Développement d'applications internet - DAI

Leonard Cseres | January 25, 2025

GitHub Platforme et service cloud pour le dev. de logiciels et le contrôle de version utilisant Git, permettant aux développeurs de stocker et de gérer leur code.

SSH Plus sécurisé et il n'y a pas besoin de s'authentifier à chaque fois par rapport à **HTTPS**.

Maven Composé de phases et de goals. Les phases chargent les goals, qui exécutent des tâches projet (par exemple, compiler, tester, empaqueter).

POM Project Object Model, XML qui contient des infos sur le projet et configuration de Maven.

MVNW Wrapper Maven, permet d'avoir une version de Maven spécifique tous les collaborateurs du projet et éviter des problèmes de compatibilité.

`maven-jar-plugin` Plugin Maven pour créer un fichier JAR. `maven-shade-plugin` Plugin Maven pour créer un fichier JAR **fat** (contient toutes les dépendances).

```
# Télécharge les dépendances & transitives
./mvnw dependency:go-offline
# Supprime les classes compilées
./mvnw clean
# Compile le code source
./mvnw compile
# Emballe l'application dans un fichier JAR
./mvnw package
# Plusieurs phases
./mvnw ...
```

Java Portable grâce à la JVM, orienté objet, multi-threadé, fortement typé, compilé en byte-code. (SDKMAN! utilisé pour gérer les versions)

JAR Java ARchive, contient des fichiers .class et des métadonnées.

ASCII 7 bits, 128 caractères.

Extended ASCII 8 bits, 256 caractères.

Unicode UTF-8, UTF-16, UTF-32, standard pour tous les caractères.

UTF-8 8 bits, 1 à 4 octets, compatible ASCII.

```
import java.io.*;
// Binary
InputStream; OutputStream; FileInputStream;
FileOutputStream; BufferedInputStream;
BufferedOutputStream;
// Text
Reader; Writer; FileReader; FileWriter;
BufferedReader; BufferedWriter;
import java.nio.charset.StandardCharsets;
// Exception
IOException; UnsupportedEncodingException;
FileNotFoundException;
```

```
// Binary Read
String filename;

try (InputStream fis = new
     FileInputStream(filename)) {
  int b;
  while ((b = fis.read()) != -1) {
    // Do nothing - simulate processing
  }
} catch (IOException e) {
  System.err.println("Error: " + e.getMessage());
}
```

```
// Binary Write
String filename;
int sizeInBytes;

try (OutputStream fos = new
     FileOutputStream(filename)) {
  for (int i = 0; i < sizeInBytes; i++) {
    fos.write(1);
  }
} catch (IOException e) {
  System.err.println("Error: " + e.getMessage());
}
```

```
// Binary Buffered Read
String filename;

try (InputStream fis = new
     FileInputStream(filename);
     BufferedInputStream bis = new
          BufferedInputStream(fis)) {
  int b;
  while ((b = bis.read()) != -1) {
    // Do nothing - simulate processing
  }
} catch (java.io.IOException e) {
  System.err.println("Error: " + e.getMessage());
}
```

```
// Binary Buffered Write
String filename, int sizeInBytes;

try (OutputStream fos = new
     FileOutputStream(filename);
     BufferedOutputStream bos = new
          BufferedOutputStream(fos)) {
  for (int i = 0; i < sizeInBytes; i++) {
    bos.write(1);
  }
  bos.flush();
} catch (java.io.IOException e) {
  System.err.println("Error: " + e.getMessage());
}
```

```
// Text Read
String filename;

try (Reader reader = new FileReader(filename,
     StandardCharsets.UTF_8)) {
  int b;
  while ((b = reader.read()) != -1) {
    // Do nothing - simulate processing
  }
} catch (IOException e) {
  System.err.println("Error: " + e.getMessage());
}
```

```
// Text Write
String filename, int sizeInBytes;

try (Writer writer = new FileWriter(filename,
     StandardCharsets.UTF_8)) {
  for (int i = 0; i < sizeInBytes; i++) {
    writer.write('a');
  }
} catch (IOException e) {
  System.err.println("Error: " + e.getMessage());
}
```

```
// Text Buffered Read
String filename;

try (Reader reader = new FileReader(filename,
     StandardCharsets.UTF_8);
     BufferedReader br = new
          BufferedReader(reader)) {
  int b;
  while ((b = br.read()) != -1) {
    // Do nothing - simulate processing
  }
} catch (java.io.IOException e) {
  System.err.println("Error: " + e.getMessage());
}
```

```
// Text Buffered Write
String filename, int sizeInBytes;

try (Writer writer = new FileWriter(filename,
     StandardCharsets.UTF_8);
     BufferedWriter bw = new
          BufferedWriter(writer)) {
  for (int i = 0; i < sizeInBytes; i++) {
    bw.write('a');
  }
  bw.flush();
} catch (IOException e) {
  System.err.println("Error: " + e.getMessage());
}
```

Docker *Bare metal* software runs directly on hardware, *virtualization* software runs on a virtual machine, *containerization* software runs in a container.

Image read-only template for container creation

Container runnable instance of an image

Registry service storing images

```
FROM eclipse-temurin:21-jre
COPY target/app.jar app.jar
ENTRYPOINT ["java", "-jar", "app.jar"]
CMD ["goodbye"]
```

DNS

Record Type	Description
NS	Name server
CNAME	Alias
A	IPv4 address
AAAA	IPv6 address
MX	Mail exchange (email server)
TXT	Text record (e.g., SPF, DKIM)

Ports Unsigned 16-bit, 0-1023 reserved

Concurrency

```
@Override
public Integer call() {
  try (ExecutorService executorService =
  Executors.newFixedThreadPool(2);) {
    executorService.submit(this::emittersWorker);
    executorService.submit(this::operatorsWorker);
  } catch (Exception e) {
    System.out.println("[Receiver] Exception: " +
          e);
    return 1;
  }
  return 0;
}
```

TODO: MORE CONCURRENCY

Protocole Applicatif

1. **Aperçu** Quel est le but?
2. **Protocole de transport** Quel protocol? Quel port? Encodage? Délimiteur? Messages texte ou binaire. Qui initialise la communication?
3. **Messages** Foncitionalité, requête et réponses
4. **Examples** Fonctionnel et non-fonctionnel

SMTP (Send) Send emails from client to server

- 25 (unencrypted)
- 465, 587 (encrypted; 587 recommended)

```
EHLO <sender>  # or HELO
MAIL FROM: <sender email address>
RCPT TO: <recipient email address>
DATA
<email content>
.
QUIT
```

POP3 (Retrieve) Download emails (no sync)

- 110 (unencrypted)
- 995 (encrypted)

IMAP (Sync) Sync emails (server-client updates)

- 143 (unencrypted)
- 993 (encrypted)

SSH Protocol that allows you to connect to a remote machine. It is a replacement for the Telnet protocol. Most common supported algorithms: *RSA*, *DSA*, *ECDSA*, *Ed25519*.

Ed25519 and *ECDSA* are the most recent and are considered as more secure than *RSA* and *DSA*.

Fingerprints Hash of the public key. It is used to identify the public key. Helps detect man-in-the-middle attacks.

Public keys are stored in the `~/.ssh/known_hosts file`.

SCP

```
scp [user@source-ip:]source [user@dest-ip:]dest
```

```
# Copy the file from the local to the remote
scp local.txt <username>@<vm public
    ip>:~/local.txt
```

```
# Copy the file from the remote to the local
scp <username>@<vm public ip>:~/remote.txt
    remote.txt
```

HTTP Hyper Text Transfer Protocol used to transfer data over the web based on TCP. It is a client-server protocol based on the request-response pattern (*stateless protocol*): a client (called user agent in the HTTP specification) sends a request to a server, the server processes the request and sends a response to the client.

HTTP Semantics		
HTTP/1.1	HTTP/2	HTTP/3
TLS/SSL (optional)	TLS 1.2+	TLS 1.3
TCP	TCP	QUIC UDP
IPv4 / IPv6		

Part	Name
Protocol	http or https
Host	Domain or FQDN (gaps.heig-vd.ch)
Domain Name	(heig-vd)
Top Domain	(ch)
Subdomain	Optional (gaps)
Port	Optional (:80, :443)
Path	Resource location (/consultation/fiches/uv/uv.php)
Query	Optional parameters (?id=6573)
Path Params	Optional (/users/user-id/view)

Method	Status Codes
`GET`	200 (OK), 404 (Not Found)
`POST`	201 (Created), 400 (Bad Request), 500 (Internal Server Error)
`PATCH`	200 (OK), 204 (No Content), 400 (Bad Request)
`PUT`	200 (OK), 201 (Created), 204 (No Content)
`DELETE`	200 (OK), 204 (No Content), 404 (Not Found)

```
# Request
<HTTP method> <URL> HTTP/<HTTP version>
<HTTP headers>
<Empty line>
<HTTP body (optional)>
```

```
GET /api/resource HTTP/1.1
Host: api.example.com
User-Agent: curl/8.1.2
Accept: application/json
<Empty line>
```

```
POST /api/resource HTTP/1.1
Host: api.example.com
User-Agent: curl/8.1.2
Content-Type: application/json
Accept: application/json
<Empty line>
```

```
# Response
HTTP/<HTTP version> <HTTP status code> <HTTP
    status message>
<HTTP headers>
<Empty line>
<HTTP body>
```

```
HTTP/1.1 200 OK
Date: Wed, 06 Dec 2023 18:01:17 GMT
Server: Apache
Content-Type: text/html; charset=UTF-8
Content-Length: 0
Connection: keep-alive
<Empty line>
```

HTTP Negociation A process where the client and server agree on the response format using headers like `Accept`, `Content-Type`, and `Accept-Language`.

Code Category	Description
1xx	Informational
2xx	Success
3xx	Redirection
4xx	Client Error
5xx	Server Error

```
curl -i \
  -X POST \
  -H "Content-Type: application/json" \
  -d '{}' \
  http://localhost:8080/users
```

```
curl -i \
  -X GET \
  -H 'Cookie: user=1;' \
  http://localhost:8080/profile
```

HTTP Sessions

Query parameter

```
C -> S: POST /login
S -> C: 302 Found (redirect to
    /profile?token=1234567890)
C -> S: GET /profile?token=1234567890
S -> C: 200 OK (profile page)
```

Cookie

```
C -> S: POST /login
S -> C: 302 Found (redirect to /profile and set a
    cookie with the token)
C -> S: GET /profile (the cookie is sent by the
    client)
S -> C: 200 OK (profile page)
```

API Application Programming Interface

CRUD Create, Read, Update, Delete

REST APIs

1. **Client/Server** Client and server are separate, communicating only through the API.
2. **Stateless** The server does not store session info; each request must be self-contained.
3. **Cacheable** Responses can be cached by the client with proper control.
4. **Layered System** Intermediary systems (e.g., cache, load balancer) can be added without affecting the client.
5. **Uniform Interface**
   - Use URIs/URLs to identify resources.
   - Responses include data and links for further interaction.
   - Standardized response formats.
6. **Code on Demand (optional)** Servers may send executable code for client customization.

Note: Not all APIs are REST APIs.

Web Infrastructures

All *load balancers* are reverse proxies, but not all *reverse proxies* perform load balancing.

Routing	Example
Host based	app1.example.com
Path based	example.com/app1

Host Header HTTP header sent by the client to indicate the domain being requested. Servers use this to differentiate between multiple websites hosted on the same IP address (known as virtual hosting).

```
networks:
  traefik_network:
    external: true

services:
  proxy:
    image: traefik/whoami:latest
    networks:
      - traefik_network
    expose:
      - 80
    labels:
      # Traefik
      - traefik.enable=true
      - traefik.docker.network=traefik_network
      # Routers
      - traefik.http.routers
          .proxy.entrypoints=https
      # We use Host and PathPrefix rules at the
          same time
      - traefik.http.routers.proxy.
          rule=Host(`whoami.DOMAIN`) &&
        PathPrefix(`/whoami`)
      # Stipprefix (https://example.com/foo ->
          /foo)
      - traefik.http.routers.proxy.
          middlewares=whoami-stripprefix
```

Sticky Sessions Round-robin LB by default but we redirect a user to the same server using a cookie.

```
# We add the sticky session configuration
- traefik.http.services.proxy.loadbalancer.
    sticky=true
# We can also configure the cookie name and its
    options (useful for multiple services)
- traefik.http.services.proxy.loadbalancer.
    sticky.cookie.name=whoami_cookie
- traefik.http.services.proxy.loadbalancer.
    sticky.cookie.httpOnly=true
```

Caching

Client-side (Private) Cache stored on the client after receiving a server response; reused for future requests.

Server-side (Shared) Cache stored on the server via reverse proxy or app; reused for similar incoming requests.

Expiration model The cache is valid for a certain amount of time. `Cache-Control: max-age=<number of seconds>`

Validation model The cache is valid until the data is modified.

Last-Modified

- `Last-Modified`: Timestamp of the last resource update.
- `If-Modified-Since`: 304 if unchanged (cache hit).
- `If-Unmodified-Since`: 412 if changed (cache miss) on update/delete.

ETag

- `ETag`: Hash/version of the resource.
- `If-None-Match`: 304 if unchanged (cache hit).
- `If-Match`: 412 if changed (cache miss) on update/delete.

Annexes

TCP

```java
public class TCPClient {
  public static void main(String[] args) {
    String serverAddr = "127.0.0.1";
    int port = 1234;

    try (
      Socket socket = new Socket(serverAddr, port);
      InputStreamReader isr = new
            InputStreamReader(socket.getInputStream(),
            StandardCharsets.UTF_8);
      BufferedReader in = new BufferedReader(isr);
      OutputStreamWriter osw = new
            OutputStreamWriter(socket.getOutputStream(),
            StandardCharsets.UTF_8);
      BufferedWriter out = new BufferedWriter(osw)) {
      // connected
      out.write("Hello, Server!\n");
      out.flush();
      String response = in.readLine();
      if (response == null) {
        // server diconnected
      } else {
        // response
      }
    } catch (IOException e) {}
  }
}
```

```java
public class TCPServer {
  public static void main(String[] args) {
    int port = 1234;
    int nbThreads = 2;

    try (
      ServerSocket serverSocket = new ServerSocket(port);
      ExecutorService e = Executors.newFixedThreadPool(nbThreads)) {
      // server started
      while (true) {
        Socket socket = serverSocket.accept();
        e.submit(new ClientHandler(socket));
      }
    } catch (IOException e) {}
  }

  static class ClientHandler implements Runnable {
    private final Socket socket;
    public ClientHandler(Socket socket) { this.socket = socket; }

    @Override
    public void run() {
      try (
        InputStreamReader isr = new
              InputStreamReader(socket.getInputStream(),
              StandardCharsets.UTF_8);
        BufferedReader in = new BufferedReader(isr);
        OutputStreamWriter osw = new
              OutputStreamWriter(socket.getOutputStream(),
              StandardCharsets.UTF_8);
        BufferedWriter out = new BufferedWriter(osw)) {
        // client connected
        String request = in.readLine();
        if (request != null) {
          out.write("Echo: " + request + "\n");
          out.flush();
        } else {
          // client diconnected
        }
      } catch (IOException e) {}
    }
  }
}
```

UDP Unicast

```java
public class UDPClient {
  public static void main(String[] args) {
    String serverAddr = "127.0.0.1";
    int port = 1234;

    try (DatagramSocket socket = new DatagramSocket()) {
      byte[] buffer = "Hello, Server!".getBytes();
      InetAddress address = InetAddress.getByName(serverAddr);

      DatagramPacket packet = new DatagramPacket(
        buffer, buffer.length, address, port);
      socket.send(packet);

      buffer = new byte[1024];
      packet = new DatagramPacket(buffer, buffer.length);
      socket.receive(packet);

      String response = new String(
        packet.getData(), 0, packet.getLength());
      System.out.println("Server Response: " + response);
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

```java
public class UDPServer {
  public static void main(String[] args) {
    int port = 1234;

    try (DatagramSocket socket = new DatagramSocket(port)) {
      System.out.println("Listening on port " + port);

      while (true) {
        byte[] buffer = new byte[1024];
        DatagramPacket packet = new DatagramPacket(
          buffer, buffer.length);
        socket.receive(packet);

        String msg = new String(
          packet.getData(), 0, packet.getLength());
        System.out.println("Received: " + msg);

        InetAddress clientAddr = packet.getAddress();
        int clientPort = packet.getPort();
        buffer = ("Echo: " + msg).getBytes();

        packet = new DatagramPacket(
          buffer, buffer.length, clientAddr, clientPort);
        socket.send(packet);
      }
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

UDP

```java
public class UDPMulticastSender {
  public static void main(String[] args) {
    String multicastGroup = "230.0.0.1";
    int port = 1234;

    try (DatagramSocket socket = new DatagramSocket()) {
      byte[] buffer = "Hello, Group!".getBytes();
      InetAddress group = InetAddress.getByName(multicastGroup);

      DatagramPacket packet = new DatagramPacket(
        buffer, buffer.length, group, port);
      socket.send(packet);

      System.out.println("Message sent to group");
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

```java
public class UDPMulticastReceiver {
  public static void main(String[] args) {
    String multicastGroup = "230.0.0.1";
    int port = 1234;

    try (MulticastSocket socket = new MulticastSocket(port)) {
      InetAddress group = InetAddress.getByName(multicastGroup);
      socket.joinGroup(group);

      System.out.println("Joined group");
      byte[] buffer = new byte[1024];
      DatagramPacket packet = new DatagramPacket(
        buffer, buffer.length);

      while (true) {
        socket.receive(packet);
        String msg = new String(
          packet.getData(), 0, packet.getLength());
        System.out.println("Received: " + msg);
      }
    } catch (Exception e) {
      e.printStackTrace();
    }

}
```

Javalin

```
package ch.heigvd.dai;

import io.javalin.Javalin;
import io.javalin.http.HttpStatus;

public class Main {
  public static final int PORT = 8080;

  public static void main(String[] args) {
    Javalin app = Javalin.create();

    app.get("/",
        ctx ->
            ctx.result(
                "Hello, world from a GET request method with a
                     `HttpStatus.OK` response status!"));
    app.post("/",
        ctx ->
            ctx.result(
                    "Hello, world from a POST request method with a
                         `HttpStatus.CREATED` response status!")
                .status(HttpStatus.CREATED));
    app.patch("/",
        ctx ->
            ctx.result(
                    "Hello, world from a PATCH request method with
                         a `HttpStatus.OK` response status!")
                .status(HttpStatus.OK));
    app.delete("/",
        ctx ->
            ctx.result(
                    "Hello, world from a DELETE request method with
                         a `HttpStatus.NO_CONTENT` response
                         status!")
                .status(HttpStatus.NO_CONTENT));

    app.get("/path-parameter-demo/{path-parameter}",
        ctx -> {
          String pathParameter = ctx.pathParam("path-parameter");

          ctx.result(
              "You just called `/path-parameter-demo` with path
                   parameter '"
                  + pathParameter
                  + "'!");
        });

    app.get("/query-parameters-demo",
        ctx -> {
          String firstName = ctx.queryParam("firstName");
          String lastName = ctx.queryParam("lastName");

          if (firstName == null || lastName == null) {
            throw new BadRequestResponse();
          }

          ctx.result("Hello, " + firstName + " " + lastName + "!");
        });

    app.post("/body-demo",
        ctx -> {
          String data = ctx.body();

          ctx.result("You just called `/body-demo` with data '" +
              data + "'!");
        });

    app.get("/content-negotiation-demo",
        ctx -> {
          String acceptHeader = ctx.header("Accept");

          if (acceptHeader == null) {
            throw new BadRequestResponse();
          }

          if (acceptHeader.contains("text/html")) {
            ctx.contentType("text/html");
            ctx.result("<h1>Hello, world!</h1>");
          } else if (acceptHeader.contains("text/plain")) {
            ctx.contentType("text/plain");
            ctx.result("Hello, world!");
          } else {
            throw new NotAcceptableResponse();
          }
        });

    app.get("/cookie-demo",
        ctx -> {
          String cookie = ctx.cookie("cookie");

          if (cookie == null) {
            ctx.cookie("cookie", "cookie-demo");

            ctx.result("You just called `/cookie-demo` without a
                 cookie. A cookie is now set!");
          } else {
            ctx.result(
                "You just called `/cookie-demo` with a cookie. Its
                     value is '" + cookie + "'!");
          }
        });

    app.start(PORT);
  }
}
```

Application Protocol

```
# Section 1 - Overview

This section defines the purpose of the protocol:

- What is the goal of the protocol?
- What is the problem that it tries to solve?
- What the application protocol is used for?

# Section 2 - Transport protocol

- What protocol(s) is/are involved? On which port(s)?
- How are messages/actions encoded?
- How are messages/actions delimited?
- How are messages/actions treated (text or binary)?
- Who initiates/closes the communication?
- What happens on an unknown message/action/exception?

# Section 3 - Messages

- What are the messages/actions?
- What are the parameters?
- What are the return values?
- What are the exceptions?

# Section 4 - Examples

- What are the examples of messages/actions?
- What are the examples of exceptions?
```

API Documentation

```
# Section 1 - Overview

- What is the API used for?
- Who is the intended audience?
- What are the main features of the API?
- How is the API accessed (authentication, base URL, versioning)?

# Section 2 - Authentication

- What authentication method(s) is/are used (e.g., API keys, OAuth,
    JWT)?
- How is the authentication implemented (headers, tokens, etc.)?
- What are the steps for obtaining authentication credentials?

# Section 3 - Endpoints

## General Structure

- What is the base URL of the API?
- How are endpoints formatted (RESTful paths, query strings)?

## Endpoint Details

### Endpoint: [Name or Purpose]

- **URL**: `/path/to/resource`
- **Method**: GET | POST | PUT | DELETE
- **Headers**: List of required/optional headers.
- **Request Parameters**:
  - Query Parameters: Name, Type, Required, Description
  - Body Parameters: Name, Type, Required, Description (for
      POST/PUT)
- **Responses**:
  - **Status Codes**:
    - 200: Success
    - 400: Bad Request
    - 401: Unauthorized
    - 404: Not Found
    - 500: Server Error
  - **Response Body**:
    - Example of the returned data.
- **Errors/Exceptions**:
  - List of common errors for this endpoint and their meanings.
```