
join Passenger on FlightTicket.passengerID = Passenger.passengerID
join Ticket on FlightTicket.ticketID = Ticket.ticketID
join Flight on FlightTicket.flightID = Flight.flightID;

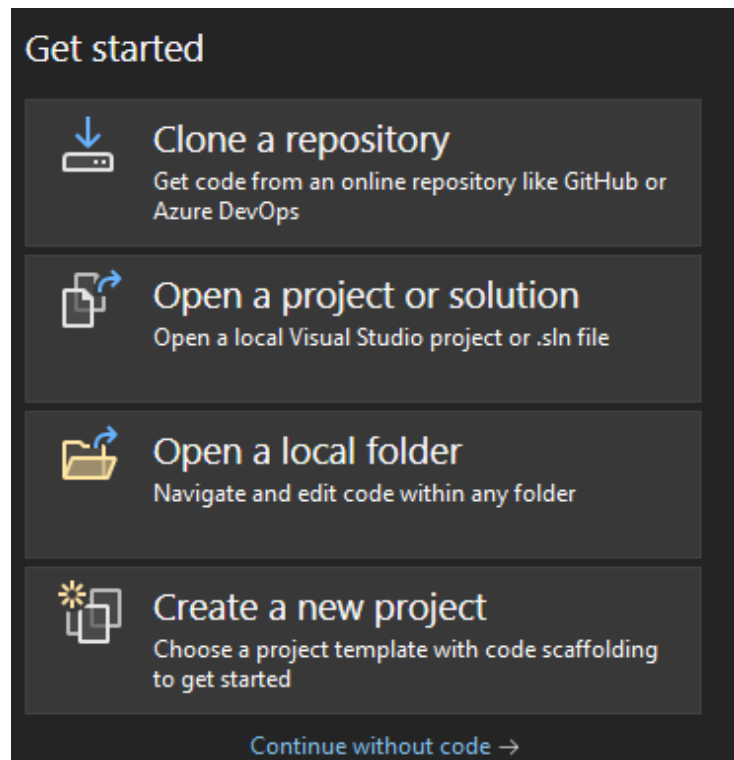
Select *
From Gate;
Select name
From Passenger;
Select runwayNumber
From Runway
where runwayID='1';

```


6. Kreiranje WPF aplikacije (desktop app)

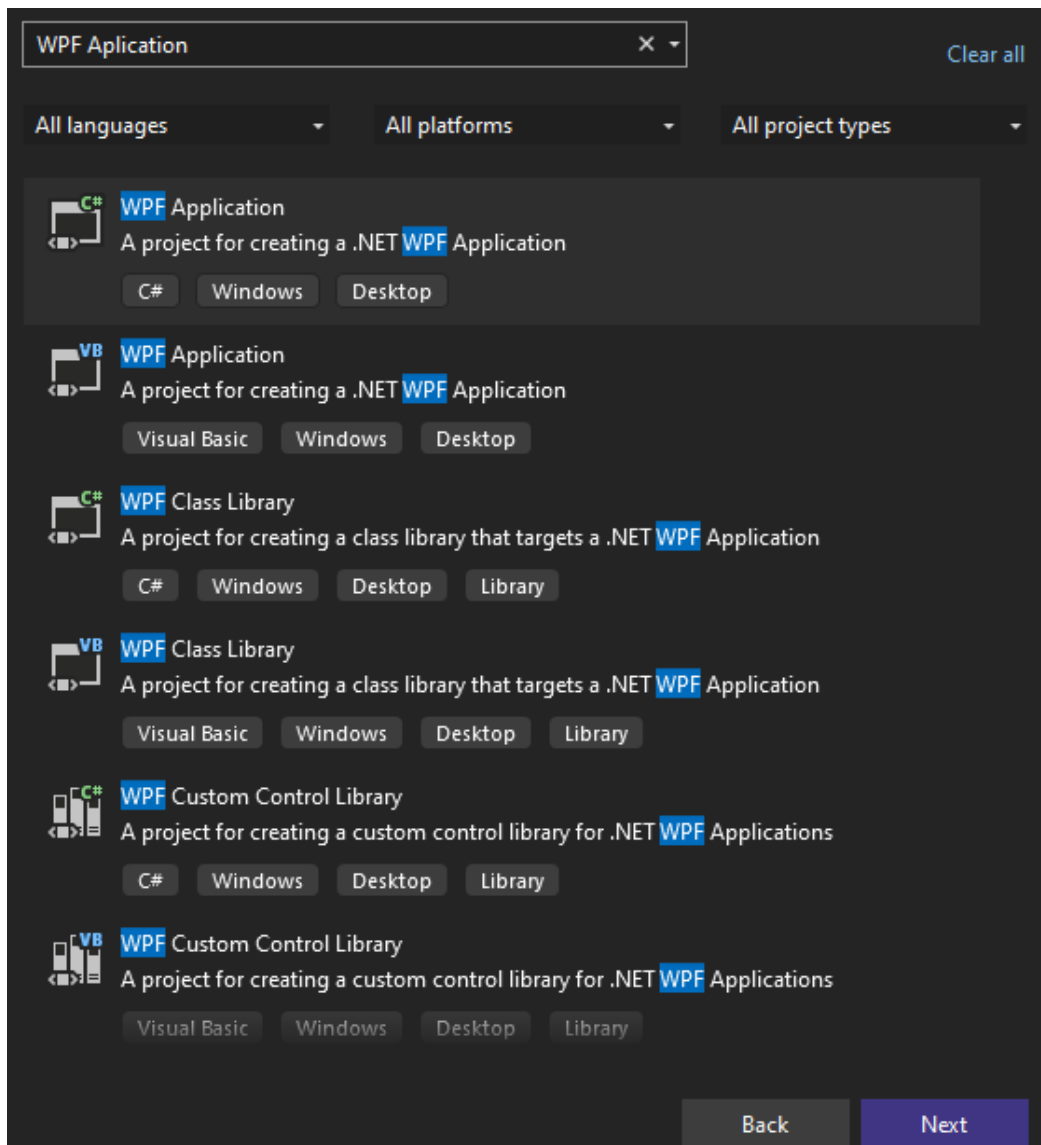
U ovom delu ćemo opisati šta nam je sve neophodno da bi se naša desktop aplikacija uspešno povezala sa bazom i omogućila nam da unosimo podatke.

Za početak pokrećemo Visual Studio i kreiramo novi projekat.



Klikom na “Create a new project” otvara nam se sledeći prozor.

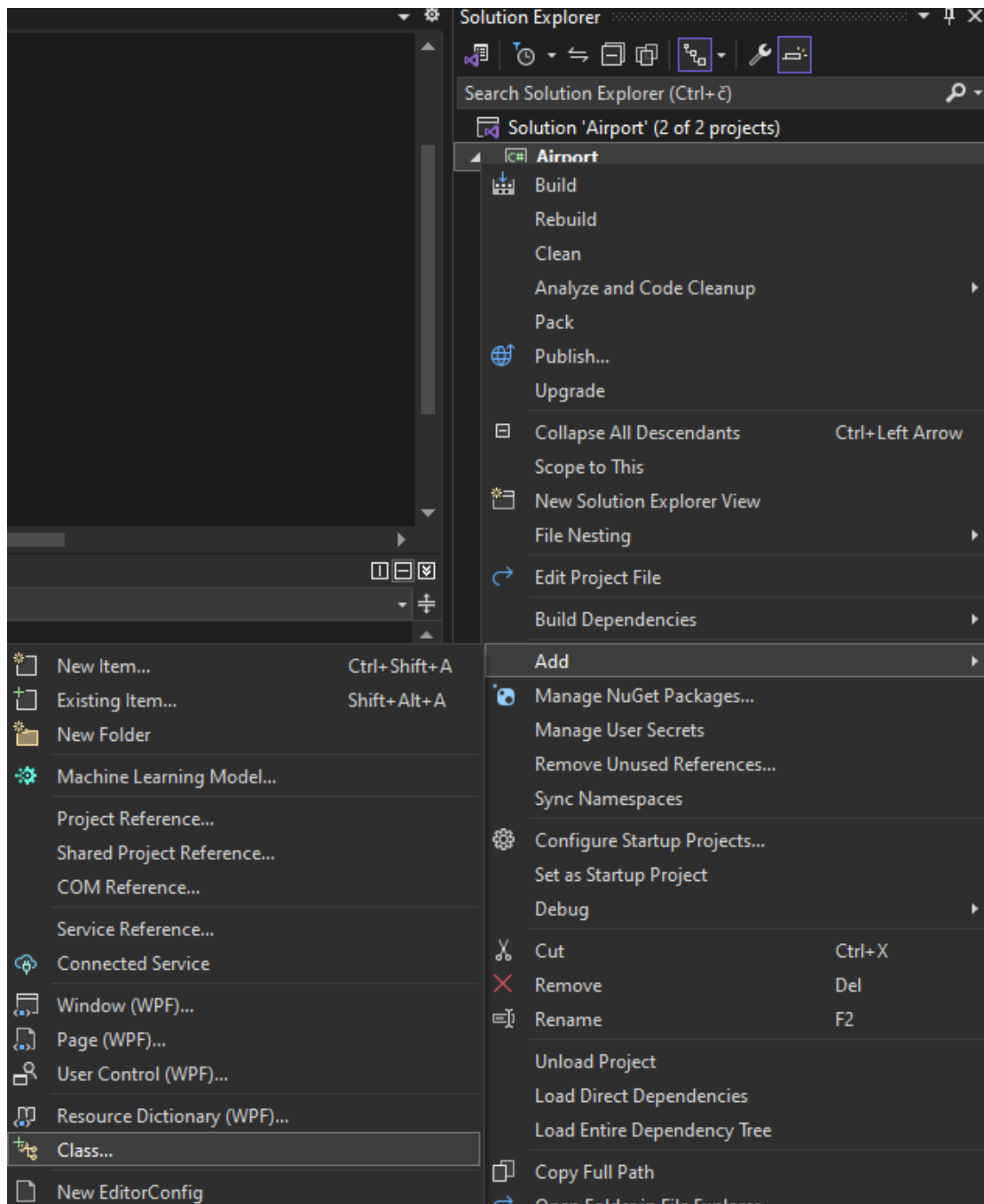
U njemu tražimo “WPF Application” stavku, C# verziju, naznačeno simbolom u desnom gornjem uglu.



Klikom na “Next” čarobnjak nas pita za ime projekta i njegovu lokaciju, to radimo po našem izboru, daljem klikom na “Next” biramo “Framework”, ovo ostavljamo na podrazumevanoj vrednosti. Konačno kliknemo “Create”.

Sada možemo da krenemo sa kreiranjem naše aplikacije.

Prvo je neophodno da definišemo klasu konekcije i u njoj sve parametre potrebne za istu.



Desni klik na naziv projekta, “Add” i onda “Class”.

```

namespace Airport
{
    22 references
    internal class Connection
    {
        11 references
        public SqlConnection generateConnection()
        {
            SqlConnectionStringBuilder connBuilder = new SqlConnectionStringBuilder
            {
                DataSource = @"PC1\SQLEXPRESS",
                InitialCatalog = "Airport",
                IntegratedSecurity = true
            };
            SqlConnection connection = new SqlConnection(connBuilder.ToString());
            return connection;
        }
    }
}

```

Ovo je dosta jednostavno, ovde upisujemo isto što nas je i SQL Server Management Studio pitao pri otvaranju, a to su server, na koju bazu se povezujemo i na koji način vršimo autorizaciju.

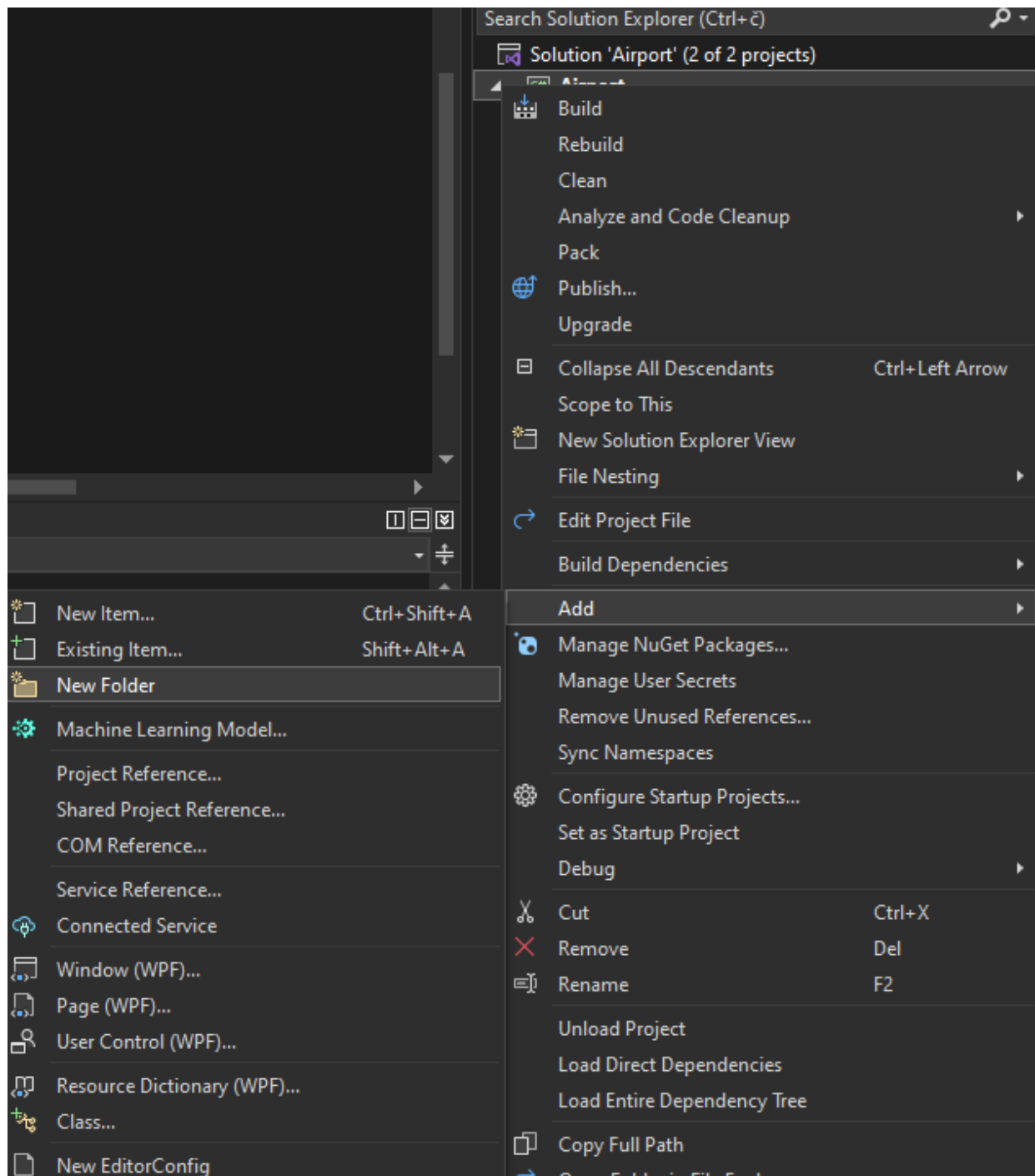
Ova klasa nam vraća konekcionu string koji naš program zna da koristi kako bi kasnije otvorio vezu sa bazom.

Nakon toga kreiramo prvu formu, to će nam biti glavni prozor naše aplikacije.

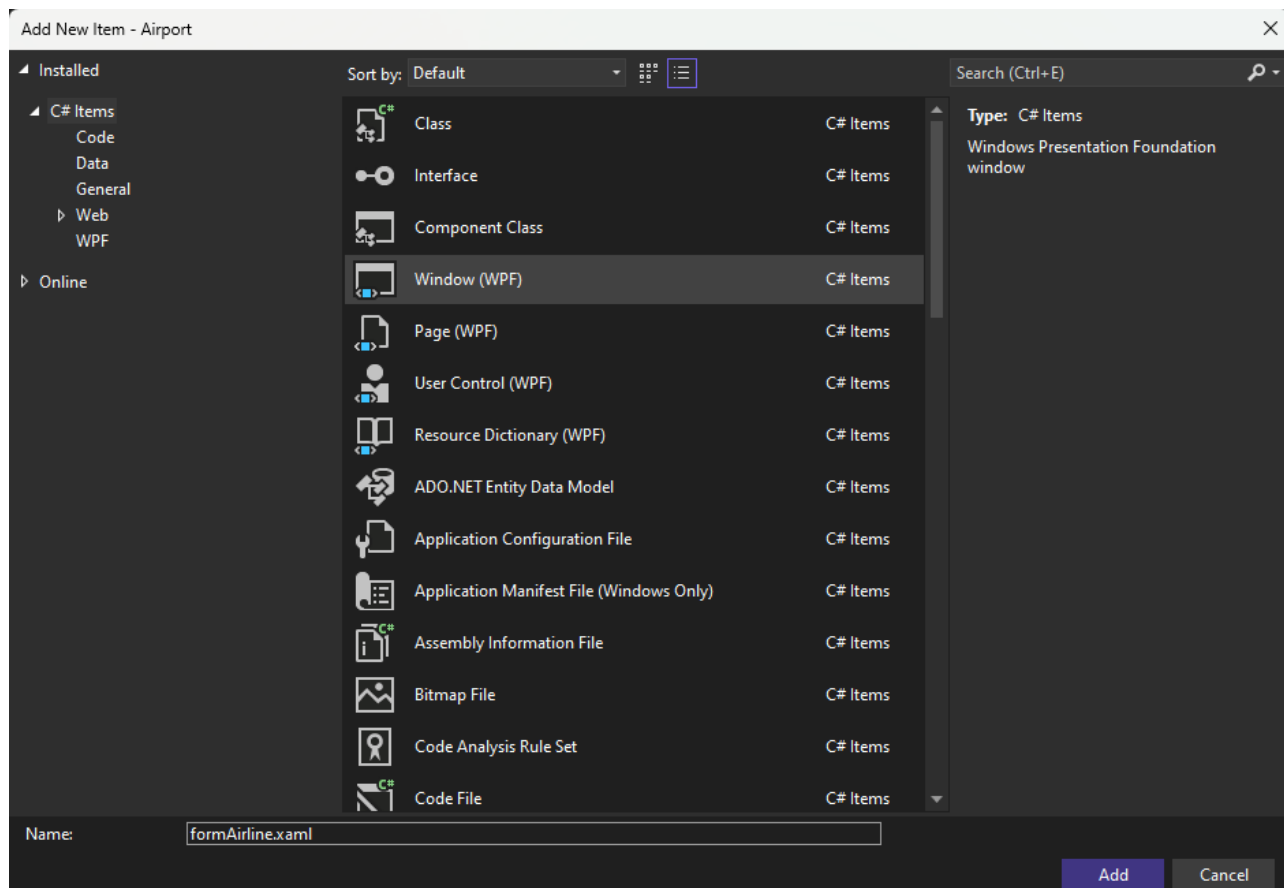
Pravimo novi folder unutar projekta koji ćemo nazvati "forms".

Primitimo da sve dodajemo na isti način, desni klik na projekat, "Add", pa onda opciju koja nam odgovara, sledeća će biti "Window (WPF)". Odmah možemo da dodamo sve forme koje su nam neophodne za dalju realizaciju sistema.

"MainWindow" formu ćemo kreirati na nivou projekta, van foldera "forms", a ostale forme će se nalaziti unutar foldera forms.



Pri dodavanju formi vidimo sledeći prozor:



Biramo Window(WPF) i dajemo ime po želji, u našem slučaju “formAirline.xaml”.

U glavnom prozoru će nam se nalaziti:

- Tabela u kojoj će se prikazivati podaci koji postoje u našem sistemu
- Dugmići sa kojim menjamo prikazanu tabelu
- Dugmići za unos podataka, izmenu i brisanje

Koristićemo elemente Button, Grid, DataGrid, WrapPanel i StackPanel kako bi kreirali malopre opisano. Kao ime ovih elemenata stavljamo ono što ćemo koristiti u kodu, “Content” predstavlja ono što vidimo u aplikaciji, pa tako za dugme “btnSave” “Content” će biti “Sačuvaj”.

Primer ovoga za “MainWindow” možemo videti dole:

```

<Window x:Class="Airport.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        mc:Ignorable="d"
        Title="MainWindow" Height="500" Width="850">
    <Grid>
        <DataGrid x:Name="dataGridCenter" VerticalAlignment="Center" HorizontalAlignment="Center" Width="auto" Height="auto"
                  IsReadOnly="true" AlternatingRowBackground="Aqua" CanUserAddRows="False" SelectionUnit="FullRow" SelectionMode="Single"/>
        <WrapPanel HorizontalAlignment="Center" VerticalAlignment="Top" Height="auto" Width="500" Orientation="Horizontal">
            <Button x:Name="btnAirline" Content="Aviokompanije" HorizontalAlignment="Left" VerticalAlignment="Top" Height="30" Width="100" Click="btnAirline_Click"/>
            <Button x:Name="btnAirplane" Content="Avioni" VerticalAlignment="Bottom" Height="30" HorizontalAlignment="Right" Width="100" Click="btnAirplane_Click"/>
            <Button x:Name="btnFlight" Content="Letovi" VerticalAlignment="Bottom" Height="30" HorizontalAlignment="Right" Width="100" Click="btnFlight_Click"/>
            <Button x:Name="btnFlightOperator" Content="Operateri leta" VerticalAlignment="Bottom" Height="30" HorizontalAlignment="Right" Width="100" Click="btnFlightOperator_Click"/>
            <Button x:Name="btnFlightTicket" Content="Parovi: karta - let" VerticalAlignment="Bottom" Height="30" HorizontalAlignment="Right" Width="100" Click="btnFlightTicket_Click"/>
            <Button x:Name="btnGate" Content="Gejt" VerticalAlignment="Bottom" Height="30" HorizontalAlignment="Right" Width="100" Click="btnGate_Click"/>
            <Button x:Name="btnPassenger" Content="Putnici" VerticalAlignment="Bottom" Height="30" HorizontalAlignment="Right" Width="100" Click="btnPassenger_Click"/>
            <Button x:Name="btnPilot" Content="Piloti" VerticalAlignment="Bottom" Height="30" HorizontalAlignment="Right" Width="100" Click="btnPilot_Click"/>
            <Button x:Name="btnRunway" Content="Piste" VerticalAlignment="Bottom" Height="30" HorizontalAlignment="Right" Width="100" Click="btnRunway_Click"/>
            <Button x:Name="btnTicket" Content="Karte" VerticalAlignment="Bottom" Height="30" HorizontalAlignment="Right" Width="100" Click="btnTicket_Click"/>
        </WrapPanel>
        <StackPanel HorizontalAlignment="Center" VerticalAlignment="Bottom" Height="auto" Width="auto" Orientation="Horizontal">
            <Button x:Name="btnAdd" Content="Dodaj" HorizontalAlignment="Left" VerticalAlignment="Top" Height="30" Width="80" Click="btnAdd_Click"/>
            <Button x:Name="btnChange" Content="Izmeni" VerticalAlignment="Bottom" Height="30" HorizontalAlignment="Right" Width="80" Click="btnChange_Click"/>
            <Button x:Name="btnDelete" Content="Obrisi" VerticalAlignment="Bottom" Height="30" HorizontalAlignment="Right" Width="80" Click="btnDelete_Click"/>
        </StackPanel>
    </Grid>
</Window>

```

Ovde nas najviše interesuje element Button, DataGrid, kao i visina i širina glavnog prozora.

```

<Button x:Name="btnAirline" Content="Aviokompanije" HorizontalAlignment="Left"
        VerticalAlignment="Top" Height="30" Width="100" Click="btnAirline_Click"/>

```

```

<DataGrid x:Name="dataGridCenter" VerticalAlignment="Center" HorizontalAlignment="Center"
           Width="auto" Height="auto" IsReadOnly="true" AlternatingRowBackground="Aqua"
           CanUserAddRows="False" SelectionUnit="FullRow" SelectionMode="Single"/>

```

Zatim za svaku formu u folderu “forms” radimo slično.

Sada koristimo kombinaciju elementa “Label”, “TextBox”, “CheckBox”, “ComboBox” i “DataPicker” kako bismo ostvarili ono što smo zamislili:

formRunway

Broj:

Orijentacija:

Duzina:

Maksimalna težina:

Zatvorena:

☐

Save

Cancel

formTicket

Broj leta:

Početak-Destinacija:

Datum:

Select a date



Save

Cancel

formFlight

Početak:

Destinacija:

Trajanje leta:

Ruta:

Broj presedanja:

Brzina kruziranja:

Visina kruziranja:

Operater leta:

Gejt:

Pista:

Save

Cancel

Primer koda za nove elemente:

```
<Label Content="Početak:" HorizontalAlignment="Left" Height="28" Margin="10,10,0,0"
VerticalAlignment="Top"/>
```

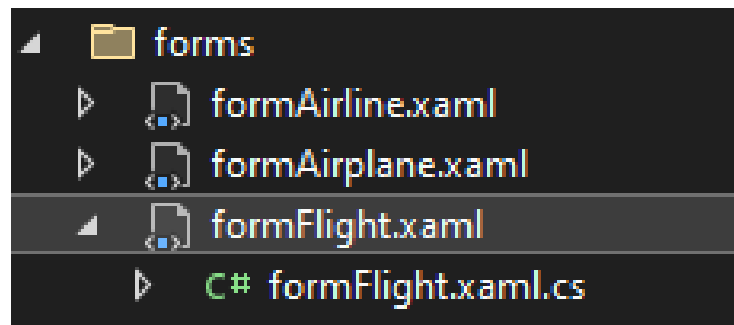
```
<TextBox x:Name="txtOrigin" TextWrapping="Wrap" Text="" Width="160"
Margin="0,12,20,0" HorizontalAlignment="Right" Height="24" VerticalAlignment="Top"/>
```

```
<ComboBox x:Name="cmbRunway" HorizontalAlignment="Right" Margin="0,282,20,0"
VerticalAlignment="Top" Width="160" DisplayMemberPath="Runway"
SelectedValuePath="runwayID"/>
```

```
<DatePicker x:Name="dpFlightDate" HorizontalAlignment="Right" Margin="0,72,12,0"
VerticalAlignment="Top" Width="143"/>
```

```
<CheckBox x:Name="ccbxClosed" Content="" HorizontalAlignment="Right"
Margin="0,138,12,0" VerticalAlignment="Top" Width="143"/>
```

Nakon kreiranih .xaml formi treba da odradimo logiku u kodu. Klikom na strelicu pored neke od formi u *Solution Explorer*-u videćemo fajl sa istim imenom ali ekstenzijom .cs



Duplim klikom na taj fajl ga otvaramo i započinjemo backend logiku.

U "MainWindow" fajlu dodajemo konekciju, to ćemo raditi za svaku formu, dodajemo *string* koji ćemo koristiti kao obeležje koju tabelu trenutno imamo prikazanu, *bool* koji predstavlja oznaku da li ažuriramo podatke ili dodajemo nove, i još jedan specijalan podatak *DataRowView* kojim pamtimmo koji red je izabran u tabeli.

```
public partial class MainWindow : Window
{
    Connection conn = new Connection();
    SqlConnection connection = new SqlConnection();
    private string? currentTable;
    private bool update;
    private DataRowView rowView;
```

Zatim moramo definisati upite koje smo već prošli u ovom dokumentu. Imamo tri vrste, insert, select i delete. Ovde koristimo select i select po *ID-u* i delete.

```
private static string airlineSelect = @"select airlineID as ID, name as Ime, numOfPlanes as 'Broj aviona',
location as 'Sedište', airportsServed as 'Broj opsluženih aerodroma', flightsOperated as 'Broj letova' from Airline";
private static string airplaneSelect = @"select airplaneID as ID, registration as Registracija, model as Model, dateOfManufacture as 'Datum proizvodnje',
livery as Farba, seatCapacity as 'Broj sedišta', range as Domet, flightCeiling as 'Maksimalna visina leta',
cruiseSpeed as 'Brzina kružiranja', maxSpeed as 'Maksimalna brzina', takeoffWeight as 'Težina pri poletanju',
landingWeight as 'Težina pri sletanju', minimumRunwayLength as 'Minimalna dužina piste' from Airplane";
private static string flightSelect = @"select flightID as ID, origin as Polazak, destination as 'Odredište', duration as 'Vreme leta',
stopovers as Presedanja, cruiseSpeed as 'Brzina kružiranja', cruiseAltitude as 'Visina kružiranja',
runwayNumber as Pista, Gate.number as 'Gejt' from Flight
join Runway on Flight.runwayID = Runway.runwayID
join Gate on Flight.gateID = Gate.gateID";
private static string flightOperatorSelect = @"select flightOperatorID as ID, Pilot.name + ' ' + Pilot.surname as Pilot, Airline.name as 'Aviokompanija',
Airplane.model + ' | ' + Airplane.registration as 'Avion' from FlightOperator
join Pilot on FlightOperator.pilotID = Pilot.pilotID
join Airline on FlightOperator.airlineID = Airline.airlineID
join Airplane on FlightOperator.airplaneID = Airplane.airplaneID";
private static string flightTicketSelect = @"select flightTicketID as ID, Passenger.name + ' ' + Passenger.surname as Putnik, Ticket.flightNumber as 'Broj karte',
Flight.origin + ' ' + Flight.Destination as Let from FlightTicket
join Passenger on FlightTicket.passengerID = Passenger.passengerID
join Ticket on FlightTicket.ticketID = Ticket.ticketID
join Flight on FlightTicket.flightID = Flight.flightID";
private static string gateSelect = @"select gateID as ID, number as Broj from Gate";
private static string passengerSelect = @"select passengerID as ID, name as Ime, surname as Prezime, age as Starost, contact as Kontakt, address as Adresa from Passenger";
private static string pilotSelect = @"select pilotID as ID, name as Ime, surname as Prezime, age as Starost, contact as Kontakt, address as Adresa,
typeRating as Licenca from Pilot";
private static string runwaySelect = @"select runwayID as ID, runwayNumber as 'Broj piste', orientation as Orijentacija, length as 'Dužina',
loadRating as 'Maks. opterećenje', closed as Zatvorena from Runway";
private static string ticketSelect = @"select ticketID as ID, flightNumber as 'Broj leta', name as 'Polazak-Odredište', date as 'Datum' from Ticket";
```

```
private static string selectStatementAirline = @"select * from Airline where airlineID=";
private static string selectStatementAirplane = @"select * from Airplane where airplaneID=";
private static string selectStatementFlight = @"select * from Flight where flightID=";
private static string selectStatementFlightOperator = @"select * from FlightOperator where flightOperatorID=";
private static string selectStatementFlightTicket = @"select * from FlightTicket where flightTicketID=";
private static string selectStatementGate = @"select * from Gate where gateID=";
private static string selectStatementPassenger = @"select * from Passenger where passengerID=";
private static string selectStatementPilot = @"select * from Pilot where pilotID=";
private static string selectStatementRunway = @"select * from Runway where runwayID=";
private static string selectStatementTicket = @"select * from Ticket where ticketID=";
```

```
private static string deleteAirline = @"delete from Airline where airlineID=";
private static string deleteAirplane = @"delete from Airplane where airplaneID=";
private static string deleteFlight = @"delete from Flight where flightID=";
private static string deleteFlightOperator = @"delete from FlightOperator where flightOperatorID=";
private static string deleteFlightTicket = @"delete from FlightTicket where flightTicketID=";
private static string deleteGate = @"delete from Gate where gateID=";
private static string deletePassenger = @"delete from Passenger where passengerID=";
private static string deletePilot = @"delete from Pilot where pilotID=";
private static string deleteRunway = @"delete from Runway where runwayID=";
private static string deleteTicket = @"delete from Ticket where ticketID=";
```

Zatim dodajemo u konstruktor forme string konekcije i učitavanje podataka, ovu metodu nismo još definisali ali uskoro ćemo.

```
public MainWindow()
{
    InitializeComponent();
    connection = conn.generateConnection();
    loadData(airlineSelect);
}
```

Sledi nam metoda za učitvanje podataka:

```
private void loadData(string selectString)
{
    try
    {
        connection.Open();
        SqlDataAdapter dataAdapter = new SqlDataAdapter(selectString, connection);
        DataTable dataTable = new DataTable();
        dataAdapter.Fill(dataTable);
        if (dataGridCenter != null)
            dataGridCenter.ItemsSource = dataTable.DefaultView;
        currentTable = selectString;
        dataAdapter.Dispose();
        dataTable.Dispose();
    }
    catch (SQLException ex)
    {
        MessageBox.Show("Neuspešno učitani podaci!\n" + ex.Message, "Greška", Message
    }
    catch (Exception ex)
    {
        MessageBox.Show("Došlo je do greške u toku obrade!\n" + ex.Message, "Greška",
    }
    finally
    {
        if (connection != null) connection.Close();
    }
}
```

Kratak opis onoga što se dešava ovde: Otvaramo konekciju u *try* bloku, pravimo objekte klase “SqlDataAdapter” i “DataTable”, popunjavamo ih podacima iz baze, kao izvor podataka za “DataGrid” stavljamo novo kreiranu tabelu, pamtimo koju tabelu smo učitali, i oslobađamo “SqlDataAdapter” i “DataTable”. Konačno u *finally* bloku zatvarmo konekciju. *Catch* blok nam služi da greške nastale u toku izvršavanja ovih linija koda uhvati i spreči pucanje programa.

Zatim ćemo definisati logiku koja će da popuni svaku od formi sa podacima u slučaju da je izabrana opcija ažuriranje.

```
connection.Open();
update = true;
rowView = (DataRowView)dataGridCenter.SelectedItems[0];
SqlCommand cmd = new SqlCommand { Connection = connection };
cmd.Parameters.Add("@id", SqlDbType.Int).Value = rowView["ID"];
cmd.CommandText = selectStatement + "@id";
SqlDataReader reader = cmd.ExecuteReader();
```

Otvaramo konekciju, stavljamo parametar *update* na *true*, preuzimamo podatak o tome koji je red izabran, generišemo upit sa kojim ćemo da nađemo te podatke, lepimo select string po *id-u* i *ID* koji smo našli. Zatim ukoliko smo uspeali da pročitamo podatke, upisujemo ih u formu.

```
if (reader.Read())
{
    if (currentTable == airlineSelect)
    {
        formAirline tempForm = new formAirline(update, rowView);
        tempForm.txtAirlineName.Text = reader["name"].ToString();
        tempForm.txtNumOfPlanes.Text = reader["numOfPlanes"].ToString();
        tempForm.txtLocation.Text = reader["location"].ToString();
        tempForm.txtAirportServed.Text = reader["airportsServed"].ToString();
        tempForm.txtFlightsOperated.Text = reader["FlightsOperated"].ToString();
        tempForm.ShowDialog();
    }
}
```

Neophodno je proveriti koju tabelu smo imali izabranu u momentu pritiskanja dugmeta "Izmeni". Kada smo pronašli odgovarajuću tabelu radimo sledeće:

Pravimo novi objekat klase *from[formName]* sa podacima o tome da želimo da ih ažuriramo i gde! Zatim popunjavamo sva polja u toj formi vodeći računa o tipu podatka sa kojim radimo i njegovoj konverziji. U našem projektu ćemo se susretati sa *string-ovima*, *datepicker-ima*, *checkbox-ovima* i *combobox-ovima*. Dalje se radi po šablonu kao što možemo da vidimo iz par primera:

```
else if (currentTable == airplaneSelect)
{
    formAirplane tempForm = new formAirplane(update, rowView);
    tempForm.txtRegistration.Text = reader["registration"].ToString();
    tempForm.txtModel.Text = reader["model"].ToString();
    tempForm.dpDateOfManufacture.SelectedDate = (DateTime) reader["dateOfManufacture"];
    tempForm.txtLivery.Text = reader["livery"].ToString();
    tempForm.txtSeatCapacity.Text = reader["seatCapacity"].ToString();
    tempForm.txtRange.Text = reader["range"].ToString();
    tempForm.txtFlightCeiling.Text = reader["flightCeiling"].ToString();
    tempForm.txtMaxSpeed.Text = reader["maxSpeed"].ToString();
    tempForm.txtCruiseSpeed.Text = reader["cruiseSpeed"].ToString();
    tempForm.txtTakeOffWeight.Text = reader["takeOffWeight"].ToString();
    tempForm.txtLandingWeight.Text = reader["landingWeight"].ToString();
    tempForm.txtMinimumRunwayLength.Text = reader["minimumRunwayLength"].ToString();
    tempForm.ShowDialog();
}
```

```

else if (currentTable == flightSelect)
{
    formFlight tempForm = new formFlight(update, rowView);
    tempForm.txtOrigin.Text = reader["origin"].ToString();
    tempForm.txtDestination.Text = reader["destination"].ToString();
    tempForm.txtDuration.Text = reader["duration"].ToString();
    tempForm.txtRoute.Text = reader["route"].ToString();
    tempForm.txtStopovers.Text = reader["stopovers"].ToString();
    tempForm.txtCruiseSpeed.Text = reader["cruiseSpeed"].ToString();
    tempForm.txtCruiseAltitude.Text = reader["cruiseAltitude"].ToString();
    tempForm.cmbFlightOperator.SelectedValue = reader["flightOperatorID"];
    tempForm.cmbGate.SelectedValue = reader["gateID"];
    tempForm.cmbRunway.SelectedValue = reader["runwayID"];
    tempForm.ShowDialog();
}

```

```

else if (currentTable == runwaySelect)
{
    formRunway tempForm = new formRunway(update, rowView);
    tempForm.txtNumber.Text = reader["runwayNumber"].ToString();
    tempForm.txtOrientation.Text = reader["orientation"].ToString();
    tempForm.txtLength.Text = reader["length"].ToString();
    tempForm.txtLoadRating.Text = reader["loadRating"].ToString();
    tempForm.ccbxClosed.IsChecked = (bool)reader["closed"];
    tempForm.ShowDialog();
}

```

Primetićemo da u slučaju *CheckBox-a* konvertujemo taj podatak u “bool”, za *ID* stranih ključeva ne pretvaramo ništa već samo selektovane vrednosti postavljamo onako kako smo ih pročitali. *DatePicker* konvertujemo u *DateTime* objekat.

Što se tiče prikazivanja ostalih tabela, za akciju klika ćemo vezati poziv funkcije “loadData” i proslediti mu odgovarajući *string*.

```

private void btnAirline_Click(object sender, RoutedEventArgs e)
{
    loadData(airlineSelect);
}

```

Ovo se radi šablonski i slično je za svaku tabelu.

Prilikom pritiska na dugme “Dodaj”, želimo da se otvori forma u koju možemo da unosimo podatke, dakle generišemo novu formu odgovarajuće klase i prikazujemo je na ekranu, nakon što se ona zatvori će se pozvati funkcija “loadData” kako bi nam ažurirala tabelu sa novim podacima.

```

Window form;
if (currentTable.Equals(airlineSelect))
{
    form = new formAirline();
    form.ShowDialog();
    loadData(airlineSelect);
}

```

Ovo je takođe šablonski i radi se slično za svaku tabelu.

Što se tiče izmene pratimo sličnu logiku, proveravamo koja tabela je izabrana i pozivamo "fillForm" metodu kako bi nam prikazala formu sa postojećim podacima iz baze. Opet je sve šablonski i nećemo prelaziti kroz svaki slučaj.

```

if (currentTable.Equals(airlineSelect))
{
    fillForm(selectStatementAirline);
    loadData(airlineSelect);
}

```

Ovim smo prešli svu logiku vezanu za glavni prozor, sada ćemo preći kroz slučajeve gde imamo padajuće liste u nekim formama, kako njih popunjavamo i generalnu strukturu ovih formi.

```

Connection conn = new Connection();
SqlConnection connection = new SqlConnection();
private bool update;
private DataRowView rowView;
2 references
public formAirline()
{
    InitializeComponent();
    txtAirlineName.Focus();
    connection = conn.generateConnection();
}

1 reference
public formAirline(bool update, DataRowView rowView) : this()
{
    this.rowView = rowView;
    this.update = update;
}

```

Prvi deo je jednostavan i već smo ga viđali, generišemo konekciju, imamo podatak o izabranom redu i da li je u pitanju ažuriranje podataka. U konstruktoru dodatno stavljamo *focus* na jedno polje kako bi korisnik odmah mogao da krene sa upisivanjem podataka.

```
private void btnSave_Click(object sender, RoutedEventArgs e)
{
    try
    {
        connection.Open();
        SqlCommand cmd = new SqlCommand()
        {
            Connection = connection
        };
        cmd.Parameters.Add("@airlineName", SqlDbType.NVarChar).Value = txtAirlineName.Text;
        cmd.Parameters.Add("@numOfPlanes", SqlDbType.Int).Value = txtNumOfPlanes.Text;
        cmd.Parameters.Add("@location", SqlDbType.NVarChar).Value = txtLocation.Text;
        cmd.Parameters.Add("@airportsServed", SqlDbType.Int).Value = txtAirportServed.Text;
        cmd.Parameters.Add("@flightsOperated", SqlDbType.Int).Value = txtFlightsOperated.Text;

        if (update)
        {
            cmd.Parameters.Add("@id", SqlDbType.Int).Value = rowView["ID"];
            cmd.CommandText = @"update Airline
                                set name = @airlineName, numOfPlanes = @numOfPlanes,
                                location = @location, airportsServed = @airportsServed, flightsOperated = @flightsOperated
                                where airlineID = @id";
            //mozda nije neophodno
            rowView = null;
        }
        else
        {
            cmd.CommandText = @"insert into Airline(name, numOfPlanes, location, airportsServed, flightsOperated)
                                values (@airlineName, @numOfPlanes, @location, @airportsServed, @flightsOperated)";
        }
        cmd.ExecuteNonQuery();
        cmd.Dispose();
        this.Close();
    }
}
```

Ovde vidimo logiku iza dugmeta *Save* odnosno *Sačuvaj*, otvaramo konekciju, dodajemo parametre komande tako što izvlačimo šta je korisnik uneo u polja, i proveravamo da li se radi o ažuriranju ili novom unosu, u odnosu na to pozivamo odgovarajuću komandu. Nakon toga oslobađamo komandu i zatvaramo formu. Naravno ovo se nalazi u *try* bloku, ispod njega su *catch* blokovi, i konačno zatvaranje konekcije u *finally* bloku.

Što se tiče popunjavanja padajućih lista, mi ove podatke moramo da pročitamo iz baze podataka, stoga radimo sledeće:

```
connection.Open();

SqlDataAdapter daFlightOperator = new SqlDataAdapter(getFlightOperators, connection);
DataTable dtFlightOperator = new DataTable();
daFlightOperator.Fill(dtFlightOperator);
cmbFlightOperator.ItemsSource = dtFlightOperator.DefaultView;
daFlightOperator.Dispose();
dtFlightOperator.Dispose();
```

Popunjavamo element *ComboBox-a* sa podacima koje smo pročitali iz baze, važno je napomenuti da se ovde oslanjamo na *DisplayMemberPath* i *SelectedValuePath* koji nam

predstavljaju ono što će se prikazati korisniku, i ono što u pozadini predstavlja strani ključ respektivno. Ovo radimo svugde gde imamo *ComboBox*-ove. Konačno smo prešli većinu koda i sad idemo po formama da objasnimo dodavanje izmenu i brisanje.

6.1. Dodavanje

Priču o dodavanju smo već započeli u prethodnom delu, tako da ćemo da se podsetimo toga kroz par slika:

```
Window form;
if (currentTable.Equals(airlineSelect))
{
    form = new formAirline();
    form.ShowDialog();
    loadData(airlineSelect);
}
```

```
private void btnSave_Click(object sender, RoutedEventArgs e)
{
    try
    {
        connection.Open();
        SqlCommand cmd = new SqlCommand()
        {
            Connection = connection
        };
        cmd.Parameters.Add("@airlineName", SqlDbType.NVarChar).Value = txtAirlineName.Text;
        cmd.Parameters.Add("@numOfPlanes", SqlDbType.Int).Value = txtNumOfPlanes.Text;
        cmd.Parameters.Add("@location", SqlDbType.NVarChar).Value = txtLocation.Text;
        cmd.Parameters.Add("@airportsServed", SqlDbType.Int).Value = txtAirportServed.Text;
        cmd.Parameters.Add("@flightsOperated", SqlDbType.Int).Value = txtFlightsOperated.Text;

        if (update)
        {
            cmd.Parameters.Add("@id", SqlDbType.Int).Value = rowView["ID"];
            cmd.CommandText = @"update Airline
                               set name = @airlineName, numOfPlanes = @numOfPlanes,
                               location = @location, airportsServed = @airportsServed, flightsOperated = @flightsOperated
                               where airlineID = @id";
            //mozda nije neophodno
            rowView = null;
        }
        else
        {
            cmd.CommandText = @"insert into Airline(name, numOfPlanes, location, airportsServed, flightsOperated)
                               values (@airlineName, @numOfPlanes, @location, @airportsServed, @flightsOperated)";
        }
        cmd.ExecuteNonQuery();
        cmd.Dispose();
        this.Close();
    }
}
```

Povlačimo podatke koje je korisnik uneo i dodajemo ih u komandu, tu komandu izvršavamo i zatvaramo konekciju. Veliku pažnju obraćamo kako smo nazivali tabele u bazi, kao i koji tip podatka se koristi, npr. *SqlDataType.Int*, navešćemo još par primera gde imamo malo drugačiju konverziju podataka:


```

DateTime temp = (DateTime)dpDateOfManufacture.SelectedDate;
string date = temp.ToString("yyyy-MM-dd");
cmd.Parameters.Add("@dateOfManufacture", SqlDbType.Date).Value = date;

cmd.Parameters.Add("@runwayNumber", SqlDbType.NVarChar).Value = txtNumber.Text;
cmd.Parameters.Add("@orientation", SqlDbType.NVarChar).Value = txtOrientation.Text;
cmd.Parameters.Add("@length", SqlDbType.NVarChar).Value = txtLength.Text;
cmd.Parameters.Add("@loadRating", SqlDbType.NVarChar).Value = txtLoadRating.Text;
cmd.Parameters.Add("@closed", SqlDbType.Bit).Value = Convert.ToInt32(ccbxClosed.IsChecked);

```

Primetićemo da u slučaju datuma moramo da napravimo objekat klase *DateTime* kako bi mogli da ga unesemo u bazu, a kod *CheckBox-a* vrednost pretvaramo u brojevu vrednost, zato što se u bazi vodi kao 0 i 1, umesto kao *false* i *true*. Sve se ovo odvija u *try* bloku kako bismo moguće greške uhvatili u *catch* bloku. Komandu pišemo po onome što smo naučili u upitima za bazu podataka: *insert*, *update*, *select*, *delete*. Ovim smo završili priču o dodavanju podataka u bazu.

6.2. Izmena

Sledeće na tapeti nam je izmena, i u ove vode smo delimično kročili pa ćemo se podsetiti:

```

if (currentTable.Equals(airlineSelect))
{
    fillForm(selectStatememntAirline);
    loadData(airlineSelect);
}

if (update)
{
    cmd.Parameters.Add("@id", SqlDbType.Int).Value = rowView["ID"];
    cmd.CommandText = @"update Runway
                        set runwayNumber = @runwayNumber, orientation = @orientation,
                        where runwayID = @id";
    //mozda nije neophodno
    rowView = null;
}

```

Ovo je otprilike sve što je novu kod izmene, umesto *insert* pozivamo *update*, umesto nove prazne forme, pravimo novu popunjenu formu onim podacima koje smo selektovali. Logiku iza toga smo već pokrili pa nećemo ići u detalje. Što se tiče uzimanja novih vrednosti, to radi isti deo koda koji i uzima vrednosti za nov unos. Praktično samo proveravamo da li je u pitanju ažuriranje ili nov unos i na osnovu toga pozivamo odgovarajuću komadnu.

6.3. Brisanje

Konačno brisanje!

Za brisanje podataka iz baze vraćamo se u *MainWindow*, proveravamo koja tabela je aktivna, i pozivamo metodu za brisanje koju ćemo uskoro da objasnimo.

```
if (currentTable.Equals(airlineSelect))
{
    deleteEntry(deleteAirline);
    loadData(airlineSelect);
}
```

Sada, ponovo u *try* bloku otvaramo konekciju, proveravamo koji je red izabran, pitamo korisnika da li je siguran da želi da obriše te podatke, i po istom principu kao i za ažuriranje lepimo *string*-ove koje smo već definisali sa ID-om izabranog unosa iz baze. Nakon ovoga imamo par *catch* blokova kako bi obradili moguće greške, poput mogućnosti da korisnik nije izabrao nijedan red, da je došlo do greške u bazi, u našoj aplikaciji itd. Konačno u *finally* bloku zatvaramo konekciju.

```
try
{
    connection.Open();
    rowView = (DataRowView) dataGridCenter.SelectedItems[0];
    MessageBoxResult result = MessageBox.Show("Da li želite da obrisite");
    if (result == MessageBoxResult.Yes)
    {
        SqlCommand cmd = new SqlCommand { Connection = connection };
        cmd.Parameters.Add("@id", SqlDbType.Int).Value = rowView["ID"];
        cmd.CommandText = deleteCommand + "@id";
        cmd.ExecuteNonQuery();
        cmd.Dispose();
    }
}
```

```
catch (ArgumentOutOfRangeException ex)
{
    MessageBox.Show("Nije odabran nijedan red iz tabele!\n\n" + ex.Message, "Greška", MessageBoxButton.OK);
}
catch (SQLException ex)
{
    MessageBox.Show("Podaci se nalaze u drugoj tabeli i ne mogu biti obrisani!\n\n" + ex.Message, "Greška", MessageBoxButton.OK);
}
catch (Exception ex)
{
    MessageBox.Show("Došlo je do greške u toku obrade!\n\n" + ex.Message, "Greška", MessageBoxButton.OK);
}
finally { if (connection != null) connection.Close(); }
```

6.4. Izgled aplikacije

Konačno dolazimo do izgleda same aplikacije, to smo već opisali kad smo pričali o kreiranju formi, ali sada nas interesuje širina i visina pojedinih elemenata, što ostavljam vama na ukus!

U nastavku ćemo videti primere izgleda gotove aplikacije (doduše dosta ružne).

MainWindow				
Aviokompanije	Avioni	Letovi	Operateri leta	Parovi: karta - let
Gejt	Putnici	Piloti	Piste	Karte

ID	Ime	Prezime	Starost	Kontakt	Adresa	Licenca
1	Marko	Pavlović	32	marko.pavlo@airserbia.rs	Ustanička 68, Beograd	CL
2	Paul	Haunders	58	paulhaunders@dublinairport.ie	7th Street 45, Dublin	CL
5	Maks	Kowalczyk	1000	kowalskianalysis@lotarilines.pl	Pabianicka 28, Lodz	NL

Dodaj

Izmeni

Obriši

formPilot

Ime:

Prezime:

Starost:

Kontakt:

Adresa:

Pilotska licenca:

Save

Cancel

formPilot

Ime:

Marko

Prezime:

Pavlović

Starost:

32

Kontakt:

marko.pavlo@airserbia.rs

Adresa:

Ustanička 68, Beograd


Pilotska licenca:

CL

Save

Cancel

Upozorenje

 Da li zelite da obrisite ove podatke?

Yes

No

7. Testiranje aplikacije

Na kraju nam ostaje da testiramo to što smo napravili, prvo ćemo da dodamo novi unos u tabelu pilota!

MainWindow

Aviokompanije	Avioni	Letovi	Operateri leta	Parovi: karta - let
Gejt	Putnici	Piloti	Piste	Karte

ID	Ime	Prezime	Starost	Kontakt	Adresa	Licenca
1	Marko	Pavlović	32	marko.pavlo@airserbia.rs	Ustanička 68, Beograd	CL
2	Paul	Haunders	58	paulhaunders@dublinairport.ie	7th Street 45, Dublin	CL
5	Maks	Kowalczyk	1000	kowalskianalysis@lotarilines.pl	Pabianicka 28, Lodz	NL

Dodaj Izmeni Obriši

MainWindow

Aviokompanije	Avioni	Letovi	Operateri leta	Parovi: karta - let
Gejt	Putnici	Piloti	Piste	Karte

ID	Ime	Prezime	Starost	Kontakt	Adresa	Licenca
1	Marko	Pavlović	32	marko.pavlo@airserbia.rs	Ustanička 68, Beograd	CL
2	Paul	Haunders	58	paulhaunders@dublinairport.ie	7th Street 45, Dublin	CL
5	Maks	Kowalczyk	1000	kowalskianalysis@lotarilines.pl	Pabianicka 28, Lodz	NL
7	Luka	Miljanović	19	miljanovic.it11.2022@uns.ac.rs	Gavrila Principa 35	AU

Dodaj Izmeni Obriši

Zatim izmena već postojećeg unosa:

formPilot

Ime: Mariusz

Prezime: Kowalczyk

Starost: 22

Kontakt: kowalskianalysis@lotarilin

Adresa: Pabianicka 28, Warsaw

Pilotska licenca: GL

Save Cancel

MainWindow

Aviokompanije

Avioni

Letovi

Operateri leta

Parovi: karta - let

Gejt

Putnici

Piloti

Piste

Karte

ID	Ime	Prezime	Starost	Kontakt	Adresa	Licenca
1	Marko	Pavlović	32	marko.pavlo@airserbia.rs	Ustanička 68, Beograd	CL
2	Paul	Haunders	58	paulhaunders@dublinairport.ie	7th Street 45, Dublin	CL
5	Mariusz	Kowalczyk	22	kowalskianalysis@lotarilines.pl	Pabianicka 28, Warsaw	GL
7	Luka	Miljanović	19	miljanovic.it11.2022@uns.ac.rs	Gavrila Principa 35	AU

Dodaj

Izmeni

Obriši

I na kraju brisanje!

MainWindow

Aviokompanije

Avioni

Letovi

Operateri leta

Parovi: karta - let

Gejt

Putnici

Piloti

Piste

Karte

ID	Ime	Prezime	Starost	Kontakt	Adresa	Licenca
1	Marko	Pavlović	32	marko.pavlo@airserbia.rs	Ustanička 68, Beograd	CL
2	Paul	Haunders	58	paulhaunders@dublinairport.ie	7th Street 45, Dublin	CL
5	Mariusz	Kowalczyk	22	kowalskianalysis@lotarilines.pl	Pabianicka 28, Warsaw	GL
7	Luka	Miljanović	19	miljanovic.it11.2022@uns.ac.rs	Gavrila Principa 35	AU

Dodaj

Izmeni

Obriši

Upozorenje



Da li zelite da obriste ove podatke?

Yes

No

Greška

Podaci se nalaze u drugoj tabeli i ne mogu biti obrisani

The DELETE statement conflicted with the REFERENCE constraint "FK_FlightOperator_Pilot". The conflict occurred in database "Airport", table "dbo.FlightOperator", column 'pilotID'.

The statement has been terminated.

OK

MainWindow

Aviokompanije	Avioni	Letovi	Operateri leta	Parovi: karta - let
Gejt	Putnici	Piloti	Piste	Karte

ID	Ime	Prezime	Starost	Kontakt	Adresa	Licenca
1	Marko	Pavlović	32	marko.pavlo@airserbia.rs	Ustanička 68, Beograd	CL
2	Paul	Haunders	58	paulhaunders@dublinairport.ie	7th Street 45, Dublin	CL
5	Mariusz	Kowalczyk	22	kowalskianalysis@lotarilines.pl	Pabianicka 28, Warsaw	GL

Dodaj

Izmeni

Obriši

Primetićemo da podatke nismo mogli da obrišemo ako se one koriste u nekoj drugoj tabeli! To smo spominjali prilikom *delete* upita u SQL bazi.

8. Zaključak

Ovim završavamo dokumentaciju i ovaj projekat. Uspešno smo kreirali imaginarni sistem, nacrtali sve slučajeve, i onda prešli u realizaciju! Kreirali smo bazu podataka, naučili kako da sa njom rukujemo, a zatim i *frontend* aplikaciju. Ovo smo postigli koristeći SQL Server Management Studio, kao i Microsoft Visual Studio. Koristeći C# jezik kao i XAML strukturu naša aplikacija može da interaktuje sa bazom, unosi, menja i briše podatke iz nje.