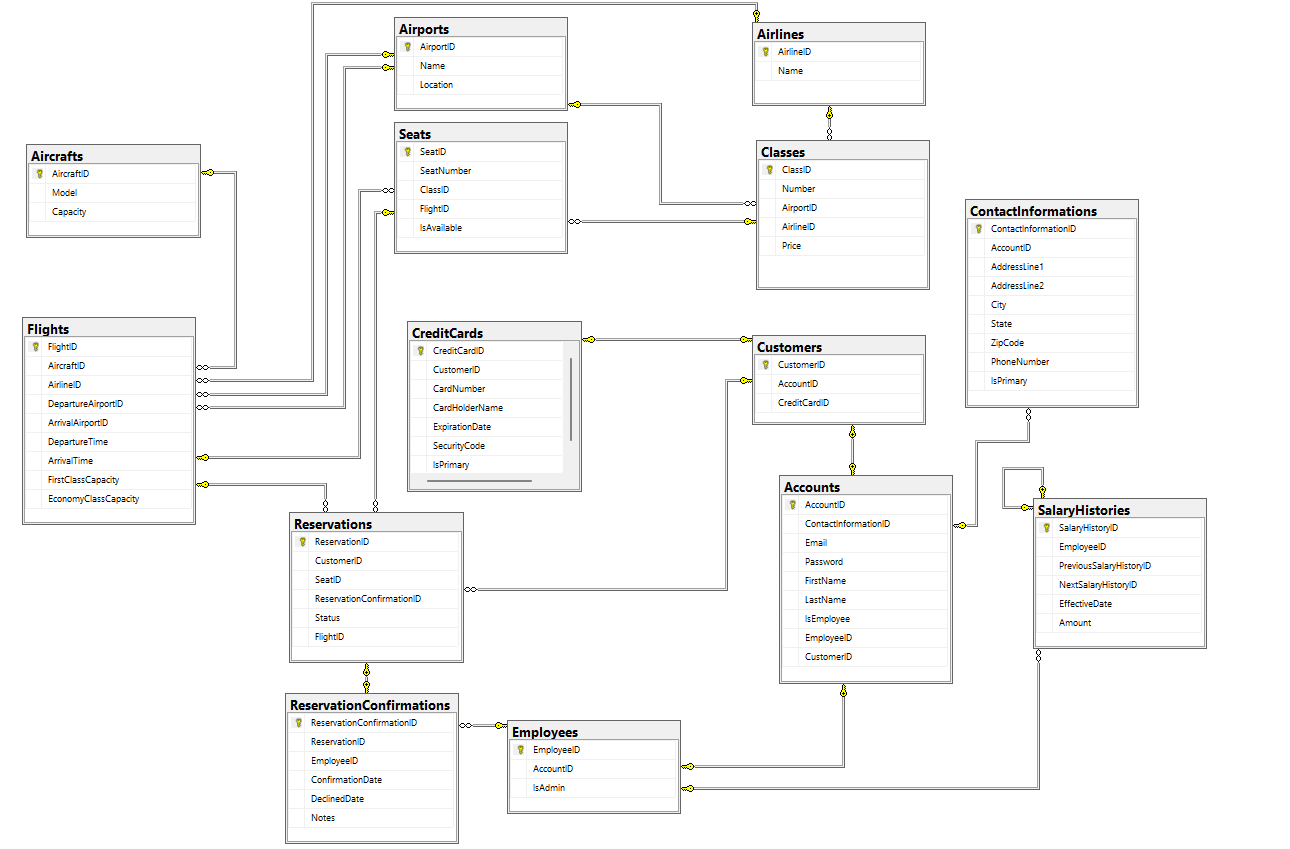
Aplicatie de rezervare bilete de avion

Tema proiectului meu este o aplicatie de rezervat bilete de avion. Aceasta nu a apucat sa ajunga la un stadiu final, aceasta fiind destul de complexa la baza. In esenta, o sa discut despre fiecare aspect al proiectului pentru a putea intelege mai bine cum functioneaza.

Aplicatia utilizeaza structura MVC (Model-View-Controller), la care a fost adaugat un repository pentru a definii mai bine lucrul cu baza de date.

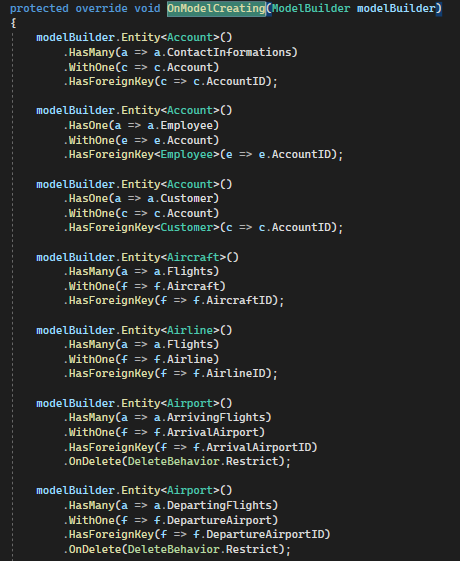
Programul are la baza 14 tabele legate intre ele cu ajutorul Foreign key-urilor. In principiu, cea mai complexa parte a aplicatiei a fost sa fac un design la baza de date sim ai apoi sa reusesc sa creez modelele si contextul. Fiind multe legaturi, acest proces a durat in mare aproape 2 saptamani, pentru ca de cate ori se incerca ceva, eram nevoit sa sterg migrarea, si am incercat mai multe abordari, dar cea mai buna a ramas metoda OnModelCreating unde am reusit sa definesc legaturile asa cum ar trebui.

Pentru inceput, voi prezenta diagrama bazei de date iar mai apoi voi prezenta si modul de legatura conceput in metoda OnModelCreating.



Dupa cum putem observa, structura este in complete legatura cu aproape toate tabelele, insemnand ca din aproape orice tabel se poate ajunge la un alt tabel (Lucru realizabil daca exista deja o date in tabelele respective).

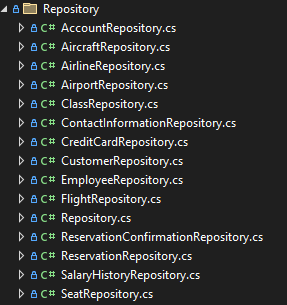
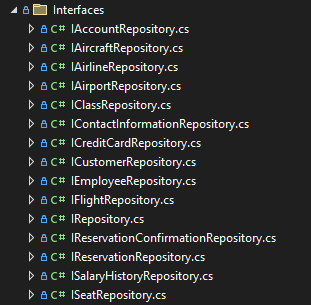
In functia OnModelCreating a pot spune ca mi-am pierdut o mare parte din timp, deoarece aici, daca nu se construiesc bine legaturile pot exista consecinte ulterior, iar una din cele mai complicate parti a fost regandirea bazei de date.





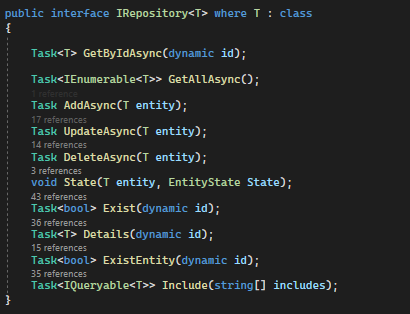
De ce am spus ca a devenit mult mai grea partea aceasta? Ei bine, la fiecare legatura dintre tabele, definita in modul meu, mai aparea cate o eroare ca tabele ar putea sa cicleze, si cel mai greu a fost sa imi dau seama ca este necesar defapt sa adaug o restrictie la OnDelete, si in unele cazuri sa anulez chiar legaturile dintre unele tabele.

In aplicatie se regasesc momentan 15 controllere care care sunt folosite in mare parte, voi trece si o sa discut despre implementarea controllerelor dupa ce o sa prezint interfetele si repository-urile.

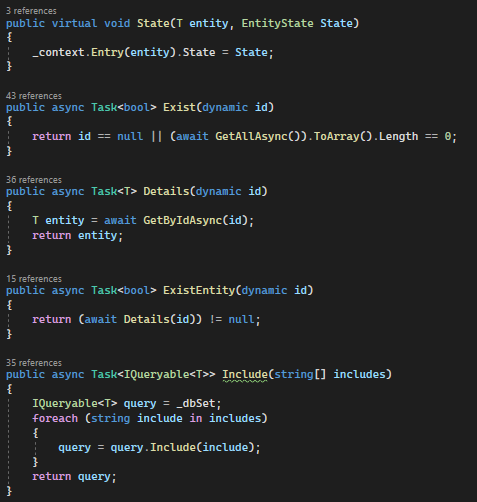
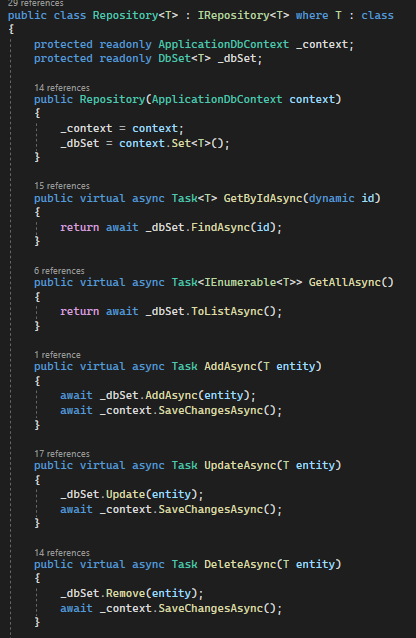


Interfete, sunt destul de multe, fiecare dintre ele implementand metodele unui Repository, iar la capitolul Repository, am incercat o abordare de mostenire, pentru a anula o mare parte din duplicarea unor sectiuni de cod si pentru a optimiza aplicatia in primul rand.

Interfata de baza este IRepository care implementeaza urmatoarele metode:



Implementarea regasita in repository o sa fie justificata ca fiind ideala ea anuland repetarea multor sectiuni de cod, astfel ca, Repository-ul de baza este conceput in modul urmator:

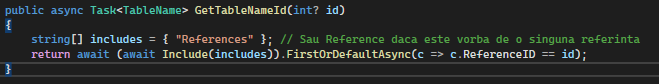


Dupa cum se poate observa, prin intermediul Parametrului T se trimite Modelul utilizat de catre Repository, mai apoi este instantiat DbSet-ul pentru a putea prelua datele necesare.

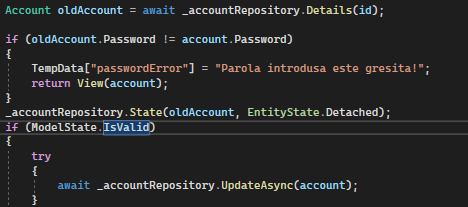
Asa cum am precizat anterior, aceasta metoda elimina multe portiuni de cod si permite programatorului sa utilizeze metodele ce nu sunt asyncrone sa devina asyncrone, adaugand in cadrul lor si functia de SaveChangesAsync pentru a salva datele.

Unele dintre metodele cele mai bune in pentru anumite task-uri sunt functia de Include, care primeste un string si poate returna fara probleme, DbSet-ul cu datele de legatura de tipul ICollection<T> ori T. In acest caz discutam despre datele regasite in model, pentru un tabel de legatura.

Sa luam ipoteza ca intr-un tabel avem ReferenceID, care face legatura cu tabelul/modelul Reference, dar in realitate ne dorim si datele din Reference la preluare. Prin intermediul functiei Include, doar specificam:



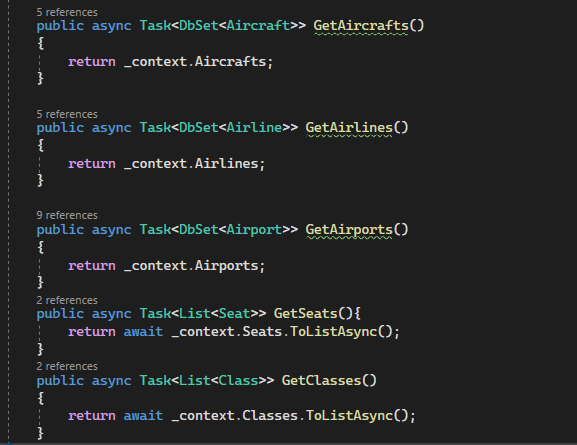
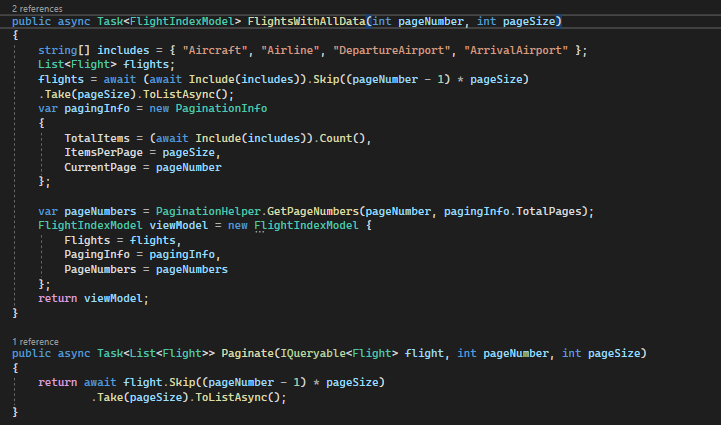
Inca o functie care a reusit sa ma scape de o problema a fost functia State, care are rolul de a schimba State-ul unei entitati, in asa fel incat sa se pot executa alte operatii in aplicatie.

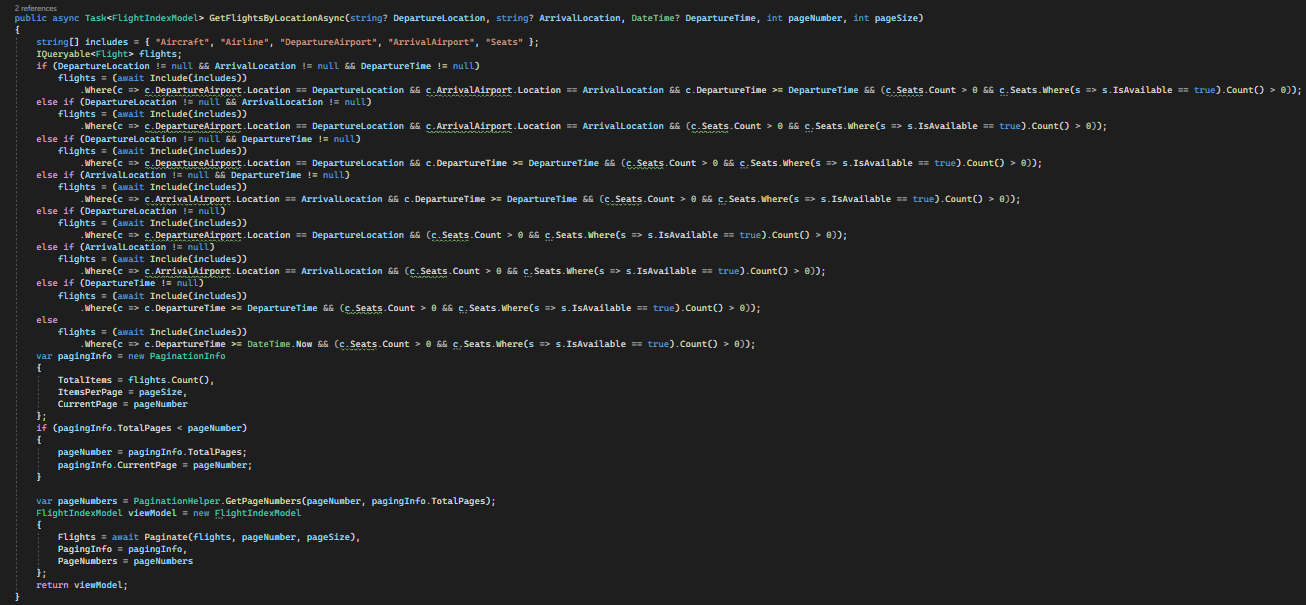


Mai sus este prezentat un mod de abordare a metodei State, in cazul acesta , oldAccount ramanea activ si nu mai permitea sa se execute alte operatii.

In continuare voi prezenta implementarea repository-ului Flight, acesta fiind unul dintre cele mai complexe, el implementand si o metoda de paginare dar si de filtrare a datelor.







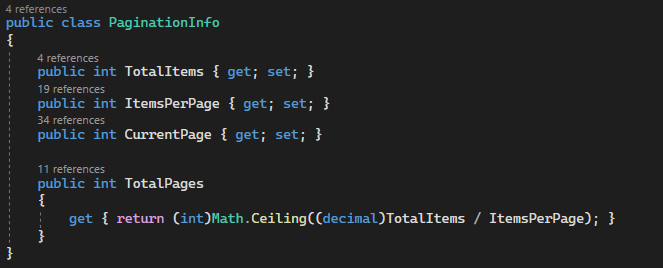
Dupa cum putem observa, avem 3 functii care se utilizeaza pentru operatiile de CRUD, toate 3 utilizand metodele definite in Repository-ul de baza al aplicatiei.

Asa cum am precizat anterior, se poate observa cum functioneaza pe un caz real metoda Include, aceasta avand in continuare acces la metodele din DbSet(Where, Skip, Take).

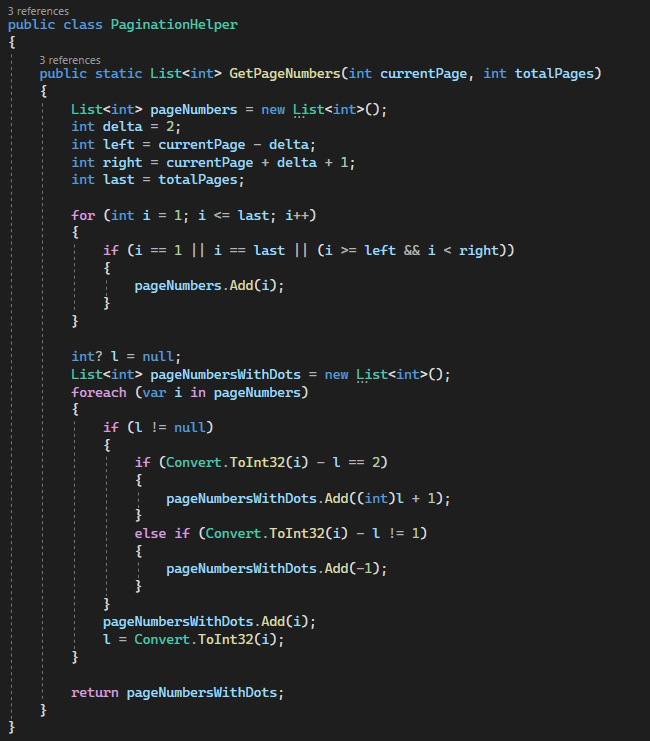
Utilizand contextul in cadrul fiecarui repository, este mult mai simplu sa preluam datele de care avem nevoie ulterior in aplicatie.

Functia GetFlightsByLocationAsync, este o functie de cautare, bazata pe 3 parametrii de cautare, cum ar fii aeroportul de plecare, aeroportul de destinatie si data plecarii.

Pentru a reusi sa pun in practica paginarea in asa fel incat sa imi permita sa utilizez si un numar de elemente pe pagina, am generat doua clase.



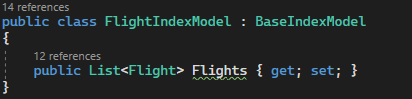
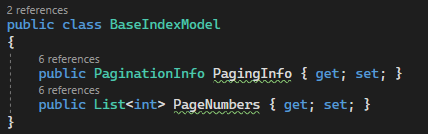
PagingInfo memoreaza numarul total de record-uri care se regasesc in lista, numarul de record-uri pe pagina si pagina curenta, dar si numarul total de pagini, care converteste la urmatorul numar daca rezultatul este cu virgula, in acest caz Math.Ceiling(4.2) = 5.



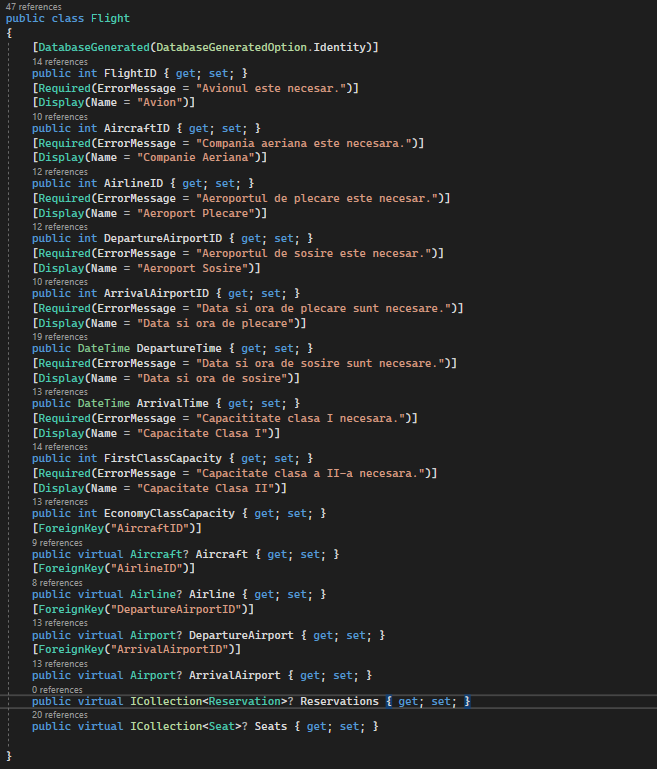
Clasa PaginationHelper imi permite sa afisez ce pagini ar trebui sa fie vizibile in aplicatie, pentru a obtine un efect de genul acesta.



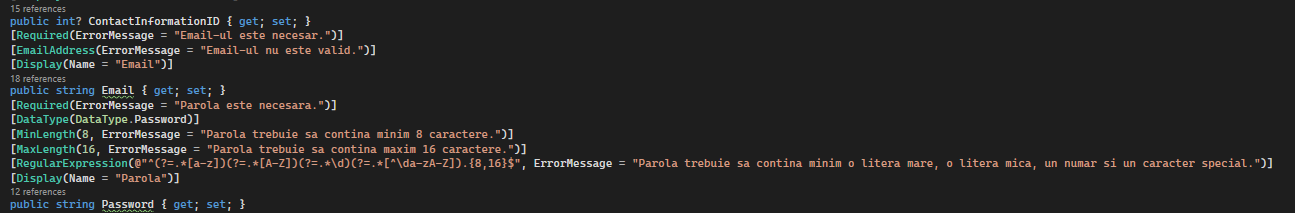
In cod mai apare generat si un model care este trimis mai apoi ca parametru, acest model este generat pentru paginile care necesita paginare, astfel ca un modelul este format tot pe baza mostenirii, unde avem un BaseIndexModel care retine toate datele despre paginare, si un ControllerIndexModel care mosteneste BaseIndexModel si implementeaza o lista a modelului utilizat in controller-ul respectiv.



Pana acum am tot vazut ca utilizez modelul Flight prin aplicatie, dar nu am vazut si modelul in sine, mai jos, voi prezenta cum arata clasa Flight, erorile pe care le implementeaza si module de afisare a acestor date in aplicatie.

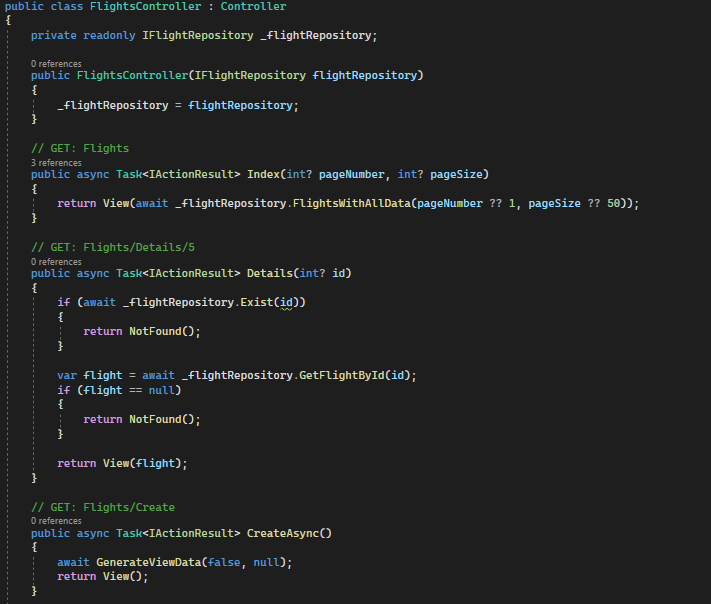


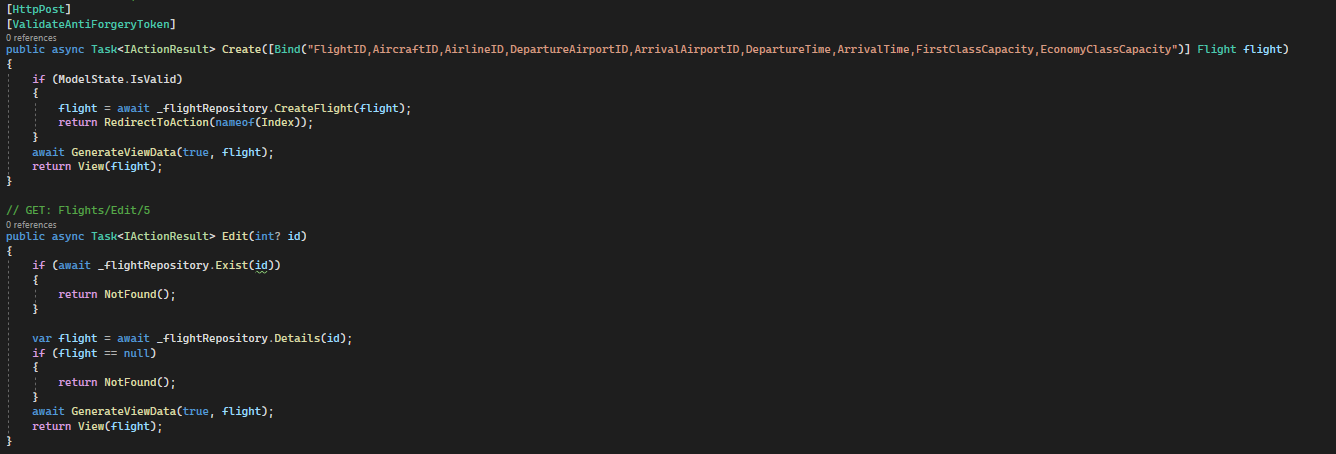
Se poate observa ca mesajele de eroare nu sunt foarte complexe, insa exista si metode de perfectionare a acestora, in urmatoarea poza, voi prezenta mesaje de validare din modelul Account.

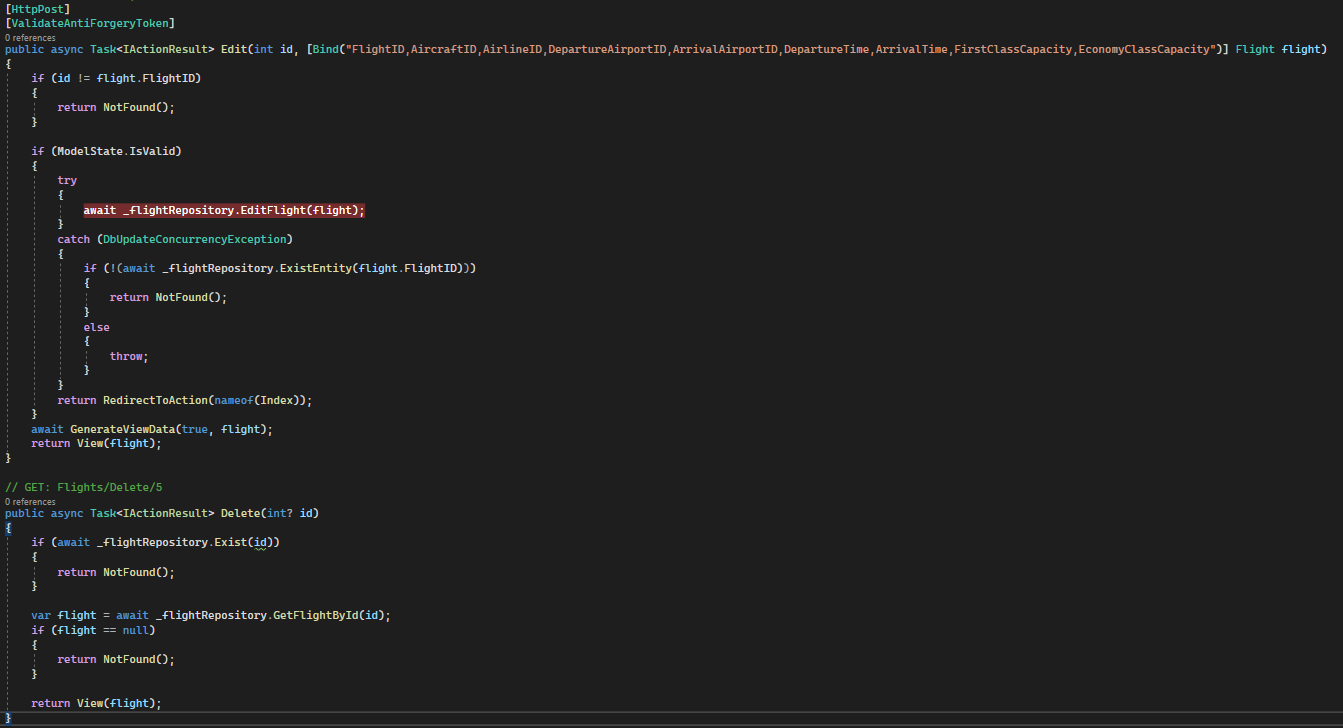


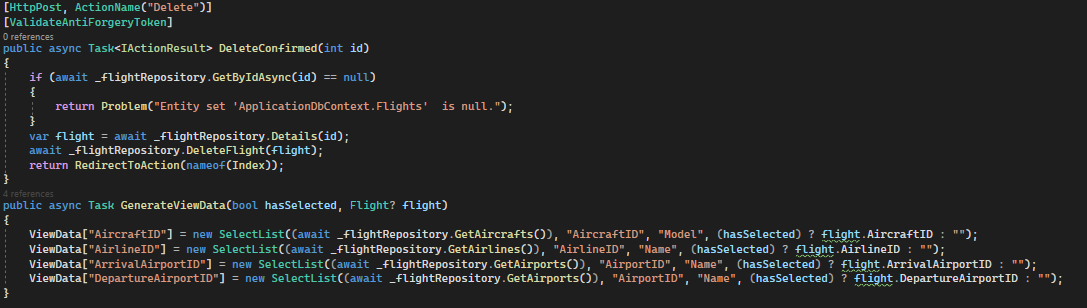
Se poate observa ca in cadrul acestor modele, se pot preciza tirpurile datelor si mai multe cerinte ce trebuie indeplinite, cum un numar limita de caractere, sau textul sa fie testat pe baza unui regex (Regular Expression).

In continuare voi prezenta implementarea unui controller, aceasta implementare se reflecta in mare parte si in restul controller-elor.









Fiind de cele mai multe ori necesara implementarea unor date in View, s-a dezvoltat o functie special conceputa pentru acest aspect in asa fel incat sa nu fie necesara o rescriere a codului de fiecare data.

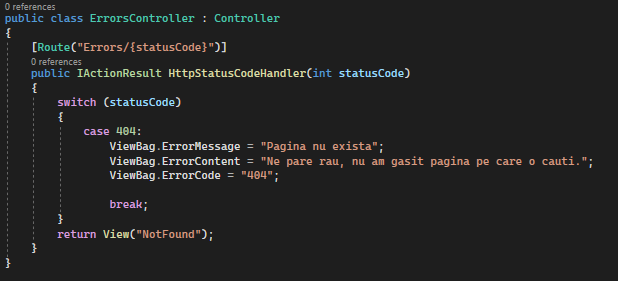
Controllerele implementeaza repository-urile aferente pentru fiecare si le utilizeaza in toate cele 4 operatii de CRUD.

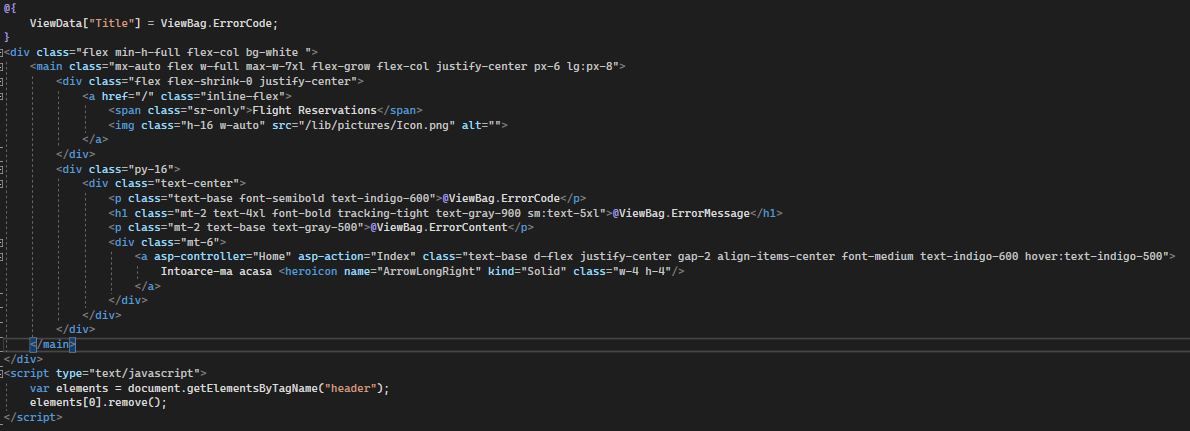
Pentru a reusi sa utilizez repository-urile in aplicatie a fost necesara includerea acestora in Program.cs prin intermediul urmatoarei sintaxe.



Tot in acest fisier, s-a creat si o ruta pentru o pagina de eroare Custom in cazul erorii 404.





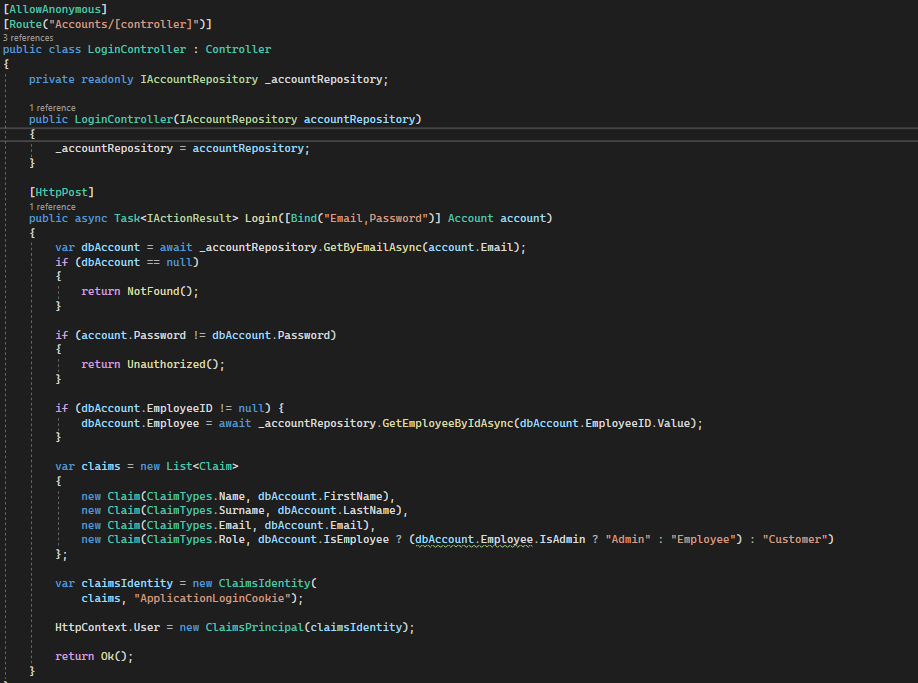


Pagina de eroare arata astfel in acest moment.



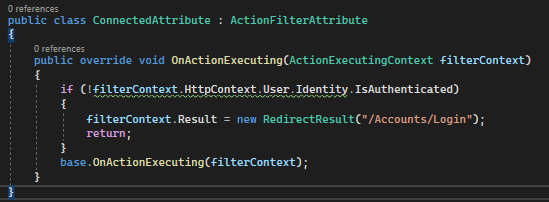
In prezent, s-a incercat autentificarea in aplicatie pe baza unui Identity. Problema este faptul ca desi in controller-ul de logare se seteaza datele in HttpContext, cand urmeaza sa consum acele date in Atributul de conectare, ma instiinteaza ca respectivul Identity nu are date.

Aici se afla implementarea pentru LoginController



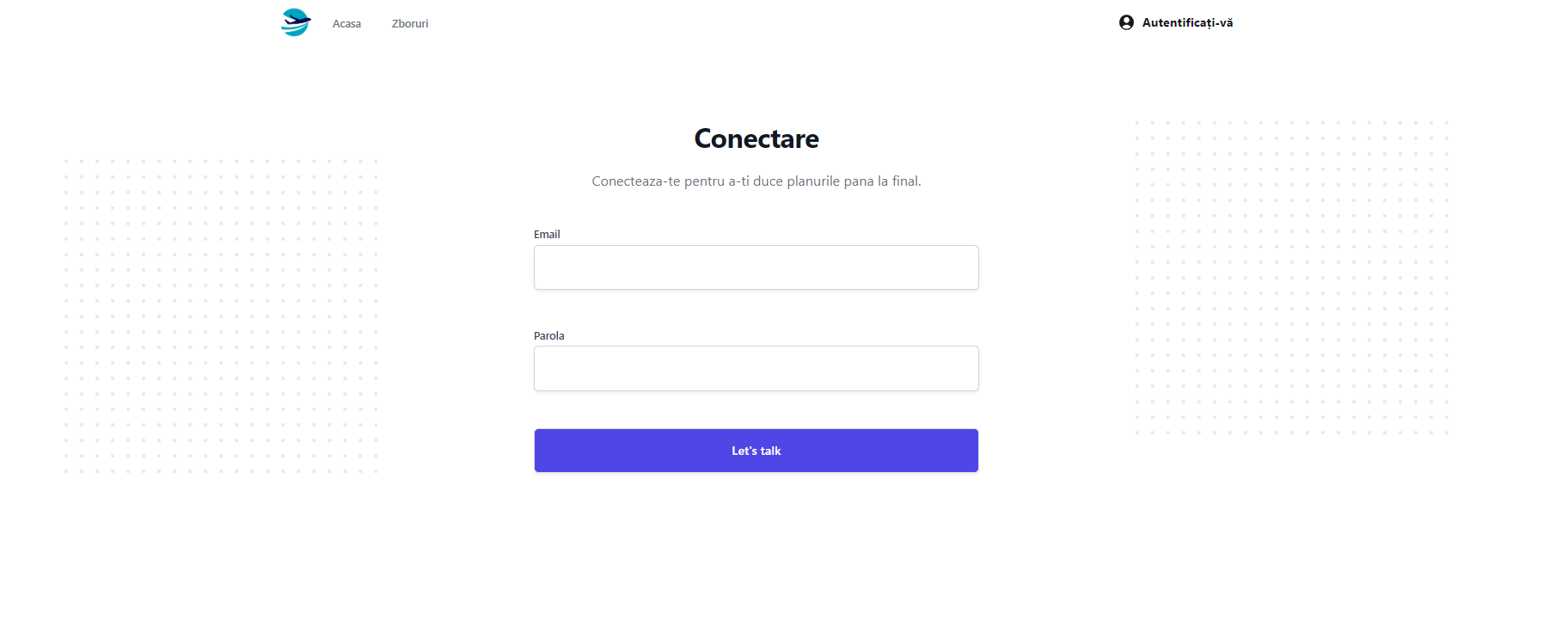
Dupa cum se poate observa se verifica daca datele sunt introduse corect inainte, bazat pe contul extras din database, se adauga toate claim-urile, inclusiv un Claim extrem de important din punctul meu de vedere, acesta urma sa fie utilizat in crearea unui Atribut de verificare, pentru a stii mult mai usor ce drepturi au utilizatorii.

In punctul de return Ok(), HttpContext-ul are datele incluse, in momentul in care eu le accesez in Atributul Connected, datele nu mai sunt salvate.

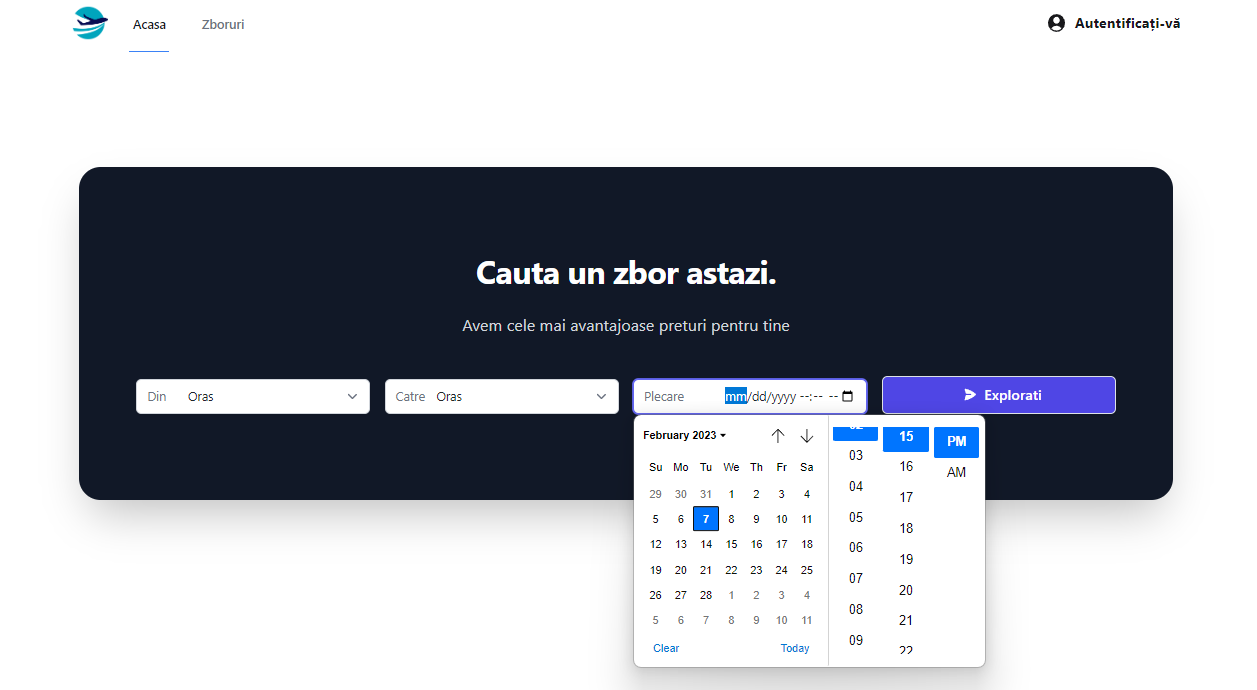


As putea spune ca in acest punct am incercat orice, am utilizat si metoda SignInAsync, dar pare ca aceasta nu salva datele nici dupa finalizare.

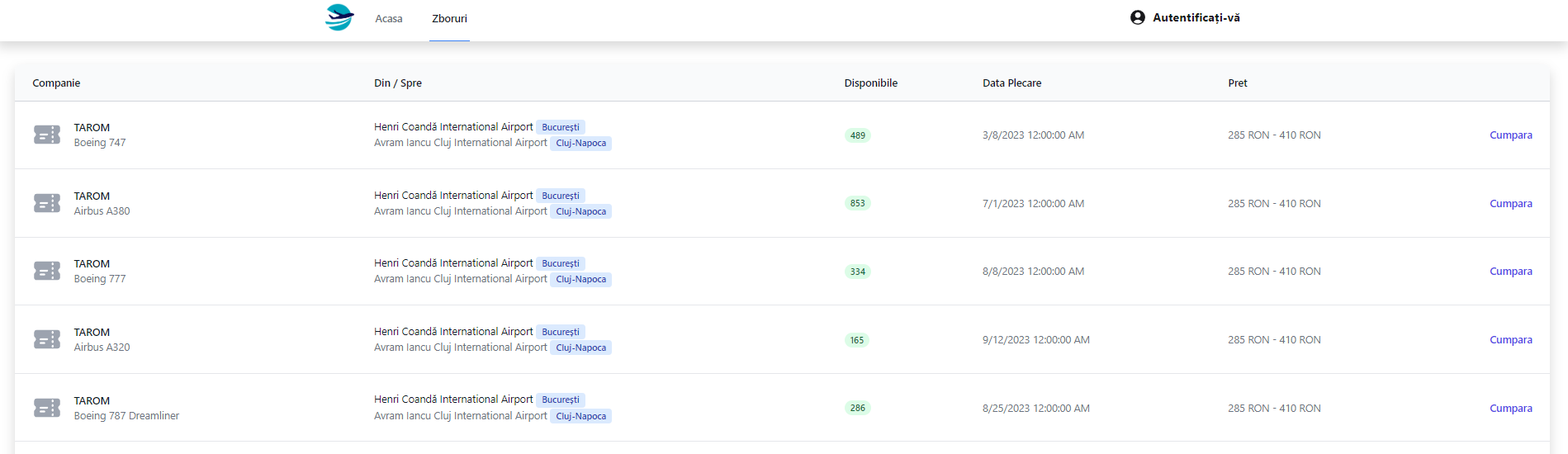
Pagina de Logare

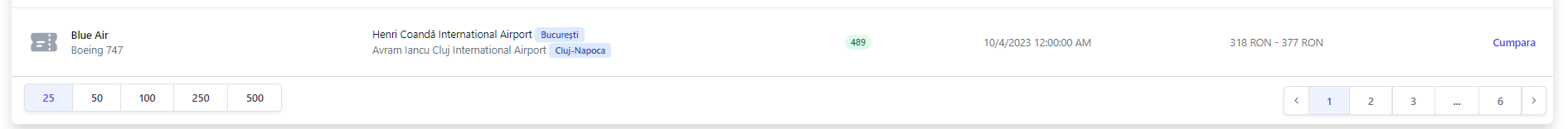


Pagina de Acasa



Pagina de Zboruri





In punctul acesta, aproape toata logica din spate este facuta, daca se creeaza o rezervare, automat un loc se va ocupa si nu va mai aparea de exemplu 489 si o sa apara 488, in cazul in care rezervarea era stearsa, locul era din nou liber iar in cazul in care confirmarea rezervarii era anulata, locul se trecea iar pe disponibil.

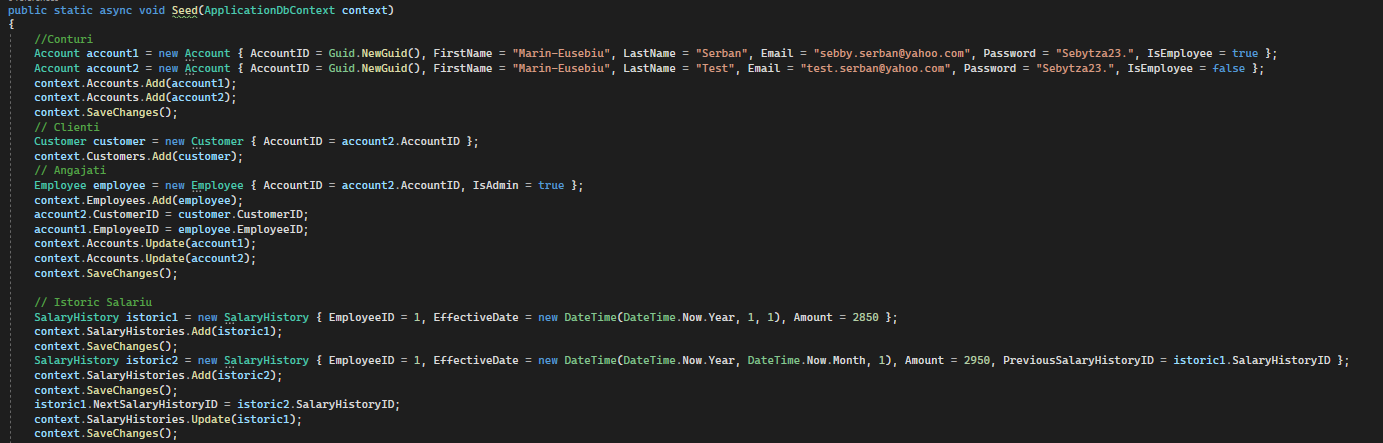
In mod normal logica urmatoare nu mai era complicata, mai aveam doar de implementat pagina de setari utilizator, unde acesta trebuia sa isi treaca datele de contact si un card bancar. Si dupa aceea mai erau de implementat 3 aspecte, pagina de istoric rezervari, pagina de status rezervare si pagina de angajat unde se vedeau rezervarile facute cu posibilitatea de a le accepta sau respinge.

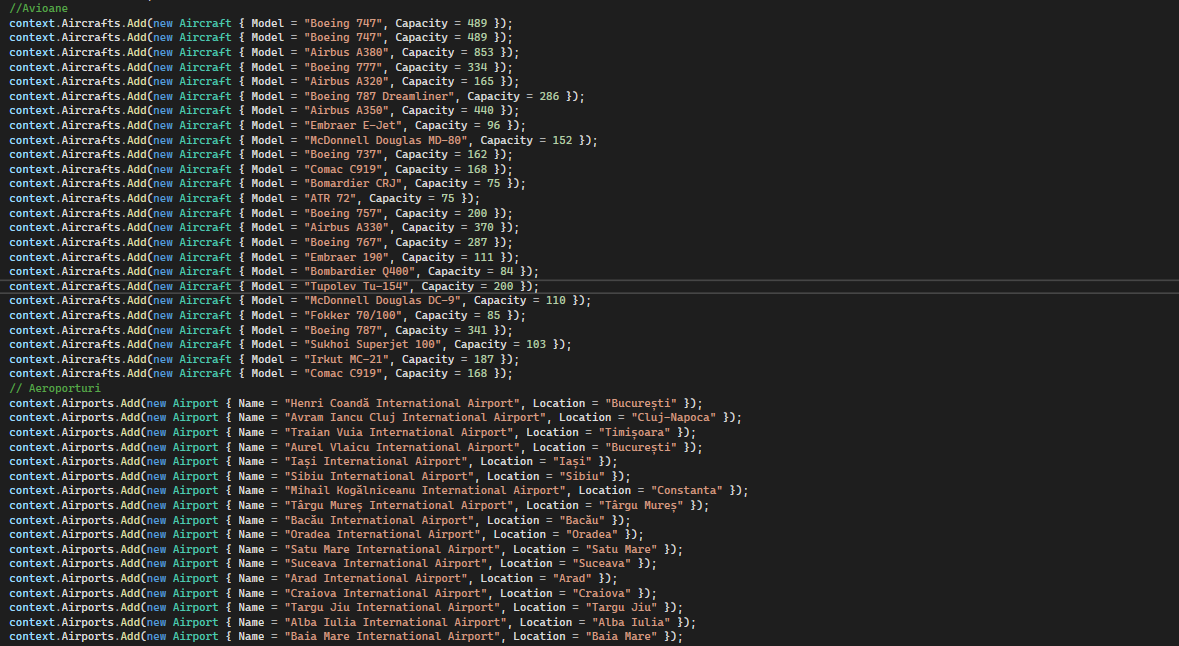
Intr-un final mai era de implementat logica ca nu se poate rezerva un bilet daca nu exista date de contact si un card in cont. Care acestea din urma nu erau complicate de realizat.

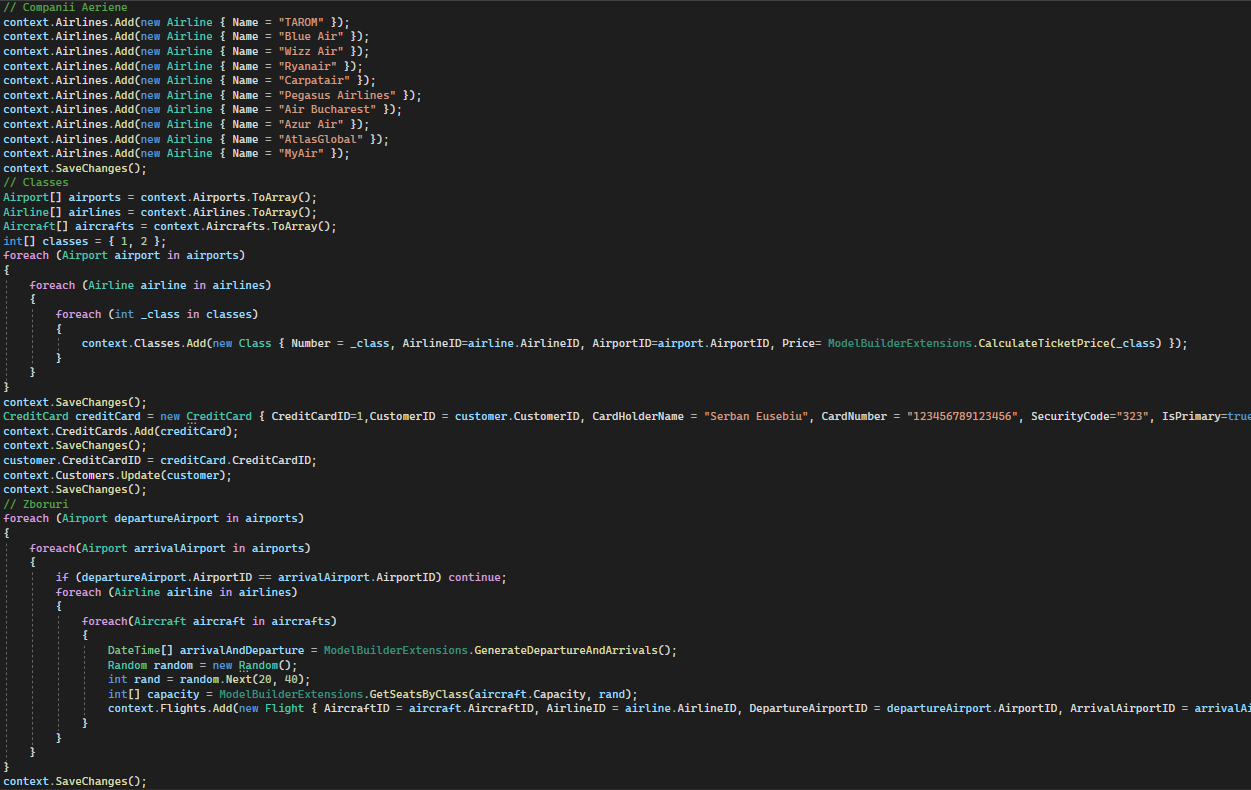
Insa, pagina de Admin este functionala (partea care implementeaza operatiile CRUD).

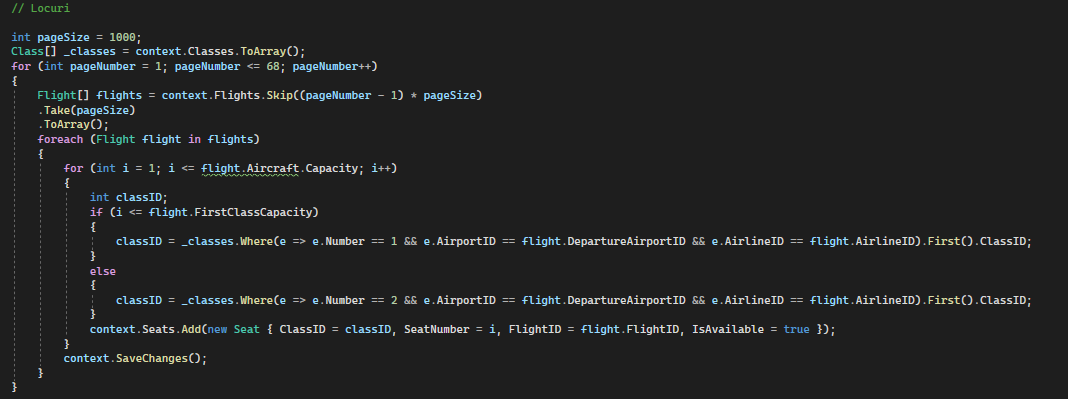
Poate daca exista mai mult timp si sa nu ma lovesc de problema aceea de conectare care a mancat extrem de mult timp, poate proiectul era terminat in ultima instanta.

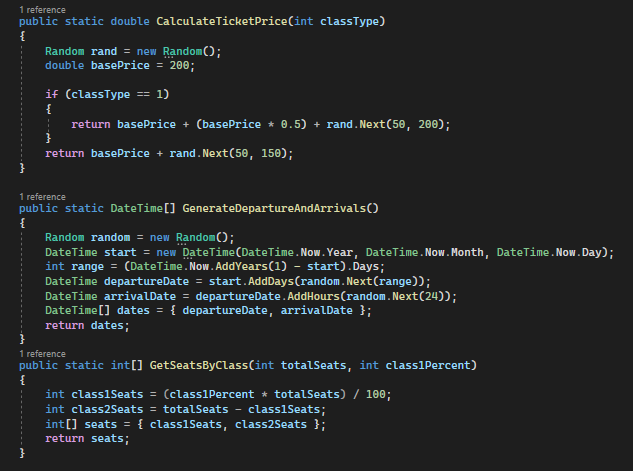
In final, am reusit sa fac o functie de Seed, nu foarte perfecta, dar pentru necesarul meu, a fost suficient.











Cele mai importante functii aici sunt cele de calculare pret in functie de clasa, metoda de generare date de plecare si sosire metoda de impartire a locurilor din avion, dupa ipoteza ca fiecare companie aeriana pune cum doreste numarul de locuri la clasa I, si plecand dupa ipoteza ca sunt mai putine locuri la clasa I decat la clasa a II.

Daca rulati scriptul, v-as ruga ori sa mai stergeti din Avioane,Aeroporturi si Companii Aeriene, sau sa anulati functia de locuri. Probabil va intrebati de ce? (o sa genereze in jur de 143.480.560.000 de record-uri, un echivalent a 4TB de date, plus ca am aflat cand am ajuns pe le 500k de record-uri cat de greu merg query-urile dupa).

Pe final, relatia dintre utilizator si aplicatie, trebuia sa fie una minimalista, sa aduca in genul unui Magazin online si sa ii permita utilizatorului sa faca ceea ce isi propune, adica sa rezerve bilete de avion.

Acesta a fost proiectul meu, sper ca, desi nu este complet, sa va intrige modul meu de abordare al implementarii.