Sprawozdanie 1 z Web 2.0 – wersja z samym API

MEAN Stack po stronie serwera (back-end)

W pierwszym sprawozdaniu oczekuję od Państwa napisania aplikacji wykazującej znajomość części technologii MEAN Stack, tej działającej po stronie serwera, czyli Node.js, Express.js i MongoDB (choć można zrobić też aplikację działającą na innej bazie, jeśli ktoś potrafi). Zależnie od liczby wykorzystanych elementów konstrukcyjnych będzie odpowiednia ocena.

Sprawozdania oddajemy na platformie moodle.pcz.pl w naszym kursie

Oddajemy dwa pliki:

- Nazwisko.pdf - sprawozdanie w pliku PDF, opis tego co zrobiliśmy
- Nazwisko.zip - spakowany katalog z projektem w formie archiwum, format zip albo inny, uwagi:
  - proszę **nie pakować katalogu node_modules**! dużo zajmuje, a nie jest potrzebny
  - oddawany plik może mieć do 20 MB – na taką aplikację powinno wystarczyć
  - w archiwum proszę załączyć dodatkowe pliki, jeśli mogą one być mi potrzebne do sprawdzenia aplikacji, np. zawartość bazy danych, jeśli ma ona działać u mnie lokalnie, jakieś skrypty, jakieś pliki konfiguracji… ja potrzebuje uruchomić aplikację i sprawdzić jak działa i co w niej jest
  - w pliku z aplikacją **powinien być plik package.json** ze wszystkimi pakietami, których wasza aplikacja używa, czyli jak coś instalujecie to z opcją `--save`, po rozpakowaniu pakietu i poleceniu `npm install` powinienem mieć aplikację gotową do uruchomienia

Sprawozdanie można oddawać:

- do **26 kwietnia 2024 do północy** (piątek), liczy się data i czas na platformie e-learning), to są ponad 4 tygodnie po skończeniu zajęć dotyczących Express.js

W sprawozdaniach oddanych po terminie będę obniżał o 0.05 oceny za każdy 1 dzień spóźnienia, ale nie więcej niż o 1 całą ocenę. Jeśli ktoś się spóźni z oddaniem 3 tygodnie, to już nie musi się spieszyć, bo pośpiech nic już nie zmieni. Ciągle można mieć 5.0 z pierwszego sprawozdania, ale trzeba zrobić zadania na przynajmniej 6.0 punktów.

Każdy student ma na początku ocenę (liczbę punktów) z tego sprawozdania 1.0. Za zaimplementowanie w aplikacji i wykazanie w sprawozdaniu każdego z poniższych elementów są dodatkowe punkty do oceny ze sprawozdania. Maksymalnie można uzyskać 6.3, ale to by wymagało zrobienia absolutnie wszystkich zadań. Większość zadań zwiększa ocenę o 0.5 – trzeba więc zrobić przynajmniej 4 zadania na zaliczenie tego, ale proponował bym celować minimum w 3.5, bo za opis PDF zwykle nie daję maksymalnej liczby punktów.

W sprawozdaniu (w pliku PDF z opisem sprawozdania) proszę **wyraźnie zapisać, które elementy z poniższej listy zostały zrealizowane**, żebym wiedział co oceniać. Można skorzystać z moich aplikacji udostępnianych do pracy na laboratoriach, ale i tak będzie trzeba je przerobić, wzbogacić, dostosować do swoich potrzeb.

To sprawozdanie **będzie miało 55% wpływu na ocenę końcową z laboratoriów**. Sprawozdanie 2 będzie miało też 45% wpływu na ocenę końcową.

Do utworzonego w sprawozdaniu API, jednego lub więcej, **TRZEBA dołączyć plik z wyeksportowanym API z programu Postman (format JSON)**. **Bez tego pliku nie będę sprawdzał zadania, albo nie dam punktów albo odeślę do poprawy**.

To jest wersja, w której backend udostępnia same końcówki, czyli API do prezentowania danych, zarządzania danymi i użytkownikami. Myślę, że mogę tutaj uwzględnić do oceny 4 różne rodzaje API, za każdy rodzaj można zdobyć osobne punkty:

1. API udostępniające dane do przeglądania ewentualnie jakiejś modyfikacji, ale dostępne dla wszystkich bez rejestracji, bez uwierzytelnienia.
2. API dla osób posiadających dostęp do danych zabezpieczonych, chyba najlepiej wykorzystać do tego token. To API powinno mieć końcówki udostępniające dane, ale również końcówki do modyfikacji czy usuwania danych (pełna obsługa CRUD).
3. API pozwalające na rejestrację i opcjonalnie aktywację konta (po adresie e-mail albo numerze telefonu i kodzie SMS). Dodatkowo można tutaj dodać też opcję zmiany hasła.
4. API pozwalające na zarządzanie użytkownikami, takie API dla admina (wymaga specjalnego przydzielenia niektórym użytkownikom praw administratorów.

Zad.	Punkty	Funkcjonalność w aplikacji
1	± 0.5	**Opis sprawozdania w pliku PDF - obowiązkowe** Zależnie od jakości tego opisu, można uzyskać z tego zadania między +0.5, a -0.5 (jak jest zrobione bardzo źle). Zwykle studenci mają tutaj +0.2 czy +0.3.
2	+ 0.5	**API nr 1** **Przygotowanie prostego REST API, np. za pomocą pakietu mongoose, które pozwoli na dostęp do danych bez uwierzytelnienia** Można użyć pakietu mongoose, żeby stworzyć jakieś proste API, z którego można skorzystać za pośrednictwem odpowiedniej grupy adresów URL i metod protokołu HTTP. Można też wykorzystać inną bibliotekę, a nawet inną bazę. Całe API powinno być opisane i dostępne, np. przez narzędzie Postman. Korzystamy z formatu JSON. Adresy URL wykorzystane w tym API nie mogą się oczywiście pokrywać z tymi udostępniającymi resztę funkcjonalności aplikacji, np. z API służącym do rejestracji czy logowania. Powinno być odczytywanie danych (metoda GET), ewentualnie można dodać coś do modyfikacji danych, ale pamiętając, że to wszystko dla użytkowników anonimowych, bez uwierzytelnienia, bez tokenu… Modyfikację danych chyba lepiej zostawić sobie na zadanie 3 (API nr 2).
3	+ 0.5	**API nr 2** **Dodanie API pozwalającego na przeglądanie i modyfikację danych w bazie (pełna obsługa CRUD) ale tylko dla uwierzytelnionych użytkowników** Należy tutaj dodać końcówki, które pozwolą na przeglądanie i modyfikację danych w bazie tym użytkownikom, którzy są uwierzytelnieni czyi mają odpowiedni token. Wystarczy jak API będzie działać na danych jednej kolekcji. Musi być to inna kolekcja, niż ta związana z logowaniem. Można do tego wykorzystać bibliotekę mongoose i jakiś utworzony specjalnie schemat i model (może być ten sam co w zadaniu 2). Powinno dać się odczytać dane (GET), dodać dane (POST), zmodyfikować dane (PUT), usunąć dane (DELETE). Ma być tutaj pełna obsługa CRUD,

4	+ 0.5	**Wykorzystanie JSON Web Token** Należy zabezpieczyć dostęp do API utworzonego w zadaniu 3 za pomocą tokenu, który jest generowany dla zalogowanego użytkownika, podobna funkcjonalność, jak ta z laboratorium 1. Wykazać działanie tej funkcjonalności można za pomocą narzędzia do obsługi REST lub innej aplikacji, która z takiego API skorzysta. Taki token najlepiej przesłać w nagłówku żądania HTTP, ale można też w adresie URL żądania. To zadanie wymaga zrobienia zadania 3, które pozwala na zaprezentowanie działania tokenów.
5	+ 0.3	**API nr 3** **Rejestracja użytkownika** Należy dodać API pozwalające na rejestrację użytkowników, która to rejestracja utworzy im konta, które po zalogowaniu udostępnią użytkownikowi token pozwalający na dostęp do zabezpieczonych API, np. edycji danych w aplikacji, albo nawet do zarządzania użytkownikami z panelu admina.
6	+ 0.3	**API nr 3** **Zmiana hasła zalogowanego użytkownika, odzyskanie hasła** Można dodać API, które pozwoli na zmianę hasła zarejestrowanego użytkownika. Nowe hasło należałoby przyjmować dwa razy i sprawdzać, czy oba wprowadzone hasła są takie same.
7	+ 0.7	**API nr 4** **API mające funkcjonalność prostego panelu administratora** Trzeba tutaj przygotować API, które będzie dostępne tylko dla administratora i będzie pozwalać na zarządzanie użytkownikami, np. nadawanie uprawnień, dodawanie i usuwanie użytkowników, blokada konta. Czego tutaj oczekuję? • Do kolekcji użytkowników trzeba dodać pole określające uprawnienia użytkownika, przynajmniej typu logicznego, jakoś trzeba określić, którzy użytkownicy mają prawa administratora. • Trzeba dodać warstwę pośrednią, która pozwoli na dostęp do API mającego funkcjonalność panelu administratora tylko dla użytkowników, którzy mają uprawnienie administratora, chyba najprościej użyć do tego tokeny. • Trzeba utworzyć osobny moduł routingu obsługujący wszystkie trasy związane z panelem administratora. Warstwę pośrednią zabezpieczającą tę funkcjonalność można dodać od razu w pliku `app.js`. • Trzeba dodać końcówkę do tworzenia użytkowników, wystarczy nazwa i tymczasowe hasło. Zakładamy, że jest w aplikacji opcja odzyskania/zmiany hasła przez użytkowników. • Trzeba dodać końcówkę do usuwania użytkowników z bazy danych. • Trzeba dodać końcówkę do zmiany uprawnień użytkowników. Może być np. przełączanie uprawnienie administratora, z `true` na `false` i odwrotnie.

8	+ 0.5	**API nr 3** **Rejestracja z potwierdzeniem aktywacji konta przez e-mail albo przez kod w wiadomości SMS** Trzeba zmodyfikować, rozszerzyć funkcjonalność związaną z API do rejestracji, w taki sposób, aby rejestracja dodawała osobę z nieaktywnym kontem, które trzeba dopiero aktywować, np. za pomocą linku aktywacyjnego w wiadomości e-mail czy za pomocą kodu przesłanego w wiadomości SMS. Podobna funkcjonalność, jaka była na laboratorium. Proponuję użyć biblioteki Nodemailer do wiadomości e-mail albo twilio do wiadomości SMS.
9	+ 0.5	**Aplikacja działająca online** Podobnie jak w poprzednim semestrze zrobienie aplikacji działającej online podnosi ocenę o pół. Można skorzystać np. z darmowego konta na cyclic.sh i MongoDB Atlas. Ma to być jednak samodzielnie zaimplementowane API działające gdzieś online. Nie można użyć jakieś usługi, w której konfigurujemy sobie API, jakie ta usługa ma nam udostępnić.
10	+ 0.5	**Uwierzytelnienie za pomocą konta Google czy innego portalu** Takie uwierzytelnienie można wykorzystać do zalogowania się, aby użytkownikowi zwrócić token, albo od razu skorzystać z konta Google, które zwróci token wykorzystany do dostępu do zasobów chronionych naszej aplikacji. Jeśli zamiast korzystać z uwierzytelnienia dostępnego w baseApp (logowanie za pomocą utworzonego w rejestracji loginu i hasła), wykorzystamy samodzielne uwierzytelnienie przygotowane w laboratorium 5, np. za pomocą konta Google, to podnosi to ocenę o 0.5. **Uwaga** Uwierzytelnienie tylko za pomocą zewnętrznego dostawcy, np. Google nie pozwoli Państwu zrobić zadania 5 i 6, a dodatkowo może też utrudnić zadanie 7 i 8. Jeśli więc chcecie zrobić w sprawozdaniu również to zadanie, to miejcie świadomość dodatkowych trudności. Można wykorzystać w aplikacji dwa rodzaje uwierzytelnienia, za pomocą passport-local (baseApp) i to dodatkowe z tego zadania. Na pewno się da, ale nie koniecznie musi być to super łatwe do zrobienia. Jak ktoś zrobi w aplikacji dwa rodzaje uwierzytelnienia i będzie to dobrze działać, to mogę dać za to bonus +0.5 do oceny czyli w sumie 0.3+0.5+0.5=1.3 punktu do oceny za działające dwa rodzaje uwierzytelnienia.

11	+ 0.5	**Porównanie wydajności Node.js z Deno albo z Bun (dwa różne silniki)** Jeśli komuś z Państwa uda się jakoś sensownie porównać wydajność tej samej aplikacji napisanej w Node.js i w Deno albo Bun, przedstawić i opisać wyniki porównania tej to dam też +0.5 do oceny. Zadanie ciekawe, ale niekoniecznie łatwe, bo trzeba dobrze obciążyć aplikację napisaną na dwóch różnych serwerach, dwóch różnych silnikach i pomierzyć liczbę odpowiedzi w jednostce czasu.
12	+ 0.5	**Coś od siebie** Można dodać coś fajnego od siebie, jakąś bibliotekę, która ma ciekawe działanie, może jakąś funkcjonalność bardziej rozbudowaną dla użytkowników… Pozostawiam to Państwa wiedzy, pomysłowości i doświadczeniu w technologiach internetowych. Powinno to być jednak **coś niezwiązanego z działaniem po stornie klienta**, bo takie rzeczy będą w sprawozdaniu 2. Można jednak za taką funkcjonalność dostać tylko jedno + 0.5. Można więc aplikację bardzo wzbogacić, ale dam za takie dodatki i tak tylko + 0.5. Przykładowe dodatki, które można by uznać na +0.5 do oceny: • API pozwalające na wysyłanie maili do jednego lub więcej odbiorców, • można pokazać na przykładzie kilku stron (adresów URL) wykorzystanie zagnieżdżonego routingu; w pliku app.js mamy dołączony moduł routingu dotyczący grupy adresów URL, w jednym z takich modułów należałoby dołączyć kolejny moduł routingu (drugi poziom zagnieżdżenia) z uszczegóławiającymi informacjami. Przykład dla krajów i miast: ◦ /kraj – główna trasa, zdefiniowana w app.js, której obsługa jest w modułu routingu w kraj.js ▪ /kraj/Polska – pierwszy moduł tras, podrzędny do kraj.js, zdefiniowany w polska.js z definicjami adresów URL konkretnych miast: • kraj/Polska/Warszawa • kraj/Polska/Krakow • … ▪ /kraj/Ukraina – drugi moduł tras, podrzędny do kraj.js zdefiniowany w ukraina.js • /kraj/Ukraina/Kijow • /kraj/Ukraina/Lwow • …

To wszystko :)