

# SPRAWOZDANIE

---

Informatyka / 2020/2021

## 1. Repozytorium

<https://github.com/pawelb95/CookBookWebPage>

## 2. Założenia

Plan pracy - harmonogram i wykres Gantta

	1	2	3	4	5	6	7	8	9	10
Analiza problemu	■									
Określenie funkcjonalności	■	■								
Modelowanie bazy danych		■								
Design GUI		■	■							
Szkielet aplikacji				■	■					
Testowanie i poprawki					■	■	■	■		
Rozwój funkcjonalności w Backendzie					■	■	■			
Autoryzacja						■	■			
Implementacja Frontendu							■	■	■	
Dokumentacja									■	■

## OPIS PROJEKTOWY

### Opis wymagań

- Wymagania funkcjonalne:
  - Aplikacja ma umożliwiać przeglądanie przepisów
  - Aplikacja ma umożliwiać sortowanie wyświetlanych przepisów
  - Aplikacja ma umożliwiać dodawanie przepisów

- Aplikacja ma umożliwiać zakładanie konta użytkownika
- Aplikacja ma umożliwiać logowanie/wylogowywanie
- Aplikacja ma umożliwiać przeglądanie dodanych przez siebie przepisów
- Aplikacja ma umożliwiać usuwanie przepisów
- Wymagania niefunkcjonalne:
  - Umożliwia funkcjonowanie aplikacji w przeglądarkach Chrome, Firefox i Edge
  - Umożliwia funkcjonowanie aplikacji gdy włączone są ciasteczka

### **Charakterystyka tematu**

Strona internetowa zawierająca bazę przepisów kulinarnych. Istnieje możliwość tworzenia konta oraz logowania się, a po zalogowaniu tworzenia przepisów. Przepisy znajdują się na stronie głównej, a po dodaniu nowego przepisu przez użytkownika jest on widoczny dla wszystkich. Użytkownik może wybrać sposób sortowania przepisów na stronie głównej: według nazwy albo czasu dodania.

### **Architektura środowiska i wykorzystywanych narzędzi**

Program jest podzielony na trzy aplikacje (część tworzonego programu jest tak nazywana w django). Pierwsza aplikacja, jest niejako główną częścią, która dzieli dostęp na część backendową ("api") oraz frontendową (" "). Dostęp jest tu dzielony na dwie kolejne aplikacje.

```
urlpatterns = [  
    path('admin/', admin.site.urls),  
    path('api/', include('api.urls')),  
    path('', include('frontend.urls')),  
]
```

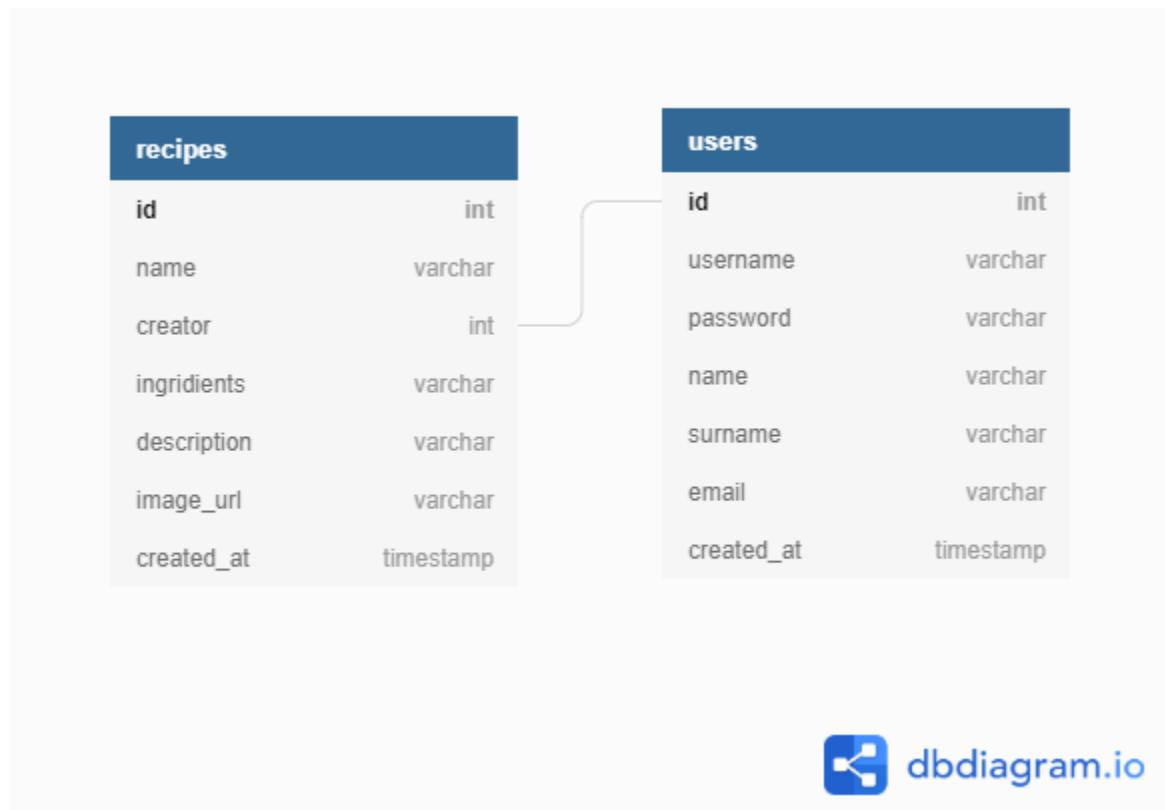
### **Wykorzystane technologie:**

- Python 3.9.0 w wersji 64 bitowej - Język programowania użyty do tworzenia serwera strony i części backendowej
- Django 3.1.3 - Framework użyty do utworzenia i zarządzania ustawieniami serwera oraz jego pracą. Użyty do tworzenia modeli bazodanowych oraz api backendowego oraz dostępu do frontendu
- MySQL 8.0 - Baza danych użyta do przechowywania użytkowników oraz przepisów
- React 17.0.1 - Framework do języka JavaScript (oraz TypeScript), który umożliwia łatwe tworzenie stron internetowych na bazie modelu komponentowego.

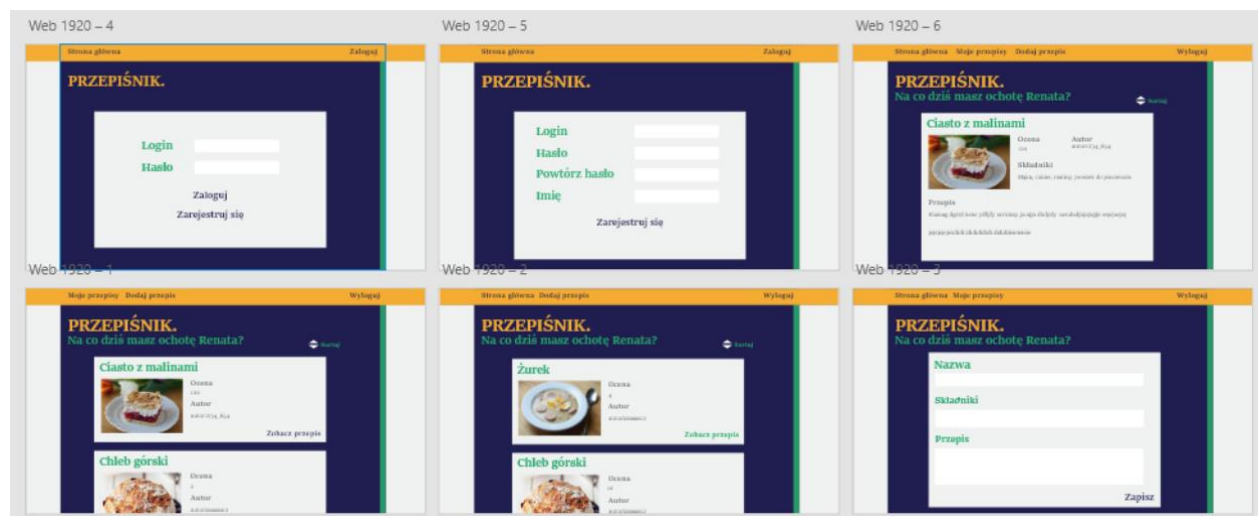
- WebPack 5.11.0 - Framework do języka JavaScript. Umożliwia “spakowanie” całej zawartości JavaScriptowej w jeden plik \*.js, tak aby był wysyłany do klienta tylko jeden plik javascript, przy czym jest on zmniejszany do jak najmniejszego rozmiaru.
- Material UI 4.11.2 - Narzędzie do popularnych frameworków typu Vue, Angular oraz React, które pozwala na używanie wcześniej zdefiniowanych komponentów jak np. przycisk czy menu rozwijane. Upraszcza pracę poprzez możliwość dowolnej modyfikacji zdefiniowanych wcześniej komponentów.
- Babel 7.12.10 - Framework, który ułatwia działanie całej aplikacji frontendowej nawet na przeglądarkach starszego typu lub takich, które nie posiadają całej potrzebnej funkcjonalności. Dzięki temu narzędziu łatwiejsze było dostosowanie naszej aplikacji do kilku przeglądarek.

## Diagram bazy danych, struktury aplikacji, przepływu danych.

Diagram bazy danych:



Struktura aplikacji:



```
graph TD
    Guest[Unregistered User (Guest)] --> Login[Login]
    Guest --> ShowAll[Show All Recipes]
    Login --> Registered[Registered User]
    Registered --> Edit[Edit Recipe]
    Registered --> ShowUser[Show User Recipes]
    Registered --> Create[Create Recipe]
    Registered --> ChangeInfo[Change User Information]
    Registered --> ChangePass[Change User Password]
    Registered --> Logout[Logout]
    ShowAll --> RecipeDS[Recipe DataStore]
    RecipeDS --> Edit
    RecipeDS --> ShowUser
    UsersDS[Users DataStore] --> Create
    UsersDS --> ChangeInfo
    UsersDS --> ChangePass
    UsersDS --> Logout
```



Część stron jest dostępna tylko dla użytkowników, którzy są zalogowani i mają dostęp do danych. Dla użytkownika zarejestrowanego dostępne są trzy strony.

Pierwsza z nich to strona główna, która jest również dostępna dla użytkowników zarejestrowanych i zalogowanych:

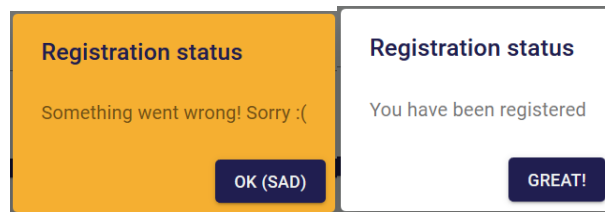


Strona ta umożliwia obejrzenie wszystkich dostępnych przepisów. Ponadto możliwe jest przeszukiwanie przepisów po nazwie. Wystarczy wpisać szukaną frazę lub jej część i następnie kliknąć przycisk **SEARCH**. Możliwe jest również sortowanie przepisów według nazwy (rosnąco i malejąco) oraz według daty utworzenia (rosnąco lub malejąco).

Drugą możliwością jest strona pozwalająca na utworzenie użytkownika w systemie:



Ta strona pozwala na utworzenie użytkownika, którego pola składają się na nazwę użytkownika (jego identyfikator, który nie jest możliwy do zmiany), hasło, adres e-mail, imię oraz nazwisko. Poprawność oraz niepoprawność rejestracji jest przedstawiana za pomocą wyskakującego okienka.



Trzecią możliwością jest strona logowania:



Ta strona umożliwia zalogowanie użytkownika. Trzeba tu wpisać nazwę użytkownika oraz hasło podane podczas rejestracji, aby można było się zalogować. Przy podaniu niepoprawnego użytkownika bądź hasła wyskakuje monit informujący o błędzie. Po podaniu poprawnych danych użytkownik jest przekierowany na stronę główną.

Oprócz strony głównej zarejestrowany użytkownik, za pomocą paska nawigacyjnego, może przejść na kolejne dostępne teraz dla niego strony.

Jedną z nich jest przeglądanie przepisów utworzonych przez zalogowanego użytkownika:


[HOME](#) [MY RECIPES](#) [ADD RECIPE](#) [MY SETTINGS](#)

The Recipe Book

LOGOUT

RECIPES OF USER: test5

## Faworki



### Ingridients

2 szklanki mąki pszennej (=300g)  
4 żółtka  
1 łyżka spirytusu (lub ocet 6%)  
½ łyżeczki cukru  
½ łyżeczki soli  
ok. 5 kopiatych łyżek gęstej, kwaśniej śmietany  
olej lub smalec do smażenia  
cukier puder do posypania

### Description

Mąkę wymieszać z cukrem i solą. Dodać żółtka, spirytus i śmietanę. Zagnieść na jednolitą masę. Następnie przełożyć ciasto na blat i zbijać drewnianym wałkiem ok. 10- 15 minut. (Należy uderzać wałkiem w ciasto rozplaszczając je, po czym ponownie zwinąć i znów zbijać wałkiem. Dzięki zbijaniu ciasto będzie jednolite, elastyczne, a po usmażeniu kruche i z dużą ilością bąbelków).

Ciasto rozwałkować porcjami cieniutko na blacie posypanym lekko mąką. (Ważne jest, aby blat podsypywać, jak najmniejszą ilością mąki. Ciasto oczekujące na rozwałkowanie należy przykryć ściereczką, żeby nie obsychało). Ciasto pokroić najpierw na paski o szerokości ok. 3- 4 cm, następnie na prostokąty lub równoległoboki o długości ok. 9- 10 cm. (Ja robię na oko tak, aby faworki ładnie wyglądały;)). Każdy kawałek naciąć w środku i przez nacięcie przeciągnąć jeden koniec.

Faworki smażyć na rozgrzanym tłuszczu z obu stron na złoty kolor. Wymować łyżką cedzakową i osączyć na ręczniku papierowym z tłuszczu. Gdy ostygną posypać grubo cukrem pudrem.

Strona to umożliwia nie tylko obejrzenie jakie są przepisy dodane przez aktualnie zalogowanego użytkownika, ale również do modyfikacji tych przepisów oraz usuwania ich ze strony internetowej za pomocą dodatkowych przycisków, które znajdują się pod nazwą potrawy. Przy opcji edytowanie wyskakuje monit, w którym można zapisać modyfikację lub anulować zmiany:



### Edit Recipe

name \*
Faworki

Ingridients \*
2 szklanki mąki pszennej (=300g)  
4 żółtka  
1 łyżka spirytusu (lub ocet 6%)  
½ łyżeczki cukru  
½ łyżeczki soli  
ok. 5 kopiatych łyżek gęstej, kwaśniej śmietany  
olej lub smalec do smażenia  
cukier puder do posypania

Description \*
Mąkę wymieszać z cukrem i solą. Dodać żółtka, spirytus i śmietanę. Zagnieść na jednolitą masę. Następnie przełożyć ciasto na blat i zbijać drewnianym wałkiem ok. 10- 15 minut. (Należy uderzać wałkiem w ciasto rozplaszczając je, po czym ponownie zwinąć i znów zbijać wałkiem. Dzięki zbijaniu ciasto będzie jednolite, elastyczne, a po usmażeniu kruche i z dużą ilością bąbelków).

Image URL
https://static.domowe-wypieki.pl/images/content/268/faworki\_3.jpg

SAVE CHANGES

CANCEL

Kolejną możliwością dla zalogowanego użytkownika jest możliwość tworzenia własnych przepisów:

HOME MY RECIPES ADD RECIPE MY SETTINGS

The Recipe Book

LOGOUT

CREATE YOUR OWN RECIPE

Name

Ingridients

Description

Image online url

ADD RECIPE TO COOKBOOK

Tworzenie przepisów składa się na wpisanie jego nazwy, składników, opisu jak wykonać daną potrawę oraz adresu url do zdjęcia w internecie np.:

[https://static.domowe-wypieki.pl/images/content/268/faworki\\_3.jpg](https://static.domowe-wypieki.pl/images/content/268/faworki_3.jpg)

Poprawne utworzenie jak i błędne wypełnienie danych jest przedstawione za pomocą wyskakującego okienka.

Następną możliwością dostępną dla zalogowanych użytkowników jest możliwość edytowania jego zapisanych wcześniej danych:

HOME MY RECIPES ADD RECIPE MY SETTINGS

# The Recipe Book

LOGOUT

## SETTINGS

### User Data

test3@test.pl

test3

test3

SAVE CHANGES

### Password Changing

Current Password

Password

Password confirmation

CHANGE PASSWORD

Jak widać na powyższym obrazie istnieje możliwość zmiany podstawowych danych w pierwszej części, na które składa się e-mail użytkownika, jego imię oraz nazwisko. Niemożliwa jest modyfikacja nazwy użytkownika, ponieważ jest to pole unikalne w bazie danych i jest identyfikatorem użytkownika. Możliwa jest również zmiana hasła w drugiej części strony, która wymaga podania również aktualnego hasła, aby możliwa była zmiana. Po próbie wykonania dowolnej operacji użytkownik jest informowany o jej powodzeniu.

Ostatnią stroną, na którą może wejść zalogowany użytkownik to strona wylogowania:

HOME MY RECIPES ADD RECIPE MY SETTINGS

# The Recipe Book

LOGOUT

You are logged out. Soon you will be redirected to homepage.

Jak pokazuje informacja przedstawiona na stronie, jest to moment wylogowania użytkownika, który kończy się przeniesieniem użytkownika na stronę główną.

### 3.3. Specyfikacja dla administratora

Aplikacja jest napisana za pomocą Pythona oraz frameworka Django. Dlatego aby uruchomić stronę internetową muszą być spełnione pewne warunki. Po pierwsze musi zostać zainstalowany i uruchomiony serwer MySQL w wersji 8.0. Do tego musi być utworzona na tym serwerze baza danych o nazwie *cookbookdb*. Użytkownik, który musi zostać utworzony, aby aplikacja miała dostęp do bazy nosi nazwę *taiuser* z hasłem *taiuser*. Możliwa jest modyfikacja tych ustawień w pliku *my.conf* w folderze *dbconf*. Po wykonaniu tych operacji, należy przejść do folderu, w którym znajduje się plik *manage.py* i wykonać poleceni, które spowodują migrację modeli do bazy danych:

```
$ python manage.py makemigrations
```

```
$ python manage.py migrate
```

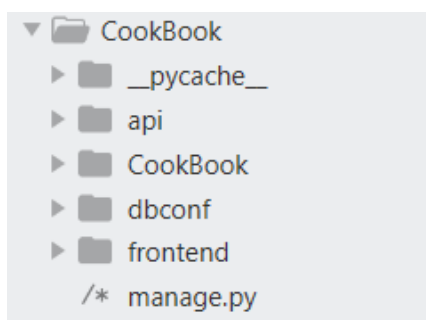
Po tych operacjach jest już utworzona baza danych i można uruchomić nasz serwer ze stroną internetową. Można to zrobić za pomocą kolejnego polecenia:

```
$ python manage.py runserver
```

Po tej operacji można przejść na jedną z obsługiwanych przeglądarek i wejść na naszą stronę internetową.

## 4. Specyfikacja Wewnętrzna

Jak już wspomniano w punkcie 1 sprawozdania aplikacja podzielona jest na 3 części: aplikację główną, frontendową i backendową. Struktura naszego programu przedstawia się następująco:



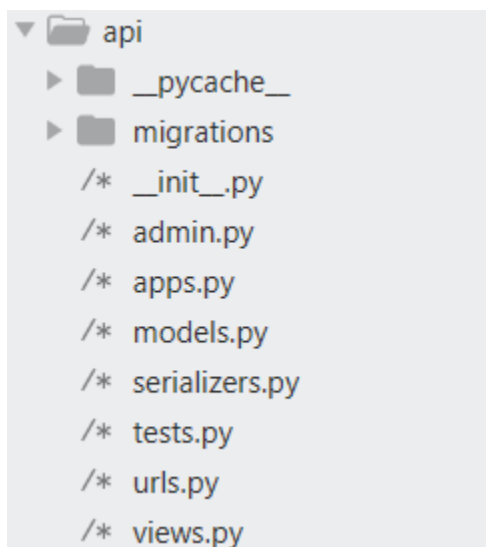
Folder *“api”* przechowują część backendową i logikę aplikacji. *“CookBook”* przechowuje aplikację główną wraz z ustawieniami. *“frontend”* jak sama nazwa wskazuje przechowuje część frontendową napisaną za pomocą React’a. Natomiast folder *“dbconf”* przechowuje jeden plik *“my.conf”*, który przechowuje ustawienia potrzebne do połączenia z bazą danych.

### 4.1. Aplikacja główna

Jest użyta jako główny element naszego programu. Scala wszystkie części i komponenty w jedną spójną aplikację, która zarządza dostępem. Jedną z najważniejszych funkcji jest możliwość połączenia z bazą danych MySQL. Niestety nie jest to rozwiązanie trywialne, ponieważ sterowniki do komunikacji z bazą danych trzeba szukać i dodawać ręcznie. Dodatkowo nie działają one z najnowszymi zabezpieczeniami bazy danych MySQL, więc potrzebna jest dodatkowa modyfikacja programu. Jest to wina nie zaktualizowanych jeszcze sterowników. W tej aplikacji są również pozostałe ustawienia jak np. sposób przechowywanie haseł w bazie danych.

#### 4.2. Aplikacja backendowa

Ta aplikacja przechowuje tak jak wcześniej wspomniano przede wszystkim logikę naszej strony. Składa się na to modyfikowanie rekordów w bazie danych, dostęp do rekordów, aby mogły zostać wysłane użytkownikowi, autentykacja użytkownika oraz inne elementy niezbędne do działania strony internetowej. Poniżej widoczna jest struktura tej aplikacji.



Jeden z ważniejszych plików, to plik “models.py”, który umożliwia zamodelowanie obiektów, tak jak będą później widoczne w bazie danych. Modele w formie tabel zostały wcześniej przedstawione.

Kolejny plik “serializers.py” ułatwia działanie na obiektach pobieranych z bazy danych oraz tych, które są tam dodawane poprzez tworzenie mechanizm, który przypomina translację elementów bazy danych na obiekty o podanych polach.

“urls.py” zawiera wszystkie ścieżki obsługiwane przez tą aplikację. Ze względu na to, że podlega pod aplikację główną, aby obsłużyć przedstawione poniżej punkty końcowe należy dodać przedrostek “/api”.

```
urlpatterns = [
    path('', index, name="apiIndex"),
    path('login', login, name="apiLogin"),
    path('logout', logout, name="apiLogout"),
    path('checkloginstatus', checkloginstatus),
    path('onlyforloginuser', onlyForLoginUser, name="apiOnlyForLoginUser"),
    path('users', UserView.as_view()),
    path('users/<int:user_id>', OneUserView.as_view()),
    path('updateuser/<int:id>', UpdateUserView.as_view()),
    path('updateuserpassword/<int:id>', UpdateUserPasswordView.as_view()),
    path('createuser', CreateUserView.as_view()),
    path('recipes', RecipeView.as_view()),
    path('recipes/<int:recipe_id>', OneRecipeView.as_view()),
    path('searchrecipes', getrecipesbynamepart),
    path('createrecipe', CreateRecipeView.as_view()),
    path('recipes/user/<int:user_id>', UserRecipeView.as_view()),
    path('removerecipe/<int:id>', RemoveRecipeView.as_view()),
    path('updaterecipe/<int:id>', UpdateRecipeView.as_view())
]
```

Najważniejszym plikiem jest “views.py”, który zawiera definicję tych punktów końcowych. Mówi o tym co ma zostać wykonane oraz w jaki sposób ma to zostać zrobione.

W części backendowej występowały miejsca szczególne, do których dojście zajęło nam sporo czasu. Przykładem takiego typu rzeczy są np. wspomniany wcześniej mechanizm “serializers”. Poprawnie użyte upraszczają wyszukiwanie i tworzenie obiektów modeli. Szczególnym przypadkiem, choć bardziej związany z zabezpieczeniami, było automatyczne szyfrowanie hasła, podczas modyfikacji hasła użytkownika. Najważniejszym jest tutaj użycie funkcji “to\_representation”, które zmienia wskazane u użytkownika pola:

```
17 class UpdateUserPasswordSerializer(serializers.ModelSerializer):
18     class Meta:
19         model = User
20         fields = ('id', 'username', 'password')
21
22     def to_representation(self, obj):
23         return {
24             "id": obj.id,
25             "username": obj.username,
26             "password": make_password(obj.password)
27         }
```

Ciekawym fragmentem kodu jest również wyszukiwanie przepisów. Sam w sobie fragment nie jest zbyt innowacyjny, jednak interesująca jest możliwość przeszukiwania pól obiektów szukając podśłów. Można to zrobić również uwzględniając wielkość liter. Najważniejsze tutaj jest użycie funkcji filter zwracającej obiekty, jeżeli takie istnieją w postaci specyficznej listy “QuerySet”, zawierającej nazwę pola z dopiskiem “name\_\_icontains”, wskazując, że szukamy w polu *name* każdego obiektu podanego modelu zawierającym szukaną frazę.

```

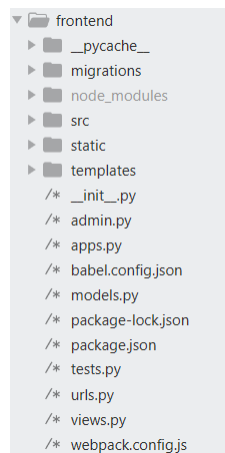
202 def getrecipesbynamepart(request):
203     request.data = json.loads(request.body)
204     search_fraze = request.data['search_fraze']
205     if search_fraze == "":
206         return HttpResponse("No search fraze", status=status.HTTP_400_BAD_REQUEST)
207     else:
208         found_recipes = Recipe.objects.filter(name__icontains=search_fraze)
209         if found_recipes:
210             serialized_found_recipes = RecipeSerializer(found_recipes, many=True)
211             return HttpResponse(json.dumps(serialized_found_recipes.data), status=status.HTTP_200_OK)
212         else:
213             return HttpResponse("Nothing matches", status=status.HTTP_400_BAD_REQUEST)
214

```

### 4.3. Aplikacja frontendowa

Druga aplikacja (frontendowa), składa się przede wszystkim z trzech ważnych elementów: pliku html (prosty plik zawierający bardzo prosty szablon strony), pliku css (prosty plik zawierający podstawowe ustawienia strony), pliku js (najważniejszy plik, który jest tworzony za pomocą nodeJS oraz Reacta, do udostępnienia widoków. Dostęp jest tu przesyłany z aplikacji głównej i całą resztą zajmuje się tutaj już React.) Podejście, w którym React jest wbudowany w aplikację pozwala na pominięcie problemu dotyczącego Cross-Origin. Problematyczne było wiele kwestii, w tym te związane z ustawieniami stron, tym jak mają wyglądać, jak ma być kolorystyka. Były również problemy techniczne, które spowodowały, że zdecydowaliśmy się na użycie React-Hooks, aby można było odpowiednio przesyłać dane między komponentami React'owymi. Kolejnym problemem było utrzymanie informacji o autentykacji, ze względu na to, że aplikacje napisane za pomocą tego Frameworka, są typowymi stronami SPA, które potrzebują specyficznego zarządzania tymi zasobami. Dodatkowo istniały problemy z tokenem csrf, które ma służyć zapobieganiu wkradnięcia się w komunikację przez inny komputer.

Struktura tej aplikacji prezentuje się następująco:



Z tej struktury można wyodrębnić trzy foldery:

- templates - zawiera stronę w postaci pliku .html, który służy do dołączenia pliku stylów .css oraz do dołączenia pliku .js z całą obsługą strony

- static - zawiera przede wszystkim plik .css, który zawiera część zdefiniowanych stylów (są też style zdefiniowane za pomocą React'a bezpośrednio w kodzie). Tutaj powinny również trafić pliki, które zawsze są wyświetlane na stronie jak np. loga stron. W naszym wypadku nie jest to potrzebne.
- src - najważniejszy z folderów, ponieważ zawiera definicję wszystkich komponentów utworzonych za pomocą frameworka React. Składają się na nie definicje stron, które są widoczne dla użytkownika oraz dodatkowe komponenty ułatwiające wyświetlanie jak np. pojedynczy przepis. Tak naprawdę to tutaj odbywa się całe działanie części frontendowej. Resztę plików można uznać za konfiguracyjne lub niezbędne do pobrania potrzebnych elementów takich jak Babel, czy Material-ui

Zarówno jak w przypadku aplikacji backendowej, frontendowa również posiada elementy, o których warto powiedzieć.

Jednym z takich elementów jest dodawanie tokena csrf do każdego zapytania, które jest wysyłane do serwera. Nie służy on do autentykacji, tylko do upewnienia się, że między nie zostanie przechwycona komunikacja i serwer nie zacznie nagle rozmawiać z innym komputerem. Jest do tego potrzebne dodanie specjalnego nagłówka w zapytaniu wraz z zawartością tokena, przy czym sam token musi zostać wzięty z ciasteczka przeglądarki.

```

73     handleSubmit(event) {
74         const {
75             name,
76             ingredients,
77             description,
78             creator,
79             image_url,
80         } = this.state;
81         axios.defaults.xsrfCookieName = "csrftoken";
82         axios.defaults.xsrfHeaderName = "X-CSRFToken";
83         axios.defaults.headers.post["X-CSRFToken"] = cookie.load("csrftoken");
84         axios.post("http://localhost:8000/api/createrecipe", {
85             name: name,
86             description: description,
87             ingredients: ingredients,
88             creator: creator,
89             image_url: image_url
90         },
91         { withCredentials: true }
92         ).then(response => {
93             this.setState({okDialogOpen: true})
94         }).catch(error => {
95             this.setState({wrongDialogOpen: true})
96         })
97     }

```

Również godnym wspomnienia jest użycie zdefiniowanych przez “material-ui” dialogów, czyli wyskakujących okienek, które mogą o czymś poinformować użytkownika, bądź zbierać od niego informacje. Pozwoliło to na ograniczenie ilości tworzonych komponentów, pozwalając na zwiększenie czytelności struktury programu i jego kodu. Przykładowy “Dialog” poniżej przedstawia wyświetlenie potwierdzenia rejestracji użytkownika i wygląda prawidłowo. Wspomnieć można wcześniejsze

pokazywania prac projektu, w których te okna były w bardzo słabym stanie rozciągając przyciski i źle wyświetlając dane.

```
197 <Dialog open={this.state.okDialogOpen}>
198   <DialogTitle classes={{root: classes.popupTitle}}> Registration status </DialogTitle>
199   <DialogContent>
200     <DialogContentText id="alert-dialog-description">
201       You have been registered
202     </DialogContentText>
203   </DialogContent>
204   <DialogActions>
205     <Button onClick={this.closePopup} color="primary" variant="contained" classes={{root: classes.buttonStyle}}> Great! </Button>
206   </DialogActions>
207 </Dialog>
```

Trudnym elementem, który zajął nam ogromną ilość czasu było przekazywanie informacji do komponentów, które są wyżej w hierarchii. Ze względu na to, że React stosuje model wodospadowy, czyli komponenty podrzędne nie mogą wpływać na komponenty nadrzędne (przynajmniej w założeniach podstawowego Reacta, bez Redux), w przypadku zmiany np. informacji o logowaniu, jeżeli sytuacja odbyła się w “niższym komponentcie” taka informacja powinna zostać przekazana do elementu wyżej. Naszym rozwiązaniem było użycie React Hooks, czyli podstawowego narzędzia do przekazywania informacji między komponentami, w celu przekazania pewnego rodzaju “odnośnika” do funkcji komponentu nadrzędnego, aby wywołał funkcję sprawdzającą stan zalogowania użytkownika.

```
46 class HomePage extends Component {
47   constructor(props) {
48     super(props);
49
50     this.state = {
51       data : [],
52       isEmpty : true,
53       isError: false,
54       sortType: "nameAsc",
55       username: this.props.userData.user.username,
56       search_fraze: "",
57       during_search: false
58     }
59     this.updateUserLoginState()
60
61     this.handleSortChange = this.handleSortChange.bind(this)
62     this.handleChange = this.handleChange.bind(this)
63     this.handleSearch = this.handleSearch.bind(this)
64     this.getSearchData = this.getSearchData.bind(this)
65   }
66
67
68   updateUserLoginState = () => {
69     this.props.updateUserLoginState()
70   }
```

## 5. Wnioski

Wykonanie projektu obejmującego aplikację internetową pozwoliło zapoznać się z technologiami, które można wykorzystać przy tworzeniu aplikacji tego typu: m.in. Django, React, WebPack i Material UI. Pomimo braku doświadczenia w tej dziedzinie i problemów, które zostały naotkane w trakcie realizacji projektu, efekt końcowy w



postaci działającej aplikacji był taki jak oczekiwano. Dzięki wykorzystaniu bazy danych MySQL do przechowywania danych aplikacji, nauczyłem się łączyć tę właśnie technologię z Django. Wszystkie z wymienionych technologii są szeroko wykorzystywane w praktyce i w zależności od wyboru ścieżki kariery, mogą pomóc w późniejszym życiu zawodowym.

Projekt ten może być użyty nie tylko jako książka do przepisów, ale także zbiór innych instrukcji jak chociażby rodzaje możliwych do utworzenia robót szydełkowych. Mogłoby to być wiele podobnych aplikacji przy zmianach wymagających niewielkiego nakładu pracy. Jest więc to na swój sposób aplikacja uniwersalna. Oczywiście nasze rozwiązanie mogłoby być rozwinięte o większą ilość funkcjonalności jak np. opinie o przepisach, czy podział na kategorie. Ciekawym byłoby również napisane tej samej aplikacji jednakże tym razem z użyciem innych technologii takich jak samego NodeJS'a, lub nawet języka JavaEE do zaprojektowania backendu oraz framework'ów Angular lub VueJS do utworzenia frontendu. Można by było również przetestować inne rozwiązania wspomagające programistów stron internetowych jak użycie BootStrap'a zamiast Material-ui.

## 6. Bibliografia

- <https://docs.djangoproject.com/en/3.1/>
- <https://material-ui.com/>
- <https://reactjs.org/>
- <https://nodejs.org/en/>
- <https://dev.mysql.com/doc/relnotes/mysql/8.0/en/>
- <https://docs.djangoproject.com/en/3.1/ref/databases/>
- <https://www.digitalocean.com/community/tutorials/how-to-create-a-django-app-and-connect-it-to-a-database>
- <https://medium.com/@itIsMadhavan/reactjs-props-vs-state-ff3a7680930d>
- <https://devpebe.com/2020/04/04/komunikacja-api-w-react-z-axios-kurs-react-cz-9/>
- <https://www.django-rest-framework.org/api-guide/serializers/>
- <https://www.django-rest-framework.org/api-guide/generic-views/>