Syllabus for the Course «Infrastructure and Architecture of Information Systems»

Course volume: 3 ECTS credits (108 hours)
Semester: 1
Certification form: credit with grade
Class workload:

- Lectures - 32 hours
- Seminars/Digital Labs - 32 hours

Independent work - 44 hours.
Implementation format: AI- augmented learning (learning using intelligent assistants).

1. General characteristics of the discipline

The course "Infrastructure and Architecture of Information Systems" is part of the professional module of the master's degree program "AI- based IT Engineering" in the field of 09.04.02 "Information Systems and Technologies".
The course is implemented in the first semester and opens the educational track “Information Systems Architecture”.
The course is of an overview and methodological nature and forms in students a systematic understanding of the relationship between the architecture of software systems and the infrastructure for their implementation.
This course does not aim to provide a detailed understanding of individual infrastructure technologies (cloud platforms, containerization, storage systems, etc.). These technologies are covered at the level of architectural principles and practical applicability, forming the basis for their more in-depth study in subsequent disciplines of the track.
A distinctive feature of the course is the use of large language models (LLM) as a tool for analyzing architectural decisions, identifying engineering trade-offs, and supporting student independent work.

2. The place of the discipline in the structure of the educational program

This course is the initial course in the architectural track of the program and serves as an introduction to the infrastructural dimension of digital systems.
It forms a basic understanding:

- the role of infrastructure in the architecture of information systems;
- the relationship between software architecture and runtime environment;
- architectural limitations of modern computing platforms;
- principles of scalability, fault tolerance and operation of digital systems.

The course is closely related to the disciplines of the first semester:

- Algorithms and data structures

01.01 RPD Algorithms and structures…

- Algorithmization and programming languages
- AI Fundamentals and LLM: Prompt and Context Engineering

05.01 RPD_AI_LLM_Prompt_Context ...
and creates a methodological basis for subsequent disciplines in the architectural and infrastructure profile:

- containerization and Kubernetes ;

- cloud computing;
- distributed systems;
- operational engineering;
- architecture of high-load systems.

Thus, the discipline provides a transition from software thinking to architectural thinking of digital systems.

3. Objectives of mastering the discipline

The aim of the course is to develop in students a systemic understanding of the infrastructural foundations of information systems architecture.

Particular attention is paid to:

- understanding infrastructure as an architectural level of digital systems;
- analysis of the relationship between architectural solutions and the execution environment;
- mastering the principles of scalability and fault tolerance of systems;
- developing the ability to analyze alternative infrastructure solutions;
- developing architectural analysis skills using LLM.

4. Objectives of the discipline

The main objectives of the discipline are:

- formation of a systemic understanding of the structure of modern digital infrastructure;
- studying the role of computing platforms, networks and storage systems in system architecture;
- mastering the principles of virtualization and containerization as infrastructure mechanisms;
- studying cloud computing architecture;
- analysis of architectural solutions to ensure scalability and fault tolerance;
- study of the principles of monitoring, observability and operation of digital systems;
- development of skills for comparative analysis of infrastructure solutions;
- development of a culture of using LLM in engineering analysis.

5. Planned learning outcomes

As a result of mastering the discipline, the student must:

Know:

- architectural structure of modern information systems;
- the role of infrastructure in shaping the architecture of digital systems;
- main types of computing infrastructures;
- principles of virtualization and containerization;
- architectural models of cloud platforms;
- principles of scalability and fault tolerance;
- fundamentals of infrastructure monitoring and observability;
- limitations and risks of using LLM in the analysis of engineering solutions.

Be able to:

- analyze the architecture of an information system from an infrastructure perspective;
- identify infrastructural limitations of architectural solutions;
- compare alternative infrastructure approaches;
- to provide a reasoned justification for architectural decisions;
- use LLM for architectural analysis and identification of alternative solutions.

Be proficient in:
- methods of architectural analysis of digital systems;
- skills in comparative analysis of infrastructure solutions;
- culture of engineering argumentation;
- practices of conscious use of intelligent assistants.

6. Methodological concept of the discipline

6.1 Architecture of digital systems as a multi-level structure

A modern information system is a multi-level engineering structure that includes:
- software architecture,
- infrastructure platform,
- operating environment.

The system architecture is formed at the intersection of system requirements and execution environment constraints.

Therefore, infrastructure is considered not as an auxiliary technical subsystem, but as a full-fledged architectural level of the digital system.

6.2 Methodological principle of the course

The methodological of the discipline is expressed by the principle:

"The runtime environment forms the architecture of the system."

Architectural decisions for an information system are always made within the context of infrastructural constraints.

Such restrictions include:
- computing resources,
- network characteristics,
- data storage architecture,
- security requirements,
- operational limitations.

System architectural design therefore involves managing trade-offs between system requirements and infrastructure capabilities.

6.3 Architectural thinking through analysis of alternatives

Within the discipline, architectural solutions are viewed not as a set of technologies, but as a space of alternatives.

Students analyze contrasts:
- centralized and distributed systems;
- local infrastructure and cloud platforms;
- virtual machines and container environments;
- various types of data storage architectures.

The goal of the course is to develop the ability to select architectural solutions depending on the conditions and constraints of the execution environment.

6.4 Infrastructure and life cycle of a digital system

Infrastructure decisions have a significant impact on the system life cycle.
The choice of infrastructure depends on:

- system scalability;
- failure tolerance;
- complexity of operation;
- cost of support.

Thus, the infrastructure determines not only the initial architecture of the system, but also the possibilities for its further development.

7. Using LLM in the educational process

Within the discipline, large language models are used as a tool to support engineering analysis.
LLM are used in the following modes.

LLM as a reference tool

The model is used to explain architectural terms, technologies and infrastructure mechanisms.

LLM as an intelligent assistant

The LLM helps students analyze architectural solutions, identify alternative approaches, and formulate arguments for choosing solutions.

LLM as an analytical opponent

LLM workshops are used to identify weaknesses in architectural solutions, generate alternative infrastructure approaches, and ask clarifying questions.

LLM as a feedback tool

When completing independent work, the model can be used for an initial analysis of architectural notes, identifying gaps in the analysis, and preparing questions for discussion with the instructor. The final grade is determined by the instructor.

Critical Position: Domain-Specific Risks of LLM in Architectural Analysis

The use of LLM in architectural analysis requires particular caution, as the model systematically reproduces a number of characteristic errors that are externally indistinguishable from correct engineering reasoning.

Typical LLM mistakes in the context of this discipline:

1. **Recommendations without load context.** The model tends to provide firm advice on choosing an infrastructure solution (e.g., "use a microservices architecture" or "move to the cloud") without eliciting or considering load characteristics such as request volume, peak patterns , or latency requirements. A correct architectural decision is always tied to specific load parameters.
2. **Reproduction of marketing descriptions.** When comparing cloud platforms or infrastructure approaches, the model often reproduces the providers' official descriptions instead of an engineering analysis of the tradeoffs. Responses may sound like "AWS is highly reliable and flexible"—which is a marketing claim, not an architectural argument.

3. **Incompatible "alternatives."** When requesting alternative architectural solutions, the model may suggest options that formally correspond to the request topic but are incompatible with the constraints stated in the task (budget, team, security requirements, regulatory restrictions).
4. **False symmetry of tradeoffs.** When analyzing oppositions (centralized/distributed, on-premises/cloud, etc.), the model tends to present both options as equally valuable, avoiding a definitive conclusion. This creates the illusion of a balanced analysis where the engineering situation requires a specific, justified choice.
5. **Outdated technical data.** The infrastructure landscape is changing rapidly. Specific platform characteristics, pricing models, and service limitations may no longer be relevant in the model's responses. Any specific technical parameters require verification against current documentation.

Mandatory elements of LLM results verification:

- Check: Does the model recommendation specify the specific load characteristic it is based on?
- Check: is the argument given engineering (measurable) or marketing (evaluative)?
- Check: Is the proposed solution compatible with all stated constraints of the problem?
- Check: does the analysis contain a clear conclusion with justification, or is the model limited to a list of “pros and cons”?
- Record in the analytical note: which model statements were accepted, which were rejected and why.

The last element is a mandatory artifact of independent work. Failure to reflect on the LLM results is tantamount to incomplete completion of the assignment.

Key principle: The language model operates through a form of architectural reasoning. The engineering content—the solution's compliance with the real constraints of the execution environment—is provided by the student.

8. Educational technologies

The discipline uses an AI-augmented learning model, which combines traditional forms of learning with the use of intelligent assistants.

The educational process includes:

- lecture classes;
- seminar classes;
- digital laboratory work;
- analytical tasks;
- use LLM as an analytical tool.

9. Organization of independent work of students

Independent work is aimed at developing architectural analysis skills.

Main forms of SDS (Self-Directed Study):

- analysis of digital systems architecture;
- comparative analysis of infrastructure solutions;

- preparation of analytical reports;
- architectural essays;
- working with LLM assistants.

A distinctive feature of the discipline is the use of LLM for:

- analysis of the structure of argumentation;
- identifying missed architectural factors;
- generation of alternative solutions.

10. Forms of control

The following forms of control are used within the framework of the discipline:
current assessment:

- discussion of architectural cases;
- analysis of infrastructure solutions;
- assessment of students' analytical work.

final assessment:
credit with a grade, including:

- architectural case analysis;
- comparative analysis of infrastructure solutions;
- reasoned justification of the architectural choice.

LLM can be used for:

- generation of case variants;
- primary analysis of written work;
- preparing analytical reports for the teacher.

Curriculum schedule for the course

Key:

- **Lecture (Lect.)** — theoretical material delivery.
- **Practical / Lab.** — hands-on activities, simulations and discussions.
- **Self-directed work (SDS)** — independent study on project tasks and analysis.
- **LLM use** — applying Large Language Models for analysis, generation and decision support (integrated into independent study tasks).

Week	Content	Lect. (h)	Practice (h)	SDS (h)
1	**Information System as an Engineering Construction** IS as a multi-layered construction: software architecture, infrastructure platform, operational environment. **Key question:** Why is infrastructure an architectural level, not a supporting subsystem? **Case study:** Reconstruct the three-tier architectural structure of a real digital platform.	2	2	2

Week	Content	Lect. (h)	Practice (h)	SDS (h)
	Discussion: Where does software architecture end and infrastructure begin? **LLM use:** - **Reference:** Clarify terms. - **Controller:** Check completeness of architectural description in the note — indicate which levels are not described.			
2	**Architectural Constraints and Engineering Trade-Offs** Performance, scalability, reliability, cost as interrelated constraints. CAP theorem as an example of architectural trade-off. **Key question:** Why is there no «optimal architecture» without specifying context? **«Trade-off in action»:** Two architecture variants with different load profiles — argue choice for three scenarios. **Requirement:** Each argument refers to a specific constraint, not a general statement. **LLM use:** - **Alternative generator:** Ask to propose a third architecture variant. - **Verification:** Check if load parameters of the task are taken into account. Record what is correct and what is without context.	2	2	2
3	**Evolution of Computing Infrastructure** Trajectory: mainframes → client-server → distributed systems → cloud platforms. Each transition is a response to a new infrastructural constraint. **Key question:** What made the previous paradigm insufficient? **Historical case:** Analyse a specific technological transition — reconstruct the constraints that made it inevitable and the new constraints it created. **LLM use:** - **Analytical assistant:** Ask «Why didn't the transition to the cloud solve problem [X]?». - **Verification:** Is it an engineering argument or a general description?	2	2	2
4	**Infrastructure as an Architectural Level** Principle: «Execution environment shapes the system architecture». Infrastructural constraints (computing, network, storage, security, operation) define the space of acceptable architectural solutions. Infrastructure impact on system lifecycle. **«From requirements to constraints»:** A set of functional requirements + infrastructural parameters of the environment. **Task:** Identify feasible and infeasible requirements — and why. **LLM use:** - **Controller:** Check the note for completeness — are all constraints taken into account? - **Verification:** Does the model suggest «removing» a constraint instead of working with it?	2	2	2
5	**Modern Computing Platforms and Data Centers** Data center architecture: computing cluster, network fabric,	2	2	2

Week	Content	Lect. (h)	Practice (h)	SDS (h)
	storage, power. Platform types: bare metal, virtualised servers, GPU/TPU. **Key question:** How does the physical organisation of a data center affect the architecture of hosted systems? **Comparative analysis:** Own DC vs. colocation vs. cloud. Analyse architectural consequences of each choice, not cost. **LLM use:** - **Technology reference:** Clarify platform characteristics. - **Verification:** Are these engineering parameters or marketing descriptions? Record examples of both types of answers.			
6	**Virtualisation and Containerisation as Infrastructural Mechanisms** Virtualisation: hypervisors types 1 and 2, resource isolation, overhead. Containerisation: namespace, cgroups, images. **Key question:** What exactly do these mechanisms isolate — and what do they not isolate? **Two deployment scenarios** (monolith and microservices): argue choice of isolation mechanism for each. **Requirement:** Argument relies on architectural characteristic, not technology popularity. **LLM use:** - **Analytical opponent:** Present justification and ask for three counterarguments. - **Verification:** Which counterarguments are correct, which miss the task context?	2	2	2
7	**OS as a Management Layer: From Computing to Storage** OS as a computing scheduler and I/O manager. File systems as abstraction over physical storage. **Key question:** Which architectural decisions of an application depend on OS behaviour? **«Data path»:** Trace one file read request in three environments (bare metal, VM, container). Record additional abstraction layers and their cost. **LLM use:** - **Reference:** Explain file system and scheduler mechanisms. - **Controller:** Check correctness of data path tracing in the note.	2	2	2
8	**Architectures of Data Storage Systems** Three architectural models: block, file, object storage. Characteristics: access semantics, scalability, consistency. **Key question:** How do data characteristics and access patterns determine model choice? **«Storage choice»:** Three systems with different data (transactional DB, media service, document archive). Argue model choice for each. Discuss edge cases. **LLM use:** - **Edge case:** «A system with requirements for both transactionality and scalability of object storage — how to design?». - **Verification:** Specific architectural solution or evasion of choice?	2	2	2

Week	Content	Lect. (h)	Practice (h)	SDS (h)
9	**Cloud Computing and Service Models** Architectural logic of cloud: division of responsibility between provider and consumer. Shared Responsibility model. **Key question:** What is «given to» the provider in each model — and what architectural consequences does this entail? **«Responsibility boundary»:** Analyse a public post-mortem of a cloud failure. Determine on which side of the boundary the problem occurred; how the client's architectural decision could mitigate the incident. **LLM use:** - **Architectural analysis:** Compare two cloud providers on a specific characteristic. - **Verification:** Engineering analysis or reproduction of provider documentation?	2	2	2
10	**Architectures of Cloud Infrastructures: Public, Private, Hybrid Clouds** Three deployment models: architectural and operational differences. Multi-cloud strategies. Vendor lock-in as an architectural risk. **Key question:** Which factors (technical, regulatory, organisational) determine model choice? Three organisations with different constraints (medicine, fintech, startup): argue optimal model for each. **LLM use:** - **Multi-cloud strategy generator:** Generate a multi-cloud strategy for a specific class of systems. - **Verification:** Are regulatory constraints taken into account or only technical arguments?	2	2	2
11	**Infrastructure Security as an Architectural Level** Security as an architectural quality, not a protection layer. Defence in Depth applied to infrastructure. **Key question:** Which decisions make a system secure «by default», not «with correct configuration»? **Architectural security audit:** Analyse attack surface of a simplified scheme. Identify architectural decisions that expand the attack surface; propose structural (not configurational) countermeasures. **LLM use:** - **Vulnerability identifier:** Present a scheme and ask to indicate security risks. - **Verification:** Distinguish architectural problems (structural) from configurational ones (solvable without architecture change).	2	2	4
12	**Access Management and Identity** Identification, authentication, authorisation as architectural components. Models: RBAC, ABAC, Zero-Trust. IAM as an infrastructural service. **Key question:** How does IAM architecture affect scalability and manageability of the entire system? **Designing IAM architecture** for a multi-component system with different trust levels between components. **Discussion:**	2	2	4

Week	Content	Lect. (h)	Practice (h)	SDS (h)
	When does centralised IAM become a bottleneck? **LLM use:** - **Question:** «How will IAM architecture change when moving from monolith to microservices?». - **Verification:** Are specific consequences taken into account (proliferation of service identities, distributed policy enforcement) or only general statements?			
13	**Scalability and Distributed Infrastructures** Vertical and horizontal scalability: architectural implications of each. Stateless vs. stateful — key architectural distinction for scalability. **Key question:** What makes a system scalable — and what makes scalability impossible without redesign? **Scalability diagnostics:** Identify «bottlenecks» in an architectural scheme — components that do not scale horizontally without architecture change. **LLM use:** - **Load scenario modeller:** Propose a scalability strategy. - **Verification:** Are stateful components of the system taken into account in the proposal?	2	2	4
14	**Reliability and Fault Tolera**nce of Infrastructure SLA, SLO, SLI as engineering tools. Architectural patterns of fault tolerance: replication, redundancy, circuit breaker, graceful degradation. Key question: What is the architectural «tax» for each additional «nine» of reliability? Post-mortem analysis: Reconstruct failure chain, identify architectural decisions that amplified or limited incident scope. Propose structural solutions. LLM use: - Risk analyser: Identify single points of failure (SPOF) in an architectural scheme. - Verification: Distinguish technical SPOFs from organisational ones — the latter the model tends to miss.	2	2	4
15	**Monitoring and Observability of Infrastructure** Three pillars of observability: metrics, logs, tracing. Monitoring vs. observability: architectural difference. **Key question:** Which architectural decisions make a system «observable by default»? **Designing an observability system:** For a given architectural scheme — minimal sufficient set of metrics, logs and tracing points for diagnosing typical failures. **LLM use:** - **Question:** «Which metrics are needed to detect DB degradation before affecting users?». - **Verification:** Concrete engineering answer or general description of monitoring systems?	2	2	4

Week	Content	Lect. (h)	Practice (h)	SDS (h)
16	**Evolution of Infrastructure and Development of Digital System Architectures** Return to the principle «execution environment shapes architecture» across all course topics. Architectural debt as a consequence of past infrastructural decisions. Trends in infrastructure evolution and their architectural implications. **Comprehensive case:** A real digital system with development history. Reconstruct: how infrastructural decisions at each stage defined subsequent constraints. **Discussion:** What would be decided differently with course knowledge? **LLM use:** - **Architectural opponent:** Present the final note and ask to formulate three architectural risks the student underestimated. - **Verification:** Which risks did the model identify correctly, which — without considering the system specifics?	2	2	4