Хошаба А.М.,
кандидат технических наук, доцент,
доцент кафедры программного обеспечения,
Винницкий национальный технический университет

РАЗРАБОТКА МИКРОСЕРВИСНЫХ АРХИТЕКТУР НА ПРИМЕРЕ СОЗДАНИЯ КЛАСТЕРА RabbitMQ

Анотація: *Пропонується розробка мікросервісної архітектури на прикладі створення кластера RabbitMQ. В роботі показано особливості використання мікросервісних та сервіс-орієнтованих архітектур. На прикладі створення кластера RabbitMQ визначені переваги використання мікросервісних архітектур та контейнерних технологій. Показані приклади використання кластера RabbitMQ у веб додатках.*
Ключові слова: *мікросервісні архітектури, сервіс-орієнтовані архітектури, особливості використання мікросервісні архітектури, кластер RabbitMQ.*

Abstract: *The paper proposes the development of a microservice architecture using the example of creating a RabbitMQ cluster. Features of using microservice and service-oriented architectures are shown. Using the example of creating a RabbitMQ cluster, the advantages of using microservice architectures and container technologies are determined. Examples of using the RabbitMQ cluster in web applications are shown.*
Keywords: *microservice architectures, service oriented architectures, features of using microservice architectures, RabbitMQ cluster.*

Особенности использования микросервисных архитектур

Микросервисная архитектура (МА) представляет собой особый вариант сервис-ориентированной структур программного обеспечения. Такая структура ориентирована на взаимодействие небольших, слабо связанных и легко изменяемых модулей. Эти модули получили название микросервисов.

Начало распространения МА считается [1-3] середина 2010-х годов когда стали развиваться практики гибкой разработки программного обеспечения и средства DevOps.

МА появилась как продолжение в развитии сервис-ориентированным системам. Известно [4], что сервис-ориентированные архитектуры (СОА) являются достаточно сложными программными системами, а взаимодействие между отдельными модулями которые опираются на стандартизованные тяжеловесные протоколы (такие, как SOAP, XML-RPC). В отличии от СОА [5], в МА используются компоненты, которые выполняют относительно элементарные функции. Также, в МА используют легковесные сетевые коммуникационные протоколы, к примеру JSON, Protocol Buffers, Thrift.

Основой функционирования МА является уменьшение степени взаимодействия увеличение связности между отдельными модулями, что позволяет проще добавлять и изменять функциональность системы в любое время. Также, при использовании МА акцент выполняется на простоту, независимость развёртывания и обновления каждого из микросервисов. Модули могут быть реализованы с помощью различных языков программирования, фреймворков.

Отдельные модули могут также выполняться в различных средах контейнеризации, виртуализации, облачных платформах, под управлением различных операционных систем и аппаратного обеспечения.

Философия МА похожа на методологию Unix, согласно которой:

- каждый сервис должен «делать что-то одно, и делать это хорошо»;
- взаимодействовать с другими модулями простыми средствами;
- микросервисы минимальны и предназначены для реализации минимальной функциональности.

Наиболее популярными средами реализации микросервисов являются системы управления контейнерными приложениями: Kubernetes, надстройки OpenShift и CloudFoundry, Docker Swarm, Apache Mesos и другие.

При работе с такими средами каждый из микросервисов изолируется в отдельный контейнер или небольшую группу контейнеров, доступную по компьютерной сети другим микросервисам или пользователям. Также, к функциям таких платформ относятся управление средой

оркестрации, обеспечение отказоустойчивости и балансировки нагрузки. Типовой практикой является использование средств DevOps, систем непрерывной интеграции, обеспечение автоматизации обновления программных приложений и развёртывания микросервисов.

Необходимость использования кластера RabbitMQ

RabbitMQ представляет собой менеджер очередей который обмениваться данными между процессами, приложениями и серверами [6,7]. В его основные функции входят определение и работа с очередями к которым могут подключатся различные приложения и передавать или получать сообщения (рис. 1).

Сообщение может включать в себя любую информацию. К примеру, это может быть простое текстовое сообщение или информация о процессе или задаче которую необходимо выполнить другим приложением.

Менеджер очередей RabbitMQ может сохранять сообщение до тех пор пока другое приложение или сервер, которому адресовано сообщение, не подключиться к кластеру и не заберет (получит) сообщение из очереди. Затем проложение-получатель обрабатывает сообщение нужным ему образом.

Рисунок 1. Общая схема работы RabbitMQ

Другим примером использования RabbitMQ является сценарий когда веб приложение позволяет пользователям загружать информацию на сайт. В

этом примере сайт должен обработать эту информацию, генерировать PDF и отправлять данные обратно пользователю на электронную почту.

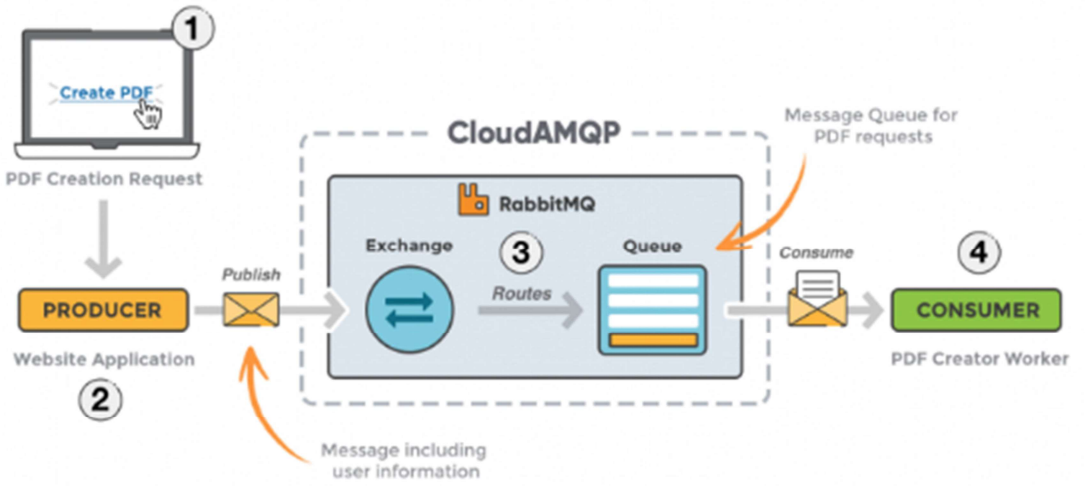


Рисунок 2. Работа менеджера RabbitMQ с веб приложением

В этом случае, обработка информации, генерация PDF и отправка email обычно занимает несколько секунд и состоит в следующем. Когда пользователь введет информацию в веб интерфейс, приложение создаст задание на генерацию PDF и вся информация в сообщении и данные пользователя будут помещены в очередь, которая определена в кластере RabbitMQ. Данный механизм работы менеджера RabbitMQ с веб приложением (рис. 2) заключается в следующем:

1. Пользователь формирует запрос на создание PDF документа.

2. Приложение (Producer) посылает сообщение в менеджер RabbitMQ, включая в запрос информацию про имя и электронный адрес пользователя.

3. Обработчик принимает сообщения от приложения поставщика и направляет его в нужную очередь сообщений RabbitMQ.

4. Воркер обработки сообщений (Consumer) получает задание и начинает генерацию PDF документа.

Данный пример показывает как можно эффективно использовать очереди сообщений для разработки веб приложений.

Создание кластера RabbitMQ

В кластере менеджер RabbitMQ использует несколько сервисов, у которых существуют общие пользователи, настройки и очереди. Сервисы

могут добавляться и удаляться в процессе выполнения, располагаться на разных микросервисах. Однако, для подключённого клиента они будут выглядеть как один кластерный RabbitMQ сервис. Такой механизм является достаточно эффективным для горизонтального мастшабирования в случаях, когда пользователей становится много для обработки запросов одного брокера.

Механизм кластеризации микросервисов является подобным к принципам репликации или высоконагруженных ресурсов. Использование первого и второго указанного принципа не влияет на доступность данных и работу сервисов при больших нагрузках. При классическом использовании кластеров, узлы не являются взаимозаменяемыми. Хотя пользователи и настройки прикладных систем будут дублироваться на каждом из узлов, где бы они не создавались. Однако, с очередями сообщений основные операции выполняются по другим принципам. Эти отличии заключаются в том, что если какой-либо из узлов перешел в состояние оффлайна, то его очереди будут недоступны для других приложений.

Создание кластера RabbitMQ не является довольно сложным процессом. Для этого необходимо всем узлам будущего кластера нужно задать одинаковую Erlang cookie и вызвать команду rabbitmqctl join_cluster. Далее, необходимо подготавить менеджер RabbitMQ узлы для микросервисов Docker или Kubernetes.

Для создания кластера необходимо иметь как минимум два независимых узла (хоста). Самый простой способ получить их — запустить два Docker контейнера с RabbitMQ внутри. В репозитории Docker Hub для этих целей есть два образа: rabbitmq и rabbitmq:management. Более лучше использовать второй так как в нём уже присутствует панель веб-администратора.

Также, необходимо чтобы контейнеры могли обмениваться информацией между собой в общей компьютерной сети, применять доменные имена Erlang cookie. Выполняется это требования с помощью docker-compose.

При создании файла конфигурации, автоматически будут присваивать имена хостов, выполняться сетевые соединения, передаваться параметры между хостами, запускаться сервисы и т.д. К примеру, такой файл конфигурации может иметь следующий вид:

```
version: '2'
services:
  rabbit:
    image: rabbitmq:management
    hostname: rabbit
    ports:
      - "15677:15678"
    environment:
      - RABBITMQ_ERLANG_COOKIE='secret'
  cent01:
    image: rabbitmq:management
    hostname: hamster
    ports:
      - "15673:15672"
    environment:
      - RABBITMQ_ERLANG_COOKIE='secret'
```

В этом файле создаются два сервиса (контейнера), которые после запуска формируют RabbitMQ узлы кластера с именами rabbit и cent01. Имена контейнерных хостов (hostname) заданы явно. Они будут использоваться в именах RabbitMQ узлов для взаимодействия между ними.

Панель администратора для кластера RabbitMQ будет доступна из контейнера через порт 15678 со стороны хоста. Также, в официальный образ rabbitmq можно передать Erlang cookie в качестве переменой окружения — RABBITMQ_ERLANG_COOKIE.

Далее, запускаются сервисы docker-compose.yml следующим образом:

```
docker-compose up
```

#Creating network "cluster_default" with the default driver

#Creating cluster_hamster_1

#...

#hamster_1 |

#hamster_1 | RabbitMQ 3.6.9. Copyright (C) 2007-2019 Pivotal Software, Inc.

#.....

Таким образом выполняется создание, запуск и разработка микросервисных архитектур на примере кластера RabbitMQ.

Заключение

В работе предложена разработка микросервисной архитектуры на примере создания кластера RabbitMQ. Показаны особенности использования микросервисних и сервис-ориентированных архитектур. На примере создания кластера RabbitMQ определенны преимущества использования микросервисних архитектур и контейнерных технологий. Показаны примеры использования кластера RabbitMQ в веб приложениях.

Список использованой литературы

1. Irakli Nadareishvili. Microservice Architecture: Aligning Principles, Practices, and Culture. O'Reilly Media; 1 edition, 2018. P. 146.
2. Sam Newman. Building Microservices: Designing Fine-Grained Systems 1st Edition. O'Reilly Media; 1 edition, 2015. P. 280.
3. Susan Fowler. Production-Ready Microservices, 2017. O'Reilly Media; 1 edition. P. 172.
4. Thomas Erl. Service-Oriented Architecture (SOA): Concepts, Technology, and Design. Prentice Hall, 2005. P. 792.
5. Robert C. Martin. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Prentice Hall, 2017. P. 432.
6. Gavin M. Roy. RabbitMQ in Depth. Manning Publications; 1 edition, 2017. P. 264.

7. Alvaro Videla. RabbitMQ in Action: Distributed Messaging for Everyone. Manning Publications; 1 edition, 2012. P. 312.

Черноволик Г. О.,
кандидат технічних наук,
доцент кафедри програмного забезпечення,
Вінницький національний технічний університет, Україна
Гончарук Д. В.,
студент групи 2ПІ-18м, факультет інформаційних технологій і комп'ютерної інженерії, Вінницький національний технічний університет, Україна

РОЗРОБКА ПРОГРАМНОГО ЗАСОБУ ДЛЯ VR 3D ВІДОБРАЖЕННЯ ІСТОРИЧНИХ ПАМ'ЯТОК

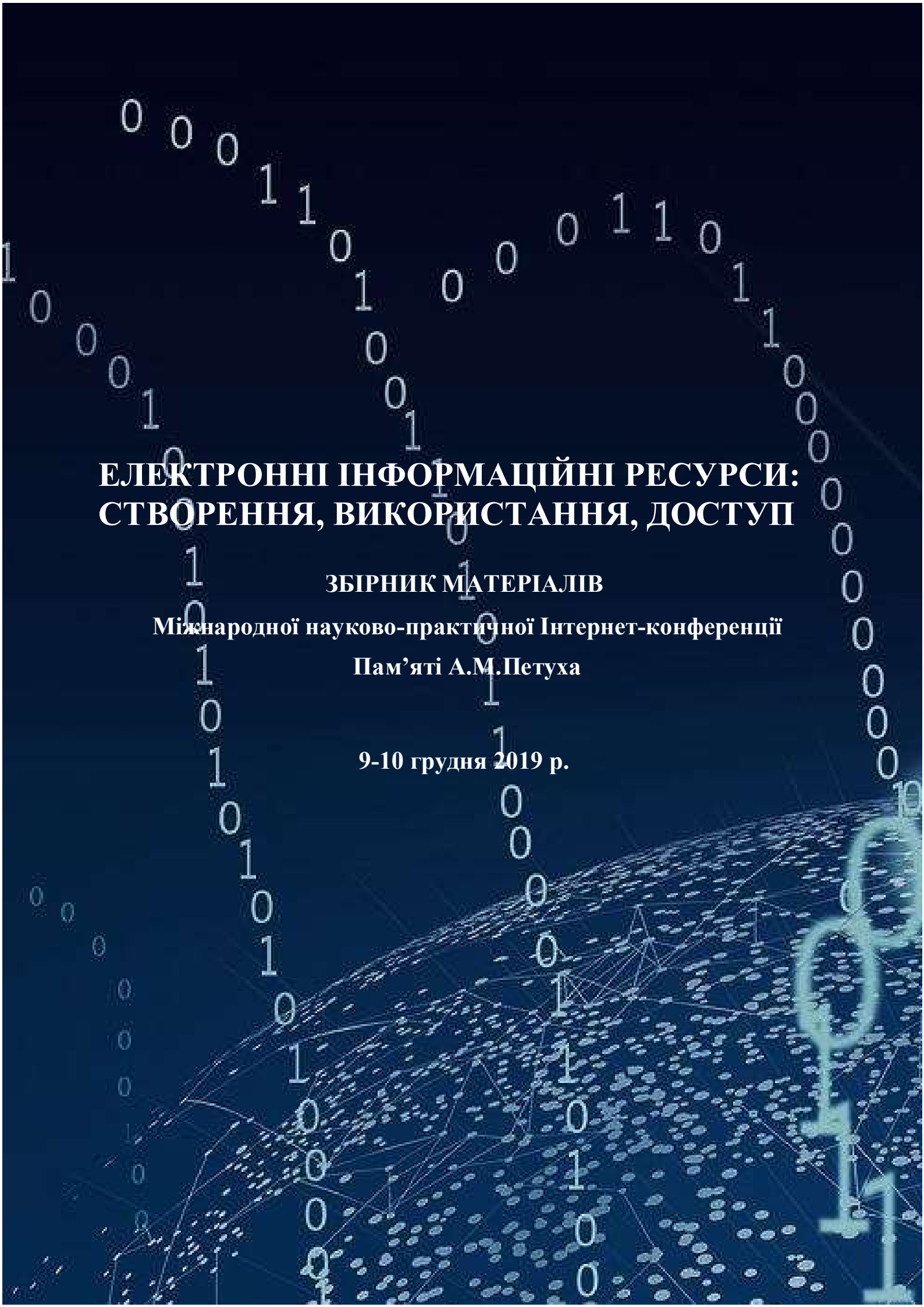
Анотація: *Розроблено метод та програмний засіб для VR 3D відображення історичних пам'яток*
Ключові слова: *геолокація, 3D відображення, віртуальна реальність.*

Abstract*: The method and software for VR 3D display of historical monuments has been developed*
Keywords: *geolocation, 3D rendering, virtual reality.*

Вступ

Розробка орієнтована на використання у сфері туризму, у навчальних цілях (до прикладу, під час уроків історії), на людей, які цікавляться історією та пам'ятками архітектури для свого саморозвитку, а також на активних користувачів соцмереж (яких зараз переважна більшість).

Учасник віртуального туру є лише пасивним глядачем. Але головна перевага таких віртуальних турів – їх доступність для всіх користувачів мережевих технологій та відсутність необхідності будь-яких істотних додаткових витрат. Цільова аудиторія для таких віртуальних турів є надзвичайно широкою, зокрема малозабезпечені верстви населення. Фактори



ЕЛЕКТРОННІ ІНФОРМАЦІЙНІ РЕСУРСИ: СТВОРЕННЯ, ВИКОРИСТАННЯ, ДОСТУП

ЗБІРНИК МАТЕРІАЛІВ

Міжнародної науково-практичної Інтернет-конференції

Пам’яті А.М.Петуха

9-10 грудня 2019 р.

Міністерство освіти і науки України
Вінницький національний технічний університет
Національна академія Державної прикордонної
служби України ім. Богдана Хмельницького
Вінницький національний медичний
університет ім. М.І. Пирогова
Вінницька академія неперервної освіти
КЗ Сумський обласний інститут післядипломної
педагогічної освіти
Люблінська політехніка (Польща)
Новий університет Лісабону (Португалія)

ЕЛЕКТРОННІ ІНФОРМАЦІЙНІ РЕСУРСИ: СТВОРЕННЯ, ВИКОРИСТАННЯ, ДОСТУП

ЗБІРНИК МАТЕРІАЛІВ
Міжнародної науково-практичної Інтернет-конференції
Пам'яті А.М.Петуха

9-10 грудня 2019 р.

Суми/Вінниця
НІКО/ВНТУ
2019

УДК 004
ББК 32.97
Е50

Рекомендовано до видання Вченою радою Вінницького національного технічного університету Міністерства освіти і науки України (протокол № 9 від 25.11.2019 р.)

Електронні інформаційні ресурси: створення, використання, доступ: Збірник матеріалів Міжнародної науково-практичної Інтернет конференції. Пам'яті А.М.Петуха. – Суми/Вінниця : НІКО/ВНТУ, 2019. – 306 с.

ISBN 978-617-7422-11-1

Збірник містить матеріали Міжнародної науково-практичної Інтернет-конференції «Електронні інформаційні ресурси: створення, використання, доступ».

Матеріали збірника подано у авторській редакції. Автори опублікованих матеріалів несуть повну відповідальність за підбір, точність наведених фактів, цитат, статистичних даних, власних імен та інших відомостей, Матеріали відтворюються зі збереженням змісту, орфографії та синтаксису текстів, наданих авторами.

УДК 004

ISBN 978-617-7422-11-1

Перестало битися серце відомого вінницького науковця Анатолія Пєтуха, професора ВНТУ. У Вінницькому національному технічному університеті Анатолій Михайлович пропрацював майже 45 років.

Анатолій Михайлович народився в 1944 році. У 1965-му закінчив Львівський політехнічний інститут, де також навчався в аспірантурі з 1967 по 1970 роки. В 1972 році захистив кандидатську дисертацію на тему "Аналіз та розробка пристроїв лічильно-імпульсного вимірювання частот в слідкуючому режимі" (м. Львів).

Ступінь доктора технічних наук отримав у 1994 році в ВДТУ. Дисертацію захистив по темі: "Дослідження дискретно-фазових імпульсних потоків в інформаційно-вимірювальних системах".

Він є автором наукових праць у галузях:

- дослідження дискретно-фазових імпульсних послідовностей;
- формування та перетворення зображень;
- нові форми подання сигналів та величин;
- людино – машинна взаємодія;
- нові технології навчання на принципах колективної взаємодії.

А. Пєтух більше 25 років очолював кафедру програмного забезпечення ВНТУ, був членом Ученої ради ВНТУ, членом Учених рад ВНТУ по захисту кандидатських та докторських дисертацій, членом підкомісії з напряму програмна інженерія науково-методичної комісії МОН України.

Мав 20 науково-дослідницьких розробок. В 1971 та 1984 роках нагороджений срібними медалями ВДНГ СРСР. Неодноразово нагороджувався на міжнародних виставках винаходів:

- "Наука та техніка СРСР на службі миру та прогресу", Бомбей, 1988р.
- EAST-WEST EVRO INTELLECT" , Софія, 1996р. – золоту медаль.
- "EURECA", Брюссель, 1996р. – золоту медаль.
- "INPEX", Пітсбург, 1997р. – бронзову медаль за експонат "Мистецтво подання величин".

За останні роки, можна виокремити науково-дослідну роботу «Національна освітня інфраструктура удосконалення інноваційної та підприємницької діяльності ІТ-студентів» в рамках міжнародного проекту Tempus. Завдяки цьому проекту, кафедра отримала доступ до найсучасніших європейських технологій та програм навчання студентів. Багато кращих студентів отримали можливості стажування в провідних європейських університетах.

ЕЛЕКТРОННІ ІНФОРМАЦІЙНІ РЕСУРСИ: СТВОРЕННЯ, ВИКОРИСТАННЯ, ДОСТУП:
Збірник матеріалів
Міжнародної науково-практичної Інтернет-конференції.
Пам'яті А.М.Петуха

Редактор Н.А. Ніколаєнко
Комп'ютерне верстання М.С. Ніколаєнко

Підписано до друку 26.11.2019 Гарнітура Times New Roman
Формат 60x84/16 Папір офсетний
Друк цифровий Ум. друк. арк. 17,8
Тираж 300 пр. Зам. № 9/19

Видавництво НІКО
м.Суми, вул.Харківська, 54
Свідоцтво про внесення до Державного реєстру
суб'єктів видавничої справи України
серія СМв № 044
від 15.10.2012
E-mail: ms.niko@i.ua
Телефон для замовлень: +38(066) 270-64-68