

# ODAS 2023Z - Projekt

*Karol Borecki*

# Konfiguracja Docker

```
docker-compose.yml
You, 4 hours ago | 1 author (You)
1 version: '3.8'
2
3 services:
4   flask:
5     build: .
6     command: python app_runner.py run -h 0.0.0.0
7     volumes:
8       - ./flask
9     expose:
10      - 3000
11     env_file:
12      - ./env.dev
13
14   nginx:
15     build: ./nginx
16     ports:
17       - 443:443
18       - 80:80
19     depends_on:
20       - flask
```

```
Dockerfile
You, 4 hours ago | 1 author (You)
1 FROM python:3.11.3-slim-buster
2
3 WORKDIR /flask
4
5 COPY . .
6
7 EXPOSE 3000
8
9 ENV PYTHONDONTWRITEBYTECODE 1
10 ENV PYTHONUNBUFFERED 1
11
12 RUN pip install --upgrade pip
13 RUN pip install -r requirements.txt
14
15 RUN touch ./db/database.db
16
17 CMD ["python3", "app_runner.py"]
```

```
nginx Dockerfile
You, 4 hours ago | 1 author (You)
1 FROM byjg/nginx-extras
2
3 RUN rm /etc/nginx/conf.d/default.conf
4 COPY nginx.conf /etc/nginx/conf.d
5
6 COPY cert.pem /etc/nginx/cert.pem
7 COPY key.pem /etc/nginx/key.pem
```

```
.env.dev
1 AES_SECRET_KEY="238r70rw38vm087r3hhgr3hou3rngou34v08"
2 SECRET_KEY="nfguh34g8yh13480ht0834nvi8u134nv8u134n8g"
3 DATABASE="./db/database.db"
4 CLEAR_DB=True
5 MAIL_SERVER="smtp-mail.outlook.com"
6 MAIL_PORT=587
7 MAIL_USERNAME="b...@barack@...com"
8 MAIL_PASSWORD="..."
9 DNS_NAME="localhost"
```

- Instalacja środowiska z requirements.txt
- Konfiguracja zmiennych środowiskowych w ukrytym pliku .env.dev (W celu trzymania kodu na repozytorium)

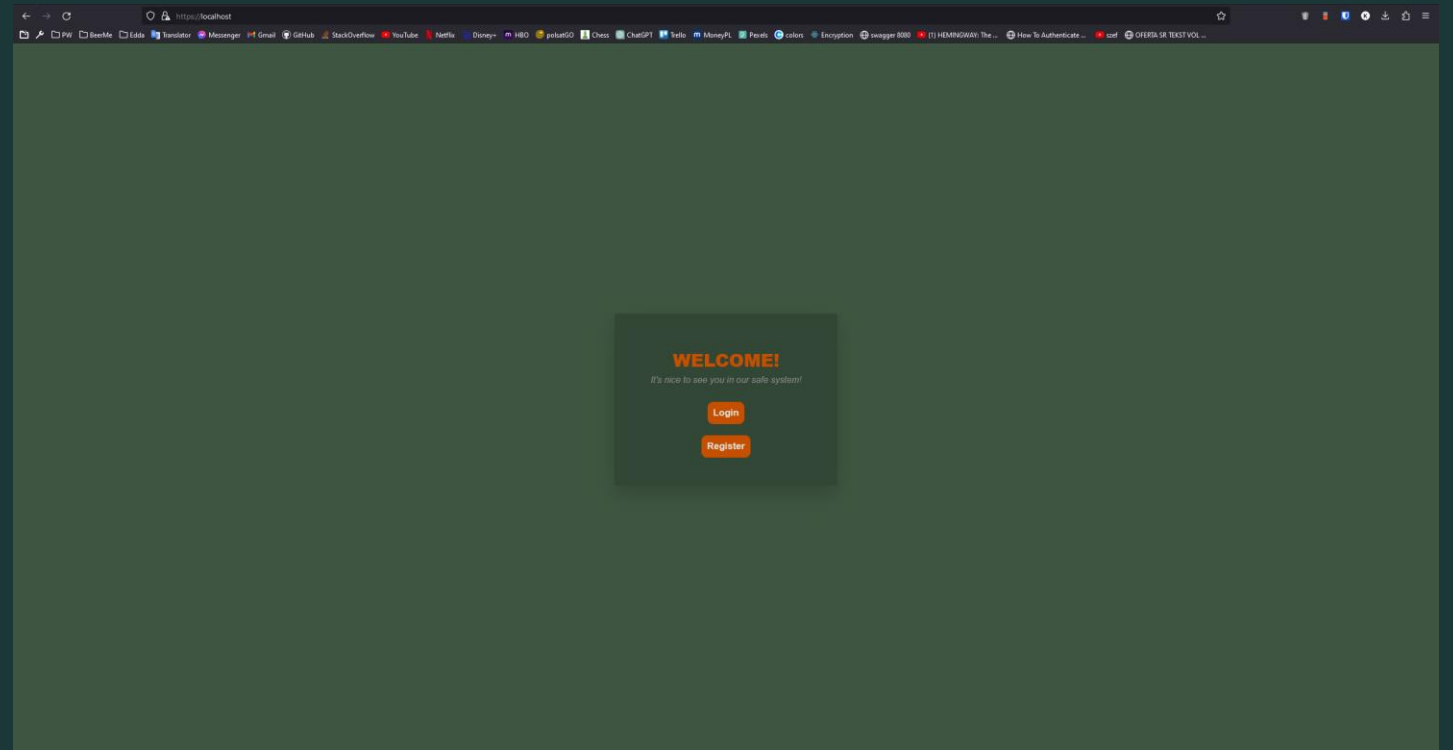
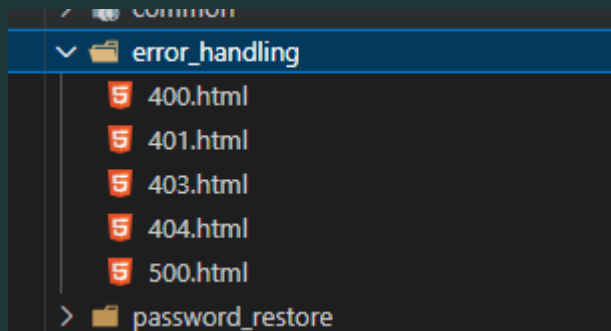
# Konfiguracja NGINX

```
You, 4 hours ago | 1 author (You)
1  upstream flask {
2      server flask:3000;
3  }
4
5  server {
6      listen 80;
7      server_name _;
8      return 301 https://$host$request_uri;
9      server_tokens off;
10     proxy_hide_header Server;
11 }
12
13 server {
14     add_header Strict-Transport-Security "max-age=31536000; includeSubDomains preload"; # 1 year
15     add_header Content-Security-Policy "default-src 'self'";
16
17     listen 443 ssl;
18     ssl_certificate cert.pem;
19     ssl_certificate_key key.pem;
20
21     server_tokens off;
22     proxy_hide_header Server;
23
24     location / {
25         proxy_pass http://flask;
26         proxy_redirect off;
27         more_clear_headers 'Server';
28     }
29 }
30 }
```

- Przekierowanie 80 -> 443
- Certyfikaty ssl
- Usunięcie nagłówka **Server**
- Dodanie nagłówka **Strict-Transport-Security-Policy** do ograniczenia ładowania stron przez HTTP
- Dodanie nagłówka **Content-Security-Policy** do ograniczenia ładowania treści z nieznanych źródeł

# Główna strona

- Gdy strona wykryje ciasteczko z żetonem JWT przekierowuje na /main – główną stronę zalogowanych użytkowników
- Najpopularniejsze błędy są obsługiwane, więc nie fallbackujemy do dziwnych stron



# Formularz rejestracji

- Sprawdzanie siły hasła na podstawie entropii – kod w JS, na żywo
- Walidacja danych i ich sanityzacja
- Unikalna nazwa użytkownika i email
- Dodatkowo generowany numer klienta
- Numer dokumentu szyfrowany AES CBC, ale zapisywana również wykropkowana wersja
- Hasło mieszane SHA256, 80000 rund z randomową solą i pieprzem
- Dane rozproszone poprzez wiele tabel – wprowadzanie lekkiego chaosu do systemu

```
20 def hash_password(self, password):  
21     pepper = self.get_pepper()  
22     password = password + pepper  
23     return (  
24         sha256_crypt.hash(password, rounds=AppSettings().get("auth_hash_rounds")),  
25         pepper,  
26     )  
27
```

User with provided username already exists

## REGISTER

Username

Email address

Password

Repeat password

Document number

Password strength:

Go back

Login

Register

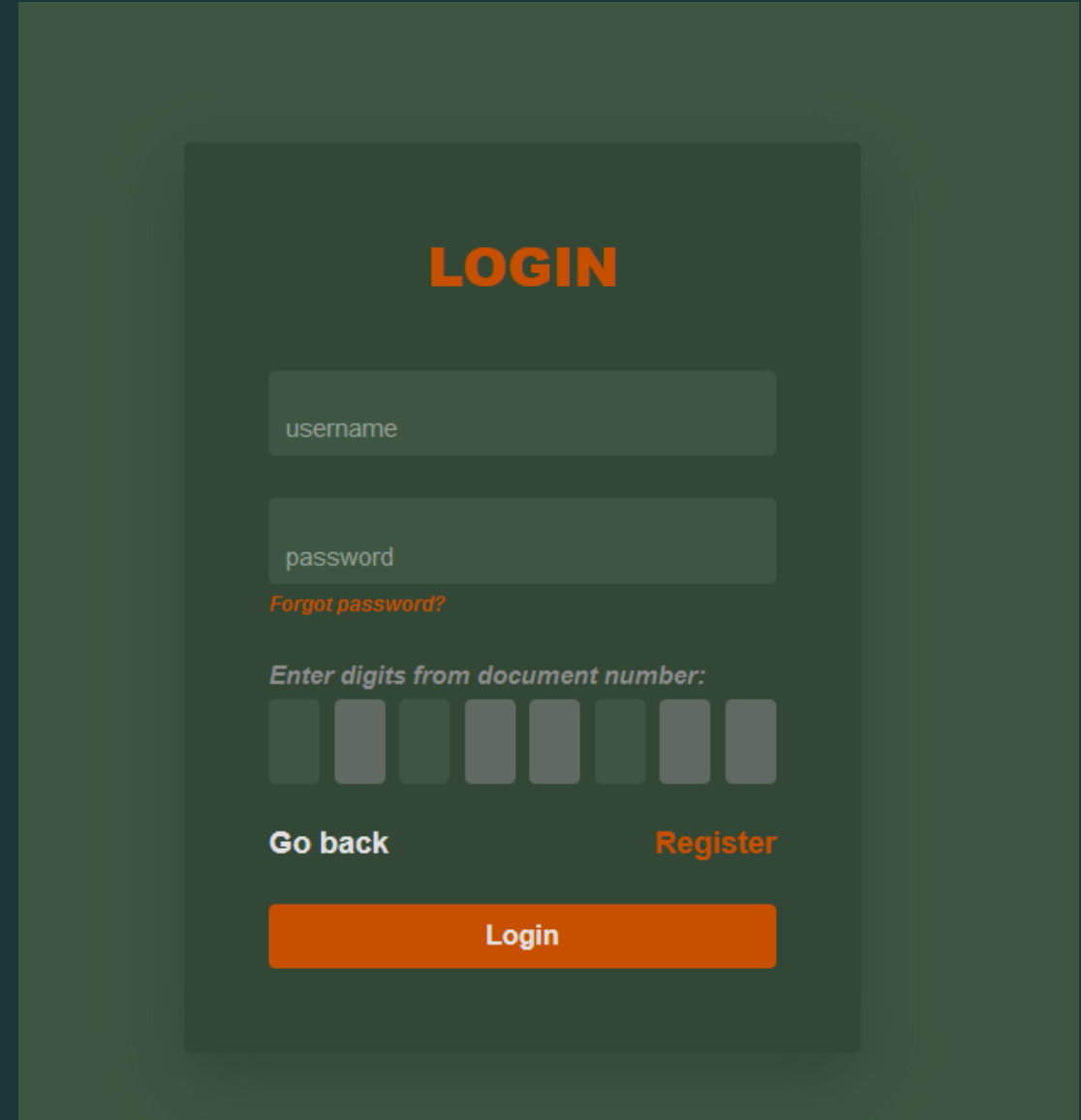
```
86  
87 def create_user(self, email, username, password, document_number):  
88     hash, pepper = HashService().hash_password(password)  
89     hidden_document_number = "*" * (len(document_number) - 3) + document_number[-3:]  
90     aes_document_number = AESService().encrypt(document_number)  
91  
92     client_nr = "".join(  
93         random.choice(string.digits)  
94         for _ in range(AppSettings().get("client_nr_length"))  
95     )  
96  
97     user_id = self.pass_repository.insert({"username": username, "hash": hash})  
98     self.pepper_repository.insert({"user_id": user_id, "pepper": pepper})  
99     self.data_repository.insert(  
100         {  
101             "user_id": user_id,  
102             "email": email,  
103             "document_nr": hidden_document_number,  
104             "client_nr": client_nr,  
105         }  
106     )  
107     self.document_repository.insert(  
108         {"user_id": user_id, "document_nr": aes_document_number}  
109     )  
110     return user_id  
111
```

# Formularz logowania

- Przekierowanie na /main – gdy wykryjemy ciasteczko JWT
- Hasło maskowalne jako randomowe 3 cyfry z numeru dokumentu (za każdym razem inne)
- Odzysk hasła
- Maksymalna liczba nieudanych prób: 3 (blokowanie konta na 1h)
- Po zalogowaniu w ciasteczkach użytkownika zapisywany jest żeton JWT,
  - Wygasa po 15min.
  - W nim również jest adres IP, dla którego został przypisany

Invalid username or password

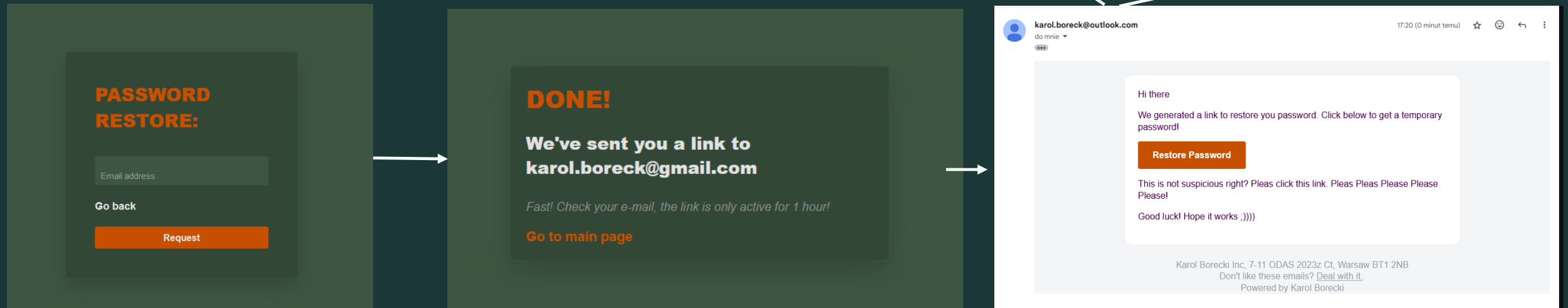
Invalid username or password, tries left: 1



The image shows a login form interface with a dark green background. At the top, the word "LOGIN" is displayed in large, bold, orange letters. Below it, there are two input fields: "username" and "password", both with light green borders. Under the password field, there is a link "Forgot password?" in orange. Below the input fields, there is a section titled "Enter digits from document number:" in light green. This section contains eight input boxes for digits, with the first three being light green and the remaining five being light gray. At the bottom left, there is a link "Go back" in white. At the bottom right, there is a link "Register" in orange. A large orange button labeled "Login" is positioned at the bottom center.

# Formularz odzysku hasła

- Po wpisaniu hasła wysyłamy email z linkiem do odzysku hasła – link jest ważny 10min (na stronie jest błąd że mamy 1h)
- Link to **32 znakowy** endpoint na który wchodzimy i generuje nam się nowe **12 znakowe** hasło, którego możemy użyć do zalogowania się



# Główna strona

- Dalej wszystkie strony są autoryzowane żetonem JWT – jeśli go brakuje lub jest nieważny to przekierowujemy na /login

```
11
12 ~ def token_required(f):
13     @wraps(f)
14     def decorator(*args, **kwargs):
15         token = None
16         if "Authorization" in request.cookies:
17             token = request.cookies["Authorization"]
18         if not token:
19             return redirect("/login")
20         try:
21             data = JWTService().decode(token)
22             # You, 5 hours ago * ]
23             date = data["exp"]
24             if date < datetime.datetime.utcnow().timestamp():
25                 return redirect("/login")
26             ip_addr = data["ip"]
27             if ip_addr != request.remote_addr:
28                 return redirect("/login")
29             user_id = data["user_id"]
30             username = data["username"]
31             if UserService().does_user_exist(username):
32                 current_user = CurrentUser(user_id, username)
33             else:
34                 return redirect("/login")
35             except Exception as e:
36                 return redirect("/login")
37         else:
38             return redirect("/login")
39         return f(current_user, *args, **kwargs)
40
41
42
43
44 return decorator
45
```

**HELLO, KAROL!**

*Account*

*Transactions*



# Informacje o koncie

- Wyświetlone wszystkie informacje poza numerem dokumentu – aby go odblokować musimy wpisać hasło

**HELLO, KAROL!**

---

**Your info:**

Unique id: 1

Username: Karol

E-mail: karol.boreck@gmail.com

Client nr: 24558

Document nr: \*\*\*\*\*678 [See full](#)

---

**Actions:**

*Edit account*

*See logging history*

*Logout*

**Go back**

# Pobieranie numeru dokumentu

- Po wpisaniu hasła odszyfrowujemy numer dokumentu i pokazujemy go użytkownikowi
- Sanityzacja, walidacja
- Żeton CSRF

```
23     return render_template("account/account_info.html", user=current_user)
24     if request.method == "POST":
25         form = NumberRequestForm(request.form)
26         password = form.password.data
27         if UserService().verify_password(current_user.user_id, password):
28             document_number = UserService().get_user_document(current_user.user_id)
29             print(document_number)
30             current_user.document_number = document_number
31             current_user.is_document_number_visible = True
32             return render_template(
33                 "account/account_info.html",
34                 user=current_user,
35                 msg="Document number request successfully",
36                 msg_type="success",
37             )
38         else:
39             return render_template(
40                 "account/account_info.html",
41                 user=current_user,
42                 msg="Invalid credentials",
43                 msg_type="error",
44             )
45     else:
46         return render_template("account/account_info.html", user=current_user)
```

```
17     <div class="form-content">
18         <h2 class="caption">Request the document:</h2>
19         <form action={{ url_for('account.account_info') }} method="post" class="form">
20             <input type="hidden" name="csrf-token" value="{{ csrf_token }}" />
21
22             {% from "common/form_field.html" import form_field %}
23             {{ form_field(form.password) }}
```

Document number request successfully

**HELLO, KAROL!**

### Your info:

Unique id: 1

Username: Karol

E-mail: karol.boreck@gmail.com

Client nr: 24558

Document nr: 12345678

### Actions:

[Edit account](#)

[See logging history](#)

[Logout](#)

[Go back](#)

**REQUEST THE DOCUMENT:**

.....

[Go back](#)

[Request](#)

# Historia logowań

- Użytkownik ma dostęp do informacji jakie IP, kiedy i czy udało się zalogować

## HISTORY:

[GO BACK](#)

IP	Date	Success?
172.20.0.3	2024-01-21 16:29:40	✓
172.20.0.3	2024-01-21 16:29:33	✗
172.20.0.3	2024-01-21 16:25:02	✓
172.20.0.3	2024-01-21 16:20:58	✓
172.20.0.3	2024-01-21 16:09:19	✓
172.19.0.3	2024-01-21 11:54:01	✓

# Pobieranie numeru dokumentu

- Edycja konta na bazie podanego hasła I żetony JWT
- Zaimplementowany żeton CSRF
- Ponownie mierzona siła hasła
- Sanityzacja, walidacja

```
<!DOCTYPE html>
<html class=" qggzfkzca idc0_350" lang="en">
  <script type="text/javascript"> </script>
  <head> </head>
  <body>
    <div class="form">
      <div class="form-container">
        <div class="form-content">
          <h2 class="caption">Edit account:</h2>
          <form class="form-form" action="/account/edit" method="post">
            <input type="hidden" name="csrf-token" value="4RwAmxbenEYdjoTTOORu1X3T3sVzwSAw">
            <div class="input-box">
```

```
74
75 elif request.method == "POST":
76     got_csrf = request.form['csrf-token']
77     if got_csrf != UserService().get_csrf(current_user.user_id):
78         return unauthorized(None)
79
80     if form.validate():
81         password = form.oldPassword.data
82         if UserService().verify_password(current_user.user_id, password):
83             UserService().update_user(
84                 current_user.user_id, form.email.data, form.username.data, form.newPassword1.data
85             )
86             return render_template(
87                 "account/edit_account.html",
88                 form=form,
89                 csrf=got_csrf,
90                 msg="Account updated successfully",
91                 msg_type="success",
92             )
93             return render_template(
94                 "account/edit_account.html",
95                 form=form,
96                 csrf=got_csrf,
97                 msg="Provided credentials are invalid",
98                 msg_type="error",
99             )
100 else:
101     return render_template("account/edit_account.html", form=form, csrf=got_csrf)
102 else:
103     return unauthorized(None)
104
```

## EDIT ACCOUNT:

Karol

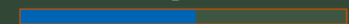
karol.boreck@gmail.com

.....

.....

.....|

Password strength:



Go back

Apply changes

# Wysyłanie transakcji

- Musimy znać numer klienta użytkownika
- Sanityzacja, walidacja
- Żeton CSRF

```
43
44
45 @transactions.route("/transactions/new", methods=["GET", "POST"])
46 @token_required
47 def transactions_new(current_user):
48     form = NewTransactionForm(request.form)
49     if request.method == "GET":
50         csrf = HashService().generate_csrf_token()
51         UserService().set_csrf(current_user.user_id, csrf)
52         return render_template("transaction/new_transaction.html", form=form, csrf=csrf)
53     elif request.method == "POST":
54         got_csrf = request.form['csrf-token']
55         if got_csrf != UserService().get_csrf(current_user.user_id):
56             return unauthorized(None)
57
58         if form.validate():
59             to_id = UserService().get_user_id_by_client_nr(form.to.data)
60             from_id = current_user.user_id
61             if to_id and to_id != from_id:
62                 TransactionService().new(
63                     form.title.data, from_id, to_id, form.value.data
64                 )
65                 return render_template(
66                     "transaction/new_transaction.html",
67                     form=form,
68                     csrf=got_csrf,
69                     msg="Transaction created.",
70                     msg_type="success",
71                 )
72             else:
73                 return render_template(
74                     "transaction/new_transaction.html",
75                     form=form,
76                     csrf=got_csrf,
77                     msg="Invalid client number.",
78                     msg_type="error",
79                 )
80             else:
81                 return render_template(
82                     "transaction/new_transaction.html",
83                     form=form,
84                     csrf=got_csrf,
85                     msg="Provided invalid data.",
86                     msg_type="error",
87                 )
88
89     return unauthorized(None)
90
```

```
43
44
45 @transactions.route("/transactions/new", methods=["GET", "POST"])
46 @token_required
47 def transactions_new(current_user):
48     form = NewTransactionForm(request.form)
49     if request.method == "GET":
50         csrf = HashService().generate_csrf_token()
51         UserService().set_csrf(current_user.user_id, csrf)
52         return render_template("transaction/new_transaction.html", form=form, csrf=csrf)
53     elif request.method == "POST":
54         got_csrf = request.form['csrf-token']
55         if got_csrf != UserService().get_csrf(current_user.user_id):
56             return unauthorized(None)
57
58         if form.validate():
59             to_id = UserService().get_user_id_by_client_nr(form.to.data)
60             from_id = current_user.user_id
61             if to_id and to_id != from_id:
62                 TransactionService().new(
63                     form.title.data, from_id, to_id, form.value.data
64                 )
65                 return render_template(
66                     "transaction/new_transaction.html",
67                     form=form,
68                     csrf=got_csrf,
69                     msg="Transaction created.",
70                     msg_type="success",
71                 )
72             else:
73                 return render_template(
74                     "transaction/new_transaction.html",
75                     form=form,
76                     csrf=got_csrf,
77                     msg="Invalid client number.",
78                     msg_type="error",
79                 )
80             else:
81                 return render_template(
82                     "transaction/new_transaction.html",
83                     form=form,
84                     csrf=got_csrf,
85                     msg="Provided invalid data.",
86                     msg_type="error",
87                 )
88
89     return unauthorized(None)
90
```

Transaction created.

## NEW TRANSACTION:

Transakcja1

12341|

58661

Go back

Apply changes

# Historia transakcji

- Podgląd na wykonane I otrzymane transakcje

Konto obdarowane

TRANSACTIONS:			+NEW	GO BACK
Transakcja1	2024-01-21 16:34:38	12341 PLN → 24558		
Transakcja1	2024-01-21 16:33:57	12341 PLN → 24558		

Konto obdarowujące

TRANSACTIONS:			+NEW	GO BACK
Transakcja1	2024-01-21 16:34:38	12341 PLN → 58661		
Transakcja1	2024-01-21 16:33:57	12341 PLN → 58661		

# Bibliografia

- **[CSRF]** <https://portswigger.net/web-security/csrf>
- **[WTForms]** <https://wtforms.readthedocs.io/en/3.1.x/>
- **[Content-Security-Policy]** <https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP>
- **[Strict-Transport-Security]** <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Strict-Transport-Security>
- **[sqlite3]** <https://docs.python.org/3/library/sqlite3.html>
- **[Flask]** <https://flask.palletsprojects.com/en/3.0.x/>
- **[Login form template]** <https://drive.google.com/drive/folders/1gXrS8EuiHOvquF-nUrrF6UnJ9HPQosMB>
- **[Flask mail]** <https://pythonhosted.org/Flask-Mail/>
- **[HTML symbols]** <https://www.toptal.com/designers/htmlarrows/symbols/>
- **[Passlib]** <https://pypi.org/project/passlib/>
- **[Email template]** <https://github.com/leemunroe/responsive-html-email-template>
- **[Mask passwords]** <https://en.ing.pl/bank-safely>
- **[AES]** [https://en.wikipedia.org/wiki/Block\\_cipher\\_mode\\_of\\_operation](https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation)
- **[NGINX headers]** [http://nginx.org/en/docs/http/nginx\\_http\\_proxy\\_module.html](http://nginx.org/en/docs/http/nginx_http_proxy_module.html)