KARTA OPISU PRZEDMIOTU - SYLABUS

Nazwa przedmiotu
Narzędzia informatyki [S1Cybez1>NINF]

Przedmiot

Kierunek studiów
Cyberbezpieczeństwo

Rok/Semestr
2/3

Studia w zakresie (specjalność)
–

Profil studiów
ogólnoakademicki

Poziom studiów
pierwszego stopnia

Język oferowanego przedmiotu
polski

Forma studiów
stacjonarne

Wymagalność
obligatoryjny

Liczba godzin

Wykład
16

Laboratorium
16

Inne
0

Ćwiczenia
0

Projekty/seminaria
0

Liczba punktów ECTS

2,00

Koordynatorzy

dr inż. Łukasz Kułacz
lukasz.kulacz@put.poznan.pl

Wykładowcy

Wymagania wstępne

Student powinien posiadać podstawową wiedzę z zakresu programowania, w szczególności pojęcia zmiennych, klas i funkcji. Powinien posiadać umiejętność implementacji prostych programów i dostrzegać ryzyko związane z tworzeniem niedopracowanego oprogramowania.

Cel przedmiotu

Celem przedmiotu jest przekazanie studentom podstawowej wiedzy dotyczącej narzędzi wykorzystywanych podczas tworzenia oprogramowania. Narzędzia te dotyczą przede wszystkim przechowywania i kontroli wersji kodu w lokalnych i zdalnych repozytoriach, współpracy nad tym samym kodem, technik zapewnienia jakości poprzez testy o różnym poziomie szczegółowości i skomplikowania, podstaw metod ciągłej integracji i ciągłego wdrażania oraz rozszerzeń do edytorów kodu ułatwiających tworzenie dobrej jakości oprogramowania.

Przedmiotowe efekty uczenia się

Wiedza:
Student ma podstawową wiedzę teoretyczną i praktyczną w zakresie repozytoriów oprogramowania, testów oprogramowania, automatyzacji procesu testowania i zasad tworzenia poprawnego i czytelnego

kodu. Student zna podstawowe narzędzia ułatwiające prace nad oprogramowaniem.

Umiejętności:
Student potrafi efektywnie wykorzystać do pracy lokalne repozytorium oprogramowania, a także synchrozniować efefy swojej pracy poprzez zdalne repozytorium. Student potrafi współpracować z innymi twórcami oprogramowania poprzez zdalne repozytoria i łączyć z nimi swoje oprogramowanie. Student potrafi przygotować podstawowe testy do kodu i przetestować swój kod zarówno lokalnie jak i poprzez automatyzację połączoną z zdalnym repozytorium oprogramowania. Student potrafi również wykorzystać podstawowe narzędzia do ułatwienia pracy nad tworzeniem oprogramowania i tym samym poprawić jakość tworzonego kodu.

Kompetencje społeczne:
Student ma świadomość możliwości i ograniczeń podczas tworzenia oprogramowania, a w szczególności konieczność zapewnienia dobrej jakości kodu. Rozumie potencjalny wpływ niepoprawnie przygotowanego oprogramownia. Jest świadomy wyzwań i zagrożeń związanych z współpracą wielu programistów nad tym samym kodem.

Metody weryfikacji efektów uczenia się i kryteria oceny

Efekty uczenia się przedstawione wyżej weryfikowane są w następujący sposób:

Efekty uczenia się przedstawione wyżej weryfikowane są w następujący sposób:
W zakresie laboratorium weryfikowanie założonych efektów kształcenia realizowane jest przez: ocenę merytoryczną zadań indywidualnych lub grupowych wykonywanych w ramach zajęć lub w postaci zadań do wykonania po zajęciach, aktywność na zajęciach.
W zakresie wykładu weryfikowanie założonych efektyów kształcenia realizowane jest poprzez pisemne zaliczenie w charakerze testu, który może zawierać pytania jednokrotnego, wielokrotnego wyboru oraz pytania otwarte.
W każdej formie zaliczenia przedmiotu ocena zależy od liczby zdobytych przez studenta punktów w stosunku do maksymalnej liczby punktów obowiązkowych. Warunkiem pozytywnego zaliczenia jest otrzymanie co najmniej 50% punktów możliwych do zdobycia. Zależność oceny od liczby punktów definiuje Regulamin Studiów. Dodatkowo zasady zaliczania przedmiotu i dokładne progi zaliczeniowe zostaną przekazane studentom na początku semestru z wykorzystaniem uczelnianych systemów elektronicznych oraz na pierwszych zajęciach (w każdej formie zajęć).

Treści programowe

1. Repozytorium kodu.
2. Testy jednostkowe, integracyjne, funkcjonalne i wydajności.
3. Automatyzacja testów i cykl projektowania oprogramowania (CI/CD).
4. Narzędzia kontroli jakości oprogramowania.
5. Narzędzia ułatwiające pracę podczas tworzenia oprogramowania.

Tematyka zajęć

1. Podstawy systemu git.
2. Współpraca z wykorzystaniem systemu git.
3. Przygotowanie testów jednostkowych i integracyjnych.
4. Przygotowanie testów funkcjonalnych i wydajności.
5. Automatyzacja testów na platformie GitHub.
6. Kontrola jakości kodu.
7. Narzędzia ułatwiające pracę podczas tworzenia oprogramowania.

Metody dydaktyczne

Wykład: prezentacja multimedialna, uzupełniana przykładami i dodatkowymi wyjaśnieniami na podstawie fragmentów kodu.
Laboratoria: praca przy komputerze, wykonanie indywidualnych zadań, wykonanie grupowych zadań.

Literatura

Podstawowa:

1. Jakość oprogramowania. Podręcznik dla profesjonalistów. Michał Sobczak
2. TDD w praktyce. Niezawodny kod w języku Python. Harry Percival
3. Git i GitHub. Kontrola wersji, zarządzanie projektami i zasady pracy zespołowej. Mariot Tsitoara

Uzupełniająca:
1. Testowanie i jakość oprogramowania. Modele, techniki, narzędzia. Adam Roman

Bilans nakładu pracy przeciętnego studenta

	Godzin	ECTS
Łączny nakład pracy	57	2,00
Zajęcia wymagające bezpośredniego kontaktu z nauczycielem	32	1,00
Praca własna studenta (studia literaturowe, przygotowanie do zajęć laboratoryjnych/ćwiczeń, przygotowanie do kolokwiów/egzaminu, wykonanie projektu)	25	1,00