TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Sten-Markus Vaska 206333IACB
Oliver Paljak 206640IACB

E-toidupoodide *web scraper* „vasjak”

Tarkvaraprojekti (IAS1410) aruanne

Juhendaja: Trevor Kallaste
MSc

Tallinn 2021

Autorideklaratsioon

Kinnitame, et oleme koostanud antud töö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autorid: Sten-Markus Vaska ja Oliver Paljak

02.05.2021

Lühendite ja mõistete sõnastik

Cache	Puhver; puhverdama; andmeid salvestama nii, et neid saaks vajadusel kiiresti kätte.
GUI	Graafiline kasutajaliides (*Graphical User Interface*); arvuti graafikakuvamise võimalusi kasutav tarkvaraliides, mis teeb programmide kasutamise lihtsamaks. [1]
HTML	Hüpertekst-märgistuskeel (*HyperText Markup Language*); enimlevinud kodeerimissüsteem (tekstivorming) veebidokumentide loomiseks. [1]
JSON	*JavaScript Object Notation*; laialdaselt kasutatav ja kergesti mõistetav andmekirjelduskeel. [2]
Web scraper	Programm, mis saab veebilehtedelt (detailseid) andmeid ja tagastab need kasutajale.

Sisukord

1 Sissejuhatus

Projekti eesmärgiks oli luua terminali kasutajaliidesega *web scraper*, mis lihtsustaks e-toidupoodidest ostmist ja mugavdaks raha säästmist, võimaldades kasutajal otsida programmi kaudu märksõnu kasutades tooteid, mille järel tagastatakse antud toodete eri poekettide hindade võrdlus.

Projekti edukuse objektiivsemaks hindamiseks seati projektile neli mõõdikut:

- programm on võimeline andmeid koguma ja töötlema vähemalt kahelt erinevalt poeketi veebilehelt;
- programm võimaldab otsida mitut toodet korraga (näiteks saab ette anda ostunimekirja faili);
- tulemuse väljastamine kahel viisil – inimloetavas formaadis ja ka andmekirjelduskeeles (JSON) edaspidiseks töötluseks;
- tähtaegadest kinnipidamine – projekt peab saavutama ülejäänud kolm mõõdikut käesoleva semestri jooksul.

2 Töö protsess

2.1 Projekti kulg

Projekti alustati kolmest liikmest koosneva meeskonnana. Kahjuks pidi algselt valitud grupijuht tervislikel põhjustel meeskonnast lahkuma. Ühe liikme lahkumise tõttu tuli ka projekti algset eesmärki muuta – kolme poeketi asemel otsustati teha kahe poeketi hindasi võrdlev programm. Algse grupijuhi lahkumisest tingituna seisis mõnda aega programmi arendamine, kuid ajakavast kinni pidamist see ei mõjutanud. Kõik projekti etapid täideti tähtaegselt graafiku järgi.

2.2 Tööjaotus

Pärast algse grupijuhi lahkumist otsustati uueks pealikuks valida Sten-Markus Vaska. Lõpuks jaotusid meeskonna rollid ära järgnevalt:

Sten-Markus Vaska: grupijuht, ideoloogiline juht, katsetaja

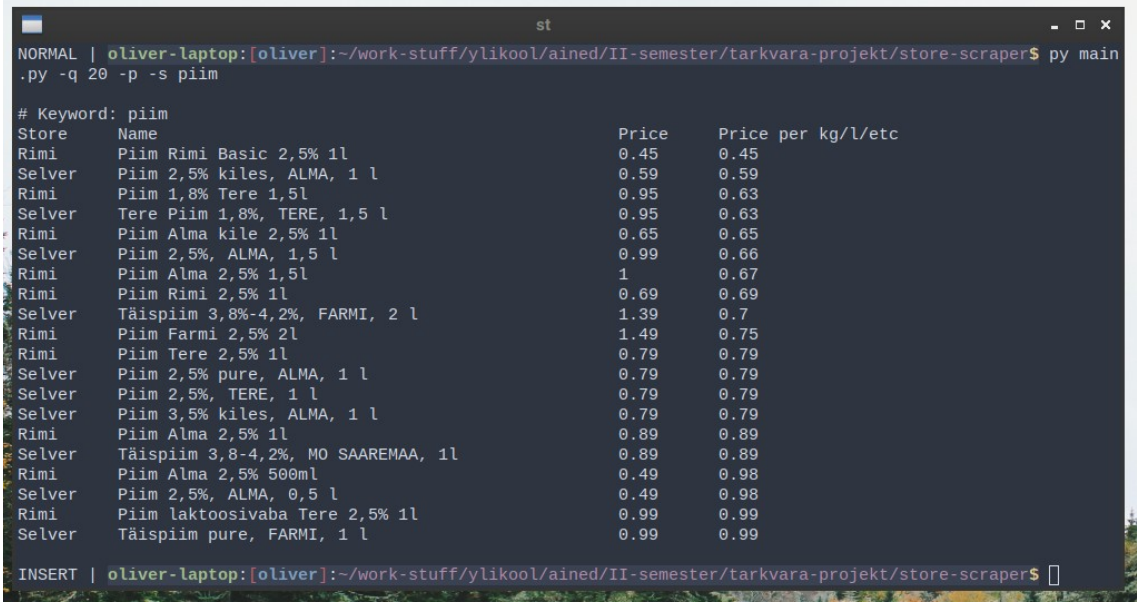
Oliver Paljak: tarkvaraarendaja

Siit tuleneb ka projekti nimi – Vaska ja Paljak ehk vasjak.

Enamuse programmi koodist kirjutas valmis Oliver Paljak. Sten-Markus Vaska mõtles välja, milline võiks programm välja näha, millised funktsioonid võiksid sellel olla ning kontrollis üle valmis kirjutatud koodi. Kuna mõlemad meeskonna liikmed elavad samas linnas, saadi ka mõnel korral kokku ning arendati programmi ühiselt: koos kirjutati koodi ning mõeldi välja erinevaid funktsioone. Projekti alguses plaaniti kokku saada ning koos programmi arendada oluliselt tihedamini, kuid ootamatult laialdase viiruse *Covid-19* leviku tõttu tuli enamik kokkusaamisi ära jätta.

3 Programm

Programm kirjutati programmeerimiskeeles Python [3] ja lisaks kasutati teeke Requests [4] päringute tegemiseks e-poodidele ja BeautifulSoup [5] HTML-ide parsimiseks. Programm on litsentseeritud AGPLv3+ all ja selle lähtekoodi on võimalik vaadata siit: https://github.com/OliverPaljak/vasjak.



Joonis 1: Ekraanitõmmis programmist

3.1 Töö põhimõte

Nagu võib arvata, siis programm oma tööpõhimõttelt sarnaneb tavalise *web scraperiga* – kõigepealt on andmete (vastavate e-toidupoodide HTML-ide) saamine, parsimine ning lõpuks filtreerimine (ehk otsingule vastavate toodete leidmine) ja sorteerimine (ehk toodete järjestamine), et näidata kasutajale relevantseid tulemusi. Lisaks sellele, on programmis kasutatud ka puhverdamist (*cache*'imist) ehk parsitud tooted salvestatakse, et hiljem oleks kasutajal võimalik neid ilma suurema ajakuluta erinevalt filtreerida ja ka sortida. Kuigi alguses ei olnud plaanis puhverdamist lisada,

siis programmi arendamise käigus tuli välja, et siiski mingit vahetulemuste süsteemi oleks vaja ja seda kahel põhjusel: esiteks, lisaterminite kasutamisel ilma puhverdamiseta oleks pidanud põhimõtteliselt tegema samat päringut ja parsimist mitu korda ning teiseks, suuremate toodete koguste puhul oli HTML-ide allalaadimine ja nende parsimine üsna ajamahukas.

Programm on kirjutatud nii, et parsereid oleks mugav muuta ja ka lisada. See tähendab, et ülejäänud programmi osadele läheb ainult korda, et parserid tagastaksid tulemusi kindlal viisil – kuidas parser need tulemused saab, ei ole oluline näiteks sorteerimis- ja filtreerimisprotsessis. Praegusel hetkel on valmis kirjutatud Rimi ja Selveri e-poodide parserid.

3.2 Funktsionaalsus

3.2.1 Sätted

Kuigi tegemist on „lihtsa” *web scraperiga*, siis disainiti programm põhimõttega, et kasutaja saaks muuta võimalikult palju programmi töökorraldust – seega on sätete hulk üsna suur:

- -c – muuda puhvri eluaega sekundites, väärtus 0 keelab puhverdamise, vaikimisi 43200 sekundit ehk 12 h;
- -d – sorteeri kahanevalt;
- -f – ostunimekirja fail;
- -g – liimi samad põhiterminid kokku, oluline säte lisaterminite juures;
- -h – näita abiteksti programmi kohta (vt Lisa 1 – Programmi abitekst);
- -j – anna vastused inimloetava formaadi asemel andmekirjelduskeeles JSON;
- -k – muuda sorteerimisviisi päringus ehk lase e-poel tooted sorteerida, võimalikud väärtused on „default” (ära sorteeri/kasuta e-poe vaikimisi sorteerimismeetodit – tavaliselt otsingutermini relevantsuse järgi), „asc” (sorteeri kasvavalt), „desc” (sorteeri kahanevalt), „asc_per” (sorteeri liitri- või

kilohinna järgi kasvavalt) või „desc_per” (sorteeri liitri- või kilohinna järgi kahanevalt), vaikimisi „default”;

- -l – limiteeri, mitu toodet väljastatakse vastuses (otsingutermini kohta), vaikimisi 20;
- -m – muuda põhitermini otsingurežiimi filtreerimisprotsessis, võimalikud väärtused on „none” (ära kontrolli, kas põhitermin on toote nimes või ei ole), „precise” (põhitermin peab olema toote nimes) või „strict” (põhitermin peab olema toote nimes ja see kas peab olema eraldi sõna või sõna lõpp – näiteks „piim” ja „täispiim”, kuid mitte „piimašokolaad”);
- -o – muuda inimloetava vastuse formaati vastavalt väärtusele, võimalikud väärtused on „n” (toote nimi), „p” (hind), „e” (liitri- või kilohind), „u” (toote URL) ja „s” (poe nimi): näiteks väärtus „p n s” prindib vastuse lahtritega Toote hind, Toote nimi ja Poe nimi;
- -p – sorteeri liitri- või kilohinna järgi;
- -q – limiteeri, mitu toodet päritakse ühest e-poest, vaikimisi 100;
- -s – ära väljasta midagi peale vastuse.

Kuna tegemist on terminali programmiga, siis saab soovitud sätteid rakendada nagu igal teiselgi terminali programmil – lisades vastava sätte tähe (ja väärtuse) käsureal pärast main.py välja kutsumist: `python3 main.py [SÄTTED] PÕHITERMIN[:LISATERMIN...]...` Näiteks, kui soovime otsida e-poodidest tooteid piim ja sink ning tahame, et päring tagastaks ainult esimesed 20 toodet ja sorteeriks need liitri- või kilohinna järgi, siis näeks käsureal olev käsk välja järgmine: `python3 main.py -q 20 -p piim sink`.

3.2.2 Lisaterminid

Üheks suurimaks probleemiks, mis programmi arendamise käigus tekkis, oli nende toodete näitamine kasutajale, mida kasutaja tegelikult soovib näha. Probleem oli seda tõsisem, kuna e-poed tihtipeale tagastavad tooteid, kus otsingutermin ei pea olema toote nimes, vaid võib ka avalduda toote kirjelduses ja kategoorias. Näiteks, kui kasutaja tahab võrrelda erinevate piimade hindasid, siis vaevalt ta soovib näha tulemustes tooteid

nagu kohupiim, piimašokolaad või isegi tooteid nagu kanamunad (näiteks Rimi e-poes asuvad kanamunad piimadega samas kategoorias). Seega oli vaja välja mõelda, kuidas tooteid paremini filtreerida. Üks võimalus selleks on muuta otsingurežiimi sätet (-m), mis kontrollib, kas põhitermin (see otsingutermin, mille põhjal tehti päring) ikka sisaldub toote nimes või ei, kuid sellest jääb väheks – baseerudes eelmisele näitele, siis filtreeritakse välja tooted nagu piimašokolaad ja ka kanamunad, kuid kohupiim jääb. Edasi oleks ainult põhitermini pealt keeruline filtreerida, kuna näiteks täispiim oleks praegusel juhul täiesti loogiline toode, mida vastuses näidata, kuid oma struktuurilt on ta ülimalt sarnane kohupiimaga – seega, põhinedes filtreerimisel ainult põhiterminile, peaksime koos kohupiimaga eemaldama vastusest ka täispiima. Et sellist olukorda vältida ja anda kasutajale rohkem kontrolli filtreerimisprotsessi üle, toodi sisse lisaterminid – need on terminid, mille põhjal päringut ei tehta, kuid neid kasutatakse mittevajalike tootede välja sõelumiseks.

Oma olemuselt on lisatermin lihtne – lisades termini põhiterminile kontrollib programm, kas see avaldub toote nimes või ei. Lisaterminite arv ei ole piiratud ning on võimalik kasutada ka eitust lisades lisatermini algusesesse märgi „^”. Näiteks järgnev käsk `python3 main.py piim:2,5%:^alma` tagastab kõik piimad, mille nimes sisaldub sõne „2,5%”, kuid sõne „alma” ei avaldu. Nagu võib aru saada, siis programm kasutab antud juhul kontrolliks loogilist korrutamist (ehk kas termin1 ja mitte termin2 sisalduvad toote nimes), kuid on võimalik kasutada ka loogilist liitmist (ehk kas üks või teine sõne sisaldub toote nimes). Selleks tuleb kasuks säte -g ehk samade terminite „liimimine”. Näiteks kui sisestada järgnev käsk: `python3 main.py piim:farmi piim:alma`, siis programm tagastab kaks eraldi nimekirja, kus esimeses on välja toodud piimad, mille nimedes sisaldub sõne „farmi”, ja teises piimad, mille nimedes sisaldub sõne „alma”. Kui aga lisada käsule säte -g ehk `python3 main.py -g piim:farmi piim:alma`, siis see ütleb programmile, et sama põhiterminiga otsingud tuleb panna ühte nimekirja, kuid mis kõige olulisem – praeguse käsu juures tagastatakse need piimad (samas nimekirjas), mille nimedes kas sisaldub sõne „farmi” või sõne „alma”.

4 Ideed edasiarenduseks

Kuigi programm on täiesti funktsionaalne, siis on mõningaid mõtteid, mida saab programmi juures täiendada ja täiustada. Esimeseks ideeks oleks tõsta programmi jõudlust HTML-ide allalaadimisel ja parsimisel, kuna need on kõige ajamahukamad tegevused – üheks võimalikuks lahenduseks oleks kasutada paralleelsust – see tähendab, et selle asemel, et tõmmata HTML-id järjest alla ja need parsida (nagu seda tehakse praegu), saaks teha nii, et kõiki HTML-e tõmmatakse alla korraga ja ka parsitakse korraga ühel ja samal ajal. Teiseks täienduseks võiks olla GUI ehitamine tehtud scraperile, mis kindlasti muudaks programmi kasutamise lihtsamaks ja tuttavamaks paljudele kasutajatele. Kolmandaks ideeks oleks loomulikult uute e-poodide parserite lisamine programmile.

5 Kokkuvõte

Projekti eesmärgiks seati luua terminali kasutajaliidesega *web scraper*, mis lihtsustaks e-toidupoodidest ostmist ja mugavdaks raha säästmist, võimaldades kasutajal otsida programmi kaudu märksõnu kasutades tooteid, mille järel tagastatakse antud toodete eri poekettide hindade võrdlus. Lisaks seati projektile mitmeid mõõdikuid, mis aitaks projekti edukust paremini hinnata. Kõik eesmärgid ja mõõdikud saavutati.

Kasutatud kirjandus

[1] e-Teatmik: IT ja sidetehnika seletav sõnaraamat, 2021. [Online]. Loetud aadressil: http://vallaste.ee/ Kasutatud: 02.05.2021.

[2] JSON, *Introducing JSON*, 2021. [Online]. Loetud aadressil: https://www.json.org/json-en.html Kasutatud: 02.05.2021.

[3] Python Software Foundation, *Welcome to Python.org*, 2021. [Online]. Loetud addressil: https://www.python.org/ Kasutatud: 02.05.2021.

[4] Crummy, *Beautiful Soup Documentation*, 2020. [Online]. Loetud aadressil: https://www.crummy.com/software/BeautifulSoup/bs4/doc/ Kasutatud: 02.05.2021.

[5] Requests: HTTP for Humans, *Requests 2.25.1 documentation*, 2019. [Online]. Loetud aadressil: https://docs.python-requests.org/en/master/ Kasutataud: 02.05.2021.

Lisa 1 – Programmi abitekst

```
Usage:
    python3 main.py [OPTION...] SEARCH_TERM[:ADDITIONAL_TERM...]...
Options:
    -c SECONDS
        Change the cache keep-alive time period. 0 disables the cache (no
        reading/writing will be done from/to the cache file). Default is
        43200 second(s).

    -d  Sort descendingly (used in the filtering process). See also the -k
        option. By default not used.

    -f FILE
        Shopping list/input file for search terms. Every search term (with
        its additional terms) should be on its separate line. Can be used
        together with search terms defined on the command line.

    -g  Glue duplicate main terms together. For more information see the
        Additional terms section at end of this help text.

    -h  Print help

    -j  Format the output as JSON, otherwise the output is in human readable
        plain text. By default not used.

    -k default|asc|desc|asc_per|desc_per
        Query sort. When making requests to the e-stores use one of these
        sorting options in the request query. This is different from using the
        options -d and -p as this option will request the products in a
        particular order from e-stores rather than sort them in the filtering
        process/during program execution (i.e. sort using e-store own sorting
        or sort using the request). It is still possible to use program sorting
        (options -d and -p) together with this option. Altough it might seem
        like a good idea to let the e-store handle the sorting, then in reality
        using this option will probably result in inaccurate results when the
        requested product count is small (< 500) and also all the listed
        options do not work in all e-stores. For starters, it is recommended to
        use the default value but experimentation is key...

        default - do not sort at all/use the e-store's default way of
                  sorting (this usually means by relevance or by
                  popularity). Gives the most accurate results.
        asc - sort ascendigly
        desc - sort descendingly
        asc_per - sort ascendingly but use the price per kg/liter/etc
        desc_per - sort descendingly but use the price per kg/liter/etc

        Rimi supports all the values, selver supports default, asc and desc
        values. By default the default value is used.

    -l NUMBER
        Limit the number of products in the answers (per search term). By
        default 20.

    -m none|precise|strict
        Search mode. This option controls how is the search conducted during
        the filtreation process.

        If value "none" is used, then this means that we relay purely on the
        e-store results (no filtering is taken place, except for the additional
        terms if they are provided).

        If value "precise" is used, then it means that the main search term
        (the search term that the query is based off) be inside the product's
        name - often times the e-stores return products that do not have the
```

```
        requested search term inside the product names but appear for example
        in the product's description, category etc. Precise search mode
        prevents this.

        Strict search mode means that the search term must be either a separate
        word or the end of the word inside the products name. For example, if
        the user searches for "piim" (milk in estonian) and strict search is
        enabled, then it means that products like "piim", "täispiim" (whole
        milk in estonian) and "kohupiim" (curd in estonian) will come out but
        products like "piimašokolaad" (milk chocolate) and "kohupiimakreem"
        (curd cream) will be skipped. In technical terms it just means that a
        space will be added to the end of the search term.

        Further narrowing down can be achieved using additional search terms.
        By default "precise".

    -o FORMAT
        The user provided string that the format string is based off. The only
        enabled values/fields in the arg_str are n, p, e, u, s - where:
            n - product name,
            p - price,
            e - price per kg/liter/etc,
            u - product url,
            s - store name
        It is possible to rearrange and leave the fields out (at minimum one
        field is required) and the output will correspond to the provided
        format/arrangment.

        Example: "p n s" - will be translated to fields:
            Price, Product name, Store name

        NOTE: Does not affect the non-human readable (JSON) output

    -p  Sort by price per kg/liter/etc. By default not used.

    -q NUMBER
        Query limit. Limit the number of requested/downloaded products. By
        default 100

    -s  Silent mode. Do not print anything except the end result. By default
        not used.

    Default options can be changed in the config.py file.

Additional terms:
    Additional terms can be used to further narrow down search results. The
    basic usage for additional search terms is the following:
        main_term:add_term1:add_term2:^neg_add_term

    For example "piim:1l:2,5%:^alma" syntax translation is the following:
        piim (milk in estonian) - is the main search term
        1l - is the first additional search term
        2,5% - is the second additional search term
        ^alma - is the third additioanl search term
    and means the following (with precise searching mode): include products
    that have the strings "piim", 1l and 2,5% inside their product name but
    does not have the string "alma" in them (NOTE: if precise or strict
    searching mode is not used, then the word "piim", the main search term,
    does not have to be inside the product's name)

    As one can see, it is like a logical query where every additional search
    term must be inside (or not be inside) the product's name (show the product
    if the name has <term1> AND <term2> AND <term3> AND NOT <term4> ...).
    To use inversion, just put "^" before an additional search term.

    It is also possible to make OR statments. For this comes handy the -g
    option (glue duplicate main terms). The -g option says that do not filter
    duplicate main terms separately.

    For example not using -g option on a query like this one:
        python3 main.py piim:alma piim:farmi
    would just return two (separate) result/lists (first list: milks that have
    the string "alma" in their names and second list: milks that have the
    the string "farmi" in their names). But using the -g option:
        python3 main.py -g piim:alma piim:farmi
```

```
    affects the program so that it would return only one list based on the
    terms - it translates to: show me products that have the words "piim" and
    "alma" in their titles OR show me products that have the words "piim" and
    "farmi" in their titles (in a single list).

    With these tools it is possible to do some advanced queries:
        python3 main.py -g piim:alma piim:farmi piim:rimi:^basic
    which translates to (when using presice or strict searching mode):
    show me products that have the words
        (piim AND alma) OR (piim AND farmi) OR (piim AND rimi AND NOT basic)
    in their names.

    NOTE: The -g option puts only the duplicate terms in a single list. For
    example:
        python3 main.py -g piim:alma piim:farmi vorst:rakvere vorst:rimi
    would put milks ("piim") on the same list and sausageses ("vorst") on
    a different list.

    NOTE: If you want to check whether the characaters ":" and "^" are in the
    product's name, then escape these characters with a backslash (e.g. \: and
    \^)

Examples:
    python3 main.py piim
        Search for piim (milk in estonian) using the default values as defined
        in the config.py

    python3 main.py piim juust
        Search for piim (milk in estonian) and juust (cheese in estonian) using
        the default values as defined in the config.py

    python3 main.py -q 20 vorst
        Search for vorst (sausage in estonian) but limit the requested products
        to 20 (per store)

    python3 main.py -q 20 -l 10 -p juust:viil:^sulatatud
        Search for juust (cheese in estonian) but include only results where
        the word "viil" (slice in estonian) appears but the word "sulatatud"
        does not appear. Also limit the requested products to 20 (per store)
        but show only the first 10 of the parsed products.

    python3 main.py -g piim:alma piim:farmi
        Search for piim (milk in estonian) but show only results that have
        the word "alma" or "farmi" in them (milk producers in Estonia)
```