CS:GO TRADE-UP SCRAPPER

NICOLESCU ROBERT ALEXANDRU

CE ESTE CS:GO?

CS:GO este un joc:

- Multiplayer

CE ESTE CS:GO?

CS:GO este un joc:

- Multiplayer
- Shooter

CE ESTE CS:GO?

CS:GO este un joc:

- Multiplayer
- Shooter
- De Strategie

CE ESTE CS:GO?

CS:GO este un joc:

- Multiplayer
- Shooter
- De Strategie

CU CE LUCRAM?

- Skin-uri

CU CE LUCRAM?

- Skin-uri
- Cutii

CU CE LUCRAM?

- Skin-uri
- Cutii
- Raritati

CU CE LUCRAM?

- Skin-uri
- Cutii
- Raritati
- Calitati

CU CE LUCRAM?

- Skin-uri
- Cutii
- Raritati
- Calitati
- Float ?

CU CE LUCRAM?

- Skin-uri
- Cutii
- Raritati
- Calitati
- Float ?
- Trade-up Contract

CE ESTE UN SKIN?

CE ESTE UN SKIN?

Jocul ne permite sa ne desenam armele, iar aceste desene se numesc skin-uri.

Practic reprezinta textura pe care o incarca jocul pe arma.

CE ESTE UN SKIN?

Jocul ne permite sa ne desenam armele, iar aceste desene se numesc skin-uri.

Practic reprezinta textura pe care o incarca jocul pe arma.

Skin Default:

CE ESTE UN SKIN?

Jocul ne permite sa ne desenam armele, iar aceste desene se numesc skin-uri.

Practic reprezinta textura pe care o incarca jocul pe arma.

Skin Default:

CE ESTE UN SKIN?

Jocul ne permite sa ne desenam armele, iar aceste desene se numesc skin-uri.

Practic reprezinta textura pe care o incarca jocul pe arma.

Skin Default:

Skin custom:

CE ESTE UN SKIN?

Jocul ne permite sa ne desenam armele, iar aceste desene se numesc skin-uri.

Practic reprezinta textura pe care o incarca jocul pe arma.

Skin Default:

Skin custom:

CE ESTE UN SKIN?

Acestea se impart dupa mai multe criterii:

CE ESTE UN SKIN?

Acestea se impart dupa mai multe criterii:

- Raritate

CE ESTE UN SKIN?

Acestea se impart dupa mai multe criterii:

- Raritate
- Calitate

CE ESTE O CUTIE?

- O cutie contine mai multe skin-uri;
- Se lanseaza cate una noua la aprox. 4 luni, nu exista o regula;
- Toate cutiile se numesc diferit

CE ESTE O CUTIE?

- O cutie contine mai multe skin-uri;
- Se lanseaza cate una noua la aprox. 4 luni, nu exista o regula;
- Toate cutiile se numesc diferit

CE ESTE O RARITATE?

Skin-urile sunt impartite pe raritati, care sunt asociate si cu diverse culori:

CE ESTE O RARITATE?

Skin-urile sunt impartite pe raritati, care sunt asociate si cu diverse culori:

- Covert - rosu

CE ESTE O RARITATE?

Skin-urile sunt impartite pe raritati, care sunt asociate si cu diverse culori:

- Covert - rosu
- Classified - roz

CE ESTE O RARITATE?

Skin-urile sunt impartite pe raritati, care sunt asociate si cu diverse culori:

- Covert - rosu
- Classified - roz
- Restricted – mov

CE ESTE O RARITATE?

Skin-urile sunt impartite pe raritati, care sunt asociate si cu diverse culori:

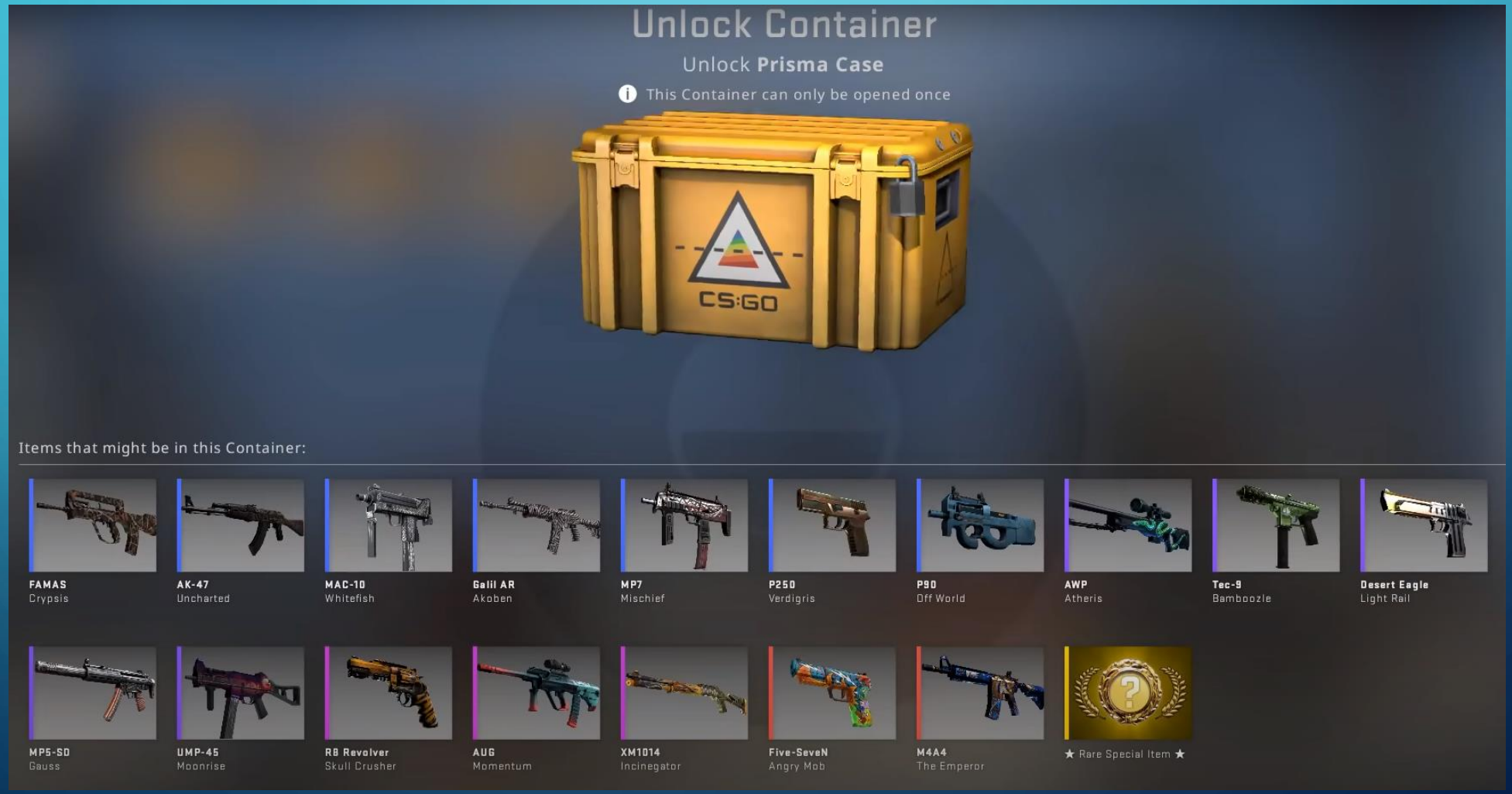
- Covert - rosu
- Classified - roz
- Restricted – mov
- Mil-Spec - albastru

CE ESTE O RARITATE?

Skin-urile sunt impartite pe raritati, care sunt asociate si cu diverse culori:

- Covert - rosu
- Classified - roz
- Restricted – mov
- Mil-Spec - albastru

CE ESTE O RARITATE?



CUM FUNCTIONEAZA O CUTIE?

CUM FUNCTIONEAZA O CUTIE?

CUM FUNCTIONEAZA O CUTIE?

CUM FUNCTIONEAZA O CUTIE?

CE ESTE O CALITATE?

Calitate ≠ Raritate

Cand ‘ne pica’ un skin dintr-o cutie, aceasta poate sa vina mai mult sau mai putin zgariata. Iar nivelul de afectare al skin-ului determina calitatea sa.

CARE SUNT CALITATILE?

CARE SUNT CALITATILE?

Factory New (FN)

CARE SUNT CALITATILE?

Factory New (FN)

Minimal Wear (WN)

CARE SUNT CALITATILE?

Factory New (FN)

Minimal Wear (WN)

Field-Tested (FT)

CARE SUNT CALITATILE?

Factory New (FN)

Minimal Wear (WN)

Field-Tested (FT)

Well-Worn (WW)

CARE SUNT CALITATILE?

Factory New (FN)

Minimal Wear (WN)

Field-Tested (FT)

Well-Worn (WW)

Battle-Scared (BS)

CE ESTE FLOAT-UL?

Pe scurt, float-ul este valoarea exacta a gradului de zgariere al unui skin.

Valoarea este cuprinsa in intervalul [0,1].

0 – 0.07 = FN

…

0.45 - 1 = BS

CE ESTE FLOAT-UL?

Factory New

Battle-Scarred

CE ESTE UN TRADE-UP CONTRACT?

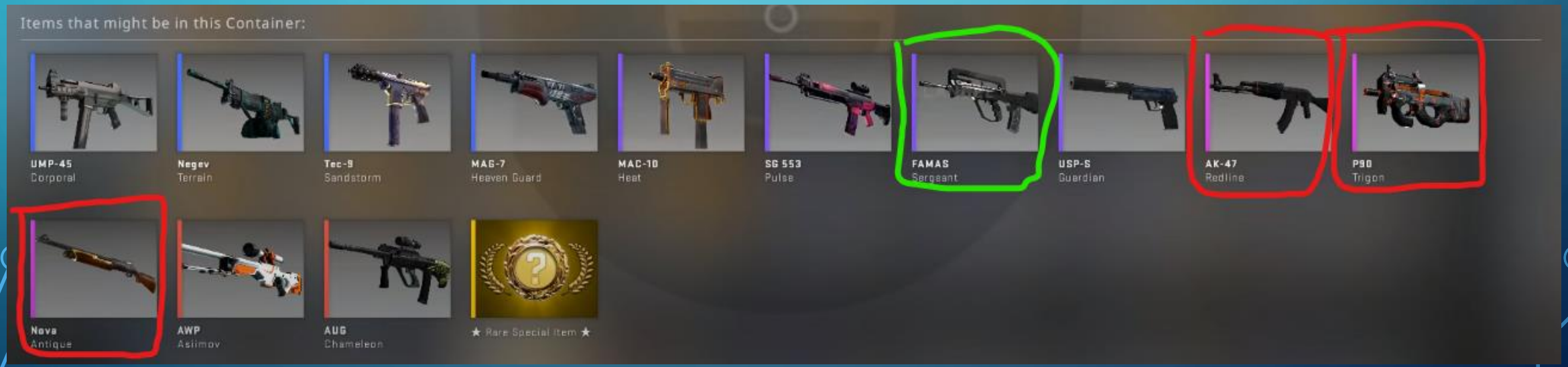
Un trade-up contracat este un trade pe care il facem cu jocul, in care noi **TREBUIE** sa punem **10** skin-uri de **ACEEASI RARITATE** (nu neaparat din aceeasi cutie, sau de aceeasi calitate), si avem sansa sa primi **UN** skin cu **O RARITATE MAI MARE** decat cea a acestora 10.

DE UNDE STIM CE SKIN-URI POT ‘PICA’?

Cazul cel mai usor: toate cele 10 skin-uri sunt din aceeasi cutie.

DE UNDE STIM CE SKIN-URI POT ‘PICA’?

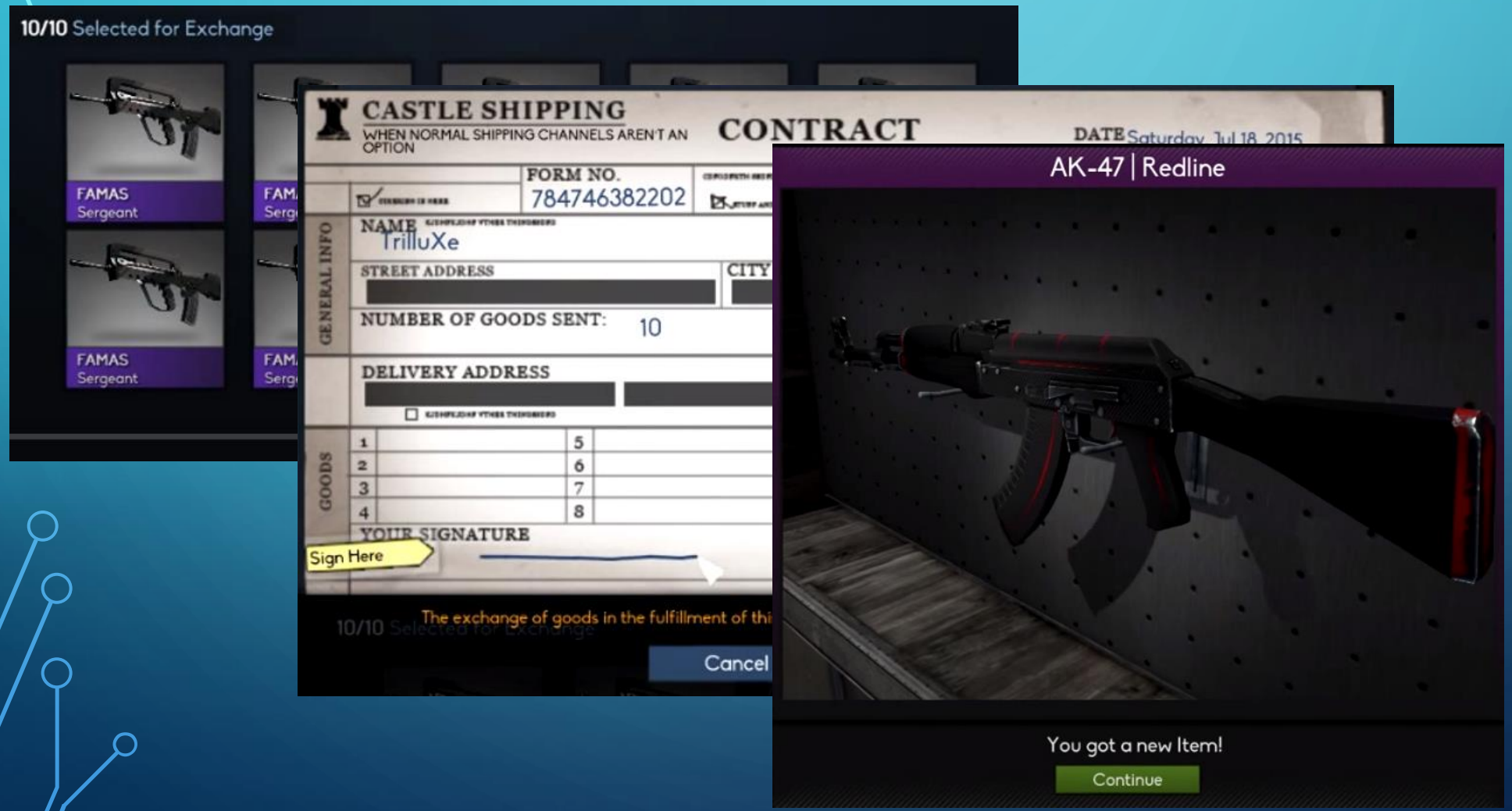
Cazul cel mai usor: toate cele 10 skin-uri sunt din aceeasi cutie.



EXEMPLU TRADE-UP CONTRACT

EXEMPLU TRADE-UP CONTRACT

EXEMPLU TRADE-UP CONTRACT



DE UNDE STIM CE CALITATE VA AVEA

Cele 10 skin-uri **trebuie** sa aibe aceeasi **raritate**, dar pot avea **CALITATI DIFERITE;**

Calitatea skin-ului primit la contract este determinata de o formula mai ciudata ce are la baza float-urile skin-urilor adaugate de noi.

$$\text{TRADE UP FLOAT VALUE} = (\text{MAX FLOAT} - \text{MIN FLOAT}) \times \text{AVG} + \text{MIN FLOAT}$$

DE CE AM EXPLICAT TOATE ASTEA?

Aceste skin-uri pot fi cumparate de pe o multime de site-uri, cu bani reali, iar toti acesti parametri influenteaza pretul unui skin.

Scopul programului meu este de a gasi oferte de trade-up contract profitabile, sau macar incearca.

CUM FACE ASTA?

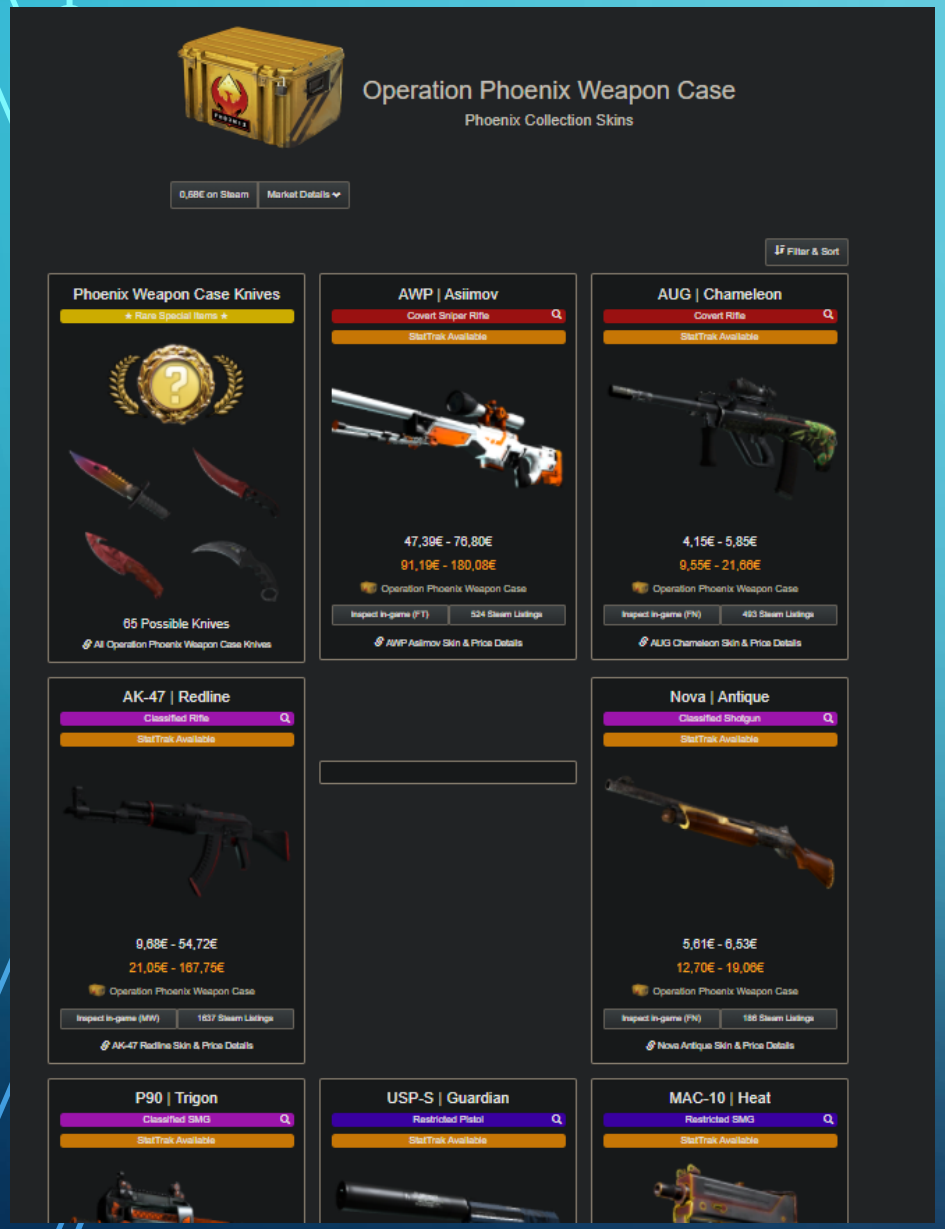
1. Web Scrapping pe CS:GO Stash
2. Web Scrapping pe Steam Analyst
3. Algoritm de gasirea contractului profitabil

1. WEB SCRAPPING PE CS:GO STASH


```python
def extract_from_cases(URL, text_Var, window):
    page = requests.get(URL)
    soup = bs(page.content, features='html.parser')

    menu = soup.find('div', {'id': 'navbar-expandable'}).find('ul')
    all_buttons = menu.findAll('li', {'class': 'dropdown'})

    ok = 0
    for button in all_buttons:

        menu = button.findAll('a', {'href': '#'})

        # Cand gasesc butonul 'Cases' ok devine 1 si for-ul se opreste
        for row in menu:
            if (row.contents[0] == 'Cases'):
                ok = 1
                break

        if ok == 1:
            break

    elem_menu = button.findAll('li')  # accesez
```

```python
def extract_case_link_name(data):
    return (data.find('a').get('href'), data.find('img').get('alt'))


# pastrez doar diviziunile cu date despre cutii
# adica fara datele extrase gresit
def filter_cases(case_text):

    img_stat = case_text.find('img') is not None
    if not img_stat:
        return False

    case_name = case_text.find('img').get('alt')
    except_1 = case_name != 'All Skin Cases'
    except_2 = case_name != 'Souvenir Packages'
    except_3 = case_name != 'Gift Packages'

    if except_1 and except_2 and except_3:
        return True

def extract_data(data):
    data_filtered = set(filter(filter_cases, data))
    return set(map(extract_case_link_name, data_filtered))
```

1. WEB SCRAPPING PE CS:GO STASH

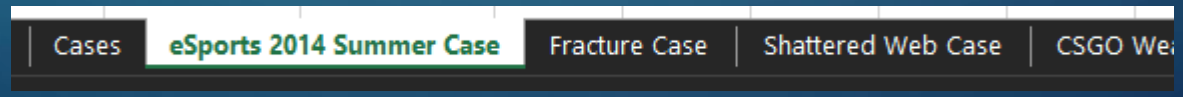



```
def extract_skin_from(URL, ObjCase, text_Var, window):
    page = requests.get(URL)
    soup = bs(page.content, features='html.parser')

    case_name = soup.find('h1', {'class': 'margin-top-sm'}).text

    skin_boxes = soup.findAll('div', {'class': 'col-lg-4 col-md-6 col-widen text-center'})


    # nu iau prima fereastra pentru ca este cu cutite, si nu poti face trade cu ele
    for box in skin_boxes[1:]:
        ObjSkin = cls.Skin()

        # extrage numele skinului si al armei, separat
        ObjSkin.weapon = box.find('h3').text.split(" | ")[0]
        ObjSkin.name = box.find('h3').text.split(" | ")[1]
        ObjSkin.rarity = box.find('p', {'class': 'nomargin'}).text.split(' ')[0]
        ObjCase.Skins.append(ObjSkin)

        # salvez imaginea skin-ului pentru a o folosi in interfata
        image = box.find('img')
        image_url = image['src']
        image_final = Image.open(requests.get(image_url, stream=True).raw)
        image_final.save('Files/Skins/' + ObjSkin.weapon + ' ' + ObjSkin.name + '.png', 'PNG')

        # afisez in log viewer detalii despre skin, ca sa arat ca programul functioneaza

        print_to_log('Extracted skin: ' + ObjSkin.name + ' | ' + ObjSkin.weapon + ' from case: ' + ObjCase.name,
                     text_Var, window)
```

1. WEB SCRAPPING PE CS:GO STASH

	A	B
13	Revolver Case	https://csgostash.com/case/111/Revolver-Case
14	eSports 2013 Case	https://csgostash.com/case/2/eSports-2013-Case
15	Clutch Case	https://csgostash.com/case/238/Clutch-Case
16	Operation Vanguard Weapon Case	https://csgostash.com/case/29/Operation-Vanguard-Weapon-Case
17	Chroma 3 Case	https://csgostash.com/case/141/Chroma-3-Case
18	Danger Zone Case	https://csgostash.com/case/259/Danger-Zone-Case
19	Winter Offensive Weapon Case	https://csgostash.com/case/7/Winter-Offensive-Weapon-Case
20	eSports 2014 Summer Case	https://csgostash.com/case/19/eSports-2014-Summer-Case
21	Gamma 2 Case	https://csgostash.com/case/172/Gamma-2-Case
22	Gamma Case	https://csgostash.com/case/144/Gamma-Case
23	CS:GO Weapon Case 3	https://csgostash.com/case/10/CS:GO-Weapon-Case-3
24	Spectrum 2 Case	https://csgostash.com/case/220/Spectrum-2-Case
25	Spectrum Case	https://csgostash.com/case/207/Spectrum-Case
26	Operation Phoenix Weapon Case	https://csgostash.com/case/11/Operation-Phoenix-Weapon-Case
27	Prisma Case	https://csgostash.com/case/274/Prisma-Case
28	Operation Broken Fang Case	https://csgostash.com/case/308/Operation-Broken-Fang-Case
29	Prisma 2 Case	https://csgostash.com/case/303/Prisma-2-Case
30	Chroma Case	https://csgostash.com/case/38/Chroma-Case
31	Operation Bravo Case	https://csgostash.com/case/3/Operation-Bravo-Case
32	Huntsman Weapon Case	https://csgostash.com/case/17/Huntsman-Weapon-Case
33	Shattered Web Case	https://csgostash.com/case/277/Shattered-Web-Case
34	Operation Hydra Case	https://csgostash.com/case/208/Operation-Hydra-Case
35	Shadow Case	https://csgostash.com/case/80/Shadow-Case
36		

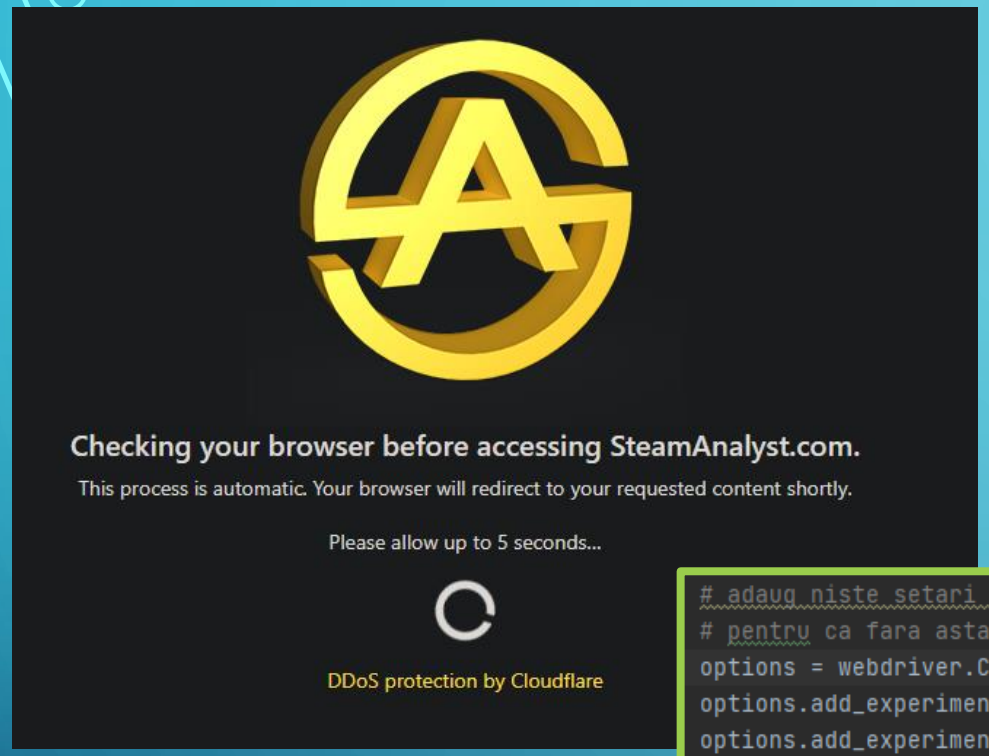
	A	B	C
1	Skin	Weapon	Rarity
2	Jaguar	AK-47	Covert
3	Bullet Rain	M4A4	Covert
4	Corticera	AWP	Classified
5	Bengal Tiger	AUG	Classified
6	Bloomstick	Nova	Classified
7	Corticera	P2000	Classified
8	Crimson Web	Desert Eagle	Restricted
9	Ocean Foam	MP7	Restricted
10	Blue Streak	PP-Bizon	Restricted
11	Steel Disruption	Glock-18	Restricted
12	Virus	P90	Restricted
13	Ultraviolet	MAC-10	Mil-Spec
14	Bratatat	Negev	Mil-Spec
15	Blood Tiger	USP-S	Mil-Spec
16	Hexane	CZ75-Auto	Mil-Spec
17	Dark Water	SSG 08	Mil-Spec
18	Red Python	XM1014	Mil-Spec



1. WEB SCRAPPING PE CS:GO STASH

```python
import xlsxwriter


# stat_prices = ii spun functiei daca are de salvat si preturi, sau nu
def to_xlsx(Cases, file_name, stat_prices):
    # Generez fisierul Excel
    wb = xlsxwriter.Workbook(file_name)

    # region Sheet Cases

    # Generez sheet pentru Cutii
    sheet_cases = wb.add_worksheet('Cases')

    # Adaug numele coloanelor: Case Name, Link
    sheet_cases.write(0, 0, 'Case Name')
    sheet_cases.write(0, 1, 'Link')

    # Adaug datele in coloane
    for index, case in enumerate(Cases):

        sheet_cases.write(index + 1, 0, case.name)

        if stat_prices == False:
            sheet_cases.write(index + 1, 1, case.link)
    # endregion

    # Generez sheet pentru fiecare Cutie in care salvez toate skin-urile din ea
    for case in Cases:
        # sheet_SingleCase
        sheet_scase = wb.add_worksheet(case.name.replace(':', ''))

        # adaug numele coloanelor
        sheet_scase.write(0, 0, 'Skin')
        sheet_scase.write(0, 1, 'Weapon')
        sheet_scase.write(0, 2, 'Rarity')

        if stat_prices == True:
            # Fara ST
            sheet_scase.write(0, 3, 'FN')
            sheet_scase.write(0, 4, 'MW')
            sheet_scase.write(0, 5, 'FT')
            sheet_scase.write(0, 6, 'WW')
            sheet_scase.write(0, 7, 'BS')

            # Cu ST
            sheet_scase.write(0, 8, 'SFN')
            sheet_scase.write(0, 9, 'SMW')
            sheet_scase.write(0, 10, 'SFT')
            sheet_scase.write(0, 11, 'SWW')
            sheet_scase.write(0, 12, 'SBS')

        # adaug numele si datele despre skin-uri
        for index, skin in enumerate(case.Skins):
            sheet_scase.write(index + 1, 0, skin.name)
            sheet_scase.write(index + 1, 1, skin.weapon)
            sheet_scase.write(index + 1, 2, skin.rarity)

            if stat_prices == True:
                # adaug preturile pt fiecare calitate a armei
                for i in range(0, 10):
                    if skin.prices[i] != -1:
                        sheet_scase.write(index + 1, i + 3, skin.prices[i])

    # Salvez Fisierul
    wb.close()
```

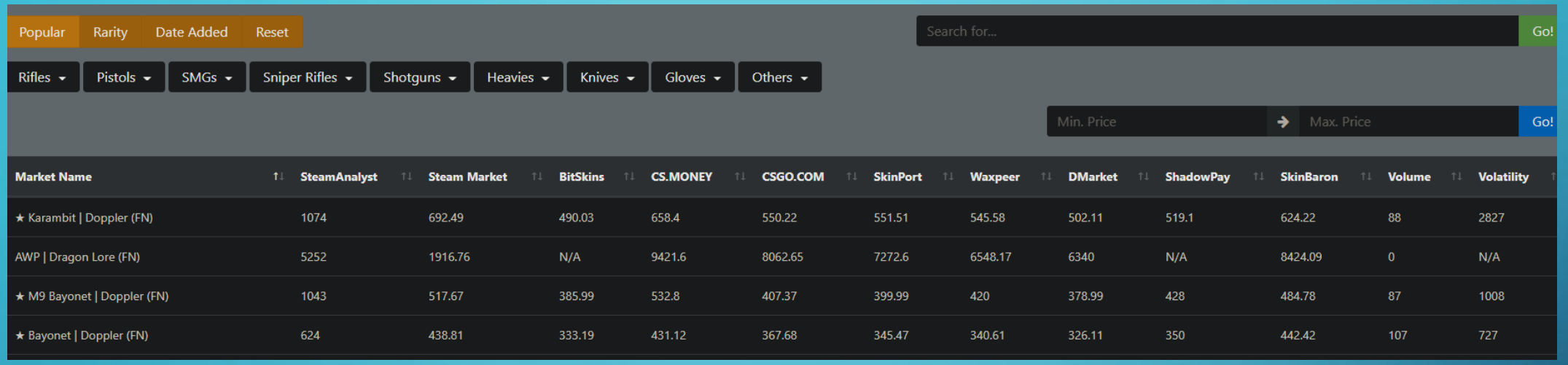
2. WEB SCRAPPING PE STEAM ANALYST




```python
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.common.exceptions import NoSuchElementException
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
import time
```

```python
# adaug niste setari care sa faca Chrome-ul sa nu se mai deschida in forma de testare (asa cum deschide Selenium)
# pentru ca fara asta nu as putea trece de verificarea Cloudflare
options = webdriver.ChromeOptions()
options.add_experimental_option("excludeSwitches", ["enable-automation"])
options.add_experimental_option('useAutomationExtension', False)
options.add_argument("--disable-blink-features=AutomationControlled")
driver = webdriver.Chrome(options=options)

driver.get('https://csgo.steamanalyst.com/markets')

# Am nevoie de 13 secunde sa creez un nou tab de chrome in care sa deschid acelasi link
# Pe acel nou tab voi trece de verificarea Cloudflare, si dupa se va incarca pagina si pe tabul de lucru
# iar programul poate rula fara probleme
time.sleep(13)
```

2. WEB SCRAPPING PE STEAM ANALYST



Market Name	SteamAnalyst	Steam Market	BitSkins	CS.MONEY	CSGO.COM	SkinPort	Waxpeer	DMarket	ShadowPay	SkinBaron	Volume	Volatility
★ Karambit | Doppler (FN)	1074	692.49	490.03	658.4	550.22	551.51	545.58	502.11	519.1	624.22	88	2827
AWP | Dragon Lore (FN)	5252	1916.76	N/A	9421.6	8062.65	7272.6	6548.17	6340	N/A	8424.09	0	N/A
★ M9 Bayonet | Doppler (FN)	1043	517.67	385.99	532.8	407.37	399.99	420	378.99	428	484.78	87	1008
★ Bayonet | Doppler (FN)	624	438.81	333.19	431.12	367.68	345.47	340.61	326.11	350	442.42	107	727

```
def search_skin(Cases, driver, displayVar, window):
    for case in Cases:
        for skin in case.Skins:

            # initializez toate preturile skinului cu -1
            for i in range(0, 11):
                skin.prices.append(-1)

            input_element = driver.find_element_by_id('searchInput')
            input_element.clear()
            input_element.send_keys(skin.weapon + ' ' + skin.name)
            input_element.send_keys(Keys.ENTER)

            # Astept sa se incarce pagina (adica sa apara numele skin-ului in prima caseta din tabel)
            WebDriverWait(driver, 120).until(EC.text_to_be_present_in_element((By.XPATH, '/html/body/div[22]/div/div/div[4]/div/div/div[2]/div/table/tbody/tr[1]'), skin.name))

            # fac media preturilor de pe site-uri si o adaug lui skin, la calitatea care trebuie
            for i in range(1, 11):
                try:
                    table_row = driver.find_element_by_xpath(
                        '/html/body/div[22]/div/div/div[4]/div/div/div[2]/div/table/tbody/tr[' + str(
                            i) + ']').text
                    data = table_row.split(' ')
```

2. WEB SCRAPPING PE STEAM ANALYST

```
    # numar cate caractere sar pentru a ajunge la preturi
    count = 0
    for index, elem in enumerate(data):
        if '(' in elem:
            count = index + 1
            break

    if 'Dragon King' in table_row:
        count += 2

    plus1 = 0
    if 'stattrak' in data[0].lower():
        plus1 += 1

    # adaug pretul la CALITATEA care trebuie
    if '(FN)' in table_row:
        skin.prices[plus1 * 5 + 0] = data[count]
    if '(MW)' in table_row:
        skin.prices[plus1 * 5 + 1] = data[count]
    if '(FT)' in table_row:
        skin.prices[plus1 * 5 + 2] = data[count]
    if '(WW)' in table_row:
        skin.prices[plus1 * 5 + 3] = data[count]
    if '(BS)' in table_row:
        skin.prices[plus1 * 5 + 4] = data[count]

    print_to_log(skin.weapon + ' | ' + skin.name + ' sells for ' + data[count] + '$ on average.', displayVar, window)

except NoSuchElementException:
    print(skin.name + ' nu se gaseste pe atatea calitati.')
```

2. WEB SCRAPPING PE STEAM ANALYST

Aceeasi structura ca fisierul cu nume, doar ca acum are si preturile

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Skin	Weapon	Rarity	FN	MW	FT	WW	BS	SFN	SMW	SFT	SWW	SBS
2	Hyper Bea	AWP	Covert	81.19	46.38	30.14	24.54	20.21	262.82	122.04	70.9	48.83	46.64
3	Aquamari	AK-47	Covert	63.4	43.48	29.87	24.25	19.68	181.93	114.93	69.86	53.42	41.7
4	Cyrex	SG 553	Classified	15.85	6.72	3.87	3.02	2.96	59.92	22.57	10.46	7.02	7.16
5	Nemesis	MP7	Classified	8.07	5.03	3.85			28.05	12.96	8.11		
6	Yellow Jac	CZ75-Autc	Classified	10.62	5.24	3.41	2.8	2.86	30.59	12.9	7.93	6.24	5.58
7	Evil Daimy	M4A4	Restricted	7.39	4.67	3.35	4.01	3.5	21.36	12.65	7.93	8.86	8.26
8	Loudmout	Negev	Restricted		4.39	0.41	0.66	0.38		16.59	1.39	2.08	1.07
9	Ruby Pois	MP9	Restricted	2.23	1.31	0.7	1.03	0.84	7.68	4.53	2.28	2.66	2.26
10	Handgun	P2000	Restricted	3.06	0.78	0.42	0.38	0.39	9.89	2.94	1.22	1.04	1.06
11	Neural Ne	FAMAS	Restricted	1.87	1.16	0.69	1.03	0.77	5.58	3.3	1.66	2.29	1.75
12	Torque	USP-S	Mil-Spec	2.02	1.66	1.19	1.41	2.05	5.92	4.4	3.22	3.67	4.27
13	Rocket Po	Galil AR	Mil-Spec	5.21	1.64	0.59	0.37	0.25	14.83	3.64	1.22	0.81	0.61
14	Elite Build	P90	Mil-Spec	1.46	0.33	0.18	0.15	0.15	7.96	1.57	0.62	0.43	0.37
15	Bunsen Bu	Glock-18	Mil-Spec	0.86	0.32	0.21	0.44	0.2	4.87	1.68	0.89	2.39	0.88
16	Ranger	Nova	Mil-Spec	0.57	0.28	0.2	0.21	0.18	1.72	0.65	0.41	0.35	0.31
17	Riot	UMP-45	Mil-Spec	0.45	0.3	0.27	0.29	0.25	1.36	0.7	0.39	0.49	0.32

3. ALGORITMUL

```python
# pentru fiecare calitate stim pretul
class Skin:
    def __init__(self):
        self.name = ""
        self.weapon = ""
        self.prices = []
        self.rarity = ""


# fiecare cutie cu skin-uri + link-ul cu date despre skin-urile din ea
class Case:
    def __init__(self):
        self.name = ""
        self.link = ""
        self.Skins = []
        self.byRarity = []
```

```python
def get_offer(prices_list, min_profit_procent, max_loss_procent):
    Cases = []
    db = xl.readxl(prices_list)
    sheet_names = db.ws_names

    for sheet in sheet_names[1:]:
        ObjCase = cls.Case()
        ObjCase.name = sheet
        read_xl(db, sheet, ObjCase)
        Cases.append(ObjCase)

    best_deals = generate_offer(Cases, min_profit_procent, max_loss_procent)

    # returnez o oferta random din cele ce respecta procentele
    random_trade = random.randint(0, len(best_deals)-1)

    return best_deals[random_trade]
```

```python
# va fi apelata pentru fiecare cutie
def read_xl(db, sheet_name, Cases):
    Covert = []
    Classified = []
    Restricted = []
    Mil = []
    [roww, coll] = db.ws(sheet_name).size
    data = db.ws(sheet_name).range('A2:N' + str(roww))
    for row in data:
        ObjSkin = cls.Skin()
        ObjSkin.name = row[0]
        ObjSkin.weapon = row[1]

        for i in range(0, 10):
            ObjSkin.prices.append(row[i + 3])

        if row[2] == 'Covert':
            Covert.append(ObjSkin)
        if row[2] == 'Classified':
            Classified.append(ObjSkin)
        if row[2] == 'Restricted':
            Restricted.append(ObjSkin)
        if row[2] == 'Mil-Spec':
            Mil.append(ObjSkin)

    Cases.byRarity.append(Covert)
    Cases.byRarity.append(Classified)
    Cases.byRarity.append(Restricted)
    Cases.byRarity.append(Mil)
```

3. ALGORITMUL

```
def generate_offer(Cases, procent_min_win, procent_max_loss):
    best_deals = []
    for case in Cases:

        # luam calitatile pe rand (FN, MW, FT, ...)
        concatenated = chain(range(0, 4), range(5, 9))
        for i in concatenated:

            # pentru fiecare raritate de skin
            # index_0 = indecele raritatii cu 1 mai mare
            for index_0, skin_list in enumerate(case.byRarity[1:]):

                cheap_skin = cheapest_skin(skin_list, i)
                if cheap_skin != -1:

                    # pretul skinului de cumparat * 10
                    trade_price = 10 * float(cheap_skin.prices[i])

                    # desi toate cele 10 skin-uri bagate in trade au a
                    # in functie de o valoarea numita float (unica per
                    # skin cu o calitate mai mica. Ca sa am acest floa
                    # fiecare site in parte si sa fac cate o extractie
                    # Pentru moment vom avea o sansa de 50/50 pentru c
                    # calitate, sau cu una mai proasta.
                    float_chance = random.randint(0, 1)
```

```
cheap_skin_2 = cheapest_skin(case.byRarity[index_0], i + float_chance)
expensive_skin_2 = expensive_skin(case.byRarity[index_0], i + float_chance)
if cheap_skin_2 != -1 and expensive_skin_2 != -1:
    # float(cheap_skin_2.prices[i + float_chance])
    # float(expensive_skin_2.prices[i + float_chance])
    loss = float(cheap_skin_2.prices[i + float_chance]) / trade_price
    win = float(expensive_skin_2.prices[i + float_chance]) / trade_price
    if loss >= (1 - procent_max_loss / 100) and win >= (1 + procent_min_win / 100):
        # case.name = numele cutiei din care sunt skin-urile
        # cheap_skin = skin-ul de cumparat (cel mai ieftin de CALITATEA i si RARITATEA index
        # i = indicele calitatii la care sa il cumparam
        # float_chance = indicele calitatii skin-ului/urilor care poate pica
        # case.byRarity[index_0] = lista cu skin-ul/urile care pot pica
        best_deals.append([case.name, cheap_skin, i, float_chance, case.byRarity[index_0],
                           cheap_skin_2, expensive_skin_2])
```

get_offer

```
best_deals = generate_offer(Cases, min_profit_procent, max_loss_procent)

# returnez o oferta random din cele ce respecta procentele
random_trade = random.randint(0, len(best_deals)-1)

return best_deals[random_trade]
```

INTERFATA


```python
def extract_skins(event):
    # dezactivez butonul
    btn_UpdateDB['state'] = 'disabled'
    btn_UpdateDB.unbind("<Button-1>")
    # curat fereastra de log
    displayVar.set('')
    window.update()

    # functia propriu-zisa
    Cases = cs_scrp.extract_from_cases('https://csgostash.com/containers/skin-cases', displayVar, window)
    save.to_xlsx(Cases, 'Files/Skin_names.xlsx', False)

    # reactivez butonul
    btn_UpdateDB['state'] = 'active'
    btn_UpdateDB.bind("<Button-1>", extract_skins)

    # activez si butonul pentru preturi in cazul in care era dezactivat
    btn_UpdatePrices['state'] = 'active'
    btn_UpdatePrices.bind("<Button-1>", extract_prices)
```

```python
def extract_prices(event):
    # dezactivez butonul
    btn_UpdatePrices['state'] = 'disabled'
    btn_UpdatePrices.unbind("<Button-1>")

    # curat fereastra de log
    displayVar.set('')
    window.update()

    #functia propriu-zisa
    db = readxl('Files/Skin_names.xlsx')
    sheet_names = db.ws_names

    for sheet in sheet_names[1:]:
        ObjCase = cls.Case()
        ObjCase.name = sheet
        alg.read_xl(db, sheet, ObjCase)
        Cases.append(ObjCase)

    pricey.extract_prices('Files/Skin_names.xlsx', 'Files/Skin_prices.xlsx', displayVar, window)

    # reactivez butonul
    btn_UpdatePrices['state'] = 'active'
    btn_UpdatePrices.bind("<Button-1>", extract_prices)

    # activez butoanele pt cautarea de oferta
    btn_risk1.config(background='gold')
    btn_risk1['state'] = 'active'
    btn_risk1.bind("<Button-1>", lambda event, risk=1: search_offer(event, risk))
```

INTERFATA


```
btn_risk1 = tk.Button(frame, text="1", image=pixel, relief='solid', bd=0, bg='gold', fg='black',
                      font=('Montserrat Black', 19), width=35, height=35, compound="c")
btn_risk1.place(x=680, y=310)
btn_risk1.bind("<Button-1>", lambda event, risk=1: search_offer(event, risk))

btn_risk2 = tk.Button(frame, text="2", image=pixel, relief='solid', bd=0, bg='gold', fg='black',
                      font=('Montserrat Black', 19), width=35, height=35, compound="c")
btn_risk2.place(x=740, y=310)
btn_risk2.bind("<Button-1>", lambda event, risk=2: search_offer(event, risk))

btn_risk3 = tk.Button(frame, text="3", image=pixel, relief='solid', bd=0, bg='gold', fg='black',
                      font=('Montserrat Black', 19), width=35, height=35, compound="c")
btn_risk3.place(x=800, y=310)
btn_risk3.bind("<Button-1>", lambda event, risk=3: search_offer(event, risk))

# daca fisierul cu preturile nu exista, programul nu poate cauta oferte
if not path.exists('Files/Skin_prices.xlsx'):
    #btn_risk1.config(background='gray63')
    btn_risk1['state'] = 'disabled'
    btn_risk1.unbind("<Button-1>")
```

```
def search_offer(event, risk):
    deal = []

    if risk == 1:
        # max_loss = 0 deci nu poate iesi in pierdere,
        # dar de regula si castigurile sunt mici
        deal = alg.get_offer('Files/Skin_prices.xlsx', 0, 0)
    if risk == 2:
        deal = alg.get_offer('Files/Skin_prices.xlsx', 100, 40)
    if risk == 3:
        # max_loss = 100 sunt dispus sa pierd toti banii intr-un trade (imposibil oricum)
        deal = alg.get_offer('Files/Skin_prices.xlsx', 200, 100)

    # afisez cele 10 skinuri
    global skin_in
    skin_in = get_n_resize(deal[1].weapon + ' ' + deal[1].name)
    print_trade_in(skin_in)

    # afisez date despre skinul de cumparat
    skin_in_q = alg.num_to_quality(deal[2])

    invalid_values = ['N/A', ' ', '']

    if deal[1].prices[deal[2]] not in invalid_values:
        price_0 = str(round(float(deal[1].prices[deal[2]]), 2))
    else:
        price_0 = '-'

    skin_in_data = tk.Label(frame, image=skin_in,
                            text=' ' + deal[1].name + '\n PRICE: ' + price_0 + '$\n Q: ' + skin_in_q,
                            font=('Montserrat Black', 16), justify='left', fg='Dark green', background='SpringGreen3', width=270,
                            height=108, compound='left')
    skin_in_data.place(x=15, y=620)
```

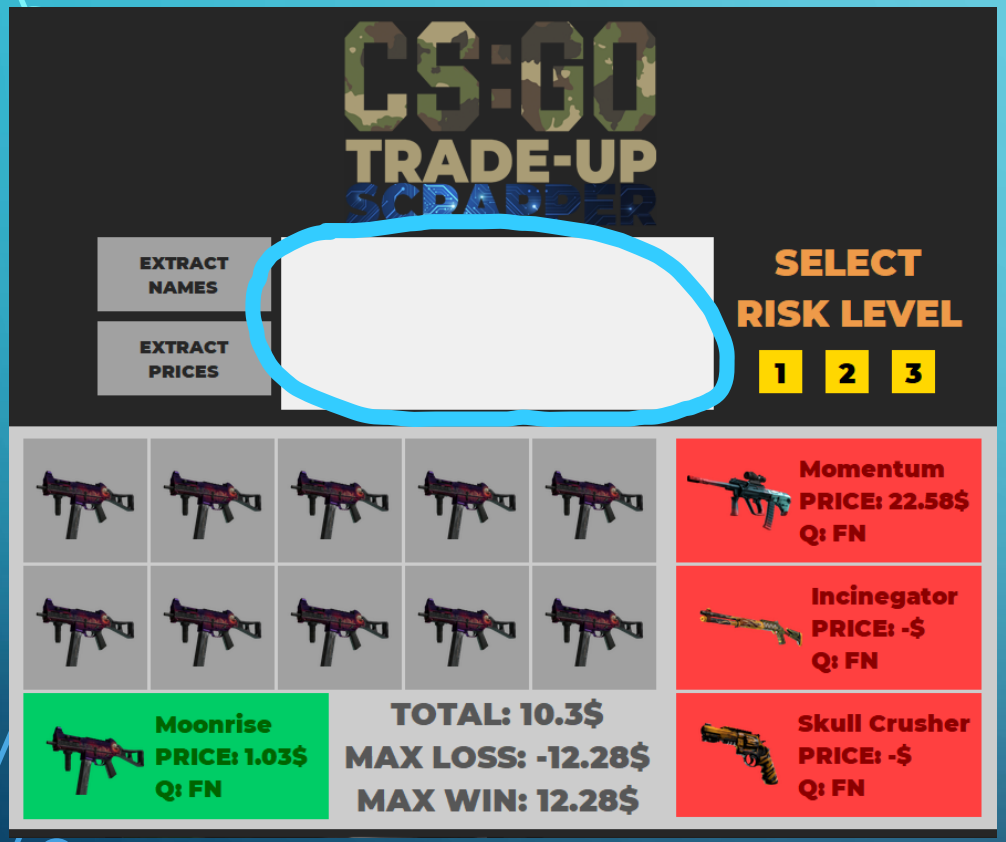
```
skin_out_q = alg.num_to_quality(deal[2] + deal[3])
if len(deal[4]) >= 1:

    global skin_out_1
    skin_out_1 = get_n_resize(deal[4][0].weapon + ' ' + deal[4][0].name)

    if deal[4][0].prices[deal[2] + deal[3]] not in invalid_values:
        price_1 = str(round(float(deal[4][0].prices[deal[2] + deal[3]]), 2))
    else:
        price_1 = '-'

    skin_out_data_1 = tk.Label(frame, image=skin_out_1,
                               text=' ' + deal[4][0].name + '\n PRICE: ' + price_1 + '$\n Q: ' + skin_out_q,
                               font=('Montserrat Black', 16), justify='left', fg='red4', background='brown1', width=270, height=106,
                               compound='left')
    skin_out_data_1.place(x=605, y=390)
```

INTERFATA


```
max_loss = alg.cheapest_skin(deal[4], deal[2] + deal[3])
max_win = alg.expensive_skin(deal[4], deal[2] + deal[3])
max_loss_f = float(max_loss.prices[deal[2] + deal[3]])
max_win_f = float(max_win.prices[deal[2] + deal[3]])
total = round(float(price_0) * 10, 3)

select_risk = tk.Text(frame, font=('Montserrat Black', 21), bd=0, fg='gray34', bg='gray80', width=15, height=3)
select_risk.tag_configure('center', justify='center')
select_risk.insert('1.0', 'TOTAL: ' + str(total) + '$\n'
                   + 'MAX LOSS: ' + str(round(total - max_loss_f, 2)) + '$\n'
                   + 'MAX WIN: ' + str(round(max_win_f - total, 2)) + '$')
select_risk.tag_add('center', '1.0', 'end')
select_risk.place(x=300, y=617)
# endregion
```

```
def print_trade_in(skin_in):
    x = 15
    y = 390

    for i in range(0, 10):
        skin_in_label = tk.Label(frame, image=skin_in, background='gray63', width=108, height=108)

        if i == 5:
            y = 505
            x = 15

        skin_in_label.place(x=x, y=y)
        x += 115
```

INTERFATA




```
def updateDisplay(myString):
    displayVar.set(displayVar.get() + '\n' + myString)
```

```
# region Log Viewer Window
displayVar = tk.StringVar()
# justify = left (aliniez textul la stanga)
# anchor = 'sw' (ultimul text introdus ramane in josul paginii si oricum redimensionez Label-ul
#               el ramane jos de tot, fara a da resize la fereastra)
displayLab = tk.Label(frame, textvariable=displayVar, height=10, width=55, justify='left', anchor='sw')
displayLab.place(x=248, y=208)
#endregion
```