API Test
Dokumentacja

Ustawienie Postman

W zakładce „Autoryzacja”:

Key – APPKEY
Value- myszkaApi1234
Add to – Header

Ustawienie środowiska

1. Zmiana długości sesji – w pliku „C:\Users\marty\Desktop\ApiTest\.env” pole „MAX_SESSION_TIME” informuje w minutach ile sesja będzie aktywna bez żadnych akcji. Sesja jest przełądowywana przy wykonaniu każdego zapytania GET i ustawiana ponownie z maksymalnym czasem wygaśnięcia.

Baza danych aplikacji

Link: localhost:8081/index.php?route=/database/structure&db=api_test
Login: root
Hasło: haslohaslo123

Użytkownicy do aplikacji

Login: test
Hasło: test

1. Logowanie

(POST) localhost/api/api.php

Dane json:

```
{
  "c": "login",
  "f": "loginToPanel",
  "login": "test",
  "password": "test"
}
```

„c” – typ api do którego wysyłamy zapytanie*
„f” – funkcja api do wykonania*
„login” – login użytkownika*
„password” – hasło użytkownika bez hasha jako zwykły tekst*

W przypadku poprawnego logowania system zwraca

```
{
  "login":true
}
```

Błędy:

- Inny sposób requesta PUT, DELETE błąd - „Błędny request method”
- Brak parametru „c” lub „f” - „Brak wymaganych danych”
- Brak login lub hasło - „Brak danych do logowania”
- W przypadku nie znalezienia użytkownika - „Błędne dane do logowania”

2. Sprawdzenie czy użytkownik jest zalogowany

(GET) localhost/api/api.php?c=Panel&f=checkPage

Dane get:
„c” – typ api do którego wysyłamy zapytanie*
„f” – funkcja api do wykonania*

W przypadku poprawnego sprawdzenia system zwraca

```
{
  "pageCheck":true
}
```

Błędy:

- Inny sposób requesta PUT, DELETE błąd - „Błędny requqst method”
- Brak parametru „c” lub „f” - „Brak wymaganych danych”
- Brak sessionId lub appKey – „pageCheck: false”
- Brak logowania przed sprawdzeniem – „pageCheck: false”

3. Pobranie loginu zalogowanego użytkownika *(wymagana autoryzacja)*

(GET) localhost/api/api.php?c=Users&f=getUserLogin

Dane get:
„c” – typ api do którego wysyłamy zapytanie*
„f” – funkcja api do wykonania*

W przypadku poprawnego sprawdzenia system zwraca

```
{
  "login":"test"
}
```

Błędy:

- Inny sposób requesta PUT, DELETE błąd - „Błędny requqst method”
- Brak parametru „c” lub „f” - „Brak wymaganych danych”
- Inne błędy – „login: ‘Brak danych’”

4. Wylogowanie z systemu

(GET) localhost/api/api.php?c=Login&f=logoutFromPanel

Dane get:
„c” – typ api do którego wysyłamy zapytanie*
„f” – funkcja api do wykonania*

W przypadku poprawnego wylogowania system zwraca

```
{
  ”logout”:true
}
```

Błędy:

- Inny sposób requesta PUT, DELETE błąd - „Błędny requqst method”
- Brak parametru „c” lub „f” - „Brak wymaganych danych”
- Brak sessionId lub appKey – „pageCheck: false”
- Wcześniejsze wygaśnięcie sesji– „pageCheck: false”

5. Dodawanie nowego użytkownika

(POST) localhost/api/api.php

Dane json:

```
{
  "c": "NewAccount",
  "f": "createNewAccount",
  "login": "test",
  "password": "test",
}
```

„c” – typ api do którego wysyłamy zapytanie*
„f” – funkcja api do wykonania*
„login” – login użytkownika*
„password” – hasło użytkownika bez hasha jako zwykły tekst*

W przypadku poprawnego wykonania system zwraca

```
{
  ”addNewUser”:true
}
```

Błędy:

- Inny sposób requesta PUT, DELETE błąd - „Błędny requqst method”
- Brak parametru „c” lub „f” - „Brak wymaganych danych”
- Brak login lub hasło - „Brak danych do utworzenia konta”
- Jeżeli już jest taki login użyty - „Konto już istnieje w systemie”

6. Aktualizacja użytkownika *(wymagana autoryzacja)*

(PUT) localhost/api/api.php/{id użytkownika do aktualizacji}

Dane json:

```
{
  "c": "Users",
  "f": "updateUser",
  "login": "test",
  "newPassword": "test",
  "active": 0
}
```

„c” – typ api do którego wysyłamy zapytanie*
„f” – funkcja api do wykonania*
„login” – login użytkownika
„newPassword” – hasło użytkownika bez hasha jako zwykły tekst
„active” – użytkownik aktywny „1”, nie aktywny „0”

Co najmniej jeden parametr musi być uzupełniony

W przypadku poprawnego wykonania system zwraca

```
{
  ”updateUser”:true
}
```

Błędy:

- Inny sposób requesta PUT, DELETE błąd - „Błędny requqst method”
- Brak parametru „c” lub „f” - „Brak wymaganych danych”
- Brak login lub hasło - „Brak danych do utworzenia konta”
- Jeżeli już jest taki login użyty - „Konto już istnieje w systemie”

7. Pobranie wszystkich użytkowników *(wymagana autoryzacja)*

(GET) localhost/api/api.php?c=Users&f=getAllUsers

Dane get:
„c” – typ api do którego wysyłamy zapytanie*
„f” – funkcja api do wykonania*

W przypadku poprawnego sprawdzenia system zwraca


```
    {
      "1":{
            "id":"1",
            "login":"test",
            "password":"123f930c1bd849b96f8576871e738a97988db6b851",
            "visible_password":"test",
            "create_time":"2023-07-04 20:32:03",
            "active":"1"
      }
}
```


Błędy:

- Inny sposób requesta PUT, DELETE błąd - „Błędny requqst method”
- Brak parametru „c” lub „f” - „Brak wymaganych danych”

8. Dodanie nowego Produktu *(wymagana autoryzacja)*

(POST) localhost/api/api.php

Dane json:

```
{
  "c": "Products",
  "f": "addNewProduct",
  "name": "Produkt testowy",
  "description": "Krótki opis produktu",
  "quantity": 2,
  "price": 10
}
```

„c” – typ api do którego wysyłamy zapytanie*
„f” – funkcja api do wykonania*
„name” – nazwa produktu*
„desc” – opis produktu*
„quantity” – ilość produktu na stanie w INT*
„price” – cena produktu w FLOAT*

W przypadku poprawnego wykonania system zwraca

```
{
  ”addProduct”:true
}
```

Błędy:

- Inny sposób requesta PUT, DELETE błąd - „Błędny requqst method”
- Brak parametru „c” lub „f” - „Brak wymaganych danych”
- Brak danych - „Brak danych aby dodać produkt”

9. Aktualizacja produktu *(wymagana autoryzacja)*

(PUT) localhost/api/api.php/{id produktu do aktualizacji}

Dane json:

```
{
  "c": "Products",
  "f": "updateProduct",
  "name": "Produkt testowy",
  "description": "Krótki opis produktu",
  "quantity": 3,
  "price": 11
}
```

„c” – typ api do którego wysyłamy zapytanie*
„f” – funkcja api do wykonania*
„name” – nazwa produktu
„desc” – opis produktu
„quantity” – ilość produktu na stanie w INT
„price” – cena produktu w FLOAT

Można dodać tylko jeden parametr do zmiany pozostałe będą bez zmiany, ale musi być uzupełniony co najmniej jeden inaczej będzie błąd.

W przypadku poprawnego wykonania system zwraca

```
{
  ”updateProduct”:true
}
```

Błędy:

- Inny sposób requesta PUT, DELETE błąd - „Błędny requqst method”
- Brak parametru „c” lub „f” - „Brak wymaganych danych”
- Brak danych - „Brak danych do aktualizacji”

10. Lista wszystkich produktów *(wymagana autoryzacja)*

(GET) localhost/api/api.php?c=Products&f=getProducts&productId=1

Dane get:
„c” – typ api do którego wysyłamy zapytanie*
„f” – funkcja api do wykonania*
„productId” – id produktu do aktualizacji

ProductId uzupełnaimy tylko w przypadku kiedy chcemy konkretny produkt

W przypadku poprawnego wykonania system zwraca


```
{
  "1":{
        "id":"1",
        "name":"Produkt testowy",
        "description":"Kr\u00f3tki opis produktu",
        "quantity":"3",
        "price":"11"
  }
}
```


Błędy:

- Inny sposób requesta PUT, DELETE błąd - „Błędny requqst method”
- Brak parametru „c” lub „f” - „Brak wymaganych danych”

11. Dodaj zamówienie *(wymagana autoryzacja)*

(POST) localhost/api/api.php

Dane json:

```
{
  "c": "Orders",
  "f": "createOrder",
  "productId": "1",
  "userId": 1,
  "quantity": 1
}
```

„c” – typ api do którego wysyłamy zapytanie*
„f” – funkcja api do wykonania*
„productId” – id produktu do aktualizacji*
„userId” – id użytkownika*
„quantity” – ilość produktu na stanie w INT*

Można dodać tylko jeden parametr do zmiany pozostałe będą bez zmiany, ale musi być uzupełniony co najmniej jeden inaczej będzie błąd.

W przypadku poprawnego wykonania system zwraca

```
{
  ”addOrder”:true
}
```

Błędy:

- Inny sposób requesta PUT, DELETE błąd - „Błędny requqst method”
- Brak parametru „c” lub „f” - „Brak wymaganych danych”
- Brak danych - „Brak danych aby dodać zamówienie”
- ilość w zamówieniu więcej niż w magazynie - „Brak wystarczającej ilości produktu”

12. Aktualizacja zamówienia *(wymagana autoryzacja)*

(PUT) localhost/api/api.php/{id zamówienia do aktualizacji}

W przypadku aktualizacji ilości zmienia się też dostępna ilość w produktach.

Dane json:

```
{
  "c": "Orders",
  "f": "updateOrder",
  "productId": 1,
  "userId": 1,
  "quantity": 3,
  "sumPrice": 11
}
```

„c” – typ api do którego wysyłamy zapytanie*
„f” – funkcja api do wykonania*
„productId” – id produktu
„userId” – id użytkownika
„quantity” – ilość produktu na stanie w INT
„sumPrice” – cena produktu w FLOAT

Można dodać tylko jeden parametr do zmiany pozostałe będą bez zmiany, ale musi być uzupełniony co najmniej jeden inaczej będzie błąd.

W przypadku poprawnego wykonania system zwraca

```
{
  ”updateOrder”:true
}
```

Błędy:

- Inny sposób requesta PUT, DELETE błąd - „Błędny requqst method”
- Brak parametru „c” lub „f” - „Brak wymaganych danych”
- Brak danych - „Brak danych do aktualizacji”
- Błędny id zamówienia - „Brak zamowienia o id”
- Zmiana ilości na większą jeżeli jest za mało w produktach błąd - „Brak wystarczajacej ilosci produktow do zmiany”

13. Lista wszystkich zamówień *(wymagana autoryzacja)*

(GET) localhost/api/api.php?c=Orders&f=getAllOrders

Dane get:
„c” – typ api do którego wysyłamy zapytanie*
„f” – funkcja api do wykonania*
„startDate” – data od której pobrać zamówienia -timestamp
„endDate” – data do której pobrać zamówienia -timestamp

Daty nie są wymagane wtedy pobierane są wszystkie zmaówienia

W przypadku poprawnego wykonania system zwraca


```
    {
      "1":{
            "id":"1",
            "product_id":"1",
            "user_id":"1",
            "quantity":"1",
            "sum_price":"11",
            "create_time":"2023-07-05 21:22:21"
      },"2":{
            "id":"2",
            "product_id":"1",
            "user_id":"1",
            "quantity":"2",
            "sum_price":"22",
            "create_time":"2023-07-05 21:25:07"
      }
}
```


Błędy:

- Inny sposób requesta PUT, DELETE błąd - „Błędny requqst method”
- Brak parametru „c” lub „f” - „Brak wymaganych danych”

14. Kasowanie produktu *(wymagana autoryzacja)*

(DELETE) localhost/api/api.php/{id produktu do skasowania}

Dane json:

```
{
  "c": "Products",
  "f": "deleteProduct"
}
```

„c” – typ api do którego wysyłamy zapytanie*
„f” – funkcja api do wykonania*

W przypadku poprawnego wykonania system zwraca

```
{
  ”deleteProduct”:true
}
```

Błędy:

- Inny sposób requesta PUT, DELETE błąd - „Błędny requqst method”
- Brak parametru „c” lub „f” - „Brak wymaganych danych”
- Brak id w link – „Brak id”

15. Kasowanie zamówienia *(wymagana autoryzacja)*

(DELETE) localhost/api/api.php/{id zamówienia do skasowania}

W przypadku skasowania zmienia się też dostępna ilość w produktach.

Dane json:

```
{
  "c": "Orders",
  "f": "deleteOrder"
}
```

„c” – typ api do którego wysyłamy zapytanie*
„f” – funkcja api do wykonania*

W przypadku poprawnego wykonania system zwraca

```
{
  ”deleteOrder”:true
}
```

Błędy:

- Inny sposób requesta PUT, DELETE błąd - „Błędny requqst method”
- Brak parametru „c” lub „f” - „Brak wymaganych danych”
- Brak id w link – „Brak id”

16. Kasowanie użytkownika *(wymagana autoryzacja)*

(DELETE) localhost/api/api.php/{id użytkownika do skasowania}

Dane json:

```
{
  "c": "Users",
  "f": "deleteUser"
}
```

„c” – typ api do którego wysyłamy zapytanie*
„f” – funkcja api do wykonania*

W przypadku poprawnego wykonania system zwraca

```
{
  ”deleteUser”:true
}
```

Błędy:

- Inny sposób requesta PUT, DELETE błąd - „Błędny requqst method”
- Brak parametru „c” lub „f” - „Brak wymaganych danych”
- Brak id w link – „Brak id”

17. Dodaj klucz API dla użytkownika *(wymagana autoryzacja)*

(POST) localhost/api/api.php

Dane json:

```
{
  "c": "Panel",
  "f": "createApiKeyForUser",
  "userId": 1
}
```

„c” – typ api do którego wysyłamy zapytanie*
„f” – funkcja api do wykonania*
„userId” – id użytkownika*

W przypadku poprawnego wykonania system zwraca

```
{
  ”apiKey”:xxxxxxxxxxxxxxxxx
}
```

Błędy:

- Inny sposób requesta PUT, DELETE błąd - „Błędny requqst method”
- Brak parametru „c” lub „f” - „Brak wymaganych danych”
- Błędny id - „Uzytkownik o wskazanym id nie istnieje”
- Dodanie ponownie api key dla użytkownika - „Uzytkownik ma juz api key”