17.06.2016

Autorzy:

Anna Kuszyńska
Dawid Rdzanek

Podstawy Teleinformatyki – dokumentacja

Webscraper

Spis treści:

1. Wstęp
   1.1. Ogólne założenie projektu
   1.2. Dlaczego wybraliśmy temat
2. Cele i zadania projektu
3. Harmonogram pracy
   3.1. Metodologia
   3.2. Harmonogram
4. Wykorzystane technologie
   4.1. Python
   4.2. MongoDB
   4.3. Beautiful Soup
   4.4. Selenium
5. Wykorzystane narzędzia
   5.1. PyCharm
   5.2. Robomongo
   5.3. Git
6. Implementacja
   6.1. Modele baz danych
   6.2. Worker'y
   6.3. Fragmenty kodu
7. Działanie programu
   7.1. Przygotowywanie projektu do uruchomienia
   7.2. Prezentacja działania

8. Podsumowanie

8.1. Wykonane zadania

8.2. Nie wykonane zadania

8.3. Możliwe kierunki rozbudowy systemu

1. Wstęp

1.1. Ogólne założenia projektu

Naszym zadaniem było stworzenie webscrapera. Zdecydowaliśmy się na zbieranie danych ze strony „www.oddsportal.com”. Zbieramy informacje na temat różnych rodzajów zakładów bukmacherskich na mecze piłki nożnej na całym świecie.

1.2. Dlaczego wybraliśmy temat

Temat wybraliśmy ponieważ wydawał się nam ciekawy. Chcieliśmy zapoznać się z technikami zbieranie i magazynowania informacji.

Dodatkowo chcieliśmy podnieść swoje umiejętności pisania w języku Python. Zapoznaliśmy się z tym językiem już jakiś czas temu i obojgu nam się spodobał ze względu na dostępność wielu bibliotek.

2. Cele i zadania projektu

Celem projektu było stworzenie aplikacji wyciągającej interesujące nas informację ze strony internetowej. Program miał za zadanie automatycznie przechodzić przez odpowiednie podstrony z których kodu źródłowego pobierał pewne dane a następnie zapisywać je do bazy danych w taki sposób aby ułatwić jego późniejsze przetwarzanie. Proces taki nazywamy web scrapingiem.

Web scraping - technika pozyskiwania informacji ze stron internetowych.

Dokładnym zadaniem naszego projektu było opracowanie oraz zaimplementowanie skryptu w języku Python wyciągającego informację o aktualnych w danym momencie kursach meczy piłkarskich z różnych portalów bukmacherskich. Doskonałym portalem zawierającym te informację okazał się http://www.oddsportal.com, na którym dla każdego wydarzenia sportowego wyświetlane są kursy różnego typu zakładów dla różnych portali

3. Harmonogram pracy

3.1. Metodologia

Wykorzystaliśmy lekko zmieniona metodykę SCRUM. Podzieliliśmy projekt na zadania, które przydzieliliśmy poszczególnym osobom w zespole i również mieliśmy sprinty, które zawsze kończyły się działającą wersją programu. Niestety mieliśmy za mało osób w zespole, żeby w całości wykorzystać metodykę SCRUM.

Mieliśmy trzy sprinty. Pierwszy skończył się zbieraniem link'ów meczy ze strony. Po drugim sprincie mieliśmy już program wyciągający wszystkie potrzebne nam informacje ze strony i zapisujący je do naszej bazy danych. W ostatnim sprincie dodaliśmy "worker'y", które wykonują zadania z różną częstotliwością, ta aby nie przeciążać systemu, ale zachować aktualność danych.

3.2. Harmoogram

Osoba	Zadanie
Anna Kuszyńska	Zaprojektowanie bazy danych
Dawid Rdzanek	Stworzenie bazy danych
Anna Kuszyńska	Zbieranie linków meczy
Dawid Rdzanek	Zbieranie danych ze strony
Anna Kuszyńska	Przetwarzanie danych
Dawid Rdzanek	Zapisywanie danych do bazy danych
Dawid Rdzanek	Stworzenie "worker'ów"
Anna Kuszyńska	Podzielenie pracy programu na "worker'y"

4. Wykorzystane technologie

4.1. Python

Cały projekt został napisany w języku Python.
Python jest językiem programowania starającym się zawrzeć w sobie najlepsze rozwiązania i intencje innych języków programowania, które pojawiły się i były udoskonalane w przeciągu ostatniego dwudziestolecia. Jest on interpretowanym, obiektowym, wysokopoziomowym językiem co ułatwia jego testowanie i stosowanie w sposób interaktywny. Python to oprogramowanie typu Open-Source zarządzany przez Python Software Foundation, działające na wielu platformach, takich jak: Linux, Mac OS czy Windows.

Python jest językiem bardzo elastycznym, posiada bogatą bibliotekę standardową oraz ogromną liczbę paczek dostępnych w repozytoriach. Jego wadą jest stosunkowo niska wydajność w porównaniu z językami kompilowanymi (np. C++). Jest to koszt prostoty i szybkości tworzenia oprogramowania.

W naszym przypadku wydajność użytego języka nie miała znaczenia, ponieważ większość czasu aplikacja oczekuję na dane z przeglądarki.

4.2. MongoDB

Do przechowywania zebranych danych wykorzystaliśmy bazę MongoDB. Jest to darmowa, otwarto źródłowa, multiplatformowa nierelacyjna baza danych. W przeciwieństwie do relacyjnych baz danych, dane zapisywane są w dokumentach, które nie posiadają z góry narzuconego schematu. Dzięki takiemu podejściu możemy zapisywać do bazy niejednolite informację o meczu i jego zakładach. Dane zapisywane są w formacie BSON (Binary JSON), czyli binarnej reprezentacji JSON,a (JavaScript Object Notation). Dokumenty wchodzą w skład kolekcji. Porównując do relacyjnego modelu, kolekcje są odpowiednikami tabel, a dokumenty krotek (wierszy).

4.3. Beautiful Soup

Beautiful Soup jest biblioteką w języku python ułatwiająca przetwarzanie znaczników HTML. Biblioteka jest rozwijana od 2004 roku, dzięki czemu można ją nazwać dojrzałą. Posiada bardzo dobrą dokumentację i wsparcie ze strony społeczności.

Głównymi funkcjonalnościami są:

- dostarczenie paru prostych metod do nawigacji, wyszukiwania oraz modyfikowania parsowanego kodu HTML
- automatyczne konwertowanie przychodzącego dokumentu do kodowania Unicode i wychodzącego do UTF-8. Programista nie musi zajmować się dekodowaniem ani kodowaniem.
- dostarczenie wielu popularnych pythonowych pareserów np. lxml czy html5lib, dając możliwość wybory optymalnego dla swojego projektu

Beautiful Soup umożliwia między innymi wyszukiwanie elementów po id, znajdywanie linków (np. wszystkich, zawierających pewną fraze), szukanie po klasach czy nawet stylach css.

Przykładowo, aby wyszukać wszystkie linki na stronie wystarczy wywołać metodę find_all z parametrem ‘li’.

4.4. Selenium

Selenium jest frameworkiem ułatwiającym pisanie testów aplikacji internetowych. Składa się z paru modułów, my jednak korzystaliśmy wyłącznie z jednego - Selenium WebDriver. Za jego pomocą byliśmy w stanie wykonywać kod JavaScript na stronie którą przetwarzaliśmy. Biblioteka ta umożliwia otarcie konkretnej strony w wybranej przez nas przeglądarce internetowej (do dyspozycji mamy większość popularnych przeglądarek), wyszukiwanie, pobieranie i przetwarzanie elementów oraz wykonywanie akcji na stronie (np. kliknięcie, przesunięcie, wypełnienie formularza). Samo parsowanie kodu HTML jest o wiele trudniejsze i bardziej czasochłonne w porównaniu do Beautiful Soup, dlatego staraliśmy się ograniczać wykorzystywanie tego narzędzia do wymaganego minimum. Z naszego punktu najważniejszą możliwością Selenium było pobranie kodu strony po aktualizacji Ajaxowym zapytaniem wywołanym kilknięciem w link.

5. Wykorzystane narzędzia

5.1. PyCharm

Zintegrowane środowisko programistyczne wspomagające pisanie w języku Python stworzone przez czeskiego producenta oprogramowania JetBrains. Oprogramowanie to występuje w dwóch wersjach: darmowa community oraz komercyjna professional. Jako studenci mamy możliwość bezpłatnego korzystania z wersji professional za darmo do celów niekomercyjnych. PyCharm posiada głęboką integrację z językiem Python. Ma możliwość podpowiedzi składni, sprawdzania pisowni, wcięć,integracji z systemami kontroli wersji, pilnuje aby kod był pisany zgodnie z założeniami PEP8, ułatwia refaktoring oraz zarządzanie projektem. Jest to najbardziej kompletne środowisko dla Pythona na rynku.

5.2. Robomongo

Prosty w użyciu, lekki program do zarządzania bazami MongoDB w środowisku graficznym. Aplikacja jest darmowa o ile nie zależy nam na wsparciu ze strony producenta oraz dostępem do najnowszej wersji programu (darmowi użytkownicy otrzymują najnowszą wersję z opóźnieniem). Z poziomu aplikacji możliwe jest zarządzanie bazami, kolekcjami, dokumentami wraz z jego polami, użytkownikami i ich uprawnieniami oraz konfiguracją serwera. Możliwe jest wysyłanie zapytań do serwera. Wyniki są edytowalne i wyświetlane w wybranym przez nas sposobie (np. w formie tabeli czy JSON).

5.3. Git

Jako system kontroli wersji wykorzystaliśmy system Git. Jest on napisany w języku C przez twórce jądra Linuxa Linusa Torvaldsa na licencji GNU GPL v2. Umożliwia on pracę off-line dzięki czemu zmiany możemy zachować lokalnie aż do momentu poprawnego działania pisanej przez nas części. Ze względu na swoją popularność system ten posiada bardzo dobre wsparcie ze strony społeczności. Jako zewnętrzny hosting dla naszego repozytorium użyliśmy serwisu GitHub, który pozwala nam na stworzenie darmowego publicznego repozytorium. Istnieję również płatny plan, dający możliwość dodawania prywatnych repozytoriów. Z poziomu strony mamy możliwość przeglądania szczegółowych informacji o naszym repozytorium oraz kodzie w nim przechowywanym.

6. Implementacja

6.1. Modele baz danych

Do komunikacji z bazą danych wykorzystaliśmy paczkę MongoEngine. Wykorzystuje ona mapowanie obiektowo-relacyjne (ORM) dzięki czemu w prosty sposób mogliśmy korzystać z naszej bazy danych bez potrzeby dogłębnego zapoznania się z javascriptową składnią standardowego sterownika MongoDB.

Utworzone modele:

BaseMatch - klasa abstrakcyjna zawierająca odwzorowanie obiektowe dokumentu meczu

- title - pole typu String. Tytuł meczu: nazwy drużyn oddzielone myślnikiem.
- team_home - pole typu String. Nazwa drużyny gospodarzy.
- team_away - pole typu String. Nazwa drużyny gości.
- league - pole typu String. Nazwa ligi, w ramach której odbywa się mecz.
- country - pole typu String. Państwo, w którym odbywa się mecz.
- match_date - pole typu DateTime. Dzień i godzina rozpoczęcia meczu.
- end_date - pole typu DateTime. Dzień i godzina zakończenia meczu
- bets - lista zawierająca elementy klasy Bets. Lista zakładów na dany mecz.
- link - pole typu URLField. URL danego meczu.
- create_date - pole typu DateTime. Data dodania informacji o meczu do bazy danych.

FinishedMatch - klasa dziedzicząca po BaseMatch

- score - pole typu String. Zawiera wynik meczu.

Bets - klasa dokumentu wbudowanego (Embedded Document)

- one_x_two - pole dokumentu wbudowanego klasy OneXTwo, zawiera informacje o zakładach 1X2
- asian_handicap - pole dokumentu wbudowanego klasy AsianHandicap, zawiera informacje o zakładach Asian Handicap
- over_under - pole dokumentu wbudowanego klasy OverUnder, zawiera informacje o zakładach Over Under
- draw_no_bet - pole dokumentu wbudowanego klasy DrawNoBet, zawiera informacje o zakładach Draw No Bet
- european_handicap - pole dokumentu wbudowanego, zawiera informacje o zakładach European Handicap
- double_chance - pole typu pole dokumentu wbudowanego klasy DoubleChance, zawiera informacje o zakładach Double Chance

- correct_score - pole dokumentu wbudowanego klasy CorrectScore, zawiera informacje o zakładach Correct Score
- bets_date - pole typu DateTime. Zawiera date zebrania informacji o zakładach.
- time_to_end_match - pole typu DateTime. Czas do zakończenia meczu.

OneXTwo - klasa dokumentu wbudowanego (Embedded Document)

- full_time - lista wbudowanych dokumentów klasy Bet1X2
- first_half = - lista wbudowanych dokumentów klasy Bet1X2
- secound_half = - lista wbudowanych dokumentów klasy Bet1X2
- full_time_avg - liczba zmiennoprzecinkowa, zawiera średnią wartość typów ze wszystkich zakładów bukmacherskich dla całego meczu
- first_half_avg - liczba zmiennoprzecinkowa, zawiera średnią wartość typów ze wszystkich zakładów bukmacherskich dla pierwszej połowy meczu
- secound_half_avg - liczba zmiennoprzecinkowa, zawiera średnią wartość typów ze wszystkich zakładów bukmacherskich dla drugiej połowy meczu

DrawNoBet - klasa dokumentu wbudowanego (Embedded Document)

- full_time - lista wbudowanych dokumentów klasy BetDNB
- first_half = - lista wbudowanych dokumentów klasy BetDNB
- secound_half = - lista wbudowanych dokumentów klasy BetDNB
- full_time_avg - liczba zmiennoprzecinkowa, zawiera średnią wartość typów ze wszystkich zakładów bukmacherskich dla całego meczu
- first_half_avg - liczba zmiennoprzecinkowa, zawiera średnią wartość typów ze wszystkich zakładów bukmacherskich dla pierwszej połowy meczu
- secound_half_avg - liczba zmiennoprzecinkowa, zawiera średnią wartość typów ze wszystkich zakładów bukmacherskich dla drugiej połowy meczu

DoubleChance - klasa dokumentu wbudowanego (Embedded Document)

- full_time - lista wbudowanych dokumentów klasy BetDC
- first_half = - lista wbudowanych dokumentów klasy BetDC
- secound_half = - lista wbudowanych dokumentów klasy BetDC
- full_time_avg - liczba zmiennoprzecinkowa, zawiera średnią wartość typów ze wszystkich zakładów bukmacherskich dla całego meczu
- first_half_avg - liczba zmiennoprzecinkowa, zawiera średnią wartość typów ze wszystkich zakładów bukmacherskich dla pierwszej połowy meczu
- secound_half_avg - liczba zmiennoprzecinkowa, zawiera średnią wartość typów ze wszystkich zakładów bukmacherskich dla drugiej połowy meczu

OverUnder - klasa dokumentu wbudowanego (Embedded Document)

- full_time - lista wbudowanych dokumentów klasy BetOU
- first_half = - lista wbudowanych dokumentów klasy BetOU
- secound_half = - lista wbudowanych dokumentów klasy BetOU

CorrectScore - klasa dokumentu wbudowanego (Embedded Document)

- full_time - lista wbudowanych dokumentów klasy BetCS
- first_half - lista wbudowanych dokumentów klasy BetCS
- secound_half - lista wbudowanych dokumentów klasy BetCS

AsianHandicap - klasa dokumentu wbudowanego (Embedded Document)

- full_time - lista wbudowanych dokumentów klasy BetAH
- first_half - lista wbudowanych dokumentów klasy BetAH
- secound_half - lista wbudowanych dokumentów klasy BetAH

EuropeanHandicap - klasa dokumentu wbudowanego (Embedded Document)

- full_time - lista wbudowanych dokumentów klasy BetOU
- first_half - lista wbudowanych dokumentów klasy BetOU
- secound_half - lista wbudowanych dokumentów klasy BetOU

Bet1X2 - klasa dokumentu wbudowanego (Embedded Document)

- bookmaker - pole tekstowe przechowujące nazwę portalu bukmacherskiego
- _1 - liczba zmiennoprzecinkowa przechowująca kurs
- _X - liczba zmiennoprzecinkowa przechowująca kurs
- _2 - liczba zmiennoprzecinkowa przechowująca kurs
- payout - liczba zmiennoprzecinkowa przechowująca procent sumy wypłaconej graczom

BetDNBklasa dokumentu wbudowanego (Embedded Document)

- bookmaker - pole tekstowe przechowujące nazwę portalu bukmacherskiego
- _1 - liczba zmiennoprzecinkowa przechowująca kurs
- _2 - liczba zmiennoprzecinkowa przechowująca kurs
- payout - liczba zmiennoprzecinkowa przechowująca procent sumy wypłaconej graczom

BetDC- klasa dokumentu wbudowanego (Embedded Document)

- bookmaker - pole tekstowe przechowujące nazwę portalu bukmacherskiego
- _1 - liczba zmiennoprzecinkowa przechowująca kurs
- _X - liczba zmiennoprzecinkowa przechowująca kurs
- _2 - liczba zmiennoprzecinkowa przechowująca kurs
- payout - liczba zmiennoprzecinkowa przechowująca procent sumy wypłaconej graczom

BetCS- klasa dokumentu wbudowanego (Embedded Document)

- score- pole tekstowe przechowujące wynik dla którego jest kurs
- odds - liczba zmiennoprzecinkowa przechowująca kurs

BetAH- klasa dokumentu wbudowanego (Embedded Document)

- handicap- pole tekstowe przechowujące rodzaj handicapu
- _1 - liczba zmiennoprzecinkowa przechowująca kurs
- _2 - liczba zmiennoprzecinkowa przechowująca kurs
- payout - liczba zmiennoprzecinkowa przechowująca procent sumy wypłaconej graczom

BetOU - klasa dokumentu wbudowanego (Embedded Document)

- handicap- pole tekstowe przechowujące rodzaj handicapu
- over - liczba zmiennoprzecinkowa przechowująca kurs
- under - liczba zmiennoprzecinkowa przechowująca kurs
- payout - liczba zmiennoprzecinkowa przechowująca procent sumy wypłaconej graczom

BetEH - klasa dokumentu wbudowanego (Embedded Document)

- handicap- pole tekstowe przechowujące rodzaj handicapu
- _1 - liczba zmiennoprzecinkowa przechowująca kurs
- _X - liczba zmiennoprzecinkowa przechowująca kurs
- _2 - liczba zmiennoprzecinkowa przechowująca kurs
- payout - liczba zmiennoprzecinkowa przechowująca procent sumy wypłaconej graczom

6.2. Worker'y

Workery, czyli w naszym przypadku procesy. Każdy z nich ma inną rolę i wykonuje swoje zadania z różną częstotliwością. Każdy z workerów posiada obiekt własnej klasy DataProvider, który jest odpowiedzialny za równoległe, okresowe aktualizowanie danych potrzebnych workerowi. Każdy dostawca danych jest osobnym wątkiem i posiada własny czas na jaki zostaje uśpiony po aktualizacji danych.
Zaimplementowaliśmy 5 takich worker'ów:

- TodayWorker - proces zajmujący się odświeżaniem informacji na temat zakładów meczy odbywających się danego dnia. Uruchamiany co 10 minut. Dane aktualizowane co 15 min.

- CurrentWorker - proces zajmujący się odświeżaniem danych na temat aktualnie trwających meczy. Uruchamiany co 2 minuty. Dane aktualizowane co minutę.

- WeekWorker - proces zajmujący się odświeżaniam informacji na temat meczy, które mają się odbyć za maksymalnie tydzień. Uruchamiany co 30 minut. Dane aktualizowane co 15 minut.

- NewWorker - proces zajmujący się dodawaniem nowych meczy do bazy. Uruchamiany co godzinę. Worker posiada blokadę, która jest nakładana na dobę po wykonaniu się w całości. Uniemożliwia to dodania tych samych meczy po raz drugi przy restarcie programu.

- EndedWorker - proces zajmujący się przenoszeniem w bazie meczy już zakończonych z modelu Match do modelu FinishedMatch. Uruchamiany co 30 minut. Dane aktualizowane co 25 minut.

6.3. Fragmenty kodu

```python
class NewMatchScraper:

    def __init__(self):
        self.driver = webdriver.Opera()
        self.base_url = 'http://www.oddsportal.com/matches/soccer/'
        self.url = self.get_page_url()

    def __enter__(self):
        return self

    def __exit__(self, exc_type, exc_val, exc_tb):
        self.driver.close()

    def get_source_code(self):
        self.driver.get(self.url)
        return BeautifulSoup(self.driver.page_source, 'html.parser')

    def get_page_url(self):
        date = str(datetime.datetime.now(utc) + datetime.timedelta(days=6))
        date = date.split()[0].replace('-', '')
        return self.base_url + date

    def get_links(self):
        page_source = self.get_source_code()
        links = []
        main_table = page_source.find('table', {'class': ' table-main'})
        tbody = main_table.find('tbody')
        black_list = tbody.select('.dark')
        trs = [tr for tr in tbody.find_all('tr') if tr not in black_list]
        for tr in trs:
            all_a = tr.find_all('a')
            for a in all_a:
                if str(a.get('href')).startswith('/soccer/'):
                    links.append('http://www.oddsportal.com' + a.get('href'))
        return links
```

Klasa scrapera służąca do wyszukiwania nowych meczy. Wykorzystywana jest przez NewWorkera.

Metoda get_page_url zwraca adres strony, gdzie przechowywane są mecze, które odbędą się za 6 dni.

Metoda get_links przeszukuję stronę z get_page_url i zwraca listę linków do nowych meczy.

```
bets = {
    '1X2': {
        'major_model': OneXTwo,
        'minor_model': Bet1X2,
        'minor_fields': ['bookmaker', ' 1', ' X', ' 2', 'payout'],
        'bets_field': 'one_x_two',
    },
    'DNB': {
        'major_model': DrawNoBet,
        'minor_model': BetDNB,
        'minor_fields': ['bookmaker', '_1', '_2', 'payout'],
        'bets_field': 'draw_no_bet',
    },
    'DC': {
        'major_model': DoubleChance,
        'minor_model': BetDC,
        'minor_fields': ['bookmaker', '_1X', '_12', '_X2', 'payout'],
        'bets_field': 'double_chance',
    },
    'AH': {
        'major_model': AsianHandicap,
        'minor_model': BetAH,
        'minor_fields': ['handicap', '_1', '_2', 'payout'],
        'bets_field': 'asian_handicap',
    },
    'O/U': {
        'major_model': OverUnder,
        'minor_model': BetOU,
        'minor_fields': ['handicap', 'over', 'under', 'payout'],
        'bets_field': 'over_under',
    },
    'EH': {
        'major_model': EuropeanHandicap,
        'minor_model': BetEH,
        'minor_fields': ['handicap', '_1', '_X', '_2', 'payout'],
        'bets_field': 'european_handicap',
    },
}
```

Dict (słownik) zawierający konfigurację rodzajów zakładów. Wykorzystywany głównie w klasie MatchPageScraper do tworzenia obiektów modeli z pobranych danych.

Kluczami są nazwy rodzajów zakładów używanych na przeszukiwanym portalu w kodzie HTML. Wartością dla klucza jest kolejny dict zawierający informacje o modelach nadrzędnych i podrzędnych, o polach na dane ze strony oraz nazwie pola w głównym modelu Bets.

```
def get_bets(self):
    bets_obj = Bets()
    links = self.get_types_links()
    for link in links:
        skipped = False
        link['bet_type'].click()
        time.sleep(1)
        time_type = self.driver.find_element_by_id('bettype-tabs-scope')
        time_type_uls = time_type.find_elements_by_tag_name('ul')
        for ul in time_type_uls:
            if ul.get_attribute('style') == 'display: block;':
                lis = ul.find_elements_by_tag_name('li')
                link['time_types'].extend(lis)
        bet_config = bets[link['bet_name']]
        major_obj = bet_config['major_model']()
        for bet in link['time_types']:
            bet.click()
            time.sleep(1)
            if link['bet_name'] == 'AH':
                data = self.get_asian_handicap_bets_data()
            elif link['bet_name'] == 'O/U':
                data = self.get_over_under_bets_data()
            elif link['bet_name'] == 'EH':
                data = self.get_european_handicap_bets_data()
            else:
                data = self.get_bets_data()
                pass
            data[:] = [item for item in data if len(item) ==
                                        len(bet_config['minor_fields'])]
            time_type = bet.text
            print('typ={}, czas={}, bety={}'.format(link['bet_name'],
                                                    time_type,data))
            try:
                if '/' in data[0][1] or '+' in data[0][1] or '-' in
                                                    data[0][1]:
                    print('skipped')
                    skipped = True
                    break
            except IndexError:
                pass
            for i in range(len(data)):
                minor_obj = bet_config['minor_model']()
                for j, field in enumerate(bet_config['minor_fields']):
                    if field not in ['bookmaker', 'handicap']:
                        setattr(minor_obj, field, float(data[i][j]))
                    else:
                        setattr(minor_obj, field, data[i][j])
                getattr(major_obj, time_types[time_type]).append(minor_obj)
        if not skipped:
            setattr(bets_obj, bet_config['bets_field'], major_obj)
    return bets_obj
```

Metoda get_bets w klasie MatchPageScraper jest sercem naszego programu. Tworzy ona obiekt klasy Bets a następnie tworzy podrzędne modele, uzupełnia je pobranymi ze strony danymi, łączy wszystko w całość i zwraca wypełniony obieky klasy Bets.

```
def get_asian_handicap_bets_data(self):
    page_source = self.get_source_code()
    data = []
    odds_data_table = page_source.find('div', {'id': 'odds-data-table'})
    table header light = odds data table.select('div[class^=table-header-
                                                                light]')
    for tag in table_header_light:
        text = tag.text.split('(')
        if text[1].startswith('0'):
            continue
        data_text = text[0].split()
        handicap = ' '.join(data_text[:3])
        bet_values = data_text[3].split('%')
        payout = bet_values[0]
        values = bet_values[1].split('.')
        _1 = ''.join((values[0], '.', values[1][:2]))
        _2 = ''.join((values[1][2:], '.', values[2]))
        data.append([handicap, _1, _2, payout])
    return data

def get_european_handicap_bets_data(self):
    page_source = self.get_source_code()
    data = []
    odds_data_table = page_source.find('div', {'id': 'odds-data-table'})
    table_header_light = odds_data_table.select('div[class^=table-header-
                                                                light]')
    for tag in table_header_light:
        text = tag.text.split('(')
        if text[1].startswith('0'):
            continue
        data_text = text[0].split()
        handicap = ' '.join(data text[:3])
        bet_values = data_text[3].split('%')
        payout = bet_values[0]
        values = bet_values[1].split('.')
        _1 = ''.join((values[0], '.', values[1][:2]))
        _X = ''.join((values[1][2:], '.', values[2][:2]))
        _2 = ''.join((values[2][2:], '.', values[3]))
        data.append([handicap, _1, _X, _2, payout])
    return data

def get_over_under_bets_data(self):
    page_source = self.get_source_code()
    data = []
    odds_data_table = page_source.find('div', {'id': 'odds-data-table'})
    table_header_light = odds_data_table.select('div[class^=table-header-
                                                                light]')
    for tag in table_header_light:
        text = tag.text.split('(')
        if text[1].startswith('0'):
            continue
        data_text = text[0].split()
        handicap = ' '.join(data_text[:2])
        bet_values = data_text[2].split('%')
        payout = bet_values[0]
        values = bet_values[1].split('.')
        over = ''.join((values[0], '.', values[1][:2]))
        under = ''.join((values[1][2:], '.', values[2]))
        data.append([handicap, over, under, payout])
    return data
```

Metody te pobierają dane z odpowiednich rodzajów zakładów.

```
def get_score(self):
    page_source = self.get_source_code()
    score = page_source.select_one('.result')
    score_text = score.text.replace('Final result ', '')
    return score_text
```

Metoda pobiera wynik z zakończonego meczu.

```
def get_match_info(self):
    page_source = self.get_source_code()
    page_title = page_source.title.string.replace('Betting Odds', '')
    splited_page_title = page_title.split(',')
    title = splited_page_title[0].rstrip()
    team_home = title.split('-')[0].rstrip()
    team_away = title.split('-')[1].rstrip().lstrip()
    league = splited_page_title[1].split('-')[1].lstrip()
    match_date_str = page_source.select("p[class^=date]")[0].string.replace
                                                    (',', '').replace('  ', ' ')
    match_date = datetime.datetime.strptime(match_date_str, '%A %d %b %Y
                                                                %H:%M')
    end_date = match_date + datetime.timedelta(minutes=110)
    info = {
        'title': title,
        'team_home': team_home,
        'team_away': team_away,
        'league': league,
        'match_date': match_date,
        'end_date': end_date,
        'link': self.driver.current_url,
    }
    return info

def get_score(self):
    page_source = self.get_source_code()
    score = page_source.select_one('.result')
    score_text = score.text.replace('Final result ', '')
    return score_text
```

Metoda klasy MatchPageScraper przeszukująca stronę meczu i wyciągająca informacje o zespołach, lidze, datach meczu tytule orazaz linku.

```
class Worker(Process):

    def __init__(self, worker_delay, model, data_delay=10, name='Worker'):
        super(Worker, self).__init__()
        self.worker_delay = worker_delay
        self.data_delay = data_delay
        self.name = name
        self.model = model
        self.data_provider = None

    def run(self):
        pass


class DataProvider(Thread):
    data = []

    def __init__(self, delay, model, name):
        super().__init__()
        self.delay = delay
        self.model = model
        self.data = []
        self.name = name
        self.setName('{}\'s data provider'.format(self.name))

    def update(self):
        pass

    def run(self):
        from os import getpid
        print('Running thread "{}" on process {}'.format(self.getName(),
                                                         getpid()))
        while True:
            self.update()
            sleep(self.delay)
```

Klasy bazowe workerów oraz data providerów. Zawierają szkielet klas dziedziczących.

7. Działanie programu

7.1. Przygotowywanie projektu do uruchomienia

- Tworzymy folder projektu
  mkdir scraper
- Przechodzimy do folderu
  cd scraper
- Pobieramy projekt z repozytorium
  git clone https://github.com/akus5/Webscraper
- instalujemy virtualenv i w katalogu poziom wyżej go tworzymy

pip install virtualenv
virtualenv ../venv

- przełączamy nasz shell na virtualenv
  source ../venv/bin/activate
- instalujemy potrzebne paczki dla naszego wirtualnego środowiska
  pip install -r requirements.txt
- pobieramy, rozpakowujemy i dodajemy do naszego systemowego PATHa webdriver Chroma z https://sites.google.com/a/chromium.org/chromedriver/downloads
- uruchamiamy program
  python ./webscraper/main.py

7.2. Prezentacja działania

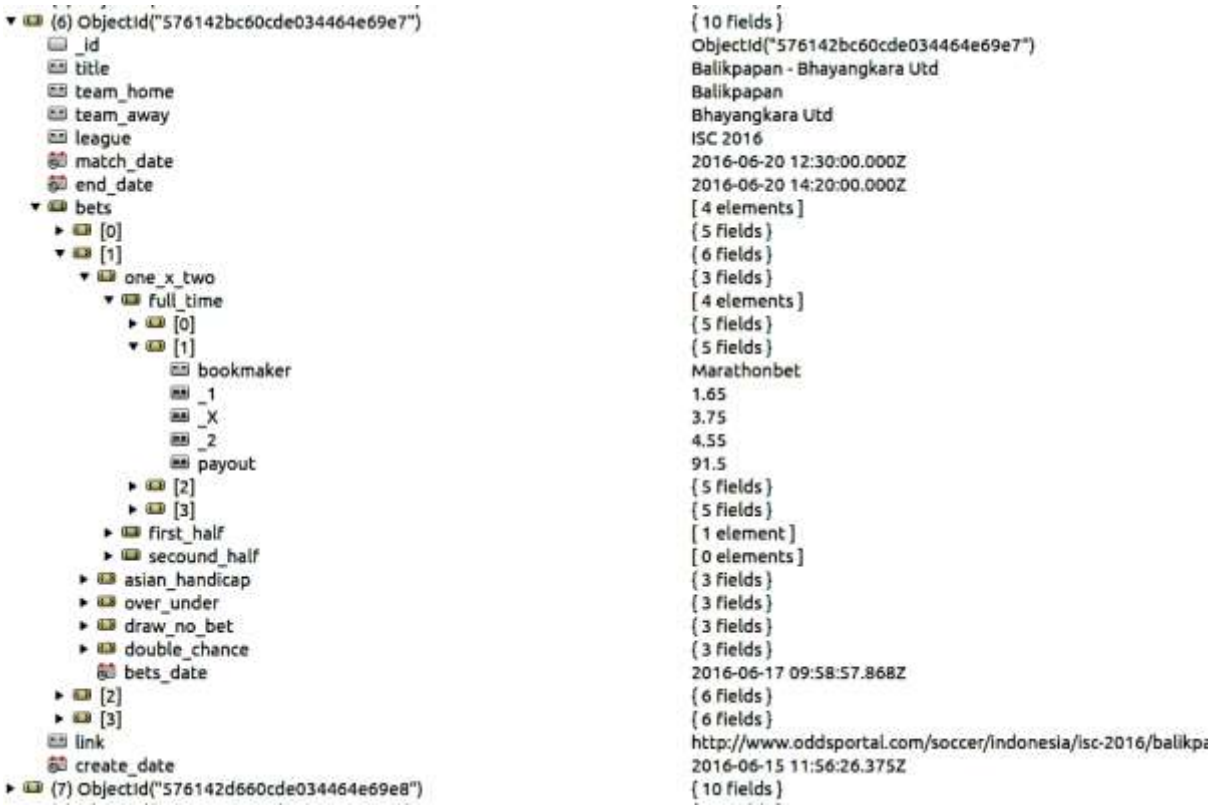
Po uruchomieniu programu tworzone i uruchamiane są wszystkie workery oraz ich dostawcy danych.

```
Running Today Worker on PID 13114
Running Current Worker on PID 13115
Running thread "Today Worker's data provider" on process 13114
Running New Worker on PID 13118
Running thread "Current Worker's data provider" on process 13115
Running Ended Worker on PID 13120
Running Week Worker on PID 13116
Running thread "Ended Worker's data provider" on process 13120
Running thread "Week Worker's data provider" on process 13116
Znaleziono 0 dzisiajszych meczy
Znaleziono 0 dzisiajszych meczy
Znaleziono 0 trwających meczy
Znaleziono 0 trwających meczy
Znaleziono 0 zakończonych meczy
Znaleziono 0 zakończonych meczy
Nie upłyneła doba od ostatniego sprawdzania meczów!
Znaleziono 79 meczy w tym tygodniu
Znaleziono 79 meczy w tym tygodniu
```

Logi dotyczące wyszukiwania danych są wyświetlane podwójnie ponieważ na starcie sam worker inicjalizuje aktualizacje danych, ponieważ ze względu na współbierzność może się wykonać szybciej niż jego dostawca.

```
typ=1X2, czas=Full Time, bety=[['1xBet', '1.67', '3.74', '4.46', '91.7'], ['Marathonbet', '1.65', '3.75', '4.55', '91.5'], ['Oto
typ=1X2, czas=1st Half, bety=[['Marathonbet', '2.21', '2.18', '5.00', '90.0']]
typ=AH, czas=Full Time, bety=[['Asian handicap -1', '1.60', '2.25', '93.5']]
typ=AH, czas=1st Half, bety=[['Asian handicap 0', '2.88', '1.35', '91.9']]
typ=O/U, czas=Full Time, bety=[['Over/Under +2.5', '1.82', '1.92', '93.4']]
typ=O/U, czas=1st Half, bety=[['Over/Under +1.5', '1.41', '2.64', '91.9']]
typ=DNB, czas=Full Time, bety=[['Winlinebet', '1.27', '3.38', '92.3']]
typ=DC, czas=Full Time, bety=[['1xBet', '1.14', '1.20', '2.03', '90.8'], ['Marathonbet', '1.14', '1.20', '2.03', '90.8'], ['Winl
typ=DC, czas=1st Half, bety=[['Marathonbet', '1.10', '1.53', '1.52', '90.1']]
Zaktualizowano mecz Balikpapan - Bhayangkara Utd
typ=1X2, czas=Full Time, bety=[['1xBet', '1.67', '3.74', '4.46', '91.7'], ['Marathonbet', '1.65', '3.75', '4.55', '91.5'], ['Oto
typ=1X2, czas=1st Half, bety=[['Marathonbet', '2.21', '2.18', '5.00', '90.0']]
typ=AH, czas=Full Time, bety=[['Asian handicap -1', '1.60', '2.25', '93.5']]
typ=AH, czas=1st Half, bety=[['Asian handicap 0', '2.88', '1.35', '91.9']]
typ=O/U, czas=Full Time, bety=[['Over/Under +2.5', '1.82', '1.92', '93.4']]
typ=O/U, czas=1st Half, bety=[['Over/Under +1.5', '1.41', '2.64', '91.9']]
typ=DNB, czas=Full Time, bety=[['Winlinebet', '1.27', '3.38', '92.3']]
typ=DC, czas=Full Time, bety=[['1xBet', '1.14', '1.20', '2.03', '90.8'], ['Marathonbet', '1.14', '1.20', '2.03', '90.8'], ['Winl
typ=DC, czas=1st Half, bety=[['Marathonbet', '1.10', '1.53', '1.52', '90.1']]
Zaktualizowano mecz Balikpapan - Bhayangkara Utd
```

Podczas aktualizowania danych dotyczących zakładów wyświetlane są logi z informacjami o pobranych danych oraz zaktualizowanych meczach.



Przykładowy dokument meczu.

8. Podsumowanie

8.1. Wykonane zadania

- Zaprojektowanie bazy danych
- Zbieranie informacji o meczach
- Zapisywanie danych do bazy danych
- Podzielenie pracy programu na workery

8.2. Nie wykonane zadania

Nie udało nam się niestety przetwarzać zebranych danych. Problem był z zostawieniem uruchomionego programu na parę dni, tak aby informacje były zbierane na bierząco tak jak zostało to zaimplementowane.

8.3. Możliwe kierunki rozbudowy systemu

Przyspieszenie działania programu oraz wykorzystanie danych do stworzenia modelu i przewidywania opłacalności zakładów.