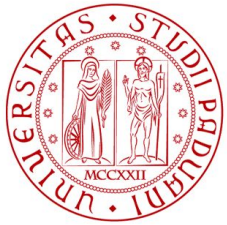
CyberSecurity: Principle and Practice

BSc Degree in Computer Science
2025-2026

Lesson 8: Web backend: HTTP and SQL

Prof. Mauro Conti
Department of Mathematics
University of Padua
conti@math.unipd.it
http://www.math.unipd.it/~conti/

Teaching Assistants
Giulio Umbrella
giulio.umbrella@phd.unipd.it
Francesco De Giudici
francesco.degiudici@studenti.unipd.it

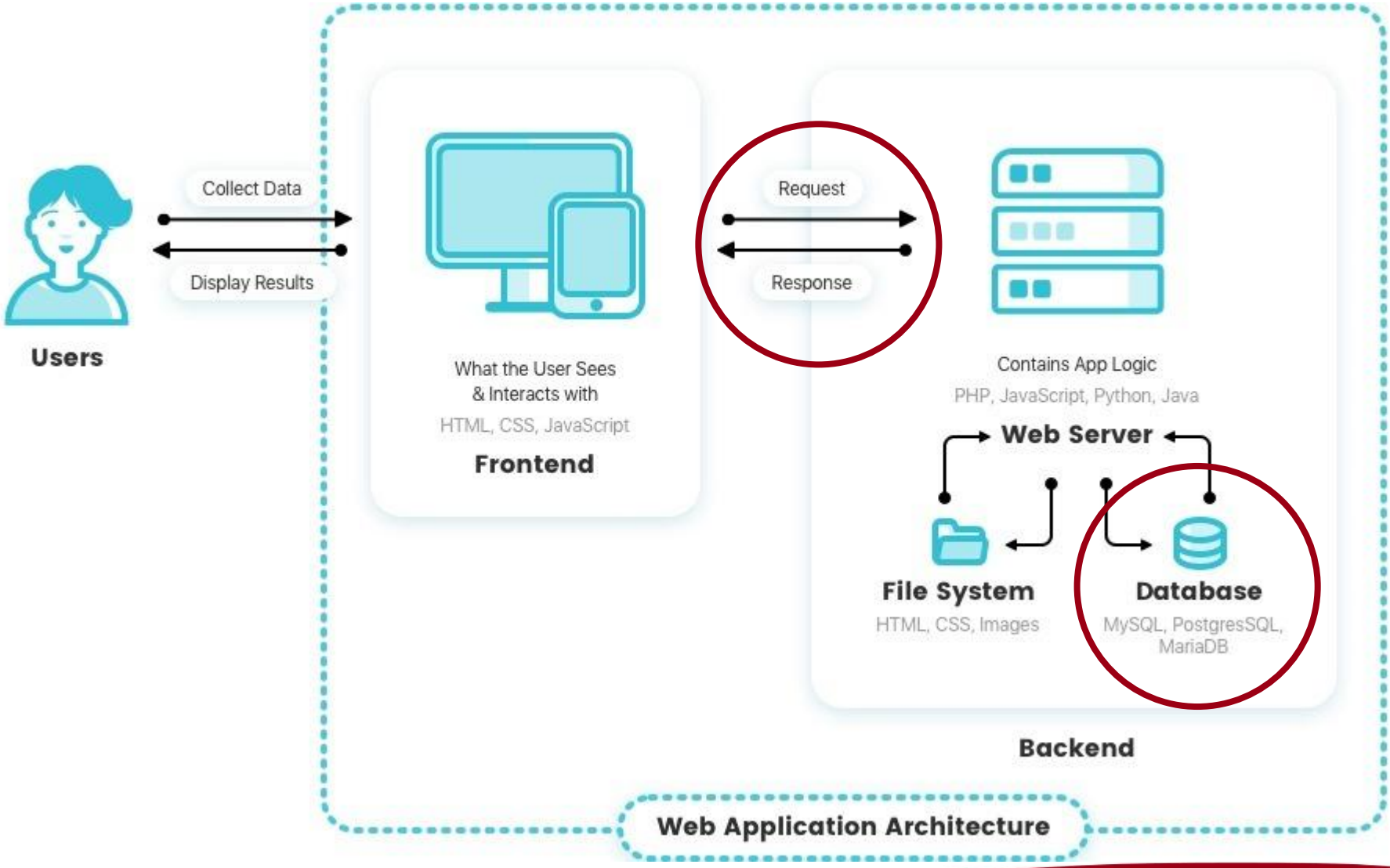


UNIVERSITÀ DEGLI STUDI DI PADOVA

SPRITZ SECURITY & PRIVACY RESEARCH GROUP

DIPARTIMENTO MATEMATICA

Web App Architecture

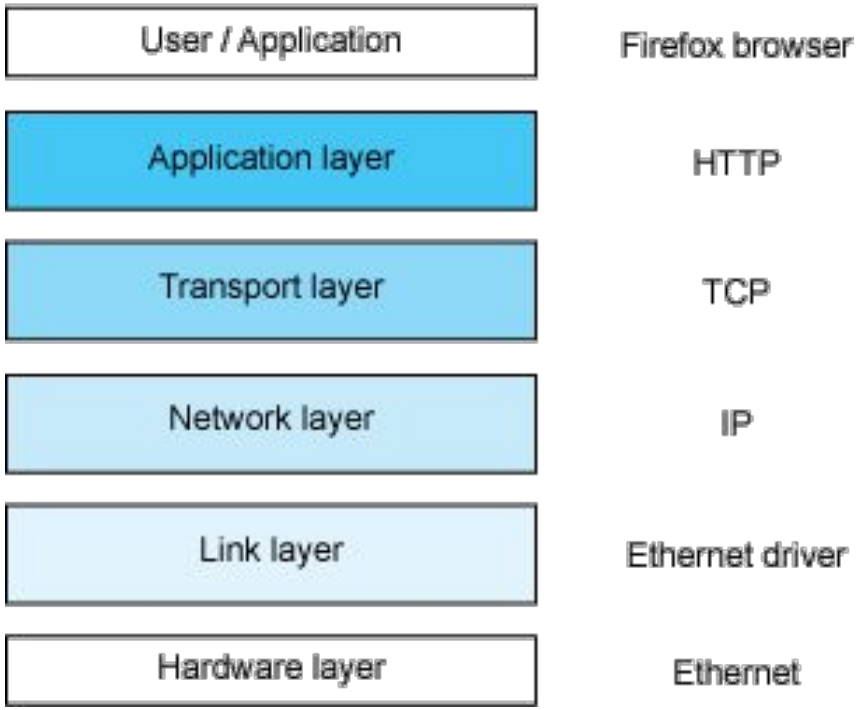


Client - Server Paradigm

- Classic architecture **client-server**
  - Client: a device that requests for a service
  - Server: who provide a service
  - There are also other paradigms
- Each device has a **unique IP address**
  - Essential for the identification
- A device might have **multiple communications**
  - The role of *PORTS*
- Devices communicates through **protocols**
  - Define some *rules*

HTTP

- HTTP is **application-layer protocol** for transmitting hypermedia documents, such as HTML
- HTTP is **stateless**, ie server maintains no information about past client requests. That's why cookies are needed.



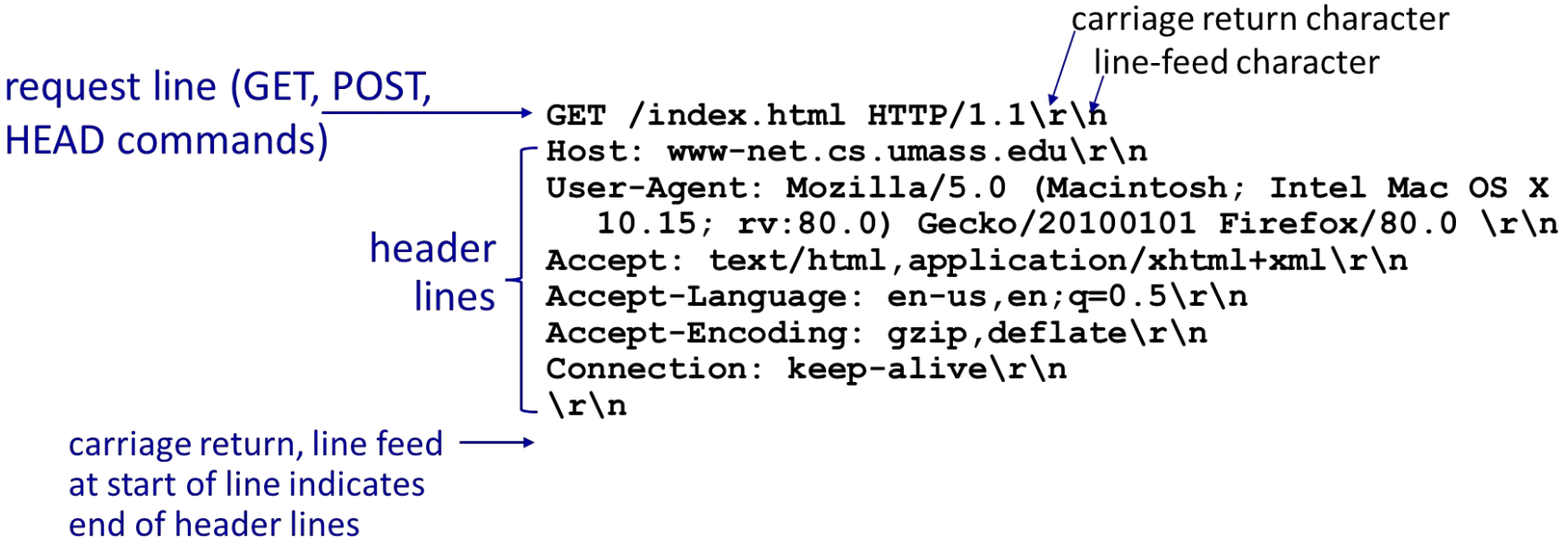
HTTP - Client Requests

- Clients requests stuffs to the server with some methods
  - e.g., **GET, POST**, PUT, HEAD, DELETE, PATCH, OPTIONS

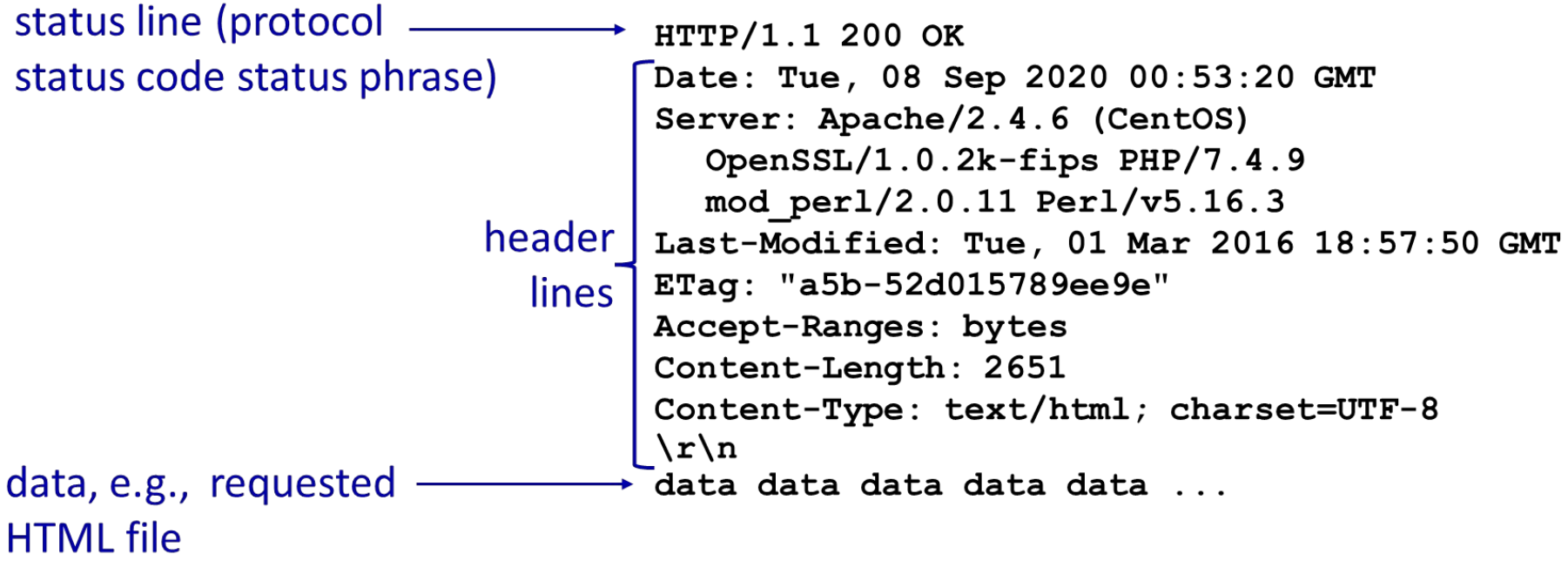
- GET is used to request data from a specified resource
  - e.g., */test/demo.php?name1=value1&name2=value2*
  - The query string is sent in the URL of a GET request

- POST is used to send data to a server to create/update a resource
  - Data sent is stored in the request body of the HTTP request
- An attacker might play with these fields ...

HTTP - Client Requests



HTTP - Server Response



HTTP - Server Response

- status code appears in 1st line in server-to-client response message.
- some sample codes:

  200 OK

  request succeeded, requested object later in this message

  301 Moved Permanently

  requested object moved, new location specified later in this message (in Location: field)

  400 Bad Request

  request msg not understood by server

  404 Not Found

  requested document not found on this server

HTTPS

- What about security?
  - **HTTPS**: secure version of http obtained by making http *using a TLS connection between two hosts*

- TLS guarantees confidentiality, data integrity, server authentication, and resistance to several specific attacks while data are being transmitted over the network

Input Validation

- Applications usually expected some inputs
  - e.g., a calculator expects some numbers
- We must check the input that our application receives
- This process is called **input validation and sanitization**
  - The app processes only feasible inputs, rejecting non feasible ones
- Where should we put these stuffs?
  - Client side: can be easily bypassed (e.g., if based on JS)
  - Server side: increase the server's overhead

SQL

In plenty of web application Relational Database Systems (RDBS) are used to store data of a certain application

- Stores data in tables
- Tables have a unique name and type description of its fields (integer, string)
- Each column stores a single piece of data (field)
- Each row represents a record (or object!)
- Each row may have/is identified by a unique primary key which may be

SQL


```
appdb=# select * from myuser;
 id | username |              password              |       salt
----+----------+----------------------------------+------------------
  1 | alice    | 5f4dcc3b5aa765d61d8327deb882cf99 | a1b2c3d4e5f6g7h8
  2 | bob      | 098f6bcd4621d373cade4e832627b4f6 | b2c3d4e5f6g7h8i9
  3 | carol    | 25f9e794323b453885f5181f1b624d0b | c3d4e5f6g7h8i9j0
(3 rows)
```

SQL

- We interact with RDBMs using Structured Query Language (SQL)
- SQL allow us to Create, Retrieve, Update and Destroy (CRUD) rows and tables.

CREATE table

```
CREATE TABLE <Table Name>
(Collumn1 DataType,
Collumn2 DataType,
Collumn3 DataType,
```

RETRIEVE data from table

```
SELECT <Column List>
FROM <Table Name> WHERE
<Condition 1> AND/OR
<Condition 2>
```

UPDATE data from table

```
UPDATE <Table Name>
SET <Column>=<Value>,
    <Column2>=<Value>,
WHERE <Search Condition>
```

DELETE data from table

```
DELETE FROM <Table Name>
WHERE <Search Condition>
```

```
appdb=# select password from myuser where id=1;
             password
----------------------------------
 5f4dcc3b5aa765d61d8327deb882cf99
(1 row)
```

Questions? Feedback? Suggestions?