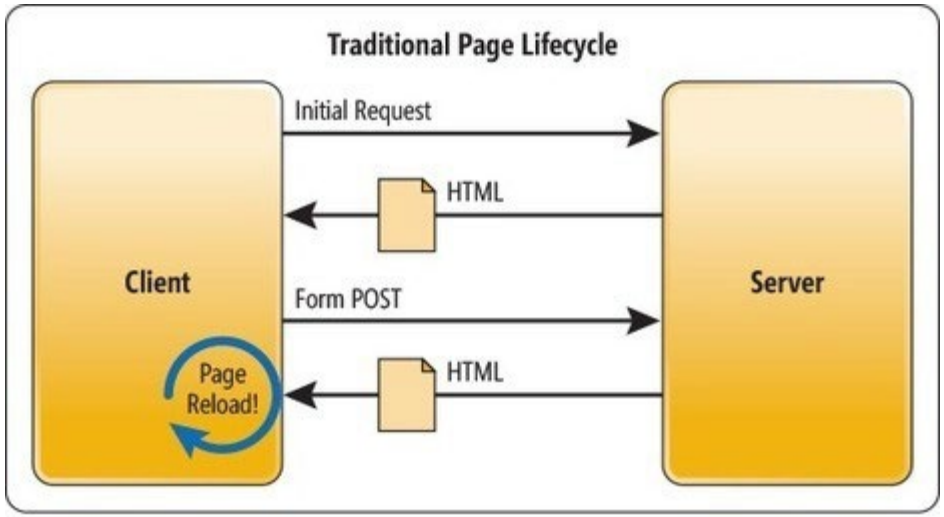
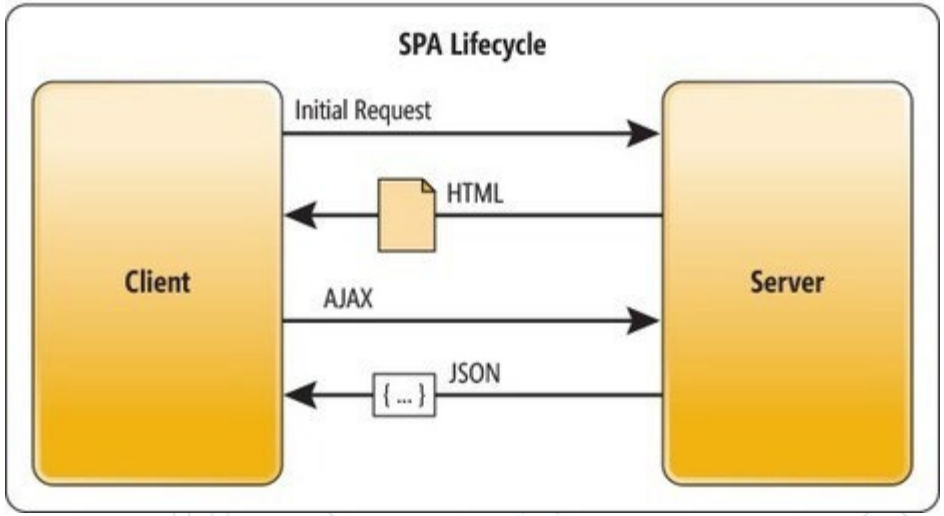
Web e pattern architetturali

Pattern architetturali

Multi-Page Application



Single-Page Application



Backend

Di cosa si occupa il backend (o i backend)

- Rispondere a richieste da parte dei client su protocollo http/https/http2/http3
- Interpretare le URL richieste/header/cookie
- Autenticare un utente
- Autorizzare un utente dopo la sua autenticazione
- Servire contenuti statici
- Generare pagine dinamiche
- Rispondere a chiamate REST da una SPA
- Gestire cache
- Servire contenuti in streaming
- ……

Come fa il backend a rispondere alle richieste?

Semplicemente utilizzando i socket ed i metodi di listen

https://docs.microsoft.com/it-it/dotnet/framework/network-programming/synchronous-server-socket-example

```
// Create a TCP/IP socket.
Socket listener = new Socket(ipAddress.AddressFamily,
    SocketType.Stream, ProtocolType.Tcp );

// Bind the socket to the local endpoint and
// listen for incoming connections.
try {
    listener.Bind(localEndPoint);
    listener.Listen(10);

    // Start listening for connections.
    while (true) {
        Console.WriteLine("Waiting for a connection...");
        // Program is suspended while waiting for an incoming connection.
        Socket handler = listener.Accept();
        data = null;

        // An incoming connection needs to be processed.
        while (true) {
            int bytesRec = handler.Receive(bytes);
            data += Encoding.ASCII.GetString(bytes,0,bytesRec);
            if (data.IndexOf("<EOF>") > -1) {
                break;
            }
        }

        // Show the data on the console.
        Console.WriteLine( "Text received : {0}", data);

        // Echo the data back to the client.
        byte[] msg = Encoding.ASCII.GetBytes(data);

        handler.Send(msg);
        handler.Shutdown(SocketShutdown.Both);
        handler.Close();
    }

} catch (Exception e) {
    Console.WriteLine(e.ToString());
}
```

https://gist.github.com/tedmiston/5935757

```
var net = require('net');

var server = net.createServer(function(socket) {
        socket.write('Echo server\r\n');
        socket.pipe(socket);
});

server.listen(1337, '127.0.0.1');
```

Ma devo implementarmi il protocollo HTTP?

Come fa il backend a rispondere alle richieste con express?

https://expressjs.com/en/starter/hello-world.html


```
const express = require('express' 4.17.1 )
const app = express()
const port = 3000

app.get('/', (req, res) => res.send('Hello World!'))

app.listen(port, () => console.log(`Example app listening on port ${port}!`))
```

Save on **RunKit** | Node 10 | help | URL: **https://jt9ee7g2hkau.runkit.sh**

Come restituire un file html

```
//assuming app is express Object.
app.get('/',function(req,res) {
  res.sendFile('index.html');
});
```

Routing: Interpretare le URL richieste

Il routing è responsabile del mapping degli URI di richiesta agli endpoint e dell'invio di richieste in ingresso a tali endpoint. Le route sono definite e configurate all'avvio.

Metodi di route

Un metodo di route deriva da uno dei metodi HTTP ed è collegato ad un'istanza delle classe `express`.

Il codice seguente è un esempio di route definite per i metodi GET e POST nella root dell'app.

```
// GET method route
app.get('/', function (req, res) {
  res.send('GET request to the homepage');
});

// POST method route
app.post('/', function (req, res) {
  res.send('POST request to the homepage');
});
```

Routing con parametri:

```
8
9   app.get('/contact', function(req, res){
10    res.send('this is the contact page');
11  });
12
13  app.get('/profile/:id', function(req, res){
14    res.send('You request get(key: ?) ofile with the id of ' + req.params.id);
15  });
16
17  app.listen(3000);
```