Wymagania SSBD'2020

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

Serwis:	WIKAMP FTIMS
Przedmiot:	Sieciowe Systemy Baz Danych 2020
Książka:	Wymagania SSBD'2020
Wydrukowane przez użytkownika:	dr inż. Mateusz Smoliński
Data:	czwartek, 5 marca 2020, 17:56

Spis treści

Wprowadzenie

Celem zajęć jest zaprojektowanie i zrealizowanie wielodostępnego system informatycznego złożonego z aplikacji zbudowanej w trójwarstwowej architekturze Java/Jakarta EE (ang. Enterprise Edition) współpracującej z relacyjną bazą danych i udostępniającej interfejs użytkownika poprzez dynamicznie generowane strony WWW.

Na całość projektu składają się:

- struktury relacyjnej bazy danych obsługiwanej przez DBMS,
- kod źródłowy aplikacji działającej w domenie serwera aplikacji,
- dokumentacja programistyczna (JavaDoc),
- 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,
- Makieta sprawozdania - dostępna w Confluence

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 można uzyskać z 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 50.
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 grupy całkowicie dyskwalifikuje projekt.

Maksymalna punktacja za poszczególne składowe projektu:

1. Sprawozdanie wstępne i realizacja zadań nim objętych – 20 pkt
2. Sprawozdanie szczegółowe i realizacja zadań nim objętych – 25 pkt
3. Sprawozdanie końcowe i realizacja zadań nim objętych – 15 pkt
4. Ocena poprawności działania finalnej wersji aplikacji – 15 pkt
5. Płyta DVD z finalną wersją aplikacji i dokumentacji – 5 pkt
6. Ocena za systematyczność pracy – 10 pkt
7. Ocena za aktywności dodatkowe – 20 pkt

Terminy i opóźnienia

Terminy

1. Zaliczanie projektu przebiega w trzech etapach odpowiadających kolejnym fazom powstawania systemu informatycznego (złożonego z aplikacji internetowej i 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.

Składowe projektu

1. Projekt składa się z następujących składowych podlegających ocenie:
    1. 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 instancji serwera aplikacyjnego, w pełni funkcjonująca aplikacja (dotyczy projektu na etapie końcowym i częściowo dla modułu funkcjonalnego MOK na etapie sprawozdania szczegółowego),
    6. Umieszczone w przydzielonej grupie bazie danych dane przykładowe, umożliwiające zaprezentowanie funkcjonowania aplikacji (dotyczy projektu na etapie końcowym i częściowo dla modułu funkcjonalnego MOK na etapie sprawozdania szczegółowego oraz wstępnego - działanie uwierzytelnienia),
    7. Płyta CD lub DVD jednorazowego zapisu z zawartością zgodną z wymaganiami określonymi dla projektu końcowego (dotyczy projektu na etapie końcowym).
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.

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 przeznaczonej do zainicjowania edycji sprawozdania na platformie Confluence.
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 aplikacji

Prezentacja MOK

1. Jednym z elementów sprawdzania na etapie *sprawozdania szczegółowego* jest zaprezentowanie przez grupę działania MOK uruchomionego w przydzielonej grupie domenie serwera aplikacyjnego.
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 aplikacji

1. Jednym z elementów sprawdzania na etapie *sprawdzania końcowego* jest zaprezentowanie przez grupę działania aplikacji internetowej uruchomionej w przydzielonej grupie domenie serwera aplikacyjnego.
2. Prezentacja musi odbywać się interaktywnie, z wykorzystaniem przeglądarki stron WWW.
3. Prezentacja nie może przekroczyć 15 minut, uwidaczniając całą funkcjonalność aplikacji.
4. Działanie aplikacji 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 aplikacji korzystając w dowolny sposób z oferowanej przez nią 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 musi zostać przekazany nauczycielom prowadzącym nie później niż przed rozpoczęciem odpowiedzi z etapu końcowego projektu.

Ocena za systematyczność pracy

1. Pula 10 punktów z ogólnej punktacji jest zarezerwowana na indywidualną ocenę systematyczności pracy każdego z uczestników grupy.
2. Oceny tej dokonuje prowadzący zajęcia w wybranym, identycznym dla wszystkich grup terminie. Wówczas porównywana jest aktywność poszczególnych grup, na podstawie zawartości umieszczonej na udostępnionych grupom kontach serwerów i usług, systemach rejestracji wykonania zadań oraz repozytoriach.
3. Szczegółowe kryteria oceny systematyczności podawane są w momencie ogłaszania ocen.
4. Ocena systematyczności pracy może być przeprowadzona bez uprzedzania grup.

Ocena za aktywności dodatkowe

1. Realizacja funkcjonalności z katalogu funkcjonalności dodatkowej MOK nie jest obligatoryjna. Grupa może zrealizować wybrane funkcjonalności i zaprezentować je w etapie projektu szczegółowego.
2. Poprawna realizacja wybranych funkcjonalności wymienionych w katalogu funkcjonalności dodatkowej MOK powoduje zwiększenie punktacji z projektu w etapie szczegółowym o dodatkowe punkty.
3. Liczba przyznanych punktów jest uzależniona od poprawnie wykonanych aktywności dodatkowych MOK, przy czym łączna liczba przyznanych za nie punktów nie może przekroczyć 20 punktów. W wyniku przyznania dodatkowych punktów ocena z projektu nie może być większa niż 100 punktów.
4. Szef grupy jest zobowiązany do zgłoszenia tuż przed oddaniem etapu szczegółowego wszystkich wykonanych aktywności z katalogu funkcjonalności dodatkowej MOK. Jedynie zgłoszone i zaprezentowane aktywności podlegają ocenie.

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 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:
        - przygotowanie dokumentacji (**dokumentacja**)
        - zarządzanie relacyjną bazą danych (**baza**)
        - zarządzanie domeną serwera aplikacyjnego i wdrażanie aplikacji (**wdrożenie**)
        - przygotowanie struktury projektów *maven* i bibliotek oraz ustalenie zależności pomiędzy nimi (**architektura**)
        - integrowanie kodu z gałęzi do głównej linii rozwojowej oraz umieszczanie przetestowanych 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,
        - wydzielona na potrzeby projektu relacyjna baza danych,

- instancja systemu rejestracji zadań,
- instancja (tzw. *domena*) serwera aplikacji Java EE, w którym będzie osadzana i uruchamiana aplikacja,
- repozytorium przeznaczone do przechowywania kodu źródłowego.

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 poprzez wiadomość e-mail adresowaną do nauczycieli prowadzących zajęcia i DW administratora slawonir.klimczak@p.lodz.pl. Rozliczenie niedostępności usług będzie liczone od momentu wysłania wiadomości e-mail.

Założenia projektowe aplikacji

Wymagane technologie

1. Trójwarstwowa aplikacja biznesowa ma zostać zrealizowana z wykorzystaniem następujących technologii:
   - język programowania Java
   - serwer aplikacji Java/Jakarta Enterprise Edition (udostępniony)
   - system zarządzania bazami danych: PostgreSQL (udostępniony)
2. Wersje używanego oprogramowania muszą być zgodne z wymaganiami.
3. Warstwa logiki biznesowej ma zostać zrealizowana z użyciem komponentów EJB w wersji nie niższej niż 3.0 osadzanych w kontenerze EJB serwera aplikacyjnego podanego w wymaganym oprogramowaniu dla projektów SSBD.
4. Warstwa prezentacji odpowiedzialna za interakcję z użytkownikami aplikacji sieciowej może zostać zrealizowana w dowolnej technologii, która będzie spełniać wymienione dalej wymagania, przy czym jej działanie zostanie zweryfikowane w kontenerze WEB serwera aplikacyjnego określonego w wymaganym oprogramowaniu dla projektów SSBD.
5. Wszystkie rozwiązania każdego z problemów zastosowane w projekcie muszą być kompletne i spójne pod względem technologicznym i implementacyjnym.

Podstawowe pojęcia i założenia architektury

1. Warstwa prezentacji musi zostać zrealizowana zgodnie z wzorcem projektowym *Model-Widok-Kontroler* w wersji 2 (*MVCv2*). Klientem aplikacji ma być dowolna przeglądarka internetowa WWW pracująca w środowisku graficznym (np. Mozilla Firefox, Google Chrome, Internet Explorer, Microsoft Edge, Opera), wystarczy spełnienie tego wymagania przez najnowsze stabilne wersję wymienionych przeglądarek.
2. Projekt wielodostępnej aplikacji biznesowej musi przewidywać podział użytkowników na kategorie (poziomy dostępu) różniące się oferowaną im funkcjonalnością, zwane **aktorami**. Na przykład aplikacja sklepu internetowego może posiadać trzech aktorów: administratora, sprzedawcę, klienta i gościa, przy czym ostatni z wymienionych aktorów nie wymaga uwierzytelnienia użytkownika.
3. Funkcjonalność aplikacji biznesowej należy podzielić na **moduły funkcjonalne**, grupujące funkcje 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, autoryzacja, zablokowanie/odblokowanie, zmiana hasła, edycja danych przypisanych do konta użytkownika, przypisanie i odebranie poziomu dostępu dla konta oraz resetowanie hasła.
        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**. 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 aplikacji ustalana jest przez grupę prowadzącą projekt, z wymogiem zaakceptowania 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 biznesowych w technologii Java/Jakarta EE.
3. Zakres funkcjonalności aplikacji 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. Każdy projekt musi posiadać przynajmniej jedną funkcjonalność związaną z dynamiczną agregacją informacji o danych biznesowych przechowywaną w bazie, wykraczającą poza podstawy funkcjonowania aplikacji z danej dziedziny.

Wymagania funkcjonalne

Poniższe wymagania dotyczą wszystkich aplikacji niezależnie od ich tematyki.

Wymagania główne

1. Aplikacja musi umożliwiać zarządzanie kontami/grupami użytkowników z poziomu interfejsu aplikacji oraz zapewniać możliwość uwierzytelnienia użytkowników (z zastosowaniem unikalnego loginu i silnego hasła) dysponującym indywidualnym kontem (nie dotyczy poziomu dostępu gość).
2. Minimalny zestaw przypadków użycia związanych z zarządzaniem kontami musi obejmować:
    1. samodzielną rejestrację (utworzenie konta) przez użytkownika anonimowego, przy czym utworzenie aktywnego konta musi uwzględniać etap autoryzacji podanych danych poprzez indywidualny adres URL udostępniony jako hiperłącze przesłane na adres email przypisany do konta (każdy zastosowany adres URL umożliwia tylko jednokrotne wykonanie akcji);
    2. przydzielenie i odebranie przez administracyjny poziom dostępu (tzw. administratora) wybranego poziomu dostępu dla konta w dowolnym momencie po jego utworzeniu;
    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 umożliwia tylko jednokrotne wykonanie akcji). W ramach modyfikacji danych konta przez administratora i właściciela należy zapewnić możliwość zmiany hasła.
    4. blokowanie i odblokowywanie możliwości uwierzytelnienia się na dane konto bez ingerencji w dane konta;
    5. 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 15 minut);
3. Aplikacja musi udostępniać możliwość przydzielenia konta użytkownika jednocześnie do przynajmniej dwóch różnych poziomów dostępu (aktorów) bez uwzględniania poziomu dostępu gość. 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. Każda akcja inicjująca operację krytyczną zmieniająca dane biznesowe przechowywane w relacyjnej bazie danych wymaga wykonania potwierdzenia przez użytkownika, jeszcze przed wykonaniem akcji. Operacja krytyczna to operacja, dla której nie można z poziomu interfejsu użytkownika wykonać operacji odwrotnej (odwracającej 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 tu implementację wszelkich komponentów/metod komponentów warstwy widoku oraz komponentów/metod komponentów warstwy logiki biznesowej, które są niezbędne dla zrealizowania danego przypadku użycia.
    2. Jeżeli komponent lub metoda są wykorzystywane przez wiele przypadków użycia (może to dotyczyć np. fasad, encji), 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 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;

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 zbioru stron zlokalizowanych na platformie Confluence.
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.

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 płytę CD lub DVD jednokrotnego zapisu z zawartością zgodną z poniższą tabelą:

Katalog	**Opis zawartości**
access	Plik z danymi dostępowymi dla istniejących kont w utworzonym systemie informatycznym, osobno dla każdego poziomu dostępu. Wymagane jawne podanie wszystkich danych dostępowych niezbędnych w procesie uwierzytelniania dla utworzonej aplikacji (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 dane niezbędne dla uruchomienia aplikacji w formacie poleceń DML (plik o nazwie *baza_dane_init.sql*).
doc	Plik PDF z finalną wersją dokumentacji systemu (zawierający dokumentację etapu wstępnego, szczegółowego i końcowego).
conf	Wszystkie pliki konfiguracyjne i moduły, sterowniki niezbędne do uruchomienia aplikacji w domenie serwera aplikacyjnego.
install	Instalacyjny pakiet aplikacji WAR (ang. *Web Archive*).
src	Pełna i ostateczna wersja kodu źródłowego aplikacji 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.
javadoc	Ostateczna wersja dokumentacji programistycznej JavaDoc wygenerowana z kodu umieszczonego w katalogu src.

1. Płyta CD/DVD musi być dołączona w papierowej kopercie.
2. Ponadto koperta i płyta musi być podpisana według następującego schematu: tytuł projektu, numer grupy w formacie ssbdXX (XX-numer grupy) wraz z dopiskiem "Sieciowe Systemy Baz Danych edycja <rok>", gdzie w miejscu pola <rok> należy wpisać rok edycji przedmiotu, z dopiskiem rodzaju studiów (studia inżynierskie stacjonarne/niestacjonarne).
3. Powyższe wymagania są wymaganiami głównymi.
4. Nauczyciele prowadzący zajęcia nie są zobowiązani do udzielania pomocy technicznej w realizacji tych wymagań.

Wymagania kodu źródłowego i wersjonowania

Wymagania główne

1. Kod źródłowy aplikacji musi być przechowywany na przydzielonym grupie repozytorium i tam rozwijany.
2. 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.
3. Autor kodu źródłowego jest zobowiązany do stworzenia/aktualizacji odpowiadającej mu dokumentacji programistycznej JavaDoc.
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 komplenty opis wykonanych zmian.
9. Łączenie kodu w gałęzi głównej powinno odbywać się z wykorzystaniem mechanizmu *pull request* dostępnego w platformie Bitbucket/Stash. 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*; dla każdej takiej osoby musi wcześniej zostać utworzone zadanie, w którym będzie ona mogła zamieścić link do żądania *pull request* jako poświadczenie wykonanej pracy. 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 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 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 na moduły

Wymagania główne

1. Budowana aplikacja biznesowa musi być przygotowana do wdrożenia jako archiwum WAR (ang. *Web Archive*).
2. Poszczególne klasy wchodzące w skład aplikacji muszą być zorganizowane w odrębne pakiety klas Javy:
    - pakiet ziaren CDI i innych klas z warstwy widoku (sugerowana nazwa pakietu: *web*)
    - pakiet klas encji JPA (sugerowana nazwa pakietu: *entities*)
    - pakiet klas wyjątków (sugerowana nazwa pakietu: *exceptions*)
    - odrębny pakiet klas komponentów EJB dla każdego modułu funkcjonalnego (sugerowana nazwa pakietu: *akronim modułu funkcjonalnego zapisany małymi literami*)
    - w razie potrzeby – pakiet pozostałych niezbędnych klas (sugerowana nazwa pakietu: *utils*)
3. W ramach każdego z wymaganych pakietów można stosować dalszy podział na podpakiety.
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 EJB w poszczególnych modułach.
5. Każda klasa komponentu EJB, które udostępniają możliwość wywoływania metod spoza kontenera EJB, musi udostępniać interfejs (ze względu na jednolity sposób dystrybuowania aplikacji wystarczy, aby był to interfejs lokalny).
6. Zabronione jest odwoływanie się do komponentów EJB spoza kontenera EJB bez pośrednictwa interfejsów.

Dokumentacja programistyczna

Wymagania główne

1. Wymagane jest zamieszczanie w kodzie czytelnego opisu kontraktów pakietów, klas, interfejsów i metod języka Java w standardzie JavaDoc.
2. Oprócz wprowadzonego opisu zaimplementowanej metody należy uwzględnić co najmniej następujące elementy (o ile występują):
    - parametry,
    - wartość zwracaną,
    - zadeklarowane wyjątki.
3. Obowiązek dokumentowania kodu nie dotyczy następujących elementów:
    - klas, interfejsów i metod, których kod został wygenerowany za pomocą narzędzi automatyzujących,
    - metod zwracających i ustawiających wartości pól klasy (get(), set()),
    - metod klasy, które zostały zadeklarowane i opisane stosownym komentarzem JavaDoc w interfejsie implementowanym przez daną klasę (o ile pierwotny opis JavaDoc jest aktualny).
4. Opis JavaDoc musi być wprowadzony w języku polskim.

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 (tzw. 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 typu BUG 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 wersje.
8. Jeżeli zadanie dotyczy projektowania lub implementacji konkretnego przypadku użycia wymagane jest przypisanie odpowiadającego mu komponentu (element systemu rejestracji zadań), którego nazwa zawiera akronim modułu funkcjonalnego i numeru przypadku użycia względem podanego modułu (separatorem jest znak kropki).
9. Każde zadanie musi posiadać przypisany typ zadania (zwykle TASK), epic oraz wersję, poziom istotności, wykonawcę, termin wykonania oraz precyzyjny opis zakresu prac. Zadania dotyczące implementacji muszą też posiadać przypisany komponent odpowiadający przypadkowi użycia. W przypadku złożonych zadań można wydzielać podzadania (sub-task), które zawsze muszą mieć określonego wykonawcę i termin wykonania.
10. Zadanie implementacyjne musi posiadać przypisaną rewizję kodu z repozytorium. Jeżeli rewizji jest kilka należy przypisać do zadania wiele rewizji, dokumentujących wykonane prace implementacyjne.
11. Każde wykonanie opisanych w zadaniu prac wymaga jego zakończenia poprzez funkcjonalność "Resolve issue", zawsze przy zakończeniu zadania wymagane podanie powodu zakończenia (np. dla zrealizowanego - Fixed). Kończąc zadanie należy podać rzeczywisty czas jego wykonania wyrażony w godzinach (wartość w polu Time Spent).
12. Potwierdzaniem wykonania zadań zajmuje się szef grupy. Potwierdzenie wykonania zadania jest realizowane poprzez jego zamknięcie. Szef grupy jest zobowiązany do weryfikacji wykonania wszystkich najpóźniej 6 godzin przed rozpoczęciem zajęć. W przypadku nie wykonania zadania szef grupy może zmienić jego termin lub przypisać zadanie innemu uczestnikowi projektu.
13. Przed każdymi zajęciami musi powstać nowa wersja projektu Maven, zawierająca scalony kod z wszystkich wykonanych zadań implementacyjnych. Wersja ta powinna być uruchomiona w przypisanej grupie domenie serwera aplikacyjnego (wymóg obowiązuje od etapu szczegółowego projektu).

14. 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)
15. Czas wykonania zadania jest liczony od momentu zdefiniowania terminu, a nie od przyjęcia zadania przez uczestnika projektu (poprzez status zadania: *in progress*).
16. Jeżeli zadanie dotyczy przygotowania lub zmian w dokumentacji projektu, wówczas konieczne jest zamieszczenie w tytule zadania numerów rozdziałów, których dotyczy opis zadania oraz przypisanie komponentu odpowiedniego dla danego rozdziału dokumentacji.
17. Nie wykonane zadanie, któremu upłynął termin szef grupy może przypisać innej osobie, otwierając ponownie (reopened), wówczas jawnie określany jest nowy termin wykonania dla zmienionego zadania. Podobnie, szef grupy ma możliwość ponownego otwarcia zamkniętego zadania jeżeli uznaje, że opisane w zadaniu prace nie zostały wykonane w całości lub nie zostały wykonane w całości.
18. Szef grupy jest zobowiązany do wprowadzenia do systemu rejestracji zadań następujących zadań typu epic: Dokumentacja, Projektowanie, Implementacja, Konfiguracja, Wersjonowanie, FunkcjeOdpowiedzialność. Dodatkowo szef grupy jest zobowiązany do wprowadzania komponentów odpowiadających impelmentowanym przypadkom użycia (format: nazwa_modułu i numer przypadku użycia, np. MOK.3) i rozdziałom dokumentacji (format. Rozdzial.X, gdzie X to numer rozdziału) oraz komponentu "Baza". Zadania implementacyjne związane z przypadkiem użycia muszą mieć przypisany odpowiedni komponent, a zadania związane z tworzeniem dokumentacji przypisany komponent rozdziału.
19. Na stronie domowej projektu szef grupy jest zobowiązany do umieszczenia wszystkich aktualnych grupowych haseł dostępowych, niezbędnych do realizacji projektu.

Pozostałe wymagania

1. Wszystkie prace do wykonania w ramach przypisanych funkcji muszą być rejestrowane przez szefa grupy jako zadania.
2. Uczestnik projektu w związku z przyznaną funkcją musi zawsze potwierdzić wykonanie zleconych prac poprzez zakończenie zadania (nie zamknięcie).
3. Po oddaniu projektu wstępnego 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 usług udostępnionych grupie.
4. Każda kolejna wersja stabilna projektu musi zapewniać możliwość skompilowania i zbudowania archiwum, a po osadzeniu na serwerze aplikacyjnym aplikacja musi udostępniać dotychczas zaimplementowaną funkcjonalność.

Wymagania technologiczne stawiane aplikacji

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 muszą być utworzone w jednym schemacie (przestrzeni nazw).
3. Każda tabela 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 z założeniami aplikacji.
5. W szczególności odpowiednie ograniczenia muszą być nałożone na pola będące kluczami obcymi spinającymi relacje. Ograniczenia te powinny odwzorowywać interpretację związku w modelu biznesowym realizowanej aplikacji, 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 dla grupy, o nazwie odpowiadającej nazwie grupy (ssbdXX).
9. Zestaw nadanych użytkownikom bazodanowym uprawnień musi być minimalny, niezbędny dla funkcjonowania aplikacji.
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 serwera baz danych, które nie są objęte podstawowym standardem SQL (takich jak procedury składowane, wyzwalacze/triggery, tabele dziedziczone itp.), jak również z mechanizmów kaskad realizowanych na poziomie serwera bazodanowego.
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.

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 konta budowanej aplikacji.

3. Jeżeli zgodnie z przyjetym 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. 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 Java Persistence API (JPA) 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 PQL (ang. Persistent Query Language) operującego na obiektach encji.
4. Dane powiązane z funkcjonalnością aplikacji muszą być przekazywane pomiędzy komponentami EJB w postaci encji.
5. Modyfikacja kluczy obcych może być zrealizowana wyłącznie poprzez wiązanie obiektów encji związkami.
6. Modyfikacja wartości kluczy głównych encji jest niedopuszczalna.

Reprezentacja danych w aplikacji - pozostałe wymagania

1. Opis klas encji za pomocą adnotacji musi dokładnie odwzorowywać struktury bazy danych uwzględniając ograniczenia nakładane na kolumny.
2. W szczególności wszystkie związki zdefiniowane pomiędzy obiektami encji 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 EJB (ang. *Enterprise Java Bean*) w wersji przynajmniej 3.0 działające w ramach kontenera EJB serwera aplikacyjnego.
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. Wykonywanie transakcji związanych z operacjami na obiektach encji musi być realizowane poprzez kontener EJB w strategii CMT (ang. *Container Managed Transactions*).
4. 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.
5. Niedopuszczalne jest korzystanie z transakcji zarządzanych przez komponent EJB w strategii BMT (ang. *Bean Managed Transactions*).
6. Dla każdej transakcji bazodanowej musi być prawidłowo (adekwatnie do operacji wykonywanych w ramach transakcji) określony najniższy możliwy poziom izolacji transakcji.
7. System blokad optymistycznych nie może bazować na wartości pola wersji obiektu encji przekazanego z warstwy widoku jako parametr metody biznesowej.
8. 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).
9. W sytuacji odwołania transakcji aplikacyjnej, która nie jest spowodowana zgłoszeniem wyjątku, należy ponowić jej wykonanie w warstwie logiki biznesowej. Limit prób ponowienia wykonania transakcji musi określać parametr inicjalizacyjny zdefiniowany w deskryptorze wdrożenia web.xml. 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.
10. Aplikacja musi generować dzienniki zdarzeń (tzw. logi) uwzględniające każde wywołanie (wraz z wartościami parametrów) oraz każde zakończenie (wraz z wartością zwracaną lub zgłoszonym wyjątkiem) metody komponentów EJB, 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. Przeglądarka WWW będąca oprogramowaniem klienta aplikacji może korzystać wyłącznie z dynamicznie generowanego przez aplikację kodu [X]HTML i ew. JavaScript; niedopuszczalne jest stosowanie innych technik prezentacji, takich jak aplety Javy czy pliki Flash.
2. 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).
3. W warstwie widoku wszelkie dane konfiguracyjne specyficzne dla tworzonej aplikacji sieciowej należy umieszczać w deskryptorze *web.xml* oraz plikach konfiguracyjnych wybranego zbioru narzędzi (ang.framework).
4. Konfiguracja warstwy prezentacji musi być niezależna od kontekstu zawartego w adresie URL.
5. Wybrany zbiór narzędzi musi wspierać ujednolicony wygląd stron aplikacji (wspólny nagłówek, stopka, menu itp), czyli tzw. szablon stron chroniący przed wielokrotnym powtarzaniem kodu. Zabrania się przy tym wykorzystania ramek HTML.
6. Niedopuszczalne jest osadzanie kodu Java zawierającego logikę przetwarzania żądania w kodzie HTML prezentującym widok aplikacji, jak również generowanie kodu HTML bezpośrednio w kodzie Java.
7. Do przekazywania danych pomiędzy warstwą prezentacji a warstwą logiki biznesowej, jak również do przechowywania danych w warstwie prezentacji można posłużyć się obiektami klas encyjnych, jednakże zabronione jest udostępnianie poza warstwą logiki biznesowej encji znajdującej się w stanie zarządzanym. W takim przypadku przesyłając dane od warstwy logiki do warstwy prezentacji należy posłużyć się kopią obiektu encji uzyskaną przez serializację lub głębokie klonowanie.
8. 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.
9. 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).
10. 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).
11. Poprawne wykonanie każdej akcji prowadzącej do zmodyfikowania zawartości bazy danych musi być potwierdzone komunikatem prezentowanym w interfejsie użytkownika.

12. Informacja o tożsamości użytkownika oraz aktualnym i możliwych do uzyskania poziomie/poziomach dostępu musi być prezentowana na stronie w graficznym interfejsie użytkownika (sugerowana lokalizacja w widocznym nagłówku strony).
13. 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.
14. Zmiana poziomu dostępu nie może wymagać ponownego uwierzytelnienia.
15. Strony udostępniające graficzny interfejs użytkownika muszą być ujednolicone w zakresie: zastosowanej kolorystyki, rozmiaru elementów, podziału oraz lokalizacji sekcji (m.in. menu, nagłówka, główna).

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 założeniami biznesowymi aplikacji.
5. Jeżeli formularz aplikacji posiada pola, których wypełnienie jest obowiązkowe, powinny one być oznaczone w interfejsie użytkownika znakiem gwiazdki (*).
6. Każda strona aplikacji powinna zawierać informację o realizowanej funkcjonalności (np. tytuł umieszczony na stronie) wraz z lokalizacją w interfejsie użytkownika (format ścieżki) rozpoczynając od strony głównej 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. Należy uwzględnić możliwość jednoczesnego modyfikowania tych samych danych w dwóch różnych sesjach użytkowników w wielodostępnym systemie informatycznym. Szczególną uwagę należy zwrócić na przypadek edycja/edycja, edycja/usunięcie i usunięcie/edycja realizowane na wspólnych danych.
4. 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.
5. 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.
6. 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 nieszyfrowanego aplikacja musi odmówić wykonania akcji lub automatycznie przenieść komunikację na protokół zapewniający poufność transmisji.
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 danych zawartych w bazie będącej częścią projektu oraz mechanizmów udostępnianych przez serwer aplikacji Java EE.
7. Aplikacja musi udostępniać uwierzytelnionemu użytkownikowi możliwość zamknięcia sesji ("wylogowania"), przy czym wykonanie tej akcji musi doprowadzić do unieważnienia sesji kontenera Web (warstwy prezentacji).
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, z którego korzysta użytkownik.
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).

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.