Wyższa Szkoła Bankowa
Gdańsk Gdynia

# Projekt i wykonanie automatycznych testów funkcjonalnych kluczowych funkcjonalności dowolnej aplikacji za pomocą Selenium WebDriver z wykorzystaniem wzorca Page Object (Python)

**Autorzy: Daria Grabska, Ramona Kwiatek,Dawid Okuniewski,Weronika Press, Adam Trędowicz**

**Promotor: mgr inż. Rafał Borowiec**

Data 18.06.22r

# Prezentacja zespołu i ról

Daria Grabska - dokumentacja i przygotowanie scenariuszy testowych

Ramona Kwiatek - dokumentacja i przygotowanie scenariuszy testowych

Dawid Okuniewski - koordynacja projektu, weryfikacja i ujednolicenie kodu, dokumentacja i przygotowanie scenariuszy testowych

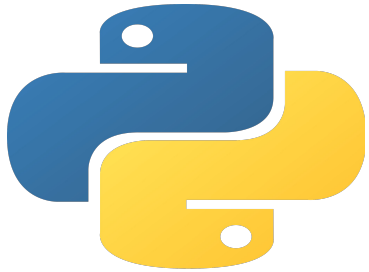
Weronika Press - dokumentacja i przygotowanie scenariuszy testowych, korekta językowa

Adam Trędowicz - dokumentacja i przygotowanie scenariuszy testowych

# Narzędzia

**Gitlab** – serwis, który udostępnia repozytorium w sieci. Używa Git, czyli rozproszonego systemu kontroli wersji.

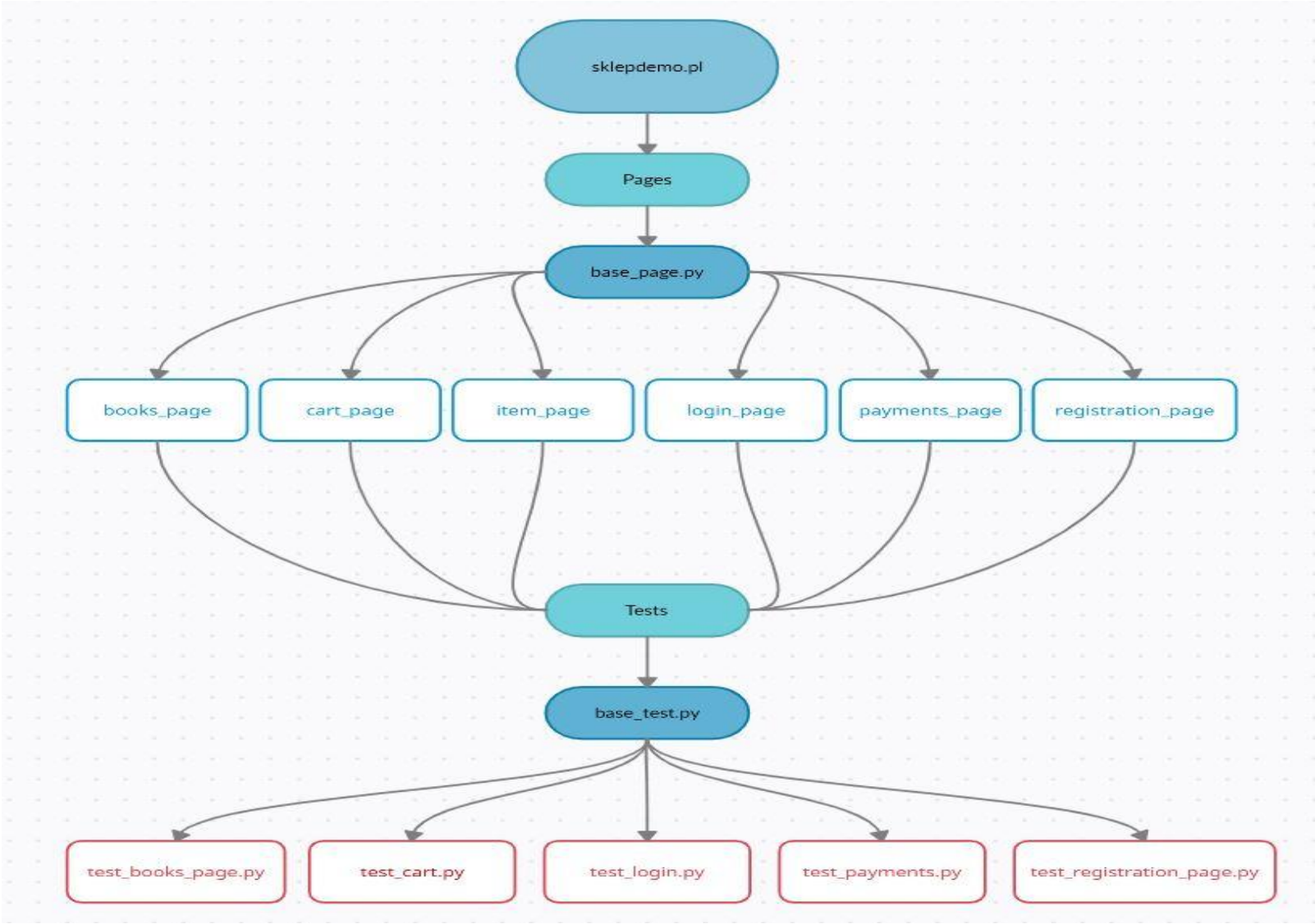
**Python** – język programowania wysokiego poziomu, z silnym i dynamicznym typowaniem. Nie wymaga deklarowania typów zmiennych. Nazwa języka pochodzi od serialu z lat 70 nadawanego przez BBC – „Monthy Python Flying Circus” ; )



**PyCharm** – zintegrowane środowisko programistyczne dla języka Python, które pozwala nam edytować i analizować kod, uruchamiać testy jednostkowe

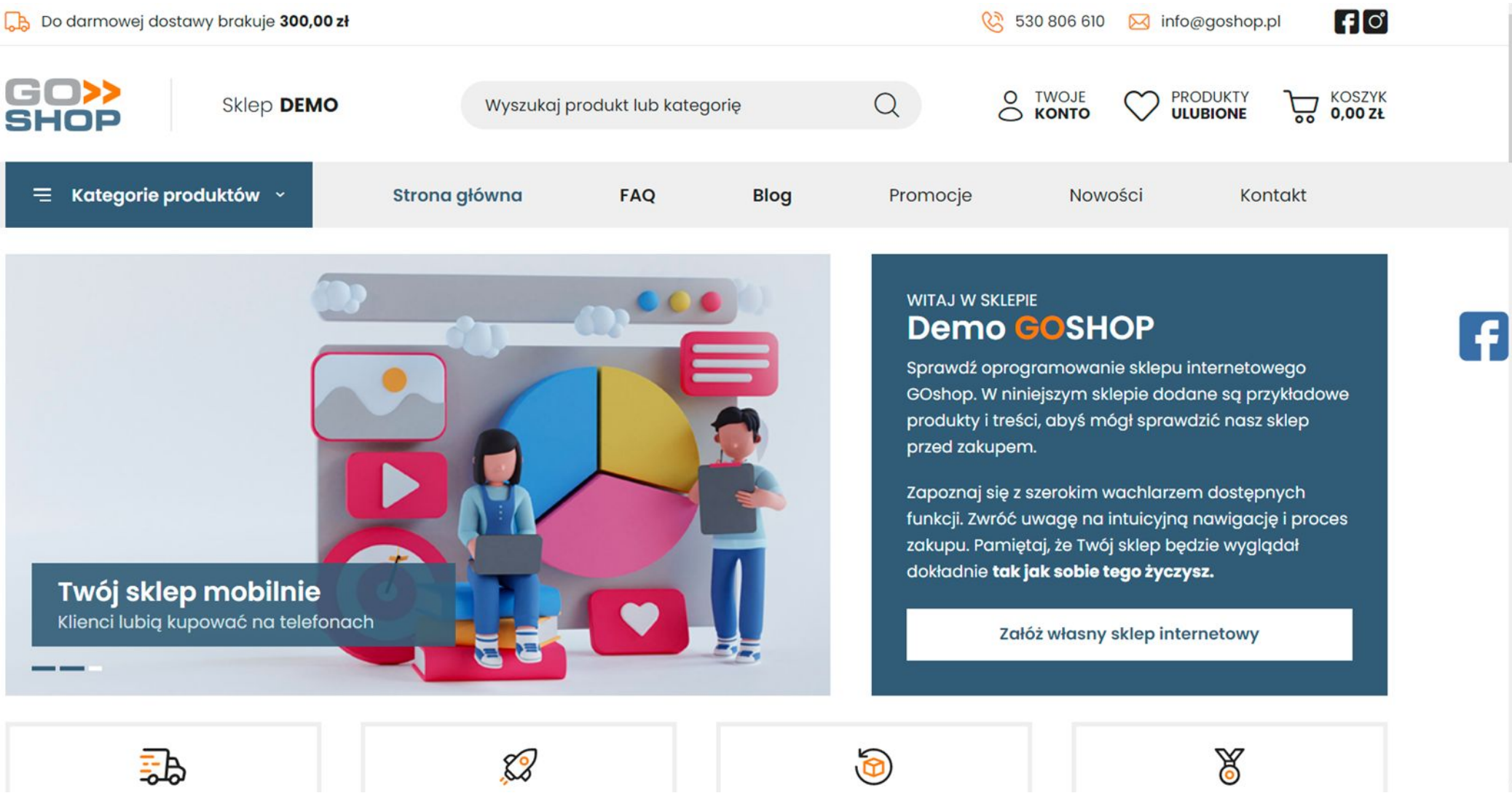

**Selenium WebDriver** - narzędzie do automatyzacji aplikacji webowych

# Page Object Model



## Strona na której pracowaliśmy to www.sklepdemo.pl

## Jest to demo sklepu udostępniane dla potencjalnych klientów i za zgodą właściciela strony przetestowaliśmy podstawowe funkcjonalności



# Struktura projektu

## Początkowy pomysł

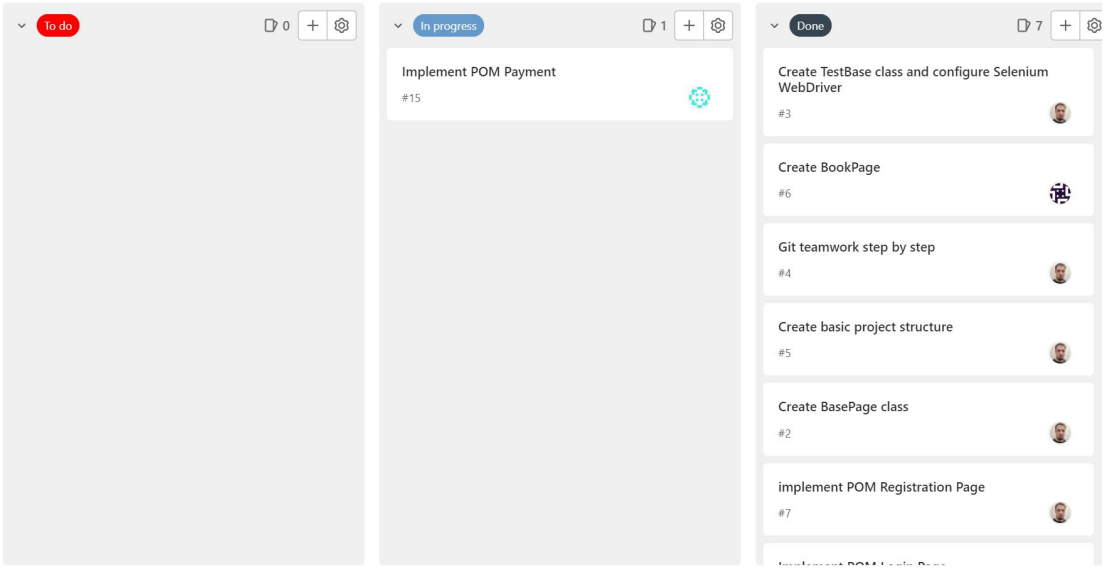

```
C:.
|   .gitignore
|   config.py
|   git_teamwork.md
|   readme.md
|   requirements.txt
|   venv-install.bat
|
+---pages
|       base_page.py
|       books_page.py
|       cart_page.py
|       item_page.py
|       locators.py
|       login_page.py
|       registration_page.py
|       __init__.py
|
\---tests
        base_test.py
        conftest.py
        messages.py
        test_books_page.py
        test_cart.py
        test_login.py
        test_registration_page.py
        __init__.py
```

## Końcowa realizacja

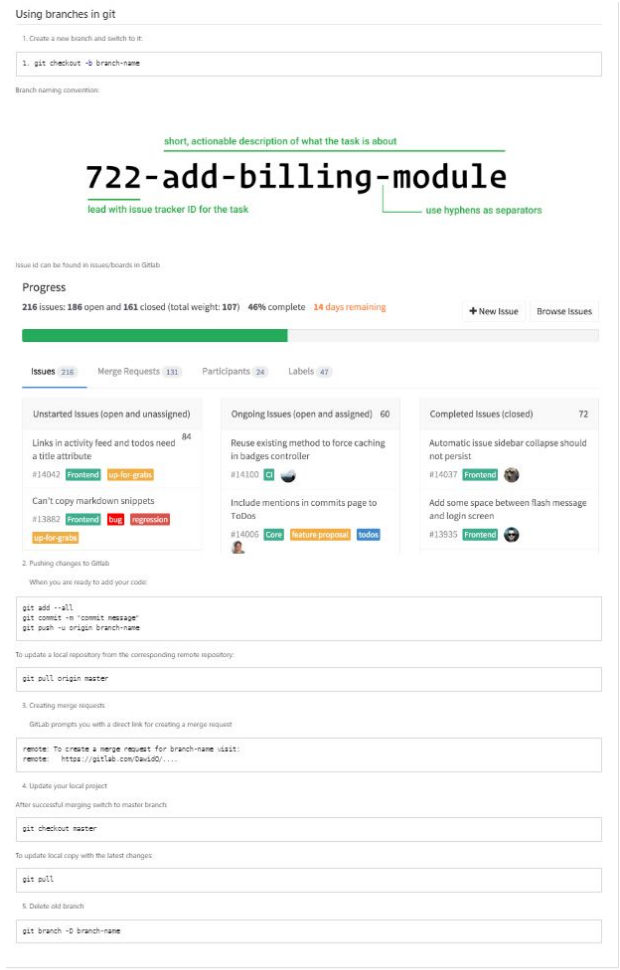
```
|   .gitignore
|   config.py
|   git_teamwork.md
|   helpers.py
|   README.md
|   requirements.txt
|   venv-install.bat
|
+---pages
|   |   base_page.py
|   |   __init__.py
|   |
|   +---books_page
|   |       books_page.py
|   |       selectors.py
|   |       __init__.py
|   |
|   +---cart_page
|   |       cart_page.py
|   |       selectors.py
|   |       __init__.py
|   |
|   +---item_page
|   |       item_page.py
|   |       selectors.py
|   |       __init__.py
|   |
|   +---login_page
|   |       login_page.py
|   |       messages.py
|   |       selectors.py
|   |       __init__.py
|   |
|   +---payments_page
|   |       payments_page.py
|   |       selectors.py
|   |
|   \---registration_page
|           messages.py
|           registration_page.py
|           selectors.py
|           __init__.py
|
\---tests
        base_test.py
        conftest.py
        test_books_page.py
        test_cart.py
        test_login.py
        test_payments.py
        test_registration_page.py
        __init__.py
```

# Usprawnienia

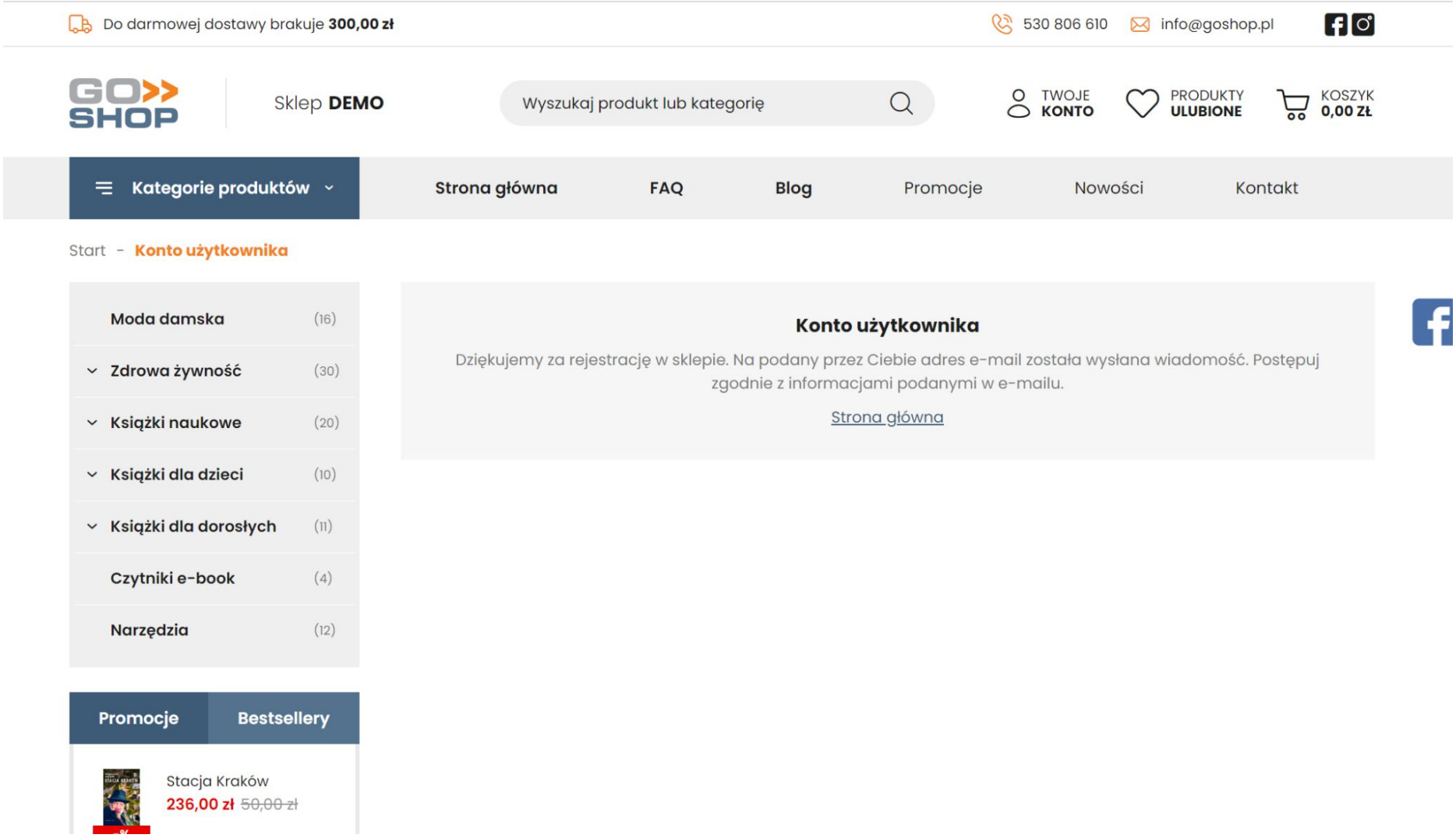
## Tablica kanban




## git teamwork



# Problemy z rejestracją i logowaniem



# Poszukiwanie rozwiązania

## Email Sandbox Service


## testmail.app

```
{namespace}.{tag}@inbox.testmail.app
```


```
{
  "result": "success",
  "message": null,
  "count": 0,
  "limit": 10,
  "offset": 0,
  "emails": []
}
```

# Rozwiązanie

ation_page.py × conftest.py × helpers.py × base_page.py × books_page.py ×

```python
def get_confirmation_link(sender, receiver, subject, tag):
    """
    Querying mailbox for new emails with given subject. Finds and returns confirmation link form email body.

    :param sender: (str) sender email address
    :param receiver: (str) Email address assign to yours account in testmail.app {namespace}.{tag}@inbox.testmail.app
    :param subject: (str) Email subject.
    :param tag: (str) Unique tag from receiver email address.

    :return:
    (str) link
    """
    curl = f"https://api.testmail.app/api/json?apikey={APIKEY}&namespace={NAMESPACE}&pretty=true&livequery=true&tag={tag}"

    r = requests.get(curl, timeout=60)
    emails = r.json()['emails']
    for e in emails:
        if e.get('envelope_to') == receiver and e.get('envelope_from') == sender and e.get('subject') == subject:
            link = find_confirmation_link(e.get('html'))
            return link
```

**testmail.app**

```python
def find_confirmation_link(html_data):
    """
    Finds and returns link from email body.

    :param html_data: (str)
    :return:
        (str) link
    """
    soup = BeautifulSoup(html_data, 'html.parser')
    link = soup.find('a')['href']
    return link
```

# Rozwiązanie

```python
@pytest.fixture(scope='session')
def create_registered_user():
    driver = webdriver.Firefox(service=Service(GeckoDriverManager().install()))
    user = RegistrationPage(driver)
    login_password = 'Test123!'
    login_email, tag = generate_unique_email_and_tag()
    user.register_with(
        email=login_email,
        password=login_password,
        rpassword=login_password,
        newsletter=False,
        terms_and_conditions=True
    )
    user.confirm_email(login_email, tag)
    driver.close()
    return login_email, login_password
```

```python
@pytest.fixture(scope='class')
def setup_test_base(request, create_registered_user, driver_instance):
    driver = driver_instance
    request.cls.login_email, request.cls.login_password = create_registered_user
    request.cls.driver = driver
```

```python
@pytest.mark.usefixtures('setup_test_base')
class TestBase:
```

```python
def test_buy_with_login(self):
    self.item.add_item_to_cart()
    self.payments.buy_with_login(self.login_email, self.login_password)
    assert self.payments.successful_buy() == 'Twoje zamówienie zostało złożone!'
```

# Prezentacja wybranych scenariuszy testowych

Sprawdź, czy po dodaniu maksymalnej ilości sztuk produktu można przycisnąć przycisk „+”

**Oczekiwany rezultat:** brak możliwości przyciśnięcia przycisku „+” bądź komunikat o błędzie

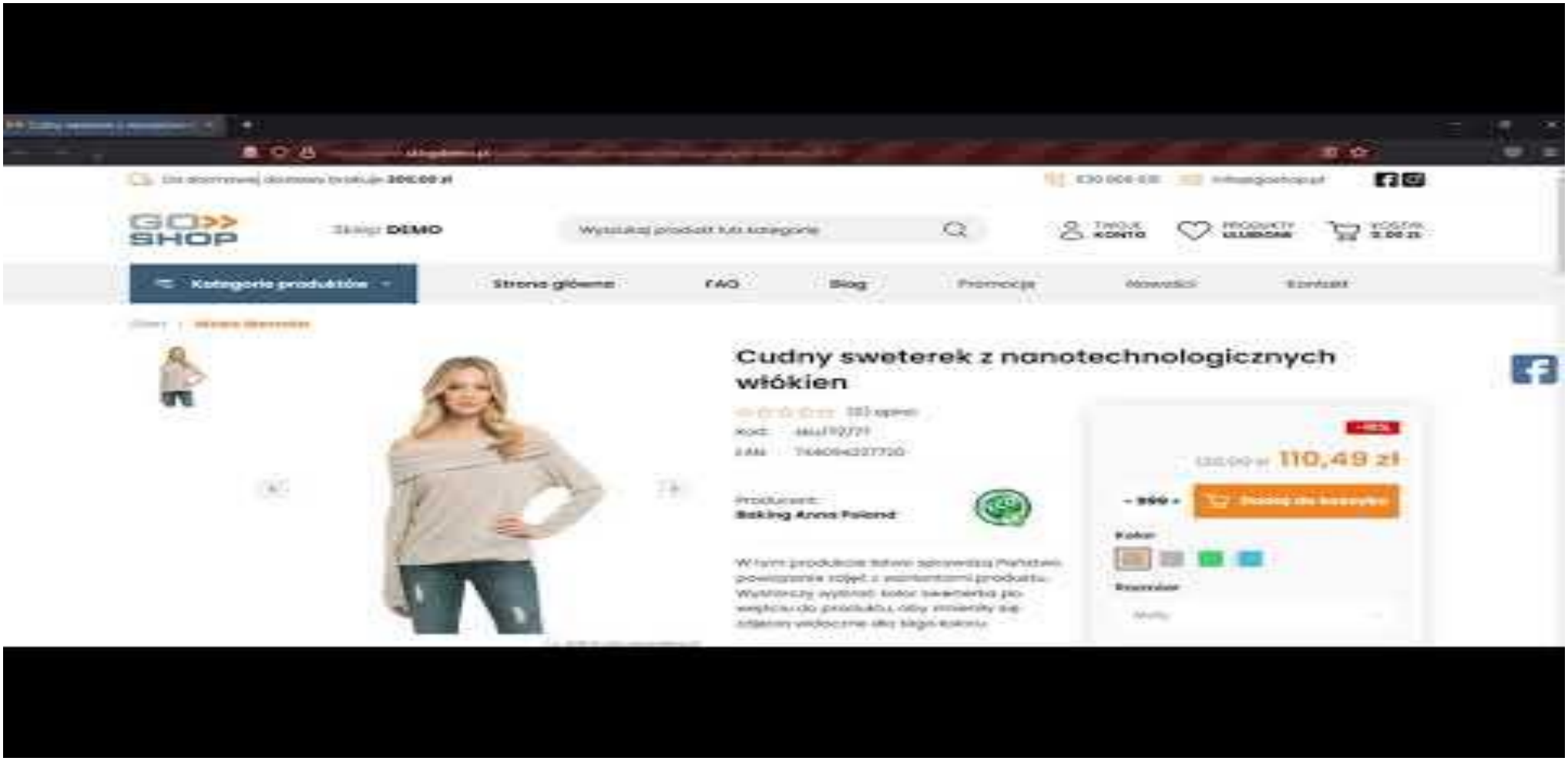
**Rzeczywisty rezultat:** Przycisk „+” można dalej przycisnąć, brak komunikatu o błędzie

```
33      def add_999_items(self):
34          self.clear(self._locators['add_few_product_btn'])
35          self.fill(self._locators['add_few_product_btn'], '999')
36          self.click_btn(self._locators['add_to_cart_btn'])
37
```

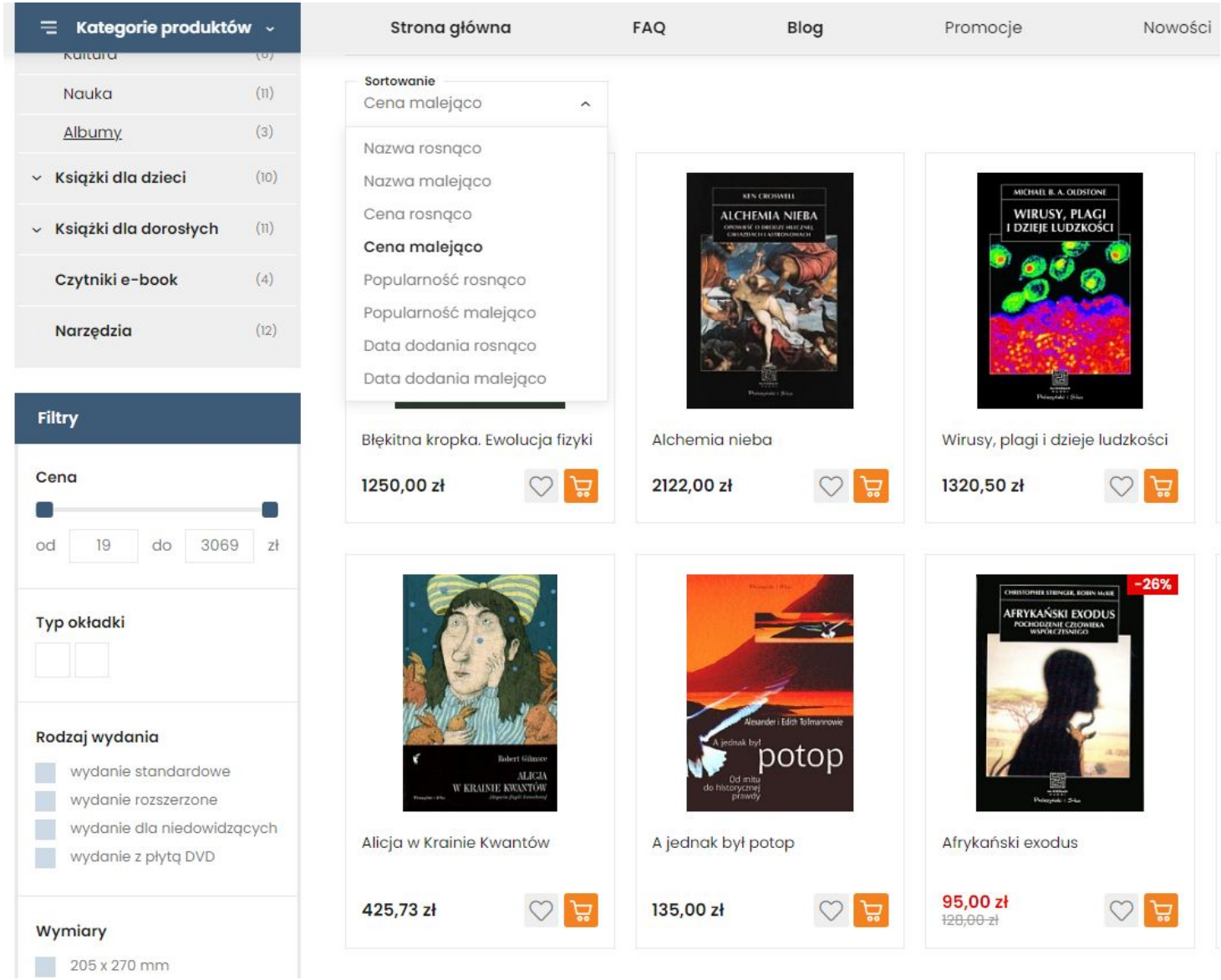
```
28      def check_how_many_items_are_in_cart(self):
29          element = self.find_element(self._locators['cart_value'])
30          items = element.get_attribute('value')
31          return items
```

```
56      def test_add_more_than_is_available(self):
57          self.item.add_999_items()
58          self.item.click_go_to_cart_button()
59          available = self.cart.check_how_many_items_are_in_cart()
60          self.cart.plus_one_item()
61          over = self.cart.check_how_many_items_are_in_cart()
62          assert over == available
63
```

# Demonstracja błędu możliwości dodania do koszyka liczby sztuk produktu ponad dostępną ilość



# Filtrowanie i sortowanie



# Błąd w sortowaniu ceną malejąco

Sprawdź, czy na stronie z książkami naukowymi po wybraniu sortowania według ceny malejąco książki zostaną poprawnie posortowane

**Oczekiwany rezultat:** Książki naukowe zostaną posortowane od najwyższej do najniższej ceny

**Rzeczywisty rezultat:** Książki nie są posortowane według ceny malejąco

*Metody:*

```
def sort_by_price_desc(self):
    self.click_btn(self._locators['sorter_button'])
    self.click_btn(self._locators['price_desc'])
```

```
def get_books_prices(self):
    prices = self.find_elements(self._locators['book_price'])
    price_list = []
    for price in prices:
        discounted = price.find_element(By.TAG_NAME, "b")
        price_list.append(discounted.text)
    new_list = []
    for i in price_list:
        x = i.replace(",", ".")
        new_list.append(float(x))
    return new_list
```

*Test:*

```
def test_sort_name_desc(self):
    self.books.sort_by_name_desc()
    names = self.books.get_books_titles()
    assert names == sorted(names, key=locale.strxfrm, reverse=True)
```

# Demonstracja błędu sortowania malejąco książek naukowvch

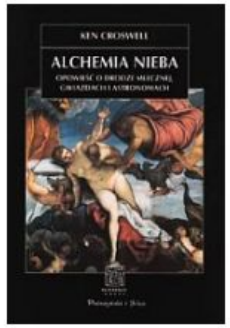
```
E       assert [1250.0,\n 2122.0,\n 1320.5,\n 439.0,\
n 425.73,\n 135.0,\n 95.0,\n 199.99,\n 247.5,\n 236.0
,\n 231.7,\n 200.0] == [2122.0,\n 1320.5,\n 1250.0,\n
 439.0,\n 425.73,\n 247.5,\n 236.0,\n 231.7,\n 200.0,
\n 199.99,\n 135.0,\n 95.0]
E         At index 0 diff: 1250.0 != 2122.0
E         Full diff:
E           [
E         +  1250.0,
E            2122.0,
E            1320.5,
E         -  1250.0,
E            439.0,
E            425.73,
E         +  135.0,
E         +  95.0,
E         +  199.99,
E            247.5,
E            236.0,
E            231.7,
E            200.0,
E         -  199.99,
E         -  135.0,
E         -  95.0,
E           ]

test_books_page.py:38: AssertionError
                                                    Tes
```

Sortowanie

Cena malejąco

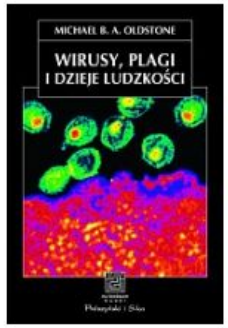

Błękitna kropka. Ewolucja fizyki

1250,00 zł



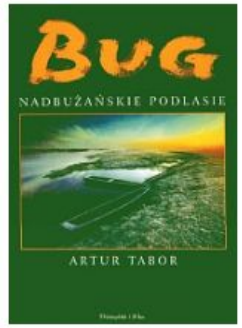

Alchemia nieba

2122,00 zł



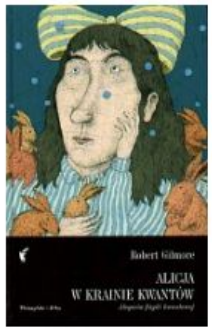

Wirusy, plagi i dzieje ludzkości

1320,50 zł



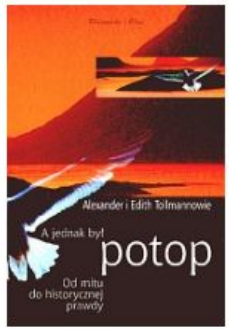

Bug. Nadbużańskie Podlasie

439,00 zł
~~149,00 zł~~



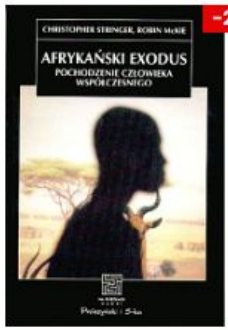

Alicja w Krainie Kwantów

425,73 zł



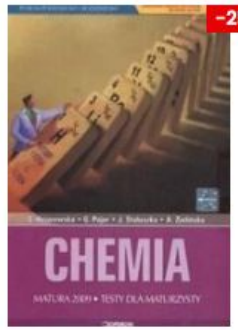

A jednak był potop

135,00 zł




Afrykański exodus

95,00 zł
~~128,00 zł~~




Purpurowa kropka. Ewolucja chemii

199,99 zł
~~267,50 zł~~

# Płatności

GO SHOP

Wyszukaj produkt lub kategorię

TWOJE KONTO | PRODUKTY ULUBIONE | KOSZYK 135,00 ZŁ

Kategorie produktów | Strona główna | FAQ | Blog | Promocje | Nowości | Kontakt

Zakupy bez rejestracji | Mam konto

Adres e-mail *
1111@wp.pl

Numer telefonu *
688888888888888888888888888888888888888888888888

Faktura do zamówienia

Kupuję jako: Osoba prywatna | Firma

Imię *
Bo4444

Nazwisko *
Coo98777

Ulica *
887

Nr domu *
jgh

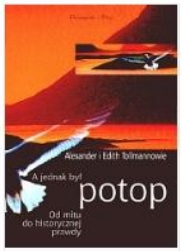
Nr lokalu

Miejscowość *
6666

Kod pocztowy *
00000

Uwagi do zamówienia

## Moje zamówienie




1 szt.

**A jednak był potop**
Nr produktu: sku/75/24
Typ okładki: okładka miękka
Rodzaj wydania: wydanie standardowe

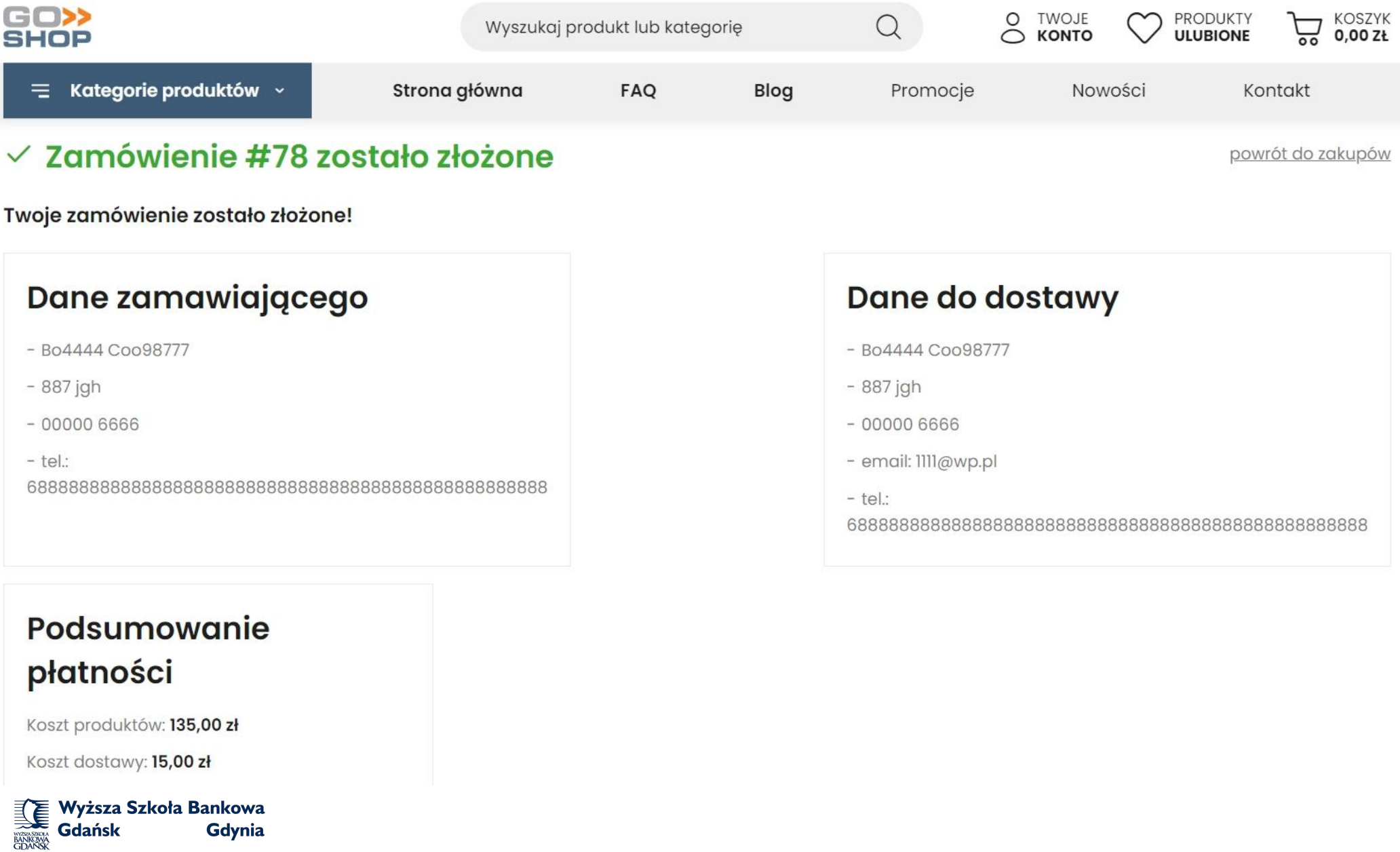
Wartość produktów:
**135,00 zł**

Płatność:
**Wpłata na konto**

Dostawa:
**Poczta Polska (15,00 zł)**

Kwota do zapłaty:
**150,00 zł**

# Płatności


GO SHOP
Wyszukaj produkt lub kategorię
TWOJE KONTO
PRODUKTY ULUBIONE
KOSZYK 0,00 ZŁ
Kategorie produktów
Strona główna
FAQ
Blog
Promocje
Nowości
Kontakt
Zamówienie #78 zostało złożone
powrót do zakupów
Twoje zamówienie zostało złożone!
Dane zamawiającego
- Bo4444 Coo98777
- 887 jgh
- 00000 6666
- tel.: 68888888888888888888888888888888888888888888888
Dane do dostawy
- Bo4444 Coo98777
- 887 jgh
- 00000 6666
- email: 1111@wp.pl
- tel.: 68888888888888888888888888888888888888888888888
Podsumowanie płatności
Koszt produktów: 135,00 zł
Koszt dostawy: 15,00 zł

# Logowanie

Sprawdź, czy zalogujesz się wpisując nieprawidłowy adres email

Oczekiwany rezultat: nie zalogujesz się do systemu wpisując błędny adres email, pojawi się błąd oznakowany znakiem “!”

Rzeczywisty rezultat:nie zalogujesz się do systemu wpisując błędny adres email, pojawi się błąd oznakowany znakiem “!”

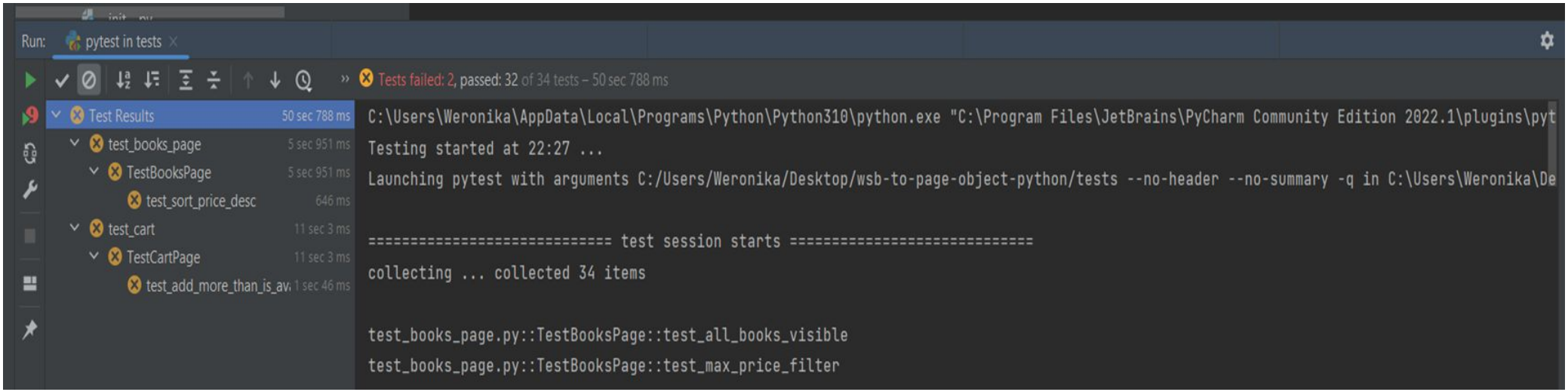
```
def test_login_with_invalid_email(self):
    self.login.login_with(
        email='invalid email',
        password='qwe123'
    )
    assert self._error_msgs['email'] == self.login.get_error_message('email')
    assert self.login.get_error_messages_count() == 1
```


https://www.sklepdemo.pl/login
BUY LIFETIME LICENSE AT WWW.MIRILLIS.COM
GO SHOP
Wyszukaj produkt lub kategorię
TWOJE KONTO
PRODUKTY ULUBIONE
KOSZYK 0,00 ZŁ
Kategorie produktów
Strona główna
FAQ
Blog
Promocje
Nowości
Kontakt
Moda damska (16)
Zdrowa żywność (30)
Książki naukowe (20)
Książki dla dzieci (10)
Książki dla dorosłych (11)
Moje konto
Adres e-mail *
invalid email
Podaj prawidłowy email
Hasło *
Nie masz konta w sklepie?
Zarejestruj się i korzystaj z dostępu do:
Historii swoich zamówień
Szybszych zakupów w sklepie
Możliwości zapamiętywania koszyków

# Demonstracja przebiegu testów

# Raport z testów

Niżej przedstawiamy raport z wykonania testów. 32 testy z 34 przeszły poprawną weryfikację i są w stanie passed oraz 2 z 34 nie przeszły poprawnej weryfikacji i są w stanie failed.



# Dziękujemy za uwagę