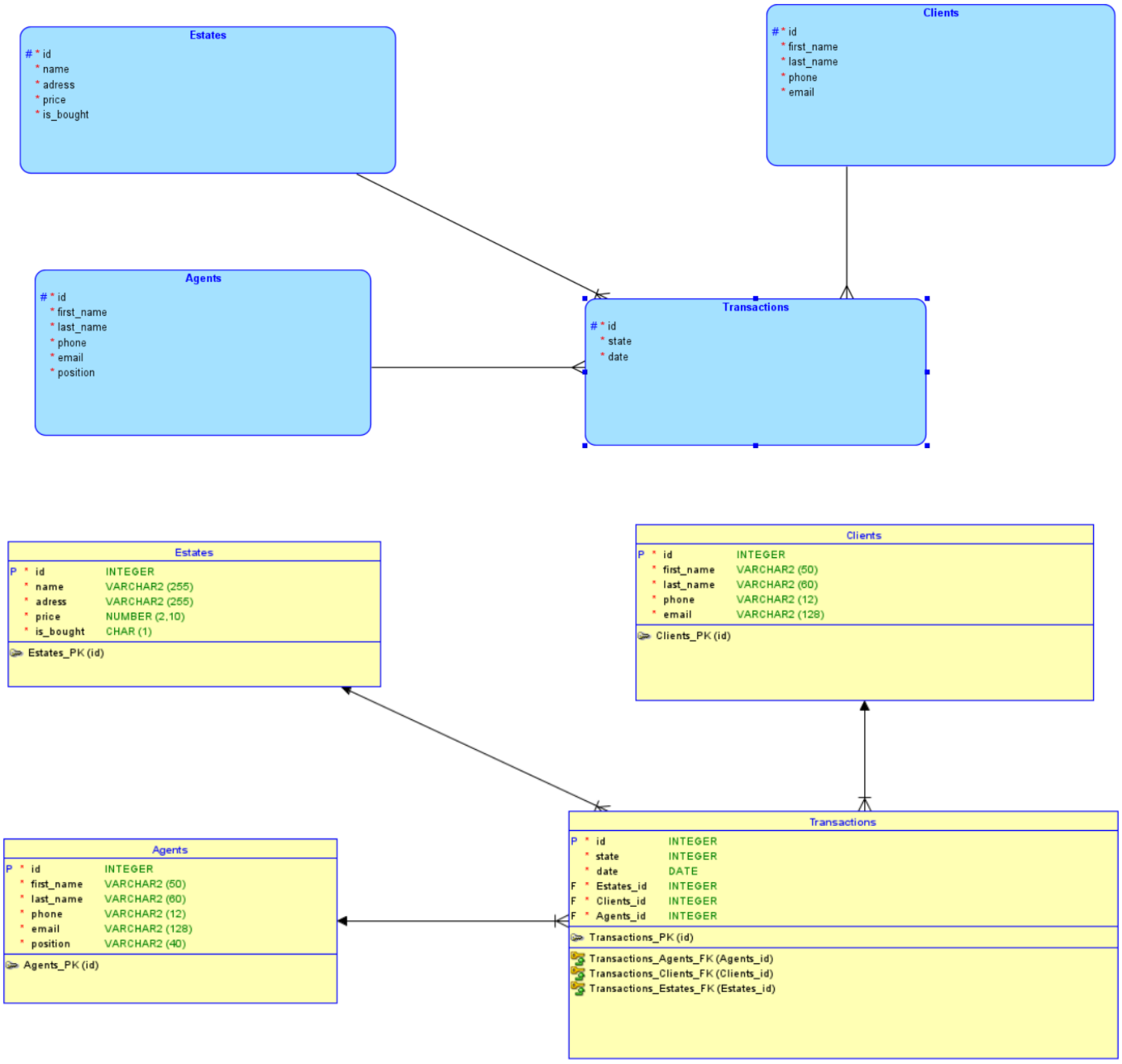
Projekt bazy danych:



Projekt API

Pobieranie informacji o klientach:

Endpoint: GET /clients

Opis: Pobiera listę wszystkich klientów.

Przykładowa odpowiedź:

{

"clients": [

```
    {
      "id": 1,
      "firstName": "Adam",
      "lastName": "Adamski",
      "phone": "123456789",
      "email": "adam.adamski@example.com"
    },
    {
      "id": 2,
      "firstName": "Barbara",
      "lastName": "Barbarska",
      "phone": "987654321",
      "email": "barbara.barbarska@example.com"
    }
  ]
}
```


Pobieranie informacji o konkretnym kliencie:

Endpoint: GET /clients/{id}

Opis: Pobiera informacje o konkretnym kliencie na podstawie identyfikatora.

Przykładowa odpowiedź:


```
{
  "id": 1,
   "firstName": "Adam",
   "lastName": "Adamski",
   "phone": "123456789",
   "email": "adam.adamski@example.com"
}
```

Dodawanie nowego klienta:

Endpoint: POST /clients

Opis: Dodaje nowego klienta do bazy danych na podstawie przekazanych danych.

Przykładowe żądanie:

```
{
  "firstName": "Adam",
  "lastName": "Adamski",
  "phone": "123456789",
  "email": "adam.adamski@example.com"
}
```

Przykładowa odpowiedź:

```
{
 "id": 1,
  "firstName": "Adam",
  "lastName": "Adamski",
  "phone": "123456789",
  "email": "adam.adamski@example.com"
}
```

Aktualizacja danych klienta:

Endpoint: PUT /clients/{id}

Opis: Aktualizuje informacje o konkretnym kliencie na podstawie identyfikatora.

Przykładowe żądanie:

```
{
  "firstName": "Adam",
  "lastName": "Adamski",
  "phone": "123456789",
  "email": "adam.adamski@example.com"
}
```

Przykładowa odpowiedź:


```
{
  "id": 1,
   "firstName": "Adam",
   "lastName": "Adamski",
   "phone": "123456789",
   "email": "adam.adamski@example.com"
}
```


Usunięcie klienta:

Endpoint: DELETE /clients/{id}

Opis: Usuwa konkretnego klienta na podstawie identyfikatora.

Przykładowa odpowiedź:


```
{
  "message": "Klient został usunięty."
}
```


Pobieranie informacji o agentach:

Endpoint: GET /agents

Opis: Pobiera listę wszystkich agentów.

Przykładowa odpowiedź:


```
{
  "agents": [
    {
      "id": 1,
      "firstName": "John",
      "lastName": "Doe",
      "phone": "123456789",
      "email": "john.doe@example.com",
```

```
    "position": "Agent"
   },
   {
    "id": 2,
    "firstName": "Jane",
    "lastName": "Smith",
    "phone": "987654321",
    "email": "jane.smith@example.com",
    "position": "Senior Agent"
   }
 ]
}
```


Pobieranie informacji o konkretnym agencie:

Endpoint: GET /agents/{id}

Opis: Pobiera informacje o konkretnym agencie na podstawie identyfikatora.

Przykładowa odpowiedź:


```
{
 "id": 1,
  "firstName": "Adam",
  "lastName": "Adamski",
  "phone": "123456789",
  "email": "adam.adamski@example.com"
  "position": "Agent"
}
```


Dodawanie nowego agenta:

Endpoint: POST /agents

Opis: Dodaje nowego agenta do bazy danych na podstawie przekazanych danych.

Przykładowe żądanie:

```
{
  "firstName": "Adam",
  "lastName": "Adamski",
  "phone": "123456789",
  "email": "adam.adamski@example.com"
  "position": "Agent"
}
```

Przykładowa odpowiedź:

```
{
 "id": 1,
  "firstName": "Adam",
  "lastName": "Adamski",
  "phone": "123456789",
  "email": "adam.adamski@example.com"
  "position": "Agent"
}
```

Aktualizacja danych agenta:

Endpoint: PUT /agents/{id}

Opis: Aktualizuje informacje o konkretnym agencie na podstawie identyfikatora.

Przykładowe żądanie:

```
{
  "firstName": "Adam",
  "lastName": "Adamski",
  "phone": "123456789",
  "email": “adam.adamski@example.com”
  "position": "Senior Agent"
}
```

Przykładowa odpowiedź:


```
{
  "id": 1,
   "firstName": "Adam",
   "lastName": "Adamski",
   "phone": "123456789",
   "email": “adam.adamski@example.com”
   "position": "Senior Agent"
}
```


Usunięcie agenta:

Endpoint: DELETE /agents/{id}

Opis: Usuwa konkretnego agenta na podstawie identyfikatora.

Przykładowa odpowiedź:


```
{
  "message": "Agent został usunięty."
}
```


Pobieranie informacji o nieruchomościach:

Endpoint: GET /estates

Opis: Pobiera listę wszystkich nieruchomości.

Przykładowa odpowiedź:


```
{
  "estates": [
    {
      "id": 1,
      "name": "Dom  1",
```

```
    "address": "Domowa 123",
    "price": 250000.00,
    "isBought": false,
   },
   {
    "id": 2,
    "name": "Dom 2",
    "address": "Domowa 234",
    "price": 350000.00,
    "isBought": true,
   }
 ]
}
```


Pobieranie informacji o konkretnej nieruchomości:

Endpoint: GET /estates/{id}

Opis: Pobiera informacje o konkretnej nieruchomości na podstawie identyfikatora.

Przykładowa odpowiedź:


```
{
    "id": 1,
    "id": 1,
    "name": "Dom  1",
    "address": "Domowa 123",
    "price": 250000.00,
    "isBought": false,
}
```

Dodawanie nowej nieruchomości:

Endpoint: POST /estates

Opis: Dodaje nową nieruchomość do bazy danych na podstawie przekazanych danych.


```
{
    "id": 1,
    "name": "Dom  1",
    "address": "Domowa 123",
    "price": 250000.00,
    "isBought": false,
}
```


Przykładowa odpowiedź:


```
{
    "id": 1,
    "name": "Dom  1",
    "address": "Domowa 123",
    "price": 250000.00,
    "isBought": false,
}
```


Aktualizacja danych nieruchomości:

Endpoint: PUT /estates/{id}

Opis: Aktualizuje informacje o konkretnej nieruchomości na podstawie identyfikatora.

Przykładowe żądanie:


```
{
    "name": "Dom  1",
    "address": "Domowa 124",
    "price": 280000.00,
    "isBought": false,
```

}
{
"id": 1,
"name": "Dom 1",
"address": "Domowa 124",
"price": 280000.00,
"isBought": false,
}

Usunięcie nieruchomości:

Endpoint: DELETE /estates/{id}

Opis: Usuwa konkretną nieruchomość na podstawie identyfikatora.

Przykładowa odpowiedź:

{
"message": "Nieruchomość została usunięta."
}

Pobieranie informacji o transakcjach:

Endpoint: GET /transactions

Opis: Pobiera listę wszystkich transakcji.

Przykładowa odpowiedź:

{
"transactions": [
{
"id": 1,
"clientId": 1,
"agentId": 1,

```
    "estateId": 1,
    "date": "2023-06-01",
    "state": 1
   },
   {
    "id": 2,
    "clientId": 2,
    "agentId": 2,
    "estateId": 2,
    "date": "2023-06-02",
    "state": 2
   }
 ]
}
```


Pobieranie informacji o konkretnej transakcji:

Endpoint: GET /transactions/{id}

Opis: Pobiera informacje o konkretnej transakcji na podstawie identyfikatora.

Przykładowa odpowiedź:


```
{
 "id": 1,
 "clientId": 1,
 "agentId": 1,
 "estateId": 1,
 "date": "2023-06-01",
 "state": 1
}
```

Dodawanie nowej transakcji:

Endpoint: POST /transactions

Opis: Dodaje nową transakcję do bazy danych na podstawie przekazanych danych.

Przykładowe żądanie:


```
{
  "clientId": 1,
  "agentId": 1,
  "estateId": 1,
  "date": "2023-06-01",
  "state": 1
}
```


Przykładowa odpowiedź:


```
{
  "id": 1,
  "clientId": 1,
  "agentId": 1,
  "estateId": 1,
  "date": "2023-06-01",
  "state": 1
}
```


Aktualizacja danych transakcji:

Endpoint: PUT /transactions/{id}

Opis: Aktualizuje informacje o konkretnej transakcji na podstawie identyfikatora.

Przykładowe żądanie:


```
{
  "clientId": 1,
```

```
  "agentId": 2,
  "estateId": 2,
  "date": "2023-06-02",
  "state": 2
}
```


Przykładowa odpowiedź:


```
{
  "id": 1,
  "clientId": 1,
  "agentId": 2,
  "estateId": 2,
  "date": "2023-06-02",
  "state": 2
}
```


Usunięcie transakcji:

Endpoint: DELETE /transactions/{id}

Opis: Usuwa konkretną transakcję na podstawie identyfikatora.

Przykładowa odpowiedź:


```
{
  "message": "Transakcja została usunięta."
}
```


Baza danych – implementacja

```sql
-- phpMyAdmin SQL Dump
-- version 5.2.1
-- https://www.phpmyadmin.net/
--
-- Host: 127.0.0.1
-- Generation Time: Jun 09, 2023 at 07:41 PM
-- Server version: 10.4.28-MariaDB
-- PHP Version: 8.0.28

SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
START TRANSACTION;
SET time_zone = "+00:00";


/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8mb4 */;

--
-- Database: `api`
--

-- --------------------------------------------------------

--
-- Table structure for table `agents`
--

CREATE TABLE `agents` (
  `id` int(11) NOT NULL,
  `first_name` varchar(50) NOT NULL,
  `last_name` varchar(60) NOT NULL,
  `phone` varchar(20) NOT NULL,
  `email` varchar(128) NOT NULL,
  `position` varchar(40) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

--
-- Dumping data for table `agents`
--

INSERT INTO `agents` (`id`, `first_name`, `last_name`, `phone`, `email`, `position`)
VALUES
(1, 'Jan', 'Kowalski', '123456789', 'jan.kowalski@example.com', 'Agent'),
(2, 'Anna', 'Nowak', '987654321', 'anna.nowak@example.com', 'Agent'),
(3, 'Piotr', 'Wójcik', '555111222', 'piotr.wojcik@example.com', 'Agent'),
(4, 'Małgorzata', 'Kamińska', '999888777', 'malgorzata.kaminska@example.com', 'Agent'),
(5, 'Krzysztof', 'Lewandowski', '444333222', 'krzysztof.lewandowski@example.com',
'Agent'),
(6, 'Agnieszka', 'Zielińska', '111222333', 'agnieszka.zielinska@example.com', 'Agent'),
(7, 'Marcin', 'Szymański', '777888999', 'marcin.szymanski@example.com', 'Agent'),
(8, 'Joanna', 'Woźniak', '222333444', 'joanna.wozniak@example.com', 'Agent'),
(9, 'Adam', 'Dąbrowski', '666555444', 'adam.dabrowski@example.com', 'Agent'),
```

```
(10, 'Katarzyna', 'Jankowska', '888999111', 'katarzyna.jankowska@example.com', 'Agent'),
(11, 'Michał', 'Mazur', '555666777', 'michal.mazur@example.com', 'Agent'),
(12, 'Karolina', 'Kowalczyk', '222111333', 'karolina.kowalczyk@example.com', 'Agent'),
(13, 'Rafał', 'Grabowski', '777666555', 'rafal.grabowski@example.com', 'Agent'),
(14, 'Monika', 'Pawlak', '999111222', 'monika.pawlak@example.com', 'Agent'),
(15, 'Tomasz', 'Michalski', '444555666', 'tomasz.michalski@example.com', 'Agent'),
(16, 'Iwona', 'Olszewska', '888111999', 'iwona.olszewska@example.com', 'Agent'),
(17, 'Łukasz', 'Nowicki', '333222111', 'lukasz.nowicki@example.com', 'Agent'),
(18, 'Magdalena', 'Jabłońska', '111444555', 'magdalena.jablonska@example.com', 'Agent'),
(19, 'Kamil', 'Król', '666777888', 'kamil.krol@example.com', 'Agent'),
(20, 'Natalia', 'Wróbel', '555444333', 'natalia.wrobel@example.com', 'Agent');

-- --------------------------------------------------------

--
-- Table structure for table `clients`
--

CREATE TABLE `clients` (
  `id` int(11) NOT NULL,
  `first_name` varchar(50) NOT NULL,
  `last_name` varchar(60) NOT NULL,
  `phone` varchar(12) NOT NULL,
  `email` varchar(128) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

--
-- Dumping data for table `clients`
--

INSERT INTO `clients` (`id`, `first_name`, `last_name`, `phone`, `email`) VALUES
(1, 'Jan', 'Kowalski', '123456789', 'jan.kowalski@example.com'),
(2, 'Anna', 'Nowak', '987654321', 'anna.nowak@example.com'),
(3, 'Piotr', 'Wójcik', '555111222', 'piotr.wojcik@example.com'),
(4, 'Małgorzata', 'Kamińska', '999888777', 'malgorzata.kaminska@example.com'),
(5, 'Krzysztof', 'Lewandowski', '444333222', 'krzysztof.lewandowski@example.com'),
(6, 'Agnieszka', 'Zielińska', '111222333', 'agnieszka.zielinska@example.com'),
(7, 'Marcin', 'Szymański', '777888999', 'marcin.szymanski@example.com'),
(8, 'Joanna', 'Woźniak', '222333444', 'joanna.wozniak@example.com'),
(9, 'Adam', 'Dąbrowski', '666555444', 'adam.dabrowski@example.com'),
(10, 'Katarzyna', 'Jankowska', '888999111', 'katarzyna.jankowska@example.com'),
(11, 'Michał', 'Mazur', '555666777', 'michal.mazur@example.com'),
(12, 'Karolina', 'Kowalczyk', '222111333', 'karolina.kowalczyk@example.com'),
(13, 'Rafał', 'Grabowski', '777666555', 'rafal.grabowski@example.com'),
(14, 'Monika', 'Pawlak', '999111222', 'monika.pawlak@example.com'),
(15, 'Tomasz', 'Michalski', '444555666', 'tomasz.michalski@example.com'),
(16, 'Iwona', 'Olszewska', '888111999', 'iwona.olszewska@example.com'),
(17, 'Łukasz', 'Nowicki', '333222111', 'lukasz.nowicki@example.com'),
(18, 'Magdalena', 'Jabłońska', '111444555', 'magdalena.jablonska@example.com'),
(19, 'Kamil', 'Król', '666777888', 'kamil.krol@example.com'),
(20, 'Natalia', 'Wróbel', '555444333', 'natalia.wrobel@example.com'),
(21, 'Paweł', 'Wieczorek', '222333444', 'pawel.wieczorek@example.com'),
(22, 'Alicja', 'Stępień', '888555444', 'alicja.stepien@example.com'),
(23, 'Artur', 'Piasecki', '777111222', 'artur.piasecki@example.com'),
```

```
(24, 'Marzena', 'Sikora', '333444555', 'marzena.sikora@example.com'),
(25, 'Dariusz', 'Baran', '111222333', 'dariusz.baran@example.com'),
(26, 'Ewa', 'Tomaszewska', '999888777', 'ewa.tomaszewska@example.com'),
(27, 'Tadeusz', 'Piotrowski', '666555444', 'tadeusz.piotrowski@example.com'),
(28, 'Agata', 'Wróblewska', '555444333', 'agata.wroblewska@example.com'),
(29, 'Robert', 'Nowakowski', '222333444', 'robert.nowakowski@example.com'),
(30, 'Dominika', 'Kwiatkowska', '888999111', 'dominika.kwiatkowska@example.com');

-- --------------------------------------------------------

--
-- Table structure for table `estates`
--

CREATE TABLE `estates` (
  `id` int(11) NOT NULL,
  `name` varchar(255) NOT NULL,
  `address` varchar(255) NOT NULL,
  `price` decimal(10,2) NOT NULL,
  `is_bought` tinyint(1) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

--
-- Dumping data for table `estates`
--

INSERT INTO `estates` (`id`, `name`, `address`, `price`, `is_bought`) VALUES
(1, 'Apartament przy Parku', 'ul. Kwiatowa 1, Warszawa', 250000.00, 0),
(2, 'Dom z ogrodem', 'ul. Słoneczna 2, Kraków', 450000.00, 0),
(3, 'Mieszkanie w centrum', 'ul. Główna 3, Wrocław', 180000.00, 0),
(4, 'Luksusowa willa', 'ul. Nadjeziorne 4, Gdańsk', 980000.00, 0),
(5, 'Kawalerka na wynajem', 'ul. Zielona 5, Poznań', 1200.00, 1),
(6, 'Apartamenty nad morzem', 'ul. Brzegowa 6, Sopot', 350000.00, 1),
(7, 'Dom w górach', 'ul. Skalista 7, Zakopane', 550000.00, 0),
(8, 'Mieszkanie z tarasem', 'ul. Spacerowa 8, Łódź', 220000.00, 0),
(9, 'Nowoczesny apartament', 'ul. Nowa 9, Szczecin', 280000.00, 1),
(10, 'Apartament w apartamentowcu', 'ul. Wieżowa 10, Katowice', 320000.00, 1),
(11, 'Mieszkanie w kamienicy', 'ul. Kamienista 11, Gdynia', 190000.00, 1),
(12, 'Dom na wsi', 'ul. Wiejska 12, Lublin', 420000.00, 0),
(13, 'Mieszkanie dla singla', 'ul. Singlowa 13, Bydgoszcz', 150000.00, 0),
(14, 'Luksusowy penthouse', 'ul. Wysoka 14, Rzeszów', 780000.00, 1),
(15, 'Apartament przy jeziorze', 'ul. Nadjeziorna 15, Olsztyn', 310000.00, 1),
(16, 'Mieszkanie dla rodziny', 'ul. Rodzinna 16, Opole', 200000.00, 0),
(17, 'Dom na osiedlu', 'ul. Osiedlowa 17, Zielona Góra', 480000.00, 0),
(18, 'Mieszkanie blisko centrum', 'ul. Centrum 18, Gorzów Wielkopolski', 170000.00, 0),
(19, 'Apartament z basenem', 'ul. Basenowa 19, Koszalin', 360000.00, 1),
(20, 'Mieszkanie dla studenta', 'ul. Akademicka 20, Legnica', 130000.00, 1),
(21, 'Komfortowy apartament', 'ul. Komfortowa 21, Częstochowa', 240000.00, 0),
(22, 'Dom na skraju lasu', 'ul. Lasowa 22, Kielce', 510000.00, 0),
(23, 'Mieszkanie z balkonem', 'ul. Balkonowa 23, Tarnów', 210000.00, 0),
(24, 'Apartament w historycznym budynku', 'ul. Historyczna 24, Płock', 330000.00, 1),
(25, 'Mieszkanie dla młodej pary', 'ul. Młodzieżowa 25, Radom', 160000.00, 1),
(26, 'Elegancki apartament', 'ul. Elegancka 26, Siedlce', 270000.00, 0),
(27, 'Dom z widokiem na jezioro', 'ul. Widokowa 27, Białystok', 590000.00, 0),
```

```
(28, 'Mieszkanie na parterze', 'ul. Parterowa 28, Ostrów Wielkopolski', 230000.00, 0),
(29, 'Apartament w stylowej kamienicy', 'ul. Stylowa 29, Kalisz', 340000.00, 1),
(30, 'Mieszkanie blisko plaży', 'ul. Plażowa 30, Toruń', 250000.00, 1),
(31, 'Wygodne mieszkanie', 'ul. Wygodna 31, Elbląg', 180000.00, 0),
(32, 'Dom z dużym ogrodem', 'ul. Ogrodowa 32, Gliwice', 460000.00, 0),
(33, 'Mieszkanie na poddaszu', 'ul. Poddasze 33, Krosno', 220000.00, 0),
(34, 'Apartament w centrum handlowym', 'ul. Handlowa 34, Tychy', 290000.00, 1),
(35, 'Mieszkanie w spokojnej okolicy', 'ul. Spokojna 35, Bielsko-Biała', 200000.00, 1),
(36, 'Dom na wzgórzu', 'ul. Wzgórzowa 36, Nowy Sącz', 550000.00, 0),
(37, 'Mieszkanie dla pracownika IT', 'ul. IT 37, Ruda Śląska', 150000.00, 0),
(38, 'Apartament z tarasem', 'ul. Tarasowa 38, Wałbrzych', 320000.00, 0),
(39, 'Mieszkanie dla emeryta', 'ul. Emerytów 39, Legnica', 190000.00, 1),
(40, 'Nowe mieszkanie', 'ul. Nowe 40, Gdynia', 240000.00, 1),
(41, 'Dom na przedmieściach', 'ul. Przedmieście 41, Lublin', 480000.00, 0),
(42, 'Mieszkanie z widokiem na rzekę', 'ul. Rzeczna 42, Bydgoszcz', 210000.00, 0),
(43, 'Apartament w luksusowym kompleksie', 'ul. Luksusowa 43, Rzeszów', 350000.00, 0),
(44, 'Mieszkanie dla rodziny z dziećmi', 'ul. Dziecięca 44, Olsztyn', 180000.00, 0),
(45, 'Dom z basenem', 'ul. Basenowa 45, Opole', 520000.00, 1),
(46, 'Mieszkanie na osiedlu zamkniętym', 'ul. Zamknięta 46, Zielona Góra', 250000.00, 0),
(47, 'Apartament blisko parku', 'ul. Parkowa 47, Gorzów Wielkopolski', 290000.00, 0),
(48, 'Mieszkanie w nowoczesnym budynku', 'ul. Nowoczesna 48, Koszalin', 200000.00, 0),
(49, 'Komfortowy apartament', 'ul. Komfortowa 49, Legnica', 280000.00, 0),
(50, 'Dom w malowniczej okolicy', 'ul. Malownicza 50, Częstochowa', 510000.00, 1),
(51, 'Mieszkanie z tarasem widokowym', 'ul. Taras Widokowy 51, Kielce', 220000.00, 0),
(52, 'Apartament w centrum handlowym', 'ul. Centrum Handlowe 52, Tarnów', 310000.00, 0),
(53, 'Mieszkanie na parterze z ogródkiem', 'ul. Parterowa 53, Płock', 190000.00, 0),
(54, 'Eleganckie studio', 'ul. Elegancka 54, Radom', 260000.00, 0),
(55, 'Dom w pobliżu lasu', 'ul. Lasowa 55, Siedlce', 530000.00, 1),
(56, 'Mieszkanie w centrum miasta', 'ul. Centrum 56, Białystok', 230000.00, 0),
(57, 'Apartament z widokiem na rzekę', 'ul. Rzeczna 57, Ostrów Wielkopolski', 330000.00,
0),
(58, 'Mieszkanie w kamienicy', 'ul. Kamienica 58, Kalisz', 200000.00, 0),
(59, 'Dom na skraju lasu', 'ul. Skraj Lasu 59, Toruń', 490000.00, 0),
(60, 'Mieszkanie dla singla', 'ul. Singlowa 60, Elbląg', 170000.00, 0);

-- --------------------------------------------------------

--
-- Table structure for table `transactions`
--

CREATE TABLE `transactions` (
  `id` int(11) NOT NULL,
  `client_id` int(11) DEFAULT NULL,
  `agent_id` int(11) DEFAULT NULL,
  `estate_id` int(11) DEFAULT NULL,
  `date` date DEFAULT NULL,
  `state` int(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

--
-- Dumping data for table `transactions`
--
```

```sql
INSERT INTO `transactions` (`id`, `client_id`, `agent_id`, `estate_id`, `date`, `state`)
VALUES
(1, 1, 2, 6, '2023-01-01', 1),
(2, 2, 3, 10, '2023-01-02', 2),
(3, 3, 1, 15, '2023-01-03', 1),
(4, 4, 2, 20, '2023-01-04', 2),
(5, 5, 3, 25, '2023-01-05', 1),
(6, 6, 1, 30, '2023-01-06', 2),
(7, 7, 2, 35, '2023-01-07', 1),
(8, 8, 3, 40, '2023-01-08', 2),
(9, 9, 1, 45, '2023-01-09', 1),
(10, 10, 2, 50, '2023-01-10', 2),
(11, 11, 3, 55, '2023-01-11', 1),
(12, 12, 1, 11, '2023-01-12', 2),
(13, 13, 2, 5, '2023-01-13', 1),
(14, 14, 3, 9, '2023-01-14', 2),
(15, 15, 1, 14, '2023-01-15', 1),
(16, 16, 2, 19, '2023-01-16', 2),
(17, 17, 3, 24, '2023-01-17', 1),
(18, 18, 1, 29, '2023-01-18', 2),
(19, 19, 2, 34, '2023-01-19', 1),
(20, 20, 3, 39, '2023-01-20', 2);

--
-- Indexes for dumped tables
--

--
-- Indexes for table `agents`
--
ALTER TABLE `agents`
  ADD PRIMARY KEY (`id`);

--
-- Indexes for table `clients`
--
ALTER TABLE `clients`
  ADD PRIMARY KEY (`id`);

--
-- Indexes for table `estates`
--
ALTER TABLE `estates`
  ADD PRIMARY KEY (`id`);

--
-- Indexes for table `transactions`
--
ALTER TABLE `transactions`
  ADD PRIMARY KEY (`id`),
  ADD KEY `client_id` (`client_id`),
  ADD KEY `agent_id` (`agent_id`),
  ADD KEY `estate_id` (`estate_id`);
```

```sql
--
-- AUTO_INCREMENT for dumped tables
--

--
-- AUTO_INCREMENT for table `agents`
--
ALTER TABLE `agents`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=21;

--
-- AUTO_INCREMENT for table `clients`
--
ALTER TABLE `clients`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=31;

--
-- AUTO_INCREMENT for table `estates`
--
ALTER TABLE `estates`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=61;

--
-- AUTO_INCREMENT for table `transactions`
--
ALTER TABLE `transactions`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=21;

--
-- Constraints for dumped tables
--

--
-- Constraints for table `transactions`
--
ALTER TABLE `transactions`
  ADD CONSTRAINT `transactions_ibfk_1` FOREIGN KEY (`client_id`) REFERENCES `clients`
(`id`),
  ADD CONSTRAINT `transactions_ibfk_2` FOREIGN KEY (`agent_id`) REFERENCES `agents`
(`id`),
  ADD CONSTRAINT `transactions_ibfk_3` FOREIGN KEY (`estate_id`) REFERENCES `estates`
(`id`);
COMMIT;

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```

app.js

C: > Users > barto > Desktop > Projekt Paw > Projekt PAW > app.js > ...

```
const express = require('express');
const mysql = require('mysql');
const bodyParser = require('body-parser');




const cors = require('cors');

const app = express();

// Enable CORS
app.use(cors());


app.use(bodyParser.json());

const connection = mysql.createConnection({
  host: 'localhost',



  user: 'admin',
  password: 'admin',
  database: 'api'
});

connection.connect(err => {
  if (err) {
    console.error('Error connecting to the database: ', err);
    return;
  }
  console.log('Connected to the database');
});

// // // AGENT START // // //

app.get('/agents', (req, res) => {
  const query = 'SELECT * FROM agents';

  connection.query(query, (err, results) => {
    if (err) {
      console.error('Error retrieving agents: ', err);
      res.status(500).send('Error retrieving agents');
      return;
    }
    res.status(200).json(results);
  });
});

app.post('/agents', (req, res) => {
  const { firstName, lastName, phone, email, position } = req.body;
```

```
  const query = 'INSERT INTO agents (first_name, last_name, phone, email, position) VALUES
(?, ?, ?, ?, ?)';
  const values = [firstName, lastName, phone, email, position];

  connection.query(query, values, (err, results) => {
    if (err) {
      console.error('Error creating a new agent: ', err);
      res.status(500).send('Error creating a new agent');
      return;
    }
    const newAgentId = results.insertId;
    res.status(201).json({ id: newAgentId, ...req.body });
  });
});


app.get('/agents/:id', (req, res) => {
  const query = 'SELECT * FROM agents WHERE id = ?';
  connection.query(query, [req.params.id], (err, results) => {
    if (err) {
      console.error('Error fetching agent: ', err);
      res.status(500).send('Error fetching agent');
      return;
    }
    if (results.length === 0) {
      res.status(404).send('Agent not found');
    } else {
      res.json(results[0]);
    }
  });
});


app.put('/agents/:id', (req, res) => {
  const agentId = req.params.id;
  const { firstName, lastName, phone, email, position } = req.body;
  const query = 'UPDATE agents SET first_name = ?, last_name = ?, phone = ?, email = ?,
position = ? WHERE id = ?';
  const values = [firstName, lastName, phone, email, position, agentId];

  connection.query(query, values, (err, results) => {
    if (err) {
      console.error('Error updating agent: ', err);
      res.status(500).send('Error updating agent');
      return;
    }
    if (results.affectedRows === 0) {
      res.status(404).send('Agent not found');
    } else {
      res.status(200).json({ id: agentId, firstName, lastName, phone, email, position });
    }
  });
});
```

```
app.delete('/agents/:id', (req, res) => {
  const agentId = req.params.id;
  const query = 'DELETE FROM agents WHERE id = ?';

  connection.query(query, [agentId], (err, results) => {
    if (err) {
      console.error('Error deleting agent: ', err);
      res.status(500).send('Error deleting agent');
      return;
    }
    if (results.affectedRows === 0) {
      res.status(404).send('Agent not found');
    } else {
      res.status(200).send('Agent deleted successfully');
    }
  });
});

// // // AGENT STOP // // //

// // // CLIENT START // // //

app.post('/clients', (req, res) => {
  const { firstName, lastName, phone, email } = req.body;
  const query = 'INSERT INTO clients (first_name, last_name, phone, email) VALUES (?, ?,
?, ?)';
  const values = [firstName, lastName, phone, email];

  connection.query(query, values, (err, results) => {
    if (err) {
      console.error('Error creating a new client: ', err);
      res.status(500).send('Error creating a new client');
      return;
    }
    const newClientId = results.insertId;
    res.status(201).json({ id: newClientId, ...req.body });
  });
});

app.get('/clients', (req, res) => {
  const query = 'SELECT * FROM clients';
  connection.query(query, (err, results) => {
    if (err) {
      console.error('Error fetching clients: ', err);
      res.status(500).send('Error fetching clients');
      return;
    }
    res.json(results);
  });
});

app.get('/clients/:id', (req, res) => {
  const query = 'SELECT * FROM clients WHERE id = ?';
  connection.query(query, [req.params.id], (err, results) => {
```

```
    if (err) {
      console.error('Error fetching client: ', err);
      res.status(500).send('Error fetching client');
      return;
    }
    if (results.length === 0) {
      res.status(404).send('Client not found');
    } else {
      res.json(results[0]);
    }
  });
});

app.put('/clients/:id', (req, res) => {
  const clientId = req.params.id;
  const { firstName, lastName, phone, email } = req.body;
  const query = 'UPDATE clients SET first_name = ?, last_name = ?, phone = ?, email = ?
WHERE id = ?';
  const values = [firstName, lastName, phone, email, clientId];

  connection.query(query, values, (err, results) => {
    if (err) {
      console.error('Error updating client: ', err);
      res.status(500).send('Error updating client');
      return;
    }
    if (results.affectedRows === 0) {
      res.status(404).send('Client not found');
    } else {
      res.status(200).json({ id: clientId, firstName, lastName, phone, email });
    }
  });
});

app.delete('/clients/:id', (req, res) => {
  const query = 'DELETE FROM clients WHERE id = ?';
  connection.query(query, [req.params.id], (err, results) => {
    if (err) {
      console.error('Error deleting client: ', err);
      res.status(500).send('Error deleting client');
      return;
    }
    if (results.affectedRows === 0) {
      res.status(404).send('Client not found');
    } else {
      res.sendStatus(204);
    }
  });
});

// // // CLIENT STOP // // //

// // // ESTATE START // // //
```

```
app.get('/estates', (req, res) => {
  const query = 'SELECT * FROM estates';

  connection.query(query, (err, results) => {
    if (err) {
      console.error('Error fetching estates: ', err);
      res.status(500).send('Error fetching estates');
      return;
    }
    res.json(results);
  });
});

app.get('/estates/:id', (req, res) => {
  const estateId = req.params.id;
  const query = 'SELECT * FROM estates WHERE id = ?';

  connection.query(query, [estateId], (err, results) => {
    if (err) {
      console.error('Error fetching estate: ', err);
      res.status(500).send('Error fetching estate');
      return;
    }
    if (results.length === 0) {
      res.status(404).send('Estate not found');
    } else {
      res.json(results[0]);
    }
  });
});

app.post('/estates', (req, res) => {
  const { name, address, price, is_bought } = req.body;
  const query = 'INSERT INTO estates (name, address, price, is_bought) VALUES (?, ?, ?,
?)';
  const values = [name, address, price, is_bought];

  connection.query(query, values, (err, results) => {
    if (err) {
      console.error('Error creating a new estate: ', err);
      res.status(500).send('Error creating a new estate');
      return;
    }
    const newEstateId = results.insertId;
    res.status(201).json({ id: newEstateId, ...req.body });
  });
});

app.put('/estates/:id', (req, res) => {
  const estateId = req.params.id;
  const { name, address, price, is_bought } = req.body;
  const query = 'UPDATE estates SET name = ?, address = ?, price = ?, is_bought = ? WHERE
id = ?';
  const values = [name, address, price, is_bought, estateId];
```

```
  connection.query(query, values, (err, results) => {
    if (err) {
      console.error('Error updating the estate: ', err);
      res.status(500).send('Error updating the estate');
      return;
    }
    res.status(200).json({ id: estateId, ...req.body });
  });
});

app.delete('/estates/:id', (req, res) => {
  const estateId = req.params.id;
  const query = 'DELETE FROM estates WHERE id = ?';

  connection.query(query, [estateId], (err, results) => {
    if (err) {
      console.error('Error deleting the estate: ', err);
      res.status(500).send('Error deleting the estate');
      return;
    }
    res.status(200).json({ id: estateId, message: 'Estate deleted successfully' });
  });
});


// // // ESTATE STOP // // //

// // // TRANSACTION START // // //

app.get('/transactions', (req, res) => {
  const sql = 'SELECT * FROM transactions';

  connection.query(sql, (err, results) => {
    if (err) {
      console.error(err);
      return res.status(500).json({ error: 'Internal server error' });
    }

    res.json(results);
  });
});

app.get('/transactions/:id', (req, res) => {
  const id = req.params.id;
  const sql = 'SELECT * FROM transactions WHERE id = ?';

  connection.query(sql, [id], (err, result) => {
    if (err) {
      console.error(err);
      return res.status(500).json({ error: 'Internal server error' });
    }

    if (result.length === 0) {
```

```
      return res.status(404).json({ error: 'Transaction not found' });
    }

    res.json(result[0]);
  });
});

app.post('/transactions', (req, res) => {
  const { client_id, agent_id, estate_id, date, state } = req.body;

  const agentCheckQuery = 'SELECT id FROM agents WHERE id = ?';
  connection.query(agentCheckQuery, [agent_id], (err, agentResult) => {
    if (err) {
      console.error(err);
      return res.status(500).json({ error: 'Internal server error' });
    }

    if (agentResult.length === 0) {
      return res.status(400).json({ error: 'Agent does not exist' });
    }

    const clientCheckQuery = 'SELECT id FROM clients WHERE id = ?';
    connection.query(clientCheckQuery, [client_id], (err, clientResult) => {
      if (err) {
        console.error(err);
        return res.status(500).json({ error: 'Internal server error' });
      }

      if (clientResult.length === 0) {
        return res.status(400).json({ error: 'Client does not exist' });
      }

      const estateCheckQuery = 'SELECT id FROM estates WHERE id = ?';
      connection.query(estateCheckQuery, [estate_id], (err, estateResult) => {
        if (err) {
          console.error(err);
          return res.status(500).json({ error: 'Internal server error' });
        }

        if (estateResult.length === 0) {
          return res.status(400).json({ error: 'Estate does not exist' });
        }

        const sql = 'INSERT INTO transactions (client_id, agent_id, estate_id, date,
state) VALUES (?, ?, ?, ?, ?)';
        const values = [client_id, agent_id, estate_id, date, state];

        connection.query(sql, values, (err, result) => {
          if (err) {
            console.error(err);
            return res.status(500).json({ error: 'Internal server error' });
          }

          res.json({ message: 'Transaction added successfully' });
```

```
        });
      });
    });
  });
});



app.put('/transactions/:id', (req, res) => {
  const id = req.params.id;
  const { client_id, agent_id, estate_id, date, state } = req.body;

  const checkQuery = 'SELECT id FROM agents WHERE id = ?';
  connection.query(checkQuery, [agent_id], (err, agentResult) => {
    if (err) {
      console.error(err);
      return res.status(500).json({ error: 'Internal server error' });
    }

    if (agentResult.length === 0) {
      return res.status(400).json({ error: 'Agent does not exist' });
    }

    const checkQuery = 'SELECT id FROM clients WHERE id = ?';
    connection.query(checkQuery, [client_id], (err, clientResult) => {
      if (err) {
        console.error(err);
        return res.status(500).json({ error: 'Internal server error' });
      }

      if (clientResult.length === 0) {
        return res.status(400).json({ error: 'Client does not exist' });
      }

      const checkQuery = 'SELECT id FROM estates WHERE id = ?';
      connection.query(checkQuery, [estate_id], (err, estateResult) => {
        if (err) {
          console.error(err);
          return res.status(500).json({ error: 'Internal server error' });
        }

        if (estateResult.length === 0) {
          return res.status(400).json({ error: 'Estate does not exist' });
        }

        const sql = 'UPDATE transactions SET client_id = ?, agent_id = ?, estate_id = ?,
date = ?, state = ? WHERE id = ?';
        const values = [client_id, agent_id, estate_id, date, state, id];

        connection.query(sql, values, (err, result) => {
          if (err) {
            console.error(err);
            return res.status(500).json({ error: 'Internal server error' });
          }
```

```
          res.json({ message: 'Transaction updated successfully' });
        });
      });
    });
  });
});

app.delete('/transactions/:id', (req, res) => {
  const id = req.params.id;
  const sql = 'DELETE FROM transactions WHERE id = ?';

  connection.query(sql, [id], (err, result) => {
    if (err) {
      console.error(err);
      return res.status(500).json({ error: 'Internal server error' });
    }

    res.json({ message: 'Transaction deleted successfully' });
  });
});


// // // TRANSACTION STOP // // //


const port = 3000;
app.listen(port, () => {
  console.log(`Server running on port ${port}`);
});
```

Prezentacja API