ЗАТВЕРДЖУЮ
Начальник кафедри інформаційних технологій та систем електронних комунікацій, к.т.н., доцент
Назарій БУРАК
«____» _____ 20___ р.

МЕТОДИЧНА РОЗРОБКА

для проведення лекційного заняття із здобувачами освіти першого (бакалаврського) рівня вищої освіти спеціальності F3 «Комп'ютерні науки»
з дисципліни «WEB-технології та WEB-програмування»

Лекція № 1. Тема 1.1: «Вступ до вебтехнологій та вебпрограмування»

Мета: сформувати у здобувачів освіти цілісне уявлення про принципи функціонування вебу, архітектуру «клієнт-сервер», протокол HTTP/HTTPS, будову клієнтської та серверної частин вебзастосунку й інструменти веброзробника.

Час: 2 години.

Місце заняття: лекційна аудиторія (або онлайн у MS Teams).

Матеріальне забезпечення: мультимедійний проєктор, ПК з доступом до мережі Internet, презентація до лекції, електронний курс дисципліни у «Віртуальному університеті».

Навчальні питання:

1. Історія та класифікація вебтехнологій.
2. Архітектура «клієнт-сервер».
3. Протоколи HTTP/HTTPS: запит, відповідь, методи, коди стану, заголовки.
4. Cookies і сесії.
5. Клієнтська та серверна частини вебзастосунку. Статичні, динамічні сайти та SPA.
6. Інструменти веброзробника. Наскрізний проєкт курсу.

Після лекційного заняття здобувачі освіти повинні:

знати: етапи розвитку вебу; структуру HTTP-запиту і відповіді; призначення методів GET, POST, PUT, DELETE та класів кодів стану; механізм

cookies і сесій; відмінності статичних, динамічних сайтів і односторінкових застосунків;

вміти: аналізувати HTTP-обмін засобами розробника браузера та утилітою curl, пояснювати шлях запиту від браузера до сервера і назад.

Література:

1. Віртуальне навчальне середовище ЛДУ БЖД «Віртуальний університет» [Електронний ресурс]. – Доступний з: https://virt.ldubgd.edu.ua/course/view.php?id=2111
2. MDN Web Docs: HTTP [Електронний ресурс]. – Доступний з: https://developer.mozilla.org/en-US/docs/Web/HTTP
3. Fielding R., Nottingham M., Reschke J. RFC 9110. HTTP Semantics. – IETF, 2022.
4. Schildt H. Java: The Complete Reference. 13th ed. – McGraw Hill, 2024. – 1280 p.
5. Walls C. Spring in Action. 6th ed. – Manning, 2022. – 520 p.

І. Підготовка до заняття

Перед початком лекційного заняття викладач зобов’язаний переконатися у справності технічних засобів навчання та доступі до мережі Internet (для демонстрації засобів розробника браузера). Усі методичні матеріали мають бути розміщені у віртуальному навчальному середовищі.

ІІ. Зміст лекції

Вступ

Практично кожна сучасна інформаційна система – від державних реєстрів і систем оповіщення до інтернет-магазинів – має вебінтерфейс або вебAPI. Вебзастосунок складається з двох частин: **клієнтської**, що виконується в браузері, та **серверної**, що обробляє запити, працює з базою даних і реалізує бізнес-логіку. Дисципліна зосереджена на серверній частині мовою Java: від сервлетів і контейнера Apache Tomcat до фреймворку Spring Boot. Протягом семестру на лекціях і лабораторних заняттях ми поступово будуватимемо навчальний вебзастосунок «Реєстр укриттів міста», а кожен здобувач паралельно розроблятиме власний вебсервіс, який стане основою курсової роботи.

1. Історія та класифікація вебтехнологій

Всесвітню павутину (World Wide Web) запропонував Тім Бернерс-Лі у 1989 р. в CERN. Він поєднав три ідеї: мову розмітки гіпертексту **HTML**, адресацію ресурсів через **URL** і протокол передавання гіпертексту **HTTP**. Технічні передумови – поширення мережі Internet і стеку TCP/IP; економічні – потреба організацій публікувати інформацію і надавати послуги без встановлення спеціальних програм у клієнтів.

Етап	Технології	Особливості
Web 1.0 (1990-ті)	HTML, HTTP/1.0, CGI	статичні сторінки, «вебсайт як брошура», мінімум інтерактивності
Динамічний веб (кінець 1990-х – 2000-ні)	PHP, Java Servlet/JSP, ASP, СУБД	сторінки формуються на сервері з даних БД, з’являються сесії і форми
Web 2.0 (2005–2015)	AJAX, JavaScript, REST, JSON	соціальні сервіси, оновлення частини сторінки без перезавантаження
Сучасний веб (2015 →)	SPA (React, Angular, Vue), Spring Boot, хмари, контейнери	розділення фронтенду і бекенду через API, мобільні клієнти, мікросервіси

Таблиця 1 – Етапи розвитку вебтехнологій

Вебтехнології класифікують за місцем виконання: **клієнтські** (HTML, CSS, JavaScript, WebAssembly) і **серверні** (Java, C#, Python, PHP, JavaScript/Node.js); за призначенням: розмітка і оформлення, програмування логіки, зберігання даних (SQL- і NoSQL-СУБД), обмін даними (HTTP,

WebSocket, JSON, XML), розгортання (вебсервери, контейнери, хмарні платформи).

2. Архітектура «клієнт-сервер»

> **Клієнт-сервер** – архітектура, у якій **клієнт** ініціює запит до ресурсу, а **сервер** очікує запити, обробляє їх і повертає відповідь. Один сервер обслуговує багато клієнтів одночасно.

Розглянемо, що відбувається, коли користувач вводить у браузері адресу `https://shelters.example.ua/shelters`: (1) браузер розбирає URL на схему (`https`), хост, порт (за замовчуванням 443) і шлях; (2) служба DNS перетворює ім'я хоста на IP-адресу; (3) встановлюється TCP-з'єднання, а для HTTPS – ще й захищений TLS-канал; (4) браузер надсилає HTTP-запит; (5) вебсервер передає запит застосунку, той звертається до БД і формує відповідь; (6) браузер отримує HTML, довантажує CSS, JavaScript, зображення і відображає сторінку.

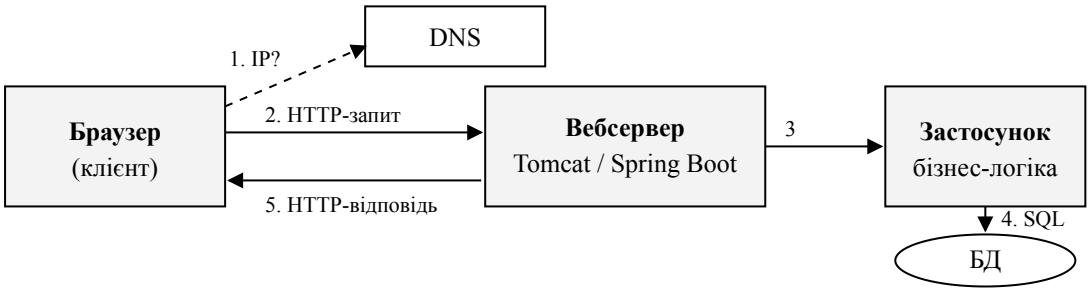

Рисунок 1 – Шлях запиту від браузера до бази даних

Сервер не «пам'ятає» клієнта між запитами – протокол HTTP **не зберігає стан** (stateless). Це спрощує масштабування: запит може обробити будь-який з кількох однакових серверів. Для «пам'яті» про користувача використовують cookies і сесії (питання 4).

3. Протоколи HTTP/HTTPS

HTTP (HyperText Transfer Protocol) – протокол прикладного рівня для обміну запитами та відповідями. Версії: HTTP/1.1 (текстовий, 1997), HTTP/2 (двійкове кадрування, кілька запитів одним з'єднанням, 2015), HTTP/3 (працює поверх QUIC/UDP, 2022). **HTTPS** – це HTTP усередині захищеного каналу TLS: дані шифруються, а сертифікат підтверджує, що клієнт спілкується саме з потрібним сервером. Сьогодні HTTPS – обов'язковий стандарт для будь-якого публічного сайту.

Запит складається з **рядка запиту** (метод, шлях, версія), **заголовків**, порожнього рядка і необов'язкового **тіла**. Відповідь – з **рядка стану** (версія, код, пояснення), заголовків і тіла:

```
POST /api/shelters HTTP/1.1                    HTTP/1.1 201 Created
Host: localhost:8080                           Location: /api/shelters/7
Content-Type: application/json                 Content-Type:
application/json
Authorization: Basic YWRtaW46YWRtaW4=
                                               {"id":7,"name":"Укриття №
7",
{"name":"Укриття № 7","capacity":70}            "status":"Не готове"}
```

Лістинг 1 – HTTP-запит (ліворуч) і відповідь (праворуч)

Методи визначають, що клієнт хоче зробити з ресурсом:

Метод	**Призначення**	**Безпечний / ідемпотентний**	**Приклад**
GET	отримати ресурс	так / так	`GET /shelters/3`
POST	створити ресурс, надіслати форму	ні / ні	`POST /shelters`
PUT	замінити ресурс повністю	ні / так	`PUT /shelters/3`
PATCH	змінити частину ресурсу	ні / ні	`PATCH /shelters/3`
DELETE	видалити ресурс	ні / так	`DELETE /shelters/3`
HEAD, OPTIONS	лише заголовки; доступні методи (CORS)	так / так	`HEAD /`

Таблиця 2 – Основні методи HTTP

Безпечний метод не змінює стан сервера; *ідемпотентний* – повторний виклик дає той самий результат, що й одиничний. Тому браузер може безпечно повторити GET, але перепитує користувача перед повторним надсиланням POST-форми.

Коди стану поділено на п'ять класів: **1xx** – інформаційні; **2xx** – успіх (200 OK, 201 Created, 204 No Content); **3xx** – перенаправлення (301 Moved Permanently, 302 Found, 304 Not Modified); **4xx** – помилка клієнта (400 Bad Request, 401 Unauthorized – потрібна автентифікація, 403 Forbidden – немає прав, 404 Not Found, 409 Conflict); **5xx** – помилка сервера (500 Internal Server Error, 503 Service Unavailable).

Заголовки передають метадані. Найважливіші: `Host` – ім'я сервера; `User-Agent` – програма клієнта; `Accept` / `Content-Type` – бажаний і фактичний формат даних (MIME-тип, напр. `text/html`, `application/json`); `Content-Length` – розмір тіла; `Authorization` – облікові дані; `Cache-Control`, `ETag`, `Last-Modified` – кешування; `Location` – адреса для перенаправлення або нового ресурсу.

4. Cookies і сесії

Cookie – невеликий фрагмент даних, який сервер передає заголовком `Set-Cookie`, а браузер повертає в кожному наступному запиті до цього сайту в заголовку `Cookie`. Атрибути: `Expires`/`Max-Age` – термін дії; `Path`, `Domain` – область дії; `HttpOnly` – недоступне для JavaScript (захист від викрадення через XSS); `Secure` – лише через HTTPS; `SameSite` – обмеження передавання з інших сайтів (захист від CSRF).

Сесія – дані про користувача, що зберігаються **на сервері** (наприклад, хто увійшов до системи). Браузер отримує лише ідентифікатор сесії в cookie (у Java-серверах – `JSESSIONID`):

```
HTTP/1.1 302 Found                                    ← відповідь на
успішний вхід
Set-Cookie: JSESSIONID=8F3A…C1; Path=/; HttpOnly
Location: /shelters

GET /shelters HTTP/1.1                                ← наступний запит
браузера
Cookie: JSESSIONID=8F3A…C1
```

Лістинг 2 – Ідентифікатор сесії в cookie

Так поверх «безстанового» HTTP будується вхід до системи, кошик інтернет-магазину, збереження обраних фільтрів. Детально роботу із сесіями ми розглянемо в темі 1.4 (Servlet API) і реалізуємо автентифікацію засобами Spring Security у темі 2.4.

5. Клієнтська та серверна частини. Статичні, динамічні сайти та SPA

Клієнтська частина (frontend): **HTML** описує структуру і зміст сторінки (заголовки, таблиці, форми), **CSS** – оформлення і адаптивність до розміру екрана, **JavaScript** – поведінку (обробку подій, запити до API без перезавантаження сторінки). **Серверна частина (backend)** приймає HTTP-

запити, перевіряє права доступу і дані, виконує бізнес-логіку, працює з БД і повертає HTML або JSON. У курсі серверна частина розробляється мовою Java.

Тип	Як формується сторінка	Приклад у курсі
Статичний сайт	готові HTML-файли однакові для всіх користувачів	ЛР 1: сторінка зі списком укриттів
Динамічний (серверний рендеринг)	сервер для кожного запиту генерує HTML з даних БД (JSP, Thymeleaf)	ЛР 4 (сервлети + JSP), ЛР 7 (Thymeleaf)
SPA (односторінковий застосунок)	браузер один раз завантажує JavaScript-застосунок, далі отримує від сервера лише дані JSON через REST API	ЛР 7: REST API, який може використати SPA або мобільний застосунок

Таблиця 3 – Способи формування вебсторінок

Серверний рендеринг простіший і краще індексується пошуковими системами; SPA забезпечує «відчуття застосунку» і дає змогу одному бекенду обслуговувати і вебклієнт, і мобільні застосунки. На практиці підходи поєднують.

6. Інструменти веброзробника. Наскрізний проєкт курсу

JDK 21 – комплект розробника Java (компілятор, віртуальна машина, утиліти, зокрема простий вебсервер `jwebserver`). **IntelliJ IDEA** або **Visual Studio Code** – середовища розробки. **Maven** – збирання проєкту і керування залежностями. **Git** і **GitHub** – контроль версій і здача лабораторних робіт (посилання на репозиторій із тегом `lab-0N`). **Засоби розробника браузера** (DevTools, F12): вкладка Network показує кожен запит, його заголовки, код стану і час; Application – cookies; Lighthouse – перевірку продуктивності й доступності. **curl** і **Postman** – надсилання HTTP-запитів вручну:

```
curl -I https://ldubgd.edu.ua          # лише заголовки відповіді
(метод HEAD)
curl -v http://localhost:8000/         # повний обмін: запит і
відповідь
curl -i http://localhost:8000/missing  # код стану 404
```

Лістинг 3 – Дослідження HTTP утилітою curl

Наскрізний проєкт «Реєстр укриттів міста» (https://github.com/SidiromUA/shelter-registry) показує, як застосунок зростає від теми до теми: статична сторінка (ЛР 1) → база даних і JDBC (ЛР 2) → багаторівнева архітектура і доступність (ЛР 3) → сервлети на Tomcat (ЛР 4) →

Spring Boot (ЛР 5–6) → вебсторінки і REST API (ЛР 7) → Spring Data JPA, безпека і тести (ЛР 8). Стан проєкту на кожному етапі відкривається командою `git checkout lab-0N`. Кожен здобувач виконує ті самі етапи для власної предметної області, яку обирає до виконання ЛР 2.

Висновки

Веб побудований на архітектурі «клієнт-сервер» і протоколі HTTP, який не зберігає стан між запитами. Запит і відповідь мають чітку структуру: метод, URL, заголовки, тіло, код стану. Cookies і сесії дають змогу «пам'ятати» користувача. Сучасний вебзастосунок поєднує клієнтську частину (HTML, CSS, JavaScript) і серверну частину, яку в цьому курсі ми розробляємо на Java та Spring Boot.

Контрольні запитання

1. Які три технології запропонував Т. Бернерс-Лі? Яке їхнє призначення?
2. Опишіть кроки обробки запиту від введення URL до відображення сторінки.
3. Що означає, що HTTP не зберігає стан? Як це впливає на масштабування?
4. З яких частин складаються HTTP-запит і HTTP-відповідь?
5. Чим відрізняються методи GET і POST, PUT і PATCH? Які з них ідемпотентні?
6. Назвіть класи кодів стану. Чим відрізняються коди 401 і 403?
7. Для чого призначені атрибути cookie HttpOnly, Secure, SameSite?
8. Чим сесія відрізняється від cookie?
9. Порівняйте серверний рендеринг і односторінкові застосунки.

Завдання для самостійної роботи

Опрацювати матеріал лекції. Дослідити заголовки і коди стану HTTP для трьох обраних вебсайтів (DevTools, curl). Завершити та оформити лабораторну роботу № 1. Пройти тест № 1 у «Віртуальному університеті».

III. Підведення підсумків заняття

Узагальнити основні положення лекції, відповісти на запитання здобувачів освіти. Наголосити на зв'язку матеріалу з лабораторною роботою № 1. Видати завдання для самостійної роботи.

Методичну розробку розробив:

Доцент кафедри ІТтаСЕК, к.т.н. Роман ГОЛОВАТИЙ

Методична розробка обговорена на засіданні кафедри ІТтаСЕК
Протокол № ____ від «____» _____ 20___ р.