Podstawy Informatyki i Programowania - Projekt **103A-INxxx-ISP-PIPR**		
Projekt zaliczeniowy: **TravelNow** **Aplikacja webowa do reklamowania i rezerwacji noclegów**	Jakub Ciarka	
Prowadzący:	**Data:**	**Ocena:**

Spis treści

1 Temat projektu

Celem projektu jest przygotowanie aplikacji WEB umożliwiającej reklamowanie i rezerwację noclegów.

2 Projekt aplikacji

2.1 Założenia projektowe

- Backend aplikacji należy przygotować w języku JAVA. Wybór technologii frontonowych jest dowolny. Podobnie nie postawiono wymagań odnośnie wybóru typu serwera hostującego aplikację jest,

- Aplikacja powinna korzystać z dedykowanej bazy danych, przechowującej potrzebne informacje. Do realizacja aplikacji powinna zostać wykorzystana baza danych Oracle. W ramach projektu możliwe jest skorzystanie z bazy danych hostowanej na serwerze uczelni,

- Aplikacja powinna udostępniać przynajmniej następujące funkcjonalności:
  - Przeglądanie ofert usług hotelarskich zamieszczonych przez właścicieli noclegów (możliwe filtrowanie po lokalizacji, przedziale cenowym; możliwość sortowania po przedziale, cenowym opiniach, popularności),
  - Sprawdzanie dostępności pokoi w wybranym hotelu w zadanym przedziale czasu,
  - Składanie rezerwacji noclegów przez osoby zalogowane jako użytkownicy, zarządzanie rezerwacją (przeglądanie rezerwacji, usuwanie rezerwacji, zmiana terminu rezerwacji),
  - Dodawanie opinii i komentarzy o noclegach,
  - Dodawanie lub edycja profili hotelarskich przez osoby zalogowane jako właściciele hotelów.

- Należy zadbać o odpowiedną ergonomię aplikacji. Obsługa aplikacji powinna być prosta i przystępna dla użytkownika,

- Należy zadbać o odpowiedną staranność kodu, potwierdzoną dokumentacją z walidacji. Należy korzystać z dobrych praktyk programowania i wzorców projektowych.

2.2 Wykorzystywane technologie

Po wstępnej analizie wymagań i założeń projektowych zdecydowano się na wykorzystanie następujących technologii:

- **Serwer**: TOMCAT,

- **Backend** – został przygotowany w języku java w wersji 11 w oparciu o framework Spring Boot. Dane szczegółowe odnośnie technologii:
  - **Framework**: Spring boot,
  - **Kod aplikacji**: java 11
  - **Dostęp do bazy danych**: JPA (dostęp do relacji User), JDBC (dostęp do pozostałych relacji),
  - Kontrola dostępu i uwierzytelnianie: Spring Security
  - Uwierzytelnianie API: JSON Web Token

- **Frontend** – początkowe prace odnoście frontendu aplikacji prowadzone były z wykorzystaniem JSP. W trakcie realizacji projektu została podjęta decyzja o zmianie technologii na aplikację React. W technologii React przygotowano pełną wersję aplikacji, dostarczającą wszystkie zakładane funkcjonalności. Frontend dostępny przez technologię JSP został ograniczony do funkcjonalności wyszukiwania hoteli. Stanowi ona jedynie przykład dydaktyczny i w wersji produkcyjnej część aplikacji związana z JSP powinna zostać usunięta:
    - **Generowanie kodu HTML**: React (pełna aplikacja) i JSP (jedynie przykładowa funkcjonalność – wyszukiwarka hoteli),
    - **Kod apliakcji**: JSX, CSS, Java Script
    - **Zewnętrzne arkusze stylów**: Bootstrap 4

- Baza danych: Oracle DB 19c EE (dostępna na serwerze uczelni ora4.ii.pw.edu.pl)

2.3 Wzorce projektowe

Aplikację przygotowano w oparciu o wzorce projektowe MVC i DAO. Duży nacisk położono na umożliwienie łatwego wprowadzania poprawek i zmian do aplikacji oraz jej rozbudowę:

- MVC – (ang. Model View Controller) aplikacja udostępnia dla API, gdzie poszczególne punkty dostępowe implementowane są przez metody klas kontrolerów. Dane udostępniane przez API oraz przyjmowane jako parametry metod PUT oraz POST stanowią klasy modelu serializowane do JSON. Część aplikacji odpowiedzialna za widok zrealizowana jest poprzez kod JSX aplikacji React.

- DAO (ang. Data Access Object) – dostęp do bazy danych powinien być zrealizowany za pomocą klas dostępu do danych, reprezentowanych przez odpowiedni interfejs. Dzięki temu oraz wykorzystaniu mechanizmu wstrzykiwania zależności, możliwe jest łatwa wymiana implementacji klas dostępu do danych. Dostęp do klas DAO realizowany jest przez klasy serwisów, które wywoływane są z klas kontrolerów oraz realizują weryfikację poprawności poszczególnych parametrów.

2.4 Struktura aplikacji

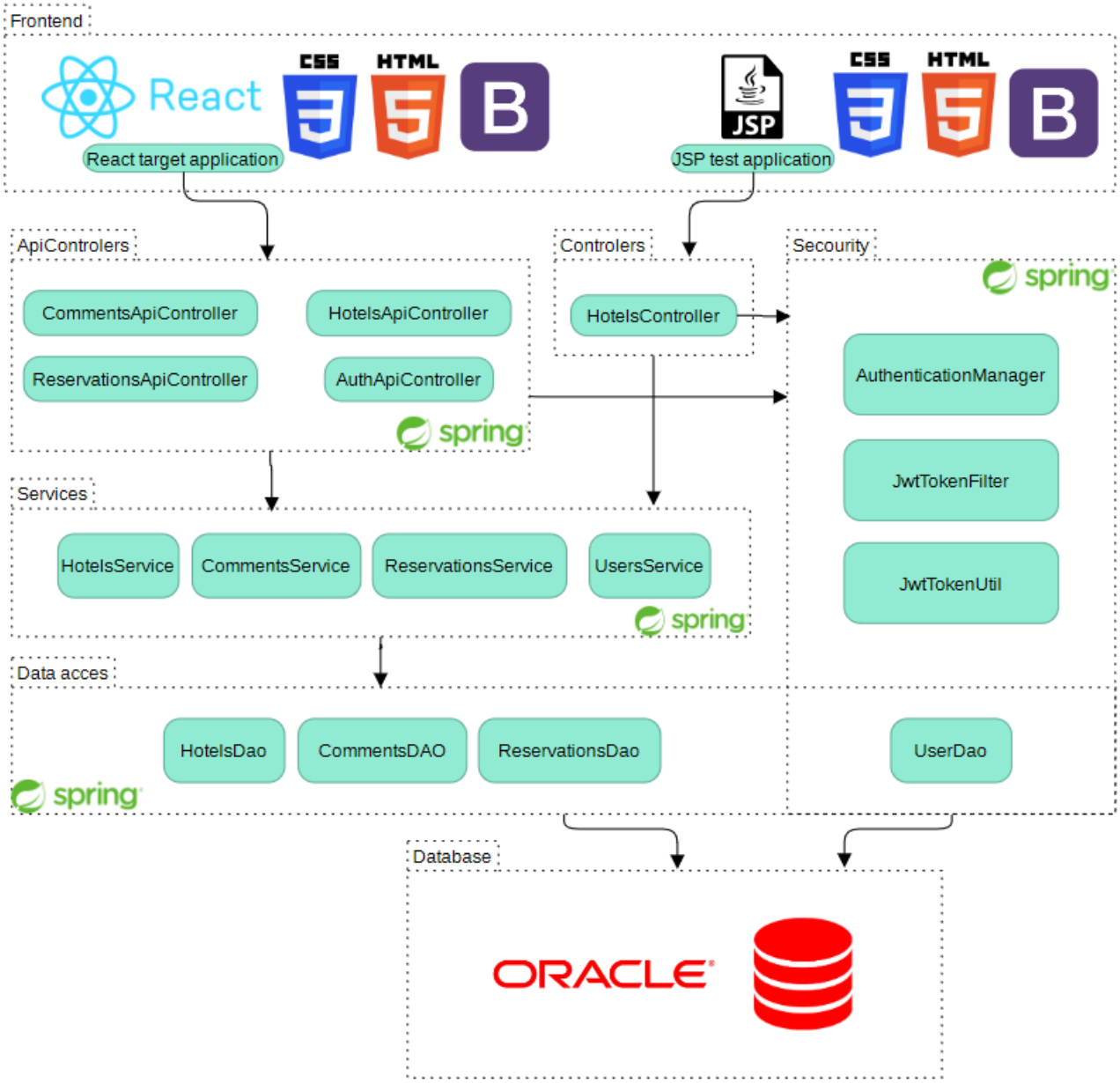
2.4.1 Backend

Diagram aplikacji realizującej backend przedstawiono na Rys. 2.1. Aplikacja React odwołuje się poprzez API do poszczególnych kontrolerów. Jeśli wywołanie wymaga uwierzytelniania, odpowiedni filtr (klasa JwtTokenFilter) sprawdza, czy wśród nagłówków wiadomości pojawił się wpis *Authentication* zawierającej token poprzedzony słowem *Bearer*. Na podstawie tokenu określana jest tożsamość osoby realizującej wywołanie. Jeśli osoba ma uprawnienia do danego punktu końcowego, działanie jest kontunuowanie. W przeciwnym przypadku zwracany jest błąd 401. Przewiduje się dwie role użytkowników:

- Role_User – rola nadawana użytkownikom wykorzystującym aplikacje do poszukiwania ofert noclegowych,

- Role_Owner – rola nadawana użytkownikom reklamującym w aplikacji swoje usługi hotelarskie.

Token oprócz ochrony dostępu do określonych akcji, wykorzystywany jest również do określenia ID użytkownika, na potrzeby realizacji różnego rodzaju operacji na bazie danych. Przekazywanie ID poprzez API stworzyłoby poważne zagrożenie bezpieczeństwa (poprzez wprowadzenie modyfikacji do wiadomości http umożliwiłoby zalogowanemu użytkownikowi modyfikację danych innych użytkowników).



Rys. 2.1 - Struktura aplikacji – Backend

Klasy kontrolerów odwołują się do odpowiadającym im klas serwisów, które realizują walidacje parametrów wywołania. Następnie serwisy odwołują się do odpowiednich klas DAO, które realizują dostęp do bazy danych.

Aplikacja będzie miała budowę modułową. Z każdym modułem związany jest kontroler, serwis i klasa dostępu do danych. Przewiduje się realizację niezależnego modułu dla każdego z obszarów:

- **Użytkownik i autoryzacja** – moduł realizuje punktu dostępowe związane z logowaniem oraz tworzeniem nowych kont,

- **Hotele** – moduł realizuje wszystkie operacje związane z wyszukiwaniem oraz administracja hotelami:

    - Szczegółowe dane hotelu o zadanym id,
    - Listowanie podstawowych danych hoteli w określonym mieście i hoteli właściciela o określonym ID,
    - Listowanie pokoi hotelu o danym ID, dostępnych w danym okresie,
    - Udostępniania fotografii hotelu o określonym ID,
    - Dodawanie nowych hoteli,
    - Modyfikacja poszczególnych składników danych hotelu.
- **Rezerwacje** – moduł umożliwia:
    - Listowanie rezerwacji danego użytkownika w danym okresie,
    - Listowanie rezerwacji hotelu o danym ID w danym okresie,
    - Dodawanie rezerwacji pokoju w danym zakresie,
    - Anulowanie rezerwacji pokoju w danym zakresie.
- **Komentarze** – moduł umożliwia:
    - Listowanie komentarzy hotelu o danym ID,
    - Dodawanie komentarza.

2.4.2 Frontend

Aplikacja we frameworku React zbudowana jest na bazie modułów dostarczających określone funkcjonalności lub implementujących określone elementy interface'u użytkownika. Modułu można podzielić na następujące obszary:

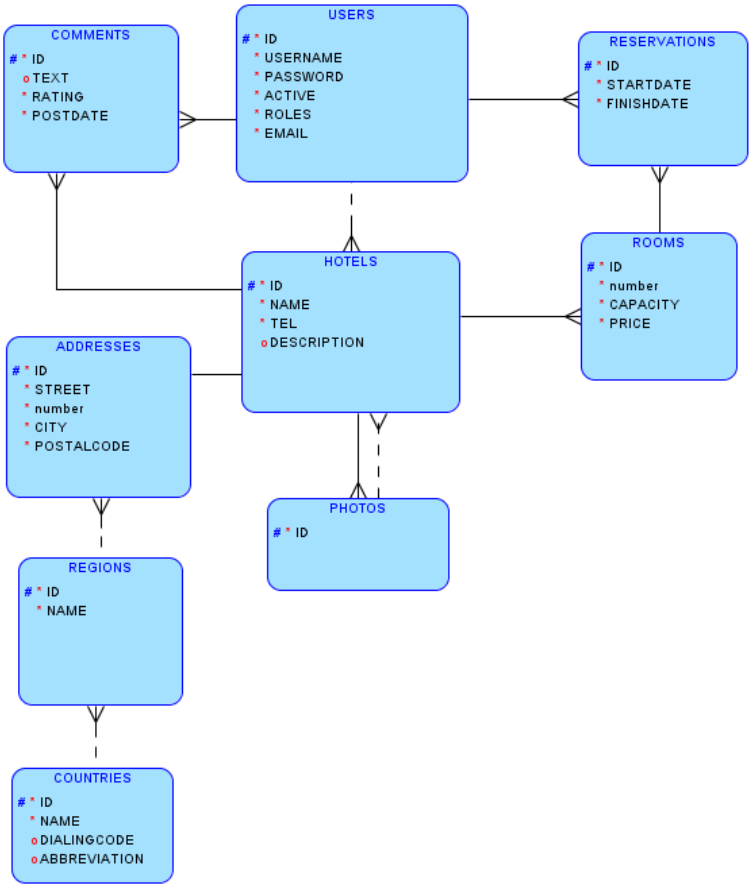
- **Moduły przechowujące kontekst aplikacji:**
    - AuthContextProvider – moduł przechowuje dane o użytkowniku oraz token autoryzacji.
    - ImagesContextProvider – moduł dostarcza ścieżkę dostępu do punktu końcowego udostępniającego fotografię w formacie binarnym na podstawie jej nr ID. (wykorzystywana jako ścieżka src elementów img).
- **Moduły związane z logowaniem oraz dodawaniem kont:**
    - LoginFrom - panel logowania,
    - CreateAccountPage - panel tworzenia nowego konta.
- **Moduły implementujące wyszukiwarkę hoteli**. Najważniejsze z nich to:
    - Browser – nadrzędny moduł wyszukiwarki hoteli,
    - BrowserForm – panel udostępniający funkcję wyszukiwania oraz filtrowania,
    - BrowserList – moduł prezentujące listę hoteli zgodnych z filtrami wyszukwiania,
    - HotelDetails – panel prezentujący dane szczegółowe o hotelu,
    - HotelComments – panel komentarzy w widoku danych szczegółowych danego hotelu.

- Pozostałe moduły w tej sekcji prezentują poszczególne elementy powyższych modułów, w tym m.in. komentarz, kartę pokoju, kartę hotelu.

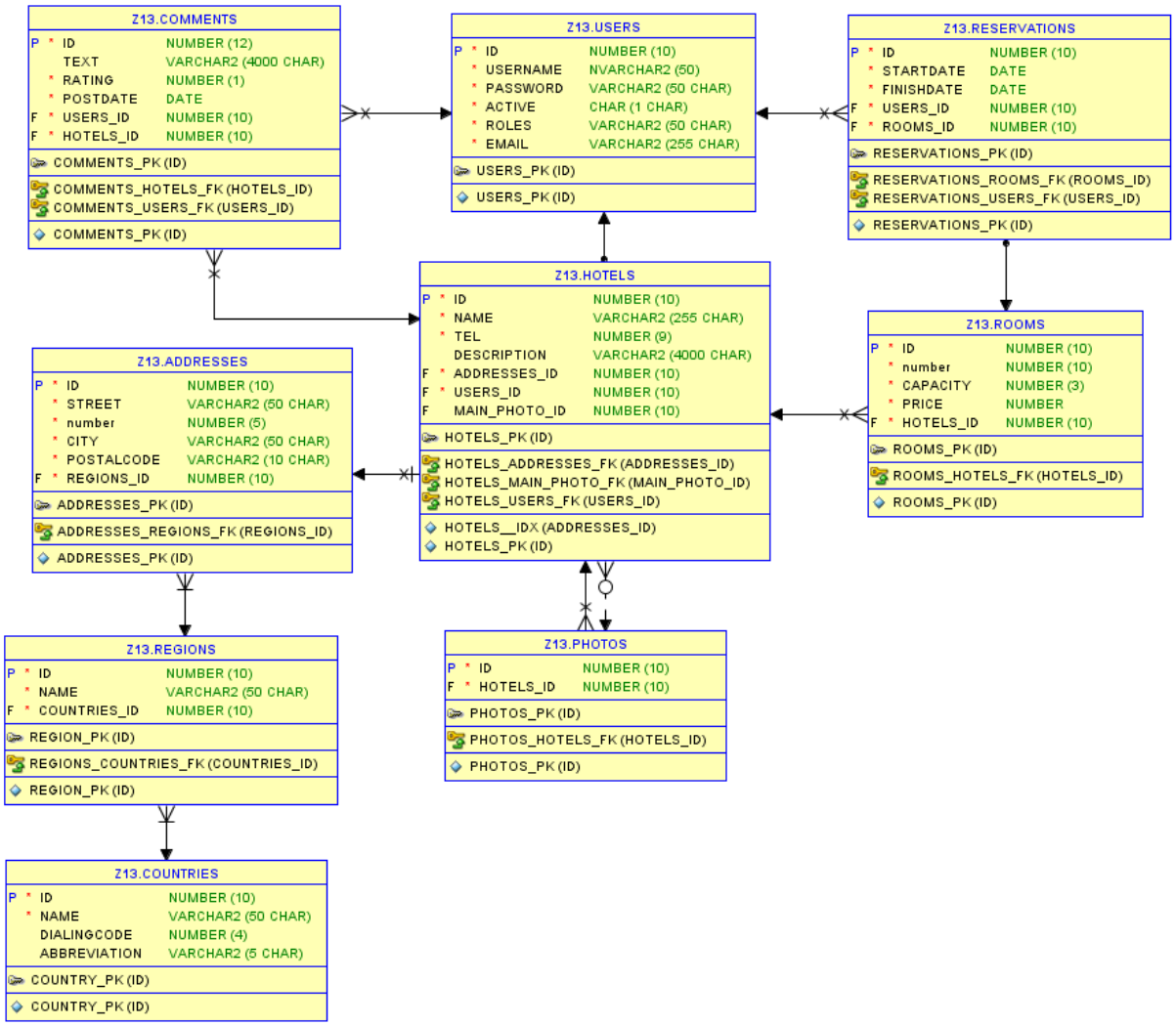
- **Moduły implementujące panel administracji hotelami**. Najważniejsze z nich to:
  - HotelsManager – główny moduł panelu zarzadzania hotelami,
  - HotelsAddPage – formularz dodawania nowego hotelu,
  - HotelManagePage – strona edycji poszczególnych elementów hotelu oraz zarządzania rezerwacjami hotelu,
  - W tym obszarze pojawia duża liczba modłów implementujących poszczególne części interfejsu powyższych modułów. Ze względu na ich sporą ilość ich samo opisujące nazwy, nie będą one wymieniane.
- **Moduł pozwalające na zarządzanie rezerwacjami użytkownika**. Głównym modułem zarządzania rezerwacjami użytkownika jest ReservationManager

2.4.3 Baza danych

Projekt koncepcyjny bazy danych został przedstawiony w postaci diagramu ER na Rys. 2.2. Odpowiadający mu model logiczny został przedstawiony na Rys. 2.3.



Rys. 2.2 - Projekt bazy danych - model koncepcyjny



Rys. 2.3 - Projekt bazy danych - model logiczny

Najbardziej rozbudowana encją modelu jest tablica hoteli. Jest ona powiązana na następującymi relacjami:

- Tablicą adresów – relacja one-to-one. Zakłada się że nie może być zarejestrowanych dwóch hoteli pod jednym adresem,
- Tablicą zdjęć:
  - Relacja one-to-many. Wskazuje wszystkie zdjęcia danego hotelu. Dopuszcza się sytuację, ze hotel nie ma ani jednego zdjęcia,
  - Relacja one-to-one. Wskazuje zdjęcie główne danego hotelu. Dopuszcza się brak zdjęcia głównego.
- Tablicą użytkowników – relacja many-to-one. Wskazuje administratora danego hotelu. Dopuszcza się sytuacje, że administrator zarządza kilkoma hotelami,
- Tablicą pokoi – relacja one-to-many – hotel powinien mieć 0 lub więcej pokoi. Sytuacje zero pokoi dopuszcza się na etapie tworzenia profilu hotelu. Możliwe jest dodanie hotelu bez pokoi, a następnie dodanie pokoi do hotelu w kolejnym kroku, przez panel zarzadzania hotelami,

- Tablica komentarzy - relacja one-to-many.

Ponadto zakłada się że zdjęcia przechowywane będą w plikach zamiast bazy danych. Takie rozwiązanie wynika z ograniczeń nałożonych na zakres dostępnych typów w bazie danych na politechnice. Dla studentów zablokowana jest możliwość tworzenia danych typy LOB. W bazie danych będzie przechowywane jedynie id zdjeć. Plik zdjęcia będzie miał nazwę taka samą ja ID zdjęcia.

Każdy użytkownik będzie mógł tworzyć wiele komentarzy oraz składać wiele rezerwacji. Dopuszcza się sytuację, że użytkownik będzie nie będzie miał przyporządkowanego żadnego komentarza oraz żadnej rezerwacji.

3 Testowanie i Walidacja kodu

3.1 Testy jednostkowe

Jednym z zadań było przygotowanie przykładowych testów jednostkowych. Podjęto decyzję, że realizacja testów jednostkowych zostanie przygotowana na bazie klasy HotelsService do sprawdzenia metod realizujących walidację wprowadzanych danych dla formularzy tworzenia nowego hotelu oraz edycji danych istniejących hotelów. Sprawdzane są następujące informacje, przyporządkowane do trzech grup tematycznych:

- Podstawowych danych o hotelu, na które składają się testy:
  - Czy pole nazwy hotelu nie jest puste,
  - Czy pole opisu hotelu nie jest puste,
  - Czy numer telefonu hotelu składa się z dziewięciu lub dziesięciu cyfry,
- Danych adresowych, na które składają się testy:
  - Czy nazwa ulicy nie jest pusta,
  - Czy numer posesji nie jest pusty,
  - Czy pole regionu nie jest puste,
  - Czy nazwa kraju nie jest pusta,
  - Czy pole kodu pocztowego nie jest puste.
- Danych o nowych pokojach, na które składają się testy:
  - Czy numery pokojów są unikatowe w ramach dodawanej grupy pokoi,
  - Czy numery pokojów są unikatowe w ramach istniejących już pokoi hotelu,
  - Czy pole pojemności pokoju nie jest puste oraz czy liczba cyfr pojemności jest mniejsza równa 3,
  - Czy pole ceny nie jest puste.

Niespełnienie dowolnej z reguł powinno skutkować rzuceniem wyjątku z odpowiednim opisem, który zostanie wyświetlony na interfejsie w formularzu dodawania lub edycji hotelów.

Dla każdej z w.w. funkcji możliwe było przygotowanie niezależnej metody testu jednostkowego. Takie rozwiązanie wygenerowałoby dużą ilość powtarzalnego kodu związanego z przygotowaniem klasy “mocka” imitującego bazę danych. Z tego względu podjęto decyzję przygotowaniu trzech metod testów, gdzie każda jest poświęcona jednej z wymienionych grup, a każdej z reguł zrealizowano asercje sprawdzające zdziałanie zarówno przy poprawnym oraz niepoprawnym zestawie danych.

Dla wszystkich reguł działanie jest prawidłowe. W wszystkich przypadkach, kiedy dane nie są prawidłowe generowany jest wyjątek z właściwą wiadomością.

3.2 Walidacja kodu

3.2.1 Walidacja kodu HTML

Frontend aplikacji tworzony jest w frameworku React, gdzie strony HTML generowane są z plików JSX. Nie ma dostępnych walidatorów kodu JSX. Z tego względu uzyskano zgodę aby pominąć ten krok.

3.2.2 Walidacja kaskadowych arkuszy stylów

Większość aplikacji została stworzona w oparciu o bibliotekę Bootstrap 4. Przygotowany został tylko jeden nierozbudowany plik z arkuszami styli, które implementują wygląd elementów interfejsu niedostępnych w bibliotece, dotyczący:

- Czcionek,
- Elementu radio buton dotyczącego wyboru oceny hotelu w panelu komentarzy,
- Panelu komentarzu.

Wynik walidacji jest bez błędów (załącznik nr 1). Wygenerowane zostały natomiast 3 ostrzezenia dotyczące wybranych czcionek. Są to domyślne czcionki generowane w szablonie projektu React. Podjęto decyzję o nie wprowadzaniu zmian w tym zakresie.

3.2.3 Walidacja kodu java

Walidacja kodu została wykonana z użyciem Checkstyle Maven Plugin (załącznik nr 2). Zostało wykrytych 1605 błędów, z czego dzielą się one na:

- 117 błędów w kodowaniu
- 303 błędy w projektowaniu
- 3 błędy związane z importowaniem
- 444 błędy związane z brakującymi komentarzami Javadoc
- 278 brakujących oznaczeń final do parametrów
- 110 błędów spowodowanych pustymi liniami
- 215 błędów spowodowanych za długimi liniami kodu
- 125 błędów spowodowanych brakiem spacji

4 Uruchomienie aplikacji

Aplikacja przygotowana została w technologii Spring Boot oraz React, więc jej uruchomienie jest stosunkowo proste. Aby uruchomić aplikację należy przeprowadzić następujące kroki:

1. Sklonować z repozytorium projekt zawierający backend aplikacji. Projekt wykonano w javie w wersji 11, więc zaleca się wykorzystanie tej wersji podczas testowania,
2. Projekt korzysta z bazy danych na PW, więc nie są wymagane żadne kroki związane z jej inicjacją. Jeśli użytkownik chciałby przenieść aplikację na inną bazę danych musi dodatkowo wykonać następujące kroki:

- W pożądanej bazie danych uruchomić skrypt Travel_NOW_DB_CREATE.sql załączony do repozytorium w folderze DB, który utworzy wszystkie tablice i inne obiekty bazy danych,
- W pożądanej bazie danych uruchomić plik skrypt Travel_NOW_DB_INSERT.sql załączony do repozytorium w folderze DB, który utworzy przykładowe dane testowe.

3. Sklonować z repozytorium projekt forntendu,
4. Upewnić się, że na lokalnej maszynie zainstalowany jest Node.js oraz menager pakietów npm. (Ubuntu: **sudo apt install nodejs, sudo apt install npm;** Windows: oprogramowanie należy pobrać z https://nodejs.org/en/download/)
5. Otworzyć projekt backendu w visual studio code lub dowolnym innym IDE do programowania w javie oraz uruchomić aplikację ("F5" w vs), upewnić się, że aplikacja pracuje na porcie 8080, jeśli nie należy dokonać zmiany w konfiguracji proxy w aplikacji React (plik package.json w katalogu głównym) na port, na którym pracuje aplikacja serwera,
6. Otworzyć katalog główny projektu React oraz uruchomić aplikację, wpisując w wiersz poleceń **npm start**, aplikacja będzie dostępna na porcie 3000. Można ją obsługiwać w dowolnej przeglądarce internetowej pod adresem **localhost:3000/**

5 Wnioski

5.1 Ocena wykorzystanego stosu technologicznego

Do realizacji aplikacji wykorzystano jeden z najbardziej popularnych stosów technologii wykorzystywanych do budowania aplikacji sieciowych. Pełną aplikację zrealizowano jako kliencką aplikację SPA w React, sprzęgniętą z serwerem zaprogramowanym z wykorzystaniem frameworku Java Spring. Takie podejście okazało się znacznie bardziej elastyczne porównaniu do technologii JSP, którą przetestowano na podstawie implementacji pierwszych kilku funkcjonalności aplikacji. Aplikacje SPA okazują się szczególnie dogodne do implementacji mechanizmów takich jak filtrowanie, ponieważ dzięki zachowaniu stanu nie jest wymagane przekazywanie długich ciągów parametrów w URL zapytań http. Ponadto znacznie wygodniej w tych technologiach implementuje się elementy animacyjne prezentowane na stronie, przez co witryna nabiera dużo korzystniejszego wyglądu. Wymagany jest jednak nieco większy nakład pracy ze względu na konieczność przygotowania API do obsługi komunikacji z serwerem po stronie klienta.

5.2 Napotkane problemy

W trakcie realizacji projektu pojawiło się kilka problemów. Pierwszy z nich dotyczył przenośności informacji przechowywyanych w bazie danych. Początkowo utrzymywane były dwie niezależne bazy lokalne na komputerach uczestników projektu. Następnie podjęto decyzje, aby przenieść dane na wspólną bazę działającą na serwerze uczelni.

Na bazę danych Oracle na serwerze uczelni nałożone są stosunkowo restrykcyjne ograniczenia odnośnie uprawnień nadanych studentom. Jednym z nich jest brak możliwości użytkowania typów LOB. Rezultatem był problem z przechowywaniem zdjęć hoteli, które początkowo były składowane jako dane binarne. Podjęto decyzję aby zdjęcia przechowywać w strukturze plików projektu, a w bazie danych przechowuje się jedynie informacje o istnieniu tych zdjęć i więzy integralności do związanych z nimi relacji.

5.3 Możliwości rozwoju aplikacji

Aplikacja implementuje jedynie podstawowe funkcjonalności związane z przemysłem hotelarskim. Można było ją rozbudować jeszcze o wiele funkcjonalności, które z pewnością byłyby przydatne przy użytkowaniu takiej aplikacji. Są to m.in.:

- Silniejsza separacja części portalu związanej z wyszukiwarką hoteli między częścią związaną z administracją hotelami. W rzeczywistej aplikacji funkcjonalności te prawdopodobnie realizowałyby dwie zupełnie niezależne aplikacje,
- Na panelu głównym aplikacji przygotowano przyciski, które mogłyby przenosić do niezaimplementowanych, ale potencjalnie pożądanych przez użytkowników części witryny. Zakres przeglądarki można byłoby rozszerzyć o cześć związaną z wyszukiwaniem restauracji, atrakcji turystycznych, obiektów rekreacyjno-sportowych i innych powiązanych tematycznie obiektów,
- Aplikację warto byłoby wyposażyć w otwarte API z dokładną dokumentacją, tak aby operatorzy innych wyszukiwarek hoteli mogli korzystać z naszej bazy oraz pośredniczyć w rezerwacji hoteli,
- W aplikacji przydatny byłby moduł ułatwiający komunikację pomiędzy klientem, a obsługą hoteli, przyjmujący formę skrzynki na wiadomości lub czatu,
- Ważnym elementem rzeczywistej aplikacji byłby funkcje związane ze zbieraniem dokładnych statystyk użytkowników oraz określaniu ich preferencji. Takie informacje można byłoby wykorzystać do personalizowania sposobu wyszukiwania,
- Na stronie musiałby również pojawić się określone elementy związane z modelem biznesowym przedsiębiorstwa, takie jak promowanie hoteli, które wykupiły pakiety premium lub inne formy reklamy,
- Bardziej zaawansowany portal pośredniczyłby również w przekazywaniu płatności.

5.4 Wnioski końcowe

- W ramach projektu przygotowano aplikację sieciową, która spełnia wszystkie wymagania postawione przez prowadzących przedmiot. Aplikacja ma formę znacznie uproszczoną względem komercyjnych odpowiedników. Niemiej jednak starano się postawić duży nacisk na jej niezawodność. W tym celu podjęto kroki, takie jak dokładna walidacja danych formularzy, czy różne zachowanie niektórych elementów interfejsu, w zależności od stanu zalogowania użytkowania, tak aby zachować spójność aplikacji. Starano się również dużą wagę przyłożyć do wyglądu aplikacji. Z pewnością również pod tym kątem nie dorównuje ona aplikacjom komercyjnym, ale jej wygląd uznać za „przystępny”.
- Do realizacji klas dostępu do danych wykorzystano JDBC. Dzięki temu uzyskano wysoką wydajność zapytań. Nakład pracy na przygotowanie kodu był jednak niewspółmiernie większy niż z wykorzystaniem np. Hibernate. W kolejnych projektach związanych z językiem Java uczestnicy tego projektu z pewnością zdecydują się na wykorzystanie drugiego z tych rozwiązań.

- Bardzo korzystną decyzją na początku pracy z projektem było porzucenie technologii JSP i realizację pełnej aplikacji z wykorzystaniem React. Dzięki temu przygotowano bardziej reaktywny interfejs użytkownika. Uniknięto też wielu problemów z przekazywaniem skomplikowanych parametrów przez URL zapytań.

- Kolejną korzystną decyzją projektową było wykorzystanie gotowej biblioteki arkuszy stylów Bootstrap. Dostarczyła ona implementację stylów dla większości elementów. Tylko wybrane obiekty zostały przygotowane ręcznie. Należy do nich m.in. panel komentarzy oraz tablice administracji zdjęciami hoteli w formularzach edycji i zarządzania hotelami.

Załączniki

1. Wynik walidacji kaskadowych arkuszy stylów - W3C CSS Validator results for index.css (CSS level 3 + SVG).pdf

2. Wynik walidacji kodu java – checkstyle_results.html