Projektowanie oprogramowania – Podgrupa1

I. Opis biznesowy „świata rzeczywistego” w języku klienta – aplikacja „Zapisy na zajęcia”

1. Opis zasobów ludzkich

1.1. Pracownik Uczelni, zarządzający zasobami systemu zapisów: wprowadzane dane dotyczące zasobów Uczelni:

Wydział	Stopień studów	Typ studiów:	Kierunek	Specjalność	Prowadzacy zajęcia	Kurs-główna forma	Kurs-forma cząstkowa	Grupa-wykład	Grupa-laboratorium	Grupa-ćwiczenie	Grupa-seminarium	Grupa-projekt	Student	Wykaz zajęć studenta	Dane sali	Termin zajęć	Termin zapisów
Lista wydziałów	I i II	stacjonarne, niestacjonarne	Kierunki na wydziałach	Specjalności na kierunkach	Lista nauzycieli akademickich	Główna forma – do protokołu	Kurs cząstkowy	Dane grupy	Dane grupy	Dane grupy	Dane grupy	Dane grupy	Dane studenta		Budynek, nr sali	Terminy zajęć	

1.2. Prowadzący zajęcia, studenci

Prowadzący może prowadzić zajęcia tylko w jednej grupie w danym terminie. Student nie powinien zapisać się do dwóch różnych grup, których czas zajęć się pokrywa. Liczność grup powinna być ograniczona. System powinien wysyłać powiadomienia do studenta o terminach zapisów. Po przekroczeniu terminu student nie może zapisać się na zajęcia. Odblokowanie wymaga zgody ze strony Uczelni. Grupy nieobsadzone są likwidowane. Podobnie, jeśli w grupie jest zbyt mało studentów, grupa może być zlikwidowana wtedy, gdy studenci mogą być przydzieleni do innych grup - pod warunkiem, że nie zostanie przekroczona górna granica liczności grupy. Podczas zapisów należy sprawdzić, czy student może zapisać się na zajęcia np kontrolując jego przynależność do grupy studentów z takimi uprawnieniami.

2. Przepisy

Podczas wyboru grupy student musi kierować się regulaminem studiów.

3. Dane techniczne

Należy zrealizować system z wykorzystaniem technologii Java EE, gdyż Uczelnia posiada dział wspierający utrzymanie oprogramowania w tej technologii. Dane dotyczące wydziałów, przedmiotów oraz kursów, są stabilne tzn nie wymagają częstych zmian. Zapisy odbywają się głównie w okresie poprzedzającym kolejny semestr. Uczelnia ma kilka pomieszczeń, w których mogą odbywać się zapisy.

II. **Lista wymagań funkcjonalnych (wraz z minimalnym zestawem atrybutów)**

1. Dodawanie konta studenta jako zasobu Uczelni (należy podać atrybuty, należy podać sposób identyfikacji)
2. Dodawanie wydziału jako zasobu Uczelni (należy podać atrybuty, należy podać sposób identyfikacji)
3. Dodawanie stopnia studiów jako zasobu Uczelni (należy podać atrybuty, należy podać sposób identyfikacji)
4. Dodawanie kierunku studiów (należy podać atrybuty, należy podać sposób identyfikacji)
5. Dodawanie specjalności jako zasobu Uczelni (należy podać atrybuty, należy podać sposób identyfikacji)
6. Dodawanie prowadzacego zajęcia jako zasobu Uczelni (należy podać atrybuty, należy podać sposób identyfikacji)
7. Dodawanie kursu głównego: wykład, seminarium, projekt jako zasobu Uczelni (należy podać atrybuty, należy podać sposób identyfikacji)
8. Dodawanie form towarzyszących kursowi głównemu: laboratorium, ćwiczenia, projekt, seminarium jako zasobu Uczelni (należy podać atrybuty, należy podać sposób identyfikacji)
9. Dodanie grup jako zasobu Uczelni (należy podać atrybuty, należy podać sposób identyfikacji)
10. Dodawanie terminów zajęć jako zasobu Uczelni (należy podać atrybuty, należy podać sposób identyfikacji)
11. Dodawanie terminów zapisów jako zasobu Uczelni (należy podać atrybuty, należy podać sposób identyfikacji)
12. Dodawanie danych pomieszczeń jako zasobu Uczelni (należy podać atrybuty, należy podać sposób identyfikacji)
13. Dodanie przydziału kursów do grup zajęciowych, pomieszczeń i terminów zajęć
14. Dodawanie wykazu zajęć studenta wynikający z typu i rodzaju studiów, roku studów, wydziału, kierunku, specjalności oraz kursów
15. Dodawanie zapisu studenta na obowiązkowe i wybieralne zajęcia oparty na wyborze: konta studenta, jego wykazu zajęć, grup zajęciowych wynikających z wykazu zajęć
16. *Analiza zapisów (dane wejściowe do ustalenia, zastosowanie wybranego algorytmu typu Data mining)

III. Lista wymagań niefunkcjonalnych (do opracowania)

1. Liczba poszczególnych danych – studentów, zapisów,
2. Liczba grup i sal zajęciowych
3. Ograniczenia wydajnościowe
4. Czy jest wymagany masowy dostęp (Internet)?
5. Proponowane technologie

IV. **Lista przypadków użycia - propozycja . Harmonogram prac poszczególnych sprintów zostanie podany w osobnym pliku**

Tabela 1.Sprint 1 – dodawanie zasobów biura Uczelni (4 tygodnie)

Podgrupy 1 osoba jako Scrum Master do pomocy w poszczególnych podgrupach	Przypadki użycia – model, implementacja (logika biznesowa i GUI SE), testy: jednostkowe, akceptacyjne
1-a podgrupa (2 osoby)	1. PU Dodawanie wydziału 2. PU Dodawanie typu i rodzaju studiow, powiązanie z wydziałem 3. PU Dodawanie konta studenta i powiazanie go z wydziałem
2-a podgrupa (3 osoby)	1. PU Dodawanie kierunków na poszczególnych latach studiów 2. PU Dodawanie specjalności i powiązanie z kierunkami studiów 3. PU Dodawanie kursów oraz form cząstkowych 4. PU Dodawanie terminów zajęć
3-podgrupa (2 osoby)	1. PU Dodawanie grup zajęciowych 2. PU Dodawanie sal zajęciowych 3. Dodawanie prowadzących zajęcia

Tabela 2. Harmonogram realizacji 1 sprintu (tabela 1)

Opis realizacji sprintu dla trzech podgrup zespołu					
Nr tygodnia Semestru/ nr tygodnia sprintu	**Sprint**	**Spotkanie**	**Uwagi dotyczące realizacji zadań przez każdą z dwóch podgrup zespołu**	**Liczba punktów (do oceny)**	**Zadania Scrum Master**
2 /1	**1**	Sprint planning meeting (90 min)	Zajęcia organizacyjne (podział na grupy i podgrupy, przydzielenie ról projektowych, uzyskanie dostępu do wymaganych narzędzi) • **User Stories** – Analiza dostarczonego modelu biznesowego „świata rezczywistego”systemu– *udział wszystkich grup projektowych*. Każda grupa projektowa otrzymuje ogólny opis procesów biznesowych, ale może opracować dokładnie wybrany fragment opisu biznesowego • **Sprint Backlog (formy pośrednie: Product Backlog), Sprint planining)** - ewnetualna dostarczonych modyfikacja wymagań funkcjonalnych i niefunkcjonalnych– *udział wszystkich grup projektowych.* Każda grupa dokładnie weryfikuje część wymagań funkcjonalnych wynikających z otrzymanego fragmentu opisu „świata rzeczywistego”	3-5	Współdziałanie z wykonawcami z podgrup

			• **Definicja PU** (przypadku użycia): opis słowny wg standardowego formularza – scenariusz należy wykonać za pomocą diagramu aktywności.. Każda grupa opracowuje przypadki użycia jako specyfikację tych wymagań funkcjonalnych, które opracowała w poprzednich krokach**. Można wykonać kod wg scenariuszy PU, jeśli został zidentyfikowany scenariusz działania, analogiczny jak w przypadku dodawania obiektow typu TTitle (instrukcje do lab1) oraz obiektów TRachunek lub TProdukt1 i TProdukt2 w instrukcji do lab8.** **Wyniki prac są umieszczane:** • **projektu Registration for classes** w repozytorium **Repository_team1,** • **projektu Travel agency w repozytorium Repository_team2,** **i zostaną ocenione oceniane przez prowadzącego zajęcia.**		
3/2	**1**	Daily Scrum of Scrums (20 min)	**Przedstawienie wyników prac grup z 1 tygodnia (wersja początkowa). Projekty zawierają:** 1) diagram przypadków użycia, 2) diagramy aktywności 3) kod na podstawie scenariuszy PU i rozwiązań podanych w instrukcjach do lab1 i lab8. P. Inzynieria Oprogramowania **Podczas 2-tygodnia wyniki prac umieszczane są w repozytoriach** 1) Podgrupa 1 – wykonanie kodu (cd) i GUI dla jednego PU, testy jednostkowe i akceptacyjne implementowanego PU 2) Podgrupa2 (2osoby) – wykonanie kodu (cd) i GUI dla jednego PU, testy jednostkowe i akceptacyjne implementowanego PU 3) Podgrupa3 – wykonanie kodu i GUI dla jednego PU, testy jednostkowe i akceptacyjne implementowanego PU **Przykład połączenia wartswy klienta (GUI – projekt Library1_client1_SE) i logiki biznesowej (projekt Library1)**: **Przykład programu**	3-5	Scrum Master i jeden student z podgrupy 2: • integrują diagramy przypadków użycia, korygują scenariusze przypadków użycia, • tworzą diagram wymagań • dodają diagramy aktywności W wyniku ma powstać jeden projekt UML integrujący diagramy przypadków użycia trzech grup jako jeden diagram, diagramy aktywności wykonane podczas 2 tygodnia,diagram wymagań
4/3	**1**	Daily Scrum of Scrums (20 min)	**Przedstawienie wyników prac grup z 2 tygodnia oraz Scrum master i wybranej osoby z podgrupy 2.** Prezentacja obejmuje: 1) DPU prezentujący zintegrowany DPU oraz wykonane diagramy aktywności i diagram wymagań 2) Kody trzech podgrup zawierające dwie warstwy: klienta (GUI – podobnie jak projekt **Library1_client1_SE**) i logiki biznesowej (podobnie jak projekt **Library1**), prezentacja wyników testów	3-5	Współdziałanie z wykonawcami z podgrup Wykonanie diagramu klas i dodanie do projektu UML na podstawie klas wykorzystanych w kodzie trzech podgrup

			Podczas 3-tygodnia wyniki prac umieszczane są w repozytoriach 1) Podgrupa 1 – wykonanie kodu (cd) i GUI – kontynuacja (2-i PU), testy jednostkowe i akceptacyjne implementowanego PU 2) Podgrupa2 (3 osoby) – wykonanie kodu i GUI – kontynuacja (2-i i 3-i PU), testy jednostkowe i akceptacyjne implementowanych PU 3) Podgrupa3 – wykonanie kodu i GUI—kontynuacja (3-i PU), testy jednostkowe i akceptacyjne implementowanego PU		
5/4	**1**	Daily Scrum of Scrums (20 min)	**Przedstawienie wyników prac grup z 2 tygodnia oraz Scrum master i wybranej osoby z podgrupy 2.** Prezentacja obejmuje: 1) DPU prezentujący diagram klas 2) Kody trzech podgrup zawierające dwie warstwy: klienta (GUI – podobnie jak projekt **Library1_client1_SE**) i logiki biznesowej (podobnie jak projekt **Library1**) **Podczas 4-tygodnia wyniki prac umieszczane są w repozytoriach** 1) Podgrupa 1 – wykonanie kodu (cd) i GUI – kontynuacja (3-i PU), testy jednostkowe i akceptacyjne implementowanego PU 2) Podgrupa2 (3 osoby) – wykonanie kodu i GUI – kontynuacja (4-y PU), testy jednostkowe i akceptacyjne implementowanych PU 3) Podgrupa3 – wykonanie kodu i GUI—kontynuacja (3-i PU), testy jednostkowe i akceptacyjne implementowanego PU	3-5	Współdziałanie z wykonawcami z podgrup

<u>Sprint 2</u> – dodawanie powiązań pomiędzy danymi (4 tygodnie)

Podgrupy 1 osoba jako Scrum Master do pomocy w poszczególnych podgrupach Należy wybrać z grupy nowego Scrum Master. Poniżej podaję skład dwóch podgrup. Podgrupa 1 pierwsza składa się z 5 osób – jeden z uczestników powinien zostac wybrany do pełnienia roli Scrum Master	Przypadki użycia – model, implementacja (logika biznesowa, GUI EE, JPA), testy: jednostkowe, akceptacyjne, funkcjonalne
1-a podgrupa – 4 osoby Kacper Piasecki, Aleksander Drozd, Damian Stolarek, Krzysztof Mikucki, Bartosz Herczyk	1. PU Powiązanie kursów z kierunkami i/lub specjalnościami 2. PU Powiązanie prowadzących zajęcia z grupami, kursami i terminami Przetwarzanie obiektów klas: Faculty, Lecturer.w metodach klasy Facade 1. Dodawanie obiektów typu Faculty- metoda Facade 2. Dodawanie obiektów typu FieldOfStudy (oraz ModeOfStudy, LevelOfStudy) - metody: Facade, Faculty 3. Dodawanie obiektów typu Specialty- metody:Facade, Faculty, FieldOfStudy. 4. Dodawanie CoursesGroup- metody Facade, Faculty, FieldOfStudy 5. Dodawanie Course (razem z CourseType) - metody Facade, Faculty, FieldOfStudy, CourseGroup 6. Dodawanie obiektów typu Classes – metody: Facade, Faculty, FieldOfStudy, CourseGroup, Course, 7. Dodawanie obiektów typu Term (razem z WeekParity)- metody 8. Facade, Faculty, FieldOfStudy, CourseGroup, Course, Classes, 9. Dodawanie obiektów typu Lecturer-metoda Façade 10. Dodawanie powiązania miedzy obiektami typu Lecturer i obiektami typu Classes: metoda Facade:. wyszukanie obiektu typu Lecturer (Facade), oraz wyszukanie obiektu typu Classes (metoda Façade wyszukuje, Faculty. Nastepnie wywoluje metode Faculty, ktora wyszukuje, FieldOfStudy. Metoda obiektu typu FieldOfStudy wyszukuje obiekt typu, CourseGroup i wywoluje jego metode wyszukującą obiekt typu, Course. Od wyszukanego obiektu typu Course wywołuje metodę wyszukującą obiekt typu Classes. Obiekt ten przez return wraca do metody rozpoczynającej wyszukiwanie z klasy Facade). Wywowłanie metody

	wyszukanego obiektu typu Lecturer i przekazanie jej jako parametru, wyszukanego obiektu typy Classes, zwróconego jako wynik kaskadowego wyszukania obiektu typu Classes. **Poniżej tabeli Sprint 2 podałam informacje dotyczącą konieczności zrefaktoryzowania kodu na przykładzie dodawanie obiektów typu Specialty**.
2-a podgrupa – 3 osoby 1-a podgrupa: Marcin Okroy, Patryk Witkowski, Jakub Małyjasiak	1. PU Rozklad zajęć studenta Przetwarzanie obiektów klas: Hall, Student w metodach klasy Facade 1. Dodawanie obiektu typu Hall – metoda Facade 2. Dodawanie obiektu typu Student-metoda Facade 3. Tworzenie rozkladu zajec obiektu typu Student: Metoda Facade wyszukuje obiekt typu Student oraz obiekt typu Course (Facade, Faculty, FieldOfStudy, CourseGroup i Course) i /lub (Facade, Faculty, FieldOfStudy, Sepcialty, CourseGroup, Course). Nalezy wiec polaczyc klase Student z klasa Course relacja 1…*.. Otrzymamy w ten sposób rozklad zajec obiekty typu Student.w postaci kolekcji obiektow typu Course. Bedzie to potrzebne do pozniejszego zapisu na zajecia. Mozna to wykonac w osobnym programie na tych samych klasach wykonac kod tych operacji. Wtedy nalezy zainnicjowac dane w kolekcjach powiazanych klas, i zrealizowac kod. Kod do wyszukiwania obiektów typu: Facukty, FieldOfStudy zrealizowac we współpracy z podgrupą 1. **Poniżej tabeli Sprint 2 podałam informacje dotyczącą konieczności zrefaktoryzowania kodu na przykładzie dodawanie obiektów typu Specialty..**

Wyjaśnienie: Obecny diagram klas określa, jaką refaktoryzację kodu należy dokonać.

Klasa Facade zarządza jedynie obiektami klas: Lecturer, Hall, Student i Faculty. Może wstawiać, usuwać, modyfikować i przeszukiwać kolekcje obiektów klas.

Wstawianie obiektów typu Specialty musi przebiegać podobnie jak obiekt typu TTile_book, który kontynuuje wstawianie obiektów typu TBook (kod z lab.1), rozpoczęty przez obiekt typu TFacade. Oznacza to, że metoda public Boolean addSpeciality(Object[] data) z klasy Façade powinna wyszukać obiekt typu Faculty, korzystając z części elementów tablicy data. Po wyszukaniu właściwego obiektu typu Faculty powinna wywołać jego metodę public Boolean addSpeciality(Object[] data), która jest zdefiniowana w klasie Faculty i ta metoda wyszukuje właściwy obiekt typu FieldOfStudy korzystając z kolejnych elemetów tablicy data. Po znalezieniu wywołuje metodę public Boolean addSpeciality(Object[] data) od znalezionego obiekty typu FieldOfStudy. W tej metodzie zdefiniowanej w klasie FieldOfStudy nastąpi wyszukanie obiektu typu Specialty korzystając z pozostałych elementów tablicy data. Jeśli nie zostanie znaleziony obiekt o takich danych, zostanie wstawiony nowy obiekt typu Specialty do kolekcji obiektu typu FieldOfStudy. Podobnie należy wykonać wstawianie obiektów innych typów niż Lecturer, Hall, Student i Faculty. To ma wynikać z powiązań z diagramu klas.

Tabela3. *Harmonogram realizacji 2 sprintu (tabela 1)*

Opis realizacji sprintu dla trzech podgrup zespołu					
Nr tygodnia Semestru/ nr tygodnia sprintu	**Sprint**	**Spotkanie**	**Uwagi dotyczące realizacji zadań przez każdą z dwóch podgrup zespołu**	**Liczba punktów (do oceny)**	**Zadania Scrum Master**
6,7/1, 2 6.04.17-20.04.17	**2**	Sprint planning meeting (90 min)	Zajęcia organizacyjne (podział na grupy i podgrupy, przydzielenie ról projektowych, uzyskanie dostępu do wymaganych narzędzi) 1. **User Stories** – Analiza wykonanego diagramu klas. Propozycja przydziału logiki biznesowej na poszczególne klasy diagramu- wynikające z powiazań pomiędzy klasami. 2. **Sprint Backlog (formy pośrednie: Product Backlog), Sprint planining)** - modyfikacja scenariuszy przypadków użycia 3. **Projekt i implementacja PU** - : zadania realizowane przez dwie podgrupy wg tabeli Sprint 2 3.1. **Należy wykonać diagramy sekwencji i aktywności przypadków użycia działających na powiązanych danych z diagramu klas, zidentyfikowanych na podstawie scenariuszy przypadków użycia z 1 sprintu.** 3.2. **Wykonać kod wg diagramów sekwencji. Należy zmodyfikować scenariusze przypadków użycia, wynikające ze zidentyfikowanych powiązań pomiędzy danymi.** Przykłady diagramów sekwencji złożonych operacji na powiązanych danych przedstawiono w instrukcji do lab1 **(http://zofia.kruczkiewicz.staff.iiar.pwr.wroc.pl/wyklady/IO_UML/Instrukcja_1_2.pdf), lab8 (http://zofia.kruczkiewicz.staff.iiar.pwr.wroc.pl/wyklady/IO_UML/Instrukcja_6_1.pdf) i lab.9-10 (http://zofia.kruczkiewicz.staff.iiar.pwr.wroc.pl/wyklady/IO_UML/Instrukcja_7_1.pdf)** (semestr 5, Inżynieria Oprogramowania) w przypadku dodawania obiektow typu TTitle (instrukcje do lab1) oraz obiektów TRachunek lub TProdukt1 i TProdukt2 w instrukcji do lab8. Przykłady diagramów aktywności złożonych operacji na powiązanych danych przedstawiono w instrukcji do lab5 **(http://zofia.kruczkiewicz.staff.iiar.pwr.wroc.pl/wyklady/IO_UML/Instrukcja_4_1.pdf)** **Wyniki prac są umieszczane:**	3-5	Współdziałanie z wykonawcami z podgrup

			• **projektu Registration for classes** w repozytorium **Repository_team1,** • **projektu Travel agency w repozytorium Repository_team2,** **i zostaną ocenione oceniane przez prowadzącego zajęcia.**		
7/3 20.04.17-27.04.17	**2**	Daily Scrum of Scrums (20 min)	**Przedstawienie wyników prac grup z 1 i 2 tygodnia sprintu 2. Projekty zawierają:** 1) diagram przypadków użycia, diagram klas, diagramy sekwencji, diagramy aktywności 2) kod na podstawie scenariuszy PU i rozwiązań podanych w instrukcjach do lab1 i lab8, 9. P. Inzynieria Oprogramowania **Podczas 3-tygodnia wyniki prac umieszczane są w repozytoriach** 3) Podgrupa 1 (3 osoby) – wykonanie kodu (cd) i GUI realizowanych PU, testy jednostkowe i akceptacyjne implementowanych PU 4) Podgrupa2 (4 osoby) – wykonanie kodu (cd) i GUI dla realizowanych, testy jednostkowe i akceptacyjne implementowanego PU **Przykład połączenia wartswy klienta (GUI – projekt Library1_client1_SE) i logiki biznesowej (projekt Library1)**: **Przykład programu**	3-5	Scrum Master i jeden student z podgrupy 2: • integrują diagramy przypadków użycia, diagramy sekwncji i aktywności, W wyniku ma powstać jeden projekt UML integrujący diagramy przypadków użycia dwóch grup jako jeden diagram, zawierający diagramy aktywności, sekwencji wykonane podczas 1 i 2 tygodnia sprintu 2 oraz diagram klas zawierający definicje operacji klas wynikających z diagramów sekwencji.
8/4 27.04.17-4.05.17	**3**	Daily Scrum of Scrums (20 min)	**Przedstawienie wyników prac grup z 3 tygodnia sprintu 2 oraz Scrum master i wybranej osoby z podgrupy 2.** Prezentacja obejmuje: 1) DPU prezentujący zintegrowany DPU oraz wykonane diagramy aktywności i diagram wymagań 2) Kody dwóch podgrup zawierające dwie warstwy: klienta (GUI – podobnie jak projekt **Library1_client1_SE**) i logiki biznesowej (podobnie jak projekt **Library1**), prezentacja wyników testów **Podczas 4-tygodnia 2 sprintu wyniki prac umieszczane są w repozytoriach** 3) Podgrupa 1 – wykonanie kodu (cd) i GUI – kontynuacja w wersji Enterprise (Java EE), 4) Podgrupa2 (3 osoby) – wykonanie kodu (cd) i GUI – kontynuacja w wersji Enterprise (Java EE), **Przykład programu w wersji Java EE:** **http://zofia.kruczkiewicz.staff.iiar.pwr.wroc.pl/wyklady/PiWSI/Budowa_aplikacjiEE.pdf**	3-5	Współdziałanie z wykonawcami z podgrup **1 i 2** Zapoznaznie się z technologą Java EE i JPA – przykład zastosowania tych technologii: **http://zofia.kruczkiewicz.staff.iiar.pwr.wroc.pl/wyklady/OBPROG/Instrukcja_ORM_PO_INS.pdf** oraz wykonanie programu dla wybranego kodu wykonanego podczas 1 i 2 tygodnia sprintu 2.

Sprint 3 – dodawanie powiązań pomiędzy danymi (cd – 3 tygodnie)

Podgrupa 1 osoba jako Scrum Master do pomocy w poszczególnych podgrupach	Przypadki użycia – model, implementacja (logika biznesowa, GUI EE JPA), testy: jednostkowe, akceptacyjne, funkcjonalne
1-a podgrupa	1. PU Integracja wyników prac grupy 1 i 2 ze sprintu 2 2. PU Dodawanie zapisów na zajęcia
2-a podgrupa	2. *PU Analiza danych dotyczących zapisów

Sprint 4 – kontynuacja implementacji (3 tygodnie)

Podgrupa 1 osoba jako Scrum Master do pomocy w poszczególnych podgrupach	Przypadki użycia - kontynuacja implementacji na platformie Java EE oraz testów
1-a podgrupa	1. PU Dodawanie zapisów na zajęcia (cd)
2-a podgrupa	2. *PU Analiza danych dotyczących zapisów (cd)