Servere și Comunicația Client-Server

I. Introducere în Servere

Un server este un sistem de calcul care furnizează servicii sau resurse către alte calculatoare, numite clienți.

Tipuri de servere:

1. Servere web: Livrarea de pagini web.
2. Servere de baze de date: Gestionarea și stocarea datelor.
3. Servere de aplicații: Găzduirea și rularea aplicațiilor.
4. Servere de fișiere: Stocarea și gestionarea fișierelor.

II. Comunicația Client-Server

Modelul client-server: Un client trimite cereri către server, iar serverul răspunde cu resursele sau informațiile solicitate.

- Componente
  - Client: Inițiază cererea (ex: browser web).
  - Server: Procesează cererea și trimite un răspuns.
- Fluxul de lucru
  - Clientul inițiază o conexiune.
  - Serverul acceptă conexiunea.
  - Clientul trimite o cerere.
  - Serverul procesează cererea.
  - Serverul trimite un răspuns.
  - Conexiunea se închide sau se menține deschisă pentru alte cereri.

III. Protocoale

- HTTP/HTTPS: Protocoale de transfer hypertext, folosite pentru paginile web.
- HTTP: Protocol fără securitate.
- HTTPS: Protocol securizat cu criptare.
- FTP: Protocol de transfer fișiere, utilizat pentru transferul de fișiere între client și server.
- SMTP/POP3/IMAP: Protocoale de email.
- SMTP: Trimiterea de emailuri.
- POP3/IMAP: Recepționarea de emailuri.
- TCP/IP: Suite de protocoale fundamentale pentru internet.
- TCP: Asigură transferul de date fiabil.
- IP: Adresează și transmite pachetele de date.

IV. Runtime

Node.js = Un runtime de JavaScript construit pe motorul V8 de la Chrome.

Ce este un runtime?

Un mediu în care codul JavaScript este executat, simulează un mediu de browser pentru a rula codul JavaScript în afara browserului.

Caracteristici

- Asincron și bazat pe evenimente.
- Ideal pentru aplicații scalabile, de tip I/O intensiv.
- Folosit pentru dezvoltarea de aplicații web, API-uri și alte aplicații server-side.

Instalare

- Se descarcă de pe site-ul official.
- Se instalează folosind pachetul potrivit pentru sistemul de operare.

Package Manager: npm (Node Package Manager).

Ce este un Package Manager?

Un instrument care automatizează procesul de instalare, actualizare, configurare și gestionare a pachetelor software.

Funcționalități

- Instalarea de pachete și module.
- Gestionarea dependențelor proiectului.
- Partajarea și descărcarea pachetelor din registrul npm.

Comenzi de bază

- `npm init`: Inițializează un nou proiect Node.js și creează un fișier `package.json`.
- `npm install <nume-pachet>`: Instalează un pachet.
- `npm update`: Actualizează toate pachetele.
- `npm uninstall <nume-pachet>`: Dezinstalează un pachet.

V. Comunicarea Fizică și Logică între Client și Server

Aspecte Fizice

- Cererile sunt trimise la adrese IP specifice (ex: 9.6.45.127).
- Comunicarea se realizează prin intermediul porturilor.

Aspecte Logice

- Cererea clientului călătorește de la aplicația client la adresa IP a serverului.
- Cererea ajunge la server și este procesată.
- Serverul returnează un răspuns către client.
- Comunicarea se bazează pe endpoint-uri și headere pentru a specifica destinația și limbajul utilizat.

VI. Tutorial Aplicatie Server

Cerințe : Node.js, npm

Configurarea Proiectului

1. Creează un director pentru proiect:

```sh
mkdir express-api
cd express-api
```

2. Inițializează un nou proiect Node.js:

```sh
npm init -y
```

3. Instalează Express.js:

```sh
npm install express
```

4. Creează directorul și fișierele necesare:

```sh
mkdir db
touch db/data.json
touch index.js
```

5. Pregătirea Fișierului de Bază de Date

```json
{
  "posts": []
}
```

6. Crearea Aplicației Express și Configurarea Portului

```js
const app = express();
const port = 5000;

app.use(express.json());
```

7. Pune serverul să asculte pe portul configurat:

```js
app.listen(port, () => {
    console.log(`Serverul rulează pe portul ${port}`);
});
```

Funcții pentru Citirea și Scrierea Datelor

```js
function readData() {
    const rawData = fs.readFileSync('db/data.json');
    return JSON.parse(rawData);
}

function writeData(data) {
    fs.writeFileSync('db/data.json', JSON.stringify(data, null, 2));
}
```

Creează un endpoint simplu pentru ruta rădăcină:

```js
app.get("/", (req, res) => {
    res.send("Hello World altceva");
});
```

Endpointuri REST

```js
app.get("/posts", (req, res) => {
    const data = readData();
    if (data.posts !== undefined && data.posts.length > 0) {
        res.status(200).send(data.posts);
    } else {
        res.status(400).send("No data");
    }
});
```

```js
app.get("/posts/:id", (req, res) => {
    const id = req.params.id;
    const data = readData();
    const post = data.posts.find(p => p.id == id);
    if (!post) {
        res.status(404).json("Postarea nu există");
    } else {
        res.status(200).json(post);
    }
});
```

```js
app.post('/posts', (req, res) => {
    const data = readData();
    const newPost = req.body;

    // Adaugă validările aici

    newPost.id = data.posts.length + 1; // ID Auto-increment
    data.posts.push(newPost);
    writeData(data);
    res.status(201).json(newPost);
});
```

```js
app.put('/posts/:id', (req, res) => {
    const { id } = req.params;
    const data = readData();
    const index = data.posts.findIndex(post => post.id === parseInt(id));
    if (index === -1) {
        return res.status(404).json({ error: 'Postarea nu a fost găsită' });
    }

    // Adaugă validările aici

    data.posts[index] = { ...data.posts[index], ...req.body };
    writeData(data);
    res.status(206).json(data.posts[index]);
});
```

```js
app.patch('/posts/:id', (req, res) => {
    const { id } = req.params;
    const data = readData();
    const index = data.posts.findIndex(post => post.id === parseInt(id));
    if (index === -1) {
        return res.status(404).json({ error: 'Postarea nu a fost găsită' });
    }

    Object.assign(data.posts[index], req.body);
    writeData(data);
    res.status(206).json(data.posts[index]);
});
```

```js
app.delete('/posts/:id', (req, res) => {
    const { id } = req.params;
    const data = readData();
    const index = data.posts.findIndex(post => post.id === parseInt(id));
    if (index === -1) {
        return res.status(404).json({ error: 'Postarea nu a fost găsită' });
    }

    data.posts.splice(index, 1);
    writeData(data);
    res.status(204).send();
});
```

Rularea Serverului : **node index.js**

Testarea API-ului : Poți folosi unelte precum Postman sau cURL pentru a testa endpoint-urile.