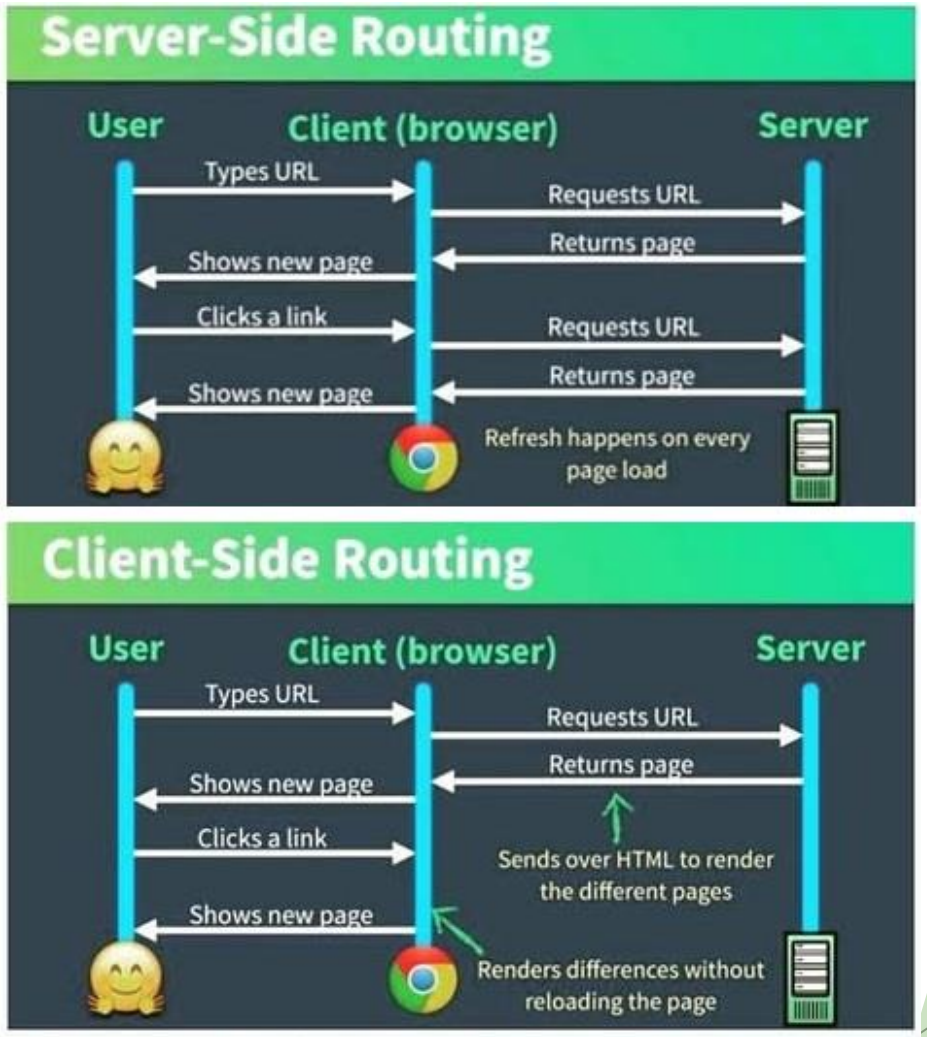
React Router

Intelligenza Artificiale e Tecnologie Web - Laboratorio



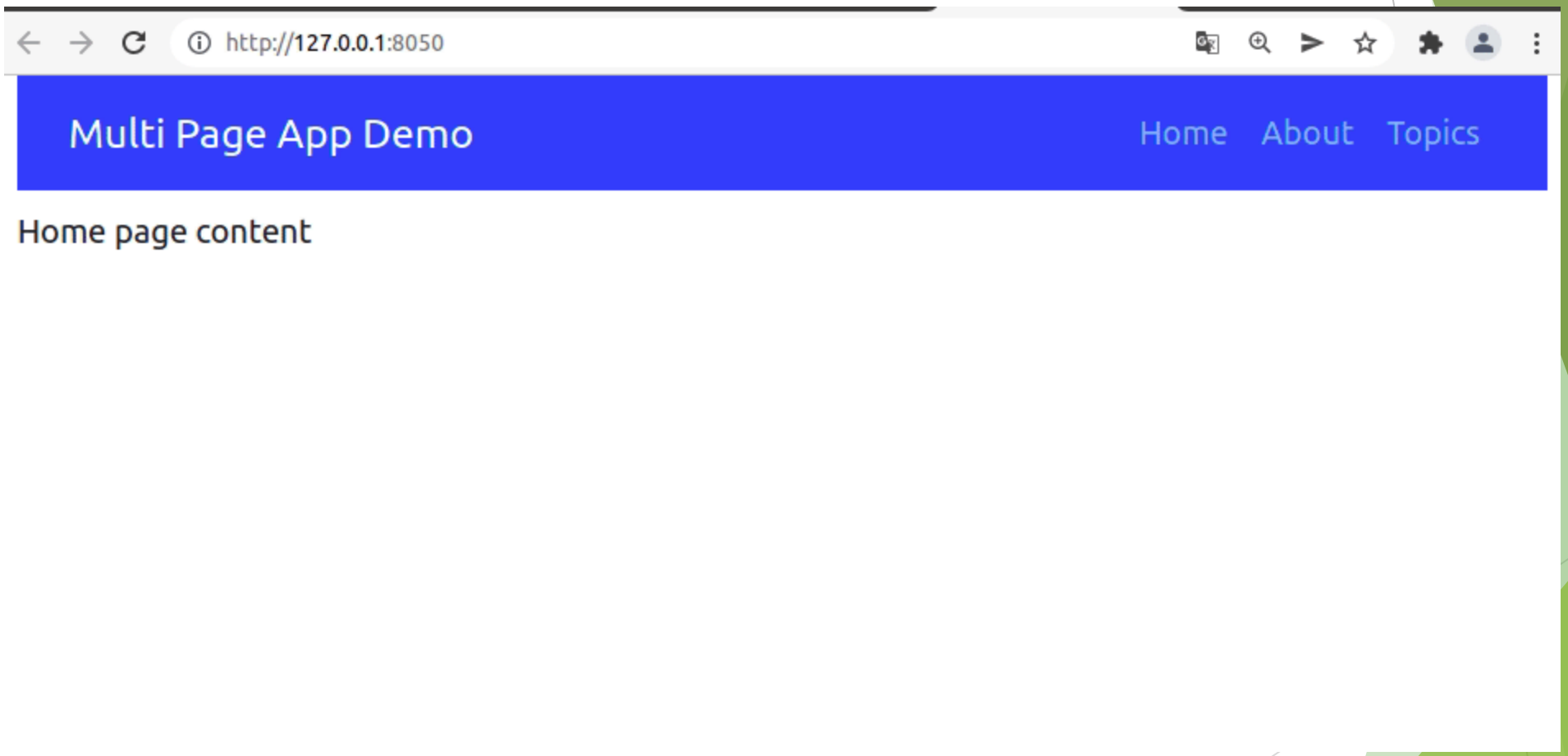
Web Routing

- Esistono due varianti del routing nelle applicazioni web:
  - Il **routing tradizionale lato server** comporta l'invio di richieste al server per ogni nuova sezione del sito, con conseguenti ricaricamenti completi della pagina.
  - Il **routing lato client**, invece, gestisce i cambi di sezione direttamente nel browser, senza dover effettuare richieste al server.

Routing lato client

- In React, il routing lato client funziona così:
  - **Gestione dell'URL:** React Router intercetta cambi di URL e aggiorna l'interfaccia grafica appositamente
  - **Preservazione dello stato**: l'applicazione mantiene lo stato durante la navigazione.
  - **Aggiornamenti istantanei:** le transizioni da una sezione all'altra di un'applicazione avvengono istantaneamente senza ritardi.

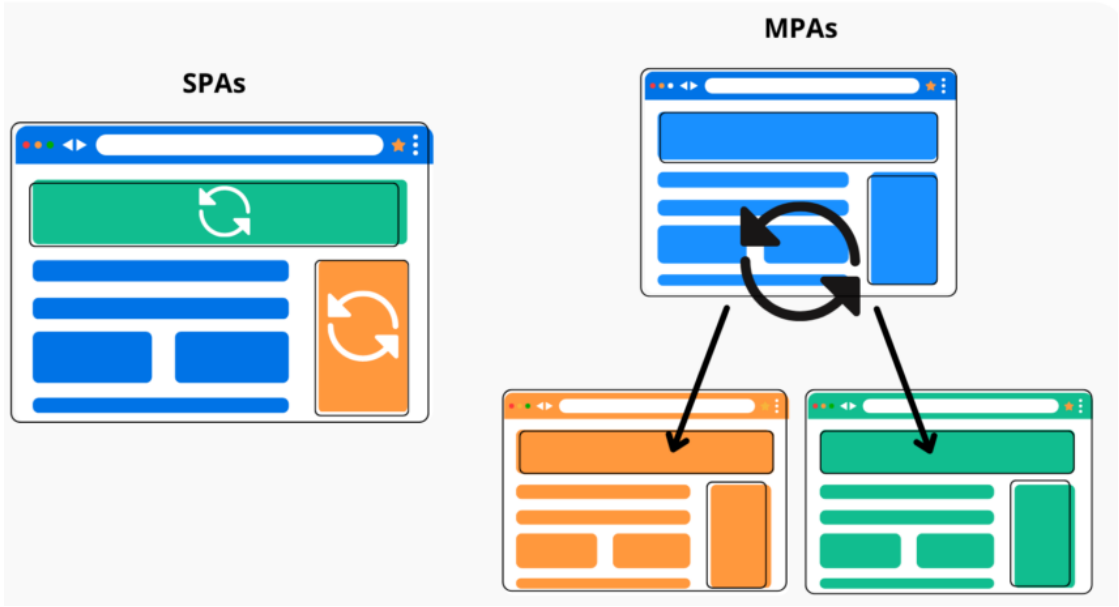
Routing lato client

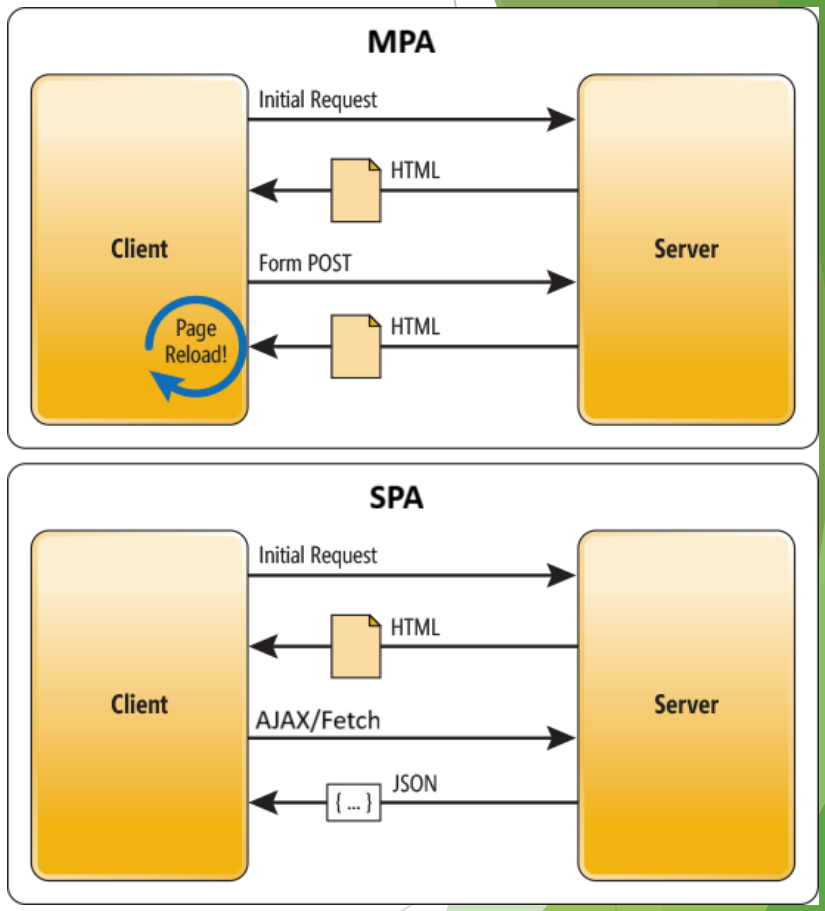


Single-Page application (SPA)

- React consente l'implementazione delle Single-Page application, caratterizzate dai seguenti aspetti:
  - **Caricamento iniziale**: il browser carica tutto il codice HTML, JavaScript e CSS con la singola richiesta iniziale di accesso all'applicazione.
  - **Aggiornamento dinamico del codice HTML:** quando l'utente naviga verso uno URL, il codice HTML associato ad esso viene dinamicamente riscritto sostituendo quello associato all'URL di partenza.
  - **Scambio asincrono dei dati**: lo scambio dei dati con il server avviene in modo asincrono.

Single-Page application vs Multi-Page Application





Il routing client-side e lo scambio di dati in modo asincrono rendono superflui i ricaricamenti completi durante il ciclo di vita dell'applicazione. Invece attraversare più pagine HTML, è l'HTML che viene caricato e sostituito dinamicamente all'interno della stessa pagina (da qui il nome di single-page application).

React Router

Route: serve ad associare un percorso URL a un componente React specifico, il quale verrà renderizzato quando l'utente naviga verso un determinato URL.

Link: serve a creare collegamenti ipertestuali che permettono di navigare all'interno di un'applicazione React senza ricaricare l'intera pagina.

Routes: è il contenitore che raggruppa le diverse definizioni di route e gestisce il rendering del componente corrispondente all'URL corrente. Esso segna l'area specifica della pagina in cui verrà visualizzata la componente selezionata in base all'URL di navigazione.

BrowserRouter: è il componente fondamentale che avvolge l'intera applicazione o le sezioni destinate alla navigazione e che abilita il routing lato client. Esso usa l'HTML5 History API per aggiornare l'URL e sincronizzare l'interfaccia grafica di conseguenza. Quando viene cliccato un *<Link>* o digitato un nuovo indirizzo, BrowserRouter "ascolta" il cambiamento e si assicura che il componente corretto venga renderizzato.

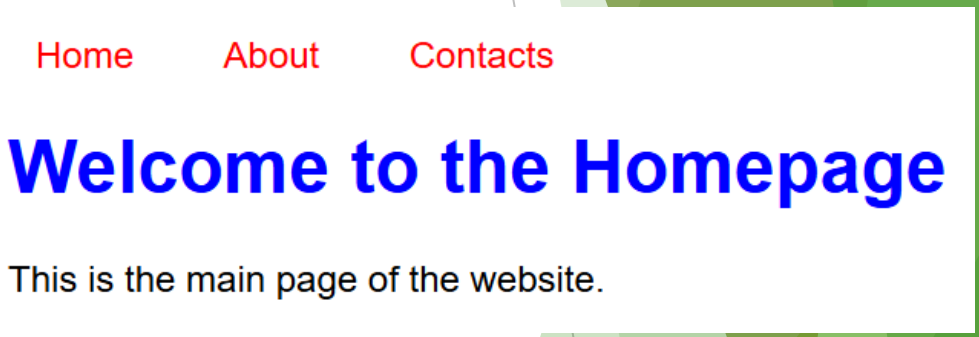
React Router

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Flask React SPA</title>

    <link rel="stylesheet" href="static/style.css">
    <script type="importmap">
        {
          "imports": {
            "react": "https://esm.sh/react@19?dev",
            "react-dom/client": "https://esm.sh/react-dom@19/client?dev",
            "react-router-dom": "https://esm.sh/react-router-dom@7?dev",
            "react-router": "https://esm.sh/react-router@7?dev"
          }
        }
    </script>
    <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
</head>
```

```jsx
import {
    BrowserRouter,
    Routes,
    Route,
    Link,
} from "react-router-dom";

function App() {
    return (
    <BrowserRouter>
        <div>
        <nav>
            <ul className="nav-horizontal">
            <li><Link to="/">Home</Link></li>
            <li><Link to="/about">About</Link></li>
            <li><Link to="/contacts">Contacts</Link></li>
            </ul>
        </nav>
        <Routes>
            <Route path="/" element={<Home />} />
            <Route path="/about" element={<About />} />
            <Route path="/contacts" element={<Contacts />} />
        </Routes>
        </div>
    </BrowserRouter>
    );
}
```

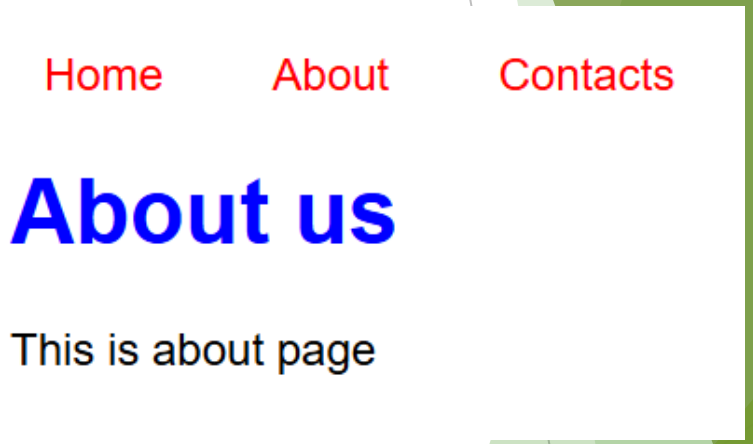
React Router

```
function App() {
    return (
    <BrowserRouter>
        <div>
        <nav>
            <ul className="nav-horizontal">
            <li><Link to="/">Home</Link></li>
            <li><Link to="/about">About</Link></li>
            <li><Link to="/contacts">Contacts</Link></li>
            </ul>
        </nav>
        <Routes>
            <Route path="/" element={<Home />} />
            <Route path="/about" element={<About />} />
            <Route path="/contacts" element={<Contacts />} />
        </Routes>
        </div>
    </BrowserRouter>
    );
}
```



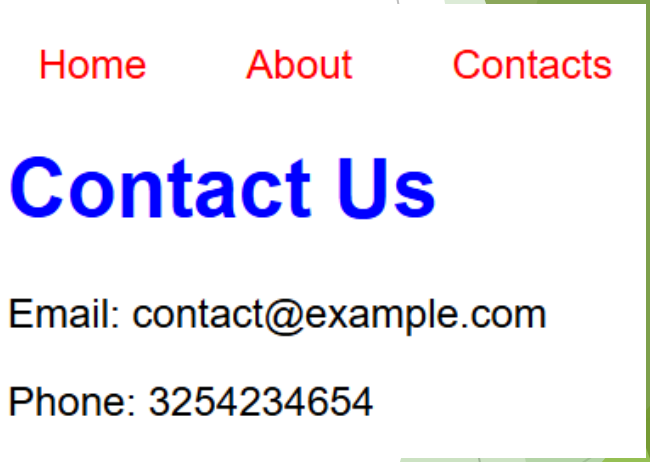
React Router

```
function App() {
    return (
    <BrowserRouter>
        <div>
        <nav>
            <ul className="nav-horizontal">
            <li><Link to="/">Home</Link></li>
            <li><Link to="/about">About</Link></li>
            <li><Link to="/contacts">Contacts</Link></li>
            </ul>
        </nav>
        <Routes>
            <Route path="/" element={<Home />} />
            <Route path="/about" element={<About />} />
            <Route path="/contacts" element={<Contacts />} />
        </Routes>
        </div>
    </BrowserRouter>
    );
}
```



React Router

```
function App() {
    return (
    <BrowserRouter>
        <div>
        <nav>
            <ul className="nav-horizontal">
            <li><Link to="/">Home</Link></li>
            <li><Link to="/about">About</Link></li>
            <li><Link to="/contacts">Contacts</Link></li>
            </ul>
        </nav>
        <Routes>
            <Route path="/" element={<Home />} />
            <Route path="/about" element={<About />} />
            <Route path="/contacts" element={<Contacts />} />
        </Routes>
        </div>
    </BrowserRouter>
    );
}
```



Esercizio 1

- Trasformare il sito in una single-page application:
  - Trasformare pagine in componenti React.
  - Impostare un routing lato client per navigare tra le componenti.

```python
from flask import Flask, render_template

app = Flask(__name__)

@app.route('/')
def home():
    return render_template('home.html')

@app.route('/about')
def about():
    return render_template('about.html')

@app.route('/contacts')
def contacts():
    return render_template('contacts.html')

if __name__ == '__main__':
    app.run(debug=True)
```

Routing Dinamico

- Il Routing dinamico permette di gestire dei pattern variabili negli URL, in base agli elementi che compaiono stessi degli URL.

- Utili per creare i route flessibili e data-driven che rispondono alle interazioni con gli utenti per visualizzare un contenuto specifico.

- React Router introduce i Dynamic Segments, parmetri URL che catturano i valori dal path URL, definiti nel seguente modo:

  - */url_path/:<nome_variabile>*

Routing Dinamico

```
import { useParams } from "react-router-dom";

function ProductDetail(){
  const params = useParams();
  const productCode = params.productCode;

  return (
    <div>
      <h2>Product details</h2>
      <p>Viewing product: {productCode}</p>
    </div>
  );
}

function App() {
        return (
        <BrowserRouter>
          <Routes>
            <Route path="/products" element={<ProductsTable />} />
            <Route path="/products/:productCode" element={<ProductDetail />} />
          </Routes>
        </BrowserRouter>
        );
}
```

Il parametro nel percorso cattura qualsiasi valore in quella posizione dell'URL e l'hook *useParams* lo rende disponibile all'interno della componente.

Esercizio 2

- Nella pagina *index_react.html* è implementata la componente *ProductsTable* che visualizza la seguente tabella, associata alla route */products*.

Products

ID	Title	Price
P001	Smartphone XYZ	699
P002	Notebook Ultra X	1299
P003	Bicicletta da Corsa Pro Sport	799
P004	Scarpe Running FastTrack	120
P005	Lavatrice EcoWash 3000	499

- Implementare la componente *ProductsDetail* che visualizza in dettaglio le informazioni e di ogni componente. Associarla alla route */products/:productCode*.

- Al click di una riga della tabella, deve essere visualizzata la componente *ProductsDetail* con i dati del prodotto cliccato.

Esercizio 2

- Esempio di visualizzazione:

Product Detail

ID: P001
Title: Smartphone XYZ
Price: 699
Description: Smartphone di ultima generazione con fotocamera avanzata e display AMOLED.
Back to Products List