Online Skincare Shop

Proiect realizat de Alexandrescu Andra si Nitu Teodora

Link catre proiect: <https://github.com/andialexandrescu/Skincare_Online_Shop_.NET> (varianta de Windows)

<https://github.com/tteeeoooo/proiect2> (varianta de Mac)

# Etapa 1 - 07 – 14 Noiembrie

## Sprint 1 (saptamanile 6, 7)

Overall:

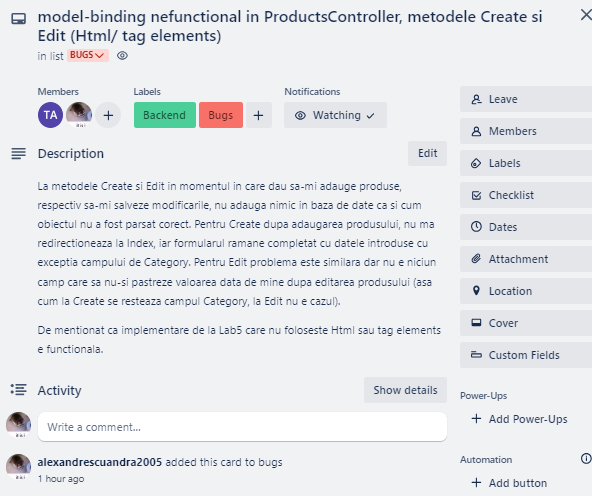
Inceputul proiectului a pornit de la trei tabele din diagrama asociata aplicatiei (Products, Categories, Reviews, 2 relatii one-to-many configurate la nivel de ApplicationDbContext), impreuna cu views-urile pentru Products si Categories.

Etape imediat urmatoare:

De terminat CRUD-ul pentru Categories si Reviews (controller + views).

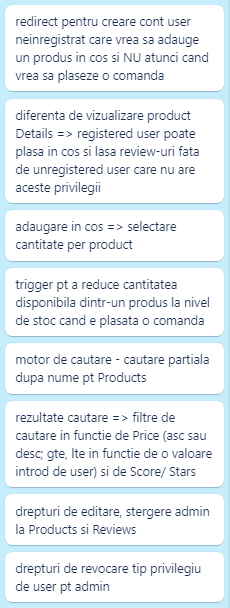
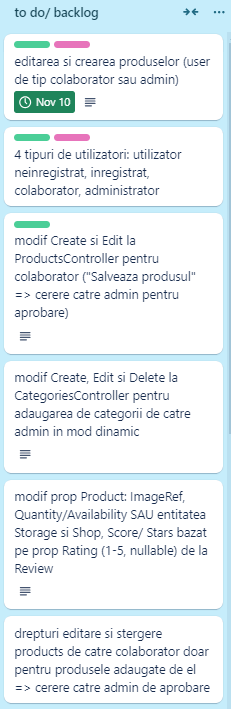
Probleme intalnite/ Bugs:

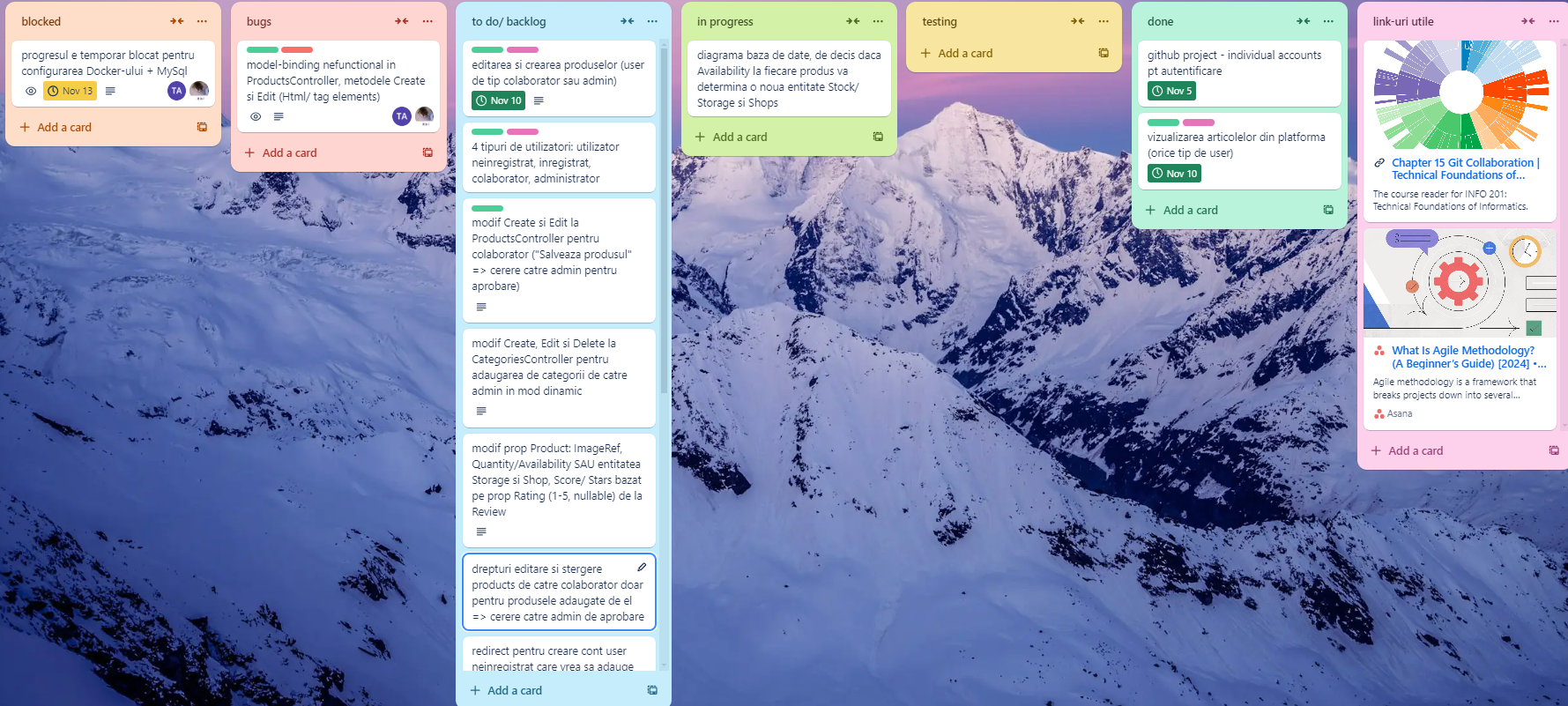
[X] Merge conflicts rezolvate din cauza diferentelor de configuratii de proiect pentru Mac/ Windows => vom incerca folosind Docker (sapt 7)

[X] Model-binding nefunctional pentru ambii membri ai echipei

[X] Schimbarea unor proprietati la nivel de Products (Andi <- Teo: Name <- Description, Description <- Content, (+)Brand, (+)Ingredients) => modificare ProductsController, Products’ views

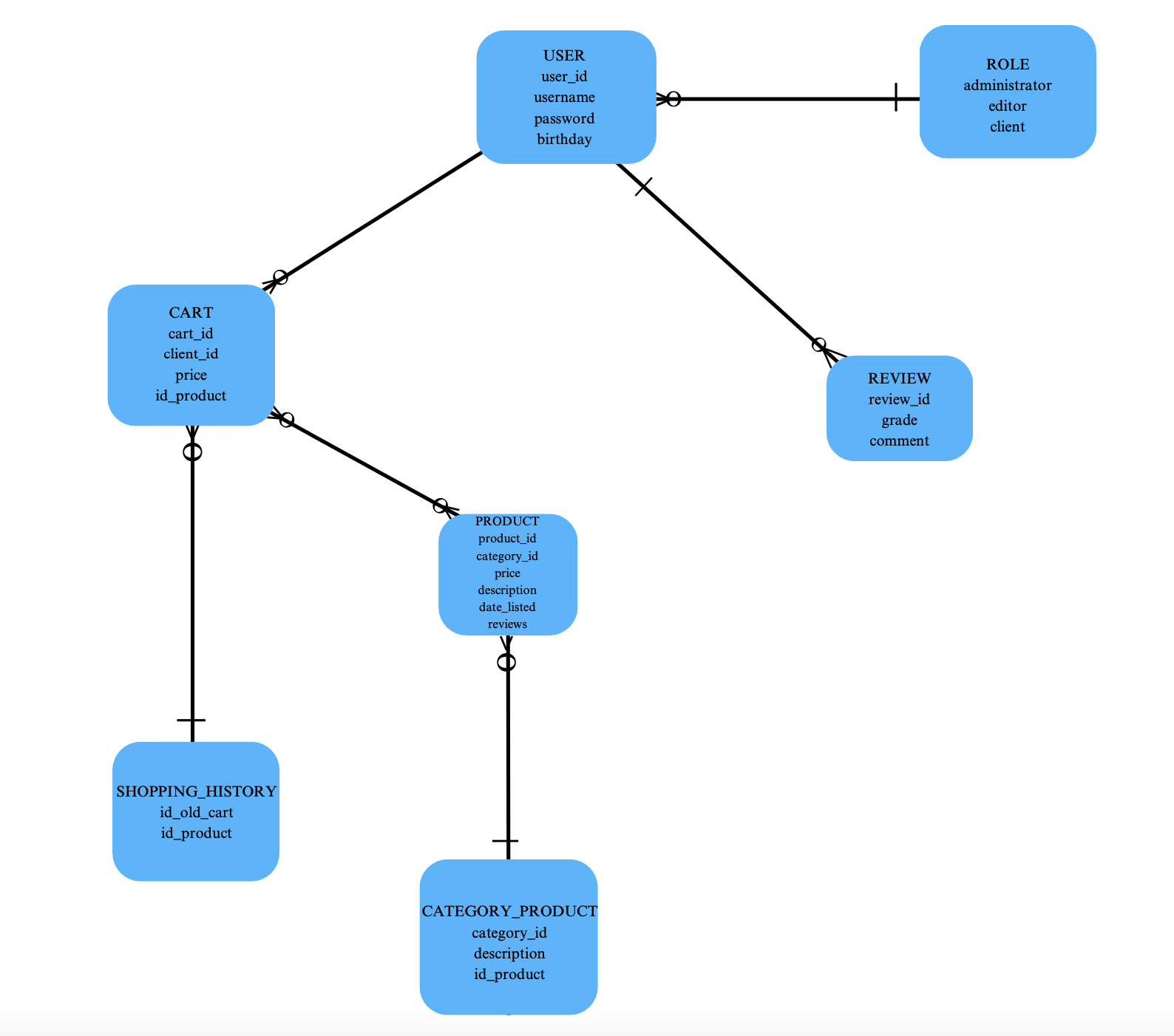
Trello Set-Up - Planificare task-uri si sprinturi





Modul de lucru

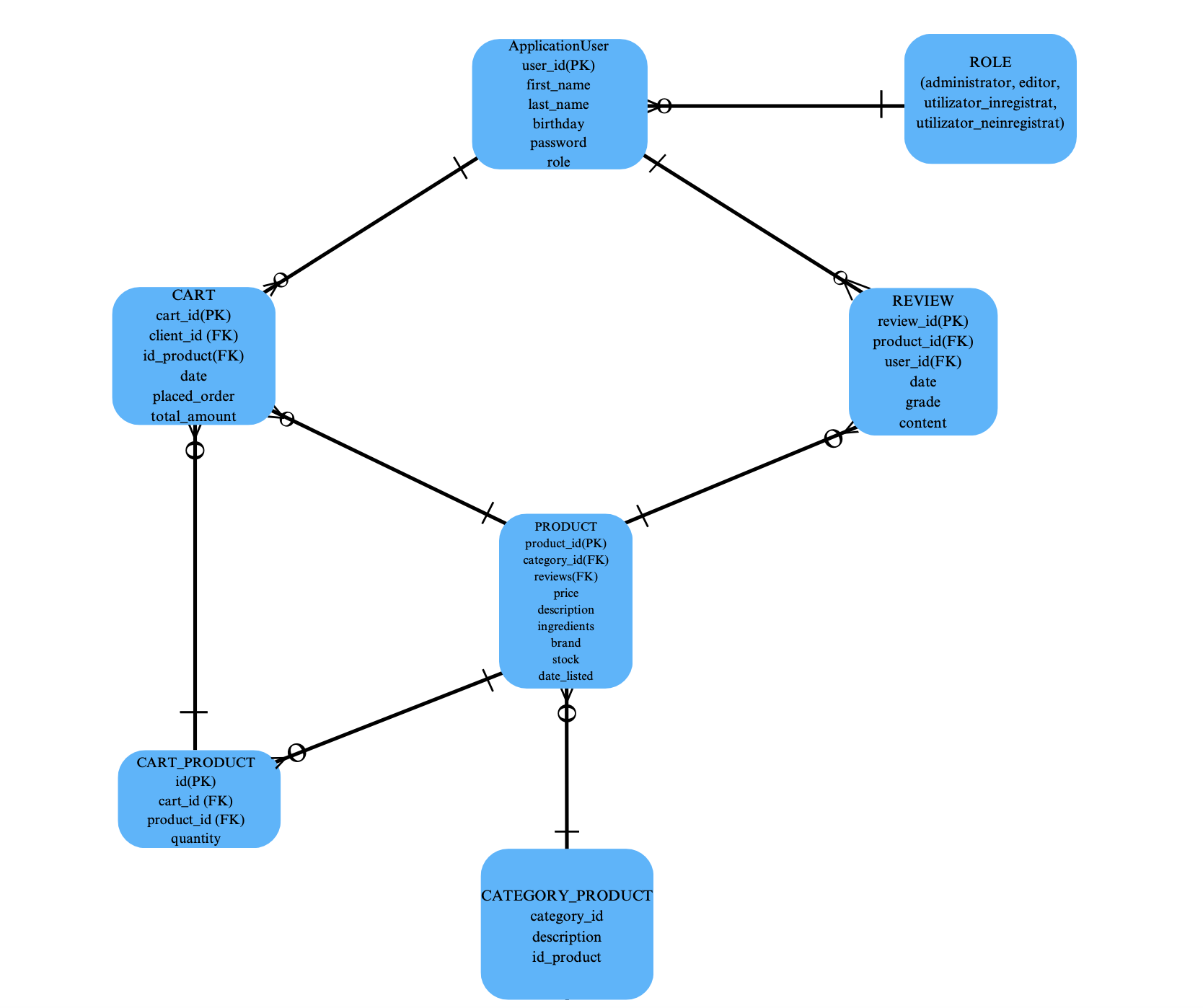
Sprint-uri de cate doua saptamani, ce inseamna ca task-urile din acel sprint vor fi finalizate la finalul sprint-ului, existand date intermediare de finalizare (semnalizate pentru fiecare card din workspace-ul de pe Trello) in functie de prioritatea lor. Am decis sa folosim metodologia Agile. Intalniri pentru a scrie cod si discuta vor avea loc o data pe saptamana, in rest lucrand in paralel in functie de task-ul fiecareia.

Diagrama bazei de date (varianta initiala, in continuarea documentului se va gasi si varianta finala):

# Etapa 2 - 15 – 19 Noiembrie

Overall:

Am inceput implementarea in detaliu a claselor, dar si definirea bazei de date (migratii etc). Momentan suntem in etapa de definire a modului de lucru al aplicatiei. Inca sunt mici diferente dintre arhitecturile pe care lucreaza fiecare, dar vom incerca rezolvarea lor prin Docker. In final, am convenit la o implementare comuna pentru Products, Categories si Reviews, ca dupa sa avem o baza pentru a dezvolta Carts, CartProducts (care rezolva relatia many-to-many dintre produse si cosuri de cumparaturi), care implica existenta atributului de Quantity in CartProducts si alte chei straine impreuna cu prorietati virtuale pentru asocierea tabelelor carora le apartin acele foreign keys, avand posibilitatea accesarii altor proprietati. Rezolvarea relatiei many-to-many se vede in ApplicationDbContext. In plus, am finalizat diagrama proiectului.



# Etapa 3 - 20 Noiembrie – 16 Decembrie

Overall:

Am incercat in repetate randuri folosirea Docker-ului, insa am avut diferite probleme tehnice pe parcurs, asa ca am revenit la folosirea unui proiect colaborativ pe Github. Vom incerca sa rezolvam individual diferentele de arhitectura (prin configurarea separata a fisierelor pentru diferitele tipuri de roluri pentru useri, a fisierului appsettings.json etc.). Varianta principala ramane cea de Windows, insa pe Mac incercam dezvoltarea altor aspecte care ar trebui sa mearga pe ambele arhitecturi, si anume modelele, view-urile si controller-ele. Singurele diferente dintre cele doua fiind ca pe Mac lipsesc motorul de cautare, afisarea paginata si implementarea a metodelor Create si Edit pentru imaginea unui produs, din moment ce un membru s-a ocupat de aceste aspecte pentru Windows, aceea fiind varianta pe care am decis sa o prezentam.

In aceasta perioada am dezvoltat ideea de a avea 4 utilizatori, insemnand ca fiecare actiune publica din fiecare controller a primit atributele de Authorize(Roles=””) corespunzatoare si am stabilit ca Admin-ul sa nu aiba restrictii aproape deloc asupra datelor pe care le poate sterge sau edita. De retinut ca utilizatorul neinregistrat trebuie numai sa isi creeze un cont, devenind implicit User si putand deveni Partner (acel utilizator care colaboreaza cu magazinul si adauga produse) sau chiar Admin daca unul dintre administratori doreste acest lucru (prin intermediul UsersController).

In continuare, am modificat putin logica de a cunoaste exact cine a postat produse si review-uri, astfel incat folosind metoda privata SetAccessRights de la nivelul fiecarui controller (ProductsController si ReviewsController), sa se afiseze butoane in Views pentru stergere si editare daca produsul/ review-ul respectiv apartin utilizatorului (Partner sau User, din moment ce Admin nu are nicio restrictie).

Urmatorul pas urmat in acesta etapa a fost sa manipulam in sfarsit atributul Image din clasa Product, de tip IFormFile, ce se poate observa in actiunile Create si Edit din ProductsController.

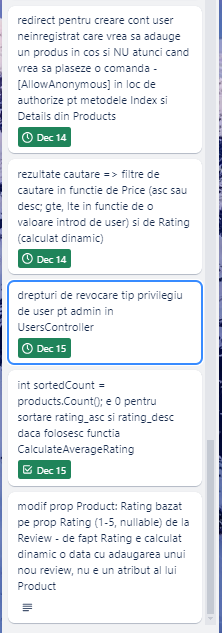
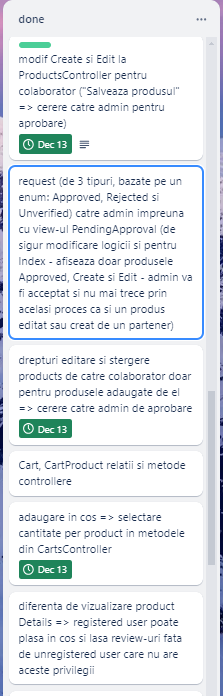
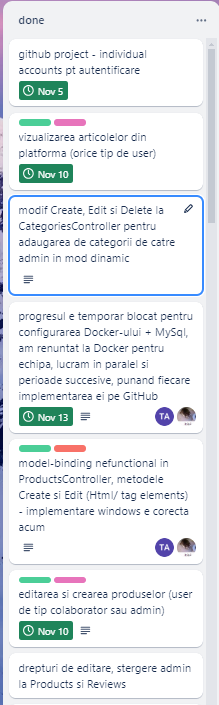
In plus, o data ce am avut destule inserari de produse in baza de date locala, am implementat paginarea, filtrarea rezultatelor dupa pret si o medie a rating-urilor de la nivelul tuturor review-urilor asociate fiecarui produs in parte, impreuna cu un motor de cautare. Legat de paginare si cum functioneaza alaturi de filtrare, daca nu e selectat niciun tip de sortare crescatoare/ descrescatoare, atunci nu se va aplica niciun filtru, nefiind afectata paginarea. E specificat in cerinta ca acest motor de cautare trebuie sa aiba o cautare partiala pentru atributul Name de la Product numai dupa ce a fost introdus in search bar (de retinut ca ar putea fi adaugata o implementarea asemanatoare d.p.d.v. al cautarii partiale pentru Content din Review). De altfel, media de Rating a review-urilor postate pentru un produs anume va fi calculat dinamic, adica o data cu adaugarea fiecarui review, fiind numai o afisare in view-ul partajat din Shared, numit ProductInfo, care calculeaza media in sectiunea de Overall Rating (am mentionat asta pentru ca initial ne gandisem ca Product sa aiba atributul de Rating si el sa fie modificat de fiecare data o data cu crearea, editarea sau stergerea unui review, dar ar fi insemnat sa ne complicam, deci am gasit o solutie destul de buna care permite o sortare dupa criteriile rating\_asc si rating\_desc folosind functii lambda).

Dupa, am decis sa facem partea cea mai importanta de logica a magazinului online, anume sa existe acele cereri de aprobare, respingere a produsului propus sa fie vandut la magazin de catre un partener al magazinului. A fost exprimata ideea asta la nivel de cod folosind un enum cu trei posibile valori: Approved, Rejected si Unverified. Acest lucru inseamna ca o data ce a fost creat un produs, cererea e trimisa in metoda Create din ProductsController, iar pana nu e aprobata/ respinsa de un Admin care intra in View-ul PendingAppoval (ce afiseaza numai acele produse pentru care atributul RequestStatus este Unverified), produsul pur si simplu nu va fi afisat pe pagina principala a site-ului cu o conditie de afisare doar pentru produsele Accepted in Index din ProductsController (ne referim la ce a fost setat ca si cale URL default in Program.cs in MapControllerRoute, adica View-ul Index din Products). De sigur, ca intregul proces a insemnat ca si metoda Edit, cu toate ca produsul sa zicem ca a fost aprobat prima oara sa apara in magazinul online, va modifica atributul produsului editat RequestStatus din Accepted in Unverified. De aceea, apar noi metode la care doar Admin-ul are acces: PendingApproval care are un View corespunzator si Approve si Reject care nu au View-uri separate (logica e continuta tot in View-ul PendingApproval).

De asemenea, cand vrea sa fie adaugat un produs in cos, acel user trebuie sa fie logat, daca nu e logat nu are acces la prea multe informatii de pe site, numai metodele Index, Details din ProductsController via AllowAnonymous, de aceea o data ce se apasa pe butonul Add To Cart din pagina de Details a unui produs, utilizatorul e prompted sa se logheze.

Concluzie:

Cu toate dificultatile pe care le-am avut din cauza diferentelor arhitecturale, am reusit completarea partii de backend, alaturi de un frontend reusit, pe care insa il vom imbunatati pe parcurs prin folosirea Bootstrap-ului. Proiectul este in final functional, are o baza de date care imita foarte bine una folosita de magazinele online reale (de produse de infumusetare: makeup, skincare etc). In screenshot-urile de mai jos, din Trello-ul folosit pe parcursul proiectului, am inclus tot ceea ce am vrut sa realizam si am si reusit, cuprinzand toate cerintele de backend ale proiectului.



Aici sunt cateva screenshot-uri cu proiectul finalizat: