

Politechnika Świętokrzyska w Kielcach

Podsumowanie Projektu

Kierunek: Informatyka

Rok: 3

Semestr: 5

Skład zespołu: Patryk Grzywacz i Dominik Grudzień

Nazwa przedmiotu: Podstawy Inżynierii Programowania (Projekt)

Temat projektu: Aplikacja bankowa

Cel projektu

Celem projektu było opracowanie i zaimplementowanie aplikacji na temat wybranego systemu informacyjnego. Projekt musiał być zrealizowany za pomocą relacyjnej bazy danych (PostgreSQL) oraz obiektowego języka programowania (Java). Aplikacja musiała posiadać interfejs użytkownika, przetwarzanie danych oraz składowanie danych. Dodatkowo projekt musiał być podzielony na kilka rodzajów użytkowników (Administrator, Pracownik, Klient), której liczba jest większa bądź równa liczbie osób w zespole.

Użyte biblioteki i narzędzia w projekcie

Podczas pracy przy projekcie pomogły nam następujące biblioteki i narzędzia:

Backend (Język programowania Java 16)

Biblioteki:

- Spring Boot 2.5.5
 - Data JPA
 - Spring Boot Mail
 - Spring Security
 - Spring Boot Test
 - Spring Doc (Swagger 2.0)
- Maven 4.0
- JUnit 5
- JSON Web Token
- Vladmihalcea (Hibernate-Types)
- Mockito
- Project Lombok
- PostgreSQL
- Net.kaczmarzyk Specifications
- Eclipse Yasson
- Javax JSON
- Awaitility
- Jasypt Encryptor
- BCryptPasswordEncryptor

Narzędzia:

- Postman
- Heroku
- IntelliJ IDEA 2021.2, Github
- Discord
- Pg Admin

Frontend (Języki programowania: TypeScript 4.4.4, SCSS. Język znaczników: HTML)

Biblioteki:

- React 17.0.2
- Axios
- Bootstrap 5
- React-bootstrap

- React-bootstrap icons
- React-bootstrap-table-next
- React-datepicker
- React-number-format
- React-toastify
- React-router
- Sass
- Yup
- DayJs
- Formik
- Jwt-decode
- Lodash.isequal
- Moment

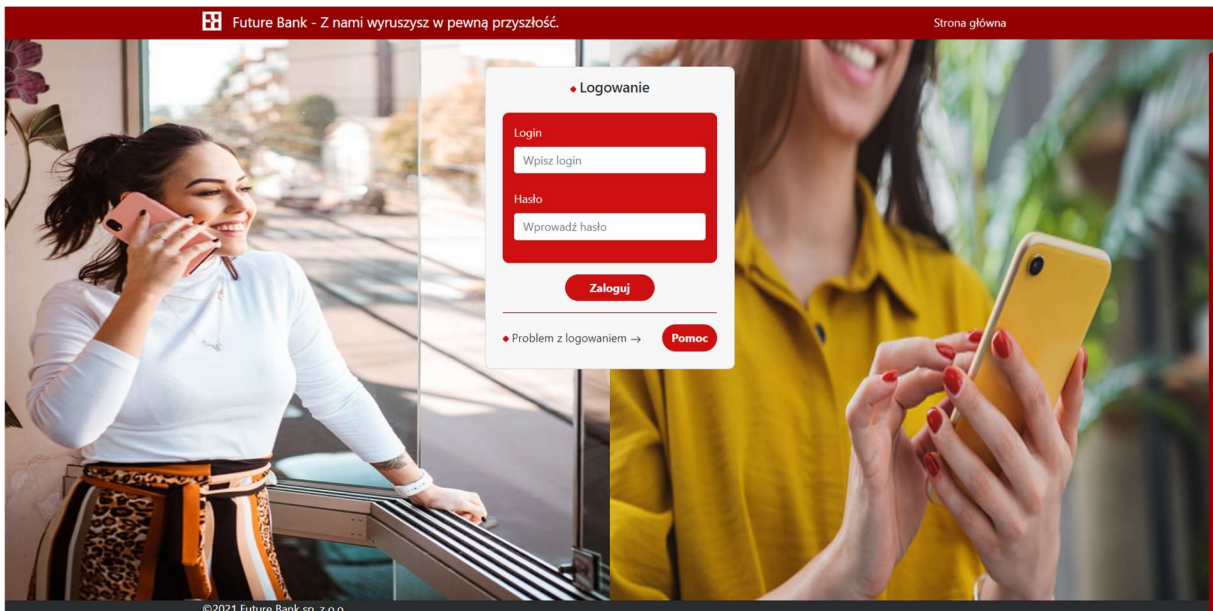
Narzędzia:

- WebStorm 2021.2.1
- Konsola Google Chrome

Sposób wykonania projektu

Frontend

- W interfejsie użytkownika został zachowany duży poziom szczegółowości. Każde detale zostały dokładnie wystylizowane za pomocą naszych selektorów SCSS oraz predefiniowanych klas z biblioteki Bootstrap 5.



Przykład strony logowania do aplikacji

- Staraliśmy się unikać powtórzeń w kodzie za pomocą tworzenia osobnych zbiorów wyrażeń, które wykonują jakieś zadanie oraz stosowaniem typów generycznych.

```
export const useFetchRawData = <T extends unknown>(endpoint: string, params?: any) => {
  const [rawData, setRawData] = useState<T>();
  const [isPending, setIsPending] = useState(false);

  const fetchData = useCallback(async (callbackParams?: any) => {
```

```

setIsPending(true);
try {
  const { data } = await axios.get<T>(endpoint, { params: callbackParams || params });
  setRawData(data);
} catch (e) {
  toast.error("Coś poszło nie tak przy pobieraniu danych: ${e}");
} finally {
  setIsPending(false);
}
}, [setIsPending, setRawData, endpoint, params]);

useEffect(() => {
  fetchData().catch();
}, [fetchData]);

return { rawData, fetchData, isPending };
};

```

Przykład funkcji pobierającej dane z aplikacji backendowej

- Podczas pisania komponentów, które korzystają ze wspólnych stanów stosowaliśmy tzw. konteksty. Dzięki nim w łatwy sposób mogliśmy uzyskać dostęp do danych z różnych poziomów zagnieżdżeń bez konieczności przesyłania ich między komponentami.

```

const Loan = () => {
  return (
    <LoanProvider>
      <Row>
        <Col xs={8}>
          <ClientCardLayout location={LocationHeaders.Loan} style={{ height: '45.2rem' }}>
            <NewLoan/>
          </ClientCardLayout>
        </Col>

        <Col xs={4} className="mt-2" style={{ height: '45.2rem' }}>
          <ActiveLoan/>
        </Col>
      </Row>
    </LoanProvider>
  );
};

```

Przykład inicjacji kontekstu za pomocą dostawcy LoanProvider

```

const NewLoanForm = () => {
  const { currentLoan } = useLoan();
  return (
    <Form className='position-relative mt-4 h-100' noValidate>
      <FormBlob style={{ zIndex: 1 }}/>

      <Row>
        <Col {...colProps}>
          <TextInput
            name='initialRatesNumber'
            label='Liczba rat'
            type='number'
            className='rounded-0 rounded-start float-start'
            labelClassName='fw-bold'
            placeholder='Wpisz liczbę rat'
            disabled={currentLoan.isActive}
          />
        </Col>
      </Row>
    </Form>
  );
};

```

Przykład dostępu do danych z formularza zagnieżdżonego 3 komponenty dalej od inicjacji kontekstu

- Dzięki językowi TypeScript oraz interfejsom mogliśmy w łatwy sposób otypować dane. Usprawniło to pracę oraz czytelność kodu.

```
const { currentUser } = useCurrentUser<ClientModel>();
const foo = currentUser;
return (
  <ClientCardLayout>
    <h1 className='title'>
      <div className='content'>
        <h2>
          <TextWithDialog>
            Konto
          </TextWithDialog>
        </h2>
      </div>
    </ClientCardLayout>
  </div>
);
```

accountNumber?	string
balance?	number
city?	string
email?	string
clientId?	number undefined
dateOfBirth?	string
dateOfCreation?	string undefined
fullName?	string
homeAddress?	string
identificationNumber?	string
numberOfCreditsCards?	number undefined
password?	string

Przykład zastosowania interfejsu ClientModel

- Stosowanie zewnętrznych bibliotek do React'a ułatwiło rozwiązywanie problemów związanych z walidacją formularzy, czy budowaniem tabel za pomocą kolumn wierszy oraz danych

```
import * as Yup from 'yup';

export const TransferValidationSchema = Yup.object().shape({
  amount: Yup.number().required('Podaj kwotę.').
    .min(1, 'Kwota musi być większa niż 0.').
    .max(5000, 'Kwota musi być mniejsza od 5000'),
  transferDate: Yup.string().
    .when('isCyclicalTransfer', {
      is: true,
      then: Yup.string().required('Podaj datę przelewu cyklicznego.').
    }),
  category: Yup.string().required('Podaj kategorię przelewu.').
  receiver_sender: Yup.string().required('Podaj odbiorcę przelewu.').
  title: Yup.string().required('Podaj tytuł przelewu.').
  toAccountNumber: Yup.string().required('Podaj numer konta odbiorcy.').
    .min(26, 'Niepoprawny numer konta odbiorcy'),
});
```

Przykład schematu walidującego (formularz nowego przelewu)

Efekt zastosowania walidacji w formularzu

```
const columns = [
  {
    ...defaultColumnStyle,
    dataField: 'transferDate',
  },
];
```

```

      text: 'Data',
      sort: true,
    },
    {
      ...defaultColumnStyle,
      dataField: 'category',
      text: 'Kategoria',
      classes: getCategoryCellStyle,
    },
    {
      ...defaultColumnStyle,
      dataField: 'receiver_sender',
      text: 'Odbiorca / Nadawca',
    },
    {
      ...defaultColumnStyle,
      dataField: 'displayAmount',
      text: 'Kwota',
      classes: getAmountCellStyle,
    },
  ],
];

const HistoryTable = () => {
  const { transfers } = useHistory();
  const { data, isPending } = transfers;

  const {
    toggleVisibility,
    showModal,
    entity,
  } = useModalState<TransferDisplayModel>();

  const tableProps = useTableProps<TransferDisplayModel>(
    { data, isPending },
    { transferId },
    { initialSortBy: 'transferDate' },
    (e: any, row: TransferDisplayModel) => toggleVisibility(row),
  );

  return (
    <>
      <BootstrapTable
        {...tableProps}
        columns={columns}
      />

      <TransferDetailsModal
        showModal={showModal}
        toggleVisibility={toggleVisibility}
        data={entity || {} as TransferDisplayModel}
      />
    </>
  );
};

```

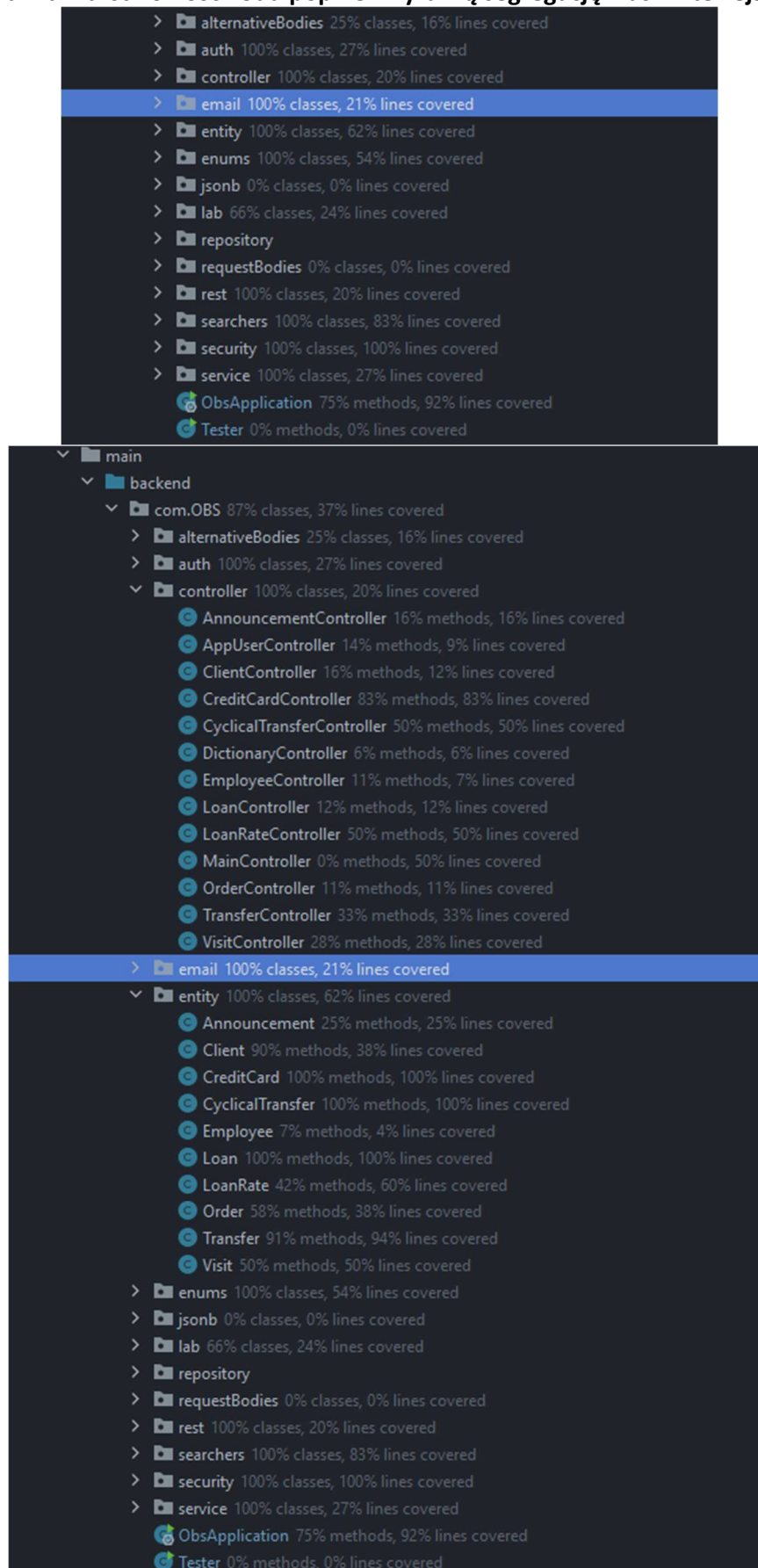
Przykład zaimplementowania tabelki przelewów

Tabela
ostatnich
przelewów

Data ▾	Kategoria	Odbiorca / Nadawca	Kwota
30.12.2021	Wydatki Bieżące	Michał Tester	-1.00 PLN
28.12.2021	Wynagrodzenie	Michał Tester	-3.00 PLN
26.12.2021	Wynagrodzenie	s	-1.00 PLN
26.12.2021	Wynagrodzenie	s	-2.00 PLN
26.12.2021	Wynagrodzenie	Micas	-3.00 PLN
26.12.2021	Wydatki Bieżące	tatst	-4.00 PLN
26.12.2021	Rachunki	s	-1.00 PLN
26.12.2021	Wydatki Bieżące	s	-3.00 PLN
26.12.2021	Rachunki	s	-2.00 PLN
26.12.2021	Wynagrodzenie	s	-12.00 PLN
26.12.2021	Wydatki Bieżące	Mchał Tester	-5.00 PLN
26.12.2021	Wynagrodzenie	s	-3.00 PLN
26.12.2021	Rachunki	asdad	-3.00 PLN
26.12.2021	Rachunki	lhgfd	-1.00 PLN

Backend

- Pisząc naszą aplikację serwerową postaramy się stawiać na jak najlepsze praktyki programowania, bezpieczeństwa i organizacji pracy.
 - Podział na warstwowość kodu poprzez wyraźną segregację klas i interfejsów w podgrupy



- Zastosowanie Waterfall Spring Boot typu : Controller->Service->Repository przyniosło korzyści rozgałęzienia kodu dostępowego do serwera, kodu wykonującego logikę i operacje oraz metod dostępowych do bazy danych . np.:

```
@RestController
@RequestMapping(path = "/dictionary")
@AllArgsConstructor
public class DictionaryController {
    ...

    @GetMapping(path = "/orders")
    public List<Order> getOrders() {
        return orderService.getOrders(ADMIN.name());
    }

    @GetMapping(path = "/orders/for-employees")
    public List<Order> getOrdersForEmployees() { return
orderService.getOrders(EMPLOYEE.name()); }

    ...
}
```

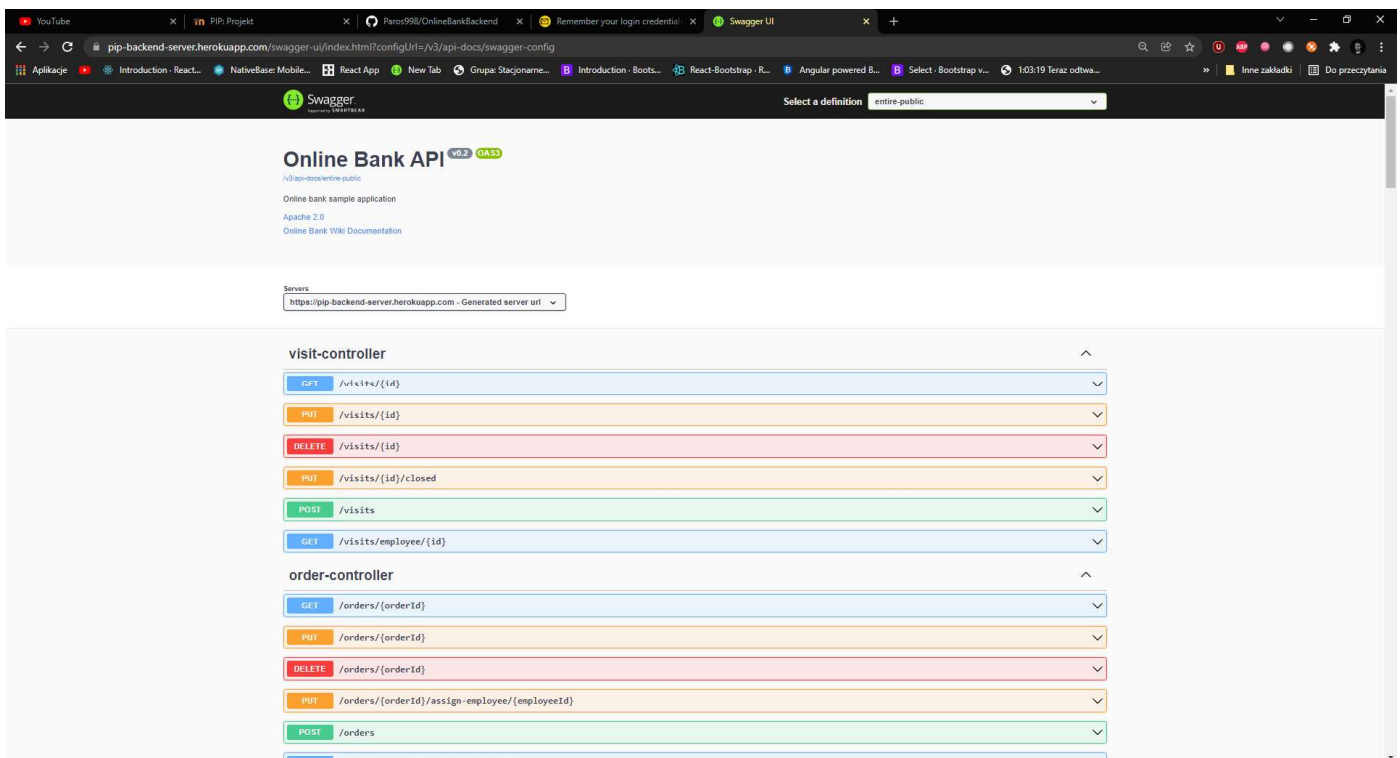
```
@Service
@RequiredArgsConstructor
public class OrderService {
    ...
    public List<Order> getOrders(String role) {
        specification = new BuilderSpecification<Order>()
            .add(new SearchCriteria("createDate", LocalDateTime.now().minusDays(1),
                SearchOperation.GREATER_THAN_DATE))
            .add(new SearchCriteria("employee", null, SearchOperation.EQUAL_NULL));
        if (!Objects.equals(role, ADMIN.name())) {
            ((BuilderSpecification<Order>) specification)
                .add(new SearchCriteria("orderType", changeEmployee.toString(),
                    SearchOperation.NOT_EQUAL))
                .add(new SearchCriteria("orderType", changeUser.toString(),
                    SearchOperation.NOT_EQUAL))
                .add(new SearchCriteria("orderType", createUser.toString(),
                    SearchOperation.NOT_EQUAL));
        }
        return orderRepository.findAll(specification,
            Sort.by(Sort.Direction.ASC, "createDate"));
    }
    ...
}
```

```
@Repository
public interface OrderRepository extends JpaRepository<Order, Long>,
    JpaSpecificationExecutor<Order> {
    ...
}
```


- Zastosowanie JpaSpecificationExecutor w interfejsach dostępnych do bazy danych, pozwala nam na sprawne pisanie kryteriów mniej złożonych zapytań, które następnie mapowane są do zapytań bazodanowych. Np: przypadek wyżej lub przykład z parametrami dynamicznymi (czasami nawet pustymi) z aplikacji frontendowej.

```
@RestController
@AllArgsConstructor
@RequestMapping(path = "/clients")
public class ClientController {
    ...
    @GetMapping(path = "/filtered")
    public List<Client> getClientsSorted(@Conjunction(value = {
        @Or({
            @Spec(path = "fullName", params = "personalNumber_personName",
                spec = StartingWith.class),
            @Spec(path = "personalNumber", params = "personalNumber_personName",
                spec = StartingWith.class)
        }),
        and = @Spec(path = "dateOfBirth",
            params = "birthDate",
            spec = Equal.class)
    }) Specification<Client> filterClientSpec) {
        return clientService.getClients(filterClientSpec);
    }
    ...
}
```

- Włączenie do aplikacji Swagger Open Api, pozwala nam na przyjazne wyświetlanie endpointów, struktur obiektów zwracanych oraz testowanie aplikacji . np.:



- ```
corsConfiguration.setAllowedOrigins(Arrays.asList("http://localhost:3000",
 "http://localhost:5555",
 "http://localhost:19006" ,
 "https://pip-frontend-server.herokuapp.com"
));
```

- 

**Authorization:** Bearer eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJwYXJvczMyMSIsImF1dGhvcm10akVzIjpbeYjhdXRob3RpdHkiOiJST0xFOX0FETU0In1dLCJ1c2VySWQIojEsIm1hdCI6MTY0MzU2NTMzMiwiaXNjaXQzNTg3MjAwFQ..N2u4eVvt68IsfHYj1lnqgE1bcQEwc2ixY4C7X-YJpxr1eD9FkYbwuDW1nqELw9imW5FsL64IeLfJ28dXI5BA

- ```
File Edit View Navigate Code Refactor Build Run Tools Git Window Help
ProjektPIP_Mobile src / main / resources application.properties
ProjektPIP_Mobile src / main / resources application.properties
  LoanRateController.java  OrderController.java  DictionaryController.java  OrderService.java  OrderRepository.java  ClientController.java  pom.xml (OBS)  ObsApplication.java  application.properties
1 spring.datasource.url=jdbc:postgresql://ec2-34-251-245-108.eu-west-1.compute.amazonaws.com/dbtclh67ap2q1
2 spring.datasource.username=ENC(XTGANEWC8NG6RDJ/IUV28ZWeU8bA7ocnfXCLnf8lUlfklfZFPgN+EF83cPyPEXp)
3 spring.datasource.password=ENC(UejbJAamxI88bMoKE4n9sp7591z@WabMN9YVqW9SRKgmxfduYub5IgxQvRqPhZW7K09T072Zr1KXkb7bSp3G1wbn2ftyfZgX3G2bL8/s/08N7DguZWsLo+CxPLbe/VRODa8eLAtuU0fVeRYJ0Jg==)
4 spring.jpa.hibernate.ddl-auto=update
5 spring.jpa.show-sql=true
6 spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect
7 spring.jpa.properties.hibernate.format_sql=true
8 server.error.include-message=always
9 spring.mail.host=smtg.gmail.com
10 spring.mail.port=587
11 spring.mail.username=ENC(7RjyX47mxytPCnD1UzGVEFS0/WvFUl0o5banf2XeuN1xstt0A0XntiA1+/h58T4JehoyRbExnILRYU/09qWKw==)
12 spring.mail.password=ENC(z5tG/BL1SJTF6tHtoH13QZA7LWnkvLQHkfjg1upOWJil4Gf6HbkW8WSVC20b3BE7)
13 spring.mail.properties.mail.smtp.auth=true
14 spring.mail.properties.mail.smtp.ssl.trust=smtg.gmail.com
15 spring.mail.properties.mail.smtp.starttls.enable=true
16
17
```

- Szyfrowanie haseł użytkowników za pomocą BcryptPasswordEncoder kluczami RSA

user_id	app_user_role	email	enabled	locked	password	username
1	ADMIN	patryk198@gmail.com	true	false	\$2a\$04\$bnbV7LYk.UyQ2J3KLC.q3u2ph..	paros321
3	ADMIN	choinka001@wp.pl	true	false	\$2a\$05\$bnbV7LYk.UyQ2J3KLC.q3u2ph..	shandis
4	CLIENT	mail1990@wp.pl	true	false	\$2a\$05\$24PWaHxMwHwa0U0P9L7Nt.wFE..	patrykClient
9	CLIENT	tenzty@op.pl	true	false	\$2a\$05\$ttToLwVf80A9d5d8e9ji.CBW..	janMatejko
10	CLIENT	dominik.grudzien0@gmail.com	true	false	\$2a\$05\$nmooXirodQ3540pyWQ6fd.w1F..	shandis808
11	CLIENT	dominik.grudzien001@gmail.com	true	false	\$2a\$05\$PdcYigK8g9.juIm5VfgeR.b..	shandis01
12	CLIENT	cos@email.com	true	false	\$2a\$05\$wDcYUL0eRVaQ3XEkmbKv.97..	Wron432
8	EMPLOYEE	part4@op.pl	true	false	\$2a\$05\$uXnjRbWfV4I52byweaBUXf..	pg12F
15	CLIENT	cos@hello.com	true	false	\$2a\$05\$JLVDSM3tuUuoia15pxjr.D4J..	AR 121
22	CLIENT	halo@op.pl	true	false	\$2a\$05\$GEdRUC1wPwF004U5gkqp.DrF..	12414bdfgfgd
11	CLIENT	cos2@email.com	true	false	\$2a\$05\$lg2tzL03cptZz5fbcZs0CLr..	pg10
12	CLIENT	domin04@onet.pl	true	false	\$2a\$05\$0mwpV2gfY90LzK9y08YAD0vR..	shandisClient
13	CLIENT	hello@there.com	true	true	\$2a\$05\$scr2snyb0EycAK80C.djPvclh..	jozek
14	CLIENT	sfsad@cos.pl	false	true	\$2a\$05\$g8g8U03UzH9DZCZ1F0e50g..	Rysiek620

- Role użytkowników usprawniają autoryzację dostępową do poszczególnych zasobów

```

.addFilter(new FormLoginUsernameAndPasswordAuthenticationFilter(authenticationManager(), appUserService(), relationshipSearcher()))
.addFilterAfter(new JwtTokenVerifier(), FormLoginUsernameAndPasswordAuthenticationFilter.class)
.authorizeRequests()
.antMatchers("/index", "/css/**", "/js/**", "/swagger-ui.html").permitAll()

.antMatchers(HttpMethod.GET, "/dictionary/orders/for-employees/**").hasAnyRole(ADMIN.name(), EMPLOYEE.name())
.antMatchers(HttpMethod.GET,
"/dictionary/clients/**",
"/dictionary/credit-cards/**",
"/dictionary/orders/employees/**",
"/dictionary/visits/**",
"/dictionary/announcements/**",
"/dictionary/transfers/**",
"/dictionary/cyclical-transfers/**",
"/dictionary/loans/**",
"/dictionary/loans-rates/**").hasAnyRole(ADMIN.name(), EMPLOYEE.name())
.antMatchers(HttpMethod.GET, "/dictionary/**").hasRole(ADMIN.name())

.antMatchers(HttpMethod.POST, "/visits/**").anonymous()
.antMatchers("/visits/**").hasAnyRole(ADMIN.name(), EMPLOYEE.name())

.antMatchers(HttpMethod.PATCH, "/users/**").anonymous()
.antMatchers(HttpMethod.GET, "/users/client/**").hasAnyRole(ADMIN.name(), EMPLOYEE.name(), CLIENT.name())
.antMatchers(HttpMethod.GET, "/users/employee/**").hasAnyRole(ADMIN.name(), EMPLOYEE.name())
.antMatchers("/users/**").hasRole(ADMIN.name())

.antMatchers(HttpMethod.GET, "/employees/id/**").hasAnyRole(ADMIN.name(), EMPLOYEE.name())
.antMatchers("/employees/**").hasAnyRole(ADMIN.name())

.antMatchers(HttpMethod.GET, "/clients/id/**").hasAnyRole(CLIENT.name(), ADMIN.name(), EMPLOYEE.name())
.antMatchers("/clients/**").hasAnyRole(ADMIN.name(), EMPLOYEE.name())

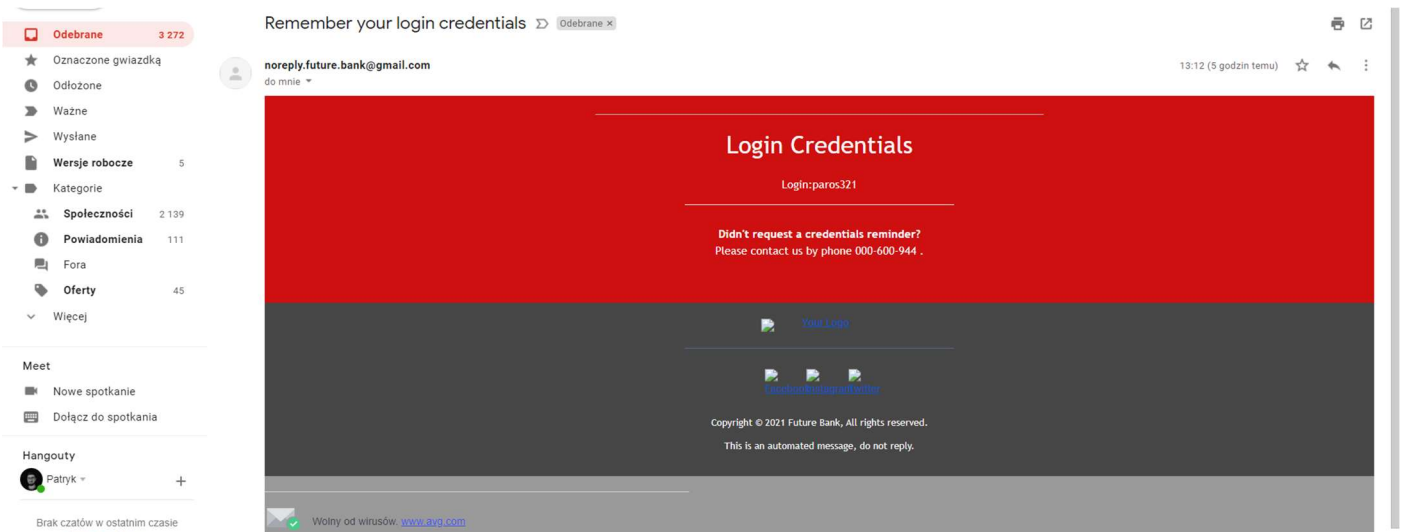
.antMatchers("/announcements/**").hasRole(ADMIN.name())

.antMatchers(HttpMethod.GET, "/credit-cards/client/{clientId}/**").hasAnyRole(CLIENT.name(), ADMIN.name(), EMPLOYEE.name())
.antMatchers("/credit-cards/**").hasAnyRole(ADMIN.name(), EMPLOYEE.name())

.antMatchers(HttpMethod.GET, "/transfers/client/{clientId}/**").hasAnyRole(CLIENT.name(), ADMIN.name(), EMPLOYEE.name())
.antMatchers(HttpMethod.GET, "/transfers/recent/**").hasRole(CLIENT.name())
.antMatchers(HttpMethod.POST, "/transfers/**").hasRole(CLIENT.name())
.antMatchers("/transfers/**").hasAnyRole(ADMIN.name(), EMPLOYEE.name())

```

- Usługa wysyłki maili z informacjami do użytkowników poprzez SMTP Gmail



○ Testy jednostkowe i integracyjne napisane w JUNIT 5

Element	Class, %	Method, %	Line, %
com	87% (88/101)	43% (274/632)	37% (556/1489)

Tests passed: 18 of 18 tests = 12 sec 153 ms

Test results list:

- JUnit Vintage
 - TransferServiceTest
 - performTestTransferBetweenTwoClients: 802 ms
 - LoanServiceTest
 - testPaymentRealizationAndDeletionOf: 5 sec 215 ms
 - getFutureLoanCalculatedAndSendOrder: 3 sec 96 ms
 - OrderControllerTest
 - testAddingOrder: 713 ms
 - testDeletingOrder: 580 ms
 - LoanTest
 - testLoanCreation: 696 ms
 - VisitControllerTest
 - testAddingVisit: 208 ms
 - testDeletingVisit: 408 ms
 - JUnit Jupiter
 - CreditCardControllerTest
 - testAddingCreditCard: 370 ms
 - testSwitchingActiveStateOfCreditCard: 396 ms
 - testGetClientCreditCards: 239 ms
 - testDeletingCreditCard: 397 ms
 - CyclicalTransferControllerTest
 - testAddingCyclicalTransfer: 368 ms
 - testDeletingCyclicalTransfer: 494 ms
 - testGetClientEstimatedTransfer: 862 ms
 - testDeletingCyclicalTransfer: 302 ms
 - TransferControllerTest
 - testPerformingTransfer: 741 ms
 - testDeletingTransfer: 717 ms

○ Relacyjna baza danych PostgreSQL

