Zaawansowane metody programowania

Smakoowa

1 Opis funkcjonalny systemu.

Projekt “Smakowa" zakłada stworzenie cross platformowej aplikacji z przepisami kulinarnymi. Po utworzeniu konta użytkownik będzie posiadać dostęp do bazy przepisów pisanych przez społeczność. Przepisy będą umieszczane na zasadzie CRUD-a. Wyszukiwanie będzie odbywało się poprzez wyszukiwarkę nazwy danego dania. Zakładamy możliwość ocenienia przepisu (like) co również jest jednoznaczne z dodaniem przepisu do podstrony użytkownika “polubione", komentarze.

1.1 Opis funkcjonalny systemu.

- Wyszukiwanie
- Tagi Przepisów
- Ocena przepisu
- Polubione przepisy
- Dodawanie przepisów CRUD

2 Streszczenie opisu technologicznego

2.1 Aplikacja internetowa - Mikołaj Śniechowski

- Vue 3 - otwartoźródłowy framework do aplikacji webowych typu front-end, oparty na języku JavaScript
- Pinia - state managment służący do przechowywania danych oraz metod z których mogą korzystać wszystkie komponenty w projekcie
- HeroIcons - zbiór podstawowych ikon do wykorzytania przy tworzeniu UI
- ESLint - narzędzie do analizy statycznej kodu JavaScript, służące wychwytywaniu błędów i niezgodności
- Prettier - narzędzie do formatowania stylu kodu, zapewniające jego czytelność
- Vercel - platforma służąca do publikacji statycznych stron internetowych,
- Github - internetowa platforma która umożliwia przechowywanie, udostępnianie i zarządzanie projektem lub kodem źródłowym z wykorzystaniem systemu kontroli wersji Git.

2.2 Aplikacja komputerowa - Oliwier Jędrzejczak

- Electron - Framework do tworzenia aplikacji desktopowych przy użyciu technologii webowych (HTML, CSS, JavaScript).
- React - Biblioteka JavaScript do tworzenia interfejsów użytkownika, która umożliwia tworzenie dynamicznych, reaktywnych i modułowych komponentów.

- Vite - Nowoczesne narzędzie do szybkiego budowania aplikacji front-end z wydajnym systemem budowania i obsługą hot-reloadingu.
- React Router - Biblioteka routingu dla aplikacji React, umożliwiająca nawigację między różnymi widokami i zarządzanie adresem URL.
- Webpack - Narzędzie do budowania aplikacji webowych, które pakuje moduły JavaScript, style CSS i inne zależności wzdłuż zależności, umożliwiając efektywne zarządzanie i optymalizację kodu.
- MaterialUI - Biblioteka komponentów UI dla Reacta, która zapewnia gotowe do użycia komponenty zgodne z Material Design.
- GitHub - Platforma hostingowa i usługa zarządzania kodem źródłowym, która umożliwia programistom współpracę, kontrolę wersji i udostępnianie projektów.
- ESLint - Narzędzie do statycznej analizy kodu JavaScript, które pomaga w utrzymaniu jednolitego stylu kodu, identyfikowaniu potencjalnych błędów i zwiększaniu jakości kodu.
- Prettier - Narzędzie do automatycznego formatowania kodu, które pomaga w zachowaniu spójnego stylu i formatu kodu w projekcie.

2.3 Aplikacja mobilna - Krzysztof Balicki

- Flutter + Dart - otwartoźródłowy zestaw narzędzi dla programistów przeznaczony do tworzenia natywnych, wieloplatformowych aplikacji mobilnych, komputerowych oraz internetowych, stworzony przez firmę Google.
- flutter_secure_storage - biblioteka umożliwiająca zapisywanie danych w bezpieczny sposób poprzez szyfrowanie danych. Działa na wielu platformach między innymi android, IOS, web.
- http - biblioteka dla języka dart umożliwiają wysyłanie i odbieranie requestów w aplikacji.
- GetX- to wyjątkowo lekkie i wydajne rozwiązanie dla Fluttera. Szybko i praktycznie łączy wydajne zarządzanie stanem, inteligentne wstrzykiwanie zależności i zarządzanie trasami. W aplikacji wykorzystane zostały głównie możliwości routingu i bezkontekstowych okien dialogowych.
- Github - internetowa platforma która umożliwia przechowywanie, udostępnianie i zarządzanie projektem lub kodem źródłowym z wykorzystaniem systemu kontroli wersji Git.

2.4 API - Bartłomiej Wołyniec

- ASP.NET Core 6.0 - framework do tworzenia aplikacji webowych, stworzony przez Microsoft, oparty na języku C#.
- Entity Framework - framework pozwalający na mapowanie relacyjne obiektów w języku programowania na struktury bazodanowe i odwrotnie.
- MSSQL - Microsoft SQL Server, czyli relacyjna bazy danych która służy do przechowywania danych i zapewnienia łatwego dostępu do nich za pomocą różnych aplikacji.
- xUnit - framework do testowania dla języka C#, umożliwia pisanie testów jednostkowych i integracyjnych.
- Github - internetowa platforma która umożliwia przechowywanie, udostępnianie i zarządzanie projektem lub kodem źródłowym z wykorzystaniem systemu kontroli wersji Git.

3 Streszczenie wykorzystanych wzorców projektowych w każdej aplikacji

3.1 Aplikacja internetowa

- Conditional Rendering - wzorzec pozwalający na wyświetlanie różnych komponentów lub elementów w zależności od spełnienia określonych warunków.
- State Management - wzorzec który umożliwia dzielenie staniu aplikacji pomiędzy wszystkimi jego komponentami.
- Observer - to wzorzec projektowy, który umożliwia obiektom reagowanie na zmiany stanu innego obiektu (zwany podmiotem) poprzez subskrybowanie jego zdarzeń.
- Lazy Loading - to technika optymalizacji wydajności ładowania stron internetowych i aplikacji. Polega na dynamicznym ładowaniu elementów strony, takich jak obrazy, skrypty JavaScript, arkusze stylów CSS i inne zasoby, tylko wtedy, gdy są potrzebne, zamiast ładować je wszystkie na raz podczas pierwszego ładowania strony.
- Lifecycle hooks - możliwość wykonywania określonych akcji w poszczególnych stanach (przed stworzeniem, po odłączeniu itp.) obiektu.

3.2 Aplikacja komputerowa

- Factory - udostępnia interfejs do tworzenia obiektów w ramach klasy bazowej, ale pozwala podklasom zmieniać typ tworzonych obiektów.
- Observer - pozwalający zdefiniować mechanizm subskrypcji w celu powiadamiania wielu obiektów o zdarzeniach dziejących się w obserwowanym obiekcie.
- Facade - wyposaża bibliotekę, framework lub inny złożony zestaw klas w uproszczony interfejs.
- Singleton - pozwala zapewnić istnienie wyłącznie jednej instancji danej klasy.
- Composite - pozwala komponować obiekty w struktury drzewiaste, a następnie traktować te struktury jakby były osobnymi obiektami.

3.3 Aplikacja mobilna

- Facade - zapewnia uproszczony interfejs do skomplikowanego systemu lub zestawu klas. Wzorzec ten opiera się na idei, że tworzona jest nowa klasa, która działa jako interfejs pomiędzy klientem a systemem. Fasada zapewnia uproszczony interfejs do wywoływania złożonych operacji, dzięki czemu klient nie musi znać szczegółów implementacyjnych systemu.
- Observer - Obserwator to behawioralny wzorzec projektowy, który pozwala zdefiniować mechanizm subskrypcji w celu powiadamiania wielu obiektów o wszelkich zdarzeniach, które przytrafiają się obiektowi, który obserwują.
- Dependency injection - umożliwia oddzielenie tworzenia obiektów od ich zależności (innych obiektów), poprzez przekazywanie tych zależności jako parametrów konstruktora lub metod obiektu zamiast tworzyć je wewnątrz. Dzięki takiemu rozwiązaniu zależności są odseparowane co ułatwia rozwijanie aplikacji.

- Builder - to kreacyjny wzorzec projektowy, który pozwala krok po kroku konstruować złożone obiekty. Wzorzec umożliwia tworzenie różnych typów i reprezentacji obiektu przy użyciu tego samego kodu konstrukcyjnego.

3.4 API

- Dependency injection - umożliwia oddzielenie tworzenia obiektów od ich zależności (innych obiektów), poprzez przekazywanie tych zależności jako parametrów konstruktora.

- Repository - umożliwia oddzielenie logiki dostępu do danych (takich jak baza danych, pliki, itp.) od pozostałej części aplikacji. Wzorzec ten opiera się na idei, że każdy rodzaj danych powinien być reprezentowany przez osobny obiekt, tzw. repozytorium (repository), który dostarcza interfejs do operacji CRUD (Create, Read, Update, Delete) na tych danych.

- Generic Repository - rozszerzenie wzorca Repository, które umożliwia łatwe tworzenie repozytoriów dla różnych rodzajów danych, używając jednego uniwersalnego interfejsu. W tym wzorcu interfejs repozytorium jest parametryzowany typem encji, co umożliwia korzystanie z jednego interfejsu dla wielu różnych typów danych.

- Chain of responsibility - umożliwia przetwarzanie danych przez szereg obiektów (handlerów), z których każdy może obsłużyć dane lub przekazać je dalej do następnego handlera. Wzorzec ten opiera się na idei, że każdy handler odpowiada tylko za pewien aspekt przetwarzania danych i może zignorować lub obsłużyć dane w sposób, który mu odpowiada.

- Template method - umożliwia definiowanie szablonu algorytmu w postaci klasy abstrakcyjnej, a następnie pozostawienie szczegółów implementacyjnych dla klas dziedziczących. Wzorzec ten opiera się na idei, że pewne etapy algorytmu są stałe, ale inne mogą się różnić w zależności od konkretnego przypadku użycia.

- Singleton - wzorzec zapewnia, że tylko jeden obiekt danej klasy może istnieć w całej aplikacji.

- Facade - zapewnia uproszczony interfejs do skomplikowanego systemu lub zestawu klas. Wzorzec ten opiera się na idei, że tworzona jest nowa klasa, która działa jako interfejs pomiędzy klientem a systemem. Fasada zapewnia uproszczony interfejs do wywoływania złożonych operacji, dzięki czemu klient nie musi znać szczegółów implementacyjnych systemu.

4 Instrukcja lokalnego i zdalnego uruchomienia testów oraz systemu

4.1 Aplikacja internetowa

Uruchomienie systemu
Lokalnie

- Pobierz Visual Studio Code, Git oraz Node,js.

- Sklonuj repozytorium `https://github.com/SmaqwaTeam/SmakoowaWebApp`.

- W katalogu głównym uruchom terminal oraz komendę ‘npm install’.

- Następnie uruchom komendę ‘npm run dev’.
- Aplikacja jest gotowa do użytku.

Zdalnie

- Aplikacja jest dostępna pod adresem https://smakoowa-web-app.vercel.app/.

Uruchomienie testów
Lokalnie

- Przygotuj środowisko jak napisano wyżej.
- W katalogu projektu uruchom terminal i wpisz ‘npx cypress open’.
- W oknie przeglądarki wybierz testy jakie chcesz przeprowadzić.

4.2 Aplikacja komputerowa

Uruchomienie testów
Lokalnie

- Należy zainstalować Node.js ze strony `https://nodejs.org`.
- W terminalu naszego edytora: git clone -branch main https://github.com/SmaqwaTeam/Smakoowa-Desktop.
- ZaInstaluj zależności oraz “jest"-a: npm install; npm install –save-dev jest.
- Uruchom testy, wpisując w terminalu npm test.

Uruchomienie systemu
Lokalnie poprzez vsc

- Należy zainstalować Node.js ze strony `https://nodejs.org`.
- W terminalu naszego edytora: git clone -branch main `https://github.com/SmaqwaTeam/Smakoowa-Desktop`.
- Instalacja zależności: npm install.
- Uruchomienie: npm run dev.

Uruchomienie systemu
lokalnie poprzez plik.exe

- Pobieramy zip z github-a build_app.
- Wypakowywujemy i uruchamiamy plik.exe.

4.3 Aplikacja mobilna

Uruchomienie systemu
Lokalnie

- Pobierz Visual Studio Code lub Android Studio
- Po instalacji zainstaluj dodatki flutter i dart.
- Sklonuj repozytorium `https://github.com/SmaqwaTeam/Smakoowa_Mobile.git`
- Uruchom emulator systemu android i połącz go z edytorem.
- W katalogu projektu uruchom konsole i wpisz "flutter get pub"
- Następnie flutter run.
- Aplikacja jest gotowa do użytku.

Uruchomienie testów
Lokalnie

- Przygotuj środowisko jak napisano wyżej.
- W katalogu projektu uruchom konsole i wpisz "flutter test"

4.4 API

Uruchomienie systemu
Lokalnie

- Pobierz Visual Studio 2022 Community
- Przy instalacji zaznacz "Opracowywanie zawartości dla platformy ASP.NET" oraz "Magazynowanie i przetwarzanie danych"
- Sklonuj repozytorium `https://github.com/SmaqwaTeam/Smakoowa-Api.git`
- Przejdź do Packet Manager Console, wpisz w konsoli polecenia:
- update-database -context DataContext
- update-database -context BackgroundDataContext
- Uruchom aplikację klawiszem F5
- Api jest gotowe do użytku, możliwe jest jego testowanie za pomocą Swagger: `https://localhost:7188/swagger/index.html`

Zdalnie
Api jest dostępne pod adresem: `https://smakoowaapi.azurewebsites.net/swagger/index.html`

Uruchomienie testów
Lokalnie

- Pobierz, zainstaluj Visual Studio i sklonuj repozytorium tak jak zostało to opisane w lokalnym uruchomieniu systemu
- Uruchom Test Explorer
- Skrótem klawiszowym Ctrl+R, T uruchom testy

Zdalnie

- Przejdź do `https://github.com/SmaqwaTeam/Smakoowa-Api/actions`
- Otwórz workflow “Integration Tests"

5 Wnioski projektowe

W ramach projektu grupowego udało się stworzyć crossplatformową aplikację na komputer, przeglądarkę oraz telefon z przepisami kulinarnymi. W trakcie realizacji projektu udało się zrealizować część założonych celi i wykorzystać kilka wzorców projektowych w.w. w Punkcie 3. Jednym z głównych celów projektu było stworzenie łatwej w obsłudze aplikacji, która pozwoli użytkownikom na szybkie wyszukiwanie i przeglądanie przepisów kulinarnych. W tym celu zdecydowano się na stworzenie intuicyjnego interfejsu użytkownika, który pozwala na szybkie poruszanie się po aplikacji oraz wyszukiwanie interesujących przepisów. Kolejnym celem projektu było stworzenie aplikacji, która będzie responsywna i dostosowana do różnych urządzeń.