Dzień 1

Zadania wstępne:

1. Zainstaluj wszystkie potrzebne narzędzia do rozwoju aplikacji tworzonej w języku Java oraz frameworku Spring Boot czyli Java 11, Git, Maven, Intellij Idea. Sprawdź czy wszystkie narzędzia dostępne są z linii komend. Jeśli nie to trzeba je dodać w zmiennej systemowej PATH w ustawieniach zaawansowanych komputera.
2. Załóż swoje konto na portalu GitHub (jeśli jeszcze go nie posiadasz). Kod tworzony na zajęciach powinien znaleźć się właśnie w tym repozytorium w nowym projekcie np. spring-lbd odpowiednio w branch'ach day1, day2 i day3.
3. Przejdź przez kurs podstaw Gita. Przećwicz zaprezentowane tam komendy. https://www.javappa.com/szybki-dostep/git-pierwsze-kroki
4. Stwórz przykładowy projekt za pomocą Spring Initializer (strona lub Intellij Idea). Jedyna potrzebna zależność to możliwość tworzenia projektów webowych. Spushuj kod do brancha day1.
5. Zbuduj kod z linii poleceń za pomocą polecenia mvn clean install. Znajdź wygenerowany plik jar. Zmień rozszerzenie tego pliku na zip i przeglądnij zawartość tego archiwum.
6. Przeglądnij plik pom.xml i jego poszczególne sekcje.
7. Stwórz pakiety na podstawowe warstwy aplikacji tj. entity, repository, controller i service.
8. W pakiecie service stwórz podpakiet employee oraz interfejs EmployeeService z jedną metodą List findAll() oraz dwie klasy implementujące ten interfejs.
9. W klasie uruchomieniowej projektu za pomocą adnotacji @Autowired wstrzyknij EmployeeService. Następnie za pomocą adnotacji @PostConstruct stwórz przykładowy kod, który wywoła metodę findAll z EmployeeService. Czy taki kod zostanie poprawnie uruchomiony? Zdiagnozuj problem, który wystąpił.
10. Rozwiąż problem z punktu 9 najpierw za pomocą adnotacji @Primary, a następnie oznaczając klasy odpowiednimi adnotacjami @Qualifier.
11. W klasie EmployeeService stwórz metodę getEmployeeNickname(String firstName, String lastName). Metoda ta zwraca nickname pracownika tworzony zgodnie z następującym wzorcem: prefix + 3 pierwsze litery imienia + 3 pierwsze litery nazwiska + suffix. Prefix i suffix powinny być pobierane z ustawień w pliku application.properties. Napisz test, który sprawdzi czy metoda dla przykładowych danych zwraca poprawnie wygnerowane nickname'y.
12. Usuń wcześniej użyte adnotacje @Primary/@Qualifier z implementacji EmployeeService. W zamian tego zastosuj profile dev i prod. Sprawdź czy po ustawieniu ich w konfiguracji projektu uruchamiane są odpowiednie implementacje tej klasy (tryb debug).
13. Dodaj logowanie do wcześniej stworzonej metody getEmployeeNickname. Na początku metody zaloguj przekazywane parametry wejściowe, a na jej końcu wynik działania metody.
14. Zmień poziom logowania dla pakietu org.springframework na DEBUG i zobacz jakie dodatkowe informacje są logowane przy starcie aplikacji.
15. Zmień format daty dla logów na dzień/miesiąc godzina:minuta.
16. W pakiecie entity stwórz nowy podpakiet employee a w nim klasę Employee zawierającą dane pracownika takie jak id, imię, nazwisko, pesel, numer dowodu osobistego, telefon.
17. W serwisie EmployeeService dodaj mapę typu <Long, Employee> która będzie emulowała bazę danych za pomocą prostej mapy.
18. Dopisz dwie metody do serwisu EmployeeService:
    a. findByName – która wyszukuje pracownika po imieniu lub nazwisku (tzn. można podać imię lub nazwisko pracownika i powinniśmy otrzymać jego pełne dane).
    b. save(Employee employee), która zapiszę dane pracownika do mapy.

19. Napisz testy integracyjne, które przetestują poprawność działania powyższych dwóch metod.
20. Sprawdź zmieniając poziom logowania dla autokonfiguracji, które z konfiguracji zostały zainicjalizowane.
21. Dodaj odpowiednie zależności w pliku pom.xml by możliwe było użycie narzędzia Spring Actuator. Włącz wszystkie możliwe dostępne informacje i narzędzia monitorujące.
22. Na stronie głównej Actuator zapoznaj się z dostępnymi endpointami. Przejrzyj każdy z nich i napisz krótki opis każdego z nich. Wyślij go do mnie pod adres: wpieprzyca@fis-sst.pl.
23. Zainstaluj narzędzie Postman, które służy do wysyłania żądań typu zgodnych ze standardem REST API. Spróbuj za pomocą tego narzędzia wysłać do aplikacji request, który dynamicznie zmieni poziom logowania dla całej aplikacji z domyślnego INFO na DEBUG. Sprawdź czy przyniosło to pożądany skutek.

Zapoznaj się z poniższym opisem projektu, który będziemy wykorzystywali na kolejnych zajęciach.

Firma SH jest producentem oprogramowania. Realizuje swoje usługi w formie projektów wykonywanych na rzecz klientów zewnętrznych z wielu różnych krajów. Naszym celem będzie zaprojektowanie aplikacji, która będzie wspomagać zarządzanie takim portfelem projektów.

Nasi klienci mogą mieć swoje oddziały w różnych krajach i miastach (musimy umieć je rozróżniać). Każdy z nich może zlecić nam dowolną liczbę projektów. Projekty realizowane są w różnych technologiach przez różne zespoły. Zespoły należą do jednego z 3 departamentów nazwanych roboczo Java, .NET i SAP. Interesuję nas jednak nie tylko język programowania czy główna technologia ale również bardziej szczegółowy opis stosu technologicznego każdego z projektów. Po każdym miesiącu wystawiana jest klientowi faktura. To co nas interesuje przede wszystkim to pozycje faktury i termin płatności oraz jej aktualny status (przygotowana, wysłana, zapłacona).

Zespoły to pracownicy firmy SH, którzy mogą być zatrudnieni na jeden z 3 rodzajów umów: etat, B2B oraz zlecenie. Pracownicy pełnią w zespołach różne role np. programista, analityk, architekt, tester, manager, itd. Pracownik w różnych projektach może pełnić różne role.

Projekty realizowane są w sprintach, które mogą trwać od 1 do 4 tygodni. Każdy sprint składa się z zestawu user stories, które zostały ukończone w ramach danego sprinta.

Dodatkowo trzeba wziąć pod uwagę, że firma SH ma 2 oddziały. Pierwszy w Gliwicach, drugi w Opolu. Pracownicy powinni być przypisani do jednego z tych oddziałów.

Zadania:

1) Zastanów się jak może wyglądać model rozwiązania powyższego problemu.
2) Stwórz roboczy zestaw pakietów i klas dla warstwy serwisów. Oznacz je odpowiednimi adnotacjami. Spróbuj przewidzieć czy któryś z serwisów będzie potrzebował danych albo logiki z innego serwisu. Jeśli tak to dokonaj odpowiedniego wstrzykiwania zależności między tymi serwisami.

Zapoznaj się z poniższymi materiałami:

- https://spring.io/guides/gs/maven/
- https://www.baeldung.com/spring-component-repository-service
- https://www.baeldung.com/spring-postconstruct-predestroy
- https://www.baeldung.com/spring-profiles
- https://www.baeldung.com/spring-boot-logging
- https://www.baeldung.com/spring-boot-actuators