KARTA OPISU PRZEDMIOTU - SYLABUS

Nazwa przedmiotu
Technologie rozwoju oprogramowania [S2Inf1E-IO>TRO]

Przedmiot

Kierunek studiów
Informatyka/Computing

Rok/Semestr
1/1

Studia w zakresie (specjalność)
Inżynieria oprogramowania

Profil studiów
ogólnoakademicki

Poziom studiów
drugiego stopnia

Język oferowanego przedmiotu
angielski

Forma studiów
stacjonarne

Wymagalność
obligatoryjny

Liczba godzin

Wykład
30

Laboratorium
0

Inne (np. online)
0

Ćwiczenia
0

Projekty/seminaria
30

Liczba punktów ECTS

6,00

Koordynatorzy

dr inż. Sylwia Kopczyńska
sylwia.kopczynska@put.poznan.pl

mgr inż. Michał Maćkowiak
michal.mackowiak@put.poznan.pl

Wykładowcy

Wymagania wstępne

Student rozpoczynający ten przedmiot powinien mieć podstawową wiedzę dotyczącą podstawowych algorytmów i ich złożoności obliczeniowej, programowania obiektowego, wzorców projektowych, baz danych, testowania oprogramowania i aplikacji internetowych. Student powinien potrafić rozwiązywać podstawowe problemy dotyczące analizy wymagań, tworzenia specyfikacji wymagań, projektowania systemu i umiejętności niezbędnych do zdobywania nowej wiedzy z podanych źródeł informacji. Student powinien rozumieć potrzebę rozszerzenia jej/jego wiedzy i kompetencji oraz mieć chęć pracy w zespole.

Cel przedmiotu

Celem przedmiotu jest: 1) Zapoznanie studentów z wiedzą dotyczącą .NET Framework i innych technologii, sposobu tworzenia aplikacji z wykorzystaniem Ruby on Rails Framework, tworzenia skryptów, programowania funkcyjnego, tworzenia systemów w chmurze. 2) Rozwinięcie u studentów umiejętności rozwiązywania problemów związanych z tworzeniem systemów z wykorzystaniem różnych technologii. 3) Zapoznanie studentów ze zbiorem technologii rozwoju oprogramowania potrzebnych do modelowania warstwy danych, tworzenia interfejsu użytkownika, definiowania warstwy komunikacji. 4) Rozwinięcie u studentów umiejętności pracy w zespole w kontekście tworzenia systemów oraz umiejętności nauki nowych technologii.

Przedmiotowe efekty uczenia się

Wiedza:
1) ma zaawansowaną i szczegółową wiedzę dotyczącą wybranych obszarów informatyki, tworzenia aplikacji internetowych, tworzenia interfejsów użytkownika, skryptów
2) ma wiedzę o nowych technologiach służących do rozwoju oprogramowania
3) ma zaawansowaną i szczegółową wiedzę dotyczącą cyklu życia oprogramowania, która uwzględnia rozwój systemów i ich testowanie

Umiejętności:
1) potrafi pozyskać, połączyć, zinterpretować i ocenić informacje z literatury, baz danych i innych źródeł informacji w języku ojczystym oraz w języku angielskim; wyciągać wnioski i formułować na ich podstawie konkluzje
2) potrafi połączyć wiedzę z różnych obszarów informatyki, aby sformułować i rozwiązać zadania inżynierskie związane z rozwojem oprogramowania
3) potrafi pracować w zespole, pełnić rolę programisty
4) potrafi zaprojektować i stworzyć aplikację (system) z wykorzystaniem technologii omawianych na zajęciach

Kompetencje społeczne:
1) rozumie, że wiedza i umiejętności związane z informatyką szybko się zmieniają,
2) wie jak nowe technologie rozwoju oprogramowania i narzędzia zastosować do rozwiązania praktycznych problemów takich jak stworzenie nowej aplikacji internetowej.

Metody weryfikacji efektów uczenia się i kryteria oceny

Efekty uczenia się przedstawione wyżej weryfikowane są w następujący sposób:

1) Ocena formująca:
a) wykłady: na podstawie odpowiedzi na pytania podczas testu obejmującego treści prezetnowane na wykładach lub zadane do zapoznania się samodzielnie;
b) projekt: na podstawie oceny z wykonanych zadań podczas zajęć oraz zadań domowych.
2) Ocena podsumowująca:
a) weryfikacja w ramach wykładu następuje przez test pisemny, w ramach którego student może zdobyć maksymalnie 100 punktów. Do zdania niezbędne jest minimum 50 pkt. Ocena końcowa wyliczana jest z wykorzystaniem następującej skali:
(90%, 100%] -> 5.0, (80%, 90%] -> 4.5, (70%, 80%] -> 4.0, (60%, 70%] -> 3.5, (50%, 60%] -> 3.0, (0%, 50%]
-> 2.0.
b) w ramach projektu ocena wyznaczona jest na podstawie:
- wykonania przez studenta zadań/projektów do wykonania samodzielnego lub w zespole,
- oceny wykonania projektu, w tym umiejętności pracy w zespole.
Student może zdobyć maks. 100 pkt., a ocena końcowa wyliczana jest z wykorzystaniem następującej skali: (90%, 100%] -> 5.0, (80%, 90%] -> 4.5, (70%, 80%] -> 4.0, (60%, 70%] -> 3.5, (50%, 60%] -> 3.0, (0%,
50%] -> 2.0.

Treści programowe

.NET Framework, LINQ, Entity Framework, Windows Presentation Foundation, programowanie funkcyjne z F#, programowanie dynamiczne z Ruby, Ruby on Rails, wykorzystanie chmury do rozwoju

aplikacji z Windows Azure, programowanie rozproszone z Akka.NET, tworzenie aplikacji webowych z wykorzystaniem ASP.NET, tworzenie skryptów z wykorzystaniem Powershell.

Tematyka zajęć

brak

Metody dydaktyczne

Wykład: prezentacja multimedialna, ilustrowana przykładami podawanymi na tablicy.
Projekt: prezentacja multimedialna prezentacja ilustrowana przykładami podawanymi na tablicy oraz wykonanie zadań podanych przez prowadzącego - ćwiczenia praktyczne.

Literatura

Podstawowa
1. L. Bass, P. Clements, R. Kazman, "Software architecture in practice", WNT
2. P. Kruchten, "The Rational Unified Process-An Introduction", Addison-Wesley
3. A. Troelsen, P. Japikse, "C# 6.0 and the .NET 4.6 Framework", Apress
4. D. Syme, A. Granicz, A. Cisternino, "Expert F# 4.0", Apress
Uzupełniająca

Bilans nakładu pracy przeciętnego studenta

	Godzin	ECTS
Łączny nakład pracy	150	6,00
Zajęcia wymagające bezpośredniego kontaktu z nauczycielem	60	2,50
Praca własna studenta (studia literaturowe, przygotowanie do zajęć laboratoryjnych/ćwiczeń, przygotowanie do kolokwiów/egzaminu, wykonanie projektu)	90	3,50