Wymagania projektowe SSBD

Strona:	WIKAMP FTIMS	Wydrukowane przez użytkownika:	dr inż. Mateusz Smoliński
Przedmiot:	Sieciowe systemy baz danych'2025	Data:	poniedziałek, 30 czerwca 2025, 20:04
Książka:	Wymagania projektowe SSBD		

Opis

Wymagania dotyczące wielodostępnych systemów informatycznych tworzonych w ramach przedmiotu Sieciowe Systemy Baz Danych

Spis treści

1. Wprowadzenie

2. Zasady zaliczania
2.1. Punktacja i oceny
2.2. Składowe projektu
2.3. Terminy i opóźnienia

3. Sprawdzanie projektu
3.1. Zakres i punktacja etapów
3.2. Prezentacja systemu
3.3. Weryfikacja wiedzy
3.4. Ocena za realizację rozszerzeń

4. Prowadzenie projektu

5. Założenia projektowe
5.1. Wymagane technologie
5.2. Podstawowe pojęcia i założenia architektury
5.3. Tematyka aplikacji
5.4. Wymagania funkcjonalne
5.5. Wymagania ilościowe

6. Wymagania dokumentacji projektu
6.1. Dokumentacja projektu końcowego

7. Wymagania kodu źródłowego i wersjonowania
7.1. Podział aplikacji na moduły

8. Wymagania systemu rejestracji zadań (todo)

9. Wymagania technologiczne
9.1. Warstwa danych
9.2. Warstwa logiki biznesowej
9.3. Warstwa prezentacji
9.4. Kontrola spójności danych
9.5. Wymagania bezpieczeństwa

10. Zmiany wymagań

Wprowadzenie

Celem zajęć jest zaprojektowanie i zrealizowanie wielodostępnego system informatycznego z obsługą różnych poziomów dostępu użytkowników zgodnie z modelem dostępu bazującym na rolach RBAC (Role Base Access Control). Wymagane elementy wielodostępnego systemu informatycznego:

- aplikacja internetowa zapewniająca kompletny graficzny interfejs użytkownika GUI (Graphical User Interface) we współczesnej przeglądarce internetowej użytkownika, aplikacja internetowa musi komunikować się z pozostałymi elementami systemu wyłącznie poprzez żądania wysyłane do odpowiedniej usługi REST,
- aplikacja sieciowa udostępniająca usługę sieciową REST (REpresentational State Transfer) i realizująca transakcyjne przetwarzanie danych oraz stosująca mechanizmy ORM (Object-Relational Mapping) automatycznego odwzorowania modelu obiektowego encji JPA w model relacyjny danych,
- relacyjna baza danych przechowująca dane przetwarzane w wielodostępnym systemie informatycznym. Komunikacja aplikacji sieciowej z relacyjną bazą danych wymaga zastosowania standardu JDBC (Java DataBase Conectivity).

Na całość wytworzoną przez projekt zespołowy SSBD składają się:

- statyczne struktury relacyjnej bazy danych obsługiwanej przez DBMS,
- kod źródłowy aplikacji sieciowej udostępniającej usługę REST w domenie serwera aplikacji,
- kod źródłowy aplikacji internetowej zapewniająca graficzny interfejs użytkownika GUI w przeglądarce internetowej użytkownika.
- dokumentacja projektowa (zgodna z makietą sprawozdania).

Niniejszy dokument jest uszczegóławiany przez następujące dokumenty:

- Harmonogram etapów projektu,
- Wymagane oprogramowanie,
- Tematy zrealizowane w poprzednich edycjach przedmiotu,
- Katalog funkcjonalności dodatkowych MOK,
- Katalog rozszerzeń - architektura i realizacja systemu
- Makieta sprawozdania w formacie MarkDown (todo)

Wymagania zawarte w w/w źródłach obowiązują bez konieczności jawnego odwołania się do nich w treści niniejszego dokumentu.

Zasady zaliczania projektów

Punktacja i oceny

Punktacja:

1. Maksymalna liczba punktów, jaką może uzyskać uczestnik zespołowego projektu SSBD, wynosi 100.
2. Punktacja uzyskiwana na każdym z etapów sprawdzania projektu nie może być ujemna.
3. Minimalna liczba punktów, jaka jest wymagana do uzyskania zaliczenia, wynosi 51, po zaokrągleniu liczby uzyskanych punktów do najbliższej wartości całkowitej.
4. Uzyskana punktacja nie będzie brana pod uwagę, jeżeli nie zostanie spełnione którekolwiek z wymagań głównych, wyszczególnionych w niniejszym dokumencie.
5. Nie spełnienie któregokolwiek z pozostałych wymagań powoduje obniżenie punktacji.
6. Realizacja kolejnego projektu w ramach edycji przedmiotu powoduje utratę 30pkt (tzw. projekt poprawkowy), wówczas jednoczesnej ocenie podlegają wszystkie etapy projektu.
7. Niesamodzielna praca całkowicie dyskwalifikuje możliwość zaliczenia projektu SSBD.

Maksymalna punktacja za poszczególne składowe projektu:

1. Sprawozdanie i realizacja zadań nim objętych – 50 pkt
    1. Sprawozdanie wstępne – 12 pkt
    2. Sprawozdanie szczegółowe – 21 pkt
    3. Sprawozdanie końcowe – 17 pkt
2. Ocena poprawności działania finalnej wersji aplikacji – 15 pkt
3. Archiwum ZIP zawierające kod źródłowy aplikacji i wszystkie niezbędne zasoby do uruchomienia zbudowanego wielodostępnego systemu informatycznego jednym poleceniem w środowisku konteneryzacji Docker – 15 pkt
4. Ocena za rozszerzenia (łącznie) – 30 pkt

Składowe projektu

1. Projekt składa się z następujących składowych podlegających ocenie:
    1. Statyczna struktura bazy danych wprowadzona w przydzielonej grupie relacyjnej bazie danych,
    2. Kod źródłowy aplikacji umieszczony na przydzielonym grupie repozytorium,
    3. Zakres i postęp prac opisany w systemie rejestracji zadań (wraz z potwierdzaniem ich wykonania),
    4. Dokumentacja projektu umieszczona w formie elektronicznej na platformie WIKAMP,
    5. Osadzona na przydzielonej grupie infrastrukturze, w pełni funkcjonująca aplikacja (dotyczy projektu na etapie końcowym i częściowo dla modułu funkcjonalnego MOK na etapie sprawozdania szczegółowego),
    6. Umieszczone w bazie danych aplikacji dane przykładowe, umożliwiające zaprezentowanie funkcjonowania aplikacji (dotyczy projektu na etapie końcowym i częściowo dla modułu funkcjonalnego MOK na etapie sprawozdania szczegółowego) oraz na etapie sprawozdania wstępnego - przedstawienie planowanego sposobu realizacji uwierzytelniania oraz zarządzania kontami systemu,
    7. Archiwum ZIP z zawartością zgodną z wymaganiami określonymi dla projektu końcowego (dotyczy projektu na etapie końcowym) umieszczone na platformie Wikamp.
2. Należy zwrócić szczególną uwagę na fakt, że poszczególne składowe projektu muszą być udostępnione przez grupę jedynie z wykorzystaniem platformy WIKAMP oraz serwerów przydzielonych przez prowadzących. Składowe udostępniane z wykorzystaniem innych zasobów nie będą uwzględniane w ocenie projektu.

Terminy i opóźnienia

Terminy

1. Zaliczanie projektu przebiega w trzech etapach odpowiadających kolejnym fazom powstawania systemu informatycznego (złożonego z aplikacji internetowej i aplikacji sieciowej oraz relacyjnej bazy danych) opisanymi w:
    - sprawozdaniu wstępnym
    - sprawozdaniu szczegółowym
    - sprawozdaniu końcowym
2. Poszczególnym etapom przyporządkowane zostają terminy oddania, które są ogłaszane w harmonogramie etapów projektu.
3. Zmiany harmonogramu wymagają zgody prowadzącego oraz kwalifikowanej większości uczestników zajęć i ogłoszenia przez prowadzącego w formie pisemnej (elektronicznej) do wiadomości wszystkich uczestników zajęć.
4. Etap projektu uważa się za oddany w danym terminie, jeżeli grupa zgłosi na platformie WIKAMP gotowość do oddania w sposób wskazany przez prowadzącego nie później niż w ciągu 15 minut od momentu rozpoczęcia zajęć będących wyznaczonym terminem oddania, do których przypisana jest grupa projektowa.
5. Decydującym czasem oddania dla danego etapu projektu jest znacznik czasowy zgłoszenia gotowości na platformie WIKAMP.
6. Sprawdzeniu i ocenie podlega wyłącznie stan składowych projektu w chwili oddania.

Opóźnienia

1. Oddanie etapu projektu po upływie w/w terminu oddania powoduje opóźnienie, przy czym z każdym przekroczonym kolejnym terminem oddania punktacja sprawozdania jest zmniejszana o 10.
2. Kolejne terminy oddania etapu są wyznaczane przez kolejne zajęcia, a w sesji egzaminacyjnej przez prowadzących, przy czym okres czasu pomiędzy kolejnymi terminami nie może być krótszy niż tydzień.
3. Punktacja opóźnionego sprawozdania nie może być ujemna, przy czym oddanie każdego sprawozdania jest obligatoryjne.
4. Oddany etap projektu może być sprawdzany po czasie oddania i nie ma to wpływu na punktację.
5. Okres od oddania etapu projektu przez grupę do terminu sprawdzenia nie jest uwzględniany przy naliczaniu opóźnień.
6. Jeżeli którakolwiek z wymaganych składowych etapu projektu nie została udostępniona lub nie funkcjonuje zgodnie z wymaganiami w terminie sprawdzania, wówczas od tego momentu zostają naliczane opóźnienia, a grupa zobowiązana jest do ponownego oddania danego etapu projektu.

Obecność na zajęciach

1. Obecność na zajęciach jest obowiązkowa.
2. Dodatkowo wszystkich członków grupy obowiązuje obecność w terminie sprawdzania każdego z sprawozdań.
3. W przypadku, gdy sprawozdanie zostało oddane, a grupa bez usprawiedliwienia nie stawiła się w komplecie na termin sprawdzenia, wówczas sprawozdanie będzie sprawdzane w ostatniej kolejności.

Zasady sprawdzania projektu

Zakres i punktacja etapów

1. Zakres zadań do wykonania w ramach danego etapu oraz liczba punktów możliwych do uzyskania za wykonanie danego zadania wynikają bezpośrednio z treści makiety sprawozdania.
2. Makieta sprawozdania zawiera rozdziały odpowiadające poszczególnym etapom realizacji projektu.
3. Każdy taki rozdział składa się z szeregu podrozdziałów, opisujących, jakie składowe projektu mają być udokumentowane w danym podrozdziale, oraz punktację jednostkową za realizację danego podrozdziału.
4. Punktacja uzyskana w danym etapie jest sumą punktów uzyskanych w opisujących go podrozdziałach makiety.
5. Przez realizację danego podrozdziału makiety rozumie się:
    1. Wypełnienie zawartości podrozdziału dokumentacją projektu zgodnie z opisem podrozdziału oraz wymaganiami dla dokumentacji określonymi w niniejszym dokumencie.
    2. Rzeczywistą realizację danego elementu projektu zgodnie ze standardami technologicznymi i z dokumentacją projektu oraz wymaganiami dla poszczególnych etapów projektu określonych w niniejszym dokumencie.
    3. Wykazanie się przez członków grupy znajomością podstaw teoretycznych i szczegółów technicznych zastosowanych rozwiązań.
6. Punktację za realizację zadań wymienionych w poszczególnych podrozdziałach ustala się na podstawie wymienionych wyżej kryteriów, niezależnie dla każdego podrozdziału.

Prezentacja systemu informatycznego

Prezentacja funkcjonalności MOK

1. Jednym z elementów sprawdzania na etapie *sprawozdania szczegółowego* jest zaprezentowanie przez grupę działania systemu zgodnie ze specyfikacją MOK uruchomionego w przydzielonej grupie infrastrukturze wirtualnej w platformie oVirt.
2. Prezentacja musi odbywać się interaktywnie z wykorzystaniem przeglądarki stron WWW.
3. Prezentacja nie może przekroczyć 10 minut, uwidaczniając całą funkcjonalność MOK.
4. Działanie MOK zgodne z wymaganiami funkcjonalnymi oraz technologicznymi projektu jest warunkiem koniecznym przed przystąpieniem do sprawdzania sprawozdania szczegółowego.

Prezentacja funkcjonalności dziedzinowej

1. Jednym z elementów sprawdzania na etapie *sprawdzania końcowego* jest zaprezentowanie przez grupę działania stworzonego systemu informatycznego uruchomionego w przydzielonej grupie infrastrukturze wirtualnej w platformie oVirt.
2. Prezentacja musi odbywać się interaktywnie, z wykorzystaniem przeglądarki stron WWW.
3. Prezentacja nie może przekroczyć 15 minut, uwidaczniając całą funkcjonalność dziedzinową.
4. Działanie stworzonego systemu zgodne z wymaganiami funkcjonalnymi oraz technologicznymi projektu jest warunkiem koniecznym przed przystąpieniem do sprawdzania sprawozdania końcowego.
5. W dalszym etapie projektu końcowego prowadzący zajęcia mogą weryfikować poprawność działania systemu korzystając w dowolny sposób z oferowanej przez niego funkcjonalności.

Weryfikacja wiedzy

1. Weryfikacja wiedzy teoretycznej oraz znajomości szczegółów realizacji projektu odbywa się w trakcie sprawdzania projektu, poprzez zadawanie pytań poszczególnym członkom grupy.
2. W przypadku stwierdzenia różnic pomiędzy poziomem wiedzy poszczególnych członków grupy ocena punktowa danego podrozdziału może zostać zróżnicowana.
3. Znajomość podstaw teoretycznych (dostępnych w literaturze i materiałach pomocniczych udostępnionych przez prowadzących) dotyczących zagadnień poruszanych w danym podrozdziale obowiązuje każdego członka grupy bez względu na pełnione funkcje w ramach grupy.
4. Znajomość szczegółów zastosowania danego rozwiązania w projekcie obowiązuje członków grupy będących autorami implementacji tego rozwiązania. W przypadku sporu, osoby odpowiedzialne za implementację rozwiązania wskazuje szef grupy.
5. Grupa projektowa jest zobowiązana do przygotowania promocyjnego materiału wideo nie dłuższego niż 5 minut przedstawiającego najciekawsze aspekty funkcjonalne stworzonego oprogramowania. Link do przygotowanego materiału wideo opublikowanego na platformie YouTube musi zostać przekazany nauczycielom prowadzącym nie później niż przed rozpoczęciem odpowiedzi z etapu końcowego projektu.

Ocena za realizację rozszerzeń

1. Realizacja rozszerzeń z katalogu nie jest obligatoryjna.
2. Poprawna realizacja wybranych rozszerzeń powoduje zwiększenie punktacji z projektu o dodatkowe punkty zgodnie z zapisami w katalogu.
3. Łączna liczba przyznanych za rozszerzenia punktów nie może przekroczyć 30. W wyniku przyznania dodatkowych punktów ocena z projektu nie może być większa niż 100 punktów.
4. Rozszerzenia mogą być realizowane przez część składu grupy, wówczas dodatkowe punkty zostaną przyznane tylko tym członkom grupy, którzy wzięli udział w realizacji rozszerzenia.
5. Szef grupy jest zobowiązany do zgłoszenia tuż przed oddaniem etapu szczegółowego wszystkich wykonanych aktywności z katalogu rozszerzeń - funkcjonalność dodatkowa dla MOK oraz tuż przed oddaniem etapu końcowego wszystkich wykonanych aktywności z katalogu rozszerzeń - architektura i realizacja systemu. Zgłoszenie rozszerzenia wymaga zamieszczenia jego identyfikatora na stronie tytułowej dokumentacji, jedynie zgłoszone i zaprezentowane aktywności podlegają ocenie.
6. Każdy uczestnik zespołu jest zobowiązany do wskazania w odpowiednim katalogu rozszerzeń wszystkich rozszerzeń, w realizacji których uczestniczył. Nie wskazanie rozszerzeń przed terminem szczegółowego/końcowego etapu projektu powoduje brak możliwości ich rozliczenia.

Prowadzenie projektu

1. Grupy projektowe
    1. Studenci uczestniczący w zajęciach projektowych przedmiotu Sieciowe Systemy Baz Danych dzieleni są na grupy i każda grupa prowadzi jeden projekt.
    2. Standardowa liczność grupy to 6 osób, na życzenie grupy i za zgodą prowadzącego zajęcia mogą zostać wprowadzone odstępstwa od tej reguły.
    3. Podział na grupy z założenia ma charakter dobrowolnej deklaracji poszczególnych studentów, natomiast nauczyciel prowadzący zajęcia (tzw. prowadzący) ma prawo wprowadzić arbitralny podział na grupy.
2. Funkcje uczestników grupy
    1. Grupa projektowa wybiera spośród swoich członków **szefa grupy**. Szef grupy jest osobą upoważnioną do dokonywania w imieniu grupy wszelkich ustaleń z prowadzącym zajęcia, a jego decyzje są wiążące w stosunku do całej grupy.
    2. Obowiązkiem szefa grupy jest koordynowanie prac grupy poprzez przydzielanie i kontrolę wykonania zadań poszczególnym jej członkom. Szef grupy odpowiada też za przydzielenie innych funkcji pozostałym uczestnikom projektu oraz aktualność wszystkich informacji przedstawionych na stronie tytułowej dokumentacji projektu (tzw. strona domowa).
    3. Poszczególne funkcje określają odpowiedzialność za:
        - zarządzanie przygotowaniem dokumentacji (**dokumentacja**)
        - zarządzanie relacyjną bazą danych wraz z jej zawartością (**baza**)
        - przygotowanie niezbędnych elementów infrastruktury uruchomieniowej dla budowanego systemu, zarządzanie domeną i zasobami serwera aplikacyjnego oraz uruchamianiem aplikacji stworzonych w ramach projektu (**wdrożenie**)
        - przygotowanie struktury projektów *maven* (ozn. MVN) i bibliotek oraz ustalenie zależności pomiędzy nimi oraz zarządzanie budowaniem stworzonego w ramach projektu oprogramowania i przygotowaniem archiwum ze zbudowanym oprogramowaniem (**architektura**)
        - integrowanie kodu z gałęzi do głównej linii rozwojowej oraz umieszczanie przetestowanych/zweryfikowanych wersji kodu jako kolejne tagi w repozytorium (**repozytorium**)
        - weryfikację zgodności kodu i dokumentacji z wymaganiami (**kontrola zgodności**)
        - testowanie funkcjonalności aplikacji (**kontrola jakości**)
    4. Każdy uczestnik projektu musi posiadać przynajmniej jedną z wymienionych funkcji, przy czym tylko funkcja kontrola jakości musi być przydzielona kilku uczestnikom projektu. Funkcja kontrola zgodności i repozytorium jest przypisana wyłącznie do szefa projektu.
    5. Funkcje przydzielone uczestnikowi nie zwalniają z obowiązku przygotowania dokumentacji, projektowania i tworzenia kodu aplikacji.
    6. Wszystkie zasady oddawania i zaliczania projektu dotyczą w równym stopniu szefa grupy jak i jej pozostałych uczestników.
3. Środowisko pracy
    1. Nauczyciele prowadzący zajęcia zapewniają każdej grupie możliwość korzystania z następujących zasobów:
        - dedykowany kurs na platformie edukacyjnej WIKAMP,
        - środowisko prowadzenia projektów (repozytorium, system rejestracji zadań)
        - infrastruktura wdrożeniowa
    2. Grupa może korzystać z innych zasobów, natomiast należy pamiętać, że zaliczenie projektu wymaga prowadzenia go i umieszczenia jego składowych na serwerach udostępnionych przez prowadzących zajęcia, zgodnie z zasadami zaliczania opisanymi w odrębnym rozdziale.
    3. Każda niedostępność usług sieciowych oraz serwerów przydzielonych grupie do realizacji projektu SSBD musi być natychmiast zgłoszona w wątku forum kursu przedmiotu, a w przypadku niedostępności platformy WIKAMP poprzez wiadomość e-mail adresowaną do nauczycieli prowadzących zajęcia. Rozliczenie niedostępności usług będzie liczone od momentu zgłoszenia niedostępności.

Założenia projektowe aplikacji

Wymagane technologie

1. Trójwarstwowa aplikacja biznesowa ma zostać zrealizowana z wykorzystaniem następujących technologii:
    - język programowania: Java
    - serwer aplikacji: Tomcat (jako serwer osadzony w szkielecie SpringBoot)
    - system zarządzania bazami danych: PostgreSQL
2. Wersje używanego oprogramowania muszą być zgodne z wymaganiami.
3. Warstwa logiki biznesowej odpowiedzialna za przetwarzanie danych zgodnie z regułami biznesowymi (ustalonymi poprzez specyfikację funkcjonalną) tworzonego systemu ma zostać zrealizowana z użyciem komponentów Spring Framework
4. Warstwa prezentacji odpowiedzialna za interakcję z użytkownikami aplikacji sieciowej może zostać zrealizowana w dowolnej technologii, która będzie spełniać wymienione dalej wymagania
5. Warstwa logiki biznesowej udostępnia swoją funkcjonalność w formie REST API, którego klientem jest implementacja warstwy prezentacji.
6. Wszystkie rozwiązania każdego z problemów zastosowane w projekcie muszą być kompletne i spójne pod względem technologicznym i implementacyjnym.

Podstawowe pojęcia i założenia architektury

1. Obsługę graficznego interfejsu użytkownika GUI (Graphical User Interface) musi zapewniać aplikacja internetowa pojedynczej strony SPA (Single Page Application) uruchamiana we współczesnej przeglądarce internetowej użytkownika systemu wielodostępnego (wymagane zastosowanie przynajmniej trzech różnych, współczesnych przeglądarek internetowych WWW pracujących w środowisku graficznym (np. Mozilla Firefox, Google Chrome, Internet Explorer, Microsoft Edge, Opera), wystarczy spełnienie tego wymagania przez najnowsze stabilne wersję wymienionych przeglądarek). Aplikacja internetowa SPA musi generować żądania do usługi REST udostępnionej przez aplikację sieciową, która realizuje dalsze przetwarzanie danych zgodnie z regułami logiki biznesowej (ustalone przez specyfikację funkcjonalną systemu wielodostępnego).
2. Projekt wielodostępnego systemu musi przewidywać podział użytkowników na kategorie (poziomy dostępu) różniące się oferowaną im funkcjonalnością, zwane **aktorami**. Na przykład system sklepu internetowego może posiadać trzech aktorów: administratora, sprzedawcę, klienta i gościa, przy czym ostatni z wymienionych aktorów nie wymaga uwierzytelnienia użytkownika.
3. Funkcjonalność wielodostępnego systemu należy podzielić na **moduły funkcjonalne**, grupujące przypadki użycia operujące na tych samych danych.
    1. Każdemu takiemu modułowi należy nadać unikatowy *identyfikator będący akronimem pełnej jego nazwy*.
    2. Na moduł funkcjonalny składają się poszczególne **przypadki użycia** (ang. *use case*), reprezentujące funkcje dostępne w ramach danego modułu, np. moduł obsługi kont (ozn. MOK) musi posiadać przypadki użycia: rejestracja konta, weryfikacja wiarygodności danych zarejestrowanego konta, zablokowanie/odblokowanie konta, zmiana hasła przypisanego do konta, zmiana adresu e-mail oraz edycja innych danych przypisanych do konta, przypisanie i odebranie poziomu dostępu dla konta oraz resetowanie hasła przypisanego do konta.
        1. Poszczególnym przypadkom użycia należy nadać *nazwy* oraz *numery* zapisywane cyframi arabskimi, np. 1 – rejestracja konta.
        2. Odwołując się do przypadku użycia należy podać właściwy akronim modułu funkcjonalnego wraz z numerem przypadku użycia (np. MOK.1).
        3. Należy pamiętać, że każdy przypadek użycia może przebiegać według jednego z wielu **scenariuszy,** na przykład rejestrowanie konta użytkownika może nie udać się ze względu na naruszenie ograniczeń bazy danych i jest to inny scenariusz niż w przypadku, gdy konto użytkownika zostanie prawidłowo utworzone.
            1. **Scenariusz główny** przypadku użycia nie obejmuje wystąpienia błędu.
            2. W przypadku występowania różnic w głównych scenariuszach przypadku użycia dla różnych aktorów, przedstawiany przypadek użycia należy uszczegółowić/rozszerzyć.
            3. W każdym ze scenariuszy wykonywana jest jedna lub więcej **operacji** na obiektach **encji JPA** (Java Persistence API). Dla przykładu scenariusz modyfikowania danych użytkownika zawiera operację odczytu tych danych i następnie zapisania zmodyfikowanych danych do bazy.
            4. Każda z wymienionych operacji musi być uczestnikiem **transakcji** bazodanowej, przy czym jedna transakcja bazodanowa może realizować jedną lub więcej **operacji.**
            5. Transakcje bazodanowe dzielimy na **trywialne** i **nietrywialne.**
                1. Transakcje nietrywialne to takie, które dokonują *bezpośrednich zmian* w bazie danych lub *odczytują dane na potrzeby późniejszej ich zmiany* w innej powiązanej transakcji. Wynika z tego, że każda transakcja dokonująca dodania, modyfikacji lub usunięcia danych jest nietrywialna.
                2. Podobnie nietrywialna jest transakcja odczytująca dane użytkownika wykonywana w ramach przypadku użycia „modyfikacja użytkownika”, ponieważ prowadzi bezpośrednio do transakcji zmieniającej dane w bazie danych.
                3. Natomiast za trywialną można uznać np. transakcję odczytującą dane wszystkich użytkowników wyłącznie celem zaprezentowania ich zestawienia w postaci listy.
    3. Z punktu widzenia interfejsu użytkownika przypadek użycia przekłada się na **akcję,** którą użytkownik może inicjować za pośrednictwem interfejsu.
        1. Zbiór akcji każdego z aktorów odpowiada podzbiorowi wszystkich przypadków użycia.
        2. Pojedynczy przypadek użycia może się również przekładać na wiele akcji, jeżeli korzysta z niego wielu aktorów, np. zarówno administrator, jak i sprzedawca mogą mieć możliwość korzystania z przypadku użycia „wyświetl dane klienta” i wówczas z tym przypadkiem użycia będą związane dwie akcje.

Tematyka aplikacji

1. Tematyka projektu SSBD może być ustalona przez grupę prowadzącą projekt, z wymogiem zaakceptowania przez prowadzącego zajęcia. Dopuszcza się tematykę ustaloną dla grupy przez prowadzącego zajęcia.
2. Tematyka musi być unikalna w ramach bieżących, realizowanych w danym roku projektów SSBD wraz z projektami realizowanymi we wcześniejszych edycjach przedmiotu oraz prac dyplomowych dotyczących budowy aplikacji w technologii Java/Spring.
3. Zakres funkcjonalności oprogramowania stworzonego w ramach wybranej tematyki ustala grupa z tym, że zarówno projekt, jak i realizacja nie mogą znacząco odbiegać od praktyki obserwowanej w danej dziedzinie. W szczególności projekt musi zapewniać podstawową funkcjonalność w ramach podjętego tematu oferując użytkownikom wykonywanie akcji bez zbędnych interakcji.
4. W ramach implementacji modelu dziedzinowego wyróżniamy pojęcie **funkcjonalności krytycznej**, która jest rozumiana jako zasadniczy cel danego modelu (np. zakup, wypożyczenie, rezerwacja, zgłoszenie). Przypadki użycia realizujące funkcjonalność krytyczną są dalej objęte szczególnymi wymaganiami.
    1. Funkcjonalność krytyczna musi być związana z dynamiczną agregacją informacji o danych biznesowych przechowywaną w bazie. Typowe przykłady to wszelkiego rodzaju statystyki (np. sprzedaży, wypożyczeń, rezerwacji, zgłoszeń), jakkolwiek grupa może zaproponować inny rodzaj danych.

Wymagania funkcjonalne

Poniższe wymagania dotyczą wszystkich aplikacji niezależnie od ich tematyki.

Wymagania główne

1. Stworzone oprogramowanie musi umożliwiać zarządzanie kontami/grupami użytkowników z poziomu graficznego interfejsu użytkownika GUI oraz zapewniać możliwość uwierzytelnienia użytkowników (z zastosowaniem poświadczeń: unikalnego i niezmiennego loginu nie zawierającego żadnych danych biznesowych np. adresu e-mail i silnego hasła nie krótszego niż 8 znaków) dysponującym indywidualnym kontem (nie dotyczy poziomu dostępu gość). Zasoby JDBC wykorzystywane w procesie uwierzytelniania użytkowników nie mogą być tworzone z wykorzystaniem deskryptorów wdrożenia. Profil każdego konta użytkownika przechowywany w relacyjnej bazie danych musi uwzględniać następujące informacje: znacznik czasu (data i czas) ostatniego poprawnego i niepoprawnego uwierzytelnienia wraz z adresami logicznymi oraz obowiązująca dla konta wersja językowa aktualizowana przynajmniej przy każdym poprawnym uwierzytelnieniu użytkownika. Wymienione informacje z profilu konta muszą być dostępne w interfejsie użytkownika, będącego właścicielem konta oraz dla każdego użytkownika administracyjnego. Ustalona dla konta wersja językowa obowiązuje przy internacjonalizacji wszystkich wysyłanych wiadomości e-mail na adres pocztowy przypisany do konta użytkownika.
2. Minimalny zestaw przypadków użycia związanych z zarządzaniem kontami użytkowników musi obejmować:
    1. samodzielną rejestrację (utworzenie konta) przez użytkownika anonimowego (tzw. gościa), przy czym utworzenie aktywnego konta musi uwzględniać etap weryfikacji wiarygodności podanych danych poprzez unikalny kod (nie dłuższy niż 8 znaków) lub indywidualny adres URL udostępniony jako hiperłącze przesłane na adres email przypisany do konta (każdy zastosowany adres URL zapewnia tylko jednokrotne wykonanie akcji); Brak weryfikacji konta w ustalonym limicie czasu, z wartością określaną jako parametr uruchomieniowy odczytywany z deskryptora wdrożenia lub innego pliku konfiguracyjnego (domyślna wartość 24 godziny), musi powodować automatyczne usunięcie konta z powiadomieniem e-mail wysyłanym na adres pocztowy przypisany do usuwanego konta. Przypadek użycia usuwania konta należy przypisać w dokumentacji do aktora System, który nie jest nowym poziom dostępu jaki może być przypisywany do kont użytkowników. Należy uwzględnić możliwość zaginięcia wysłanej wiadomości e-mail z jednorazowym kodem/adresem URL poprzez ponowienie wysłania wiadomości email z kodem lub adresem URL umożliwiającym aktywację konta w połowie ustalonego limitu czasu jeżeli tylko do tego czasu konto nie zostało aktywowane przez użytkownika.
    2. przydzielenie i odebranie przez administracyjny poziom dostępu (tzw. administratora) wybranego poziomu dostępu dla konta w dowolnym momencie po utworzeniu aktywnego konta; każda zmiana poziomu dostępu przypisanego do konta wymaga powiadomienia o odebraniu lub przyznaniu poziomu dostępu w wiadomości e-mail przesłanego niezwłocznie na adres e-mail przypisany do konta użytkownika.
    3. modyfikację wybranych danych personalnych/biznesowych przez administratora jak i samego użytkownika, stosownie do założeń biznesowych projektu; W sytuacji zmiany adresu e-mail przypisanego do konta musi nastąpić jej autoryzacja poprzez indywidualny adres URL udostępniony jako hiperłącze przesłane na adres email przypisany do konta (każdy zastosowany adres URL zapewnia tylko jednokrotne wykonanie akcji). Należy uwzględnić możliwość zaginięcia wysłanej wiadomości e-mail z jednorazowym adresem URL poprzez zapewnienie użytkownikowi ponowienia wysłania wiadomości e-mail. Dodatkowo musi być wysyłane powiadomienie na stary adres e-mail z informacją o nowym adresie e-mail z hiperłączem aktywującym przywrócenie ustawienia poprzedniego adresu e-mail dla konta użytkownika. W ramach modyfikacji danych konta przez administratora i właściciela należy zapewnić również możliwość zmiany hasła, przy czym zmiana hasła przez administratora musi umożliwiać ograniczone czasowo bezpośrednie ustalenie hasła przez właściciela konta.
    4. blokowanie i odblokowywanie możliwości uwierzytelnienia się na dane konto bez ingerencji w dane konta. Każde zablokowanie lub odblokowanie konta wymaga powiadomienia o odebraniu lub przyznaniu poziomu dostępu w wiadomości e-mail przesłanego niezwłocznie na adres e-mail przypisany do konta użytkownika.
    5. automatyczne blokowanie konta na ustalony czas (limit ustalany jako parametr w deskryptorze wdrożenia lub innym pliku konfiguracyjnym, domyślna wartość 24 godziny) po ustalonej liczbie kolejnych nieudanych prób uwierzytelniania na konto użytkownika (limit ustalany jako parametr w deskryptorze wdrożenia lub innym pliku konfiguracyjnym, domyślna wartość 3) z wysyłanym powiadomieniem e-mail o zablokowaniu konta na adres e-mail przypisany do konta. Po upływie ustalonego limitu czasu zablokowane wcześniej konto użytkownika musi zostać odblokowane.
    6. resetowanie hasła przypisanego do konta z wykorzystaniem jednorazowego adresu URL udostępnionego jako hiperłącze przesłane na adres email przypisany do konta użytkownika, adres URL musi prowadzić do formularza zmiany hasła przypisanego do konta (zastosowany adres URL musi mieć określony okres ważności nie przekraczający 20 minut); Resetowanie hasła nie może być realizowane dla konta użytkownika jeszcze niezweryfikowanego (brak aktywacji konta) lub dla zablokowanego konta użytkownika. Należy uwzględnić możliwość zaginięcia wysłanej wiadomości e-mail z adresem URL oraz odwrotną kolejność otrzymania wiadomości e-mail w wyniku kilkukrotnego wywołania resetu hasła przez użytkownika.
3. Aplikacja musi udostępniać możliwość przydzielenia konta użytkownika jednocześnie do przynajmniej dwóch różnych poziomów dostępu (aktorów) bez uwzględniania poziomu dostępu gość. Przydział poziomów dostępu do konta nie może wprowadzać możliwości nadużyć ze strony użytkownika dysponującego takim kontem. Implementacja wielodostępnej aplikacji nie może być zależna od liczby poziomów dostępu (np. dodanie kolejnego poziomu dostępu nie może wymagać przebudowy aplikacji).
4. Przypadki użycia związane z zarządzaniem kontami/grupami użytkowników muszą zostać zaimplementowane w wydzielonym module funkcjonalnym. W dalszej części dokumentu moduł ten będzie określany akronimem MOK (Moduł Obsługi Kont).

5. Dla każdego zbioru danych biznesowych wyświetlonych w widoku GUI wymagane zapewnienie akcji odświeżenia wyświetlonych danych, po wykonaniu której wyświetlone dane biznesowe będą zgodne z aktualnym stanem tych danych przechowywanych w relacyjnej bazie.
6. Każda akcja inicjująca operację nieodwracalną zmieniająca dane biznesowe przechowywane w relacyjnej bazie danych wymaga wykonania potwierdzenia przez użytkownika, jeszcze przed wykonaniem akcji. Operacja nieodwracalna to operacja, dla której nie można z poziomu interfejsu użytkownika wykonać operacji odwrotnej (pozwalającej przywrócić zmiany danych w bazie relacyjnej).

Wymagania ilościowe

Wymagania główne

1. Każdy z uczestników projektu musi być autorem co najmniej dwóch przypadków użycia w modułach innych niż MOK, przy czym scenariusze główne muszą prowadzić do trwałej zmiany danych w bazie danych oraz przynajmniej jeden przypadek użycia musi dotyczyć operacji wykonywanych na związkach między encjami.
    1. Przez autorstwo przypadku użycia rozumie się przygotowanie wszystkich niezbędnych widoków GUI w aplikacji internetowej SPA oraz komponentów/metod komponentów aplikacji sieciowej REST oraz zapewnienie wszystkich transferów danych pomiędzy elementami wielodostępnego systemu, które są niezbędne dla zrealizowania danego przypadku użycia.
    2. Jeżeli komponent lub metoda są wykorzystywane przez wiele przypadków użycia (może to dotyczyć np. komponenty repozytoriów, encji JPA, obiektów transferujących dane), autorzy tych przypadków użycia współodpowiadają za poprawność wykonania współdzielonych elementów aplikacji.
2. Przypadki użycia składające się na MOK są traktowane jako wspólne dzieło całej grupy i nie są uwzględniane do spełniania przez uczestnika projektu powyższego wymagania.
3. Ponadto realizowany projekt musi zawierać:
    1. podział użytkowników na przynajmniej 4 aktorów (w tym 3 uwierzytelniane i autoryzowane (zgodnie z modelem kontroli dostępu bazującego na rolach RBAC) poziomy dostępu, a nieuwierzytelniony poziom dostępu to gość), przy czym przyjęty model poziomów dostępu nie może być hierarchiczny (kolejni aktorzy nie mogą jedynie rozszerzać funkcjonalności innych).
    2. przynajmniej 5 transakcji nietrywialnych z czego co najmniej 3 wykorzystują mechanizm blokad optymistycznych bazujący na polu wersji encji;
    3. przynajmniej 8 związków między różnymi typami obiektów encji JPA;

Pozostałe wymagania

1. Należy zauważyć, że istotą realizowanego projektu nie jest tworzenie rozbudowanych systemów informatycznych. W związku z tym w aplikacji liczba przypadków użycia nie powinna przekroczyć dwukrotnie ograniczenia minimalnego.

Wymagania dokumentacji projektu

Wymagania główne

1. Dokumentacja projektu ma postać strukturalnego dokumentu Markdown z rozszerzeniami GitLab, przechowywanego w dedykowanym repozytorium.
2. Szczegółowy sposób tworzenia stron dokumentacji na podstawie wzorca jest opisany w samym wzorcu (tzw. makieta dokumentacji).
3. Dokumentacja projektu musi zachowywać strukturę dostarczonej makiety, w szczególności dotyczy to strony tytułowej, hierarchii i nazw rozdziałów.
4. Przedstawione w dokumentacji sformułowania muszą być precyzyjne i jednoznaczne w interpretacji.
5. Dokumentacja musi być przygotowana w języku polskim, w dokumentacji obowiązuje oficjalnie słownictwo, należy używać właściwych pojęć, wymagane jednolite formatowanie w całym dokumencie.
6. Każdy uczestnik zespołowego projektu SSBD jest zobowiązany do tworzenia dokumentacji projektu wyłącznie z wykorzystaniem indywidualnego konta GitLab.

Pozostałe wymagania

1. Treść sprawozdań musi być kompletna i spójna w ramach całego projektu, w szczególności dotyczy to używanego nazewnictwa.
2. Wszędzie tam, gdzie to możliwe, należy posługiwać się nazwami powszechnie uznanymi lub zdefiniowanymi w niniejszym dokumencie, w rozdziale opisującym podstawowe pojęcia i założenia architektury.
3. Jeżeli konieczne jest zdefiniowanie własnych pojęć, dokumentacja projektu musi zawierać słownik tych pojęć i musi on obowiązywać w ramach całego dokumentu.
4. Dokument nie powinien zawierać błędów (m.in. ortograficznych, literówek, interpunkcyjnych), szczególnie jeżeli są one sygnalizowane przez edytor.
5. Zachowane muszą być podstawowe zasady edytorskie.

Dokumentacja projektu końcowego

Oddając sprawozdanie końcowe grupa zobowiązana jest, oprócz umieszczenia dokumentacji na platformie edukacyjnej WIKAMP, dostarczyć archiwum ZIP zawierające kod źródłowy aplikacji internetowej i aplikacji sieciowej, które po wydaniu jednego polecenia powodują uruchomienie stworzonego systemu wielodostępnego w kontenerach Docker/Podman. W osobnym kontenerze musi zostać utworzona relacyjna baza danych osadzona w PostgreSQL, aplikacja sieciowa musi być uruchomiona w innym kontenerze oferując usługę REST i komunikując się z relacyjną bazą danych. Wysłane do oceny archiwum ZIP musi zawierać katalogi wraz z zawartością wymienioną w poniższej tabeli.

Katalog	**Opis zawartości**
access	Plik z danymi dostępowymi dla istniejących kont w utworzonym systemie informatycznym, osobno dla każdego poziomu dostępu. Wymagane jawne podanie wszystkich poświadczeń niezbędnych w procesie uwierzytelniania dla każdego konta użytkownika z przypisanym innym poziomem dostępu (plik o nazwie dane_*dostepowe.txt*).
db	Aktualny zapis struktur relacyjnej bazy danych w formie kwerend DDL, bez przykładowych danych (plik o nazwie *baza_struktury.sql*) oraz niezbędne dane inicjujące bazę relacyjną w formacie poleceń DML (plik o nazwie *baza_dane_init.sql*). Każdy z plików musi zawierać kwerendy SQL.
doc	Plik INSTRUKCJA.pdf z instrukcją uruchomienia stworzonego systemu wielodostępnego, musi zawierać opis wszystkich czynności jakie należy wykonać przed uruchomieniem skryptu run.sh (w tym generowanie wymaganych certyfikatów i kluczy).
conf	Wszystkie pliki konfiguracyjne i moduły, sterowniki niezbędne do uruchomienia aplikacji w domenie serwera aplikacyjnego wraz z konfiguracją wykorzystywanych usług zewnętrznych np. serwera poczty elektronicznej.
src_app_net	Pełna i ostateczna wersja kodu źródłowego aplikacji sieciowej wyeksportowana z repozytorium z wcześniej utworzonej wersji tag do utworzonego w src podkatalogu. Kod źródłowy wyeksportowany z repozytorium nie może zawierać żadnych dodatkowych informacji w odróżnieniu od lokalnej wersji roboczej. Nazwa utworzonego podkatalogu musi odpowiadać nazwie wersji tag, która jest oddawana.
src_app_web	Pełna i ostateczna wersja kodu źródłowego aplikacji internetowej wyeksportowana z repozytorium z wcześniej utworzonej wersji tag do utworzonego w src podkatalogu. Kod źródłowy wyeksportowany z repozytorium nie może zawierać żadnych dodatkowych informacji w odróżnieniu od lokalnej wersji roboczej. Nazwa utworzonego podkatalogu musi odpowiadać nazwie wersji tag, która jest oddawana.
run	Skrypt powłoki bash run.sh automatyzujący budowanie obu aplikacji i uruchomienie stworzonego wielodostępnego systemu informatycznego w wersji finalnej we współczesnym systemie operacyjnym GNU/Linux. Skrypt powłoki musi automatyzować tworzenie i konfigurację wszystkich niezbędnych zasobów do uruchomienia stworzonego wielodostępnego systemu informatycznego (w tym wszystkich niezbędnych elementów środowiska uruchomieniowego).
test	Kod wszystkich utworzonych testów wraz z raportem RAPORT.pdf potwierdzającym pozytywny wynik wykonania każdego z przygotowanych testów

UWAGA: Przed uruchomieniem stworzonego systemu wielodostępnego do relacyjnej bazy danych muszą zostać wczytane niezbędne struktury i dane. W przypadku zastosowania statycznych struktur relacyjnej bazy danych żadne z kont bazodanowych wykorzystanych przy nawiązaniu połączeń JDBC do relacyjnej bazy danych w komunikacji aplikacji sieciowej z relacyjną bazą danych nie może posiadać uprawnień do modyfikacji struktur danych (brak możliwości wykonania kwerend DDL: CREATE, ALTER i DROP).

Wymagania kodu źródłowego i wersjonowania

Wymagania główne

1. Kod źródłowy aplikacji musi być przechowywany w przydzielonych grupie repozytoriach:
    1. db - repozytorium dedykowane do wersjonowania struktur bazy danych
    2. doc - repozytorium dedykowane do wersjonowania dokumentacji
    3. api - repozytorium dedykowane do wersjonowania kodu aplikacji sieciowej
    4. web - repozytorium dedykowane do wersjonowania kodu aplikacji internetowej
2. W wytworzonym przez uczestników projektu zespołowego kodzie źródłowym obowiązuje jednolitość wykorzystywanych rozwiązań i stosowanie wzorców projektowych.
3. Niedopuszczalna jest sytuacja, w której grupa rozwija kod źródłowy korzystając z własnych rozwiązań wersjonowania, zaś przydzielone grupie repozytorium jest uaktualniane tylko na potrzeby oddania projektu.
4. Członkowie grupy są zobowiązani wprowadzać zmiany do repozytorium kodu źródłowego posługując się własnymi, indywidualnie przydzielonymi kontami.
5. Na etapie szczegółowym projektu każdy członek grupy musi być autorem przynajmniej trzech zmian w kodzie implementującym MOK, wprowadzanym do repozytorium, przy czym zmiany muszą dotyczyć wprowadzenia różnych rozwiązań technologicznych do projektu.
6. Na etapie końcowym każdy członek grupy musi być autorem kompletnej implementacji własnych przypadków użycia.
7. Niedopuszczalne jest sztuczne zawyżanie statystyk przyrostu kodu na repozytorium poprzez wprowadzanie informacji nie mających znaczenia dla budowy aplikacji. Wykrycie takiej próby stanowi podstawę do dyskwalifikacji projektu dla autora takich zmian w kodzie.
8. Każda zatwierdzona w repozytorium nowa wersja kodu musi posiadać precyzyjny i komplentny opis wykonanych zmian.
9. Łączenie kodu w gałęzi głównej powinno odbywać się z wykorzystaniem mechanizmu *merge request* dostępnego w platformie GitLab. Żądanie takie zgłasza programista, który zrealizował i zatwierdził w gałęzi implementacyjny przypadek użycia, zgłaszając przy tym do przeglądu kodu (*review*) osobę lub osoby pełniące funkcję *kontrola jakości*. Złączenie kodu jest możliwe dopiero po pozytywnej opinii kontrolujących.
10. Kolejne stabilne (złączone i przetestowane funkcjonalnie) wersje projektu są umieszczane w głównej linii rozwojowej (domyślnej gałęzi) repozytorium jako wersje oznaczone (otagowane)
    1. Zakres zaimplementowanej funkcjonalności w kolejnych wersjach stabilnych powinien rosnąć względem wymagań dokumentacji.
    2. Wersje stabilne oznaczamy zgodnie z zasadami wersjonowania semantycznego 2.0 odpowiednim znacznikiem (np „v0.3.14”), przy czym wersja „v1.0” powinna zawierać pełną funkcjonalność aplikacji, a następne wersje mogą jedynie dotyczyć poprawek błędów lub zmian w implementacji.
    3. Umieszczane wersje kodu muszą zawierać pełną strukturę katalogów i plików niezbędną do otworzenia kodu aplikacji jako projekty w środowisku IDE.
    4. Umieszczane wersje kodu powinny być pozbawione wszelkich zbędnych plików.
    5. Przed oddaniem projektu wstępnego, szczegółowego i sprawozdania końcowego *w każdym z repozytoriów* musi zostać utworzona odpowiadająca mu wersja, zawierająca właściwy mu stan kodu projektu, oznaczona nazwą zawierającą prefiks „W” dla etapu wstępnego, „S” dla szczegółowego, oraz „K” dla końcowego. Zatem przykładowa nazwa wersji kodu projektu „Sv0.4.3” dotyczy oddania projektu na etapie szczegółowym.

Pozostałe wymagania

1. Kod źródłowy aplikacji powinien być sformatowany zgodnie z jednym z powszechnie obowiązujących standardów.
2. Zaleca się korzystanie z edytora kodu źródłowego oferującego automatyczne formatowanie kodu.
3. Nazewnictwo pakietów, klas, składowych musi być zgodne z przyjętym słownikiem.
4. Nazwa grupy i artefaktu Maven, a co za tym idzie bazowa część nazwy pakietów klas, powinny być skonstruowane według wzoru: *pl.lodz.p.it.ssbd<rok>.ssbdXX*, gdzie <rok> - bieżący rok w formacie czterocyfrowym, XX – dwucyfrowy numer grupy.

Podział aplikacji sieciowej na moduły

Wymagania główne

1. Poszczególne klasy wchodzące w skład aplikacji sieciowej muszą być zorganizowane w odrębne pakiety klas Javy:
    - pakiet klas encji JPA (sugerowana nazwa pakietu: *entities*)
    - pakiet klas wyjątków (sugerowana nazwa pakietu: *exceptions*)
    - odrębny pakiet klas komponentów dla każdego modułu funkcjonalnego (sugerowana nazwa pakietu: *akronim modułu funkcjonalnego zapisany małymi literami*)
    - w razie potrzeby – pakiet pozostałych niezbędnych klas (sugerowana nazwa pakietu: *utils*)
2. W ramach każdego z wymaganych pakietów można stosować dalszy podział na podpakiety (sugerowany podział zgodnie z odpowiedzialnością komponentów np. kontrolery, usługi, repozytoria).
3. Wytwarzane oprogramowanie (w tym także każdy artefakt Maven) musi być oznaczone wersją nadaną zgodnie z zasadami wersjonowania semantycznego 2.0.
4. Żadna z klas pakietu odpowiadającego modułowi funkcjonalnemu nie może odwoływać się do żadnej z klas innego pakietu odpowiadającego modułowi funkcjonalnemu. Pakiety modułów funkcjonalnych nie mogą od siebie zależeć.
    - Dla spełnienia tego wymogu dopuszczalne jest częściowe powtarzanie się kodu komponentów w poszczególnych modułach.

Wymagania systemu rejestracji zadań

Wymagania główne

1. Każdy uczestnik projektu posiada w systemie rejestracji zadań indywidualne konto.
2. Zabrania się wykorzystywania konta innego uczestnika projektu.
3. Wszystkie ustalenia dotyczące projektu muszą być umieszczone w systemie rejestracji zadań.
4. Szef projektu jest zobowiązany do umieszczania zadań (najpóźniej 24 godziny po zakończeniu zajęć), precyzyjnie określających zakres prac dla poszczególnych uczestników projektu (w tym też swoich) w postaci zadań (issue) w udostępnionym systemie rejestracji zadań.
5. Pozostali uczestnicy projektu także mogą tworzyć zadania, w szczególności jest to wymagane w sytuacji ustalenia zakresu prac nad projektem w ramach przeprowadzonych uzgodnień, wykrycia zaistniałego błędu. Wszystkie niezgodności i wykryte problemy muszą być zgłaszane jako zadania utworzone w systemie rejestracji zadań (dot. w szczególności osób z funkcją kontrola jakości lub kontrola zgodności).
6. Zakres prac określony w zadaniu może być związany z projektowaniem i implementacją aplikacji, przygotowaniem dokumentacji, konfiguracji (np. względem funkcji uczestnika projektu) czy też zmian w poszczególnych składowych projektu, testowaniem nowych rozwiązań technologicznych itd.
7. Zgłoszone zadanie musi być przypisane do właściwego etapu projektu zdefiniowanego we wstępnej konfiguracji systemu rejestracji zadań jako indywidualne wydania (milestone), do kategorii (epic) odpowiadającej charakterowi zadania oraz w przypadku prac nad rozwojem oprogramowania, prac nad rozwojem dokumentacji oraz prac związanych z realizowaną funkcją - przynajmniej jedną etykietą (label) określającą dokładniej obszar działania.
    1. Jeżeli zadanie dotyczy projektowania lub implementacji konkretnego przypadku użycia wymagane jest przypisanie etykiety, której nazwa zawiera akronim modułu funkcjonalnego i numeru przypadku użycia względem podanego modułu (separatorem jest znak kropki). Obowiązuje użycie w pierwszej kolejności etykiet predefiniowanych, można także tworzyć dodatkowe etykiety np. dla kolejnych przypadków użycia.
    2. W przypadku projektowania lub implementacji związanej z realizacją rozszerzeń, wymagane przypisanie do każdego zadania etykiety z nazwą zgodną z identyfikatorem rozszerzenia z katalogu rozszerzeń.
8. Każde zadanie musi posiadać przypisanego wykonawcę, termin wykonania oraz precyzyjny opis zakresu prac. W przypadku złożonych zadań można wydzielać podzadania, które zawsze muszą mieć określonego wykonawcę i termin wykonania.
9. Zakończone zadanie implementacyjne musi posiadać przypisaną rewizję kodu z repozytorium. Jeżeli rewizji jest kilka należy przypisać do zadania wiele rewizji, dokumentujących całość wykonanych prac implementacyjnych.
10. Każde wykonanie opisanych w zadaniu prac wymaga jego zamknięcia, które potwierdza zrealizowanie zadania.
11. Potwierdzaniem wykonania zadań zajmuje się szef grupy. Potwierdzenie wykonania zadania jest realizowane poprzez jego zamknięcie. Szef grupy jest zobowiązany do weryfikacji wykonania wszystkich najpóźniej 3 godzin przed rozpoczęciem zajęć. W przypadku nie wykonania zadania szef grupy może zmienić jego termin lub przypisać zadanie innemu uczestnikowi projektu.
    1. W przypadku zadań wiążących się z rozwojem kodu w repozytorium można skorzystać z opcji automatycznego zamykania zadań po pomyślnym zrealizowaniu żądania złączenia (merge request).
12. Przed każdymi zajęciami musi powstać nowa wersja projektu Maven, zawierająca scalony kod z wszystkich wykonanych zadań implementacyjnych. Wersja ta powinna być uruchomiona w domenie serwera aplikacyjnego (wymóg obowiązuje od etapu szczegółowego projektu i dotyczy serwera aplikacyjnego zlokalizowanego w wirtualnej maszynie w platformie oVirt) i dostępna poprzez adres URL przypisany do projektu w systemie rejestracji zadań.
13. Wszystkie zgłoszone przez szefa grupy zadania w systemie rejestracji zadań muszą mieć jawnie zdefiniowany termin wykonania jako datę (dopuszczalny zakres wartości od 1 do 4 dni)
14. Czas wykonania zadania jest liczony od momentu zdefiniowania terminu.
15. Jeżeli zadanie dotyczy przygotowania lub zmian w dokumentacji projektu, wówczas konieczne jest zastosowanie etykiety związanej z danym rozdziałem sprawozdania.
16. Niewykonane zadanie, któremu upłynął termin szef grupy może przypisać innej osobie, ustalając nowy termin wykonania dla zmienionego zadania.

Pozostałe wymagania

1. Wszystkie prace do wykonania w ramach przypisanych funkcji muszą być rejestrowane przez szefa grupy jako zadania.
2. Po oddaniu projektu szczegółowego przed każdymi zajęciami musi być tworzona kolejna wersja stabilna projektu prezentująca dotychczas wykonaną implementację. Dla ostatniej stabilnej wersji projektu wytworzonego przed zajęciami należy zapewnić uruchomienie systemu informatycznego w ramach infrastruktury udostępnionej grupie.
3. Każda kolejna stabilna wersja wielodostępnego systemu musi zapewniać możliwość skompilowania i zbudowania archiwum, a po osadzeniu na serwerze aplikacyjnym aplikacja sieciowa musi udostępniać dotychczas zaimplementowaną funkcjonalność.

Wymagania technologiczne dla poszczególnych elementów wielodostępnego systemu informatycznego.

Warstwa danych

Struktura bazy danych - wymagania główne

1. Struktura relacyjnej bazy musi być statyczna i spełniać wymagania trzeciej postaci normalnej (3NF). Niedopuszczalna jest redundancja danych.
2. Wszystkie tabele bazodanowe muszą być utworzone w jednym schemacie (przestrzeni nazw).
3. Każda tabela bazodanowa musi zawierać co najmniej:
    - jawnie zdefiniowane pole klucza głównego
    - liczbowe pole wersji (na potrzeby blokad optymistycznych warstwy logiki)
4. Poszczególne kolumny każdej tabeli muszą mieć właściwie dobrane typy danych i określone wszystkie ograniczenia i modyfikatory (not null, unique, check, default) definiujące poprawność danych, zgodnie ze specyfikacją wielodostępnego systemu.
5. W szczególności odpowiednie ograniczenia muszą być nałożone na pola będące kluczami obcymi spinającymi relacje. Ograniczenia te powinny odwzorowywać interpretację związku w modelu obiektowym danych zastosowanym w realizowanej aplikacji sieciowej, np. należy rozważyć czy może występować wielokrotna relacja łącząca użytkownika i poziom dostępu.
6. Dla każdej tabeli muszą zostać określone uprawnienia dostępu dla poszczególnych użytkowników bazodanowych, zgodnie z modelem kontroli dostępu aplikacji.
7. Struktury bazy danych muszą być niezmienne w trakcie działania aplikacji.
8. Właścicielem struktur baz danych musi być konto bazodanowe utworzone przez grupę, o nazwie ssbdXXadmin. Konto to nie może być wykorzystywane w połączeniach aplikacji do bazy danych.
9. Zestaw nadanych użytkownikom bazodanowym uprawnień musi być minimalny, niezbędny dla funkcjonowania aplikacji sieciowej.
10. Każdy moduł funkcjonalny musi wykorzystywać indywidualne konta dostępu do bazy danych.
11. Konto bazodanowe nie może być wykorzystywane przez dwa różne moduły funkcjonalne.
12. Wartość klucza głównego dla istniejącej krotki nie może być modyfikowana i nie może reprezentować żadnych informacji biznesowych, związanych z funkcjonalnością aplikacji.
13. Niedopuszczalne jest korzystanie z takich mechanizmów systemu zarządzania bazami danych, które nie są objęte podstawowym standardem SQL (takich jak procedury składowane, wyzwalacze/triggery, tabele dziedziczone itp.), jak również z mechanizmów kaskad oferowanych przez DBMS.
14. Jeżeli w projekcie występuje dana biznesowa, mająca ograniczony i z góry znany zbiór wartości (np. możliwe statusy zamówienia), to zbiór wartości musi być reprezentowany w bazie danych w postaci tabeli słownikowej.
15. Żadne z haseł nie może być przechowywane w w relacyjnej bazie w postaci jawnej. Identyczny wymóg obowiązuje inne dane poufne, o ile takie informacje występują w modelu dziedzinowym.
16. Stan początkowy relacyjnej bazy danych musi być generowany wyłącznie z wykorzystaniem narzędzia Flyway/Liquibase. Wymagane wersjonowanie zmian w strukturach relacyjnej bazy danych z wykorzystaniem jednego z wymienionych narzędzi.

Struktura bazy danych - pozostałe wymagania

1. Dla kluczy obcych należy utworzyć indeksy, usprawniające odszukiwanie informacji względem wartości klucza obcego.
2. Reprezentacja informacji przechowywanej w bazie danych musi mieć poprawnie dobrane typy danych (wykorzystanie przestrzeni składowania), w szczególności wymóg ten dotyczy przechowywania skrótu z hasła przypisanego do konta budowanej aplikacji.
3. Jeżeli zgodnie z przyjętym modelem biznesowym informacja w danym polu krotki ma charakter niezmienniczy (po zainicjowaniu krotki wartość tego pola nie zmienia się) należy to odwzorować w klasie encyjnej JPA. Podobnie należy odwzorować sytuację, gdy pole krotki powinno pozostać nie wypełnione w trakcie jej wstawiania, natomiast istnieje możliwość późniejszego ustalenia wartości tego pola.

Reprezentacja danych w aplikacji - wymagania główne

1. Dostęp do warstwy danych ma zostać zrealizowany poprzez użycie klas encji (ang. *entity classes*, *entities*), będących obiektową reprezentacją danych znajdujących się w bazie, oraz interfejsu JPA (Java Persistence API) odpowiedzialnego za przenoszenie operacji wykonywanych na obiektach encji na odpowiednie kwerendy SQL.
2. Niedopuszczalne jest bezpośrednie wykonywanie kwerend języka SQL w bazie danych,
3. Dopuszczalne jest posługiwanie się kwerendami języka JPQL (ang. Java Persistent Query Language) operującego na obiektach encji JPA.
4. Dane powiązane z funkcjonalnością aplikacji muszą być przekazywane pomiędzy komponentami aplikacji sieciowej uczestniczącymi w transakcji aplikacyjnej w postaci encji JPA.
5. Modyfikacja kluczy obcych może być zrealizowana wyłącznie poprzez wiązanie obiektów encji JPA związkami.
6. Modyfikacja wartości kluczy głównych encji JPA jest niedopuszczalna.

Reprezentacja danych w aplikacji - pozostałe wymagania

1. Opis klas encji JPA za pomocą adnotacji musi dokładnie odwzorowywać struktury bazy danych uwzględniając ograniczenia nakładane na kolumny.
2. W szczególności wszystkie związki zdefiniowane pomiędzy obiektami encji JPA muszą posiadać zdefiniowaną stronę właściciela, liczność (1:1,1:N,N:1,N:M), kierunkowość (jedno lub dwukierunkowe) oraz obsługiwane operacje kaskadowe.

Warstwa logiki biznesowej

Wymagania główne

1. Warstwa logiki biznesowej dla projektów musi być zbudowana w oparciu o komponenty Spring Framework.
2. Dla każdej metody udostępnianej przez komponent musi być określony zbiór ról, które mają prawo jej wykonania, przy czym można to zrealizować wyłącznie w postaci stosownych adnotacji bezpieczeństwa deklaratywnego (ang. declarative security). Niedopuszczalne jest korzystanie z możliwości sprawdzenia roli w kodzie metody (ang. programmatic security).
3. Zabronione jest wykorzystywanie tzw. antywzorca OpenSessionInView (wymagane jawne wyłączenie, ponieważ jest on domyślnie włączony w Spring).
4. Wykonywanie transakcji związanych z operacjami na obiektach encji musi być realizowane poprzez kontener Spring Framework z użyciem SpringTransactions, w strategii CMT (ang. *Container Managed Transactions*). Każda transakcja musi mieć określony limit czasu realizacji, po upływie którego zostanie automatycznie odwołana.
5. W związku z tym każda metoda komponentu musi być opatrzona bezpośrednio lub pośrednio stosowną adnotacją określającą sposób wykorzystywania przez metodę kontekstu transakcyjnego, chyba, że domyślna wartość tej adnotacji jest odpowiednia dla metody.
6. Niedopuszczalne jest korzystanie z transakcji zarządzanych przez komponent w strategii BMT (ang. *Bean Managed Transactions*).
7. Dla każdej transakcji bazodanowej musi być prawidłowo (adekwatnie do operacji wykonywanych w ramach transakcji) określony najniższy możliwy poziom izolacji transakcji oraz jawne zadeklarowanie transakcji jako prowadzącej tylko odczyt albo odczyt i zapis.
8. W aplikacji warstwy logiki mechanizm blokad optymistycznych nie może bazować wyłącznie na jawnej wartości pola wersji obiektu encji przekazanego z warstwy widoku jako parametr metody biznesowej, wymagane zapewnienie mechanizmu kontrolującego wiarygodność wersji dla mechanizmu blokad optymistycznych.
9. Należy zwrócić uwagę na wyjątki zgłaszane w trakcie wykonania transakcji aplikacyjnych, i ich wpływ na ewentualne jej odwołanie.
    1. W sytuacji wystąpienia wyjątku związanego z logiką biznesową aplikacji, należy zgłosić weryfikowalne wyjątki aplikacyjne/biznesowe.
    2. Niedopuszczalne jest pozostawienie wyjątku bez obsługi (minimum: informacja dla użytkownika i wpis w dzienniku zdarzeń o wystąpieniu błędu).
10. Co najmniej w przypadku przypadków użycia realizujących funkcjonalność krytyczną i prowadzących do modyfikacji danych w ramach transakcji wymagana jest analiza przypadków, w których transakcja może zostać odwołana lub nie ukończona, i zaimplementowanie mechanizmu powtarzania transakcji.
    Powtórzenie transakcji jest zasadne, jeżeli:
    1. została ona odwołana na skutek wystąpienia wyjątku, przy czym powody wystąpienia wyjątku mają charakter tymczasowy i nie związany z ograniczeniami modelu biznesowego,
    2. transakcja nie została ukończona z powodów innych niż rzucenie wyjątku (np. przekroczenie limitu czasu wykonywania transakcji)
11. Limit prób ponowienia wykonania transakcji musi określać parametr inicjalizacyjny zdefiniowany w deskryptorze wdrożenia lub innym pliku konfiguracyjnym. Jeżeli każde z ponowień wykonania transakcji aplikacyjnej określone poprzez ustalony limit zostanie zakończone odwołaniem transakcji, to należy zapewnić wyświetlenie komunikatu w interfejsie użytkownika, który zainicjował akcję o niepowodzeniu wykonania akcji.
12. Aplikacja musi zgłaszać zdarzenia do dzienników zdarzeń (tzw. logi) uwzględniające każde wywołanie (wraz z wartościami parametrów) oraz każde zakończenie (wraz z wartością zwracaną lub zgłoszonym wyjątkiem) metody komponentów warstwy logiki, z uwzględnieniem tożsamości użytkownika oraz odpowiednim znacznikiem czasowym.
    1. Jeżeli parametrem lub wartością zwracaną metody jest obiekt encji (ew. ich kolekcja), wówczas należy zapisać dla każdego obiektu jej tożsamość (np. identyfikator) wraz z bieżącym numerem wersji.
    2. Dodatkowo generowane przez aplikację wpisy dziennika zdarzeń muszą uwzględniać rozpoczęcie i zakończenie każdej transakcji aplikacyjnej wraz z rezultatem (zatwierdzenie bądź odwołanie), przy czym odpowiednie wpisy w dzienniku zdarzeń muszą uwzględniać tożsamość użytkownika oraz identyfikację transakcji umożliwiając określenie jej granic.

Pozostałe wymagania

1. Wszystkie stałe wykorzystywane w warstwie logiki budowanej aplikacji należy zdefiniować w postaci parametrów inicjalizacyjnych, których wartości początkowe są zdefiniowane w odpowiednich deskryptorach konfiguracyjnych.

Warstwa prezentacji

Wymagania główne

1. Funkcjonalność aplikacji udostępniana w warstwie widoku musi być uzależniona od aktora, który zgłasza żądanie do aplikacji (z uwzględnieniem użytkownika nieuwierzytelnionego / niezalogowanego). *Dla zrealizowania tego założenia* można w warstwie widoku posłużyć się programowym identyfikowaniem aktora (ang. *programmatic security*), przy czym sprawdzenie to może być wykonywane tylko w jednym miejscu aplikacji, zaś dane umożliwiające identyfikację poszczególnych aktorów muszą być zapisane w plikach konfiguracyjnych aplikacji (np. jako parametry pobierane z plików XML).
2. W warstwie widoku wszelkie dane konfiguracyjne specyficzne dla tworzonej aplikacji internetowej należy umieszczać w deskryptorze wdrożenia lub oraz plikach konfiguracyjnych wybranego zbioru narzędzi (ang. framework).
3. Konfiguracja warstwy prezentacji musi być niezależna od kontekstu zawartego w adresie URL.
4. Wybrany zbiór narzędzi musi wspierać ujednolicony wygląd widoków GUI (wspólny nagłówek, stopka, menu itp), czyli tzw. szablon widoku chroniący przed wielokrotnym powtarzaniem kodu. Zabrania się przy tym wykorzystania ramek HTML. Poszczególne widoki udostępniające graficzny interfejs użytkownika muszą być ujednolicone w zakresie: zastosowanej kolorystyki, rozmiaru elementów, podziału oraz lokalizacji sekcji (m.in. menu, nagłówka, stopki, główna).
5. Do przekazywania danych pomiędzy warstwą prezentacji a warstwą logiki biznesowej, jak również do przechowywania danych w warstwie prezentacji nie można posługiwać się obiektami klas encyjnych JPA.
6. Poprawność danych przesyłanych przez przeglądarkę musi być weryfikowana przez aplikację (tzw. walidacja danych), przy czym ewentualna walidacja prowadzona przez samą przeglądarkę może być traktowana wyłącznie jako mechanizm pomocniczy ze względu na możliwość manipulacji danymi przez użytkownika.
7. Niezależnie od liczby identycznych żądań zgłoszonych przez przeglądarkę internetową (np. w wyniku odświeżenia/cofnięcia strony), aplikacji osadzonej na serwerze nie wolno ponownie wykonać zrealizowanych wcześniej akcji (w szczególności operacji bazodanowych).
8. Wykonanie każdej akcji prowadzącej do wprowadzenia, modyfikacji lub usunięcia informacji z punktu widzenia biznesowego (dotyczy to również zmian związków między danymi) musi być poprzedzone dodatkowym potwierdzeniem zamiaru wykonania tej akcji przez użytkownika. Wyjątki od tej zasady mogą dotyczyć wyłącznie czynności, które są odwracalne poprzez wykonanie akcji z poziomu interfejsu użytkownika bez konieczności posiadania wiedzy o poprzednim stanie danych (np. zablokowanie/odblokowanie).
9. Poprawne wykonanie każdej akcji prowadzącej do zmodyfikowania zawartości bazy danych musi być potwierdzone komunikatem prezentowanym w widoku GUI.
10. Informacja o tożsamości użytkownika oraz aktualnym i możliwych do uzyskania poziomie/poziomach dostępu musi być prezentowana w widoku GUI (sugerowana lokalizacja w widocznym nagłówku).
11. Nie można jednocześnie udostępniać w interfejsie użytkownika akcji z różnych poziomów dostępu przypisanych do konta. Interfejs użytkownika musi udostępniać możliwość przełączania się pomiędzy poziomami dostępu jakie są przypisane do konta użytkownika. Wymagane zróżnicowanie kolorystyki menu wyboru w interfejsie użytkownika dla różnych poziomów dostępu w celu czytelnego ich rozróżnienia.
12. Zmiana poziomu dostępu w ramach poziomów przypisanych do konta użytkownika nie może wymagać ponownego uwierzytelnienia.

Pozostałe wymagania

1. Warstwa prezentacji musi zapewniać możliwość zastosowania różnych wersji językowych względem preferencji przeglądarki internetowej, jednakże wymagane jest przygotowanie jedynie wersji polskiej. Dobór wersji językowej musi dotyczyć również dat i znaczników czasu, statusów odczytanych z bazy danych, nazw poziomów dostępu wyświetlanych w interfejsie użytkownika jak i treści wiadomości wysyłanych przez system na adres email użytkownika systemu.
2. Dla każdej innej wersji językowej w miejscach komunikatów muszą wyświetlać się odpowiadające im klucze. Wszystkie komunikaty muszą znajdować się w oddzielnym pliku/plikach zasobów (ang. bundle).
3. Wymieniony powyżej wymóg umieszczania komunikatów w plikach zasobów dotyczy także wszelkich komunikatów błędów oraz ostrzeżeń prezentowanych w interfejsie użytkownika przez aplikację. Komunikaty błędów nie mogą być wyświetlane w interfejsie użytkownika chwilowo, tzn. znikać samodzielnie.
4. Warstwa prezentacji powinna umożliwiać interaktywną identyfikację i wybór danych biznesowych (np. wybór produktu z listy). Wyjątki od tej zasady powinny być umotywowane wymaganiami funkcjonalnymi.
5. Jeżeli formularz aplikacji posiada pola, których wypełnienie jest obowiązkowe, powinny one być wyróżnione w interfejsie użytkownika (kolorem, znakiem gwiazdki (*) itp).
6. Każdy widok GUI aplikacji SPA powinien zawierać informację o realizowanej funkcjonalności (np. umieszczony tytuł) wraz z lokalizacją w interfejsie użytkownika (format ścieżki) rozpoczynając od widoku głównego danego poziomu dostępu tzw. ślad okruszków chleba (breadcrumb).

Kontrola spójności danych

Wymagania główne

1. Stworzony w ramach projektu system informatyczny musi być wiarygodny, cecha ta dotyczy informacji dostarczanych użytkownikowi systemu informatycznego, przetwarzanych danych oraz danych przechowywanych w bazie.
2. Wykonywanie dowolnych akcji udostępnianych przez interfejs użytkownika nie może prowadzić do naruszenia spójności danych przechowywanych w relacyjnej bazie.
3. W szczególności wymagane jest stosowanie cyfrowego podpisywania i weryfikowania podpisu, jeżeli system ze względu na brak możliwości utrzymywania stanu konwersacyjnego po stronie serwera musi polegać na danych przechowywanych w stanie konwersacyjnym klienta (np. przeglądarki), zaś modyfikacja tych danych powinna zachodzić jedynie w warstwie logiki.
4. Należy uwzględnić możliwość jednoczesnego modyfikowania tych samych danych w dwóch różnych sesjach użytkowników w wielodostępnym systemie informatycznym. Szczególną uwagę należy zwrócić na sytuacje konfliktów wynikających z akcji użytkowników: edycja/edycja, edycja/usunięcie i usunięcie/edycja realizowanych na wspólnych danych.
5. Podobnie należy kontrolować związki zawsze wymagające zestawienia oraz związki jednorazowego spięcia, które po zestawieniu nie mogą być zmieniane. Należy też uwzględnić próby ponownego nawiązywania już istniejących związków.
6. Wymagane wyświetlenie komunikatu w interfejsie użytkownika informującego o nieaktualnym stanie danych w sytuacji braku możliwości wykonania akcji ze względu na zmiany danych w bazie relacyjnej wykonane w innej sesji użytkownika.
7. Realizując funkcjonalność związaną z dynamiczną agregacją informacji o danych biznesowych przechowywaną w bazie należy uwzględnić, że wyliczanie nowej wartości agregatu może zostać zainicjowane jednocześnie w wielu wątkach aplikacji. Realizacja musi gwarantować, że wartości agregatu utrwalone w bazie są spójne z wartościami danych stanowiących podstawę wyliczania agregatu.

Wymagania bezpieczeństwa

Wymagania główne

1. Wykonanie dowolnej akcji udostępnianej przez warstwę widoku aplikacji, przy dowolnej zawartości formularzy aplikacji, nie może prowadzić do żadnej operacji bazodanowej nie zaplanowanej dla danej akcji (w szczególności modyfikacji/usunięcia danych oraz udostępnienia informacji normalnie niedostępnej dla użytkownika).
2. Użytkownik nie może posiadać możliwości wykonania akcji, do której nie jest uprawniony, lub też akcji do której jest uprawniony na obiekcie, do którego nie jest uprawniony (na przykład modyfikacja danych konta innego niż własne przez użytkownika nie będącego administratorem). Należy przy tym uwzględnić możliwość celowego manipulowania przez użytkownika zawartością formularzy lub innych danych przechowywanych przez przeglądarkę (na przykład klucza głównego edytowanego obiektu) w czasie ich wysyłania do aplikacji.
3. Komunikacja pomiędzy przeglądarką a serwerem aplikacji musi przebiegać z wykorzystaniem protokołu zapewniającego szyfrowanie (HTTPS) już w momencie uwierzytelnienia do aplikacji. W przypadku próby wymuszenia przez użytkownika korzystania z protokołu bez szyfrowania danych aplikacja musi odmówić wykonania akcji lub automatycznie przenieść komunikację na protokół zapewniający poufność transmisji. W przypadku, gdy aplikacja jest udostępniana przez serwer pośredniczący (proxy), zasady te dotyczą komunikacji między przeglądarką a serwerem pośredniczącym, zaś komunikacja między serwerem pośredniczącym a serwerem aplikacji może odbywać się bez szyfrowania, pod warunkiem, że wykorzystywana do tego sieć jest zabezpieczona przed nieuprawnionym dostępem do przesyłanych danych.
4. Wszelkie hasła przechowywane w bazie danych muszą mieć postać niejawną, a liczba znaków w jawnej postaci hasła nie może być mniejsza niż 8.
5. Zmiana hasła przez użytkownika będącego właścicielem konta powinna być realizowana pod warunkiem podania poprawnego dotychczasowego hasła tegoż konta (mimo iż użytkownik jest aktualnie uwierzytelniony).
6. Uwierzytelnianie użytkownika w aplikacji musi być realizowane przez warstwę prezentacji z wykorzystaniem wzorców poświadczeń przechowywanych w bazie relacyjnej. Po poprawnym uwierzytelnieniu rozpoczyna się sesja użytkownika, w trakcie trwania której kolejne akcje użytkownika nie mogą wymagać ponownego uwierzytelnienia użytkownika. Wymagane zapewnienie automatycznego przedłużenia sesji użytkownika jeżeli tylko wykonuje on kolejne akcje w czasie krótszym niż ustalony okres nieaktywności ustalony minimalnie na 5 minut.
7. Aplikacja musi udostępniać uwierzytelnionemu użytkownikowi możliwość zamknięcia sesji ("wylogowania"), przy czym wykonanie tej akcji musi doprowadzić do porzucenia zapamiętanych poświadczeń.
8. Każde rozpoczęcie sesji użytkownika poprzez uwierzytelnienie oraz zmiana poziomu dostępu w ramach sesji musi być odnotowana jako wpis w dzienniku zdarzeń zawierający informację o loginie i adresie logicznym IP identyfikującym lokalizację przeglądarki internetowej użytkownika w sieci Internet.
9. Wykonanie każdej nietrywialnej akcji musi być automatycznie odnotowane jako wpis w dzienniku zdarzeń zawierający szczegóły akcji i tożsamość z jaką akcja została wykonana w systemie informatycznym (np. poprzez login).
10. W ramach kontroli odpowiedzialności dla każdego obiektu encji JPA, należy zapewnić zachowanie w bazie danych dodatkowych informacji o utworzeniu i ostatnio wykonywanej zmianie z uwzględnieniem rodzaju operacji zmieniającej dane obiektu encji JPA, tożsamości użytkownika modyfikującego (jako relacja z kontem użytkownika utworzona z wykorzystaniem klucza obcego) oraz znacznika czasu wykonania zmiany (czas i data). Dozwolone jest indywidualne gromadzenie informacji o tożsamości użytkownika i znacznika czasu ostatniej wykonanej operacji osobno dla każdego rodzaju operacji zmieniających obiekt encji JPA. Wszystkie wymienione dodatkowe informacje o ostatnio realizowanych operacjach na obiekcie encji JPA muszą być udostępniane z poziomu interfejsu użytkownika.

Pozostałe wymagania

1. Każde uwierzytelnienie na konto posiadające przypisany administracyjny poziom dostępu wymaga przesłania powiadomienia o rozpoczęciu nowej sesji wraz z adresem logicznym IP na adres e-mail przypisany do tego konta.
2. Zmiana hasła przez użytkownika będącego właścicielem konta powinna być realizowana pod warunkiem, że nowo wprowadzone hasło nie jest identyczne jak poprzednie.

Zmiany wymagań

Zmiany w wymaganiach projektu są dopuszczalne jedynie pod następującymi warunkami:

1. Konieczność zmiany musi być uzasadniona okolicznościami niemożliwymi do wcześniejszego przewidzenia (choroby, wypadki losowe, rezygnacje członków grupy bądź awarie serwerów niezbędnych do prowadzenia projektów).
2. Zmiana zachodzi za obopólną zgodą prowadzącego i szefa grupy lub w zastępstwie wszystkich pozostałych członków grupy.
3. Zmiana jest dokonywana w formie pisemnej.
4. Zmiana jest podawana do publicznej wiadomości poprzez opublikowanie na platformie WIKAMP.