**MINISTERUL EDUCAŢIEI, CULTURII ȘI CERCETĂRII**

**Universitatea Tehnică a Moldovei**

**Facultatea Calculatoare Informatică şi Microelectronică**

**Departamentul Ingineria Software și Automatică**

**Proiect de an**

**Disciplina:**Tehnici și Mecanisme de Proiectare Software

**Tema:** Implimentarea si dezvoltarea unui serviciu de e-comert

Implementation and design of an e-commerce service

|  |  |  |
| --- | --- | --- |
| **Student:** | **\_\_\_\_\_\_\_\_\_\_\_\_\_** | **Popa Valeriu, TI-195** |
| **Coordonator:** | **\_\_\_\_\_\_\_\_\_\_\_\_\_** | **Cebotari Daria, asis. univ.** |
|  |  |  |

**Chişinău 2022**

**Cuprins**

[Introducere 2](#_Toc103798628)

[1 Analiza domeniului de studiu 3](#_Toc103798629)

[1.1 Scopul, obiectivele și cerințele sistemului 4](#_Toc103798630)

[1.2 Analiza sistemelor deja existente 5](#_Toc103798631)

[2. Realizarea sistemului 8](#_Toc103798632)

[2.1 Proiectarea aplicației 9](#_Toc103798633)

[2.2 Descrierea tehnologiilor pentru sistem 13](#_Toc103798634)

[2.3 Descrierea la nivel de cod pe module 14](#_Toc103798635)

[3. Documentarea produsul realizat 17](#_Toc103798636)

[Concluzii 23](#_Toc103798637)

[Bibliografie 24](#_Toc103798638)

[Anexa A 1](#_Toc103798639)

[Anexa B 1](#_Toc103798640)

# Introducere

Din cele mai vechi timpuri comertul era una dintre cele mai de vaza ocupatii a omenirii. Pe parcursul dezvoltarii oemnirii aceasta ocupatie permitea oamenilor sa-si atinga scopurile sale, fie ca sa faca ceva bani, fie sa agoniseasca vre-o unealta necesara. Aceasta ocupatie este intr-atit de simpla si intrebuintata omenirii, incit a ajuns pina in secoul *21* desigur evoluind impreuna cu civilizatia, progresind de la comert local intre 2 persoane, pieti locale si negustori care strabateau toata lumea pentru a aduce marfa exotica la *comert on-line*, unde nu mai avem asa notiune ca marfa exotica sau ceva ce nu putem procura deoarece nu este accesibil din cauza problemelor logistice. Pentru acest tip de comert, inca din anii ’90 se creeaza felurite platforme. Deci, de ce sa nu fac si eu una?

Lucrarea dată urmărește scopul de a crea un serviciu pentru comert electronic intre persoane fizice.

Pe linga aceasta, lucrarea urmareste scopul de a demonstra posibilitatea utilizarii sabloanelor de proiectare software intr-un asemenea proiect neobisnuit, deoarece acesta va fi un telegram-bot.

Lucrarea este structurată în trei capitole, unde se va analiza concret modul de proiectare și de realizare a poiectului dat.

Primul dintre aceste capitole vine să va explice tematica acestuia. Aici se vor stabili obiectivele, scopurile si cerintele acestui proiect. La fel se vor analiza sisteme de gen dat deja existente pentru a creea o inchipuire clara, cum trebuie sa fie sistemul meu.

Al doilea capitol cuprinde partea de implementarea a codului. Aici voi vorbi, cum a fost proiectat serviciul, cum si de ce anume asa a fost realizat. Voi relata tehnologiile folosite in fiecare modul. La fel voi insera bucati de cod, care realizeaza functiile principale ale sistemului. Pentru elaborarea proiectului va fi folosit limbajul **Python**. Presupun ca voi folosi 6-7 sabloane de proiectare: **Factory, Builder, Singleton, Decorator, Proxy, Observer si Strategy**. Pentru baza acestui telegram-bot va fi folosita una dintre bibliotecile Python: **telebot** sau **aiogram.** Desigur va fi adaugata si o baza de date: **SQLite** sau **MySQL**. In final se planifica un produs gata pentru explotare.

În ultimul capitol se va descrie produsul final si va fi elaborat un ghid al utilizatorului, cu descrierea pe lung al tuturor posibilitatilor.

# Analiza domeniului de studiu

În esență, comerțul electronic se referă la cumpărarea și vânzarea de bunuri și/sau servicii prin canale electronice, cum ar fi internetul. Comerțul electronic se bazează pe tehnologii precum comerțul mobil, transferul electronic de fonduri, managementul lanțului de aprovizionare, marketingul pe internet, procesarea tranzacțiilor online, schimbul electronic de date (EDI), sistemele de gestionare a stocurilor și sistemele automate de colectare a datelor. Comerțul electronic este, la rândul său, condus de progresele tehnologice ale industriei semiconductoarelor și este cel mai mare sector al industriei electronice. El utilizează de obicei web-ul pentru cel puțin o parte a ciclului de viață al unei tranzacții, deși poate folosi și alte tehnologii, cum ar fi e-mailul. Tranzacțiile tipice de comerț electronic includ achiziționarea de produse (cum ar fi cărți de la Amazon) sau servicii (cum ar fi descărcări de muzică sub formă de distribuție digitală, cum ar fi iTunes Store). [1]

Există *trei domenii ale comerțului electronic: vânzarea cu amănuntul online, piețele electronice și licitațiile online*. Comerțul electronic este susținut de afaceri electronice. Serviciile de comert electronic pot angaja urmatoarele tipuri de comert:

• Cumpărături online pentru vânzări cu amănuntul direct către consumatori prin site-uri web și aplicații mobile și comerț conversațional prin **chatbot**.

• B2B (business to business) – Aceasta implică companii care fac afaceri între ele. Un exemplu este producătorii care vând către distribuitori și angrosiştii care vând către retaileri. [2]

• B2C (business to consumer) – B2C constă în afaceri care vând publicului prin intermediul unui software de coș de cumpărături, fără a avea nevoie de nicio interacțiune umană. La asta se gândesc majoritatea oamenilor când aud „e-commerce”. Un exemplu în acest sens este Amazon. [2]

• C2B (consumator to business) – În comerțul electronic C2B, consumatorii postează online un proiect cu un buget stabilit, iar companiile licitează pentru proiect. Consumatorul analizează ofertele și selectează compania. Elance este un exemplu în acest sens. [2]

• C2C (de la consumator la consumator) – Aceasta are loc în cadrul anunțurilor online clasificate, forumurilor sau piețelor unde persoanele fizice își pot cumpăra și vinde bunurile. Exemple în acest sens sunt Craigslist, eBay și Etsy. [2]

# Scopul, obiectivele și cerințele sistemului

**Scopul** principal al acestui proiect este de a creea un serviciu de E-commerce complet functionabil ca va usura cometrul inte persoane. Ca model dupa functionalitate se va lua sistemul nostu autohton „999.md” imbinat cu „eBay”. Pe linga aceasta realizarea acestui proiect poarta scopul de a dezvolta de prinderi de proiectare a softului urmind princiile SOLID si acumularea experientei in frameworului Django, bazat pe Python.

Visele și dorințele nu sunt obiective până când nu sunt scrise ca rezultate finale specifice pe hârtie.

Din citatul de mai sus putem deduce urmatoarea, pentru a avea o aplicatie bine ginditata, este nevoie sa-i formuam niste obiective concrete dupa care sa ne conducem pe tot parcursul lucrului asupra proiectului. Eu mi-am formulat urmatoarele:

**Obiectivele proiectului sunt:**

* Crearea unui serviciu web;
* Dezvoltarea unor noi abilitati in dezvoltarea software;
* Studierea modului in care se creeaza un bot in messenger;
* Efectuarea proiectului de curs;
* Deprinderea implemetez sabloanele de proiectare;
* Deprinderea de scriere a codului respectind principiile SOLID.

Orice sistem, trebuie proiectat si dezvoltat in conformitate cu un plan. Planul acestuia se alcatuieste din cerintele stabilite la inceputul lucrului. Pentru proiectul meu sunt urmatoarele:

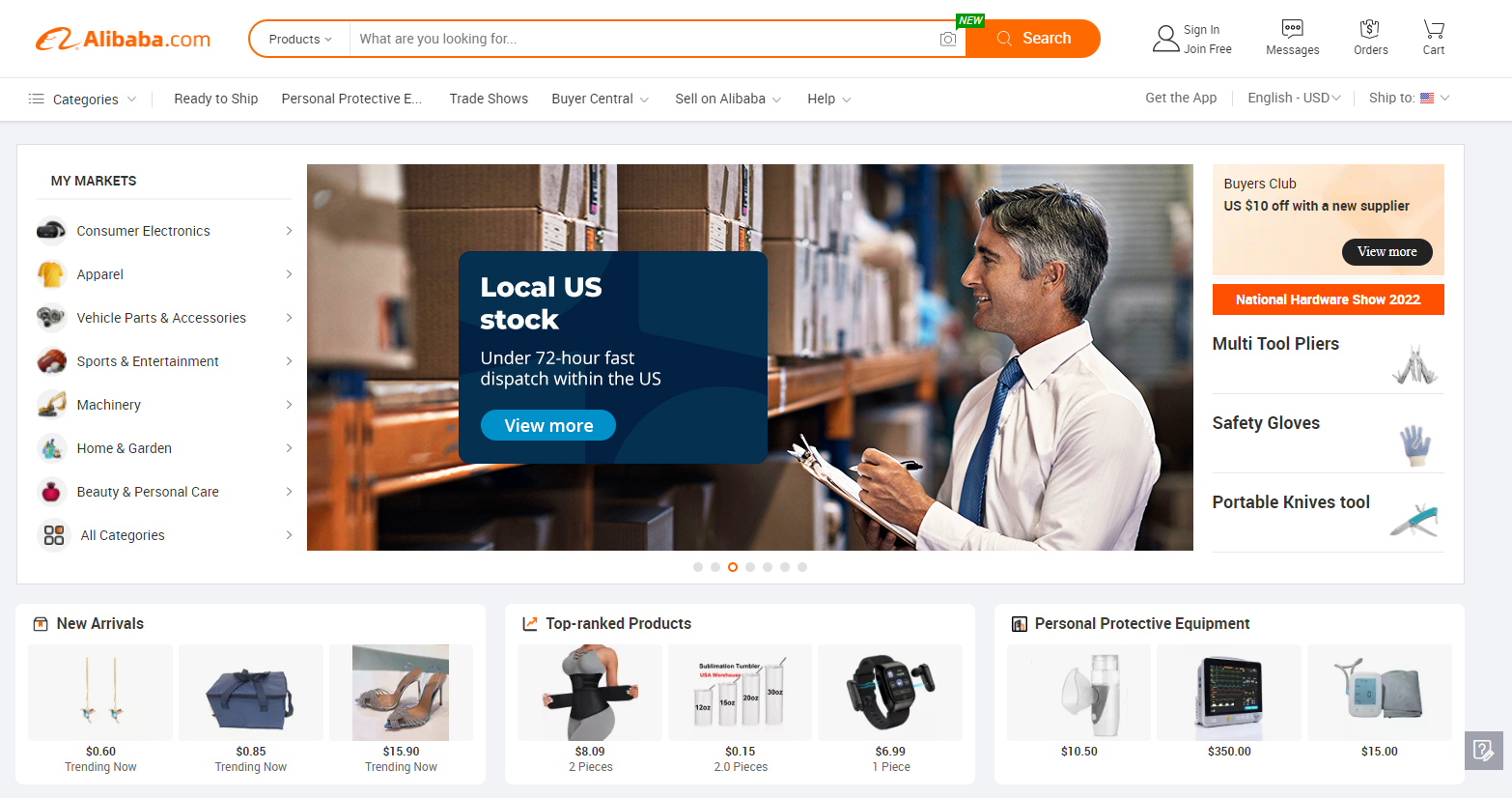
**Cerintele sistemului:**

* Modulele principale ale sistemului sa fie complet functionabile;
* Sa fie proiectat folosind sabloanele de proiectare;
* Codul sa fie scris folosind principiile SOLID;
* Sa aiba un UI cit de cit clar pentru un potential utilizator final.

# 1.2 Analiza sistemelor deja existente

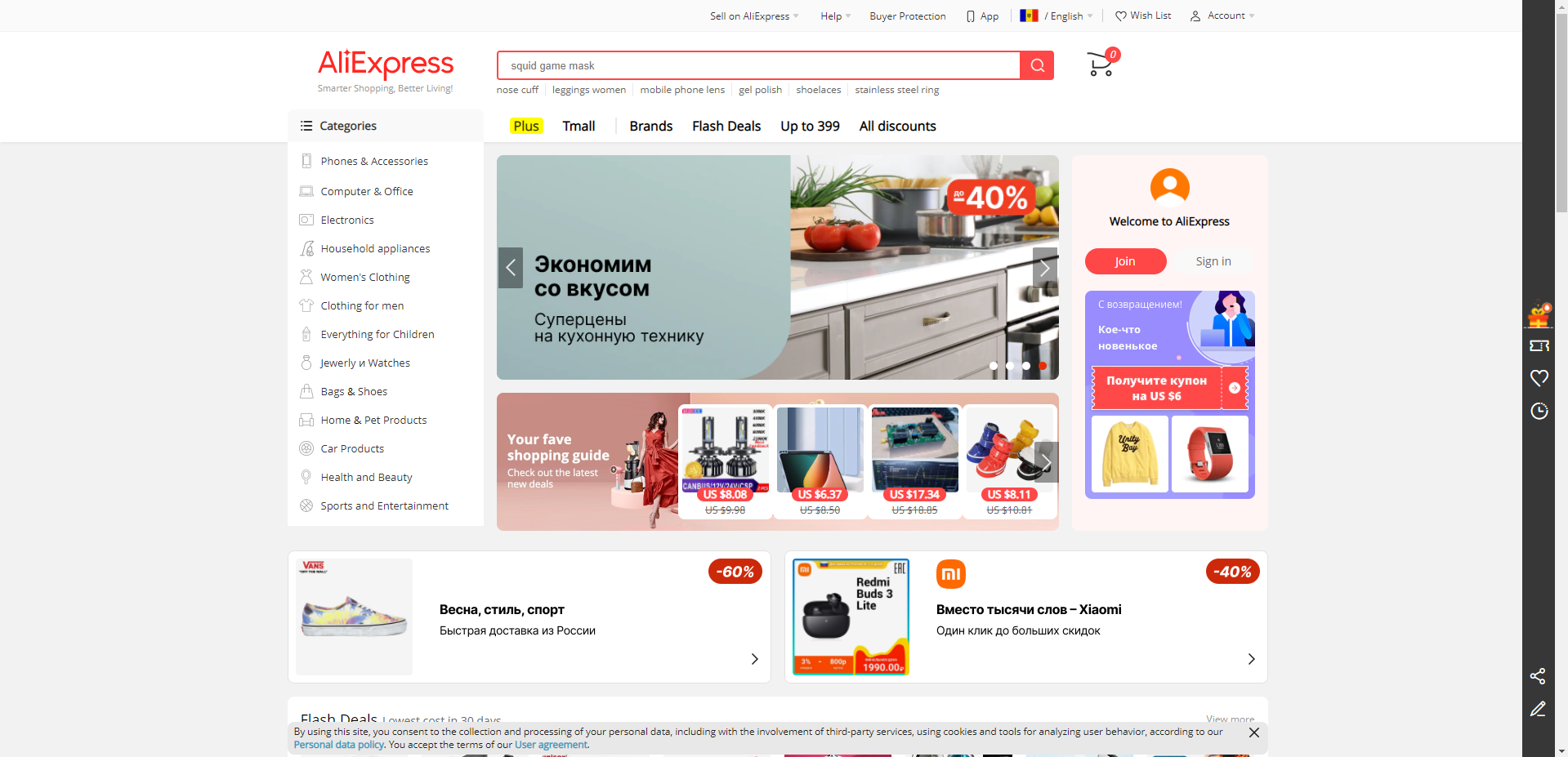
In lume avem o sumedenie de platforme pentru comert electronic. In studiul meu am ales citeva cele mai renumite la nivel mondial si in RM.

• **Alibaba** – este o companie multinațională chineză de tehnologie specializată în comerț electronic, retail, internet și tehnologie. Este cea mai mare din lume, compania furnizează servicii de vânzări consumator-to-consumer (C2C), business-to-consumer (B2C) și business-to-business (B2B) prin portaluri web, precum și servicii de plată electronică, motoare de căutare pentru cumpărături și cloud computing Servicii. Deține și operează un portofoliu divers de companii din întreaga lume în numeroase sectoare de afaceri. [3] Interfata acesteia poate fi vazuta in *Fig.1 Interfata „Alibaba”.*



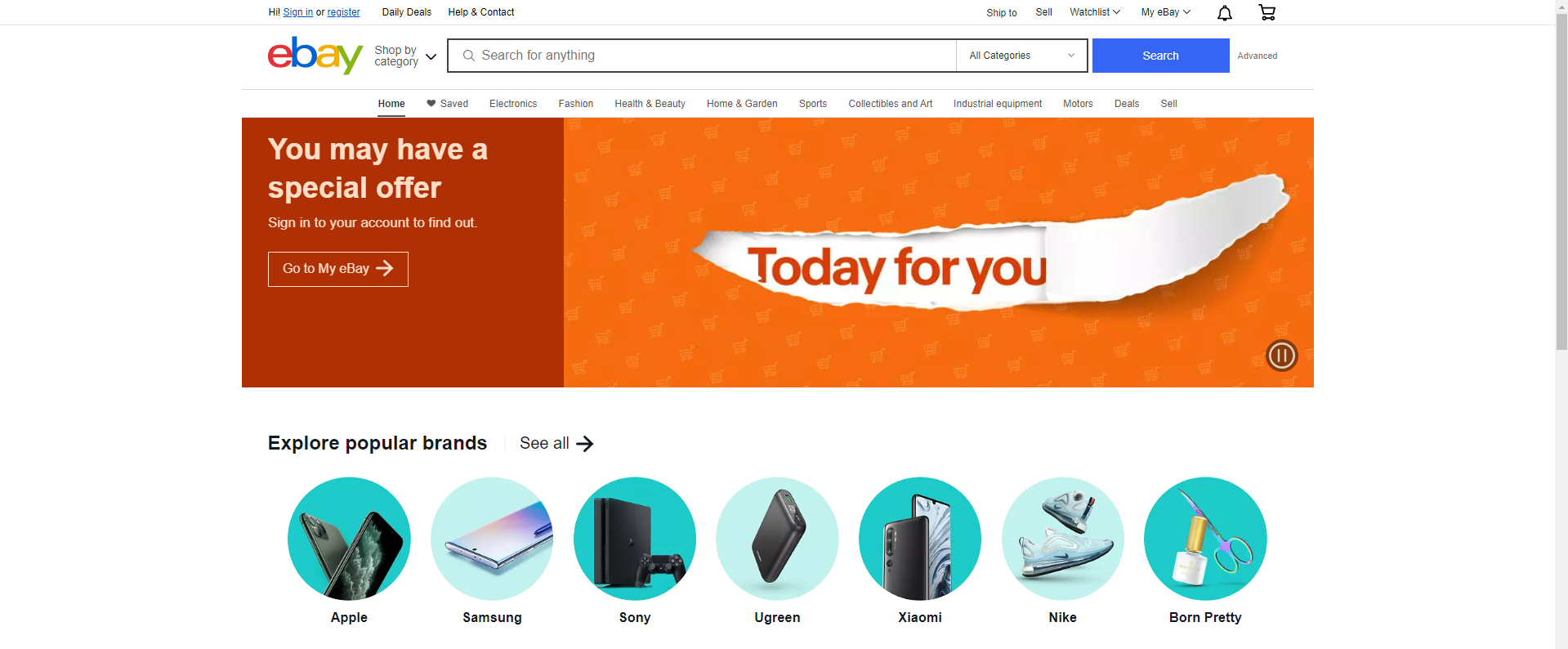
*Fig.1 Interfata „Alibaba”*

• **AliExpress** - este un serviciu de vânzare cu amănuntul online cu sediul în China și deținut de Alibaba Group. Lansat în 2010, este format din întreprinderi mici din China și din alte locații, cum ar fi Singapore, care oferă produse cumpărătorilor internaționali online. Este cel mai vizitat site de comerț electronic din Rusia. De la inceput a avut formatul B2B, dar mai apoi au aparut formatele de comert B2C si C2C. [6] Interfata acesteia poate fi vazuta in *Fig.2 Interfata „AliExpress”.*



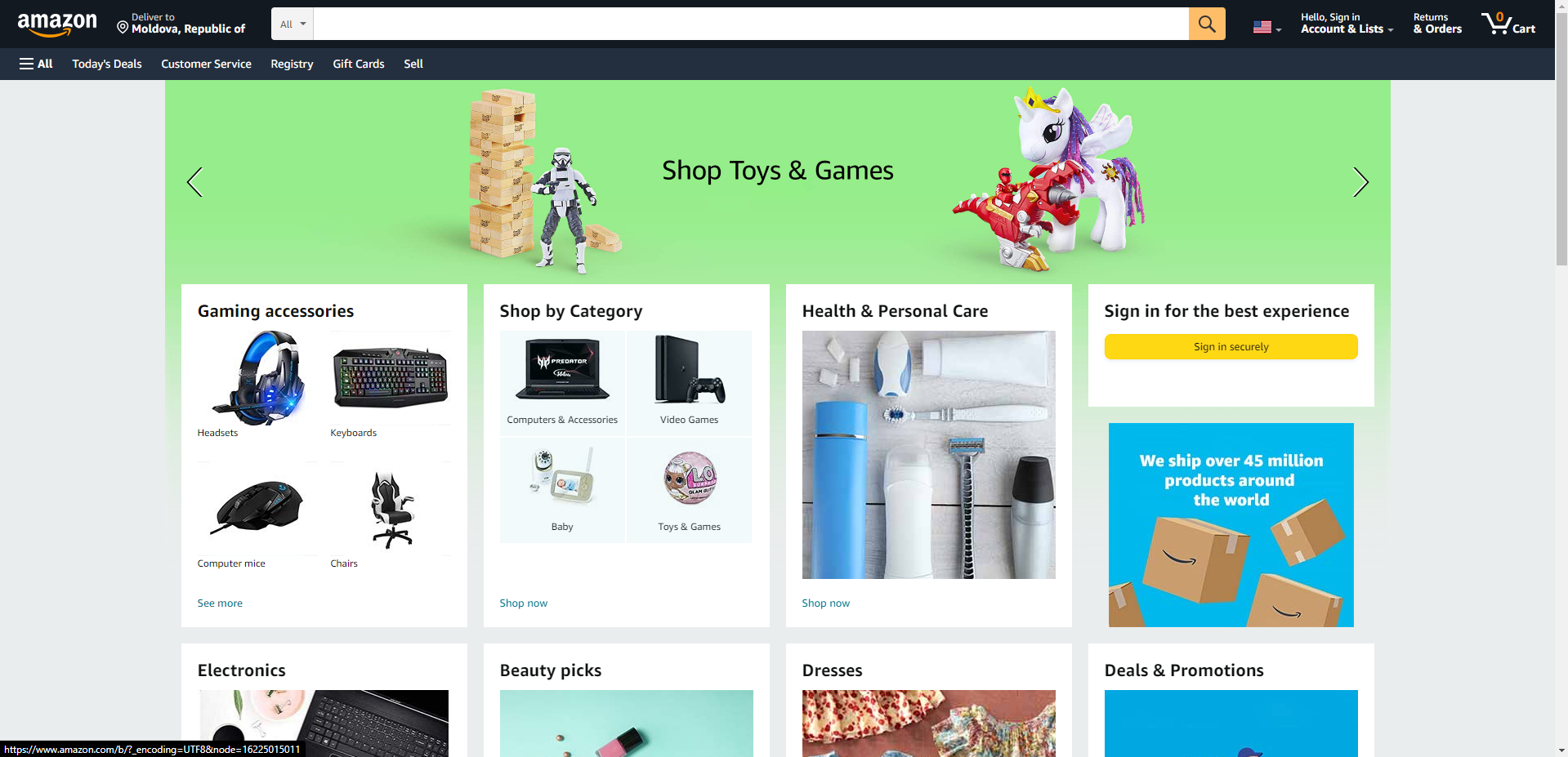
*Fig.2 Interfata „AliExpress”*

• **eBay** - este o corporație multinațională americană de comerț electronic, care facilitează vânzările de la consumator la consumator și de la întreprindere la consumator prin intermediul site-ului său web. Este cea mai renumita platforma de comert electronic in domeniu C2C, care este folosita de populatia intregului glob pamintesc pentru a vinde bunuri din proprietatea personala. [5] Interfata acesteia poate fi vazuta in *Fig.3 Interfata „eBay”.*



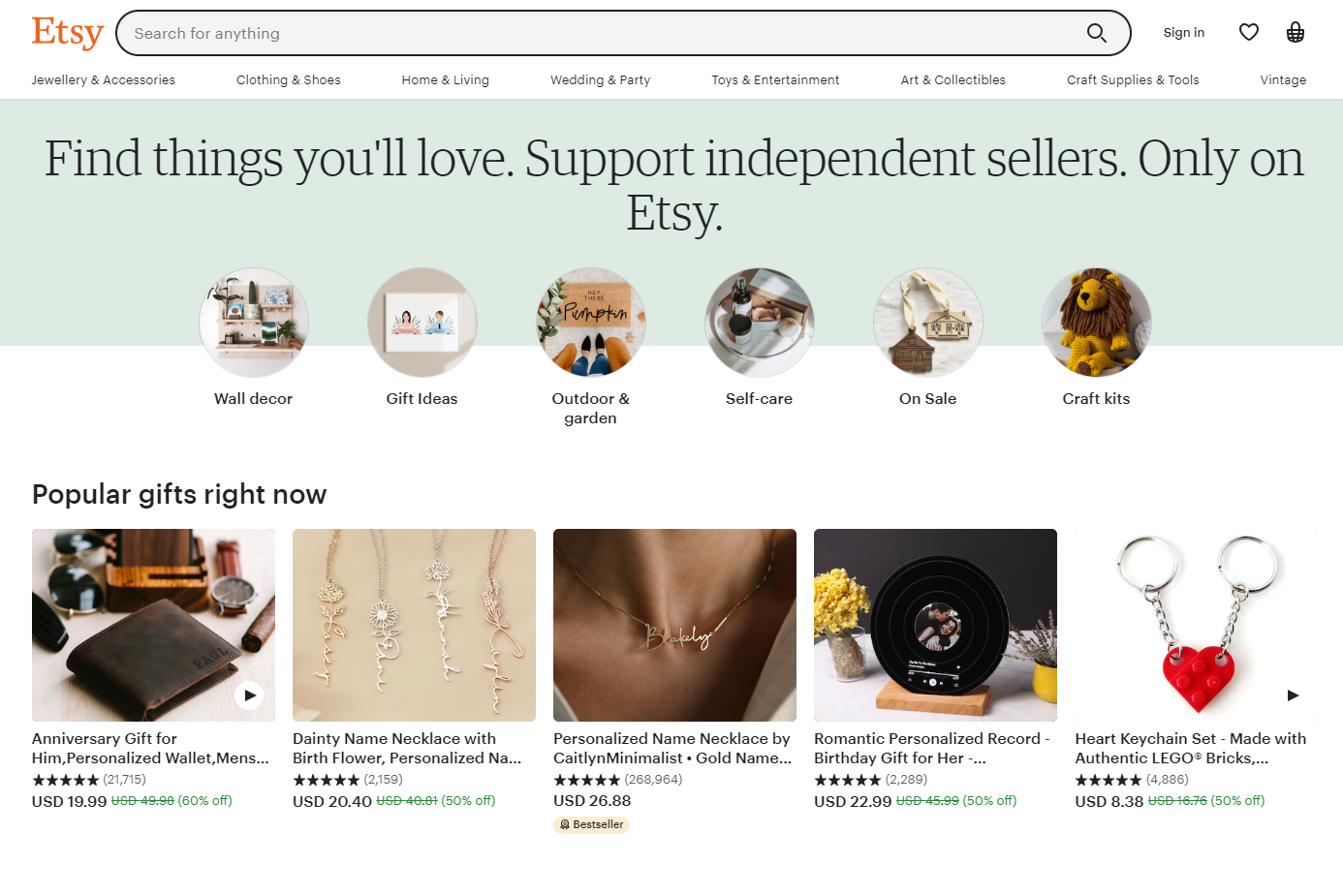
*Fig.3 Interfata „eBay”*

• **Amazon** - este o companie multinațională americană de tehnologie care se concentrează pe **comerțul electronic**, cloud computing, streaming digital și inteligența artificială. A fost denumită „una dintre cele mai influente forțe economice și culturale din lume” și este una dintre cele mai valoroase mărci din lume. Este una dintre cele cinci mari companii americane de tehnologie a informației, alături de Alphabet, Apple, Meta și Microsoft. Se ocupa de comert B2C. [4] Interfata acesteia poate fi vazuta in *Fig.4 Interfata „Amazon”.*



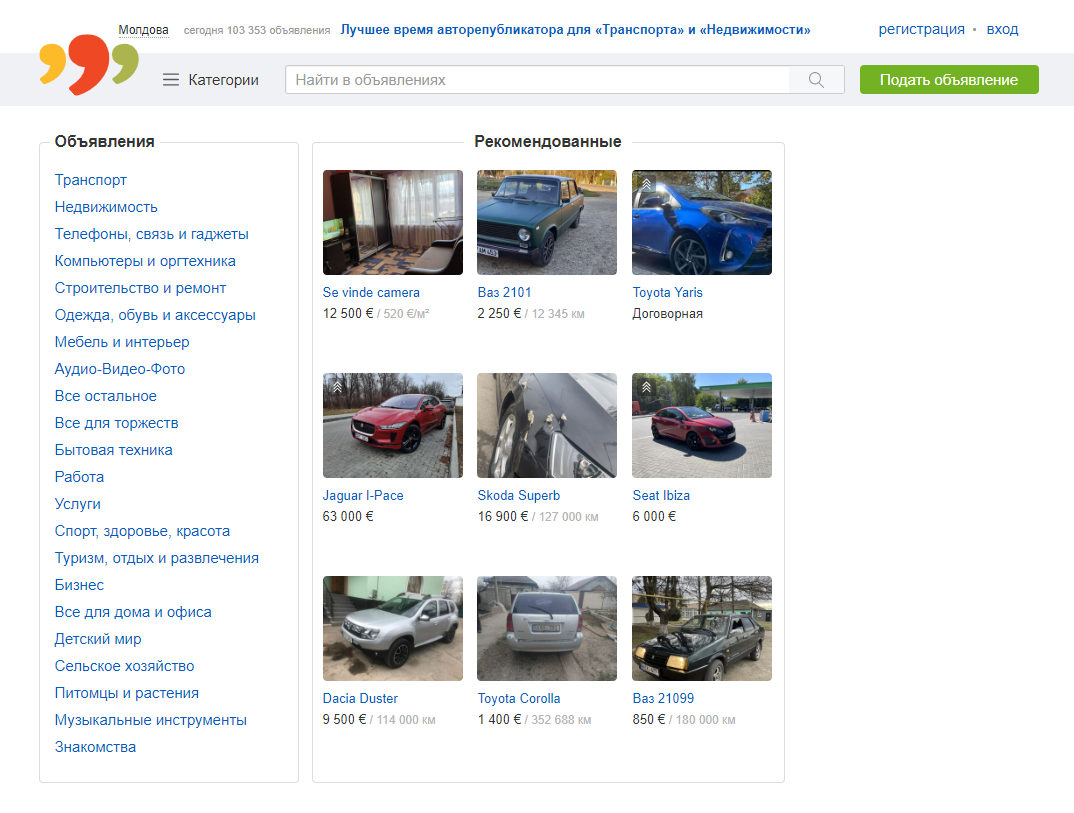
*Fig.4 Interfata „Amazon”*

• **Etsy** – este o companie americană de comerț electronic care se concentrează pe articole handmade sau vintage și rechizite artizanale. Aceste articole se încadrează într-o gamă largă de categorii, inclusiv bijuterii, genți, îmbrăcăminte, decor și mobilier pentru casă, jucării, artă, precum și rechizite și unelte de artizanat. Toate articolele de epocă trebuie să aibă cel puțin 20 de ani. Site-ul urmează tradiția târgurilor de artizanat deschise, oferind vânzătorilor vitrine personale unde își listează mărfurile la o taxă de 0,20 USD per articol. Este structurat in formatul C2C si B2C. [7] Interfata acesteia poate fi vazuta in *Fig.5 Interfata „Etsy”.*



*Fig.5 Interfata „Etsy”*

• **999.md** – este o platforma de comert electronic autohtona. Face parte din holding-ul Simpals. Este pe larg folosita de cetatenii RM pentru promovarea comertului C2C si B2C. Interfata acesteia poate fi vazuta in *Fig.6 Interfata „999.md”*



*Fig.6 Interfata „999.md”*

# 2. Realizarea sistemului

Pentru proiectu de curs la TMPS, mi-am propus sa dezvolt un sistem de e-Commerce. Platforma care mi-am ales-o pentru creearea acestuia este Telegram, si anume un telegram-bot. De ce anume asa? Telegram este o platforma populara, stabila, cu functionalitati vaste, UI frumos si usoara pentru dezvolatarea unui serviciu.

Pentru dezvoltarea acesteia am ales limpajul Python, deoarece este cel mai bine cunoscut de mine, este simplu, permite realizarea unor lucruri complicate in mod foatre simplu si este intr-un oarecare mod orientat pe obiecte ceea ce-mi permite imlimentarea sabloanelor de proiectare. In interiorul acestui lumbaj ma bazez pe biblioteca aiogram. Este una dintru multele pentru dezvoltarea unui telegram bot, insa aceasta se deosebeste de restul prin faptul ca este asincrona, ceea ce permite dezvoltarea unui bot cu timp de raspuns foatre scurt.

Interfata grafica a acestui serviciu va fi realizata complet folosind posibilitatile interioare ale Telegram. Acesta are un UI frumos, daca nu cel mai frumos dintre toate messengerurile, si permite creearea unui bot intuitiv si claroricarui.

Baza de date va fi implimentata in Sqlite, iarasi deoarece este cea mai simpla iesire, care nu necesita extra soft mare si nu creeaza complicatii la instalarea acestuia. Nu este cea mai buna iesire, dar pentru un proiect de curs, care nu va fi lansat la moment este ok. Pe aceasta ar putea fi inlocuita cu PostgreSQL, deoarece are compatibilitate ideala cu Python.

Logica acestei aplicatii este inspirata de la serviciul autohton ’’999.md’’. Planific sa implimentez tot functionalul necesar pentru o eventuala lansare a acestui in scurt timp. Adica, adaugarea de catre utilizator a anuntului in regim live, vizualizarea listei de anunturi adaugate de dinsul, stergerea anunturilor tot de catre dinsul si promovarea lor. La fel si pentru eventual cumparator, un sistem de filtre intuitiv si simplu plus optiunea de a lua legatura cu vinzatorul, atit in Telegram cit si TelegramOut.

Task listul pentru proiect il puteti vedea in Anexa B sau [aici](https://disk.yandex.com/d/QEc_JRxKexoM0Q) [Regim de access]

# 2.1 Proiectarea aplicației

In acest modul, voi relata ce este sub capot la telegram bot-ul creeat, la nivel teoretic. Ce poate, cum o face.

Pentru inceput este nevoie de a stabili functionalul ca il va avea acesta. Acestra este ilustrat in diagrama use-case din *fig.7*. Dupa cum observam avem 2 tipuri de utilizatori (Customer si Seller). Acestia vor avea functional diferit si strict necesar pentru activitatea sa.



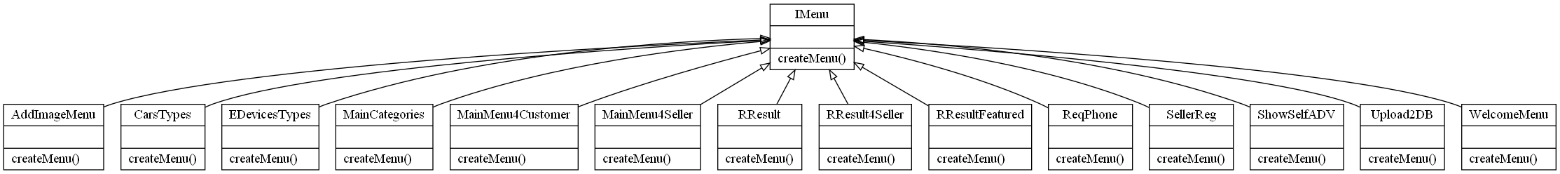
*Fig.7 Diagrama use-case pentru serviciul creeat*

Pentru a oforma back-endul am folosit sabloane de proiectare. 3 Creationale, 2 Structurale si 2 Comportamentale:

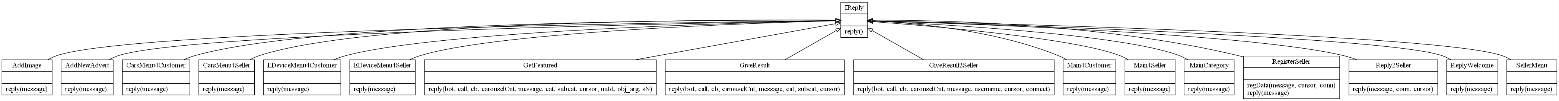
**Singleton –** pentru a conecta baza de date la serviciul; ( *fig.8 )*

  
*Fig.8 Diagrama pentru* ***Singleton***

**Factory –** pentru acreea meniurile (interfata clickabila) si pentru a defini rapunsurile botului la fiecare mesaj primit de la utilizator. Apropo, acestea 2 se aflta in strinsa legatura; ( *fig.9.1, fig.9.2* -> *Anexa A*)

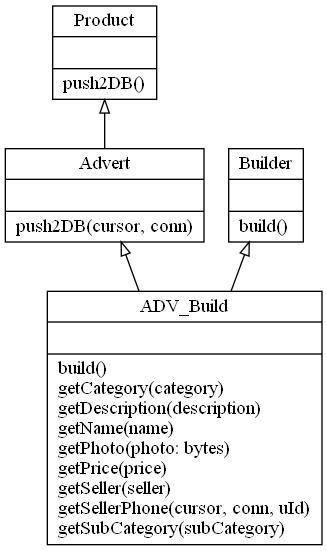
******

*Fig.9.1 Diagrama pentru* ***Factory*** *compatrimentul Meniuri*

**

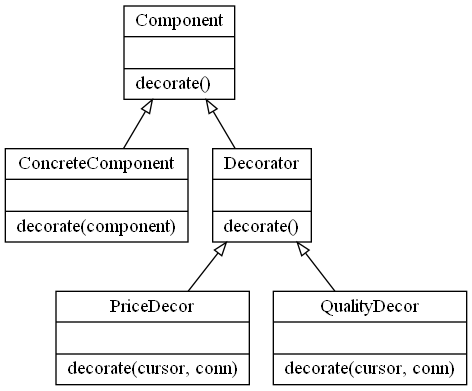
*Fig.9.2 Diagrama pentru* ***Factory*** *compartimentul Raspunsuri*

**Builder –** pentru a creea anuntul si a-l inregistra in baza de date. Aceasta este pentru Seller; ( *fig.10*  )

**

*Fig.10 Diagrama pentru* ***Builder***

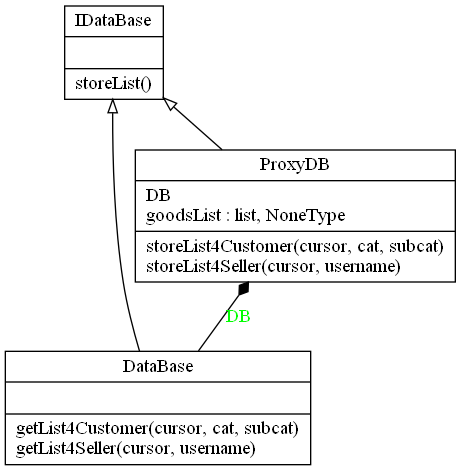
**Decorator** – pentru a creea posibilitatea pentru Seller de a adauga label-uri la anuntul sau, care il vor ajuta in promovare; ( *fig.11* )

******

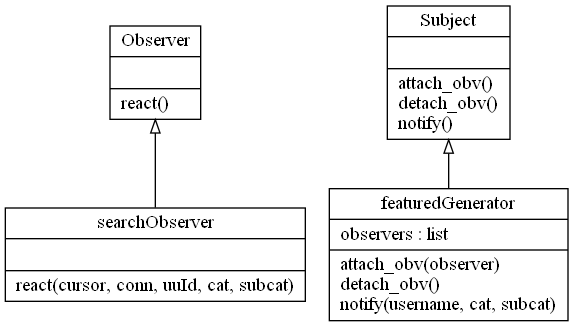
*Fig.11 Diagrama pentru* ***Decoraror***

**Proxy** – pentru a usura lucrul bezei de date, aceasta fiind un puctu de acces regulat catre baza de date;

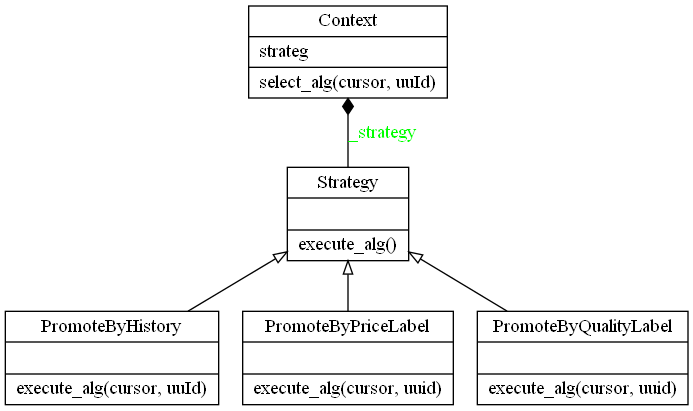
( *fig.12* )

****** *Fig.12 Diagrama pentru* ***Proxy***

**Observer** – pentru a fixa si inregistra in baza de date ultimul request al fiecarui utilizator astfel creeind un fel de istorie a cautarii, care ulterior va fi folosita in promovarea de catre serviciului a anuntirilor (asa numit target-placement); ( *fig.13* )

****** *Fig.13 Diagrama pentru* ***Observer***

**Strategy** – este folosit pentru promovarea anunturilor. Foloseste strategii bazade pe istoricul cautarii, unde este strins legat de observer, si label-uri care exista datorita decoratorului. ( *fig.14* )

****** *Fig.14 Diagrama pentru* ***Strategy***

Deasemenea diagrama generala a proiectului o puteti gasi in *Fig.28 Diagrama generala a proiectului din* ‘Anexa A’. Daca la general, anume aceste sabloane mi-fost de folos realiza functionalul propus.

# 2.2 Descrierea tehnologiilor pentru sistem

Pentru realizarea proiectului am ales sa utilizez limbajul Python. Pentru a programa bot-ul am nevoie de o biblioteca. Alegerea mea a cazut pe *aiogram*. In urmatarele paragrafe va voi vorbi despre Python si Django evidentiind avantajele acestora.

**Python** este un limbaj de programare puternic, ușor de învățat. Are structuri eficiente de date la nivel înalt și o abordare simplă, dar eficientă a programarii orientate pe obiecte. Sintaxa elegantă și tastarea dinamică a lui Python, împreună cu natura sa interpretată, îl fac un limbaj ideal pentru crearea de scripturi și dezvoltarea rapidă a aplicațiilor în multe domenii pe majoritatea platformelor. Suporta multe paradigme de programare, inclusiv programarea structurată (în special procedurală), orientată pe obiecte și funcțională. Este adesea descris ca un limbaj „cu baterii incluse” datorită bibliotecii sale standard cuprinzătoare. [11]

Cel mai des este folosit pentru:  dezvoltare web (partea server), dezvoltare de software, matematică, scripturi de sistem.  Iata citeva avantaje ale acestui limbaj:

**Python este un limbaj interpretat.** Vine cu IDLE (Mediul de dezvoltare interactiv). Acesta este un interpret și urmează structura REPL (Read-Evaluate-Print-Loop). Execută și afișează ieșirea unei linii la un moment dat. Deci, afișează erori în timp ce rulați o linie și afișează întreaga urmărire a stivei pentru eroare.  **Limbajul tipizat dinamic** Python nu este tipizat static ca Java. Nu trebuie să declarați tipul de date în timp ce definiți o variabilă. Interpretul determină acest lucru în timpul execuției pe baza tipurilor de părți ale expresiei. Acest lucru este ușor pentru programatori, dar poate crea erori de rulare. **Este** **orientat pe obiecte.** Python este orientat pe obiecte, dar acceptă atât programarea funcțională, cât și cea orientată pe obiecte. Totul în Python este un obiect. Are conceptele OOP (programare orientată pe obiecte) precum moștenirea și polimorfismul.  **Are sprijin popular și comunității mari**. Python are una dintre cele mai mari comunități de pe StackOverflow și Meetup. Dacă aveți nevoie de ajutor, comunitatea vă va răspunde la întrebări. De asemenea, au deja multe răspunsuri la întrebări despre Python. **Este Open-Source.** Python este open-source și comunitatea contribuie mereu la el pentru a-l îmbunătăți. Este gratuit și codul sursă este disponibil publicului în mod gratuit. Puteți descărca Python de pe site-ul oficial Python. **Are biblioteca standard mare.** Biblioteca standard este mare și are multe pachete și module cu funcționalități comune și importante. Dacă aveți nevoie de ceva care este disponibil în această bibliotecă standard, nu trebuie să îl scrieți de la zero. Din acest motiv, te poți concentra pe lucruri mai importante.De asemenea, pot fi instalate pachete din PyPI (Python Package Index) dacă doriți și mai multe funcționalități. **Independent de platformă.** Python este independent de platformă. Dacă scrieți un program, acesta va rula pe diferite platforme precum Windows, Mac și Linux. Nu trebuie să le scrieți separat pentru fiecare platformă. **Suport GUI.** Puteți utiliza Python pentru a crea GUI (Interfețe grafice cu utilizatorul). Puteți utiliza tKinter, PyQt, wxPython, Pygame sau Pyside pentru aceasta.Python are un număr mare de cadre GUI disponibile pentru o varietate de alte soluții multiplatforme. Se leagă de tehnologiile specifice platformei.  [8]

**Aiogram** este un framework destul de simplu și complet asincron pentru Telegram Bot API scris în Python 3.7 cu asyncio și aiohttp. [10] Daca simplu, este o biblioteca pentru Python cu ajutorul careia poate fi codat un telegram bot.

# 2.3 Descrierea la nivel de cod pe module

Structura acestui capitol va fi urmatoarea, voi decrie pe scurt cum am realizat si de ce, apoi voi anexa un exemplu de cod. Aici adaug o remarca, in Python clasele abstracte au parametru (ABC)

**Factory:** Este realizat clasic, avind o clasa abstracta – patrinte definita la inceput si clase – copii cu realizarea sa personala. L – am realizat anume pentru acest modul, deoarece anume aici avem nevoie de foarte multe obiecte tipice, asemanatoare ca structura. In continuare este prezentata o secventa de cod.

class IMenu(ABC):

    @abstractmethod

    def createMenu(self):

        pass

class WelcomeMenu(IMenu):

    def createMenu(self):

        buying = KeyboardButton('💰 Buying')

        selling = KeyboardButton('📈 Selling')

        welcomeMenu = ReplyKeyboardMarkup(resize\_keyboard = True, selective = True).row(buying, selling)

        return welcomeMenu

**Singleton:** A primit o realizare tipica a sa. El conecteaza baza de date la serviciu. In continuare este prezentata o secventa de cod.

class ConnectDB:

    \_\_instance = None

    @staticmethod

    def getInstance():

        if ConnectDB.\_\_instance == None:

            ConnectDB()

        return ConnectDB.\_\_instance

    def \_\_init\_\_(self):

        if ConnectDB.\_\_instance != None:

            raise Exception("Cannot be initialised multiple times!")

        else:

            ConnectDB.\_\_instance = self

    def connect(self):

        conn = sqlite3.connect('db/shoppyBot.db', check\_same\_thread = False)

        if conn != None:

            print ("DB reached! Connection established!")

        else:

            print("Cannot reach the DB! Please check connection!")

        return conn

**Builder:** Indeplineste o functie destul de importanta, el creeaza anuntul, preluind datele de la raspunsurile date de utilizator. Apoi, tot metoda sa “ *def push2DB(self, cursor, conn):* “ expediaza datele in BD. In continuare este prezentata o secventa de cod.

class Product(ABC):

    @abstractmethod

    def push2DB(self):

        pass

class Builder(ABC):

    @abstractmethod

    def build(self):

       pass

elif message.text == '⌛ Upload to the store':

        if isSeller:

            tempAdv = ADV\_Build().getName(itemName).getPrice(itemPrice).getCategory(cat).getSubCategory(subCat).getSeller(message.from\_user.username).getSellerPhone(cursor, conn, currUser).getDescription(itemDescr).getPhoto(itemImg).build().push2DB(cursor, conn)

**Decorator:** Functia acestui sablon este de a adauga informatie la obiectul deja existent, de aceasta si a fost folosit. In proiectul meu el adauga la anunturi deja existente label-uri cu informatie despre pret si calitate. In continuare este prezentata o secventa de cod.

class PriceDecor(Decorator):

    def \_\_init\_\_(self, status, pId):

        self.\_decorStatus = status

        self.\_pId = pId

    def decorate(self, cursor, conn):

        cursor.execute('update products set priceLabel = {} where prodId = {}'.format(self.\_decorStatus, self.\_pId))

        conn.commit()

class QualityDecor(Decorator):

    def \_\_init\_\_(self, status, pId):

        self.\_decorStatus = status

        self.\_pId = pId

    def decorate(self, cursor, conn):

        cursor.execute('update products set qualityLabel = {} where prodId = {}'.format(self.\_decorStatus, self.\_pId))

        conn.commit()

elif call == 'dp':

                menu = RResult4Seller().createMenu()

                PriceDecor(True, aId).decorate(cursor, connect)

                msg1 = await cb.message.answer('Decorated!', reply\_markup = menu)

**Proxy:** Este creeat pentru a usura lucrul pentru bazei de date. El efectueaza query la BD si stocheaza informatia in sine, cu aceasta micsorind incarcatura pe baza de date. In acest proiect acesta nu este tare observabil, insa acest este prezent pentru o oarecare dezvoltare ulterioara. In continuare este prezentata o secventa de cod.

class ProxyDB(IDataBase):

    def \_\_init\_\_(self):

        self.DB = DataBase()

        self.goodsList = None

    def storeList4Customer(self, cursor, cat, subcat):

        self.goodsList = self.DB.getList4Customer(cursor, cat, subcat)

        return self

    def storeList4Seller(self, cursor, username):

        self.goodsList = self.DB.getList4Seller(cursor, username)

        return self

**Observer:** Este folosit pentru a inregistra istoricul cautarii fiecarui utilizator. Functioneaza in felul urmator, un observator creeat de la inceput, este atasat procedurii de cautare a produselor dupa filtru. Observatorul salveaza acest filtru in baza de date. In continuare este prezentata o secventa de cod.

class searchObserver(Observer):

    def react(self, cursor, conn, uuId, cat, subcat):

        controlReq = observerHandler(cursor, uuId)

        print('Status =',controlReq)

        if controlReq == True:

            cursor.execute('update searchHistory set reqCat = ?, reqSubcat = ? where userId = {}'.format(uuId), (cat, subcat))

            conn.commit()

            print('Updated!')

        elif controlReq == False:

            cursor.execute('insert into searchHistory(userId, reqCat, reqSubcat) values (?,?,?)', (uuId, cat, subcat))

            conn.commit()

            print('Registered!')

**Strategy:** Este folosit pentru a genera o lista de anunturi promovate de serviciu pe baza anumitor criterii (label-uri si isotricul cautarii). Strategia se alege in mod random.

class PromoteByPriceLabel(Strategy):

    def execute\_alg(self, cursor, uuid):

        print('By\_Price')

        adverts = []

        out = cursor.execute('select \* from products where priceLabel = 1').fetchall()

        for data in out:

            adverts.append(data)

        return adverts

class PromoteByQualityLabel(Strategy):

    def execute\_alg(self, cursor, uuid):

        print('By\_Quality')

        adverts = []

        out = cursor.execute('select \* from products where qualityLabel = 1').fetchall()

        for data in out:

            adverts.append(data)

        return adverts

class PromoteByHistory(Strategy):

    def execute\_alg(self, cursor, uuId):

        print('By\_History')

        data = cursor.execute('select reqCat, reqSubcat from SearchHistory where userId = ?', (uuId,)).fetchall()[0]

        cat, subcat = data

        adverts = []

        out = cursor.execute('select \* from products where category = ? and subcategory = ? and qualityLabel = 1 UNION select \* from products where category = ? and subcategory = ? and priceLabel = 1', (cat, subcat, cat, subcat)).fetchall()

        for data in out:

            adverts.append(data)

        return adverts

# 3. Documentarea produsul realizat

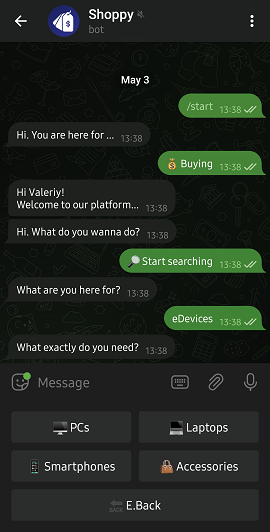
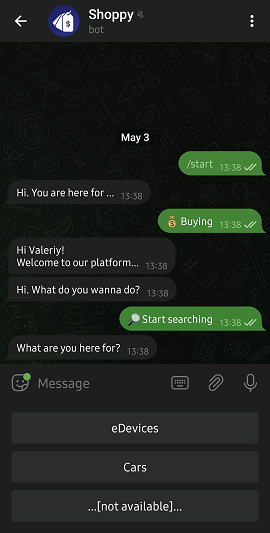
Acest capitol va fi in forma unui ghid al utilizatorului, in care se va explica si arata in screen-uri cum totul lucreaza.

De la inceput, ca si orice bot, acesta trebuie lansat de catre utilizator. Aceasta se face destul de intuitiv, apasind pe butonul START, pe asa numita pagina principala a sa,  *Fig.15 Welcome Screen.*

  
*Fig.15 Welcome Screen*

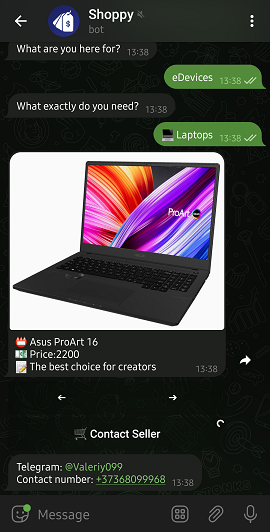
Dupa ce lansam botul avem in fata un meniu care ne pune sa alegem ce dorim sa facem aici. Aceasta alegere ne da acces la functionalul prevazut pentru acest tip de utilizator.

Daca alegem “💰 Buying”, automat devenim cumparatori. De aici ne apar 2 alegeri, incepem a cauta de ceea ce avem nevoie sau ne uitam la ce ne propune serviciul. In aceast alineat, vorbim despre prima. Pentru aceasta este nevoie sa tastam pe butonul “🔎 Start searching”. Dupa ce apasam acest buton, serviciul ne adreseaza o serie de intrebari, ca sa precizeze ce anume dorim. Acesta ne trece printr 2 filtre, ceva general si ceva mai precis. Dupa setarea ultimului, botul ne arata produsele care corespund cerintelor noastre. Aceasta este ilustrata in *Fig. 16 Procesul de cautare de catre utilizator*



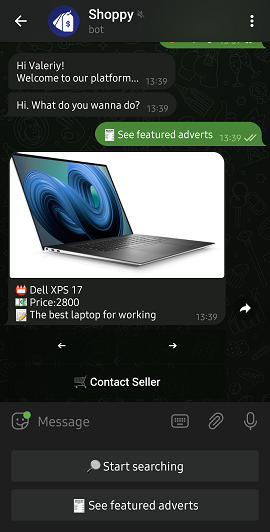
*Fig. 16 Procesul de cautare de catre utilizator*

In *fig.18* putem observa sub produs niste butoane cu sageti. Acestea nu sunt ceva, decit o carusela care ne permite rulam lista cu produse. Apasind pe butonul “🛒 Contact Seller” primim datele vinzatorului care a postat acest produs. Aceasta este ilustrat in *Fig. 17 Rezultatul rularii caruselei.*

*  
Fig. 17 Rezultatul rularii caruselei*

Acum, vorbim despre a 2-a varianta de vizionare a produsului. Tastind butonul

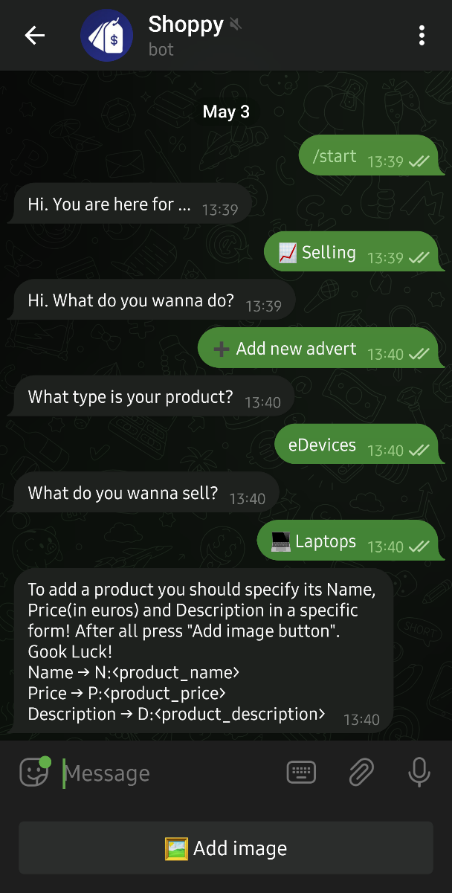
“🧾 See promoted adverts”, daca la general primim acelasi rezultat ca si la cautarea sofisticata, adica aceeasi carusela. Doar ca aici nu putem controla ce produse ne vor fi aratate. Lista produselor care ne returneaza serviciul este dirijat de un algoritm. Acest algoritm ne poate returna produse luind in consideratie ultima configurare de cautare sau ne poate returna produse promovate de catre utilizator, ce se poate de analizat in *Fig. 18 Lista produselor promovate.*



*Fig. 18 Lista produselor promovate*

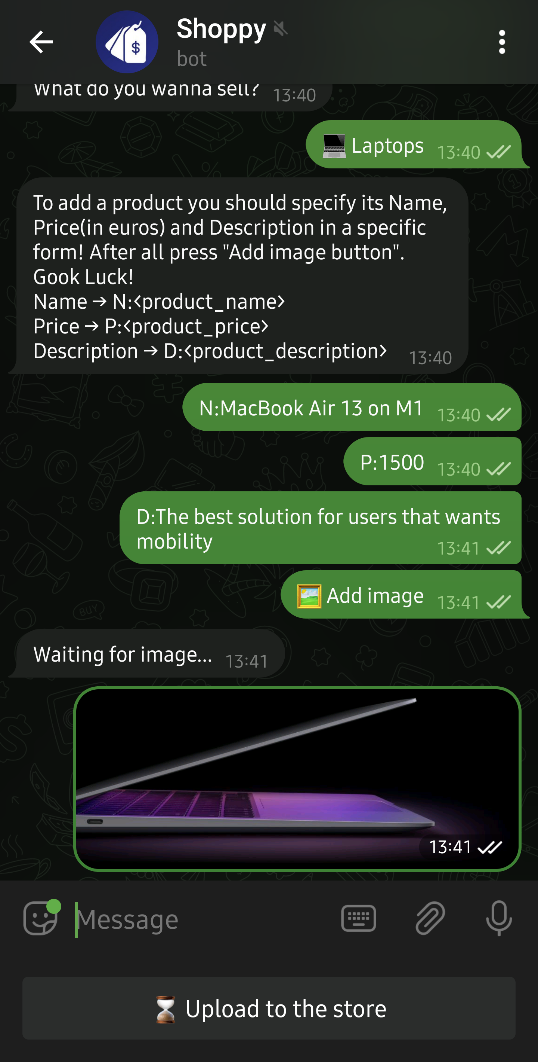
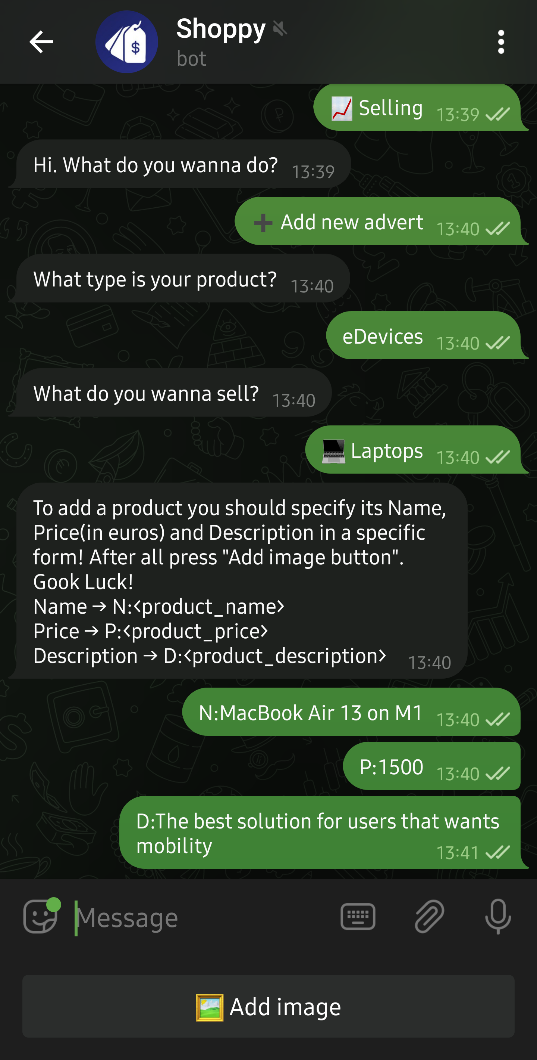
De aici, voi vorbi despre functionalul de vinzator. Pentru aceasta de la bun inceput trebuie sa tastam butonul “📈 Selling”. Dupa aceasta ne apar 2 variante, sa adaugam un anunt nou si sa ne vizualizam produsele adaugate anterior. In caz de sunteti un utilizator nou, veti fi rugat sa va inregistrati in sistem. Pentru aceasta este nevoie de urmari instructiunile. Aceasta este ilustrat in *Fig.19 Logarea ca utilizator, filtrele si instructiunea de adaugare a produsului nou*

Procedeul de adaugare a unui produs nou consta din mai multi pasi, pe parcursul carora suntem ghidati de bot. Pentru inceput, e nevoie sa stabilim tipul produsului care dorim sa-l vindem. Pentru aceasta trecem prin fix aceleasi filtre ca si cumparatorul. Dupa aceasta primim o instructiune cum trebuie sa adaugam un produs nou. Aceasta este ilustrat in *Fig.19 Logarea ca utilizator, filtrele si instructiunea de adaugare a produsului nou*



*Fig.19 Logarea ca utilizator, filtrele si instructiunea de adaugare a produsului nou*

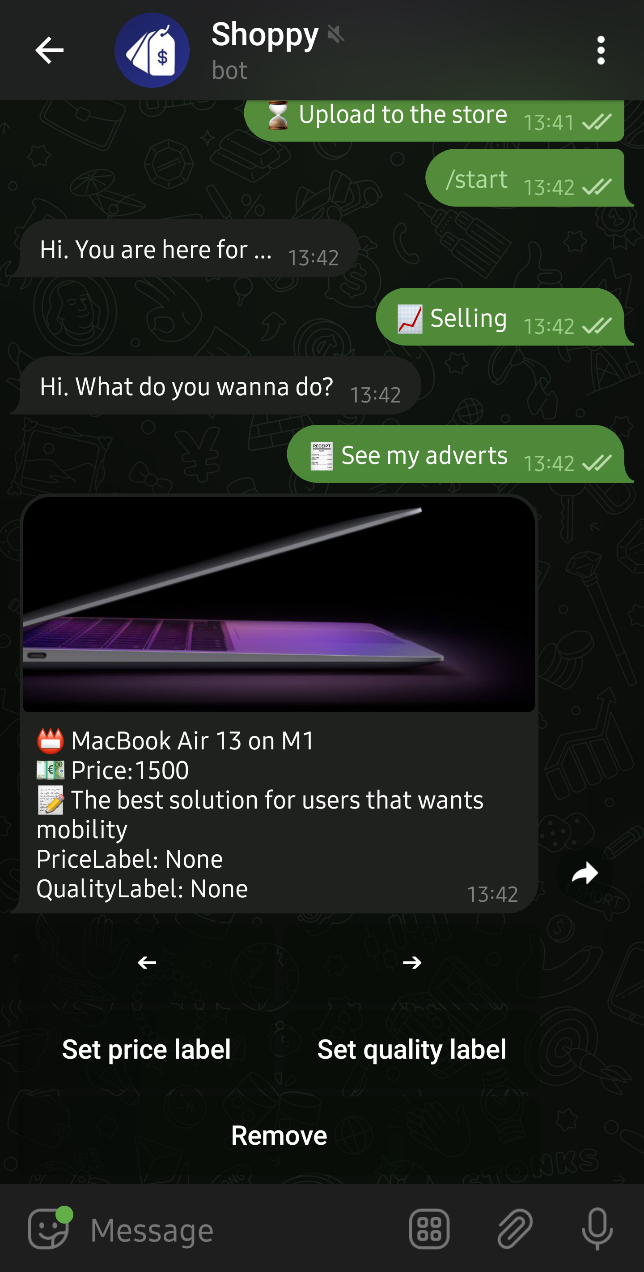
Acum este nevoie sa introducem datele. Aceasta este reprezentata in *Fig. 20 Procedeul de adaugare a produsului nou*. Pentru a adauga imaginea este nevoie sa tastam butonul “”🖼️ Add image. Dupa aceasta tastam butonul “⌛ Upload to the store “ pentru a incarca datele pe serverul magazinului.



*Fig. 20 Procedeul de adaugare a produsului nou*

Pentru final, citeva cuvinte despre functionalitatile de management a anunturilor pentru vinzator.

Tastind pe butonul “🧾 See my adverts”, acesta primeste lista cu anunturile sale. Aici vedem citeva butoane sub anunt. Acelea cu sageti ne sunt deja cunoscute, raspund de carusela. Urmatoarele 2 permit vinzatorului sa-si promoveze produsul. Utlimul, permite vinzatorului sa-si stearga produsul.



*Fig.23 Vizualizarea produselor proprii*

# Concluzii

In final mi-a reusit sa elaborez un produs finit, gata pentru exploatare. Este o platforma de e-commerce bazata pe mesengerul Telegram. Nu pot afirma ca este super comoda, deoarece mai este de lucru in ea, dar in general e destul de comoda. Printre minusurile acestui servicu, pot ilustra baza de date, deoarece este una de viteza mica, care in viitor ar putea fi inlocuita cu una mai rapida.

Back-endul acesteia este scris in Python cu folosirea sabloanelor de proiectare. Am folosit asa sabloane ca *Factory, Builder, Decorator, Observer, Strategy, Proxy*. Nu a fost greu sa le realizez in Python pentru functionalul planificat insa a fost inutil, ba mai mult e pierdere de timp utilizarea pe larg a sabloanelor de proiectare, cel putin in limbajele nu 1000% OOP cum ar fi Python sau JavaScript. In cele mai dese ori, avem nevoie de obiect cind merge vorba de Multithreading, adica cind avem nevoie de anume un obiect customizat sub fiecare utilizator aparte, care acceseaza concomitent functionalul dat, ca de exemplu la mine carusela, aici chiar, fara oformare de clasa nu e posibil altfel, in rest, totul putea fi oformat doar cu metode, functii, fara creearea obiectelor si haosuli in cod care a venit odata cu dinsele.

**Greutati intilnite:** realizarea acestui proiect utilizind sabloanele de proiectare este, asa sa zicem, un pic absurd, cel putin pentru mine si limbajul in care am realizat proiectul. De ce anume asa? Este absolut aceeasi daca as lasa in mare parte doar metodele. Aici nu este cazul sa avem o sumedenie de obiecte, spre exemplu pentru meniuri. Da, este necesara aceasta abordare, in cazul in care avem nevoie de creea multi threading pentru diferite operatiuni, spre exemlu postarea unui anunt nou, aici fiecare utilizator nou intrat trebuie sa-si aiba obiectul sau pentru a nu confunda informatia. Dar la ce anume ma refer aici, OOP este necesar si este pe alocuri CEA mai buna metoda de a scrie codul, dar nu intitdeauna cu folosirea la pattern-uri. De ce? Deoarece aduce haos in cod si multimea ceea de clase, care comunica intre ele, este inutila fiindca putem creea doar o clasa, sau ok, mai multe clase care vor fi separate si fiecare ca lucra independent, si va fi unita de alta doar prin datele care le va primi de la precedenta. Asta, apropo permite sa modificam orice clasa cum dorim in procesul initial de elaborarea a back-endului, fara a atinge celelante clase si a le perturba lucrul.

Daca la general, acest proiect mi-a fost de folos ca pentru un viitor Python developer, deoarece in decursul realizarii acestuia mi-am dezvoltat noi abilitati de programare in Python si m-am aprofundat mai mult in acest limbaj ceea ce mi-a adus cunostinte noi despre mecanicile acestuia. La fel nu sa nu mentionez faptul ca m-am perfectionat in OOP, ceea ce la fel mi-a deschis ochii pe unele subiecte in software development. Si principalul am invat ce este un sablon de proiectare, am inteles cind e bine sa le folosesc si cind mai bine fara dinsele, am insusit principiile SOLID, care fac viata unui programator cu mult mai usoara si orice cod scris conform acestora mai clar. In final, cu toata modestia mea, ca am progresat insemnificativ in domeniul de development.

# Bibliografie

1. Resursa electronica: <https://en.wikipedia.org/wiki/E-commerce> – Regim de acces

2. Resursa electronica: <https://www.the-future-of-commerce.com/2020/01/19/what-is-e-commerce-definition-examples/> – Regim de acces

3. Resursa electronica: <https://en.wikipedia.org/wiki/Alibaba_Group> – Regim de acces

4. Resursa electronica: <https://en.wikipedia.org/wiki/Amazon_(company)> – Regim de acces

5. Resursa electronica: <https://en.wikipedia.org/wiki/EBay>

6. Resursa electronica: [<https://en.wikipedia.org/wiki/AliExpress>)](https://en.wikipedia.org/wiki/Amazon_(company))

7. Resursa electronica: <https://en.wikipedia.org/wiki/Etsy>

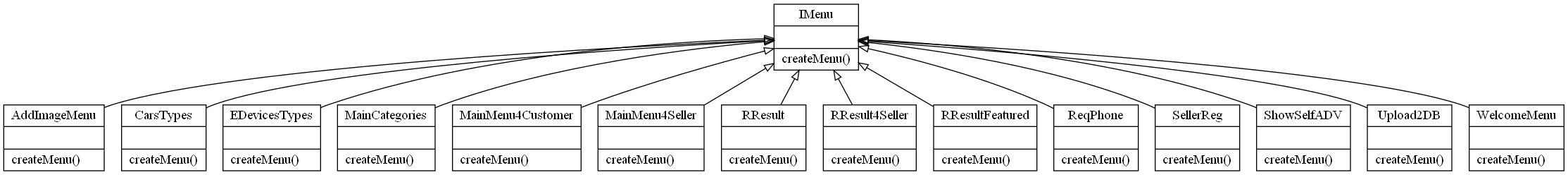
8. Resursa electronica: *Unknown*

9. Resursa electronica: *Unknown*

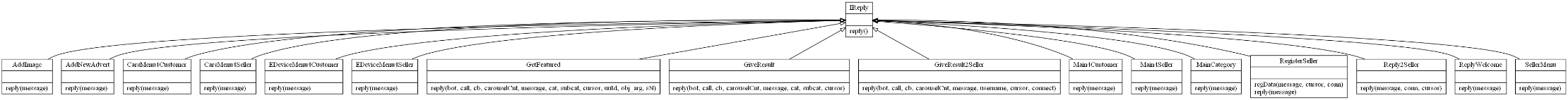
10. Resursa electronica: <https://docs.aiogram.dev/en/latest/index.html> - Regim de acces

11. Resursa electronica: <https://www.python.org/doc/essays/blurb/> - Regim de acces

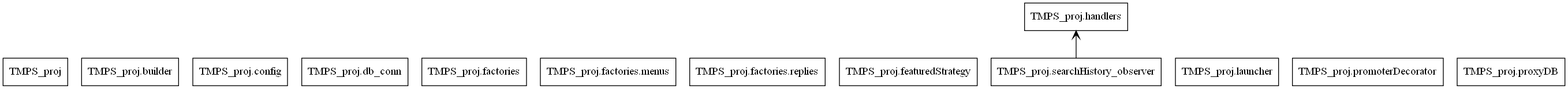
# Anexa A



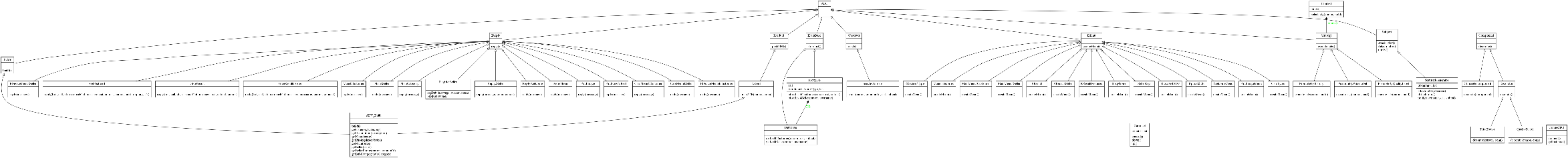
*Fig.9.1 Diagrama pentru* ***Facotry*** *compatrimentul Meniuri*



*Fig.9.2 Diagrama pentru* ***Factory*** *compartimentul Raspunsuri*

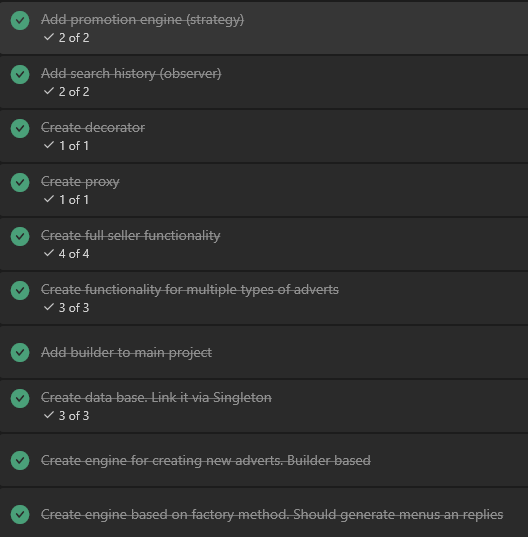
**

*Fig.27 Toate pachetele din acest proiect*

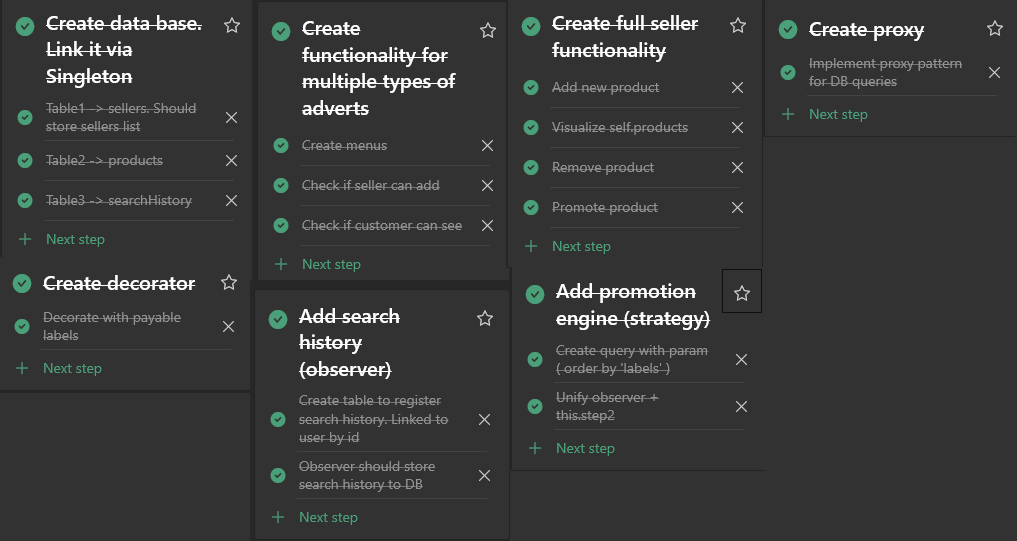


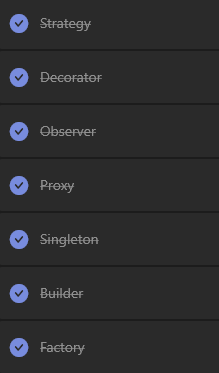
*Fig.28 Diagrama generala a proiectului*

# Anexa B

**

*Fig.29 Task list general*

*******Fig.30 Task-urile pentru fiecare compartiment aparte*

*  
Fig.31 Sabloanele planificate pentru utilizare*