AI1 2024 – projekt zaliczeniowy

Realizacja:

• projekt jest realizowany indywidualnie, w ramach jednej grupy laboratoryjnej temat projektu nie może się powtarzać,

• wybór podejścia do tworzenia aplikacji:

– MVC z *silnikiem szablonów*: framework *Laravel 11.x*, silnik *Blade*,

– REST API + SPA: *Laravel 11.x* jako *backend API* + wybór technologii *frontendowej* do wykonania *SPA* (np. *Angular 16*/*React*/*Vue.js*), format wymiany danych: *JSON*,

• wybór relacyjnej bazy danych (*MySQL*/*PostgreSQL*/*Oracle*),

• tematem projektu jest wykonanie części *backendowej* i *frontendowej* aplikacji internetowej do obsługi małego przedsiębiorstwa lub działalności hobbystycznej z której mogą korzystać zarejestrowani, zalogowani użytkownicy (korzystający oraz modyfikujący zasoby) oraz nad której całością zarządza administrator/administratorzy,

• projekt ma być realizowany zgodnie z wyznaczonymi etapami, przedstawionymi w pliku „*AI1 2024 realizacja projektu.pdf*", zalecane jest konsultowanie postępów projektu,

• do projektu ma być napisana dokumentacja – wytyczne przedstawione w pliku „*AI1 2024 dokumentacja projektu.pdf*",

• terminy wykonania projektu – przedstawione w pliku „*AI1 2024 kalendarz przedmiotu.pdf*",

• tworzony projekt ma spełniać następujące wymagania (w zależności od oceny):

Wymaganie	3.0	3.5	4.0	4.5	5.0
Laravel >= 11.x	✓	✓	✓	✓	✓
minimum dwie, powiązane ze sobą tabele	✓	✗	✗	✗	✗
minimum trzy, powiązane ze sobą tabele	✗	✓	✓	✗	✗
minimum cztery, powiązane ze sobą tabele	✗	✗	✗	✓	✓
jedna z tabel jest tabelą transakcji/rezerwacji/wypożyczeń itp.	✗	✗	✓	✓	✓
występują więzy integralności encji	✗	✓	✓	✓	✓
seed bazy przykładowymi danymi	✗	✓	✓	✓	✓
utworzenie użytkownika/ów pełniących rolę administratora, zarządzającego zasobami aplikacji	✓	✓	✓	✓	✓
administratorzy zarządzają użytkownikami	✗	✗	✓	✓	✓
możliwość przeglądania ogólnie dostępnych zasobów	✓	✓	✓	✓	✓
paginacja najliczniej występujących zasobów	✗	✗	✗	✓	✓
brak użytkowników aplikacji (np. klienci, zawodnicy)	✓	✓	✗	✗	✗
użytkownicy aplikacji, mogą się zalogować, zarządzać swoimi zasobami, zarządzać swoimi danymi	✗	✗	✓	✓	✓
nowi użytkownicy mogą się samodzielnie zarejestrować	✗	✗	✗	✓	✓
w aplikacji dostępne są funkcjonalności typu CRUD, przeprowadzany jest pełny CRUD ogólnie dostępnego zasobu	✓	✓	✓	✓	✓
występują obrazki zasobów oraz użytkownik nimi zarządza	✗	✗	✓	✓	✓
w aplikacji dostępne są funkcjonalności oprogramowane logiką biznesową	✗	✗	✗	✓	✓
nie występuje problem N+1	✗	✗	✗	✗	✓
walidacja danych po stronie backendu	✓	✓	✓	✓	✓

walidacja danych po stronie frontendu	✗	✗	✓	✓	✓
aplikacja posiada schludny responsywny GUI, spójny dla całej aplikacji	✓	✓	✓	✓	✓
GUI aplikacji zawiera inne kontrolki interfejsu (nie tylko przyciski, pola tekstowe) np. slidery, pickery, toggle…	✗	✗	✗	✓	✓
GUI aplikacji prezentuje wyliczone informacje np. statystyki, wykresy	✗	✗	✓	✓	✓
dla kodów błędów HTTP zwracane są widoki z informacją o błędzie	✗	✗	✓	✓	✓
przygotowanie skryptu startowego „*stawiającego*" bazę i aplikację	✗	✗	✓	✓	✓

✓ – wymóg, ✗ – brak wymogu

Uwagi do wymagań (ocena końcowa może być obniżona w przypadku ich niespełnienia):

• framework'iem wykorzystanym do tworzenia projektu jest *Laravel 11.x*, należy zaznajomić się dokładnie z dokumentacją, jakie elementy, funkcjonalności i mechanizmy są wbudowane, dostępne, tak aby nie programować czegoś od zera np. uwierzytelnianie użytkowników, bramki, walidacja itp.,

• można skorzystać także z starter kitów Laravel'a,

• w celu zachowania poprawnej struktury projektu, nazewnictwa (nazwy zmiennych, klas itp. obowiązkowo po angielsku) z dobrymi praktykami itp. należy zapoznać się z:

https://webdevetc.com/blog/laravel-naming-conventions/
https://github.com/alexeymezenin/laravel-best-practices
https://github.com/alexeymezenin/laravel-best-practices/blob/master/polish.md

• w przypadku komunikatów zwracanych do użytkownika (np. błędy walidacji lub informacje po wystąpieniu wyjątku) należy zadbać o jednolitość języka; jeśli interfejs użytkownika jest w języku angielskim to komunikaty mają być również tym języku, jeśli w polskim to należy dokonać ich tłumaczenia,

• w celu wykonania seed'u bazy przykładowymi danymi w większej ilości, można wykorzystać bibliotekę *Faker*,

• do liczby wymaganych tabel nie wlicza się tabel łączących oraz tabel wygenerowanych na potrzeby framework'a np. tabeli odnoszącej się do migracji,

• nie można w ogóle używać procedur składowanych,

• w projekcie nie można budować instrukcji SQL w postaci surowej, z konkatenacją treści zapytania z wartościami parametrów pochodzących od użytkownika,

• w widokach nie wolno wykonywać żadnych operacji z logiką biznesową oraz bazą danych,

• w projektach zawierających logikę biznesową mają być wyjaśnione jej zasady dla użytkownika np. na stronie głównej/podstronie składania zamówienia wyjaśnienie co i w jakim stopniu użytkownik zyskuje/traci,

• odnośnie obrazków występujących na podstronach aplikacji, mają mieć zachowaną oryginalną proporcję, tzn. żadnych zniekształceń wynikających z rozciągnięcia,

• w projektach nie można używać:

– https://github.com/LaravelDaily/Laravel-CoreUI-AdminPanel oraz podobne,

– https://jquery.com/

Uwaga! Zawartość pliku może ulec aktualizacji.

Aktualizacja dokumentu: 04.03.2024