

Informatyka E-commerce Developer 2022

Temat: Wewnętrzny system który może być używany w dowolnej firmie do kupowania produktów dla pracowników. Po rozbudowaniu o system płatności może być prostym sklepem internetowym.

Dokonanie analizy przedmiotowej badanie rynku - przykłady dostępnych aplikacji:

Jest to wersja MVP – minimum viable product dowolnego sklepu dostępnego w internecie.

Okrojony sklep internetowy posiadający funkcjonalności CRUD umożliwiający użytkownikowi który posiada uprawnienia administratora na tworzenie bazy produktów, odczytywanie jej, aktualizowanie oraz usuwanie.

Użytkownik w tym systemie będzie mógł zakupić jakiś produkt wcześniej dodany przez administratora. Po kliknięciu na przycisk „kup” ilość produktów w magazynie zmniejszy się o jeden a użytkownik zostanie poproszony o podanie swoich danych do wysyłki. Po kliknięciu przycisku podsumowującego dane. Użytkownik zostanie poinformowany, że zamówienie zostanie zrealizowane w ciągu 48h. Używając takiego systemu użytkownik musiałby zapłacić za określony produkt gotówką lub przelewem.

Wybór metod i narzędzi do wykonania zadania, Opracowanie informacji na temat użytych technologii i narzędzi.

W celu wykonania projektu użyłem xampp'a w celu postawienia lokalnego serwera www.

Potrzebowałem również zainstalować composer, symfony oraz instalator pakietów yarn.

Aby uruchomić aplikację potrzebujemy php w wersji minimum 8.1, composera i symfony.

Aplikacja jest napisana we wzorcu projektowym MVC – model view controller.

Plik composer: wszystkie paczki jakie zainstalowałem w celu zrobienia zadania

```

"require": {
    "php": ">=8.1",
    "ext-ctype": "*",
    "ext-iconv": "*",
    "doctrine/doctrine-bundle": "^2.6",
    "doctrine/doctrine-migrations-bundle": "^3.2",
    "doctrine/orm": "^2.12",
    "easycorp/easyadmin-bundle": "^4.2",
    "knplabs/knp-markdown-bundle": "^1.10",
    "knplabs/knp-time-bundle": "^1.19",
    "sensio/framework-extra-bundle": "^6.2",
    "symfony/console": "6.1.*",
    "symfony/dotenv": "6.1.*",
    "symfony/flex": "^2",
    "symfony/form": "6.1.*",
    "symfony/framework-bundle": "6.1.*",
    "symfony/maker-bundle": "^1.43",
    "symfony/proxy-manager-bridge": "6.1.*",
    "symfony/runtime": "6.1.*",
    "symfony/security-csrf": "6.1.*",
    "symfony/twig-bundle": "6.1.*",
    "symfony/validator": "6.1.*",
    "symfony/webpack-encore-bundle": "^1.14",
    "symfony/yaml": "6.1.*",
    "twig/extra-bundle": "^2.12|^3.0",
    "twig/twig": "^2.12|^3.0"
}

```

Opis najważniejszych komponentów z jakich skorzystałem:

twig – silnik szablonów strony umożliwił mi np. na używanie zmiennych w kodzie przekazanych przez kontroler.

doctrine orm – Umożliwia na zarządzania bazą danych za pomocą prostych komend.

Form – System formularzy generowanych automatycznie które można później zmodyfikować

Security-csrf – komponent dzięki któremu można ograniczyć i zapewnić dostęp do aplikacji określonej grupie użytkowników w moim przypadku są 2 grupy użytkowników tworzone ręcznie w bazie danych. („ROLE_ADMIN”), („ROLE_USER”)

Frontend:

Widoki – viewsy są zrobione za pomocą silnika szablonów twig.

Użyłem też bootstrapa w celu zrobienia lepiej wyglądających przycisków w porównaniu do domyślnych.

Wymagania niefunkcjonalne:

Aplikacja ma być dostępna bez przerwy 24/7/365.

Strona ma działać bardzo szybko – wszystkie operacje zajmują maksymalnie 1 s (Chyba, że ograniczeniem byłby ktoś wolny internet).

Aplikacja ma działać na wszystkich przeglądarkach internetowych na systemach windows/linux/android.

W kwestiach bezpieczeństwa w bazie nie ma żadnych wrażliwych danych typu numer dowodu czy numer karty płatniczej więc nawet wyciek danych nie spowodowałby jakiegś katastrofy na skalę

światową. Jedynymi danymi które mogą być dla kogoś ważne to jakie produkty sprzedajemy wewnątrz w naszej firmie oraz adresy zamieszkania wraz z imionami i nazwiskami.

Zagrożenia na które może być narażona aplikacja to ataki DDOS. Serwer na którym będzie aplikacja może się też fizycznie zepsuć jeśli będzie to serwer postawiony gdzieś w piwnicy.

Wymagania funkcjonalne:

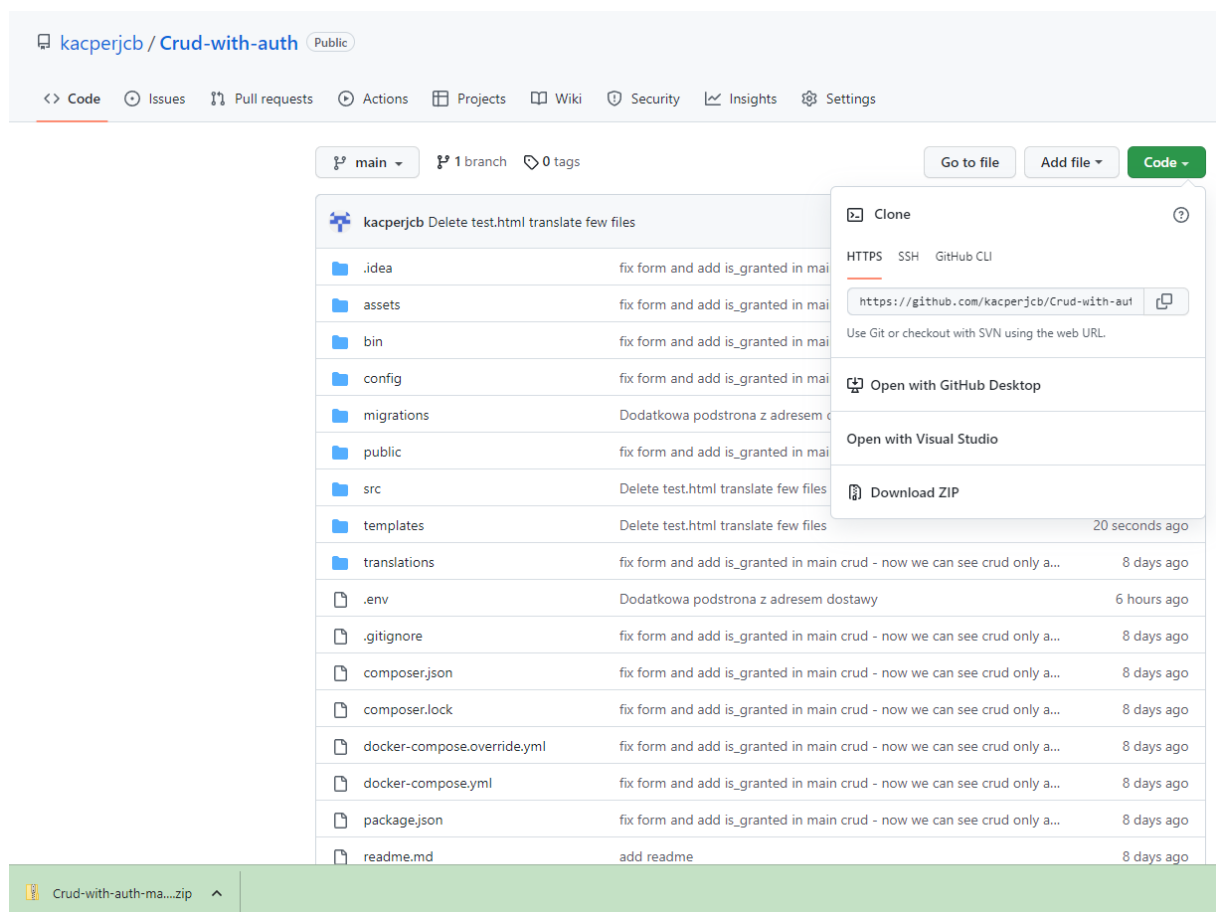
Aplikacja ma zapewnić użytkownikom możliwość zamówienia produktów które zostaną wcześniej dodane przez administratora.

Osoby postronne nie będą mieć dostępu do żadnych danych.

Instalacja i Testy aplikacji:

Najpierw pokażę jak zainstalować aplikację.

Najpierw klonujemy repozytorium lub pobieramy plik zip klikając Download Zip.



Następnie rozpakowujemy archiwum obojętnie gdzie.

Oto nasze pliki:

Dysk lokalny (C:) » Użytkownicy » Kacper » Pobrane » Crud-with-auth-main

Nazwa	Data modyfikacji	Typ	Rozmiar
.idea	15.06.2022 19:28	Folder plików	
assets	15.06.2022 19:28	Folder plików	
bin	15.06.2022 19:28	Folder plików	
config	15.06.2022 19:28	Folder plików	
migrations	15.06.2022 19:28	Folder plików	
public	15.06.2022 19:28	Folder plików	
src	15.06.2022 19:28	Folder plików	
templates	15.06.2022 19:28	Folder plików	
translations	15.06.2022 19:28	Folder plików	
.env	15.06.2022 19:28	Plik ENV	2 KB
.gitignore	15.06.2022 19:28	Dokument tekstowy	1 KB
composer.json	15.06.2022 19:28	JSON Source File	3 KB
composer.lock	15.06.2022 19:28	Plik LOCK	245 KB
docker-compose.override.yml	15.06.2022 19:28	Yaml Source File	1 KB
docker-compose.yml	15.06.2022 19:28	Yaml Source File	1 KB
package.json	15.06.2022 19:28	JSON Source File	1 KB
readme.md	15.06.2022 19:28	Markdown Source...	1 KB
symfony.lock	15.06.2022 19:28	Plik LOCK	11 KB
webpack.config.js	15.06.2022 19:28	JetBrains PhpStorm	3 KB
yarn.lock	15.06.2022 19:28	Plik LOCK	201 KB

```
C:\Windows\System32\cmd.exe - composer install
Microsoft Windows [Version 10.0.19044.1706]
(c) Microsoft Corporation. Wszelkie prawa zastrzeżone.

C:\Users\Kacper\Downloads\Crud-with-auth-main>composer install
Installing dependencies from lock file (including require-dev)
Verifying lock file contents can be installed on current platform.
Package operations: 82 installs, 0 updates, 0 removals
 - Installing symfony/flex (v2.1.8): Extracting archive
 - Installing symfony/runtime (v6.1.0): Extracting archive
 - Installing doctrine/cache (2.2.0): Extracting archive
 - Installing doctrine/collections (1.6.8): Extracting archive
 - Installing psr/cache (3.0.0): Extracting archive
 - Installing doctrine/event-manager (1.1.1): Extracting archive
 - Installing doctrine/persistence (3.0.2): Extracting archive
 - Installing doctrine/common (3.3.0): Extracting archive
 - Installing symfony/polyfill-mbstring (v1.25.0): Extracting archive
 - Installing symfony/deprecation-contracts (v3.1.0): Extracting archive
 - Installing symfony/http-foundation (v6.1.0): Extracting archive
 - Installing psr/event-dispatcher (1.0.0): Extracting archive
 - Installing symfony/event-dispatcher-contracts (v3.1.0): Extracting archive
 - Installing symfony/event-dispatcher (v6.1.0): Extracting archive
 - Installing symfony/var-dumper (v6.1.0): Extracting archive
 - Installing psr/log (3.0.0): Extracting archive
 - Installing symfony/error-handler (v6.1.0): Extracting archive
 - Installing symfony/http-kernel (v6.1.0): Extracting archive
 - Installing psr/container (2.0.2): Extracting archive
 - Installing symfony/service-contracts (v3.1.0): Extracting archive
 - Installing symfony/doctrine-bridge (v6.1.0): Extracting archive
 - Installing symfony/dependency-injection (v6.1.0): Extracting archive
 - Installing symfony/polyfill-intl-normalizer (v1.25.0): Extracting archive
```

```
C:\Users\Kacper\Downloads\Crud-with-auth-main>php bin/console doctrine:database:create
Created database `januszshop` for connection named default

C:\Users\Kacper\Downloads\Crud-with-auth-main>
```

```
C:\Users\Kacper\Downloads\Crud-with-auth-main>php bin/console doctrine:migrations:migrate

WARNING! You are about to execute a migration in database "januszshop" that could result in schema changes and data loss. Are you sure you wish to continue? (yes/no) [yes]:
>

[notice] Migrating up to DoctrineMigrations\Version20220614184501
[notice] finished in 164.1ms, used 18M memory, 4 migrations executed, 4 sql queries

C:\Users\Kacper\Downloads\Crud-with-auth-main>
```

```
C:\Users\Kacper\Downloads\Crud-with-auth-main>php bin/console security:hash-password
```

Symfony Password Hash Utility

Type in your password to be hashed:

>

Key	Value
Hasher used	Symfony\Component\PasswordHasher\Hasher\MigratingPasswordHasher
Password hash	\$2y\$13\$/V8mF0pUJFFwbeBhgtWe7.QmHrEn4w19MPXfDHGy3vwxSH2JV61U2

! [NOTE] Self-salting hasher used: the hasher generated its own built-in salt.

[OK] Password hashing succeeded

```
C:\Users\Kacper\Downloads\Crud-with-auth-main>
```

Użytkowników tworzymy w bazie danych w tabeli user.

Przykładowo stworzę email: admin@wp.pl

The screenshot shows a web application interface on the left and a terminal window on the right. The web application has a form with a 'roles' dropdown menu set to 'ADMIN' and a 'password' field. The terminal window shows the command 'php bin/console security:hash-password' being executed, resulting in a password hash.

Web Application Interface:

- roles: ADMIN
- password varchar(255):

Terminal Window:

```
C:\Windows\System32\cmd.exe

Type in your password to be hashed:
>

[ERROR] The password must not be empty.

Type in your password to be hashed:
> ^C
C:\Users\Kacper\Downloads\Crud-with-auth-main>php bin/console security:hash-password

Symfony Password Hash Utility
-----

Type in your password to be hashed:
>

Key          Value
-----
Hasher used   Symfony\Component\PasswordHasher\Hasher\MigratingPasswordHasher
Password hash $2y$13$P0sXS54Nq591vghP1cRu8upPzWm2KetZy7TI8I3Xy3Qfg7GVbNaT

! [NOTE] Self-salting hasher used: the hasher generated its own built-in salt.

[OK] Password hashing succeeded

C:\Users\Kacper\Downloads\Crud-with-auth-main>
```

oraz user@wp.pl analogicznie

Kolumna	Typ	Funkcja	Null	Wartość
id	int(11)			
email	varchar(180)			user@wp.pl
roles	longtext			["ROLE_USER"]

Konsola

\$2y\$13\$TTC3ffa0pFdDsDA89uMzOeeq/kduzy9cQHvV1cAzIiGwc1aK3TKAK

Teraz mogę uruchomić lokalny serwer symfony za pomocą komendy `symfony serve -d`, lub `symfony server:start`

```
C:\Users\Kacper\Downloads\Crud-with-auth-main>symfony serve -d

INFO  A new Symfony CLI version is available (5.4.11, currently running 5.4.8).

If you installed the Symfony CLI via a package manager, updates are going to be automatic.
If not, upgrade by downloading the new version at https://github.com/symfony-cli/symfony-cli/releases
And replace the current binary (symfony.exe) by the new one.

[WARNING] The local web server is optimized for local development and MUST never be used in a production setup.

[OK] Web server listening

The Web server is using PHP CGI 8.1.2

https://127.0.0.1:8000

Stream the logs via symfony.exe server:log

C:\Users\Kacper\Downloads\Crud-with-auth-main>
```

Powyższe komendy w kolejności:

`php bin/console doctrine:database:create`

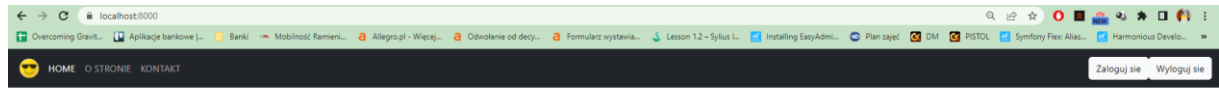
`php bin/console doctrine:migrations:migrate`

php bin/console security:hash-password

Oraz role które przypisujemy w trakcie tworzenia (jest to format json):

```
["ROLE_USER"], ["ROLE_ADMIN"]
```

Oto nasza strona startowa gdy nie jesteśmy zalogowani:



Aby zobaczyć produkty musisz się zalogować

Logowanie wygląda tak:

Gdy wpisujemy złe hasło to dostajemy powiadomienie.

Invalid credentials.

Logowanie

Email

admin@wp.pl

Hasło

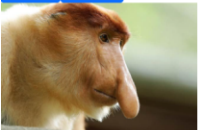
Zaloguj się

Gdy wpisujemy dobre to w zależności od tego czy mamy rolę admina czy użytkownika to wyświetli nam się nieco inny wygląd strony.

Tak wygląda panel admina zaraz po zalogowaniu gdy nie mamy żadnych rekordów.

E-commerce u Janusza

Zaloguj sięWyloguj się



Id	Nazwa produktu	Cena produktu	Stan magazynowy	Opis produktu	Akcje
no records found					

Nowy produkt

Wyszukaj produkt Pioter bo nie widzę

Search

Gdy klikniemy nowy produkt to przekierowuje nas do route /new w którym tworzymy nowy produkt.

Stwórz nowy produkt

Nazwa produktu

Cena produktu

Stan magazynowy

Opis produktu

Zapisz

Wróć do listy produktów

Dodałem przykładowo 3 produkty tak wygląda widok ze strony głównej gdy jesteśmy zalogowani jako administrator.

HOME O STRONIE KONTAKT

Zaloguj sięWyloguj się

Id	Nazwa produktu	Cena produktu	Stan magazynowy	Opis produktu	Akcje
1	Mysz hama emw-500	80	9	Mysz ergonomiczna	Pokaż Edytuj Sprzedaj
2	Podkładka steelseries qck+	40	9	Podkładka 45x40	Pokaż Edytuj Sprzedaj
3	Logitech k280e comfort	100	19	Klawiatura biurowa	Pokaż Edytuj Sprzedaj
4	Z małej litery	2354	3	test	Pokaż Edytuj Sprzedaj

[Nowy produkt](#) [Lista zamówień](#) [Same dane klientów](#)

Wyszukaj produkt po nazwie:

Szukaj

Gdy wejdziemy w opcję Pokaż to wyświetli nam się tylko jeden produkt

Produkt

Id	2
Nazwa produktu	Podkładka Steelseries QCK+
Cena produktu	40
Stan magazynowy	9
Opis produktu	Podkładka 45x40

[Wróć do listy produktów](#) [Edytuj](#)

Usuń

Opcja Edytuj umożliwia nam edycję lub usunięcie danej produktu.

Nazwa produktuPodkładka Steelseries QCK

Cena produktu40

Stan magazynowy9

Opis produktuPodkładka 45x40

Zaktualizuj

[Wróć do listy produktów](#)

Usuń

Opcja sprzedaj zmniejsza Stan magazynowy o jeden i przekierowuje nas do innego formularza w którym podajemy dane kupującego. Czyli przykładowo ktoś się do nas zgłosił, żeby coś mu kupić i chcemy w systemie to wprowadzić i wprowadzamy to z perspektywy administratora.

Admin w nawiasie pokazuje, że jesteśmy zalogowani jako admin. Jak user jest zalogowany to mamy w nawiasie „user” oraz nie mamy wtedy przycisku sprawdź listę wszystkich zamówień.

Dane do wysyłki (admin)

Imię:
Nazwisko:
Miasto:
Kod pocztowy:
Adres:

[Sprawdź listę wszystkich zamówień](#)

Gdy klikniemy przycisk Kup to zostaniemy przekierowani do strony z podziękowaniami

Dziękuję za zakup. Zakup zostanie zrealizowany w ciągu 48h

Jeśli będziemy mieli więcej produktów to będziemy mogli je też wyszukiwać po nazwie w polu znajdującym się pod przyciskiem Nowy produkt:

Id	Nazwa produktu	Cena produktu	Stan magazynowy	Opis produktu	Akcje
2	Podkładka steelseries qck+	40	5	Podkładka 45x40	Pokaż Edytuj Usuń

Wyszukaj produkt Fjoter bo nie widzę:

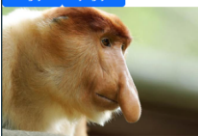
Gdy kliknę przycisk wyloguj się to automatycznie mnie wylogowuje.



Tak wygląda strona z pespektywy użytkownika:

E-commerce u Janusza

Zaloguj się Wyloguj się



Nazwa produktu	Cena produktu	Stan Magazynowy	Opis produktu	Akcje
Mysz hama emw-500	80	10	Mysz ergonomiczna	Kup
Podkładka steelseries qck+	40	5	Podkładka 45x40	Kup
Logitech k280e comfort	100	21	Klawiatura biurowa	Kup

Opcja kup działa tak samo jak sprzedaj – zmniejsza stan magazynowy o jeden i przekierowuje nas do strony z formularzem:

Dane do wysyłki (user)

Imię: Anon

Nazwisko: Kowalski

Miasto: Wrocław

Kod pocztowy: 71-234

Adres: 3 karton od prawej

Kup

Dziękuję za zakup. Zakup zostanie zrealizowany w ciągu 48h

Powrót do listy produktów

Aby później sprawdzić listę zamówień czyli klientów przypisanych do konkretnych zamówień to musimy kliknąć przycisk na stronie głównej (o ile jesteśmy zalogowani jako admin)

HOME O STRONIE KONTAKT

Zaloguj się Wyloguj się

Id	Nazwa produktu	Cena produktu	Stan magazynowy	Opis produktu	Akcje
1	Mysz hama eme-500	80	9	Mysz ergonomiczna	Pokaż Edytuj Sprzedaj
2	Podkładka steelseries qck+	40	9	Podkładka 45x40	Pokaż Edytuj Sprzedaj
3	Logitech k280e comfort	100	18	Klawiatura biurowa	Pokaż Edytuj Sprzedaj
4	Z. małej litery	2354	3	test	Pokaż Edytuj Sprzedaj

[Nowy produkt](#) [Lista zamówień](#) [Same dane klientów](#)

Wyszukaj produkt po nazwie

Szukaj

Imie	Nazwisko	Miasto	Kod_pocztowy	Adres	Nazwa_produktu	Opis_produktu
Kacper	Głowacki	Oława	55-200	Nie ważne 3/73	LOGITECH K280e Comfort	Klawiatura biurowa
Anon	Kowalski	Wrocław	71-234	3 karton od prawej	LOGITECH K280e Comfort	Klawiatura biurowa
Stefanus	J	Łódź	12-345	Sda	Podkładka Steelseries QCK+	Podkładka 45x40
Stefan	Wons	Oława	55-200	iwaskiewiczza	Mysz Hama EMW-500	Mysz ergonomiczna
Lukasz	C	Oława	550-200	Trołowa 33	LOGITECH K280e Comfort	Klawiatura biurowa
Maciej	Anoncki	Wrocław	52-340	Gromadzka 49/23	LOGITECH K280e Comfort	Klawiatura biurowa

Omówienie kodu:

CrudController – Kontroler odpowiadający za tą część kodu którą widzimy na stronie głównej oraz w opcjach gdzie tworzymy produkty.

Na górze mamy gety które pobierają wartości znakowe z inputa typu search oraz przycisk sell po którego kliknięciu otwiera się podstrona.

```

16  #Route('/')
17  class CrudController extends AbstractController
18  {
19      #Route('/', name: 'app_crud_index', methods: ['GET'])
20      public function index(CrudRepository $crudRepository, Request $request): Response
21      {
22
23          $search = $request->get('search');
24          $sell = $request->get('sell');
25
26
27          return $this->render('crud/index.html.twig', [
28              'cruds' => ($search) ? $crudRepository->search($search) : $crudRepository->findAll(),
29              'search' => $search,
30              'sell' => $sell,
31          ]);
32      }
33
34      #Route('/new', name: 'app_crud_new', methods: ['GET', 'POST'])
35      public function new(Request $request, CrudRepository $crudRepository): Response
36      {
37          $crud = new Crud();
38          $form = $this->createForm(new CrudType($crud));
39          $form->handleRequest($request);
40
41          if ($form->isSubmitted() && $form->isValid()) {
42              $crudRepository->add($crud, true);
43
44              return $this->redirectToRoute('app_crud_index', [], Response::HTTP_SEE_OTHER);
45          }
46
47          return $this->renderForm('crud/new.html.twig', [
48              'crud' => $crud,
49              'form' => $form,
50          ]);
51      }
52
53      #Route('/{id}', name: 'app_crud_show', methods: ['GET'])
54      public function show(Crud $crud): Response
55      {
56

```

Ważne jest to, że przesyłam do route'a „app_dane_klienta_new” wartość ID produktu. Dzięki temu mogę ją później przypisać do niewidzialnego formularza jako wartość domyślną. I odczytując dane z bazy wiem jaki użytkownik kupił jaki przedmiot.

```

#[Route('{id}/sell', name: 'app_crud_sell', methods: ['GET', 'POST'])]
public function sellproduct(ManagerRegistry $doctrine, int $id): Response
{
    {
        $entityManager = $doctrine->getManager();
        $crud = $entityManager->getRepository(Crud::class)->find($id);
        // $daneKlienta = $entityManager->getRepository(DaneKlienta::class);
        if (!$crud) {
            throw $this->createNotFoundException(
                'No product found for id '.$id
            );
        }

        $crud->setStanMagazynowy($crud->getStanMagazynowy()-1);
        $entityManager->flush();

        return $this->redirectToRoute(
            route: 'app_dane_klienta_new', [
                'id' => $crud->getId(),
            ]
        );
    }
}

```

Tutaj ustawiam `HiddenType::class` dzięki czemu nie widać formularza odpowiadającego za numer zamówienia bo tak jak pisałem wcześniej jest on i tak wpisywany automatycznie.

```

login.html.twig x show.html.twig x edit.html.twig x new.html.twig x AuthController.php x CrudType.php x DaneKlientaType.php x CrudController.php
1 <?php
2
3 namespace App\Form;
4
5 use ...
12
13 class DaneKlientaType extends AbstractType
14 {
15     public function buildForm(FormBuilderInterface $builder, array $options): void
16     {
17         $builder
18             ->add( child: 'Imie')
19             ->add( child: 'Nazwisko')
20             ->add( child: 'Miesto')
21             ->add( child: 'KodPocztowy')
22             ->add( child: 'Adres')
23             ->add( child: 'NumerZamowienia', type: HiddenType::class, [
24                 ]
25             );
26     }
27 }
28     public function configureOptions(OptionsResolver $resolver): void
29     {
30         $resolver->setDefaults([
31             'data_class' => DaneKlienta::class,
32         ]);
33     }
34 }
35 }
36

```

Linie które były generowane domyślnie są zakomentowane i w pliku

`Dane_klienta/_form.html.twig` znajduje się linia odpowiedzialna za domyślne przypisanie wartości `id` do formularza `app.request.query.get('id')` – funkcja działa prawdopodobnie tak samo jak `$_GET['id']` – ale w szablonach twig nie dało się jej użyć.

```

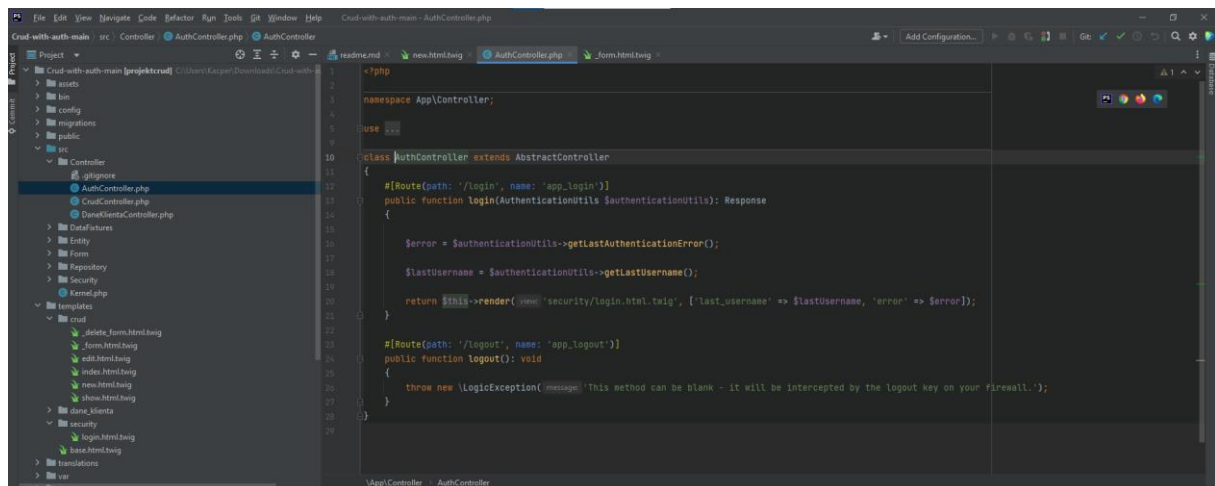
{{#{{ form_start(form) }}#}}
{{#    {{ form_widget(form) }}#}}

{{#{{ form_end(form) }}#}}
{{ form_start(form) }}

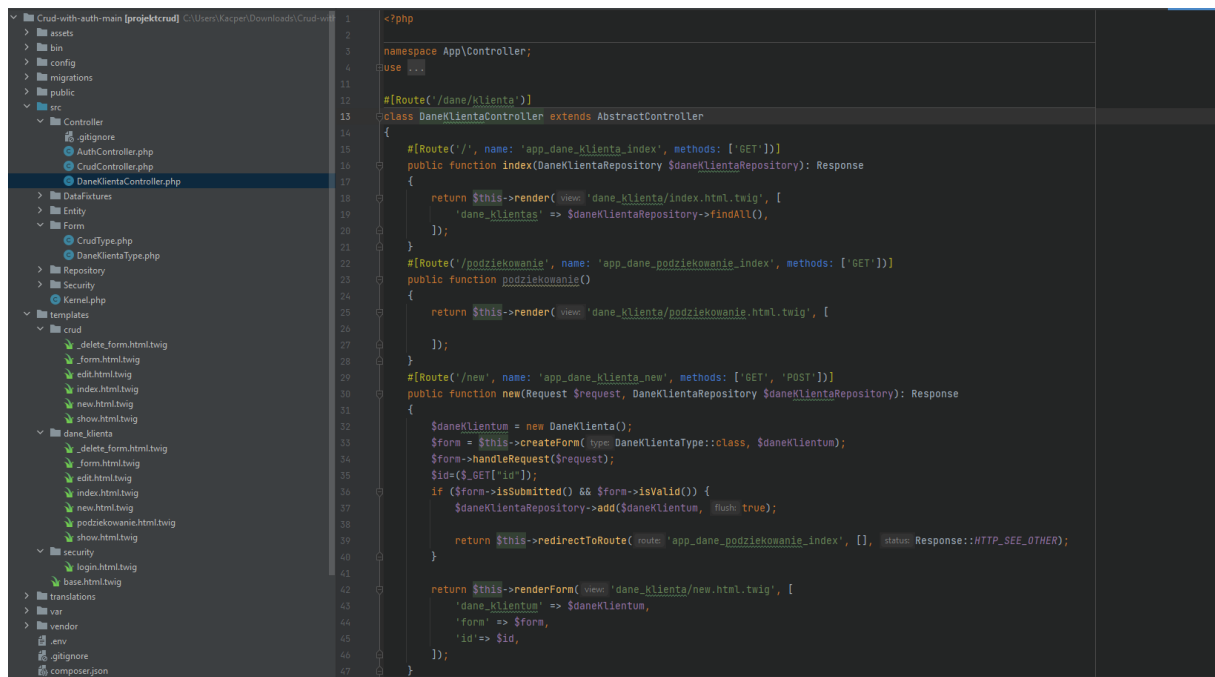
Imie:
{{ form_widget(form.Imie) }}
<br>
Nazwisko:
{{ form_widget(form.Nazwisko) }}
<br>
Miasto:
{{ form_widget(form.Miasto) }}
<br>
Kod pocztowy:
{{ form_widget(form.KodPocztowy) }}
<br>
Adres:
{{ form_widget(form.Adres) }}
{{ form_widget(form.NumerZamowienia, {'value' : app.request.query.get("id")}) }}
<br>
<button class="btn btn-primary">{{ button_label|default('Kup') }}</button>
{{ form_end(form) }}

```

Plik authController jest generowany automatycznie są w nim funkcjonalności od logowania i wylogowania



Controller/DaneKlientaController.php – odpowiada za formularz który otwiera się po przekierowaniu na stronę na której mamy do podania adres kupującego.



```
<?php
namespace App\Controller;
use ...

#[Route('/dane/klienta')]
class DaneKlientaController extends AbstractController
{
    #[Route('/', name: 'app_dane_klienta_index', methods: ['GET'])]
    public function index(DaneKlientaRepository $daneKlientaRepository): Response
    {
        return $this->render('view: 'dane_klienta/index.html.twig', [
            'dane_klientas' => $daneKlientaRepository->findAll(),
        ]);
    }

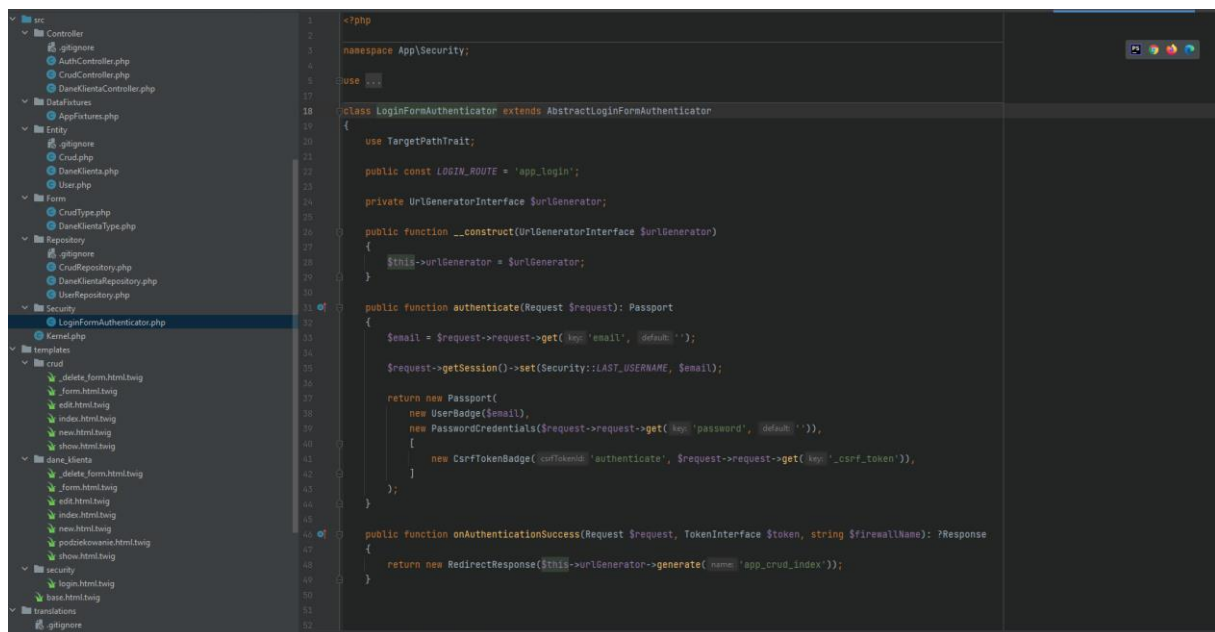
    #[Route('/podziekowanie', name: 'app_dane_podziekowanie_index', methods: ['GET'])]
    public function podziekowanie()
    {
        return $this->render('view: 'dane_klienta/podziekowanie.html.twig', [
        ]);
    }

    #[Route('/new', name: 'app_dane_klienta_new', methods: ['GET', 'POST'])]
    public function new(Request $request, DaneKlientaRepository $daneKlientaRepository): Response
    {
        $daneKlientum = new DaneKlienta();
        $form = $this->createForm(type: DaneKlientaType::class, $daneKlientum);
        $form->handleRequest($request);
        $id = ($form->isSubmitted() && $form->isValid()) ? $daneKlientaRepository->add($daneKlientum, flush: true);

        return $this->redirectToRoute(route: 'app_dane_podziekowanie_index', [], status: Response::HTTP_SEE_OTHER);
    }

    return $this->renderForm('view: 'dane_klienta/new.html.twig', [
        'dane_klientum' => $daneKlientum,
        'form' => $form,
        'id' => $id,
    ]);
}
```

LoginFormAuthenticator – Gdy wpisujemy prawidłowy login i hasło to w tym pliku w ostatniej linijce jest napisane na jaką stronę ma nastąpić przekierowanie.



```
<?php
namespace App\Security;
use ...

class LoginFormAuthenticator extends AbstractLoginFormAuthenticator
{
    use TargetPathTrait;

    public const LOGIN_ROUTE = 'app_login';

    private UrlGeneratorInterface $urlGenerator;

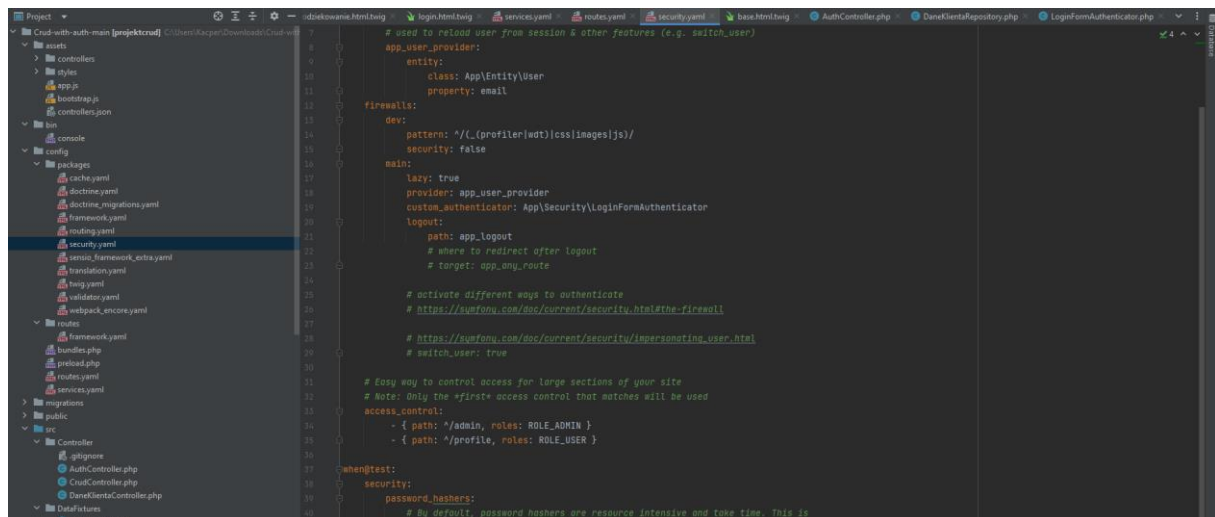
    public function __construct(UrlGeneratorInterface $urlGenerator)
    {
        $this->urlGenerator = $urlGenerator;
    }

    public function authenticate(Request $request): Passport
    {
        $email = $request->request->get('key:email', default: '');
        $request->getSession()->set(Security::LAST_USERNAME, $email);

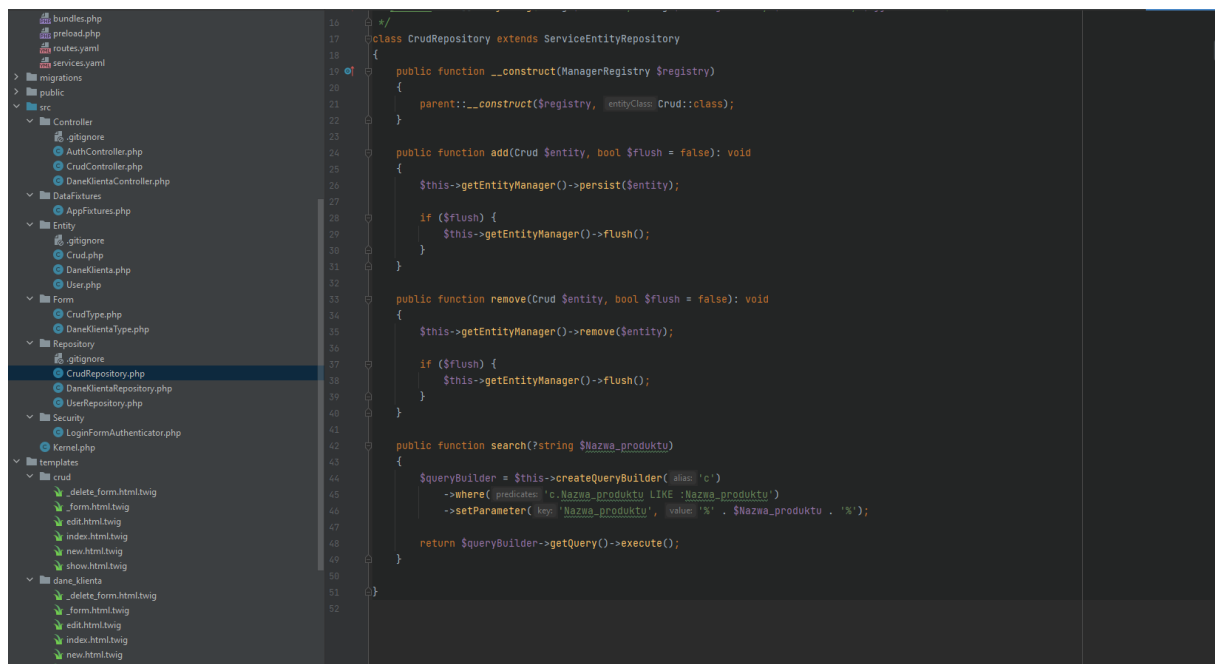
        return new Passport(
            new UserBadge($email),
            new PasswordCredentials($request->request->get('key:password', default: '')),
            [
                new CsrfTokenBadge('authenticate', $request->request->get('key:_csrf_token')),
            ]
        );
    }

    public function onAuthenticationSuccess(Request $request, TokenInterface $token, string $firewallName): ?Response
    {
        return new RedirectResponse($this->urlGenerator->generate(url: 'app_crud_index'));
    }
}
```

W pliku security.yaml musiałem odkomentować 2 linijki /admin oraz /user. Możemy w tym pliku też stworzyć więcej ról gdyby w przyszłości były potrzebne.



W pliku CrudRepository mamy zadeklarowaną funkcję search która znajduje się na stronie głównej i pozwala nam porównać string który wpisaliśmy z stringiem który posiadają nazwy przedmiotów w tabeli.



Strony tak jak wyżej napisałem są generowane za pomocą twig który umożliwia mi np. używanie instrukcji warunkowych oraz przydanych funkcji np. funkcji np. funkcja |capitalize zawsze wyświetla nam pierwszą literę wielką a resztę z małej. Nie ma znaczenia w jaki sposób ktoś wpisze nazwę produktu. Przykładowo „pOdkładka” – będzie zapisane jako „Podkładka”

Entity

.gitignore

Crud.php

DaneKlienta.php

User.php

Form

CrudType.php

DaneKlientaType.php

Repository

.gitignore

CrudRepository.php

DaneKlientaRepository.php

UserRepository.php

Security

LoginFormAuthenticator.php

Kernel.php

templates

crud

_delete_form.html.twig

_form.html.twig

edit.html.twig

index.html.twig

new.html.twig

show.html.twig

dane_klienta

_delete_form.html.twig

_form.html.twig

edit.html.twig

index.html.twig

new.html.twig

podziakowanie.html.twig

show.html.twig

security

login.html.twig

base.html.twig

translations

.gitignore

var

vendor

.env

.gitignore

composer.json

composer.lock

docker-compose.override.yml

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

```
{% if (is_granted('ROLE_ADMIN'))%} {# After successful log in we can see our crud. #}

<table class="table">
  <thead>
    <tr>
      <th>Id</th>
      <th>Nazwa_produkту</th>
      <th>Cena_produkту</th>
      <th>Stan_magazynowy</th>
      <th>Opis_produkту</th>
      <th>Akcje</th>
    </tr>
  </thead>
  <tbody>

    {% for crud in cruds %}
      <tr>
        <td>{{ crud.id }}</td>
        <td>{{ crud.NazwaProdukту|capitalize }}</td>
        <td>{{ crud.CenaProdukту }}</td>
        <td>{{ crud.StanMagazynowy }}</td>
        <td>{{ crud.OpisProdukту }}</td>
        <td>
          <a href="{{ path('app_crud_show', {'id': crud.id}) }}">Pokaż</a>
          <a href="{{ path('app_crud_edit', {'id': crud.id}) }}">Edytuj</a>
          <a href="{{ path('app_crud_sell', {'id': crud.id}) }}">Sprzedaj</a>
        </td>
      </tr>
    {% else %}
      <tr>
        <td colspan="6">no records found</td>
      </tr>
    {% endfor %}
  </tbody>
</table>

<a class="btn btn-primary" href="{{ path('app_crud_new') }}" role="button">Nowy produkt</a>

</form>
```

```

{%elseif (is_granted('ROLE_USER'))%}

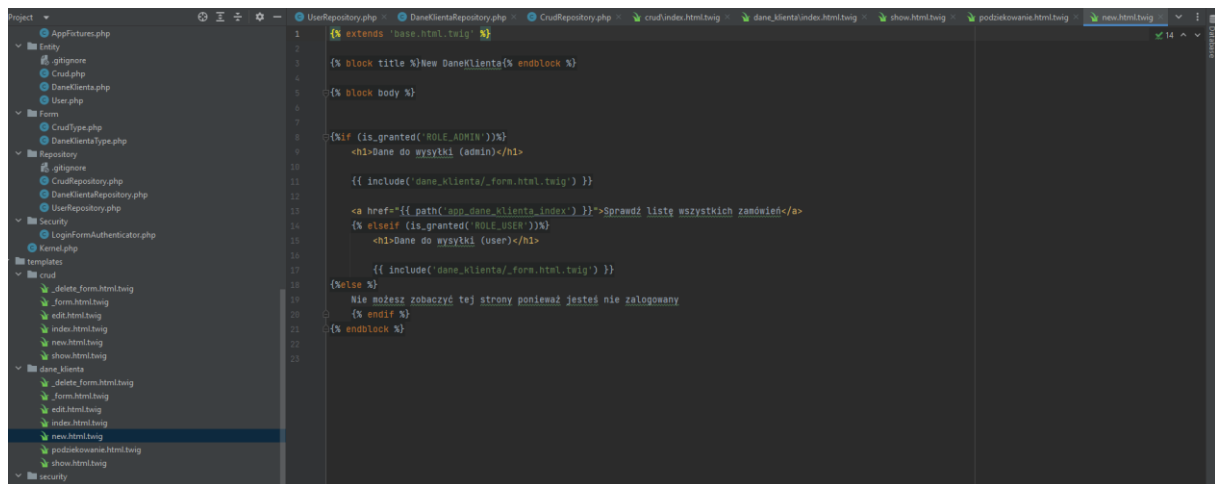
    <table class="table">
        <thead>
            <tr>
                <th>Nazwa produktu</th>
                <th>Cena produktu</th>
                <th>Stan Magazynowy</th>
                <th>Opis produktu</th>
                <th>Akcje</th>
            </tr>
        </thead>
        <tbody>

            {% for crud in cruds %}
                <tr>
                    <td>{{ crud.NazwaProduktu|capitalize }}</td>
                    <td>{{ crud.CenaProduktu }}</td>
                    <td>{{ crud.Stanmagazynowy }}</td>
                    <td>{{ crud.OpisProduktu }}</td>
                    <td>
                        <a href="{{ path('app_crud_sell', {'id': crud.id}) }}">Kup</a>
                    </td>
                </tr>
            {% else %}
                <tr>
                    <td colspan="6">no records found</td>
                </tr>
            {% endfor %}
        </tbody>
    </table>

    {% else %}
        <h1> Aby zobaczyć produkty musisz się zalogować </h1>

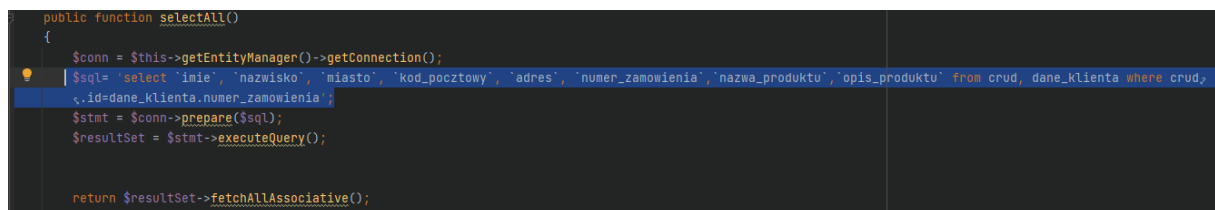
    {% endif %}
{% endblock %}

```

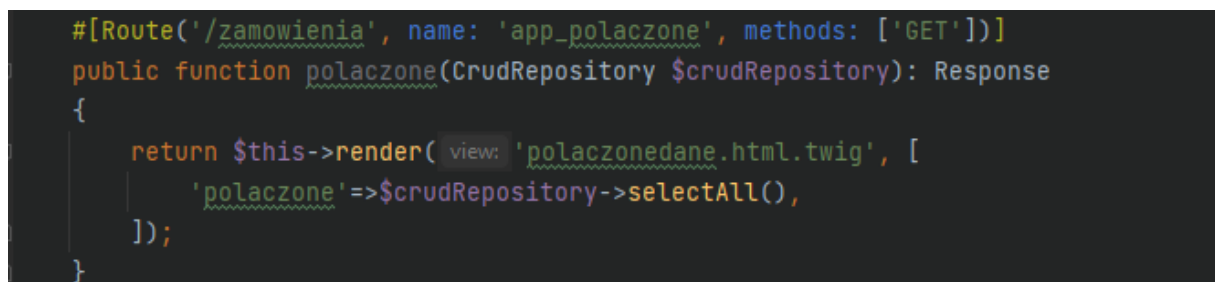


Podstrona zamówienia jest zrobiona w następujący sposób:

//crudRepository



//crudController



Istnieje też podstrona dane/klientów gdzie możemy sprawdzić same dane klientów – możemy w nią wejść też klikając w przycisk na stronie głównej

Id	Imie	Nazwisko	Miasto	KodPocztowy	Adres	actions
1	Kacper	Głowacki	Olawa	55-200	Nie ważne 3/73	show edit
2	Anon	Kowalski	Wrocław	71-234	3 karton od prawej	show edit
3	Stefanus	J	Łódź	12-345	Sda	show edit
4	Stefan	Wons	Olawa	55-200	iwaskiewiczza	show edit
5	Lukasz	C	Olawa	550-200	Trolowa 33	show edit
6	Maciej	Anoncki	Wrocław	52-340	Gromadzka 49/23	show edit

Wróć do listy produktów

Link do github: <https://github.com/kacperjcb/Crud-with-auth>

Diagram baz danych:

projektjanusz crud
id : int(11)
nazwa_produktu : varchar(255)
cena_produktu : double
stan_magazynowy : int(11)
opis_produktu : varchar(255)

projektjanusz user
id : int(11)
email : varchar(180)
roles : longtext
password : varchar(255)

projektjanusz dane_klienta
id : int(11)
imie : varchar(255)
nazwisko : varchar(255)
miasto : varchar(255)
kod_pocztowy : varchar(255)
adres : varchar(255)
numer_zamowienia : int(11)

projektjanusz doctrine_migration_versions
version : varchar(191)
executed_at : datetime
execution_time : int(11)