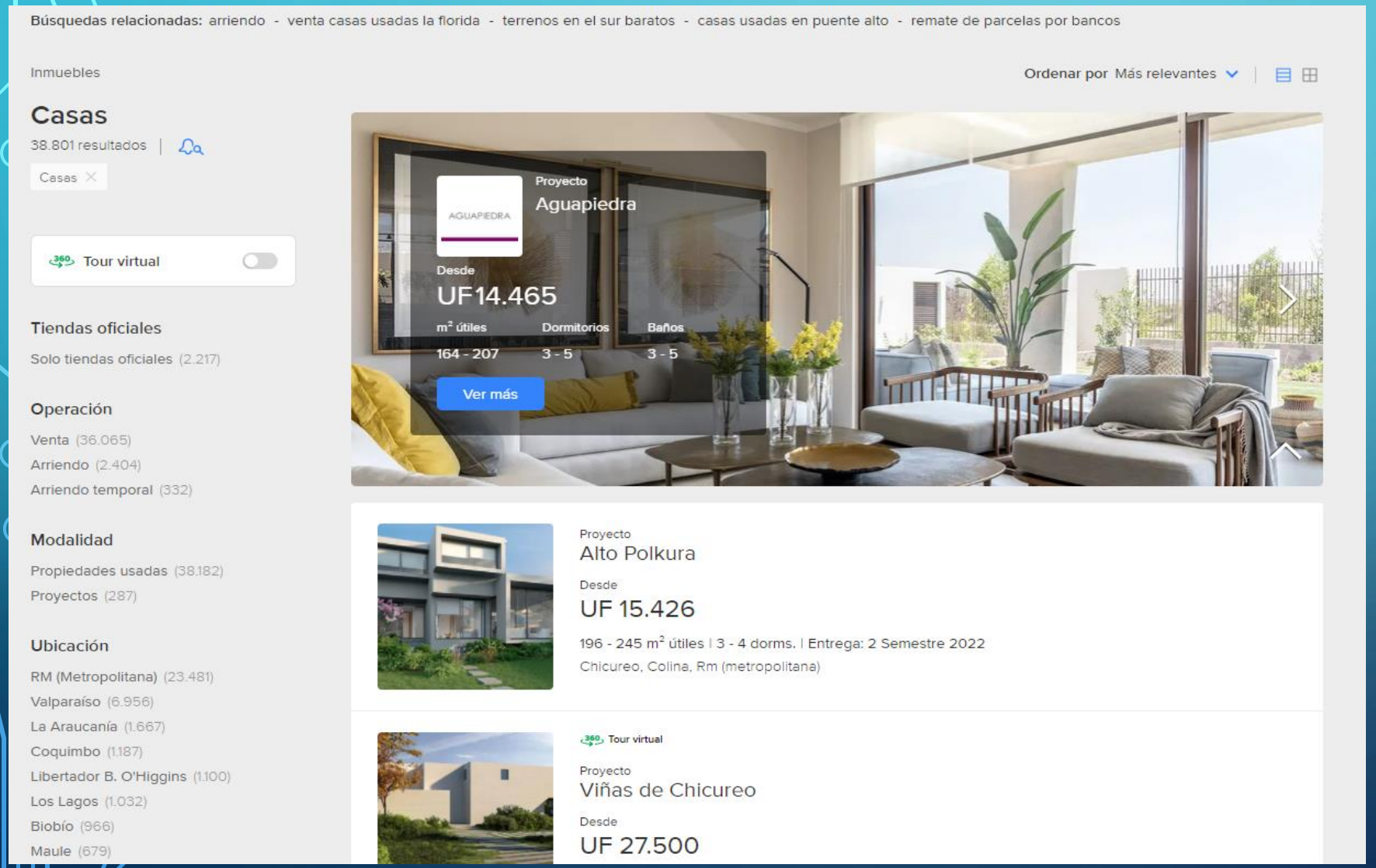
SCARAPPER MERCADO LIBRE INMUEBLES

NOMBRE: ANDRES SHRANKA

PROFESOR: MIGUEL GUEVARA

ASIGNATURA: TALLER INTEGRADO

URL SEMILLA: HTTPS://LISTADO.MERCADOLIBRE.CL/INMUEBLES/CASAS/#DEAL_PRINT_ID=A3B5C420-DFC2-11EB-8057-19E52126F5FB&C_ID=CAROUSEL&C_ELEMENT_ORDER=2&C_CAMPAIGN=CASAS&C_UID=A3B5C420-DFC2-11EB-8057-19E52126F5FB



Mercado libre tiene un gran numero de artículos publicados en su pagina en este caso yo usare la sección de inmuebles, en este caso serian las casa.

DATOS IMPORTANTES DEL SITIO

Ingrese a el primer anuncio de una propiedad para mostrar los datos de interés
Los cuales tienen flechas sobre ellos

- Imagen url
- Precio en UF
- Metros útiles
- Habitaciones
- Región
- Ciudad
- Dirección
- Titulo
- url de la publicación

TECNOLOGÍA UTILIZADA: SELENIUM,BEAUTIFOULSOUP, SQLITE3

```python
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
import os
from bs4 import BeautifulSoup
import requests
from datetime import datetime
import sqlite3
import andresBd
import math

log_file = "mercado.log"
if not os.path.isfile(log_file):
    open(log_file, 'w+')

def get_page_url(pageNumber):···

def get_address(data):···

def li_html_to_obj(house_li_html): #obtenemos los objetos python···

def parser_and_save_items(pageUrl):···

URL_use = "https://listado.mercadolibre.cl/inmuebles/casas/#deal_print_id=a3b5c420-dfc2-11eb-8057-19e52126f5fb&c_id=carousel&c_element_order=2&c_campaign=CASAS&c_uid=a3b5c420-dfc2-11eb-8057-19e52126f5fb"
driver = webdriver.Firefox(executable_path= 'geckodriver') #maneja el navegador en este caso firefox
driver.get(URL_use)
html_code = driver.page_source # tomo el html de la pagina
soup = BeautifulSoup(html_code, 'lxml')#objetohtml y el parse
articles_mount = soup.find(class_="ui-search-search-result__quantity-results")
articles_mount = float(articles_mount.text.split(" ")[0].replace(".", ""))
#print(articles_mount)
#divido de 48 ya que siempre hay 48 resultados en la pagina
page_amount = math.ceil(articles_mount/48)
#print(page_amount)

#aqui debuese recorrer las paginas y guardar en mi base de datos
for actual_number_page in range(1, page_amount+1):
   actul_url = get_page_url(actual_number_page)
   parser_and_save_items(actul_url) #no me inserta en la base
   #al ejercutarlo obtendra todos los datos de todas las paginas, asi que tomara tiempo

now = datetime.now()
dt_string = now.strftime("%d/%m/%Y %H:%M:%S")
msg = ("Ejecucion: ")
mensaje = msg + dt_string + '\n'
```

Vista del scrapper en general, 4 funciones principales en cuales encapsulo la mayoría de las funciones

```python
from selenium import webdriver
from selenium.webdriver.chrome.options import
import os
from bs4 import BeautifulSoup
import requests
from datetime import datetime
import sqlite3
import andresBd
import math

log_file = "mercado.log"
if not os.path.isfile(log_file):
    open(log_file, 'w+')
```

Importo los módulos que usare y creo el archivo log

Paso la url semilla, y los drivers que será lo que emulara trabajar con un buscador en este caso usare Mozilla, despuesto tomo el html de la pagina y con beutifulsoup lo parseo.

```python
URL_use = "https://listado.mercadolibre.cl/inmuebles/casas/#deal_print_id=a3b5c420-dfc2-11eb-8057-19e52126f5fb&c_id=carousel&c_element_order=2&c_campaign=CASAS&c_uid=a3b5c420-dfc2-11eb-8057-19e52126f5fb"
driver = webdriver.Firefox(executable_path= 'geckodriver') #maneja el navegador en este caso firefox
driver.get(URL_use)
html_code = driver.page_source # tomo el html de la pagina
soup = BeautifulSoup(html_code, 'lxml')#objetohtml y el parse
```

```python
def get_page_url(pageNumber):
    initial_range = 1 + 48 *(pageNumber-1)
    Base_url = "https://listado.mercadolibre.cl/inmuebles/casas/"
    Base_url = Base_url + "_Desde_{}".format(initial_range)
    return Base_url

def get_address(data):
    part = data.split(",")
    partAmount = len(part)
    if(partAmount == 3):
        return {'address' : part[0],'city' : part[1], 'region' : part[2]}
    elif(partAmount > 3):
        return {'address' : " ".join(part[:len(part)-3]), 'city' : part[partAmount-2], 'region' : part[partAmount-1]}
    elif(partAmount < 3):
        for i in range(0,1):
            print("error:")
            print(data)
        return {'address' : 'FAIL','city' : 'FAIL', 'region' : data}
```

La función get_page_url Como su nombre lo dice tomara la url base para luego ir avanzando por las paginas hasta llegar al final, 48 es por la cantidad de ítems en cada pagina de mercado libre.
Luego get_address la uso para separar la dirección en 3 [dirección, ciudad, región] ya que si no me data todo en una cadena

En esta busco todas las casas que estén en esta clase en especifico Luego recorro un con un for para crear el objeto que guardare

```python
def parser_and_save_items(pageUrl):
    driver.get(URL_use)

    all_house_li = soup.find_all("li", class_="ui-search-layout__item")

    for house_li_html in all_house_li:
        house_obj = li_html_to_obj(house_li_html)
        #andresBd.insert_house(house_obj) #NO me hace el insert
        print(house_obj)
```

```python
def li_html_to_obj(house_li_html): #obtenemos los objetos python

    #obtenemos url de imagen
    img = house_li_html.find("img")
    #tengo que hacer ese try ya que mercado libre tiene lazy loading
    try:
        img_url = img['data-src']
    except:
        img_url = img['src']
    #sacamos el precio y le sacamos el punto
    price = house_li_html.find(class_="price-tag-fraction").text.replace(".","")

    #titulo de la publicacion
    title = house_li_html.find(class_="ui-search-item__title").text

    #obtiene dirreccion
    address = house_li_html.find(class_="ui-search-item__location").text
    address = get_address(address)

    #obtiene tamaño y/o cantidad de habitaciones
    all_attribute = house_li_html.find_all(class_= "ui-search-card-attributes__attribute")

    size = ""
    rooms = ""
    if(len(all_attribute) > 0):
        if("útiles" in all_attribute[0].text):
            size = all_attribute[0].text
        else:
            rooms = all_attribute[0].text
    if(len(all_attribute) > 1):
        rooms = all_attribute[1].text
    #obtiene la url de la publicacion
    url = house_li_html.find("a")["href"]
    #   Devuelce el objeto con esta data
    return{"img_url":img_url,"price":price,"title":title,'address':address['address'],
            'city':address['city'],'region':address['region'],"size":size,"rooms": rooms,"url":url }
```

En esta función transformare todo a objeto Python, obteniendo todo los datos importantes que dije en las diapositivas anteriores

Aprovechando que en que tan grande es la casa dice “útiles” usare esto para buscar cuanto miden y también sacare las habitaciones

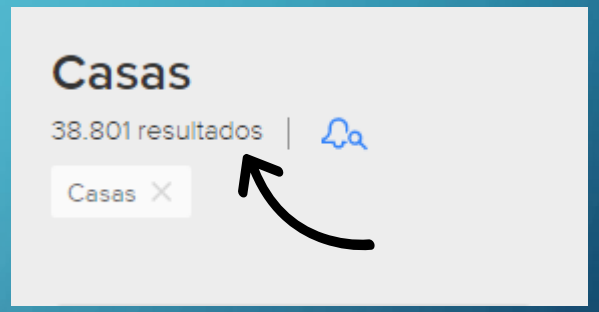
Para devolver el objeto

```python
articles_mount = soup.find(class_="ui-search-search-result__quantity-results")
articles_mount = float(articles_mount.text.split(" ")[0].replace(".", ""))
#print(articles_mount)
#divido de 48 ya que siempre hay 48 resultados en la pagina
page_amount = math.ceil(articles_mount/48)
#print(page_amount)

#aqui debuese recorrer las paginas y guardar en mi base de datos
for actual_number_page in range(1, page_amount+1):
    actul_url = get_page_url(actual_number_page)
    parser_and_save_items(actul_url) #no me inserta en la base
    #al ejercutarlo obtendra todos los datos de todas las paginas, asi que tomara tiempo

now = datetime.now()
dt_string = now.strftime("%d/%m/%Y %H:%M:%S")
msg = ("Ejecucion: ")
mensaje = msg + dt_string + '\n'
file = open(log_file, 'a+')
file.write(mensaje)
```

Con articles-mount busco cuantos resultados hay en la pagina en total y con la ayuda de math hago que quede en un numero entero y recorra todas.



Con el for siguiente recorro TODAS las paginas y guardo con ayuda de la función, debiese insertármelo en la base de datos pero no lo hace.
Luego obtengo la fecha junto con un mensaje y lo guardo en el .log

```
conn = sqlite3.connect('Mercadolibre')

conn.execute('''
        CREATE TABLE casaMercadoLibre
        ( img_url  TEXT PRIMARY KEY,
        price TEXT NOT NULL,
        title        TEXT    NOT NULL,
        address        TEXT    NOT NULL,
        city         TEXT NOT NULL,
        region TEXT    NOT NULL,
        size       TEXT   NOT NULL,
        rooms     TEXT   NOT NULL,
        url      TEXT   NOT NULL);''')

conn.close()
```

Me conecto a la base de datos y creo la tabla

Aquí tengo funciones para insertar conectar y ejecutar

```
import pymysql
import sqlite3
import pymysql.cursors

def get_bd_connection():
    conn = sqlite3.connect('Mercadolibre.db')
    return conn
#no use esta funcion
def create_bd():···
#no use esta funcion
def sql_table(conn):···

def query_execute(query, param = ""):
    bd_con = get_bd_connection()
    db_cursor = bd_con.cursor()
    if(param == ""):
        db_cursor.execute(query)
    else:
        db_cursor.execute(query,param)
    db_cursor.close()
    bd_con.commit()
    bd_con.close()

    return 1

def insert_house(house):
    query_for_insert_house = 'INSERT INTO casaMercadoLibre(img_url,price,title,address,city,region,size,rooms,url) VALUES (%s,%s,%s,%s,%s ,%s,%s,%s,%s)'
    param = (house['img_url'],house['price'],house['title'],house['address'],house['city'],house['region'],house['size'],house['rooms'],house['url'])
    query_execute(query_for_insert_house, param)
    return 1
```

ESAS SON LAS COLUMNAS DE MI BASE DE DATOS

img_url	price	title	address	city	region	size	rooms	url
Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter

Al no poder insertar los datos no puedo mostrar nada, pero realizando un aproximado serian unos 38.000 registros

CONCLUSIONES

- Durante la realización de este trabajo el primer gran problema fue como trabajar con el Scrapper, que información sacar como lograr mostrar y parsear la información para que esta sea comprensible.
- Otro gran problema fue la base de datos que como logramos ver no me inserto los datos, pero a pesar de estas dificultades fue muy entretenido obtener información de esta forma.