Zadanie 1 – Założenie projektu
Stworzyć nowy projekt typy Dynamic Web Project. Dodać dostarczone biblioteki oraz sprawdzić pliki deskryptorów.

Zadanie 2 – Facelets
a) Stworzyć szablon widoku (*layout/template.xhtml*) zawierający:
- menu po lewej stronie
  - link do strony głównej
  - link do strony służącej do zarządzania użytkownikami
- nagłówek
- tytuł strony w górnej części panelu głównego

b) Stworzyć stronę powitalną (*welcome.xhtml*) dekorowaną powyższym szablonem.

c) Stworzyć namiastkę strony logowania (*login.xhtml*), która nie jest dekorowana szablonem. Kontrolki na tej stronie będą dodane w kolejnym zadaniu.

Zadanie 3 – Komponenty graficzne
Rozbudować stronę logowania tak, aby składała się pól poprzedzonych etykietami:
- login
- hasło (ukryte)

Kliknięcie Zaloguj w przyszłości będzie przenosić na stronę powitalną.

Zadanie 4 – Facelets - Płytki
Stworzyć widok korzystający z szablonu prezentujący listę użytkowników (*users.xhtml*). Kolumny:
- zdjęcie + imię + nazwisko
- login

Z założenia zdjęcie i imię i nazwisko będą często używanym powtarzalnym elementem widoku. Należy ten „komponent widoku” przenieść do osobnej „płytki”: *layout/tiles/userFace.xhtml*

Zadanie 5 – Komponenty ManagedBeans i wstrzykiwanie zależności
a) Stworzyć bean *LoginPresenter* odpowiadający za autentykację i autoryzację użytkowników. Pola formularza logowania powinny być połączone z polami beana *LoginPresenter*, natomiast przycisk Zaloguj powinien wywoływać metodę *authenticate* beana *LoginPresenter*.

b) Stworzyć bean *CurrentUser*, który będzie reprezentował aktualnie zarejestrowanego w sesji użytkownika. Bean będzie służył m.in. do przechowywania encji *User* z modelu odpowiadającej aktualnemu użytkownikowi oraz do sprawdzania czy dany użytkownik jest administatorem.

LoginPresenter symuluje dostęp do servisu aplikacyjnego/bazy w celu sprawdzenia czy encja *User* o podanym loginie i haśle istnieje. Po pomyślnej autentykacji wywołuje metodę autoryzującą beana *CurrentUser* przekazując mu encję User z modelu.

Zadanie 6 – Komponenty ManagedBean - zasięgi

a) Komponent *CurrentUser* powinien znajdować się w zasięgu sesji.

b) Wyświetlić login zalogowanego użytkownika w nagłówku.

c) Link do strony służącej do zarządzania użytkownikami powinien być widoczny tylko wówczas, gdy zalogowany użytkownik ma rolę administratora (atrybut rendered).

Zadanie 7.1 – Komponenty Managed Beans- cykl życia

a) Stworzyć beana (*UsersPresenter*) dla widoku listy użytkowników (*users.xhtml*). Komponent powinien mieć metodę inicjującą, która przygotowuje model (listę użytkowników) do prezentacji przy pierwszym wejściu na stronę.

Uwaga: dokonać obserwacji co dzieję się gdy podczas post-back (np. usunięcie użytkownika) lista użytkowników zmieni się na poziomie repozytorium danych.

b) Komponent powinien mieć zasięg View aby podtrzymywać model listy w razie operacji na niej (sortowanie, stronicowanie, usuwanie).

Uwaga: być może w zasięgu View powinna znajdować się jedynie kolekcja danych?

c) Dodać wyszukiwanie użytkowników po imieniu i nazwisku. Dane po których filtrujemy (*UserSearchCriteria*) powinny być „podpowiadane” w wartości ostatnio wpisanej **za każdym razem** gdy wchodzimy na stronę *users.xhtml*.

Zadanie 7.2 – Komponenty Managed Beans- cykl życia
a) Na liście produktów (*products.xhtml*) dodać funkcjonalność pozwalającą na dodanie produktu do koszyka (*CartPresenter*).

b) Podsumowanie koszyka (cena i ilość produktów) powinna być zawsze widoczna w szablonie widoku *layout/template.xhtml.*

c) Stan koszyka powinien być dostępny przy każdym wejściu na dowolną stronę.
uwaga: zastanowić się czy CartPresenter powinien być komponentem sesyjnym, czy być może jego stan powinien być odtwarzany przy każdym żądaniu do serwera?

Zadanie 8 – Nawigacja
a) Dodać reguły nawigacji do strony logowania (warto pamiętać o stylu oznajmującym). Po poprawnym logowaniu użytkownik powinien zobaczyć stronę *welcome.xhtml.*

b) Dodać funkcjonalność sprawdzanie ile razy dany użytkownik popełnił błąd podczas logowania - jeżeli więcej niż trzykrotnie wówczas po błędzie przekierować widok na stronę odzyskania hasła.

Zadanie 8a – Nawigacja i parametry
Dodać funkcjonalność usuwania użytkownika z systemu – kliknięcie usuń na liście *(users.xhtml).* Po usunięciu lista powinna się odświeżyć dzięki wykorzystaniu uproszczonej nawigacji i zwróceniu null. Zaobserwować co stanie się, gdy wykorzystamy reguły nawigacji lub zwrócimy nazwę widoku.

Zadanie 9 – Nawigacja zorientowana na GET
a) Stworzyć widok podglądu szczegółów użytkownika (*userDetails.xhtml*).

b) Dostęp do niego powinien odbywać się po kliknięciu na button Szczegóły na liście użytkowników.

c) Dostęp do konkretnego użytkownika powinien być możliwy również poprzez URL . Zastosować komponent f:viewParam i f:viewAction.

Zadanie 10 – Konwerter

a) Stworzyć widok dodawania nowego użytkownika (*userEditor.xhtml*). Widok powinien pozwalać na podanie:
- imienia
- nazwiska
- loginu
- hasła
- adresu e-mail
- ról

Role należy wybrać przy pomocy komponentu h:selectManyCheckbox – wykorzystać konwerter (*RoleConverter*) obiektów modelu: *Role*.
Stworzyć ManagedBean o zasięgu aplikacji, który przechowuje słowniki (w szczególności słownik dostępnych ról).

b) Wyświetlając koszt koszyka w szablonie (*template.xhtml)* należy wykorzystać konwerter *MoneyConverter.*

Zadanie 11 – Walidator
Na widoku dodawania nowego użytkownika pole e-mail powinno zawierać adres należący do domeny firmy (*EmailValidator*).

Zadanie 12 – Bean validation
Do encji Użytkownika dodać adnotacje walidujące. Dla każdego pola edycyjnego należy również dodać wyświetlanie błędów.

Zadanie 13 – Grupy walidatorów
Rozdzielić walidację podczas dodawania użytkownika na dwa przypadki:
- nagły wypadek (wymagany jedynie login)
- standardowa procedura

Jeżeli aktualnie zalogowany użytkownik jest administratorem, to pracuje z walidacją w trybie nagłego wypadku.
Uwaga: Wstrzyknąć komponent reprezentujący zalogowanego użytkownika do komponentu edytora.

Zadanie 14 - Internacjonalizacja
Wprowadzić internacjonalizację na ekranie logowania

Zadanie 15 – Strony obsługujące błędy
a) Stworzyć stronę obsługującą błędny identyfikator użytkownika przekazany w parametrze URL do ekranu podglądu szczegółów użytkownika.

b) Stworzyć stronę obsługującą sytuację gdy użytkownik próbuje przeglądać szczegóły użytkownika, do którego nie ma praw dostępu.

Zadanie 16 – Ajax
Na formularzu dodawania nowego użytkownika dodać informację o „sile” wprowadzonego hasła. Każdorazowa zmiana wartości hasła powinna uaktualniać wskaźnik siły.

Zadanie 17 – Ajax i walidacja
a) Cały formularz dodawania nowego użytkownika powinien być walidowany bez przeładowania strony.
b) Jeżeli login istnieje w systemie (np. symulacja w logice przez warunek, że zaczyna się od pewnej listery), to wówczas przycisk Zapisz powinien być nieaktywny.

Zadanie 18 – Zasoby
Przycisk Zapisz na ekranie dodawania nowego użytkownika zastąpić ikoną z napisem (*resources**pl**icons**save.png*).

Zadanie 19 – Komponenty własne
Stworzyć własny komponent typu Input walidujący wprowadzane dane poprzez Ajax (*resources**components**extendedInput.xhtml*).

Zadanie 20 – Wywołanie metody z parametrem
Dodać sortowanie do listy użytkowników.

Zadanie 21 – Flow scope
Stworzyć 2-etapowy wizard dodawania nowego produktu. W pierwszym kroku podajemy nazwę produktu, w drugim pobieramy cenę. Produkt zostaje stworzony dopiero po zatwierdzeniu drugiego kroku, do tego czasu można przemieszczać się pomiędzy krokami bez utraty wprowadzonych danych. Każdy krok powinien być obługiwany przez osobnego MB (FlowScoped). Ostatecznym zatwierdzeniem zajmuje się trzeci MB o zasięgu Request.