Poniższe propozycje tematów prac dyplomowych (zima 2024/25) bazują w większości na zagadnieniach realizowanych w ramach „Koła naukowego systemów operacyjnych i programowania”. Tym niemniej zachęcam wszystkich Państwa do przemyślenia własnych interesujących Was zagadnień i zgłaszania się z nimi z jak największym wyprzedzeniem do prowadzących. Tematy poniżej dotyczą prac dyplomowych inżynierskich.

1. **Aplikacja do zarządzania użytkownikami serwera WireGuard (1 lub 2 osobowa)**
   Opis: Zbudowanie aplikacji umożliwiającej dostęp do wybranych serwerów Katedry Informatyki dla dyplomantów i członków kół naukowych. Aplikacja powinna składać się z panelu użytkownika i panelu administratora. Użytkownik powinien móc zgłosić prośbę o dostęp do serwera lub wybranych serwerów. Administrator powinien móc zarządzać dostępem do serwerów na bazie spisu serwerów, użytkowników i zgłoszeń dostępu do nich składanych przez studentów.
   W wersji jednoosobowej aplikacja powinna działać wykorzystując skrypty bash a w wersji 2 osobowej wykorzystywać system automatyzacji zadań ansible lub oferować dostęp do serwera NAS.
2. **Klaster Proxmox z automatycznym tworzeniem maszyn wirtualnych dla środowiska przygotowującego do egzaminów związanych z kodowaniem wykorzystująca narzędzia Vagrant i Ansible (2 osobowa)**
   Opis: Zbudowanie aplikacji pozwalającej na tworzenie zespołów maszyn wirtualnych jak dla kursów Akademii Red Hat RH124 i RH134 (minimum 2 maszyny wirtualne jak np. w kursie RH124). Każda z maszyn wirtualnych powinna być tworzona automatycznie wykorzystując Vagrant. Para (lub więcej) maszyn powinno posiadać możliwość odseparowanej od innych zespołów maszyn komunikacji wzajemnej.
   Środowisko egzaminacyjne powinno składać się z bazy pytań i rozwiązań oraz playbooków Ansible umożliwiających sprawdzenie poprawności rozwiązania i manualnej ingerencji egzaminatora w wynik każdego zadania (możliwość poprawy uzyskanej za rozwiązanie liczby punktów wraz z komentarzem). Istota części związanej z systemem ansible polega na sprawdzeniu poprawności rozwiązania zadania nie tylko typu a/b/c/d ale przede wszystkim zagadnień administracyjnych typu „skonfiguruj środowisko tak aby użytkownik studentX mógł uruchamiać polecenie „shutdown” bez pytania o hasło administratora” lub dostęp zdalny poprzez ssh bez podawania hasła itp. (kilka poleceń, zamieszczenie stosownego wpisu w odpowiednim pliku konfiguracyjnym).
3. **Wykorzystanie środowiska chmurowego OpenStack (model IaaS) do budowy zdalnego laboratorium wykorzystującego maszyny z systemem Linux (1 lub 2 osoby)**
   Opis: Zbudowanie chmury OpenStack pozwalającej na tworzenie ad hoc maszyn wirtualnych jak np. dla kursów Akademii Red Hat RH104 lub na potrzeby przedmiotu „Wstęp do systemów operacyjnych”. Zdalny dostęp powinien oferować do wyboru terminal tekstowy lub środowisko graficzne.
   W przypadku pracy 2-osobowej możliwe są dwa kierunki (dla 2-giej osoby):
   - umożliwienie dostępu maszynom wirtualnym do wspólnej karty graficznej dedykowanej do obliczeń CUDA (z elementarnymi przykładami programów CUDA) lub
   - tworzenie zespołów maszyn zdalnych jak np. dla bardziej zaawansowanych kursów Akademii Red Hat (workstation, servera, serverb …).
4. **Opracowanie, w postaci kursu, materiałów do nauki tworzenia własnego 64-bitowego systemu operacyjnego typu UNIX/Linux (1 lub 2 osoby)**
   Opis: Zasadniczo jest to temat, który realizuje już na kole naukowym Pan Patryk. Jeżeli zechce poduczyć kogoś w tej tematyce to można ją rozszerzyć i dodać 2-gą osobę.
5. **Emulator mikrokomputera 8-bitowego**
   Opis: aplikacja emulująca działanie mikroprocesora oraz współpracę z pamięcią i urządzeniami peryferyjnymi. Stworzenie prostego systemu operacyjnego umożliwiającego

wykonywanie podstawowych operacji wejścia/wyjścia za pomocą terminala. Maciej Janczara

6. **Automatyzacja wdrażania aplikacji w OpenShift przy użyciu GitOps**
   Opis:
   - Automatyzacja procesu budowania platformy OpenShift przy użyciu Ansible,
   - Przechowywanie infrastruktury w formie kodu,
   - Automatyzacja budowania obrazów z wykorzystaniem narzędzi GitOps (np. ArgoCD, Flux),
   - Regularne przebudowywanie obrazów w celu eliminowania potencjalnych luk bezpieczeństwa.

   Diego Lavayen, Hubert Bukowski.

7. **Budowa systemu telemetrii pojazdu pod systemem Linux**
   Opis: Praca realizowana na potrzeby koła naukowego Hydrogreen prowadzonego przez dr hab. inż. Jacka Czernigowskiego. Obejmuje obsługę modułu GPS, podlaczenie do magistrali can, wysyłanie danycbh przez LoRa do stacji bazowej, ew. obsługę widoku z dwóch kamer zewnętrznych dzialających jako lusterka. Michał Gagoś
8. **System aukcji internetowych**
   Opis: projekt dotyczy zbudowania strony internetowej, bazy danych i mechanizmu aukcji. Aukcje, byłby uruchamiane o określonej godzinie i sam mechanizm musi wspierać wielu użytkowników wysyłających żądania w tym samym czasie, co będzie stanowić największy problem do rozwiązania (synchronizacja, cacheowanie, utrzymywanie połączenia). Bartłomiej Niezbecki i Kacper Wiącek.
9. **Aplikacja natywna i internetowa wspomagająca proces uczenia**
   Opis: Praca powinna objąć projektowanie i implementację aplikacji internetowej i natywnej na wybrane platformy. Tematem projektu jest aplikacja wspomagająca proces uczenia poprzez wykorzystanie fiszek (flashcards) i generowanych testów. Utworzone zestawy mogą być współdzielone między użytkownikami i serwis umożliwia współpracę podczas tworzenia zestawu. Inspiracją projektu są strony typu Quizlet. Istotną różnicą jest, że projektowana aplikacja ma możliwość przechowywania danych lokalnie, co pozwala na korzystanie z tego rodzaju serwisu bez dostępu do Internetu. Również dzięki takiemu rozwiązaniu użytkownik nie musi “zaufać” serwisowi i osobom go prowadzącym, gwarantując bezpieczeństwo danych. Karolina Wołodko (aplikacja internetowa), Patryk Tofil (aplikacja natywna), Krzysztof Włosek (backend aplikacji).
10. **Klaster Proxmox z automatycznym tworzeniem maszyn wirtualnych dla środowiska przygotowującego do testów z systemu operacyjnego Linux.**
    Opis: Zbudowanie aplikacji pozwalającej na tworzenie zespołów maszyn wirtualnych do nauki systemu Linux. Każda z maszyn wirtualnych powinna być tworzona automatycznie wykorzystując Vagrant. Para (lub więcej) maszyn powinna posiadać możliwość odseparowanej od innych zespołów maszyn komunikacji wzajemnej.
    Środowisko egzaminacyjne powinno składać się z bazy pytań i rozwiązań oraz playbooków Ansible umożliwiających sprawdzenie poprawności rozwiązania i manualnej ingerencji egzaminatora w wynik każdego zadania (możliwość poprawy uzyskanej za rozwiązanie liczby punktów wraz z komentarzem). Istota części związanej z systemem ansible polega na sprawdzeniu poprawności rozwiązania zadania nie tylko typu a/b/c/d ale przede wszystkim zagadnień administracyjnych typu „skonfiguruj środowisko tak aby użytkownik studentX mógł uruchamiać polecenie „shutdown” bez pytania o hasło administratora” lub dostęp zdalny poprzez ssh bez podawania hasła itp. (kilka poleceń, zamieszczenie stosownego

wpisu w odpowiednim pliku konfiguracyjnym). VireGuard.
Maciej Dolecki, IINS 6.1/1, Borys Baczewski.

11. **Analiza porównawcza narzędzi Ansible, Terraform i Puppet w kontekście zarządzania infrastrukturą aplikacji w modelu SaaS.**
    Opis: Przedstawienie cech poszczególnych narzędzi, wraz z identyfikacją ich zalet i wad, oraz wskazanie najlepszych przypadków ich zastosowania w środowiskach SaaS. Analiza takich kryteriów jak skalowalność, łatwość wdrożenia, zdolność do utrzymania wymaganej infrastruktury oraz integracja z usługami chmurowymi. Dodatkowo, zaprezentowaie studium przypadku ilustrujące praktyczne zastosowanie omawianych narzędzi w rzeczywistym projekcie SaaS.
    Ramowy plan:
    1. Wybór przykładowej aplikacji open-source w modelu SaaS
    2. Utworzenie infrastruktury serwerowej dla aplikacji przy wykorzystaniu Terraform i wybranej chmury, bądź Vagranta
    3. Konfiguracja środowiska i instalacja aplikacji przy użyciu Ansible
    4. Utrzymywanie środowiska przy pomocy Puppet
    5. Analiza porównawcza narzędzi - wady, zalety, etc.
    6. Wnioski końcowe
    Aplikacja bazowa: gitlab w wersji community, albo moodle.
    Jakub Wójcik
12. **Automatyzacja zarządzania kontenerami przy użyciu Ansible i Kubernetes (OpenShift)**
    Opis: opracowanie i wdrożenie systemu automatyzacji zarządzania kontenerami w środowisku Kubernetes/OpenShift przy użyciu Ansible. PUproszczenie procesu wdrażania, skalowania i monitorowania aplikacji kontenerowych, eliminując konieczność ręcznej konfiguracji klastra.
    Zakres pracy:
    Przygotowanie środowiska – konfiguracja Kubernetes/OpenShift oraz Ansible.
    Stworzenie aplikacji demonstracyjnej – prostego REST API z bazą danych, uruchamianego w kontenerze.
    Automatyzacja wdrażania – przygotowanie manifestów Kubernetes i skryptów Ansible do automatycznego wdrażania aplikacji.
    Zarządzanie zasobami i skalowanie – wykorzystanie autoskalowania oraz definiowanie limitów zasobów.
    Monitoring i logowanie – integracja z Prometheus i Grafana do monitorowania stanu aplikacji i klastra.
    Dokumentacja i analiza wyników – opis działania, testy wydajności oraz wnioski dotyczące efektywności automatyzacji.
    Konrad Filipek I1N 6.2