Recapitulare Data Access

Adrian Adiaconitei

LINK Academy

Obiective

- ✓ Baze de date
- ✓ XML
- ✓ JSON

Recapitulare

Structura informatiei dintr-un SGBD relational:

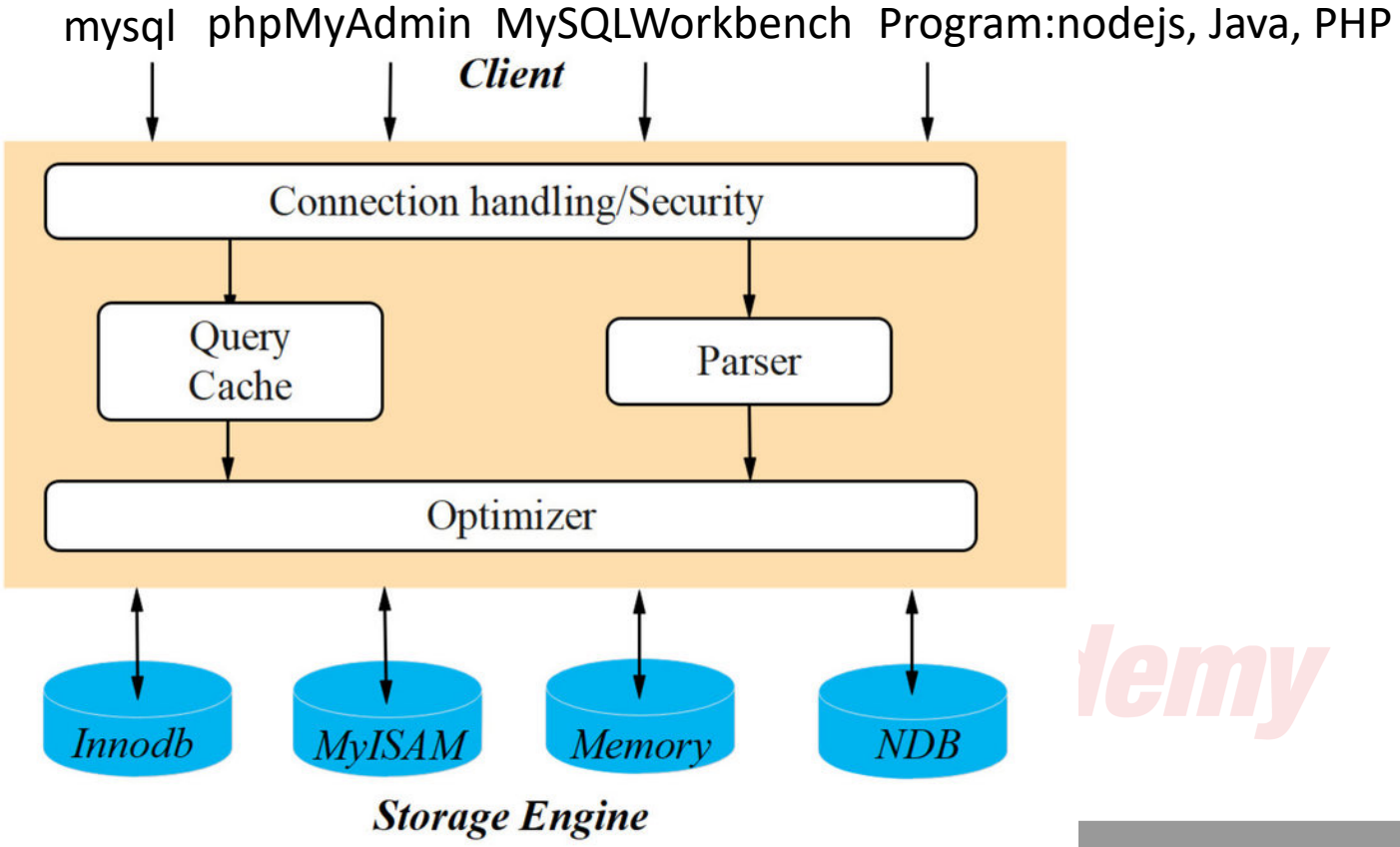
- ✓ **Baza de date**
- ✓ **Tabele** (relatii)
  - coloane(atribute)
  - randuri (inregistrari/entitati)

Limbajul SQL (MySQL)

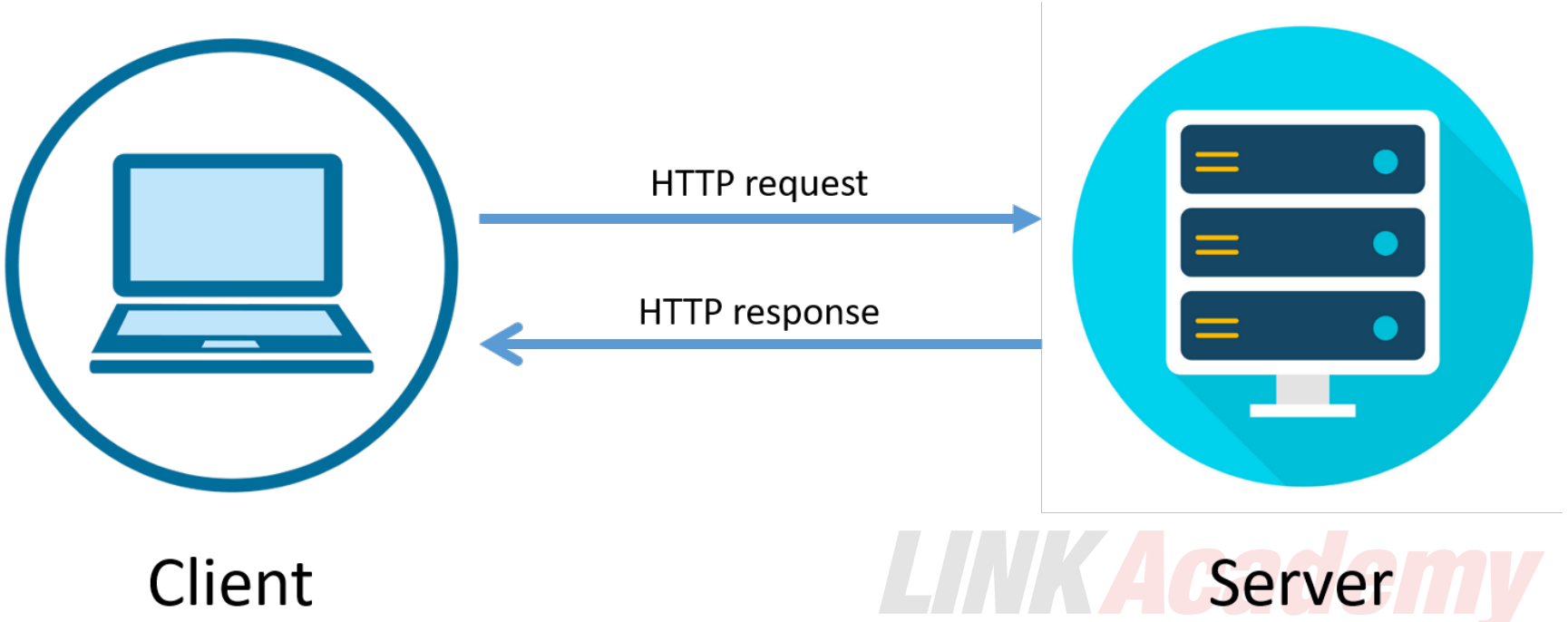
- ✓ **DDL (Data Definition Language)**: creare, modificare, stergere, vizualizare bade de date si tabele; CREATE, ALTER, DROP
- ✓ **DML (Data Manipulation Language)**: manipularea datelor din tabele : INSERT, UPDATE, DELETE
- ✓ **DQL – (Data Query Language):** permit obţinerea datelor din baza de date; **SELECT**
- ✓ **DCL** (**Data Control Language**): permit administratorilor sa controleze accesul la datele din baza de date si folosirea diferitelor privilegii ale sistemului SGBD; **GRANT, REVOKE**

https://downloads.mysql.com/docs/refman-5.7-en.a4.pdf

Limbajul SQL (MySQL)



HTTP Request/Response model



XML

- ✓ Un document/fisier XML are extensia .xml
- ✓ Un document/fisier XML este **Case sensitive**
- ✓ Un document XML este format din:
  - ✓ **marcaje (tag-uri)**
  - ✓ **date caracter**

  Un marcaj (tag) este un sir de caractere delimitat de caracterele "<" si ">".
  Datele caracter reprezinta continutul marcajelor.
- ✓ Un fisier XML cuprinde urmatoarele sectiuni:
  - ✓ **Prolog (optional , <?xml version="1.0" encoding="UTF-8"?>)**
  - ✓ **Definitia tipului de document (optionala, DTD -** un set de reguli ce defineste structura unui fisier XML**)**
  - ✓ **Elementul radacina(**Un document XML are un singur element radacina)

JSON

- ✓ JavaScript Object Notation.
- ✓ Nu îl confundaţi cu obiectul JS.
- ✓ JSON este reprezentarea textuală a obiectelor JS.

Cookies vs Session:

✓ **Cookies**

Un cookie este un fișier text mic care este salvat pe computerul utilizatorului. Dimensiunea maximă este de 4KB. Când un utilizator vizitează pentru prima dată un site web, site-ul trimite pachete de date către computerul utilizatorului sub forma unui cookie.

Informațiile stocate în cookie-uri nu sunt sigure, deoarece sunt păstrate pe partea clientului într-un format text pe care oricine îl poate vedea. Putem activa sau dezactiva cookie-urile în funcție de nevoile noastre..

Cookies vs Session:

✓ **Sesiune**

O sesiune este folosită pentru a salva informații pe server.
Este perioada de timp activă petrecută pe un site. Sesiunea utilizatorului începe când utilizatorul se conectează la o anumită aplicație de rețea și se termină când utilizatorul se deconectează din program sau închide browser-ul. (valorile sesiunii sunt șterse automat) . Trebuie să salvăm valorile în baza de date pentru a le păstra pentru totdeauna. (nu este o limită de mărime)

Valorile sesiunii sunt mult mai sigure, deoarece sunt salvate în formă binară sau criptată și pot fi decodificate doar la server.

LocalStorage:

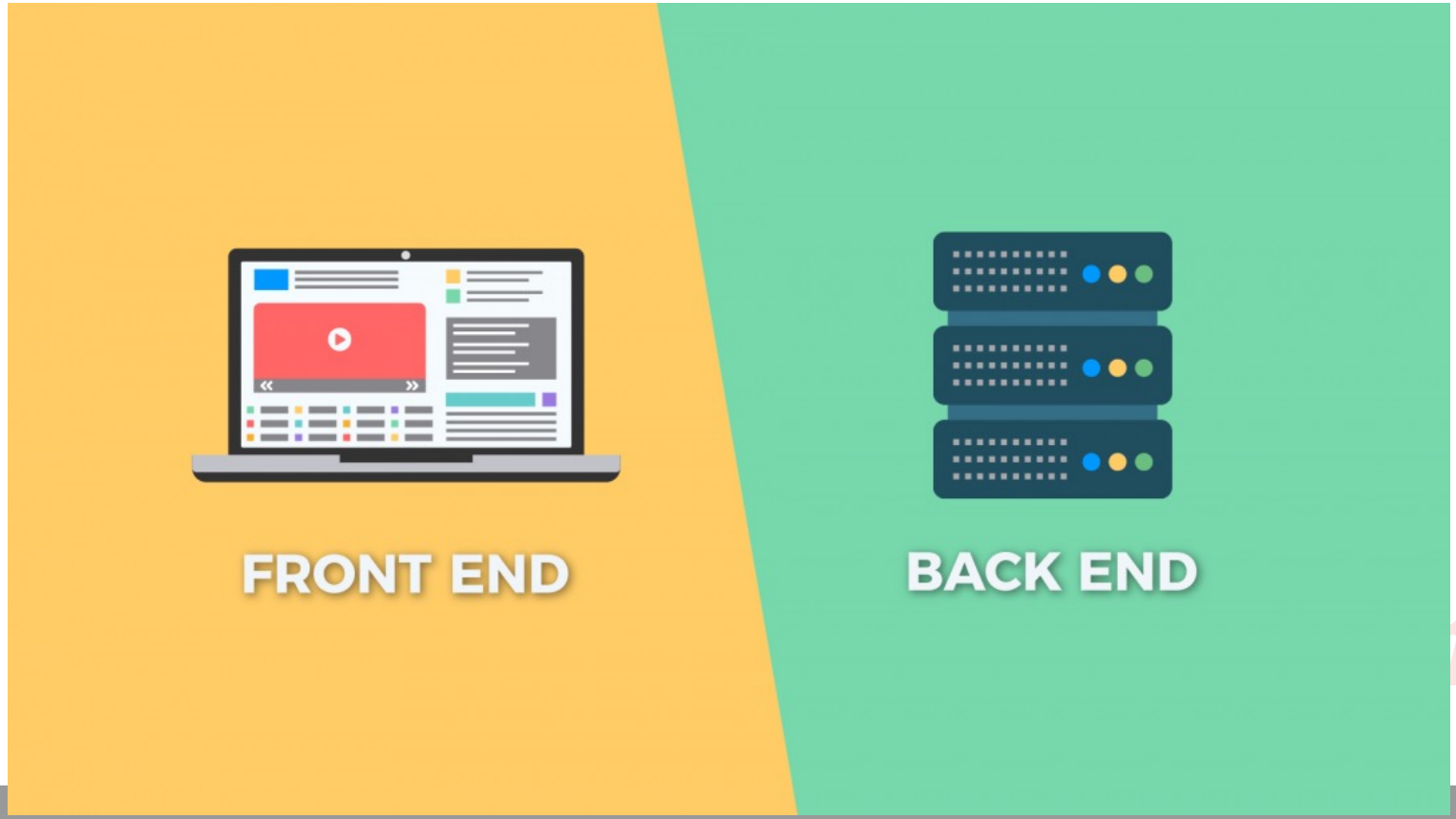
- ✓ **LocalStorage** și **SessionStorage** sunt aproape identice, au același API.
- ✓ Diferența este că, cu **SessionStorage**, datele sunt persistente doar până când fereastra sau fila este închisă.
- ✓ Cu **LocalStorage**, datele sunt păstrate până când utilizatorul șterge manual memoria cache a browserului sau până când aplicația dvs. web șterge datele.
- ✓ **LocalStorage** și **SessionStorage** au aceeași sintaxă.

IndexedDB:

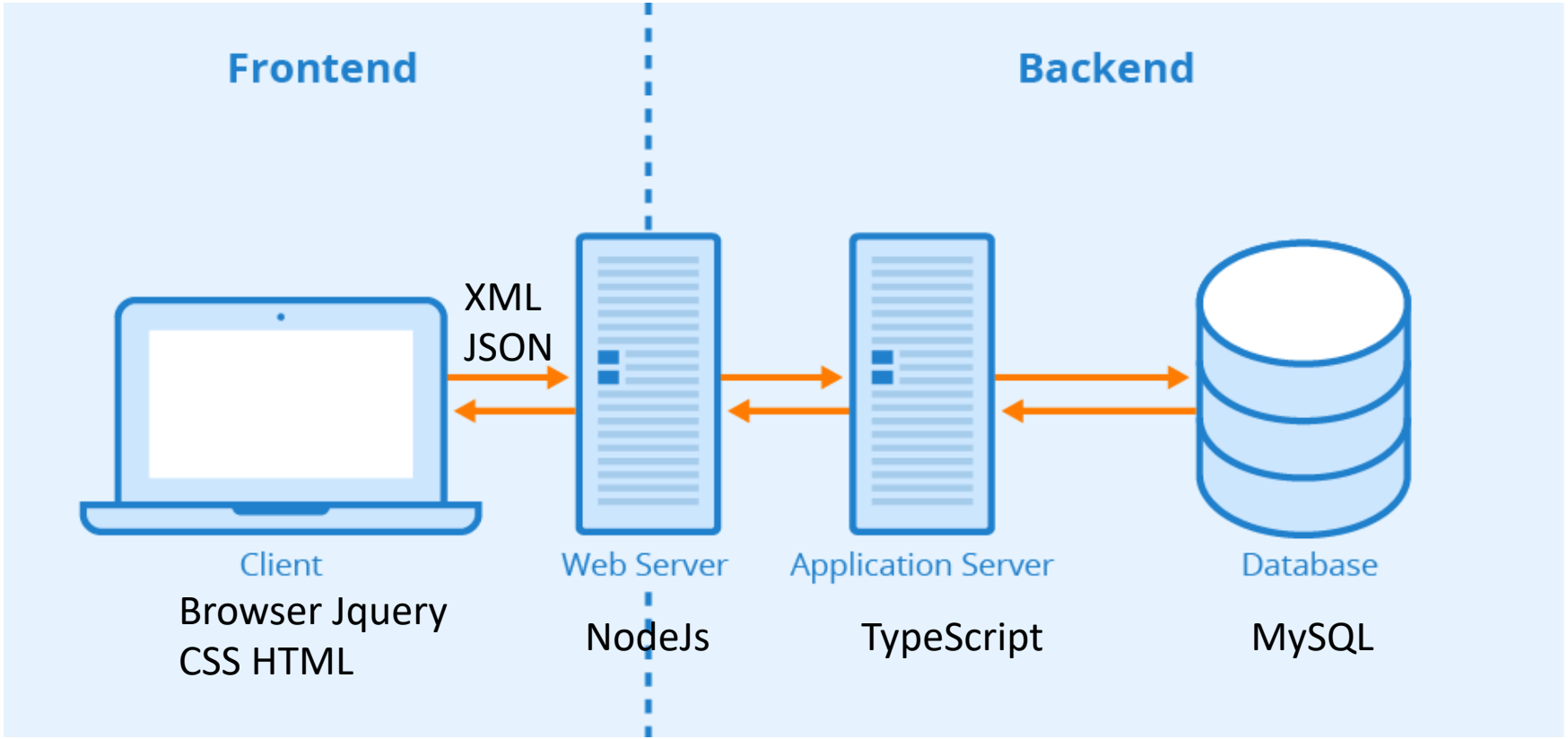
- ✓ IndexedDB vă permite să stocați în mod persistent datele folosind perechi cheie-valoare.
- ✓ Valorile pot fi de orice tip JavaScript, inclusiv boolean, număr, șir, nedefinit, nul, dată, obiect, matrice, regex, blob și fișiere.
- ✓ IndexedDB vă permite să creați aplicații web care pot funcționa atât online, cât și offline.
- ✓ Este util pentru aplicațiile care stochează o cantitate mare de date și nu au nevoie de o conexiune la internet persistentă.
- ✓ Spre deosebire de localStorage și sessionStorage, valorile stocate în IndexedDB pot fi structuri complexe precum obiecte și fișiere.

App – CRUD

App – CRUD



App – CRUD


Frontend
Backend
XML
JSON
Client
Web Server
Application Server
Database
Browser Jquery
CSS HTML
NodeJs
TypeScript
MySQL

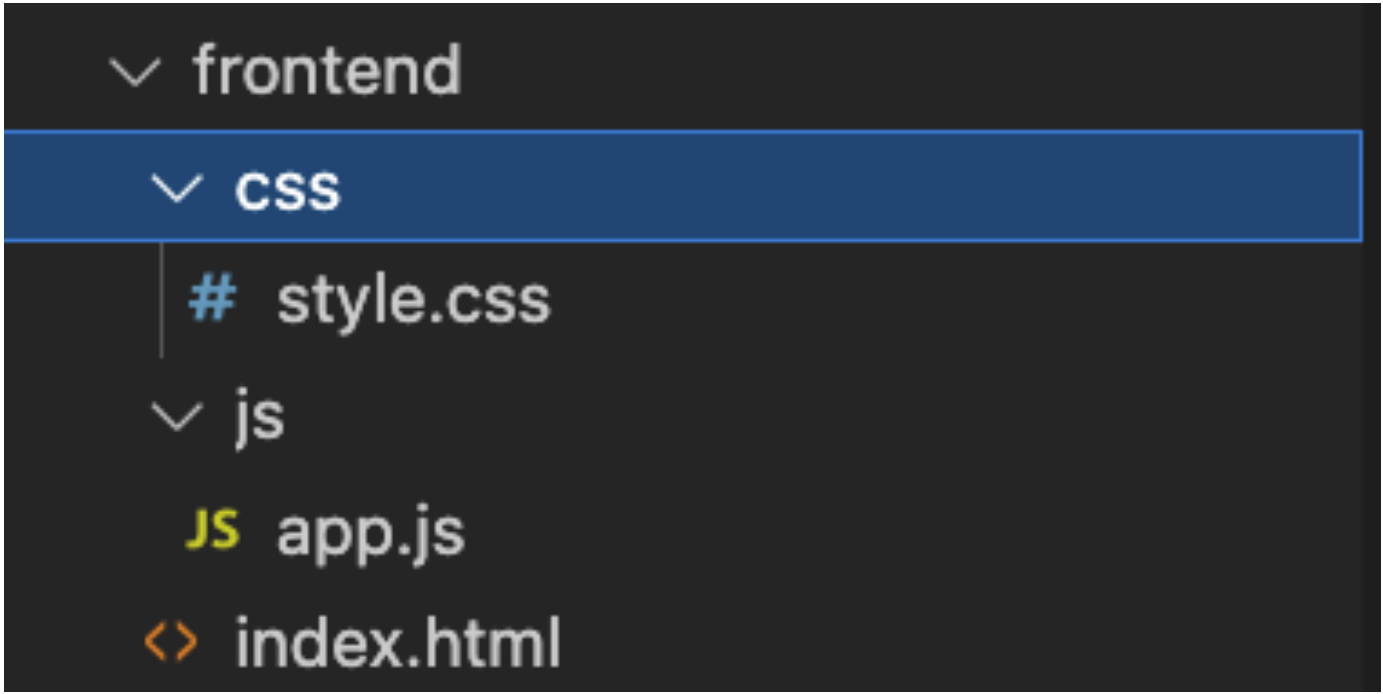
LINKAcademy

App – CRUD

Front-End

App – CRUD

Front-End



App – CRUD

Back-End

1. cd backend/
2. npm init --yes sau npm init –y =>package.json
3. npm install express dotenv cors
4. npm i -D concurrently nodemon
5. npm i -D typescript @types/express @types/node @type/cors

//npm install --dev typescript @types/express @types/node

6. npx tsc --init => tsconfig.json

App – CRUD

Back-End

7. In fisierul tsconfig.json

"compilerOptions": {

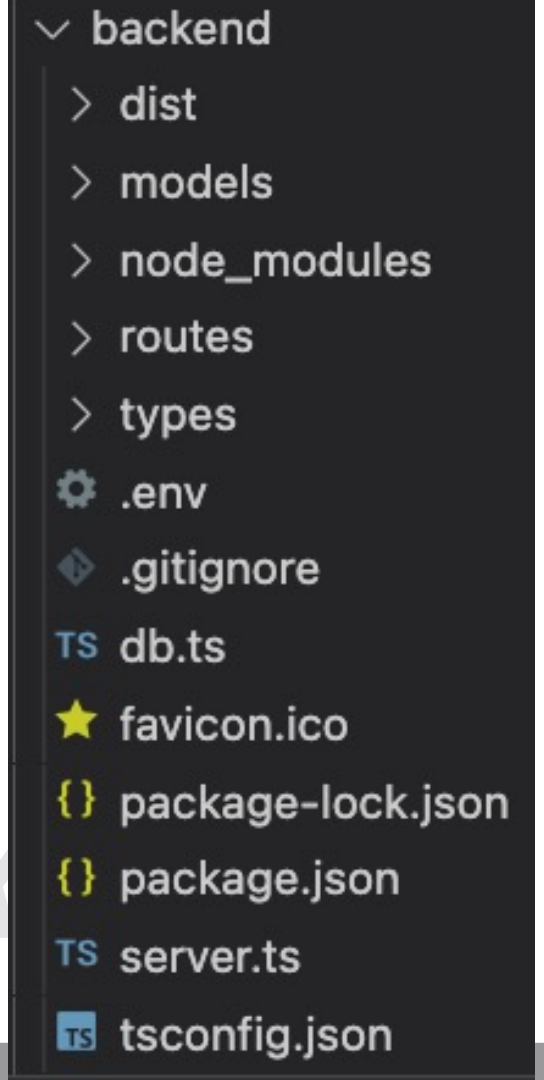
"outDir": ”./dist"

}

8. In fisierul .env

PORT=3002

9. Facem fisierul: server.ts



App – CRUD

Back-End

10. Modificam fisieul package.json

```
"scripts": {
        "build": "npx tsc",
        "start": "node dist/server.js",
        "dev": "concurrently \"npx tsc --watch\" \"nodemon -q
dist/server.js\""
}
```

11. npm run dev

Server is running at https://localhost:3002

App – CRUD

DB: MySQL

1.

Metoda 1: Instalare MySql

Metoda 2: Instalare Xampp

App – CRUD

DB: MySQL

2. Configurare baza de date

3. Folosim baza de date de la Aplicatia1 sau
Rulare fisier :curs7/mycrud/data**/db.sql**

4. Conectare la baza de date

App – CRUD

Back-End+MySql

1. npm install body-parser mysql2
2. npm install --save-dev @types/body-parser @types/mysql @types/dotenv
3. In fisierul .env

DB_HOST="localhost"

DB_USER="username"

DB_PWD="password"

DB_NAME=”dbname"

App – CRUD

Back-End+MySql

4. curs7/mycrud/backend/**db.ts**
5. curs7/mycrud/backend/**types**/User.ts
6. curs7/mycrud/backend/**models**/user.ts
7. curs7/mycrud/backend/**routes**/userRouter.ts
8. Modificam fisierul server.ts
9. npm run dev
10. http://localhost:3002/users

App – CRUD

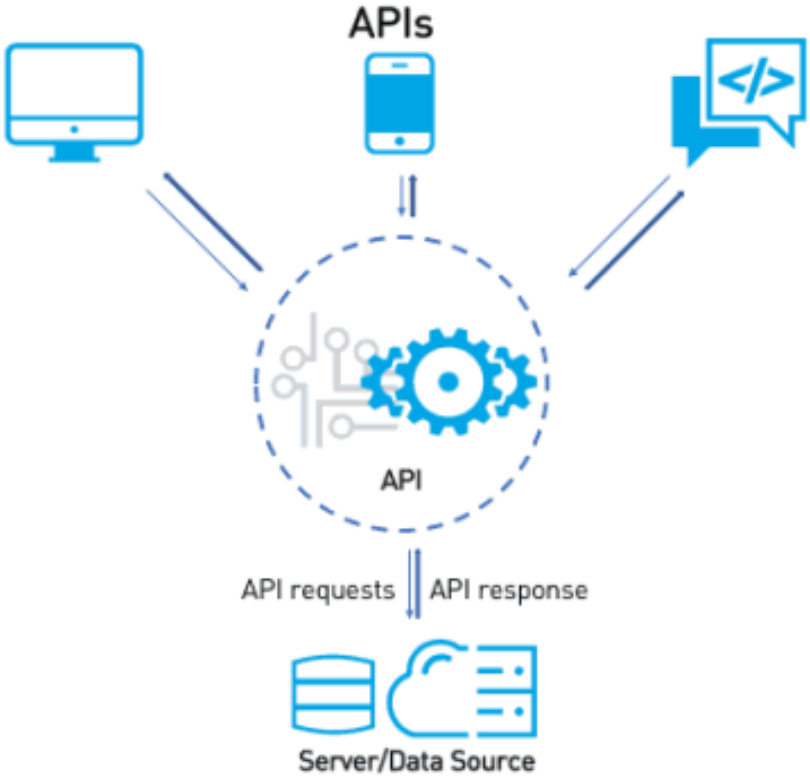
Front-end + Back-End+MySql

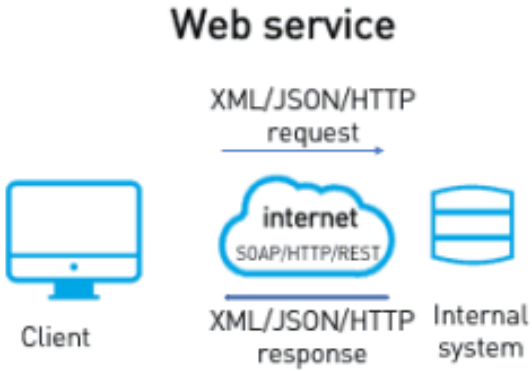
1. Adaugam cereri Ajax in app.js

2. Afisam datele in index.html

App – CRUD

Front-end + Back-End+MySql





AJAX – Suport din partea serverului

```
// CORS
app.use((req, res, next) => {
    res.append('Access-Control-Allow-Origin', ['*']);
    res.append('Access-Control-Allow-Methods',
    'GET,PUT,POST,DELETE');
    res.append('Access-Control-Allow-Headers', 'Content-
    Type');
     next();
});
```

App – CRUD

Front-end + Back-End+MySql

Index	GET	/users	returns list of all items
Show	GET	/users/:id	returns item id
Create	Post	/users	creates a new item
Update	Put/Patch	/users/:id	Updated item ID
Destroy	Delete	/users/:id	Deletes item ID

App – CRUD

Front-end + Back-End+MySql

Modificăm aplicația astfel încât să adăugăm **poza**.

App – CRUD

Front-end

Pas1. modificăm **index.html**

enctype="multipart/form-data"

<input type="file" name="image" />

App – CRUD

Front-end

Pas2. modificăm **app.js**

```
var formData = new FormData(form[0]);
data: formData, //$(this).serialize()
contentType: false, //this is requireded
processData: false, //this is requireded
```

App – CRUD

Back-End

Pas3.

Modul nodeJs necesar:

- ✓ **express-fileupload**

sau

- ✓ multer

npm install express-fileupload

npm i --save-dev @types/express-fileupload

https://www.npmjs.com/package/express-fileupload

App – CRUD

Back-End

Pas4. Modificăm serverul **server.ts**

```
import fileUpload from 'express-fileupload';

// Use the express-fileupload middleware
app.use(fileUpload());

// Log the files to the console
console.log(req.files);
```

App – CRUD

Back-End

Pas4. Salvăm poza pe server / folder, modificăm **userRouter.ts**

```
let fileToUpload:any;
let uploadPath;
fileToUpload = req.files!.poza as UploadedFile; //Object is possibly 'null' or 'undefined'.
const newFileName = `${Date.now()}-_${fileToUpload.name}`;
uploadPath = path.join(__dirname, '..', '/uploads/', newFileName);

console.log(uploadPath);
fileToUpload.mv(uploadPath);
```

App – CRUD

Back-End

Pas5. Modificăm tipul de date **User.ts** și modelul **user.ts**

poza?:string,

newUser[‘poza’] = newFileName;

App – CRUD

Back-End - MySQL

Pas6. Modificam structura tabelei **jsusers** din baza de date, adăugăm câmpul **poza**

App – CRUD

Back-End

Pas7. Modificăm modelul **user.ts** pentru a prelua poza și să o trimită către front-end

App – CRUD

Front-end

Pas8. modificăm **app.js**

```
{
"data": "poza",
render:function (data) {
return '<img src="../backend/dist/uploads/' +data+ '" />';
}
},
```

App – CRUD

Front-end

Pas9. Adăugare css pentru poza

App – CRUD

TODO

- Actualizare formular editare
- Validare fișier primit

https://github.com/avivharuzi/express-fileupload-validator