**A2R Webshop**

Tartalomjegyzék

[A webshop célja feladata 4](#_Toc164541635)

[Alkalmazott technológiák és programok 4](#_Toc164541636)

[Webtechnológiák 4](#_Toc164541637)

[Python 4](#_Toc164541638)

[Django 5](#_Toc164541639)

[Selenium 5](#_Toc164541640)

[Visual Studio Code 5](#_Toc164541641)

[Internet böngészők 5](#_Toc164541642)

[Git 6](#_Toc164541643)

[GitHub 6](#_Toc164541644)

[Adatbáziskezelő rendszer 6](#_Toc164541645)

[XAMPP/LARAGON 6](#_Toc164541646)

[Az alkalmazás adatbázisának bemutatása 7](#_Toc164541647)

[Django által automatikusan létrehozott táblák 7](#_Toc164541648)

[auth\_group 7](#_Toc164541649)

[auth\_group\_permissions 7](#_Toc164541650)

[auth\_permission 7](#_Toc164541651)

[auth\_user 8](#_Toc164541652)

[auth\_user\_groups 8](#_Toc164541653)

[auth\_user\_user\_permissions 8](#_Toc164541654)

[django\_admin\_log 8](#_Toc164541655)

[django\_content\_type 8](#_Toc164541656)

[django\_migrations 8](#_Toc164541657)

[django\_session 8](#_Toc164541658)

[Django models.py által létrehozott táblák 8](#_Toc164541659)

[webshop\_cart 8](#_Toc164541660)

[webshop\_category 9](#_Toc164541661)

[webshop\_order 9](#_Toc164541662)

[webshop\_product 9](#_Toc164541663)

[webshop\_useraddress 9](#_Toc164541664)

[webshop\_warehouse 10](#_Toc164541665)

[Backend megoldások 10](#_Toc164541666)

[Webshop app 10](#_Toc164541667)

[view.py 10](#_Toc164541668)

[Felhasznált könyvtárak: 10](#_Toc164541669)

[decorators: 11](#_Toc164541670)

[def getProductsByCategory(request, category): 11](#_Toc164541671)

[def getCategories(request): 11](#_Toc164541672)

[def reigster(request): 11](#_Toc164541673)

[def index\_view(request): 12](#_Toc164541674)

[def get\_csrf\_token(request): 12](#_Toc164541675)

[def authenticated(request): 12](#_Toc164541676)

[def logout\_req(request): 12](#_Toc164541677)

[def login\_req(request): 12](#_Toc164541678)

[def add\_cart(request, item): 13](#_Toc164541679)

[def get\_cart(request): 13](#_Toc164541680)

[def remove\_cart(request, item): 13](#_Toc164541681)

[models.py 13](#_Toc164541682)

[Felhasznált könyvtárak: 14](#_Toc164541683)

[class Category(models.Model): 14](#_Toc164541684)

[class Warehouse(models.Model): 14](#_Toc164541685)

[class Product(models.Model): 14](#_Toc164541686)

[class Cart(models.Model): 15](#_Toc164541687)

[class UserAddress(models.Model): 15](#_Toc164541688)

[class Order(models.Model): 15](#_Toc164541689)

[serializers.py 16](#_Toc164541690)

[class ProductSerializer(ModelSerializer): 16](#_Toc164541691)

[class CategorySerializer(ModelSerializer): 16](#_Toc164541692)

[class CartSerializer(ModelSerializer): 16](#_Toc164541693)

[admins.py 17](#_Toc164541694)

[Felhasznált könyvtárak: 17](#_Toc164541695)

[Példa: 17](#_Toc164541696)

[apps.py 17](#_Toc164541697)

[Felhasznált könyvtárak: 17](#_Toc164541698)

[Regisztálás: 17](#_Toc164541699)

[a2r Szerver beállítások 17](#_Toc164541700)

[settings.py 17](#_Toc164541701)

[urls.py 17](#_Toc164541702)

[A Selenium csomag alkalmazása 18](#_Toc164541703)

[chromedriver.exe 18](#_Toc164541704)

[tu.py/ta.py 18](#_Toc164541705)

[Frontend megoldások 20](#_Toc164541706)

[Javascript fájlok 20](#_Toc164541707)

[script.js 20](#_Toc164541708)

[window.onload = function() {...} 20](#_Toc164541709)

[function webshopClicked() {...} 20](#_Toc164541710)

[function categoryClicked(category) {...} 21](#_Toc164541711)

[async function registerButtonClicked() {...} 21](#_Toc164541712)

[function registerClicked() {...} 21](#_Toc164541713)

[document.getElementById(’login-button’).onclick = function() {...} 21](#_Toc164541714)

[function logoutClicked() {...} 21](#_Toc164541715)

[function productCardAddFunction() {...} 21](#_Toc164541716)

[function getCartCount() {...} 22](#_Toc164541717)

[document.getElementById(’cart’).onclick = function() {...} 22](#_Toc164541718)

[function cartItemsAddFunction() {...} 22](#_Toc164541719)

[function messageBox(title, message, button) {...} 22](#_Toc164541720)

[aboutButton.onclick = function() {...} 23](#_Toc164541721)

[contractButton.onclick = function() {...} 23](#_Toc164541722)

[afsz.js 23](#_Toc164541723)

[about.js 23](#_Toc164541724)

[Stylesheet fájlok 23](#_Toc164541725)

[style.css 23](#_Toc164541726)

[order.css 23](#_Toc164541727)

[HTML fájlok 24](#_Toc164541728)

[index.html 24](#_Toc164541729)

[orders\_generic.html 24](#_Toc164541730)

[active\_orders.html 24](#_Toc164541731)

[order\_history.html 24](#_Toc164541732)

[Kiegészítő adatok 24](#_Toc164541733)

[Az adatbázishoz tartozó dump file: 24](#_Toc164541734)

[A projekt és a dokumentáció készítői 25](#_Toc164541735)

[Szabó Ádám fullstack fejlesztő 25](#_Toc164541736)

[Mezővári Roland fullstack fejlesztő 25](#_Toc164541737)

[Borsy Ádám fullstack fejlesztő 25](#_Toc164541738)

# A webshop célja feladata

A „webshop”-unk célja az, hogy megpróbáljunk minél több tudást összegyűjteni a megírása közben. Próbáltuk minél letisztultabbra és minél egyszerűbbre csinálni, hogy ne okozzon gondot a használata egy korosztálynak se.

A végproduktum megoldását úgy próbáltuk publikálni, hogy a weboldalnak ne okozzon gondot se a telefonon való megjelenítése se a számítógépen (minimum 4:3-as kijelző skála) való megjelenítése se.

# Alkalmazott technológiák és programok

Az A2R Webshop weboldalának és fejlesztőikörnyezetének kialakításához több különböző technológiát és eszközt használtunk.

## Webtechnológiák

HTML (HyperText Markup Language), CSS (Cascading Style Sheets), JavaScript. Ezekkel a nyelvekkel oldottuk meg a weboldal teljesen dinamikus működését.

## Python

A Python kiváló nyelv alkalmazásokfejlesztésére, könnyen tanulható é solvasható, valamint kiterjedt támogatással és közösséggel rendelkezik.

## Django

A Django egy Python nyelven írt webes alkalmazás keretrendszer, amelyet gyors és hatékony fejlesztésre terveztek. A Django-t azért választottuk, mert nagyteljesítményű, jól dokumentált és könnyen bővíthető. A Django gyors fejlesztés és DRY (Don’t Repeat Yourself) elvek szerint működik, ami segít a projekt komplexitásának hatékony kezelésében.

## Selenium

A projekt tesztelésére a Django programozók közösségében népszerű Selenium nevezetű modult használjuk, amely települ az alkalmazáshoz és webböngészőn keresztül felhasználói interakciókat szimulálhatunk.

## Visual Studio Code

A VS Code (Visual Studio Code) kiváló támogatást nyúlt a Django keretrendszerhez, ezért erre a kód szerkeztőre esett a választásunk. Emellett egy Open Source (nyílt forráskódú) program ezért hatalmas a választék a kiterjesztések között, azaz van Django és Python-hoz tartozó kiterjesztése. Ezek az okok mellett a kitűnő választás volt a HTML, CSS és Javascript programozáshoz.

## Internet böngészők

A legnépszerűbb böngészők (pl.: Google Chrome, Firefox, Microsoft Edge, Safari) közül választottunk néhányat, hogy biztosítsuk a weboldal kompatibilitását és megfelelő működését a különböző platformokon és eszközökön.

## Git

Azért választottuk a Git-et a projektverziókezelésére, mert jól integrálható a Visual Studio Code-dal ,és könnyen kezelhető.

## GitHub

A GitHub egy ingyenes használható felhőszolgáltatás programozóknak, és a VS Code is automatikusan azt ajánlja, ezért kézenfogható volt, hogy a Git felhőszolgáltatásaként a GitHub-ot fogjuk használni.

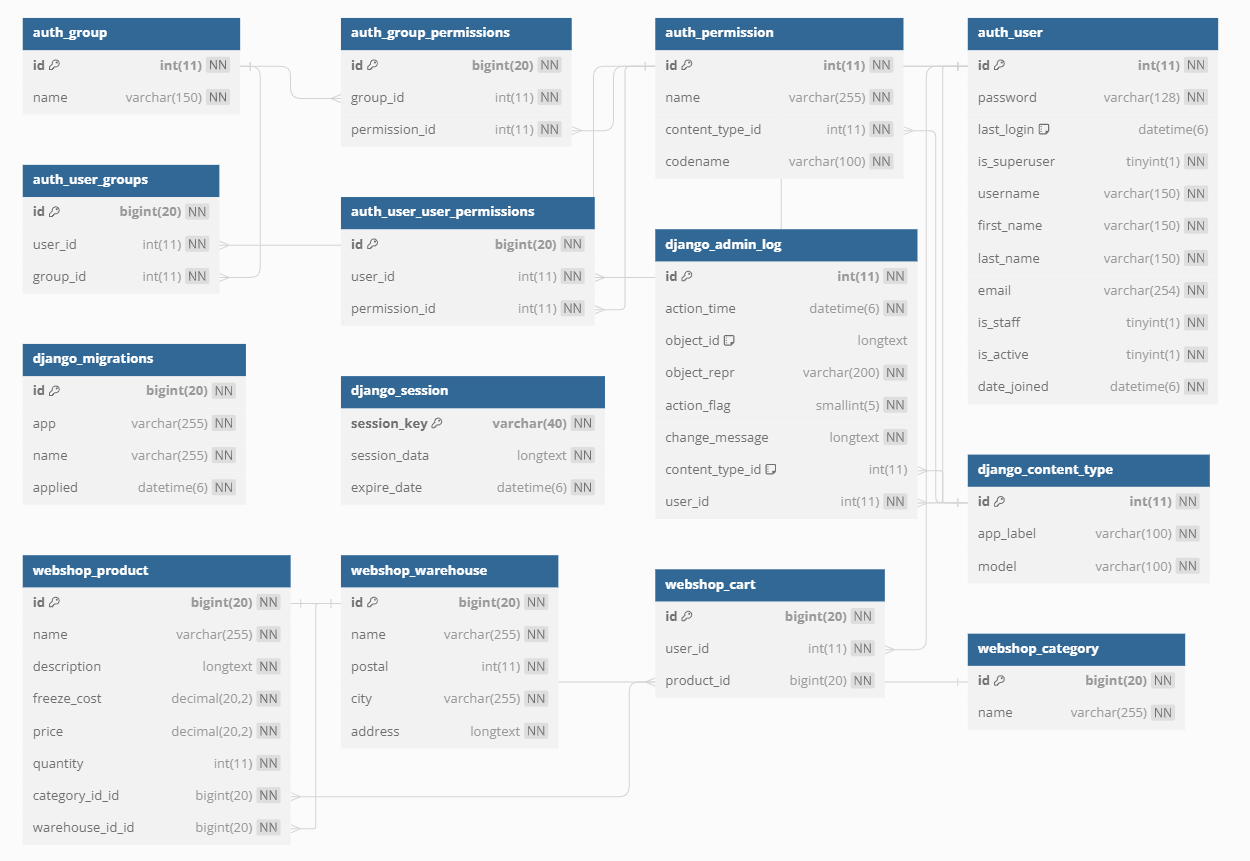
## Adatbáziskezelő rendszer

Az adatbázis kezelő rendszerben a MariaDB-t választottuk. Azért választottuk ezt a MySQL szervert, mert egyszerű a kezelése, és viszonylag stabil megbízható rendszer.

## XAMPP/LARAGON

Virtuális szerver kiszolgáló. A csapat egyik része a XAMPP-ot választotta a másik Laragon-t választott, azért, hogy több adatbázis rendszert is kipróbáljunk nyílván ugyan azon a motoron.

# Az alkalmazás adatbázisának bemutatása



A programhoz tartozó dump file-t [ide](#_Az_adatbázishoz_tartozó) kattintva található meg.

## Django által automatikusan létrehozott táblák

### auth\_group

* Ez a tábla tartalmazza a felhasználok csoportjait

### auth\_group\_permissions

* Ez a tábla a csoportokhoz rendelt jogosultságokat tartalmazza

### auth\_permission

* Ez a tábla a rendszer összes jogosultságát tartalmazza

### auth\_user

* Ez a tábla tárolja a felhasználót: a felhasználóra vonatkozó alapvető információk

### auth\_user\_groups

* Ez a tábla a felhasználók és a hozzájuk rendelt csoportok közötti kapcsolatot tartalmazza

### auth\_user\_user\_permissions

* Ez a tábla a felhasználókhoz rendelt egyedi jogosultságokat tartalmazza; ugyanúgy működik, mint az auth\_group\_permissions tábla, de itt a jogosultságok közvetlenül a felhasználóhoz vannak rendelve

### django\_admin\_log

* Ez a tábla a Django adminisztrációs felületén végzett műveletek naplóját tárolja

### django\_content\_type

* Ez a tábla a Django modelltípusokat tárolja

### django\_migrations

* Ez a tábla a Django adatbázis-migrációkat rögzíti

### django\_session

* Ez a tábla az aktív felhasználói munkamenetek adatait tárolja

## Django models.py által létrehozott táblák

A Django automatikus id hozzárendelése miatt a táblastruktúra nem tartalmazza a record egyedi azonosítóit (id az első sorban)!

### webshop\_cart

* Ez a tábla a felhasználó bevásárlókosarában lévő tételeket tárolja.
* Tábla struktúra:
  + user\_id: INT; FOREIGN\_KEY(a felhasználó azonosítój)
  + product\_id: BIGINT ;FOREIGN\_KEY(a termék azonosítója)

### webshop\_category

* Ez a tábla a webshop kategóriáit tárolja
* Tábla struktúra:
  + name: VARCHAR (a kategória neve)

### webshop\_order

* Ez a tábla a webshopban történt rendeléseket tárolja
* Tábla struktúra:
  + products: LONGTEXT (a megrendelt termékek nevét tartalmazza)
  + active: TINYINT; DEFAULT 1 (tárolja, hogy a megrendelés továbbra is aktív, a default érték django-n belűl van definiálva)
  + user\_id: INT; FOREIGN KEY (a felhasználó azonosítója)
  + address\_id: INT; FOREIGN KEY (a felhasználóhoz rendelt kiegészító adatok azonosítója)

### webshop\_product

* ez a tábla a webshopban elérhető termékeket tárolja
* Tábla struktúra:
  + name: BIGINT (a termék neve)
  + description: LONGTEXT (a termék leírása)
  + freeze\_cost: DECIMAL (a termék beszerzési ára)
  + price: DECIMAL (a termék eladási ára)
  + quantity: INT (a termék leltári számát tárolja)
  + category\_id\_id: BIGINT; FOREIGN\_KEY (a termék kategóriájának a kapcsolata)
  + warehouse\_id\_id: BIGINT; FOREIGN\_KEY (a termék leltározási helyének a kapcsolata)

### webshop\_useraddress

* Ez a tábla a felhasználóhoz tartozó kiegészítő adatokat tartalmazza
* Tábla struktúra:
  + postal: INT (a felhasználó irányítószámát tartalmazza)
  + city: VARCHAR (a felhasználó lakhelyének a város eleme)
  + address: LONGTEXT (a felhasználó lakhelyének a pontos címe elem)

### webshop\_warehouse

* Ez a tábla azokat a raktárakat tárolja, ahol a termékek elérhetők
* Tábla struktúra:
  + name: VARCHAR (a raktár megnevezése)
  + postal: INT (a raktár irányítószáma)
  + city: VARCHAR (a raktár elhelyezkedése)
  + address: LONGTEXT (a raktár pontos címe)

# Backend megoldások

## Webshop app

### view.py

Ebben a fájlban definiáltuk a Django nézeteket, amik feldolgozzák és végrehajtják a különböző HTTP kérelmeket. A következő függvényekben a request-et figyelmen kívűl fogom hagyni azokban amelyekben nincs felhasználva a belőlük kapott információ és csak a response miatt kellenek.

#### Felhasznált könyvtárak:

django.shortcuts

django.contrib.auth

django.contrib.auth.decorators

django.views.decorators.csrf

django.contrib.auth.models

rest\_framework.decorators

rest\_framework.response

django.contrib

django.http

django.middleware.csrf

django.contrib.auth

base64

#### decorators:

@api\_view(['GET'])

Ez a decorator megadja a függvénynek, hogy csak akkor fusson le, hogyha a HTTP request az egy GET metódus.

@api\_view(['POST'])

Ez a decorator megadja a függvénynek, hogy csak akkor fusson le, hogyha a HTTP request az egy POST metódus.

@requires\_csrf\_token

Ez a decorator megadja a függvénynek, hogy csak akkor fusson le, hogyha a HTTP request tartalmaz CSRF Tokent (X-CSRF-Token)

@csrf\_exempt

Ez a decorator megadja a függvénynek, hogy csak akkor fusson le, hogyha a HTTP request tartalmaz CSRF Tokent (X-CSRF-Token)

#### def getProductsByCategory(request, category):

@api\_view(['GET'])

def getProductsByCategory*(request, category)*:

    ...

Ez a függvény csak akkor működik, ha GET request-et kap. Attribútumként bekér egy kategória nevet.

#### def getCategories(request):

@api\_view(['GET'])

def getCategories*(request)*:

    ...

Ez a függvény visszaadja REST API-val a kategóriák szerializált változatát

#### def reigster(request):

@api\_view(['POST'])

@csrf\_exempt

def register*(request)*:

...

Ez a függvény attribútumként a request-et kéri és mivel ez a függvény csak POST request-et fogad el ezért a request-ből könnyen kivehető az összes kapott adat. A POST alapján létrehoz egy új felhasználót minden kiegészítő adatával, majd egy egyszerű szöveges JSON-t ad vissza ami a weboldalon megjelenik mint üzenet.

#### def index\_view(request):

def index\_view*(request)*:

...

Ez a függvény adja vissza az index.html-t, ellenőrzi, hogy a kliens felhasználója aktív-e vagy nincs letiltva az oldalról. visszatérő értékképpen render-ként az index.html-t adja a kerő felé amit a request-ből kap.

#### def get\_csrf\_token(request):

def get\_csrf\_token*(request)*:

...

Ez a függvény a X-CSRF-Token-t küldi el a kliens felé.

#### def authenticated(request):

def authenticated*(request)*:

...

Ez a függvény visszaküldi a kliens-nek, hogy jelenleg be van-e jelentkezve.

#### def logout\_req(request):

def logout\_req*(request)*:

...

Ez a függvény a kliens-t jelentkezteti ki. Visszatérési értéke a sikeresség (success) és egy üzenet lesz a kliens felé (message)

#### def login\_req(request):

@api\_view(['POST'])

@requires\_csrf\_token

login\_req*(request)*:

...

Ez a függvény feldolgozza a bejelentkezési kérelmet, és a request-ből kapott adatok alapjá bejelentkezteti a klienst. A sikerességet a Javascript ellenőrzi, visszatérési értéke mindenféleképpen egy render az index-hez.

#### def add\_cart(request, item):

def add\_cart*(request, item)*:

...

Ez a függvény hozzáad a request-ben meghatározott user-nek a kosarához egy terméket. Csak akkor adja, hozzá a terméket, hogyha be van jelentkezve. Visszatérési értékképpen JSON-ben ad vissza egy „succes” kulcsszóval egy boolean adatot.

#### def get\_cart(request):

@api\_view(['GET'])

def get\_cart*(request)*:

...

Ez a függvény a request-ben meghatározott user kosarát adja vissza. Visszatérési értékképpen a kosarának a tartalmát adja vissza.

#### def remove\_cart(request, item):

def remove\_cart*(request, item)*:

...

Ez a függvény a request-ben meghatározott user-nek a kosarából törli a megkapott „item”-ben meghatározott kosár azonosítószámát. Visszatérési értéke egy „success” amiben meg van határozva az, hogy sikeres volt-e a művelet.

### models.py

Ebben a fájlban hozzuk létre az adatbázis modelljeit.

#### Felhasznált könyvtárak:

django.db

django.contrib.auth.models

#### class Category(models.Model):

class Category(models.Model):

Ebben a model-ben a kategóriák vannak definiálva.

Referencia [ide](#_webshop_category) kattintva érhető el.

Az objektum string típússá változtatása str(obj):  
    def \_\_str\_\_*(self)*:

        return self.name

#### class Warehouse(models.Model):

class Warehouse(models.Model):

Ebben a model-ben a raktárak vannak definiálva.

Referencia [ide](#_webshop_warehouse) kattintva érhető el.

Az objektum string típússá változtatása str(obj):  
    def \_\_str\_\_*(self)*:

        return f"{self.postal} {self.address} | {self.name}"

#### class Product(models.Model):

class Product(models.Model):

Ebben a model-ben a termékek vannak definiálva.

Referencia [ide](#_webshop_product) kattintva érhető el.

Az objektum string típússá változtatása str(obj):  
    def \_\_str\_\_*(self)*:

        return f"{self.name}"

#### class Cart(models.Model):

class Cart(models.Model):

Ebben a model-ben a kosárban lévő termékek vannak definiálva.

Referencia [ide](#_webshop_cart) kattintva érhető el.

Az objektum string típússá változtatása str(obj):  
    def \_\_str\_\_*(self)*:

        return f"{self.user.username} | {self.product.name}"

#### class UserAddress(models.Model):

class UserAddress(models.Model):

Ebben a model-ben a felhasználó kiegészítő adatai vannak definiálva.

Referencia [ide](#_webshop_useraddress) kattintva érhető el.

Az objektum string típússá változtatása str(obj):  
    def \_\_str\_\_*(self)*:

        return f"{self.user.username} | {self.postal} {self.city} {self.address}"

#### class Order(models.Model):

class Order(models.Model):

Ebben a model-ben a megrendelések vannak definiálva.

Referencia [ide](#_webshop_order) kattintva érhető el.

Az objektum string típússá változtatása str(obj):  
    def \_\_str\_\_*(self)*:

        return f"{self.user.email} | {self.products}"

### serializers.py

Ebben a fájlban készítjük el az objektumok szerializálását (felbontását) a ModelSerializer segítségével:

from rest\_framework.serializers import ModelSerializer

Bekérjük azokat a model-eket amiknél szükségünk lesz a szerializációra és azokkal fogunk dolgozni.

from .models import Product, Category, Cart

#### class ProductSerializer(ModelSerializer):

class ProductSerializer(ModelSerializer):

    class Meta:

        fields = '\_\_all\_\_'

        model = Product

        depth = 1

Megadjuk a szerializáló meta-jának, hogy melyik „táblákat” vegye figyelembe és megadjuk neki a modelt. A depht beállítás azt jelenti, hogy milyen „távolságokra” menjen el, azaz, ha a model-nek van Foreign key-je akkor azokat is figyelembe veszi.

#### class CategorySerializer(ModelSerializer):

class CategorySerializer(ModelSerializer):

    class Meta:

        fields = '\_\_all\_\_'

        model = Category

        depth = 0

A leírását lásd [feljebb](#_class_ProductSerializer(ModelSerial).

#### class CartSerializer(ModelSerializer):

class CartSerializer(ModelSerializer):

    class Meta:

        fields = '\_\_all\_\_'

        model = Cart

        depth = 2

A leírását lásd [feljebb](#_class_ProductSerializer(ModelSerial).

### admins.py

Ebben a fájlban regisztráljuk a Django admin oldalhoz a model-eket.

#### Felhasznált könyvtárak:

django.contrib

#### Példa:

admin.site.register(Category)

### apps.py

Ebben a fájlban definiáljuk az app-unkat

#### Felhasznált könyvtárak:

django.apps

#### Regisztálás:

class WebshopConfig(AppConfig):

    default\_auto\_field = 'django.db.models.BigAutoField'

    name = 'webshop'

## a2r Szerver beállítások

### settings.py

Ebben a fájlban vannak meghatározva a szerver indítási paraméterei

### urls.py

Ebben a fájlban az elérési útvonalak és a hozzá tartozo view-ok vannak meghatározva. Azaz, hogy egy útvonal milyen response-zal legyen a kliens request-je felé.

## A Selenium csomag alkalmazása

### chromedriver.exe

A requirements.txt-vel indítjuk el a telepítést, mint egyéb csomagok telepítése esetén, kivéve az exe fájlt, mert azt a böngésző fejlesztői teszik elérhetővé, mint nyílt forráskódű szoftvert. Az a2r könyvtáron belül a testrobot könyvtárban található állományok között a chromedriver.exe, ami betölti azt a szerepet, hogy a VS Code kommunikáljon a böngészővel, és automatizálva kerüljön használatra, esetünkben a Google Chrome. Különféle webböngésző alkalmazható erre a célra, viszont a driverek böngésző specifikusak. Emellett elengedhetetlen, hogy a webdriver a feltelepített böngésző verziószámával megegyezzen a megfelelő működés érdekében.

### tu.py/ta.py

A tu(test user) és ta(test admin) fájlokban íródtak meg főként XML Path-ok útján a weboldalon történő navigálások, melyek segítenek kiválasztani a megfelelő elemeket a teszteléshez, így szimulálva felhasználói interakciókat.

A program kódjában a magyarázatot követően importortálások történnek, amik a következőek:

* webdriver: kapcsolat kialakítása
* Keys: billentyűzetről bevitt műveletek
* By: keresési feltételek biztosít a webes elemek azonosítására
* datetime: dátum objektumok létrehozására
* time: időzítés, interakciók késleltetése

from selenium import webdriver

from selenium.webdriver.common.keys import Keys

from selenium.webdriver.common.by import By

from datetime import datetime

import time

Böngésző megnyitása és webcím kijelölése

driver = webdriver.Chrome()

driver.get("http://127.0.0.1:8000/")

A következő sorokban letároljuk az aktuális dátumot, majd megformázzuk.

current\_date = datetime.now()

formatted\_date = current\_date.strftime("%Y\t%m%d")

Ezt követően elkezdődik az automatizált regisztráció. A Selenium lineárisan halad a kódban és a változókban megadott XPath-on keresztül rákattint a regisztáció gombra.

A send\_keys művelet beviszi a szöveget a felhasználó mezőbe.

reg\_button = driver.find\_element(By.XPATH, '//\*[@id="register"]')

reg\_button.click()

time.sleep(2)

username\_fill = driver.find\_element(By.XPATH, '//\*[@id="username"]')

username\_fill.send\_keys("roborob")

time.sleep(1)

A szükséges input mezők helyes kitöltése után a korábban megformázott aktuális dátummal is ellátja a regisztrációs oldalt.

bdate\_field = driver.find\_element(By.ID,'birthdate')

bdate\_field.send\_keys(formatted\_date)

A teszteket a testrobot mappából a tu/ta.py fájl indításával történnek és szemléltetik a weboldal működését, az elemek bejárásával.

# Frontend megoldások

## Javascript fájlok

### script.js

Ebben definiáljuk a weboldal teljes háttér működését. Ebben vannak az API lekérések és a kényelmi funkciók.

#### window.onload = function() {...}

Ebben a függvényben a weboldal betöltését kezeljük le. Megnézzük, hogy a felhasználó be van-e jelentkezve és az alapján jelenítjük meg a gombokat.

#### function webshopClicked() {...}

Ebben a függvényben kérjük le a webshop tartalmát API-val. Lekérjük benne az alapértelmezett (1. iterációját a kategóriáknak) kategóriát és megjelenítjük a benne lévő adatokat. A függvény definiálása előtt adjuk hozzá a webshop gomb event-jéhet a függvényt.

#### function categoryClicked(category) {...}

Ebben a függvényben kérjük le egy adott kategóriához tartozó termékeket. A termékekhez a [productCardsAddFunction()](#_function_productCardAddFunction()_{) függvénnyel adjuk hozzá a funkciókat.

#### async function registerButtonClicked() {...}

Ez az asszinkron függvényben van definiálva a regisztrációhoz tartozó HTTP POST request. Ebben a függvényben teljesül a regisztáció után lévő fetch. A fetch után visszatér egy üzenet amit a [messageBox(title, message, button)](#_function_messageBox(title,_message,) továbbítva megjelenik egy popup. Ebben a függvényben teljesűl a regisztráció.

#### function registerClicked() {...}

Ebben a függvényben hozza létre majd adja hozzá a .container class-hoz a regisztrációs „form”-ot ami div elemmel van megoldva a fetch pedig a [async function registerButtonClicked()](#_async_function_registerButtonClicke)-ben teljesül.

#### document.getElementById(’login-button’).onclick = function() {...}

Ebben a függvényben létrehozzuk a belépési form-ot. Ezután a form action-je miatt teljesül a fetch a form-on belül és belépteti a klienst.

#### function logoutClicked() {...}

Ebben a függvényben van meghatározva a kijelentkezés egy fetch-en keresztül, ha a fetch sikeres volt akkor a kijelentkezés végbe ment.

#### function productCardAddFunction() {...}

Ebben a függvényben vannak megoldva, ha van a „document”-en azaz a body-n belül .product-card class, akkor mindegyiken belül megkeresi a .product-button-t ezután hozzá adja a fetch-et ami hozzá adja a kliens user-ének a kosarához a terméket. Amennyiben nincs bejelentkezve a kliens akkor nem adhatja hozzá a terméket a kosárhoz.

#### function getCartCount() {...}

Lekéri a kliens kosarának a tartalmának darabját, azaz mennyi termék van a kosarában.

#### document.getElementById(’cart’).onclick = function() {...}

Amikor a kliens rányom a kosár ikonra akkor fetch-el lekéri a termékeket amik a kliens kosarában vannak az API alapján. Létrehoz egy kosár oldalt ahová betölti a visszakapot termékeket és összegzi a termék árait. Majd lefuttatja a [cartItemsAddFunction()](#_function_cartItemsAddFunction()_{..) függvényt. Amennyiben nincs bejelentkezve a kliens, ez a gomb nem működik.

#### function cartItemsAddFunction() {...}

Ez a függvény akkor tud lefutni amikor a kosár elemi léteznek, ha nincsenek nem fut le. Amennyiben vannak a kosárban lévő kártyák gombjához ad funkciót, hogy el lehessen minden terméket egyesével távolítani, ahogy a terméket eltávolítjuk ezt azonnal küldi a szervernek API-n keresztül és a szerver is eltávolítja az adatbázisból. Emellet a függvényben van megírva a „megrendelés” gomb működése.

#### function messageBox(title, message, button) {...}

Ez a függvény létrehoz egy popup üzenetdobozt. Bemenetként kéri az üzenetdoboz címét, az üzenetet, és a gomb kiírását pl.: (’OK’). Hozzáadja az gombhoz, hogy távolítsa is el.

#### aboutButton.onclick = function() {...}

Ebben a függvényben van megcsinálva az amikor rányomunk a „Rólunk” gombra, betöltse a tartalmát. Ebben a függvényben van megoldva a szavak leesési animációja.

#### contractButton.onclick = function() {...}

Ebben a függvényben van megcsinálva az amikor rányomunk a „ÁFSZ” gombra, betöltse a tartalmát.

### afsz.js

Ebben a fájlban csak az „ÁFSZ” gomb tartalma van inicializálva, azért, hogy ne legyen junk a script.js-ben.

### about.js

Ebben a fájlban a „Rólunk” gomb tartalma van inicializálva. Minden szóhoz hozzá van rendelve egy div ami „about-word” class-al van ellátva, így lett megoldva az animáció amivel a szavak leesnek.

## Stylesheet fájlok

### style.css

Ebben a fájlban van megoldva az index-en belűli elemek kinézete. Minden gomb minden kártya kinézete ebben van megoldva kivéve a „Megrendelések” oldal.

### order.css

Ebben a fájlban van megírva a „Megrendelések” oldal kinézetének a kódja.

## HTML fájlok

### index.html

Ebben van a főoldal struktúrája megírva, és a script.js ebbe helyezi bele a további tartalmakat.

### orders\_generic.html

Ez a „Megrendelések” oldal alapja, ebbe render-eli bele a Django a további tartalmakat a „content” block-hoz. A további oldal eléréseket a Django helyettesíti be.

Ide csak staff jogosúltsággal lehet belépni.

### active\_orders.html

Ezt az oldalt az [orders\_generic.html](#_orders_generic.html)-be helyettesíti be, a megkapot adatok alapján. Ide rakja a teljesítésre váró megrendeléseket.

### order\_history.html

Ezt az oldalt az [orders\_generic.html](#_orders_generic.html)-be helyettesíti be, a megkapot adatok alapján. Ide rakja be a már teljesített rendeléseket, vissza lehet nézni a rendelési előzményeket.

# Kiegészítő adatok

## Az adatbázishoz tartozó dump file:



## A projekt és a dokumentáció készítői

### Szabó Ádám fullstack fejlesztő

### Mezővári Roland fullstack fejlesztő

### Borsy Ádám fullstack fejlesztő