Lab 10. Implementation of HTTP

Objective: Learn secure request processing, input validation, and correct HTTP response handling. Understand how attacks exploit incorrect HTTP parsing and input handling.

Students design a set of endpoints and implement controllers that correctly parse headers, parameters, JSON bodies, form data. They must validate all inputs, return meaningful status codes.

Tasks:

1. Design API routes (REST) or MVC controller endpoints (MVC)

Document the endpoints (method, path, params, headers).

Explain what each should return (status codes, JSON/HTML views).

2. Input Handling

Write controllers that manually read request headers (@RequestHeader) and bodies (@RequestBody or form data for MVC).

Practice JSON vs form data.

3. Implement Basic Validation

Create a DTO with @NotNull, @Email, etc.

Create custom validator for one field.

4. Generate Proper Responses

Return 200, 400, 404, 415, 500 based on different failure paths.

Return structured error JSON or Thymeleaf error page.

Checklist:

HTTP routing and methods

- Wrote API documentation with: method, path, params, expected status codes
- Implemented GET, POST endpoints.
- OPTIONAL: Implemented PUT, DELETE, PATCH, OPTIONS, HEAD.

Request processing

- Read headers using @RequestHeader.
- Parsed JSON using DTO + Jackson.
- MVC: Parsed form data (@ModelAttribute).

Input validation

- DTO with @NotNull, @Size, @Email, etc.
- Validation errors return 400 with structured JSON/page.
- Added custom validator for one field (e.g., password rules).

Response handling

- Controllers return appropriate status codes (200, 400, 404, 500).
- Responses include correct Content-Type.
- Added global exception handling for bad input (optional).