CookApp

Dokumentacja techniczna

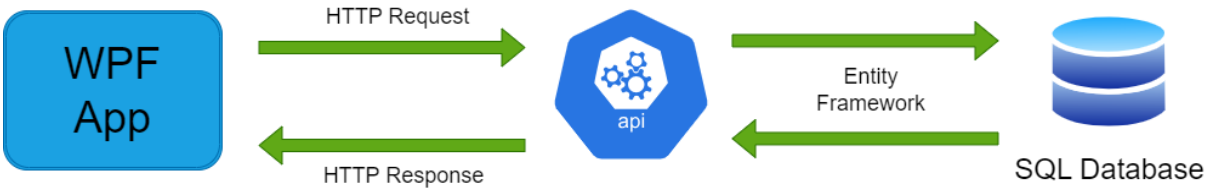
Autor: Szymon Korus

Spis treści

Czym jest CookApp?

CookApp jest aplikacją wspomagającą pracę lokalu gastronomicznego poprzez zarządzanie składanymi zamówieniami przez kelnerów i odpowiedniemu przydzielaniu ich kucharzom.

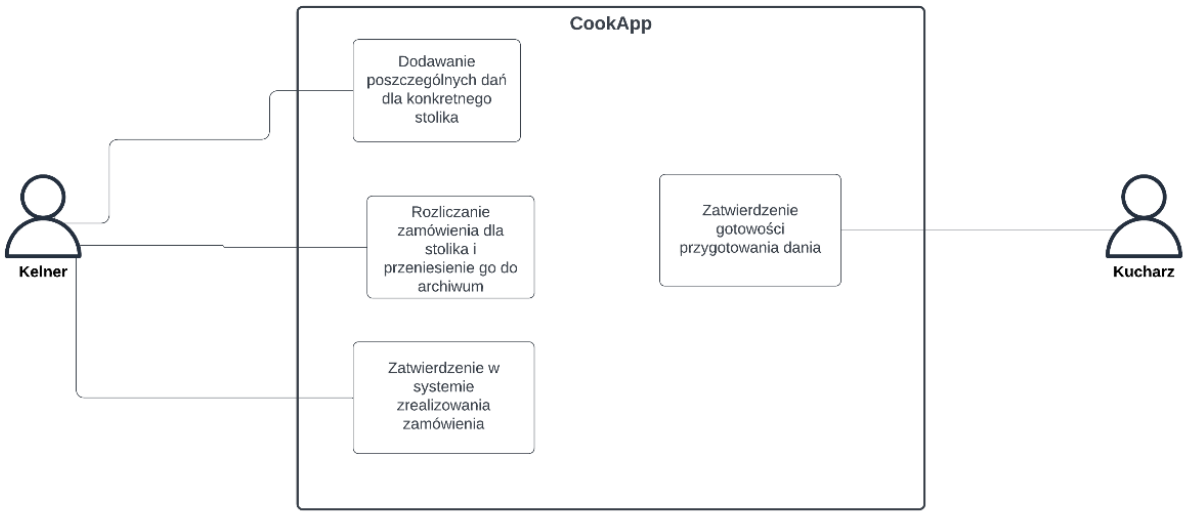
Uproszczona architektura aplikacji



Rys.2.1. Uproszczona architektura aplikacji

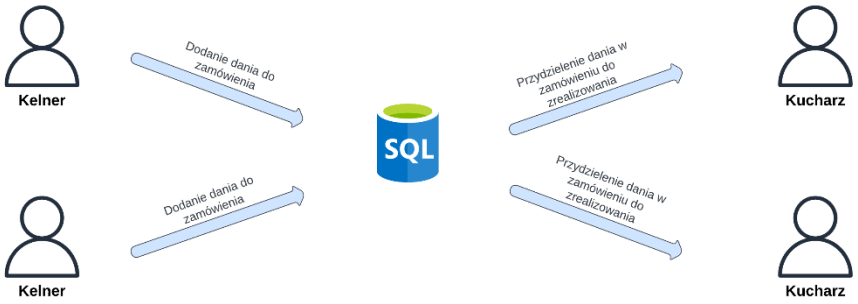
Aplikacja WPF komunikuje się z API z wykorzystaniem protokołu http. API następnie wysyła żądanie do bazy danych za pomocą Entity Frameworka i zwraca odpowiedź aplikacji. Aplikacja została utworzona używając platformy .NET.

Jak działa CookApp?



Rys.3.1. Diagram przypadków użycia

W aplikacji istnieją dwa typy kont na które się można zalogować – kelnera i kucharza. Na diagramie 3.1 zamieszczono możliwe akcje do wykonania z każdego typu konta.



Rys.3.2. Przepływ zamówienia

Kelner może dodać danie do zamówienia przy konkretnym stoliku do bazy danych. Następnie każde danie z osobna zostaje przydzielone do dostępnego oraz zalogowanego kucharza. Każdy kucharz może posiadać maksymalnie dwa dania do zrealizowania naraz. Jeśli każdy z nich już przyjął dwa, to wtedy dania zostają w bazie nieprzydzielone oczekując w „kolejce". Gdy tylko jakiś z kucharzy zatwierdzi gotowość przydzielonego mu dania, zwalnia się miejsce i natychmiast zostaje mu przydzielone następne z „kolejki", jeżeli istnieje. Po takim zatwierdzeniu, kelner który składał zamówienie otrzymuje informację o gotowości dania do zaniesienia klientowi.

Przykład:

- Kucharz 1 i 2 nie posiadają obecnie żadnych dań do realizacji.

- Kelner 1 wprowadza cztery dania do zamówienia a zaraz za nim kelner 2 wprowadza trzy.

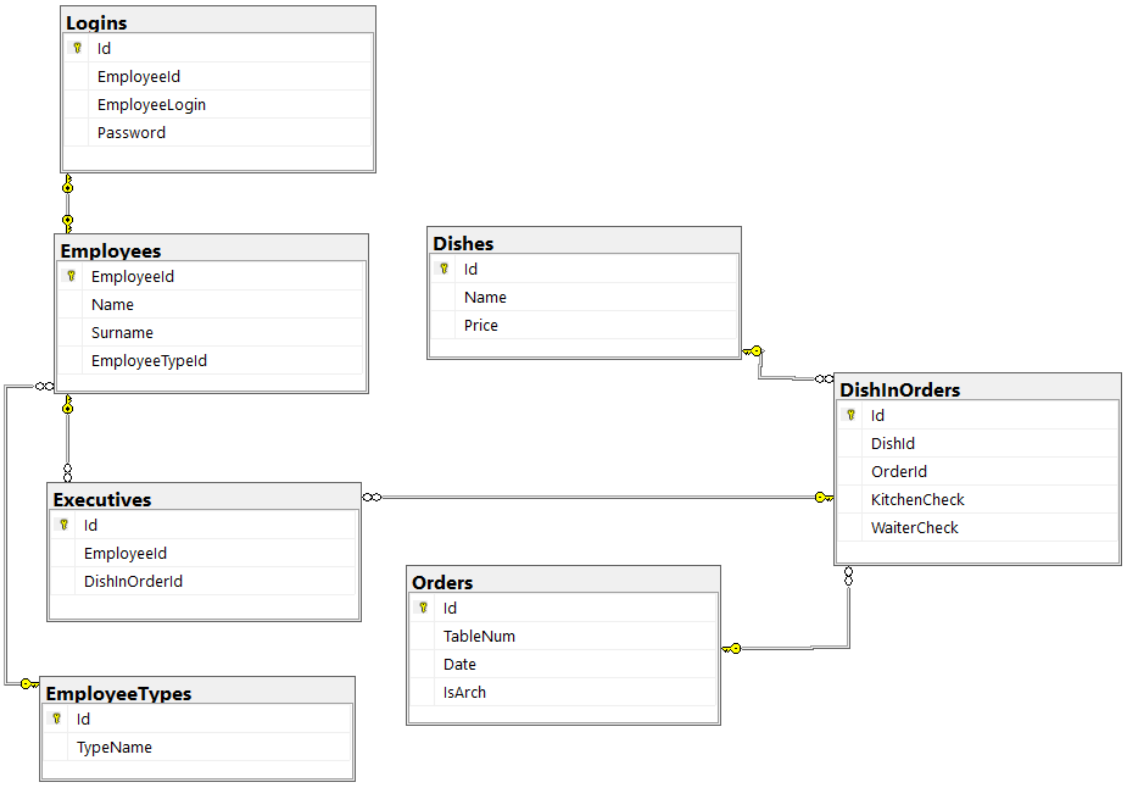
  Dwa dania kelnera 1 zostają przydzielone kucharzowi 1, a pozostałe dwa kucharzowi 2. Trzy dania złożone przez kelnera 2 oczekują na przydzielenie.

- Kucharz 2 zmienił status jednego dania na gotowe.

  Kucharz 2 od razu dostaje następne danie do realizacji z oczekujących a kelnerowi 1 gotowe danie wyświetla się na liście „do odbioru”. Uwaga, u kelnera 2 lista „do odbioru” zostaje bez zmian.

Budowa bazy danych

Baza danych została stworzona z użyciem języka SQL na lokalnym serwerze.



Rys.4.1. Budowa bazy danych

Baza danych posiada tabele widoczne na Rys.4.1. W jednym zamówieniu może znajdować się wiele dań a do jednego dania może zostać przydzielonych więcej niż jeden pracownik (konkretnie dwóch). Każde danie posiada pola KitchenCheck oraz WaiterCheck, które sygnalizują czy danie zostało przygotowane przez kucharza oraz czy zostało zaniesione przez kelnera. Każdy pracownik posiada również swój typ (kucharz lub kelner).

Komunikacja pomiędzy aplikacją i API

Komunikacja pomiędzy aplikacją WPF a API webowym odbywa się za pośrednictwem protokołu http. Wysyłane oraz odbierane dane są wysyłane w formacie JSON. Wysyłanie zapytań odbywa się z wykorzystaniem operacji asynchronicznych.

```
using (HttpClient client = new HttpClient())
{
    List<CookerDish> assignedDishes = new();
    HttpResponseMessage responseMessage = await
client.GetAsync($"{AppSettings.BaseUrl}api/DishInOrder/forCooker/{cookerId}");

    if (responseMessage.IsSuccessStatusCode)
    {
        var jsonstring = await responseMessage.Content.ReadAsStringAsync();
        assignedDishes = JsonConvert.DeserializeObject<List<CookerDish>>(jsonstring);
    }

    return assignedDishes;
}
```

Kod z panelu kucharza wysyłający do API żądanie otrzymania listy dań przypisanych obecnie zalogowanemu kucharzowi

Żądania do API

API komunikuje się z bazą danych za pomocą Entity Framework. W API do każdej tabeli jest przypisany kontroler. Lista zaimplementowanych żądań do API znajduje się w załączniku.

Budowa WPF

Aplikacja posiada osobny interfejs dla kucharza i kelnera gdzie kelner ma dostępnych więcej pod widoków. Dane zalogowanego pracownika są przechowywane w danych sesji, która tworzy się przy uruchomieniu aplikacji. Odpowiedni interfejs wyświetla się na podstawie typu logującego się pracownika.

Użyte wersje i dodatki

Aplikacja WPF:

Target framework: net7.0-windows

Dodatki:

- Microsoft.EntityFrameworkCore.Design: 7.0.13
- Microsoft.EntityFrameworkCore.SqlServer: 7.0.13
- Microsoft.EntityFrameworkCore.Tools: 7.0.13
- Microsoft.Extensions.Logging.Debug: 8.0.0

- Newtonsoft.Json: 13.0.3

API webowe:

ASP.NET Core Web API

Target framework: net7.0

Dodatki:

- Microsoft.AspNetCore.OpenApi: 7.0.13
- Microsoft.EntityFrameworkCore: 7.0.13
- Microsoft.EntityFrameworkCore.SqlServer: 7.0.13
- Microsoft.EntityFrameworkCore.SqlServer.Design: 1.1.6
- Microsoft.EntityFrameworkCore.Tools: 7.0.13
- Swashbuckle.AspNetCore: 6.5.0

Baza danych:

Microsoft SQL Server 2022 (RTM-GDR) - 16.0.1105.1 (X64)

Załączniki

- Lista zaimplementowanych żądań do API w formacie html (API_command.html)