Wzorzec MVC w tworzeniu aplikacji internetowych

Laboratorium 4

Ćwiczenia – Przygotowanie

Ćwiczenie przewiduje **cztery** (**4**) zadania. Utwórz nowy projekt w swoim **edytorze kodu** lub **IDE**, w którym utworzysz odpowiednio **cztery katalogi**:

- zad1,
- zad2,
- zadX,
- (…),
- zad4.

Gdzie **X** to numer porządkowy, każdego następnego zadania od **pierwszego** do **następnego**, aż do ostatniego. Zrobienie tej czynności na początku, ułatwi Ci zarządzanie projektem podczas zajęć, więc warto byś ten krok wykonał na starcie. Katalogi przeznaczone są na zawartość w postaci **plików wymaganych** do uruchomienia **konkretnego zadania**. Utworzone pliki, muszą mieć odpowiedni format. W zależności od zadania, będzie to plik we formacie **.js**, **.html**, **.ejs** lub też **.json**.

PRZYPOMNIENIE: Zadania są rozwiązywane **w podanej kolejności** (czyli najpierw **zadanie 1** a następnie **zadanie 2** itd.). Każde następne zadanie powinno wykorzystywać **kod z poprzedniego zadania** jako kod początkowy.

Odpowiednie **zadanie** (czyli zadanie zawarte w **reprezentującym je katalogu** np. **zad1**) zawiera **jedynie** taką ilość kodu, jaka jest potrzebna na rozwiązanie danego podpunktu.

Oddawanie zadań, które niosą znamiona np. rozwiązania **następnego punktu** w niewłaściwym zadaniu, może rzutować na ocenę końcową. Dodatkowo, oddawanie zadań:

- **kompletnych** lub tzw. **komplementarnych** (zawierających rozwiązanie na **wszystkie podpunkty**) jako np. jedynie rozwiązanie,
- oddanie zadania w sposób inny niż jako **link do repozytorium** (za pomocą **zadań teams** – dopuszczalne jest dołączenie ewentualnie **dokumentu tekstowego** zawierającego **link do repozytorium**),
- **po terminie**,
- niezgodnego z **zaproponowaną strukturą** (czyli innej niż z **podziałem zadań z ćwiczenia na katalogi** lub **podziałem zadań z ćwiczenia na branche**),

Będzie skutkowało otrzymaniem oceny **2.0.**

Zadanie 1

Utwórz nowy projekt **Node.JS** (dokładnie **Express.JS**) i zainicjalizuj go (wygeneruj plik **package.json**) a następnie zainstaluj zależności:

- **express**,
- **ejs**.

Następnie:

- utwórz serwer **express** nasłuchujący na **porcie 3000**, który po uruchomieniu wyświetla w swoim **listenListener** za pomocą **console.log** treść „**Server is running on ${{PORT}}**”,
- ustaw **silnik szablonów** na obsługę **ejs**,
- ustaw **parsowanie danych** z formularza za pomocą **body-parser**,
- utwórz prosty routing dla:
    - **GET** (ścieżka: „/”) – renderujący widok **Home.ejs**,
    - **GET** (ścieżka: „/success”) – renderujący widok **Success.ejs**,
    - **GET** (ścieżka: „/students-list”) – renderujący widok **List.ejs**,
    - **POST** (ścieżka: „/add-student) – przekierowujący na widok **AddStudent.ejs**.

Utwórz katalog **views** oraz pliki reprezentujące widoki **Home, Success** oraz **List**, wyświetlające w swoim **title** oraz w **h1** nazwę swojego pliku (**Home**, **Success** lub **List**).

Zadanie 2

Utwórz katalog **controllers** a w nim plik **students.js** oraz **error.js**, a następnie:

- dodaj widok **NotFound** wyświetlający w **title – Not found** oraz w **h1** o treści **„Page doesn’t exist”** oraz **link (a)** przekierowujący do widoku **Home**,
- dodaj obsługujący przypadek **nie znalezienia strony**, **routing**,
- umieść logikę związaną z renderowaniem **NotFound** w kontrolerze **NotFound** pod nazwą **getNotFoundPage**,
- zmodyfikuj widok **Home** tak by wyświetlał w **title** oraz **h1** treść **Add new student**, dodaj formularz wykonujący działanie dla ścieżki **add-student** i metody **POST** zawierający trzy **inputy** i wartości atrybutu **name** – **fullName, code** oraz **fieldOfStudies**,
- umieść logikę związaną z renderowaniem **Home** w kontrolerze **students** pod nazwą **getAddNewStudentPage**.

Zadanie 3

Utwórz w kontrolerze **students** tymczasową tablicę **students**, która będzie przetrzymywała nowo dodanych studentów oraz zmienną **nextId** zainicjalizowaną wartością **1** a następnie:

- dodaj w tym samym kontrolerze obsługę **route /add-student** dla metody **POST** dodającą nowego studenta do wspomnianej tablicy nadając mu dodatkowo pole **id** o wartości **nextId**,
- po każdym dodaniu nowego studenta, zwiększ wartość **nextId** o **jeden**,
- następnie, przekieruj użytkownika na widok **Success**,
- zmodyfikuj widok **Success** tak by wyświetlał w **title** oraz **h1** treść **Success**, dodaj **p** o treści **„Student has been added with success!”** oraz **linki (a)** przekierowujące do widoku **Home** oraz **List**,
- umieść logikę związaną z renderowaniem **Success** oraz **List** w kontrolerze **students** pod nazwami **getAddingNewStudentSuccessPage** oraz **getStudentsListPage**.

Zadanie 4

Uzupełnij widok **List** tak, by wyświetlał listę (**ul** oraz **li**), gdzie każdy **li**, wyświetla:

- w pierwszym **p** - **student fullName**,
- w drugim **p – student code,**
- w trzecim **p – student fieldOfStudies.**

Stwórz we **views** katalog **partials** i stwórz **navigation.ejs**, który będzie zawierał nawigację pozwalającą przejść do widoków **Home, List** oraz **AddStudent**. Umieść tego **partial** w każdym widoku. Źródłem danych dla widoku **List** ma być tablica **students**.

Przekaż **title** wszystkich podstron do widoków jako **pageTitle** i wyświetl dynamicznie. Wydziel kod w **head** do katalogu **partials** pod nazwą **head.ejs.**