[**0. Diagramy 4**](#_h2igj0w7pm4q)

[0.1 Diagram przypadków użycia 4](#_8wm44kwn2ew)

[0.2 Diagram wdrożeniowy 5](#_eskmd4i6sz2f)

[**1. Dokumentacja techniczna - Skrypt automatycznego testowania strony e-commerce w Selenium i Python 6**](#_q9jmm8ruun6e)

[1.0 Wykorzystywany kod 6](#_e516z1jt6flx)

[1.1 Opis 6](#_dnz82cab5wug)

[1.2 Użyte technologie i ich cel 6](#_bjiahw2sd1nq)

[1.2.1. Selenium: 6](#_1u4ptqgmxbfl)

[1.2.2. Python: 6](#_unnf0ekgz9nl)

[1.2.3. WebDriver: 7](#_es7h6fzd1zy6)

[1.2.4. XPath: 7](#_1k7odkmjsbkh)

[1.2.5. Pakiety biblioteczne: 7](#_zcdqgmwbsbxs)

[1.3. Przygotowanie środowiska 7](#_h7ap7a81vnn7)

[1.4. Inicjalizacja sterownika 8](#_pw2hcsurqp8m)

[**2. Dokumentacja techniczna skryptu testującego REST API 9**](#_2w1q72rvcxwp)

[2.1. Opis 9](#_7x3ed64rubkh)

[2.2. Wykorzystane biblioteki i środowisko 9](#_h6xe37z2iyuh)

[2.3. Funkcje skryptu 9](#_anxawadw7jhm)

[2.3.1. create\_user(username, password, email, first\_name, last\_name) 10](#_7nfacw76p5sf)

[2.3.2. login\_with\_token(auth\_token) 11](#_5tivotlrwhzn)

[2.3.3. upload\_image() 11](#_xivhm5opvupe)

[2.3.4. create\_new\_pets() 12](#_oekuw7hu8jau)

[2.3.5. search\_pets() 13](#_5u9pd3xj4jqt)

[2.3.6. update\_pet() 13](#_fc5pa0csc7ck)

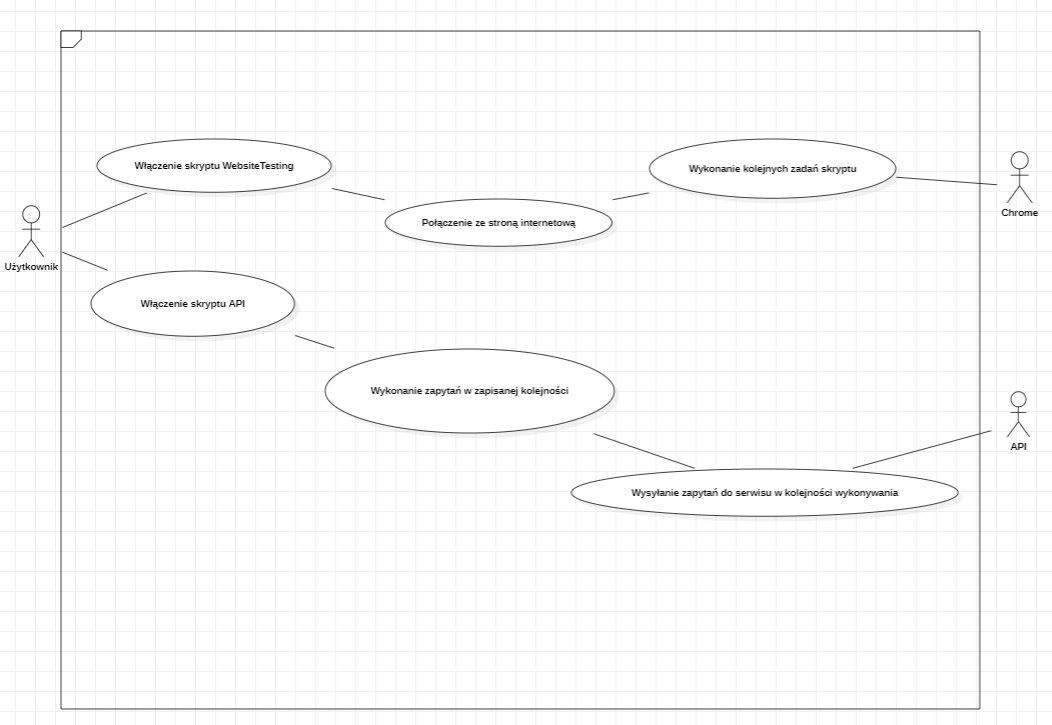
[2.3.7. delete\_pet() 14](#_sinrs271w7kl)

[2.3.8. place\_orders() 15](#_k5lssy25cqlf)

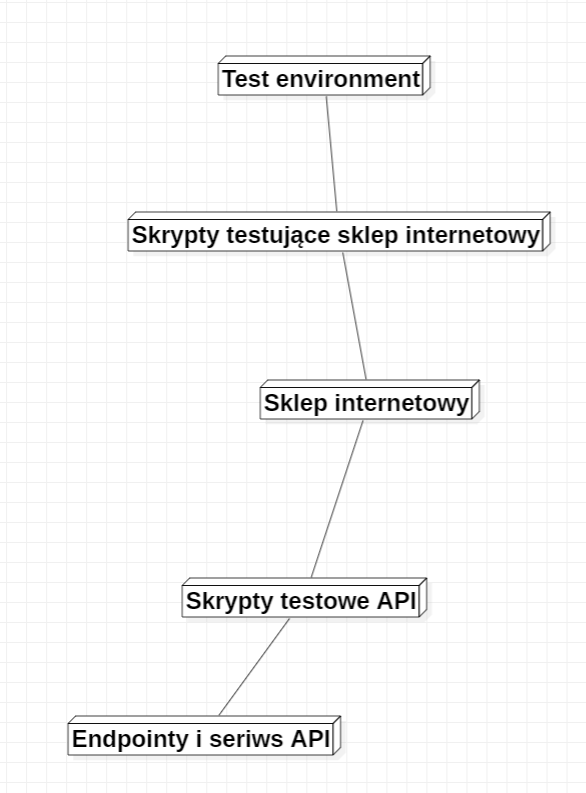
[2.3.9. find\_orders() 16](#_689ou8cvqh1q)

# **0. Diagramy**

## **0.1 Diagram przypadków użycia**



## **0.2 Diagram wdrożeniowy**



# **1. Dokumentacja techniczna - Skrypt automatycznego testowania strony e-commerce w Selenium i Python**

## **1.0 Wykorzystywany kod**

* Link do kodu używanego w projekcie: https://github.com/MateuszNowocin/AutomaticPythonTests

## **1.1 Opis**

Ten skrypt automatyzuje testowanie strony e-commerce przy użyciu biblioteki Selenium w języku Python. Skrypt wykonuje serię czynności, takich jak logowanie, dodawanie produktów do koszyka, usuwanie z koszyka i przeprowadzanie procesu zamówienia. Każda czynność jest opisana kroku po kroku, a na końcu skrypt oczekuje na wyświetlenie się komunikatu potwierdzającego zakończenie zamówienia.

## **1.2 Użyte technologie i ich cel**

Skrypt automatycznego testowania strony e-commerce w Selenium i Python wykorzystuje kilka technologii, które są istotne dla realizacji testów automatycznych. Poniżej przedstawiam opis tych technologii oraz cel ich użycia w kontekście skryptu.

### 1.2.1. Selenium:

Selenium to popularny zestaw narzędzi do automatyzacji testów aplikacji internetowych. W skrypcie użyto biblioteki Selenium WebDriver, która umożliwia interakcję ze stroną internetową poprzez symulację działań użytkownika. Selenium zapewnia funkcje takie jak znajdowanie elementów na stronie, wprowadzanie danych, wykonywanie kliknięć itp. Cel użycia Selenium w skrypcie polega na automatycznym testowaniu interakcji użytkownika z aplikacją e-commerce i weryfikowaniu jej poprawności.

### 1.2.2. Python:

W skrypcie użyto Pythona jako głównego języka programowania do implementacji testów automatycznych. Python oferuje bogatą bibliotekę standardową, w tym Selenium, co czyni go dobrym wyborem do automatyzacji testów. Cel użycia Pythona polega na stworzeniu czytelnego, elastycznego i łatwo rozbudowywalnego skryptu automatycznego testowania.

### 1.2.3. WebDriver:

WebDriver to komponent Selenium, który umożliwia komunikację między skryptem a przeglądarką internetową. W skrypcie użyto WebDrivera w połączeniu z przeglądarką Google Chrome poprzez sterownik Chrome WebDriver. Cel użycia WebDrivera polega na sterowaniu przeglądarką, otwieraniu stron internetowych, znajdowaniu elementów na stronie, wprowadzaniu danych itp.

### 1.2.4. XPath:

XPath to język zapytań, który umożliwia nawigację i lokalizację elementów w strukturze XML lub HTML. W skrypcie użyto XPath do wyszukiwania elementów na stronie, takich jak pola tekstowe, przyciski, menu rozwijane itp. Cel użycia XPath polega na precyzyjnym i elastycznym znajdowaniu elementów na stronie w celu wykonania na nich operacji.

### 1.2.5. Pakiety biblioteczne:

Skrypt wykorzystuje różne pakiety biblioteczne dostępne w Pythonie, takie jak os, time, selenium.webdriver, selenium.webdriver.common.by, selenium.webdriver.support.ui itp. Te pakiety biblioteczne oferują różne funkcje pomocnicze i narzędzia, które ułatwiają implementację testów automatycznych.

## **1.3. Przygotowanie środowiska**

1. Skrypt wymaga zainstalowanej biblioteki Selenium. Możesz ją zainstalować, wykonując poniższą komendę w terminalu:



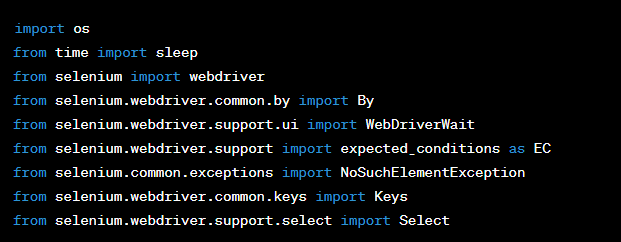
1. Skrypt wykorzystuje sterownik Chrome do obsługi przeglądarki. Upewnij się, że masz zainstalowanego Chrome'a.
2. Pobierz odpowiedni sterownik Chrome dla swojej wersji przeglądarki i systemu operacyjnego. Możesz pobrać sterownik ze strony<https://sites.google.com/a/chromium.org/chromedriver/downloads>. Upewnij się, że pobrany sterownik jest zgodny z wersją Twojej przeglądarki Chrome.
3. Umieść pobrany sterownik Chrome w odpowiednim miejscu w swoim systemie i zaktualizuj ścieżkę do sterownika w skrypcie. W linii kodu:



zastąp 'C:\SeleniumDrivers' odpowiednią ścieżką do folderu zawierającego sterownik Chrome.

## **1.4. Inicjalizacja sterownika**

1. Importuj wymagane moduły:



1. Dodaj ścieżkę do sterownika Chrome do zmiennej środowiskowej 'PATH'. Upewnij się, że ścieżka 'C:\SeleniumDrivers' wskazuje na poprawne miejsce, w którym znajduje się sterownik Chrome.
2. Inicjalizuj sterownik Chrome:



# **2. Dokumentacja techniczna skryptu testującego REST API**

## **2.1. Opis**

Skrypt testujący REST API jest napisany w języku Python i służy do testowania funkcjonalności udostępnianych przez API, które sprawdzić można pod adresem<https://petstore.swagger.io/v2>. Skrypt wykorzystuje bibliotekę *requests* do wykonywania zapytań HTTP i moduł *json* do manipulacji danymi w formacie JSON.

## **2.2. Wykorzystane biblioteki i środowisko**

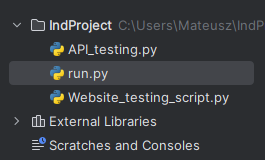
Skrypt korzysta z dwóch bibliotek:

**requests** - biblioteka do wykonywania zapytań HTTP,

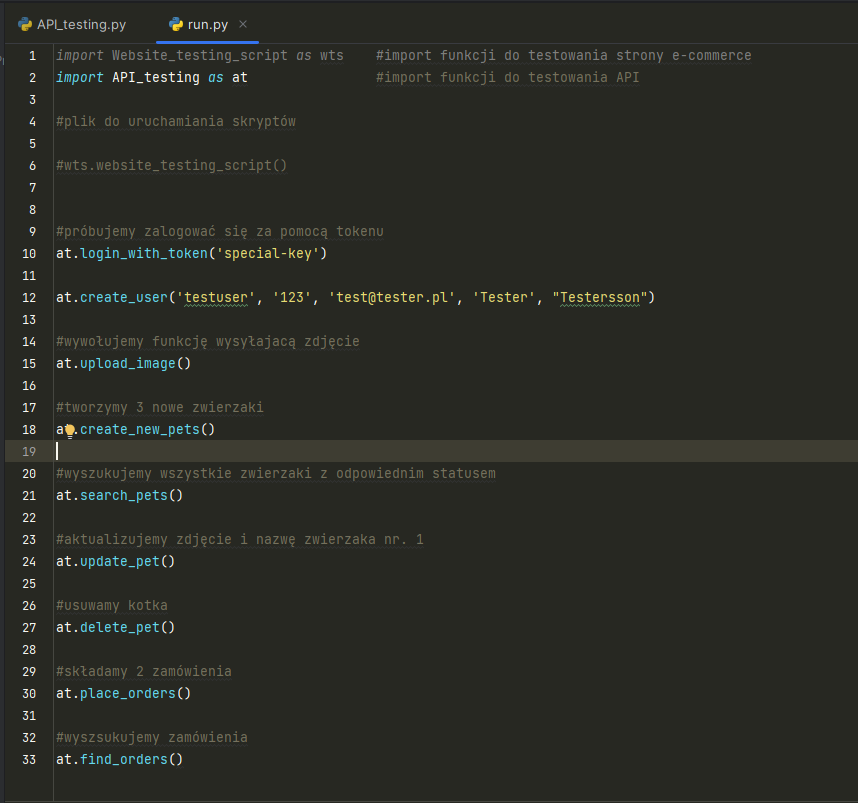
**json** - biblioteka do manipulacji danymi w formacie JSON.

Skrypt można uruchomić w odpowiednim środowisku Python, które ma zainstalowane wymienione biblioteki.

## **2.3. Funkcje skryptu**

Skrypt zawiera kilka funkcji, które testują różne endpointy udostępniane przez API. Wszystkie wymienione poniżej funkcje zawarte są w pliku *API\_testing.py*. 

Skrypt uruchamiany jest wewnątrz pliku *run.py*, gdzie kolejno wywoływane są funkcje znajdujące się w w *API\_testing*. Do skryptów załączone zostały komentarze, ułatwiające czytelność kodu i informujące o tym jaka czynność jest realizowana wywołując daną linijkę kodu. Plik *run.py* wygląda następująco:



### 2.3.1. create\_user(username, password, email, first\_name, last\_name)



Funkcja create\_user służy do tworzenia nowego użytkownika w systemie. Przyjmuje następujące parametry:

username (str) - nazwa użytkownika,

password (str) - hasło użytkownika,

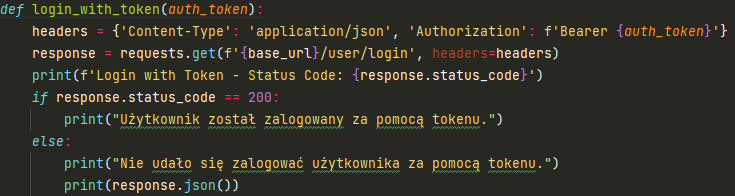
email (str) - adres e-mail użytkownika,

first\_name (str) - imię użytkownika,

last\_name (str) - nazwisko użytkownika.

Funkcja wysyła zapytanie POST na endpoint /user z danymi użytkownika w formacie JSON. Jeśli zapytanie zakończy się sukcesem (kod odpowiedzi 200), wyświetlany jest komunikat "Użytkownik został utworzony" wraz z danymi utworzonego użytkownika. W przypadku niepowodzenia wyświetlany jest komunikat "Nie udało się utworzyć użytkownika" wraz z informacjami o błędzie.

### 2.3.2. login\_with\_token(auth\_token)

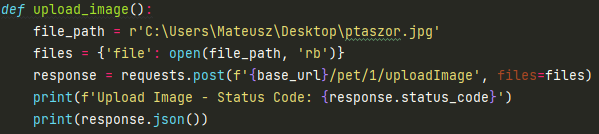


Funkcja login\_with\_token służy do logowania użytkownika przy użyciu tokena uwierzytelniającego. Przyjmuje jeden parametr:

auth\_token (str) - token uwierzytelniający.

Funkcja wysyła zapytanie GET na endpoint /user/login z nagłówkiem Authorization zawierającym token. Jeśli zapytanie zakończy się sukcesem (kod odpowiedzi 200), wyświetlany jest komunikat "Użytkownik został zalogowany za pomocą tokenu". W przypadku niepowodzenia wyświetlany jest komunikat "Nie udało się zalogować użytkownika za pomocą tokenu" wraz z informacjami o błędzie.

### 2.3.3. upload\_image()



Funkcja upload\_image służy do wysyłania obrazu na serwer. Obraz jest określany przez ścieżkę do pliku na lokalnym komputerze. Funkcja nie przyjmuje żadnych parametrów.

Funkcja wysyła zapytanie POST na endpoint /pet/1/uploadImage zawierające przesłany obraz. Wyświetla kod odpowiedzi oraz otrzymane dane w formacie JSON.

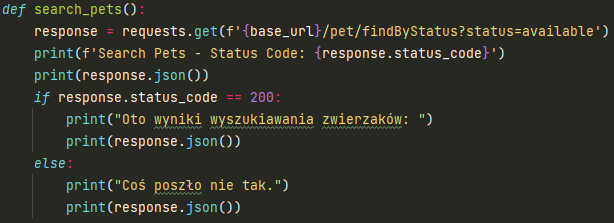
### 2.3.4. create\_new\_pets()



Funkcja create\_new\_pets służy do tworzenia nowych zwierząt w systemie. Funkcja nie przyjmuje żadnych parametrów.

Funkcja create\_new\_pets iteruje po liście danych zwierząt i dla każdego zwierzęcia wysyła zapytanie POST na endpoint /pet z danymi zwierzęcia w formacie JSON. Wyświetla kod odpowiedzi oraz otrzymane dane w formacie JSON.

### 2.3.5. search\_pets()



Funkcja search\_pets służy do wyszukiwania dostępnych zwierząt. Funkcja nie przyjmuje żadnych parametrów.

Funkcja wysyła zapytanie GET na endpoint /pet/findByStatus z parametrem status ustawionym na "available". Wyświetla kod odpowiedzi oraz otrzymane dane w formacie JSON. Jeśli zapytanie zakończy się sukcesem (kod odpowiedzi 200), wyświetlany jest komunikat "Oto wyniki wyszukiwania zwierzaków". W przypadku niepowodzenia wyświetlany jest komunikat "Coś poszło nie tak" wraz z informacjami o błędzie.

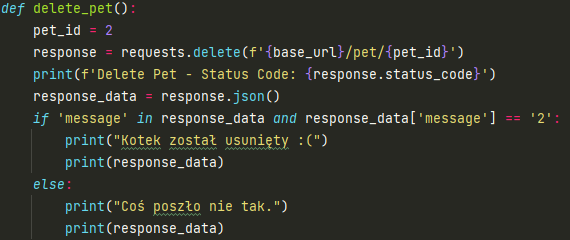
### 2.3.6. update\_pet()



Funkcja update\_pet służy do aktualizowania danych zwierzęcia. Funkcja nie przyjmuje żadnych parametrów.

Funkcja wysyła zapytanie PUT na endpoint /pet z zaktualizowanymi danymi zwierzęcia w formacie JSON. Wyświetla kod odpowiedzi oraz otrzymane dane w formacie JSON. Jeśli zapytanie zakończy się sukcesem (kod odpowiedzi 200), wyświetlany jest komunikat "Prawidłowo zaktualizowano pieseła" wraz z danymi zaktualizowanego zwierzęcia. W przypadku niepowodzenia wyświetlany jest komunikat "Coś poszło nie tak" wraz z informacjami o błędzie.

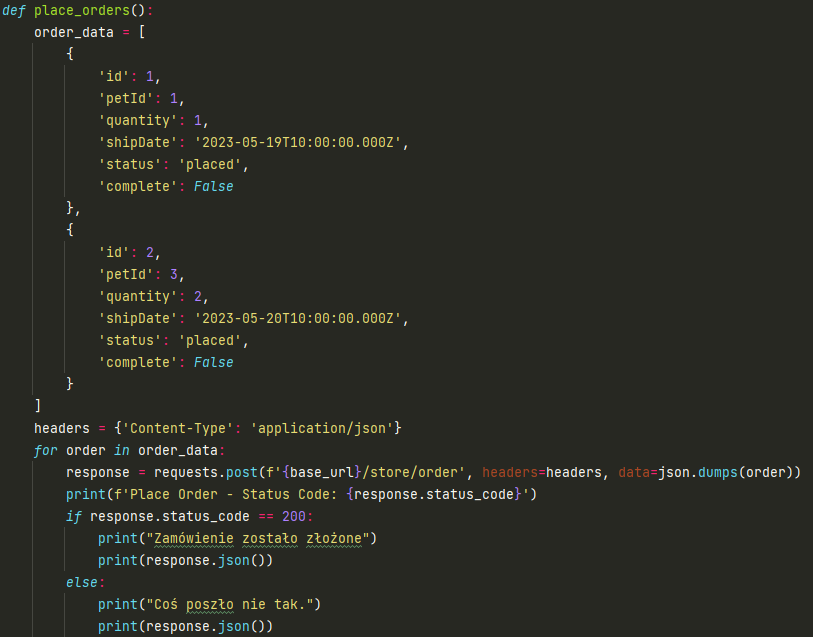
### 2.3.7. delete\_pet()



Funkcja delete\_pet służy do usuwania zwierzęcia. Funkcja nie przyjmuje żadnych parametrów.

Funkcja wysyła zapytanie DELETE na endpoint /pet/{pet\_id} z id zwierzęcia do usunięcia. Wyświetla kod odpowiedzi oraz otrzymane dane w formacie JSON. Jeśli zapytanie zakończy się sukcesem i zwróci odpowiedni komunikat o usunięciu zwierzęcia, wyświetlany jest komunikat "Kotek został usunięty" wraz z danymi zwierzęcia. W przypadku niepowodzenia wyświetlany jest komunikat "Coś poszło nie tak" wraz z informacjami o błędzie.

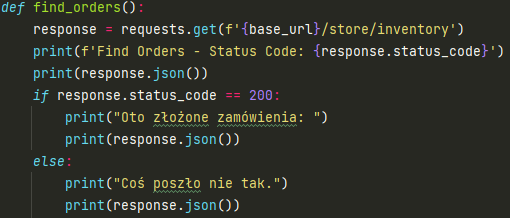
### 2.3.8. place\_orders()



Funkcja place\_orders służy do składania zamówień. Funkcja nie przyjmuje żadnych parametrów.

Funkcja iteruje po liście danych zamówień i dla każdego zamówienia wysyła zapytanie POST na endpoint /store/order z danymi zamówienia w formacie JSON. Wyświetla kod odpowiedzi oraz otrzymane dane w formacie JSON. Jeśli zapytanie zakończy się sukcesem (kod odpowiedzi 200), wyświetlany jest komunikat "Zamówienie zostało złożone" wraz z danymi złożonego zamówienia. W przypadku niepowodzenia wyświetlany jest komunikat "Coś poszło nie tak" wraz z informacjami o błędzie.

### 2.3.9. find\_orders()



Funkcja find\_orders służy do wyszukiwania zamówień. Funkcja nie przyjmuje żadnych parametrów.

Funkcja wysyła zapytanie GET na endpoint /store/inventory. Wyświetla kod odpowiedzi oraz otrzymane dane w formacie JSON. Jeśli zapytanie zakończy się sukcesem (kod odpowiedzi 200), wyświetlany jest komunikat "Oto złożone zamówienia". W przypadku niepowodzenia wyświetlany jest komunikat "Coś poszło nie tak" wraz z informacjami o błędzie.