Національний технічний університет України
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Department of Radio Engineering Systems

[RE-320] SOFTWARE INFRASTRUCTURE OF DISTRIBUTED WEB APPLICATIONS

Curriculum of the academic discipline (Syllabus)

Course details

Level of higher education	First (bachelor's)
Field of knowledge	G - Engineering, manufacturing, and construction
Specialty	G5 - Electronics, electronic communications, instrument engineering, and radio engineering
Educational program	All
Discipline status	Elective (F-catalog)
Form of higher education	Full-time
Year of training, semester	Available for selection starting from the 3rd year, spring semester
Scope of the discipline	4 credits (Lectures 16 hours, Practical classes 30 hours, Laboratory work 30 hours, Independent work 74 hours)
Semester Control/control measures	Credit
Class schedule	https://schedule.kpi.ua
Language of instruction	Ukrainian
Information about the course coordinator/teacher s	Lecturer: P. Yu. Katyn Lab: Katin P. Yu.,
Course placement	

Curriculum

1. Description of the course, its purpose, subject matter, and learning outcomes

The main objective of the discipline "Software Infrastructure of Distributed Web Applications" is to provide systematic knowledge, skills, and abilities in planning, developing, testing, and building software infrastructure based on:

- complex, distributed web application infrastructure based on ASP.NET Core MVC using Docker technology;

- complex, distributed web application infrastructure based on Python Django using Docker technology.

In addition, the discipline provides information on the general principles of distributed web application development and software infrastructure deployment based on container technology systems.

The syllabus for the course "Software Infrastructure" includes educational tasks and programming exercises. This allows students to gain practical experience in developing distributed web applications and deploying infrastructure for them based on container technologies. Students are expected to acquire knowledge in this field independently.

There is a standard approach to teaching programming, where in the early stages (programming exercises) a relatively complex task is performed, which makes it possible to interest students in further study. The final task is a distributed software infrastructure for a web application based on Docker technology, which simulates the operation of a professional web portal. To complete this task, students use the theoretical knowledge and practical skills they have acquired throughout the course.

The aim of the course is to train highly qualified software developers who have a basic understanding of software infrastructure based on modern technologies.

Subject of the course: the process of designing, developing, testing, and building a distributed infrastructure for a web application using a container system.

The study of the discipline contributes to the formation of professional competencies (PC) in students, which are necessary for solving practical problems in professional activities related to the development, improvement, and maintenance of intelligent information systems for processing multimedia data:

Ability to identify, classify, and formulate software requirements;

Ability to participate in software design, including modeling (formal description) of its structure, behavior, and

operating processes;

Ability to develop architectures, modules, and components of software systems;

Ability to apply fundamental and interdisciplinary knowledge to successfully solve software

engineering problems;

Ability to accumulate, process, and systematize professional knowledge regarding the creation and maintenance of software and recognition of the importance of

lifelong learning;

Ability to implement phases and iterations of the life cycle of software systems and information technologies based on appropriate software development models and approaches;

Ability to carry out the system integration process, apply standards and change management procedures to maintain the integrity, overall functionality, and reliability of software; The ability to make informed choices and master tools for software development and maintenance.
Studying this discipline contributes to the formation of the following program learning outcomes (PLOs) for students in the educational program:

Analyze, purposefully search for and select information and reference resources and

knowledge necessary for solving professional tasks, taking into account modern achievements

in science and technology;

SLO03 Know the main processes, phases, and iterations of the software life cycle;

Know and apply professional standards and other regulatory and legal documents in the field of software *engineering;*

Know and apply relevant mathematical concepts, methods of domain, system, and object- oriented

analysis, and mathematical modeling for software development;

Know and apply in practice the fundamental concepts, paradigms, and basic principles of the functioning of linguistic, instrumental, and computational means software engineering;

Know and be able to develop human-machine interfaces;

Be able to use methods and tools for collecting, formulating, and analyzing software requirements; Apply effective

approaches to software design in practice;

Know and apply methods for developing algorithms, designing software, and constructing data and

knowledge structures;

Be motivated to choose programming languages and development technologies to solve software creation and maintenance tasks;

Have software development skills, coordinate the design and release of all types of software documentation;

Be able to calculate the economic efficiency of software systems.

2. Prerequisites and post-requisites of the discipline (place in the structural-logical scheme of training under the relevant educational program)

Successful study of the discipline is preceded by the study of the disciplines "Informatics" and "OOP".

The theoretical knowledge and practical skills acquired during the study of the discipline ensure the successful completion of course projects and bachelor's theses.

3. Contents of the discipline

Topic 1. Technologies and architecture of web applications based on typical frameworks.
Topic 2. Software management of databases in web applications.
Topic 3. Containerization system as the basis for creating a complex distributed software infrastructure for web applications.
Topic 4. Automation of software infrastructure management in a containerization system.
Topic 5. Microservice architecture as the basis for creating the software infrastructure of distributed web applications.

Modular test. Credit.

4. Teaching materials and resources

1. *Information Systems Design: General Issues of IS Design Theory (lecture notes) [Electronic resource]: textbook for students majoring i n 122 "Computer Science" / Igor Sikorsky KPI; compiled by: O. S. Kovalenko, L. M. Dobrovska. – Electronic text data (1 file: 2.02 MB). – Kyiv: Igor Sikorsky Kyiv Polytechnic Institute, 2020. – 192 p. https://ela.kpi.ua/bitstream/123456789/33651/1/PIS_KL.pdf*

2. Software infrastructure for WEB applications. Laboratory workshop .\ ELAKPI: Software infrastructure for WEB applications. Laboratory workshop

Additional reading:

1. *Django documentation [Electronic resource] – https://docs.djangoproject.com/en/3.2/.*

2. *Adam Freeman. Essential Docker for ASP.NET Core MVC. ISBN-13 (pbk): 978-1-4842-2777-0 ISBN-13 (electronic): 978-1-4842-2778-7. 2017.*

3. *Scott Chacon, Ben Straub. Pro Git. Version 2.1.95-2-g8d45587, 19.01.2022.*

4. *Esteban Zimányi. Students: Bubacarr Jallow Shafagh Kashef. Object Relational Mapping and Entity Framework. Advanced Databases Project. 12/18/2018.*

5. *Rebeka Mukherjee. Python Scripting for System Administration. Department of Computer Science and Engineering Netaji Subhash Engineering College, Kolkata.*

6. *https://docs.docker.com/samples/django/*

7. *Literature on ASP.NET Core technologies and software infrastructure*

8. *Adam Freeman. Pro ASP.NET Core 3: Develop Cloud-Ready Web Applications Using MVC, Blazor, and Razor Pages. 2020.*

9. *Docker documentation [Electronic resource] –*
https://docs.docker.com/get-started/.

10. *Matthes E. Python Crash Course (2nd Edition) : A Hands-On, Project-Based Introduction to Programming / Eric Matthes. – San Francisco, United States: No Starch Press, US, 9. – 544 p. – (2nd Edition).*

11. *Thomas D. The Pragmatic Programmer: your journey to mastery, 20th Anniversary Edition / D. Thomas, A. Hunt. – Boston, United States: Pearson Education (US), 2020. – 352 p.*

12. *Learn Python the Hard Way: A Very Simple Introduction to the Terrifyingly Beautiful World of Computers and Code – New Jersey, United States: Pearson Education (US), 2013. – 320 p.*

13. *Learning React: Modern Patterns for Developing React Apps – Sebastopol, United States: O'Reilly Media, Inc, USA, 2020. – 300 p.*

14. *Docker: Complete Guide To Docker For Beginners And Intermediates, 2020. – 140 p.*

15. *Docker: Up & Running: Shipping Reliable Containers in Production – Sebastopol, United States: O'Reilly Media, Inc, USA, 2018. – 347 p. Docker homepage - http://www.docker.com/*

16. *Docker Hub - https://hub.docker.com*

17. *Docker blog - http://blog.docker.com/*

18. *Docker documentation - http://docs.docker.com/*

19. *Docker Getting Started Guide - http://www.docker.com/gettingstarted/*

20. *Docker code on GitHub - https://github.com/docker/docker*

21. *Docker mailing list - https://groups.google.com/forum/#!forum/docker#user*

22. *Docker on IRC: irc.freenode.net and channels #docker and #docker#dev*

23. *Docker on Twitter - http://twitter.com/docker*

24. *Get Docker help on Stack Overflow - http://stackoverflow.com/*
search?q=docker

25. Valeria Cardellini. Matteo Nardelli. Container-based virtualization: Docker. Università degli Studi di Roma "Tor Vergata" Dipartimento di Ingegneria Civile e Ingegneria Informatica Corso di Sistemi Distribuiti e Cloud Computing A.A. 2017/18.

26. Adam Freeman. Pro Angular 6 .ISBN-13 (pbk): 978-1-4842-3648-2 ISBN-13 (electronic): 978-1-4842-3649-9/2018

Educational content

5. Methodology for mastering the academic discipline (educational component)

No	Type of educational activity	Description of the educational session
Topic 1. Technologies and architecture of web applications based on typical frameworks.		
1	Lecture 1. Technology for creating web applications based on typical frameworks (Python Django, .NET Core).	Fundamentals of distributed web application infrastructure. Hardware component of infrastructure. Containerization systems for deploying complex distributed web applications. Technology for creating web applications based on typical frameworks (.NET Core). Assignment for independent study: p. 6 No. 1.
2	Lecture 2. Software architecture of a web application based on typical Python Django templates (patterns).	Web application architecture based on the MVT framework pattern. Creating a web application within a project. Project testing. JavaScript technologies as an element of web application architecture. Assignment for independent study: p.6 No. 2.
	Practical work No. 1.1	Development and testing of a web application based on a typical framework (2 hours).
3	Lecture 3. Software architecture of a web application based on .NET Core.	Architecture of a web application based on the MVC framework pattern (.NET Core). Creating a web application based on the framework (.NET Core). Basic components of .NET Core. JavaScript technologies as an element of the .NET Core web application architecture. Assignment for independent study: p.6 No. 3.
4	Lecture 4. Deployment of a web application on the network as a complex software infrastructure.	List and technologies for deploying web applications on the network. Practice of deploying a web application. Using a Docker container to deploy a web application. Basics of automation and scaling of complex software infrastructure. Assignment for independent study: p.6 No. 4.
	Practical work No. 1.2	Development and testing of a web application based on a typical framework (2 hours).
Topic 2. Software management of databases in web applications.		
	Lecture 5. Software architecture of a typical object-relational mapping (ORM) for working with the database of typical distributed web applications.	Object-relational mapping (ORM) in a web application based on the MVT framework pattern (Python Django). Database management in the Django framework (mysite/settings.py). Creating and activating a database model in the Django framework. Database management via Django API. Creating, configuring, and testing the admin panel. Testing the ORM database model through the admin panel. JavaScript technologies for working with databases. Using a Docker container to create a software infrastructure that separates the database and web application.
5	Lecture 6. Entity Framework software architecture for working with databases in typical distributed web applications.	Alternative technologies for working with databases in a web application based on the MVT pattern framework (Python Django). Classification of technologies for working with databases in .NET Core. Object-relational mapping (ORM) in web applications based on the MVC framework pattern (.NET Core). Entity Framework in .NET technology. JavaScript technologies for working with databases Assignments for independent study: p.6 No. 6, 32.
	Practical work No. 2.	Deploying a web application on the network using free hosting (2 hours).
Topic 3. Containerization system as the basis for creating a complex distributed software infrastructure for web applications.		
6	Lecture 7. General configuration of a prototype of a typical distributed web application for running in a container.	Installation of basic software components of the infrastructure for working with the container system in Django. Display system based on the MVT framework pattern (Python Django). Request management system based on the MVT framework (mysite.urls) Installation of basic software components of the infrastructure for working with the container system in .NET Core. Node.js, NPM Package, Git, Docker, .NET Core Software Development Kit. Display system based on the MVC framework pattern (.NET Core). Request management system based on the MVC framework (.NET Core). Assignment for independent study: p.6 No. 8.
7	Lecture 8. Testing a prototype of a typical distributed web application for running in a container.	Testing basic software components of the infrastructure for working with the container system in Django. Display system based on the MVT framework pattern (Python Django). Request management system based on the MVT framework (mysite.urls) Testing basic software components of the infrastructure based on the MVC framework pattern (.NET Core). Installation of basic software components of the infrastructure for working with the container system in .NET Core. Node.js, NPM Package, Git, Docker, .NET Core Software Development Kit. Display system based on the MVC framework pattern (.NET Core). Request management system based on the MVC framework (.NET Core). Assignment for independent work: p.6 No. 9.

	Practical work No. 3.1	*Designing a Docker container for deploying and developing a web application (2 hours).*
Topic 4. Automation of software infrastructure management in a containerization system		
8	*Lecture 9. Fundamentals of distributed software infrastructure based on containerization systems.*	*General architecture of a containerization system based on Docker. Managing Docker images. Container states. Basic commands and examples of using Docker based on a typical distributed web application. Assignment for independent study: p.6 No. 11.*
9	*Lecture 10. Automation of management using Dockerfile.*	*Basics of working with Dockerfile. Syntax and rules for creating Dockerfile. Practical use of Dockerfile for complex software infrastructure. Assignment for independent study: p.6 No. 12.*
	Practical work No. 3.2	*Designing a Docker container for deploying and developing a web application (2 hours).*
10	*Lecture 11. Basic elements of the Docker network.*	*General software infrastructure of distributed web applications in the Docker network. Docker network management. Assignment for independent study: p.6 No. 13*
11	*Lecture 12. Automation of complex software infrastructure management based on Docker Compose and Swarm mode technology.*	*Installing the Docker Compose system Starting and stopping the Docker Compose system. Basic syntax of the Docker Compose file. Automating management using Swarm mode. Scaling complex software infrastructure. Testing complex software infrastructure. Managing services in complex software infrastructure. Load balancing in complex software infrastructure. Manage Services commands. Assignment for independent study: p.6 No. 14.*
	Practical work No. 4.1	*Decomposition of complex software architecture of a web application into separate services and development of microservice architecture (2 hours).*
Topic 5. Microservice architecture as the basis for creating the software infrastructure of distributed web applications		
12	*Lecture 13. Fundamentals of microservice architecture.*	*General description of microservice architecture. Advantages of microservice architecture. Disadvantages of microservice architecture. Assignment for independent study: p.6 No. 15.*
13	*Lecture 14. Software infrastructure based on microservices.*	*Horizontal scaling of software infrastructure based on microservices (X-Axis Scaling). Vertical scaling of software infrastructure based on microservices (Y-Axis Scaling). Scaling software infrastructure based on microservices (Z-Axis Scaling). Assignment for independent study: p.6 No. 16.*
14	*Lecture 15. Typical microservice architecture patterns in complex distributed infrastructure.*	*Aggregator Pattern. Proxy Pattern. Chained Pattern. Branch Microservice Pattern. Shared Resource Pattern. Assignment for independent study: p.6 No. 17.*
	Practical work No. 4.2	*Decomposition of complex software architecture of a web application into separate services and development of microservice architecture (2 hours).*
16	*Lecture 16. Lecture-seminar. Reports on distributed infrastructure.*	
	Practical work No. 5.	*Creating a software infrastructure based on Docker containers for deploying and managing complex software architecture (4 hours).*
17	*Lecture 17. Lecture-seminar. Reports on distributed infrastructure management.*	
Test		

6. Independent work

The discipline is based on independent preparation for classroom sessions on theoretical and practical topics.

No	*Title of the topic for independent study*	*Number of hours*	*Literature*
1	*Preparation for lecture 1*	*1*	*1;1-7*
2	*Preparation for computer workshop 1*	*1.5*	*1;1-27*
3	*Preparation for lecture 2*	*1*	*1;1-7*
4	*Preparation for Lecture 3*	*1*	*1;8-9*
5	*Preparation for computer workshop 1 (part 2)*	*1*	*1;1-27*
6	*Preparation for lecture 4*	*1*	*1;1-27*
7	*Preparation for Lecture 5*	*1*	*1;1-27*
8	*Preparation for computer workshop 2*	*1.5*	*1;1-27*
9	*Preparation for lecture 6*	*1*	*1;1-27*
10	*Preparation for lecture 7*	*1*	*1;1-27*
11	*Preparation for computer workshop 3*	*1.5*	*1;1-27*
12	*Preparation for Lecture 8*	*1*	*1;1-27*
13	*Preparation for Lecture 9*	*1*	*1;1-27*
14	*Preparation for computer workshop 4 (part 1)*	*1.5*	*1;1-27*
15	*Preparation for lecture 10*	*1*	*1;1-27*
16	*Preparation for lecture 11*	*1*	*1;1-27*
17	*Preparation for computer workshop 4 (part 2)*	*1.5*	*1;1-27*
1	*Preparation for lecture 12*	*1*	*1;1-27*
19	*Preparation for lecture 13*	*1*	*1;1-27*

20	Preparation for computer workshop 5 (part 1)	1	1;1-27
21	Preparation for lecture 14	1	1;1-27
22	Preparation for lecture 15	1	1;1-27
23	Preparation for computer workshop 5 (part 2)	1.5	1;1-27
24	Preparation for lecture 16 (lecture-seminar)	1	1, 1-27
25	Preparation for lecture 17 (lecture-seminar)	1	1, 1-27
26	Preparation for the module test (part 1)	1	1, 1-27
2	Preparation for modular test (part 1)	4	1;1-27
2	Preparation for the exam	6	1, 1-27
29	List of frameworks for creating web applications based on different programming languages.	6	1, 1-26
30	Distributed version control systems.	4	3
31	Technologies and practices for working with HTML5 and related technologies.	2.5	27
32	Fundamentals of computer network organization.	13	27

Policy and control

7. Academic discipline policy (educational component)

Attendance at lectures is mandatory. In exceptional circumstances, attendance at lectures and other aspects of policy may be subject to change.

Attendance at computer lab classes may be sporadic and, if necessary, for consultation/defense of computer lab work.

Rules of conduct in class: be active, respect others, turn off phones. Follow the academic integrity policy.

Rules for defending computer lab work: work must be done in accordance with the tasks set and according to the option.

The rules for awarding incentive and penalty points are as follows. Incentive points are awarded for:

- accurate and complete answers in surveys based on lecture materials (maximum number of points for a survey is 3 points).

8. Types of control and rating system for assessing learning outcomes

During the semester, students complete 5 computer workshops. The maximum number of points for each computer workshop is 10 points.

Points are awarded for:

- quality of computer workshop performance: 0-5 points;

- answers during the defense of the computer workshop: 0-3 points;

- timely submission of work for defense: 0-2 points. Criteria for assessing the quality of performance:

5 points – the work is done well and in full;

4 points – the work is done well, in full, but has some flaws;

3 points – the work is completed in full, but contains minor errors; 2 points – the work is completed in full, but contains significant errors; 0 points – the work is not completed in full.

Criteria for evaluating answers:

3 points – the answer is complete and well-reasoned;

2 points – the answer is correct but has shortcomings or minor

errors; 1 point – the answer contains significant errors;

0 points – no answer or incorrect answer.

Criteria for assessing the timeliness of submitting work for defense:

2 points – the work is submitted for defense no later than the specified

deadline; 0 points – the work is submitted for defense later than the

specified deadline.

Maximum number of points for completing and defending computer practicals: 10 points

× 5 computer practicals = 50 points.

During the semester, **quizzes on the topic of the current lesson** *are held during lectures. Maximum number of points for all quizzes: 3 points. The number* **of quizzes on the topic of the current lesson** *for one student is unlimited.*

The module test *consists of 3 theoretical and 2 practical questions. Each question is worth 10*

points. The criteria for evaluating each question on the test are as follows:

9-10 points – the answer is correct, complete, and well-reasoned;

7-8 points – the answer is correct, detailed, but not very well argued; 5-6 points

– the answer is generally correct, but has shortcomings;

3-4 points – the answer contains

minor errors; 1-2 points – the

answer contains significant errors;

0 points – no answer or incorrect answer.

Maximum number of points for the

module test:

10 points × 5 questions = 50 points. The

rating scale for the discipline is:

$R = R_C = 50$ *points* $+ 50$ *points* $= 100$ *points.*

Calendar control: conducted twice per semester as monitoring of the current status of syllabus requirements.

At the first assessment (8th week), the student receives a "pass" if their current rating is not less than 15 points (50% of the maximum number of points that a student can receive before the first assessment).

At the second assessment (14th week), the student receives a "pass" if their current rating is at least 20 points (50% of the maximum number of points that a student can receive before the second assessment).
Semester assessment: credit

Conditions for admission to semester control:

With a semester rating (R_C) of at least 60 points and all computer workshop assignments completed, the student automatically receives a credit according to the table (Table of correspondence between rating points and grades on the university scale). Otherwise, they must complete a credit test.

A prerequisite for admission to the credit test is the completion and defense of the computer workshop.

If a student does not agree with the automatic grade, they can try to improve their grade by writing a credit test, in which case their points earned during the semester are retained, and the better of the two grades received by the student is awarded (a "soft" grading system).

Table of correspondence between rating points and grades on the university scale

Number of points	**Grade**
100-95	Excellent
94-85	Very good
84-75	Good
74-65	Satisfactory
64-60	Sufficient
Less than 60	Unsatisfactory
Admission requirements not met	Not admitted

9. Additional information on the discipline (educational component)

Various operating systems may be used for work at the student's discretion and in agreement with the instructor.
If students wish to use other programming languages or technologies in the computer lab, they can be included as an additional element in the basic architecture of the distributed web application.

Description of material, technical, and information support for the discipline

Computer classroom of the Department of Radio Engineering Systems

---

Work program of the academic discipline (syllabus):
Compiled by Katin P. Yu.;
Approved by the Department of Radio Engineering Systems (Minutes No. 06/2025 dated 06/24/2025)
Approved by the methodological commission of the faculty/research institute (protocol No. 06/2025 dated 26.06.2025)