# Dokumentacja API - Cocktail API

## Opis projektu

Projekt Cocktail API jest serwerową aplikacją REST API, stworzoną w technologii \*\*Node.js\*\* z wykorzystaniem \*\*Express.js\*\* oraz \*\*MongoDB\*\*. API umożliwia:

- Pobieranie listy koktajli z paginacją,

- Filtrowanie koktajli według zawartości alkoholu oraz składników,

- Sortowanie koktajli według nazwy,

- Dodawanie i usuwanie koktajli oraz składników,

- Przechowywanie i aktualizację zdjęć składników,

- Testowanie API przy użyciu frameworka \*\*Jest\*\*.

### Główne technologie użyte w projekcie:

- \*\*Node.js\*\* - środowisko uruchomieniowe,

- \*\*Express.js\*\* - framework backendowy,

- \*\*Mongoose\*\* - ORM dla MongoDB,

- \*\*MongoDB\*\* - baza danych (NoSQL),

- \*\*Multer\*\* - obsługa przesyłania plików,

- \*\*Jest\*\* - framework testowy,

- \*\*FS\*\* - obsługa plików do usuwania zdjęć.

## Endpoints API

### 1. Dodawanie koktajlu

```

POST /api/cocktails

```

#### Body:

```json

{

"name": "Mojito",

"category": "Longdrink",

"instructions": "Mix ingredients",

"ingredients": [{ "ingredient": "id\_składnika" }]

}

```

### 2. Pobieranie listy koktajli z paginacją, filtrowaniem i sortowaniem

```

GET /api/cocktails?page={page}&limit={limit}&isAlcoholic={true|false}&ingredient={ingredient\_name}&order={asc|desc}

```

#### Parametry:

- \*\*page\*\* (liczba) - numer strony,

- \*\*limit\*\* (liczba) - ilość wyników na stronę,

- \*\*isAlcoholic\*\* (boolean) - określa, czy koktajl zawiera alkohol,

- \*\*ingredient\*\* (string) - nazwa składnika,

- \*\*order\*\* (string) - sortowanie alfabetyczne rosnąco (\*\*asc\*\*) lub malejąco (\*\*desc\*\*).

### 3. Aktualizacja koktajlu

```

PUT /api/cocktails/:id

```

#### Body:

```json

{

"name": "Nowa nazwa",

"category": "Nowa kategoria"

}

```

### 4. Usuwanie koktajlu

```

DELETE /api/cocktails/:id

```

### 5. Dodawanie składnika

```

POST /api/ingredients

```

#### Body (multipart/form-data):

- \*\*name\*\* (string) - nazwa składnika,

- \*\*description\*\* (string) - opis,

- \*\*isAlcoholic\*\* (boolean) - czy zawiera alkohol,

- \*\*image\*\* (plik) - zdjęcie składnika.

### 6. Pobieranie listy składników

```

GET /api/ingredients

```

#### Odpowiedź:

```json

[

{

"\_id": "id",

"name": "Rum",

"description": "Alkohol",

"isAlcoholic": true,

"image": "http://localhost:5000/ingredientImages/rum.jpg"

}

]

```

### 7. Aktualizacja składnika

```

PUT /api/ingredients/:id

```

#### Body (multipart/form-data):

- \*\*name\*\* (string) - nowa nazwa,

- \*\*description\*\* (string) - nowy opis,

- \*\*isAlcoholic\*\* (boolean) - nowa wartość,

- \*\*image\*\* (plik) - nowe zdjęcie (jeśli podane, usuwa stare).

### 8. Usuwanie składnika

```

DELETE /api/ingredients/:id

```

## Struktura bazy danych

Baza danych \*\*MongoDB\*\* (technologia NoSQL) zawiera dwie kolekcje:

- \*\*cocktails\*\* - przechowuje informacje o koktajlach, w tym listę składników w postaci ich `\_id`.

- \*\*ingredients\*\* - przechowuje dostępne składniki wraz z ich zdjęciami.

Baza danych została stworzona sztucznie, ale istnieje możliwość dodawania zdjęć przez Postmana lub formularz. Przesłane pliki są przechowywane na serwerze i dostępne pod odpowiednim adresem URL.

## Zasady REST API

Podczas tworzenia projektu przestrzegano zasad REST API:

1. \*\*Klient-serwer\*\* - Frontend oddzielony od backendu (\*\*React + Node.js\*\*).

2. \*\*Brak stanu\*\* - Brak sesji na serwerze, autoryzacja odbywa się za pomocą \*\*JWT\*\*.

3. \*\*Ujednolicony interfejs\*\* - Standardowe metody HTTP: \*\*GET, POST, PUT, DELETE\*\*.

4. \*\*Identyfikacja zasobów\*\* - Każdy zasób posiada unikalny URL, np. `/api/cocktails/:id`.

5. \*\*Różne formaty danych\*\* - API obsługuje JSON, ale w tym przypadku przesyłany był jedynie JSON.

## Podsumowanie

Cocktail API to elastyczne i skalowalne rozwiązanie do zarządzania koktajlami i ich składnikami. Dzięki zastosowaniu nowoczesnych technologii, takich jak \*\*Node.js\*\*, \*\*Express.js\*\*, \*\*MongoDB\*\*, \*\*JWT\*\* i \*\*Multer\*\*, API spełnia standardy REST API oraz zapewnia łatwość integracji z aplikacjami frontendowymi.