Web Application Development

2019
Who are we?

Tsotne Kekelia

Lectures, Practical sessions

Master’s student @ UT

7+ years of experience as a web developer

Tair Vaher

Practical sessions

Master’s student @ UT

5+ years of experience as a web developer
About this course

- Multitier Architecture
- HTTP
- HTML
- CSS
- Javascript
- DOM
- AJAX
- Frameworks (VueJS)
- Backend (NodeJS)
- Testing
- External APIs
- UI / UX
Course structure

Homeworks

1. Homework 1 (Week 4) - 5 points
2. Homework 2 (Week 6) - 10 points
3. Homework 3 (Week 8) - 15 points
4. Homework 4 (Week 14) - 20 points

Exam

- 50 points, multiple choice test
- Passing minimum: 51 points
- Exam minimum: 21 points
- 1 Resit
Course structure

- Attendance is not mandatory
- You will be split into group of 3
- Homeworks will be presented and graded during practical sessions, at least one person from group needs to attend.

Register to groups: https://forms.gle/EkyNakcwagWhiWd8

Communications via Slack channel:


Join by visiting: http://tiny.cc/wad2019
What are we gonna do?

1st Half of the semester

Simple HTML, CSS, Javascript project
What are we gonna do?

2st Half of the semester

Chirp - social network website.

Vue.Js for front-end

Node.Js for back-end
Pretty much any text editor will do.

Preferred: WebStorm

Webstorm: https://www.jetbrains.com/webstorm/

Atom: https://atom.io/

Sublime: https://www.sublimetext.com/

VS Code: https://code.visualstudio.com/
Web Browser

Preferred: Google Chrome

Google Chrome: https://www.google.com/chrome/


Safari: https://www.apple.com/safari/

Edge: https://www.microsoft.com/en-us/windows/microsoft-edge
**Additional**

Git:
https://git-scm.com/

Postman:
https://www.getpostman.com/
Questions?
Multitier Architecture

Multilayered Architecture
In software engineering, **multitier architecture** (often referred to as *n-tier architecture*) or **multilayered architecture** is a client–server architecture in which presentation, application processing, and data management functions are physically separated. The most widespread use of multitier architecture is the **three-tier architecture**. [1]

Three-tier Architecture

Presentation tier
GUI, UI, View

Application tier
Business logic, Logic tier, or Middle tier

Data tier
Database servers, File shares
Three-tier Architecture in Web

Front-end
- HTML, CSS, Javascript

Back-end
- Node.js application

Database
- Mysql database
Client-Server communication

- **Client**
  - Enter URL
  - Request Page
  - Display Page

- **The Internet**
  - Look Up the IP

- **Web Server**
  - Receive Request
  - Execute Request
  - Process Data
  - Return Page

- **Database**
  - Execute Query
HTTP

Hypertext Transfer Protocol
The Hypertext Transfer Protocol (HTTP) is an application protocol for distributed, collaborative, hypermedia information systems. [1]

**Hypertext**

Hypertext is text displayed on a computer display or other electronic devices with references (hyperlinks) to other text that the reader can immediately access.

**Protocol**

Protocol is a set of rules governing a communication between a client and a server.

**Hypermedia**

Hypermedia, an extension of the term hypertext, is a nonlinear medium of information that includes graphics, audio, video, plain text and hyperlinks.
URI & URL

Uniform Resource Identifiers & Uniform Resource Locators
**URI & URL**

**URI**

A Uniform Resource Identifier (URI) is a string of characters that unambiguously identifies a particular resource. To guarantee uniformity, all URIs follow a predefined set of syntax rules, but also maintain extensibility through a separately defined hierarchical naming scheme (e.g. http://)

**URL**

A Uniform Resource Locator (URL), colloquially termed a web address, is a reference to a web resource that specifies its location on a computer network and a mechanism for retrieving it. A URL is a specific type of Uniform Resource Identifier (URI), although many people use the two terms interchangeably.
Back to HTTP
HTTP state

HTTP is a stateless protocol. A stateless protocol does not require the HTTP server to retain information or status about each user for the duration of multiple requests. [1]

HTTP Methods

GET
The GET method requests a representation of the specified resource. Requests using GET should only retrieve data and should have no other effect.

HEAD
The HEAD method asks for a response identical to that of a GET request, but without the response body.

POST
The POST method requests that the server accept the entity enclosed in the request as a new subordinate of the web resource identified by the URI.

DELETE
The DELETE method deletes the specified resource.

PUT
The PUT method requests that the enclosed entity be stored under the supplied URI. If the URI refers to an already existing resource, it is modified; if the URI does not point to an existing resource, then the server can create the resource with that URI.

OPTIONS
The OPTIONS method returns the HTTP methods that the server supports for the specified URL.

PATCH
The PATCH method applies partial modifications to a resource.
Request and Response

Request Message

[METHOD] [PATH] HTTP/1.1 #Request line

[HEADER FIELD]:[VALUE] #Header fields

[BODY] #Message body (Opt.)

Response Message

HTTP/1.1 [CODE] [REASON] #Status line

[HEADER FIELD]:[VALUE] #Header fields

[BODY] #Message body (Opt.)
Request and Response

Request Message

GET /sample/example?param=1 HTTP/1.1
Host: www.example.com
Content-Type: application/json
Cache-Control: no-cache

Response Message

HTTP/1.1 200 OK
Date: Mon, 23 May 2005 22:38:34 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 138

<html>
  <head>
    <title>An Example Page</title>
  </head>
  <body>
    <p>Hello World, this is a very simple HTML document.</p>
  </body>
</html>
<table>
<thead>
<tr>
<th>Method</th>
<th>Request has body</th>
<th>Response has body</th>
<th>Safe</th>
<th>Idempotent</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>Optional</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>HEAD</td>
<td>Optional</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>POST</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>DELETE</td>
<td>Optional</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>PUT</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>OPTIONS</td>
<td>Optional</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>PATCH</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
HTTP Response codes

1xx - Informational
2xx - Successful
3xx - Redirection
4xx - Client Error
5xx - Server Error

Common response codes:

200 - OK
400 - Bad Request
401 - Unauthorized
403 - Forbidden
404 - Not found
500 - Internal Server Error
502 - Bad Gateway
503 - Service Unavailable
418 - I'm a teapot

Full list: https://en.wikipedia.org/wiki/List_of_HTTP_status_codes
Recap

- Three-tier architecture in web
  - Presentation tier = Front-end
  - Application tier = Back-end
  - Data-tier = Database

- Client communicates with server via HTTP

- **HTTP**: Hypertext Transfer Protocol

- **Hypermedia** = hypertext + graphics, audio, video

- **URL** is one particular type of **URI**

- Required Web URL components
  - Schema (*http:*)
  - Host (*www.example.com*)
  - Path (*/forum/questions/*)

- **HTTP Methods**
  - **GET** - Retrieves a resource
  - **HEAD** - Retrieves the meta data of the resource
  - **POST** - Create resource
  - **PUT** - Replace existing or Create non-existing resource
  - **PATCH** - Partially modify resource
  - **OPTIONS** - Lists available HTTP methods for the URL
  - **DELETE** - Deletes specified resource
Questions?

Next: HTML