KARTA OPISU PRZEDMIOTU - SYLABUS

Nazwa przedmiotu
Narzędzia wirtualizacyjne [S1MiKC2>NW]

Przedmiot

Kierunek studiów
Mikroelektronika i komunikacja cyfrowa

Rok/Semestr
3/5

Studia w zakresie (specjalność)
–

Profil studiów
ogólnoakademicki

Poziom studiów
pierwszego stopnia

Język oferowanego przedmiotu
polski

Forma studiów
stacjonarne

Wymagalność
obieralny

Liczba godzin

Wykład
15

Laboratorium
0

Inne
0

Ćwiczenia
0

Projekty/seminaria
15

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 aplikacji sieciowych.

Cel przedmiotu

Celem przedmiotu jest przekazanie studentom podstawowej wiedzy dotyczącej wirtualizacji i konteneryzacji. Poprzez wirtualizację student może odizolować projektowaną aplikację lub usługę od systemu operacyjnego hosta, dzięki czemu możliwe jest zapewnienie niezależnego od sprzętu, wersji systemu i sterowników działania kodu. Dodatkowo w ramach przedmiotu przedstawione zostaną zagadnienia związane z konteneryzacją, przede wszystkim, w kontekście skalowalności i niezależności aplikacji projektowanych jako tzw. mikrousługi.

Przedmiotowe efekty uczenia się

Wiedza:
Student ma podstawową wiedzę teoretyczną i praktyczną w zakresie wirtualizacji oprogramowania, konteneryzacji i orkiestracji kontenerów. Student zna podstawowe narzędzia umożliwiające tworzenie i

używanie kontenerów, a także zarządzanie nimi. Student zna cechy i ograniczenia wynikające z zastosowania kontenerów do wdrażania gotowego rozwiązania.

Umiejętności:
Student potrafi tworzyć wirtualne systemy operacyjne, korzystać z nich i nimi zarządzać. Student potrafi przygotować proste aplikacje i następnie umieszczać je w kontenerach, uruchamiać i aktualizować. Student potrafi łączyć kilka mikrousług w działający system i podmieniać elementy systemu bez konieczności uruchamiania całego systemu od nowa.

Kompetencje społeczne:
Student ma świadomość możliwości i ograniczeń związanych z tworzeniem oprogramowania i umieszczaniem go wewnątrz kontenerów. Rozumie potencjalne zyski związane ze skalowalnością i zwiększoną niezawodnością systemu jednocześnie dostrzegając ryzyka związane z zastosowaniem takiego rozwiązania.

Metody weryfikacji efektów uczenia się i kryteria oceny

Efekty uczenia się przedstawione wyżej weryfikowane są w następujący sposób:

W zakresie projektu weryfikowanie założonych efektów kształcenia realizowane jest przez: ocenę merytoryczną realizacji projektu indywidualnego. Projekt polega na zrealizowaniu aplikacji zbudowanej z kilku połączonych mikrousług. Każdy z elementów aplikacji ma przypisaną odpowiednią liczbę punktów. Punktacji podlega również połączenie niezależnych elementów aplikacji.
W zakresie wykładu weryfikowanie założonych efektów kształcenia realizowane jest poprzez pisemne zaliczenie w charakterze testu.
Zalicznie (dotyczy obu częście) wymaga uzyskania przynajmniej 50% punktów. Przyjęta skala ocen to:
2.0 <0%; 50%>
3.0 (50%; 60%>
3.5 (60%; 70%>
4.0 (70%; 80%>
4.5 (80%; 90%>
5.0 (90%; 100%>

Treści programowe

W ramach przedmiotu przedstawione zostaną zasady wirtualizacji systemów operacyjnych, zasady konteneryzacji, koncepcja mikrousług, a także wybrane zagadnienia związane z orkiestracją kontenerów.

Tematyka zajęć

Wykłady:
1. Zasady wirtualizacji systemów operacyjnych. (1 jednostka zajęć)
2. Koncepcja konteneryzacji. Docker - koncepcja obrazu i kontenera. (2 jednostka zajęć)
3. Porównanie koncepcji monolitu i mikrousług. (1 jednostka zajęć)
4. Podstawy orkiestracji kontenerów. Kubernetes - zarządzadnie kontenerami. (1 jednostka zajęć)
5. Tworzenie usługi. Narzędzia docker compose i Helm - łączenie wielu kontenerów celem utworzenia usługi. (1 jednostka zajęć)
6. Integracja mikrousług. (1 jednostka zajęć)
Projekt:
Przygotowanie (w grupie kilkuosobowej) projektu na wybrany temat zrealizowany za pomocą narzędzi wirtualizacji i konteneryzacji.

Metody dydaktyczne

Wykład: prezentacja multimedialna, uzupełniana przykładami i dodatkowymi wyjaśnieniami na podstawie fragmentów kodu.
Projekt: praca przy komputerze, wykonanie indywidualnego zadania. W ramach zajęć studenci mają możliwość uzyskania od prowadzącego wskazówki i pomoc w realizacji przydzielonego zadania. Dodatkowo, również poza zajęciami studenci mogą skorzystać z dodatkowych konsultacji.

Literatura

Podstawowa:
1. Rozwijanie mikrousług w Pythonie. Budowa, testowanie, instalacja i skalowanie. Tarek Ziade
2. Architektura aplikacji w Pythonie. TDD, DDD i rozwój mikrousług reaktywnych. Harry Percival
3. Nauka Dockera w miesiąc. Elton Stoneman
4. Nauka Kubernetesa w miesiąc. Elton Stoneman

Uzupełniająca:
1. Docker. Niezawodne kontenery produkcyjne. Praktyczne zastosowania. Sean Kane

Bilans nakładu pracy przeciętnego studenta

	Godzin	ECTS
Łączny nakład pracy	60	2,00
Zajęcia wymagające bezpośredniego kontaktu z nauczycielem	30	1,00
Praca własna studenta (studia literaturowe, przygotowanie do zajęć laboratoryjnych/ćwiczeń, przygotowanie do kolokwiów/egzaminu, wykonanie projektu)	30	1,00