Programmation Web – Avancé

JavaScript & Node.js

Introduction aux SPAs et à leurs communications

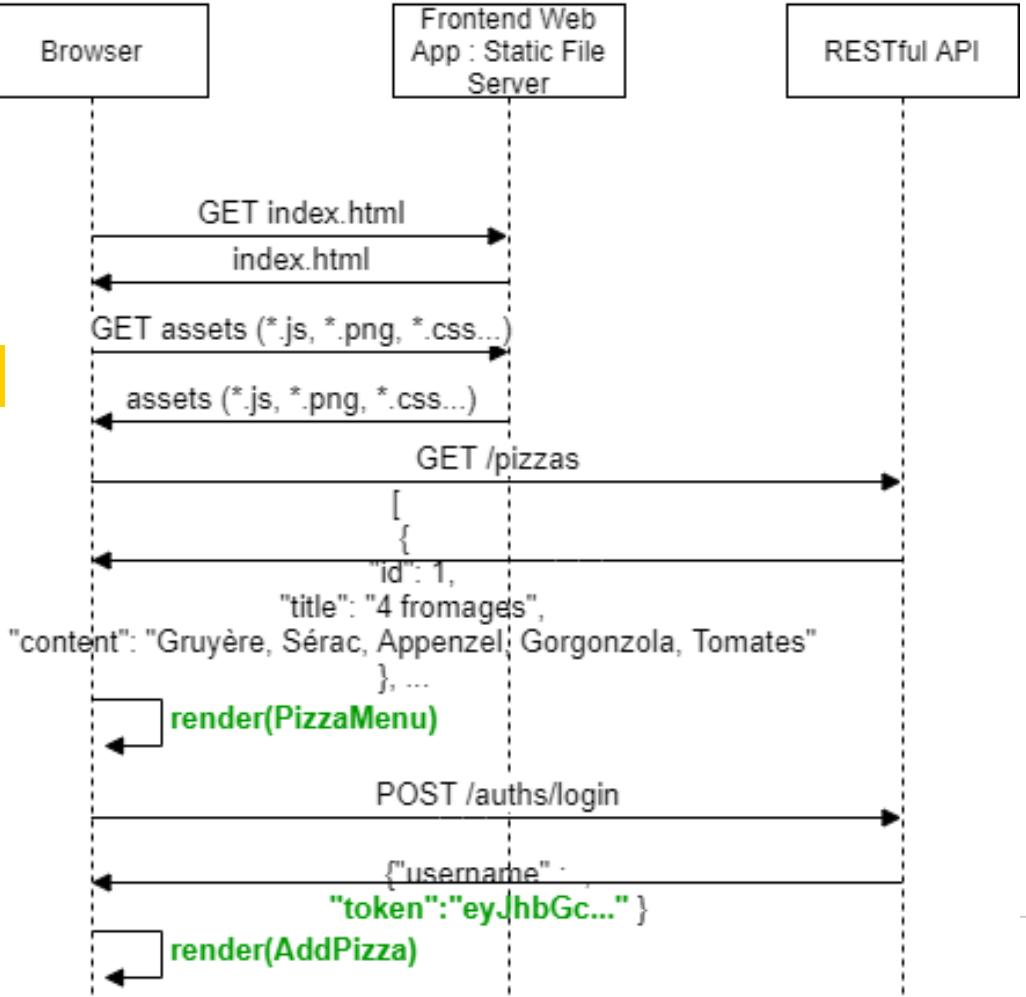
Introduction aux Single Page Applications (SPAs)

Et si l'on chargeait un contenu de façon asynchrone...

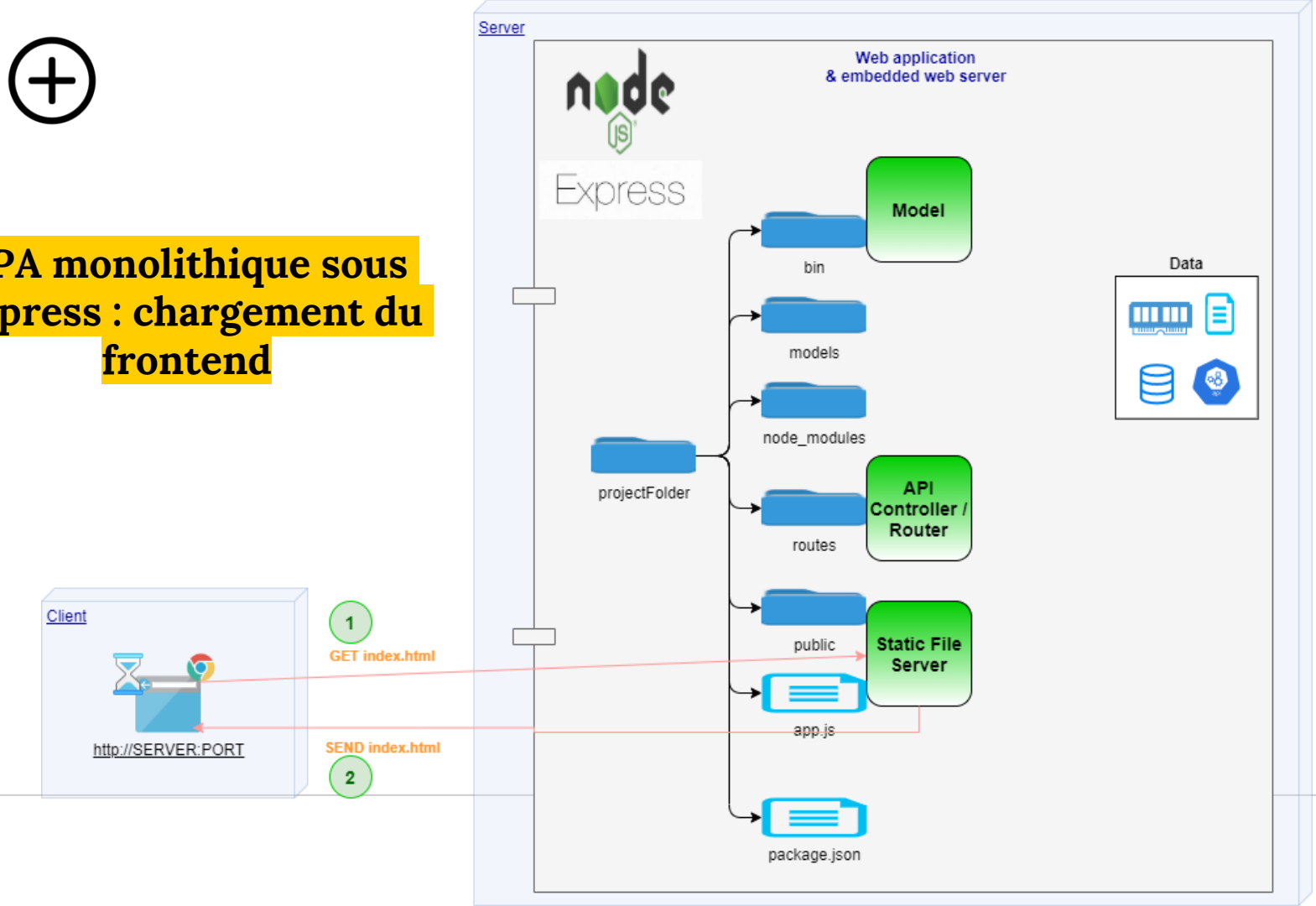
Qu’est-ce qu’une SPA ?

- Pas de rechargement de page
- Réécriture dynamique
- Expérience utilisateur augmentée

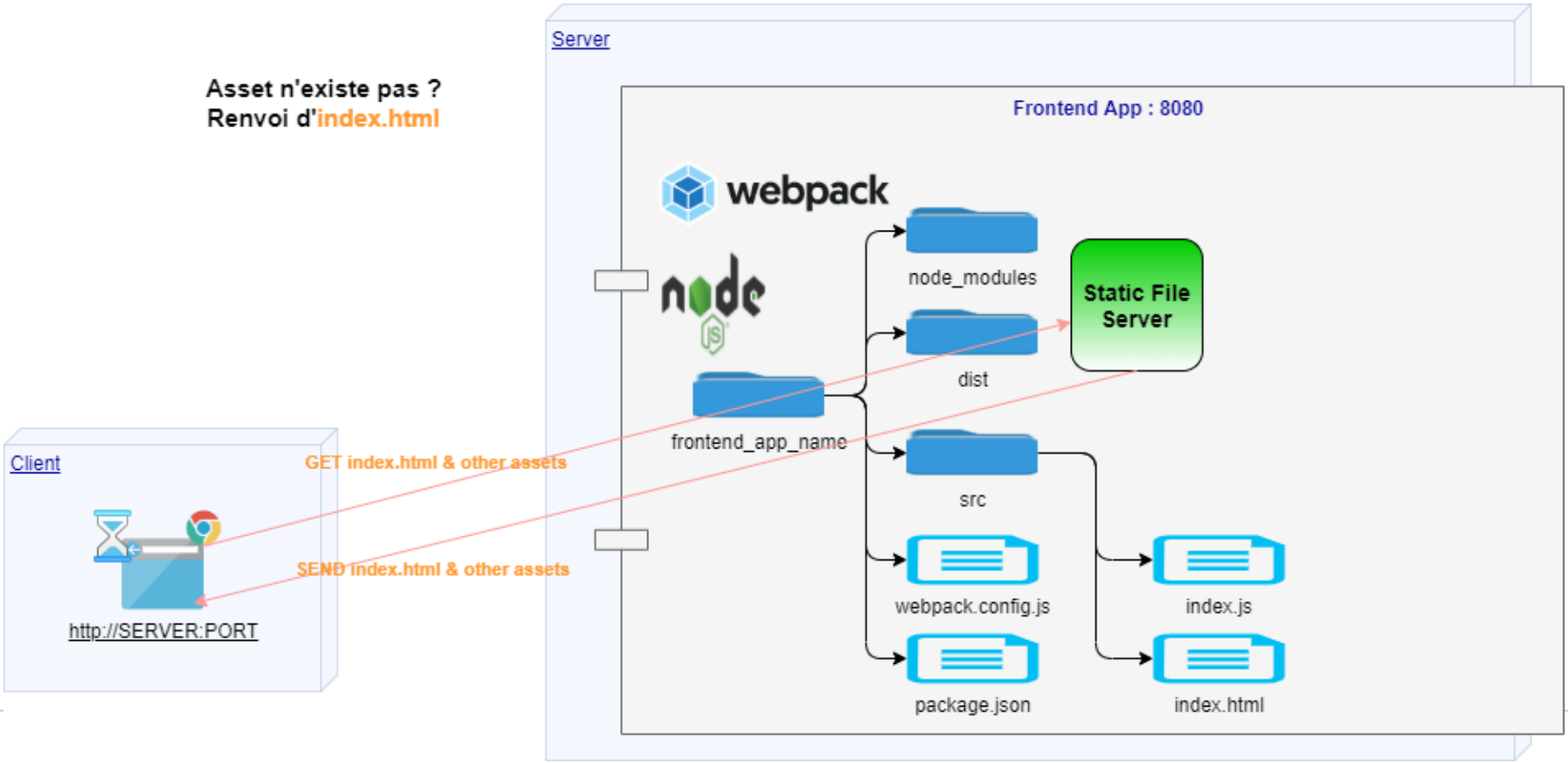
Exemple de Workflow classique d'une SPA



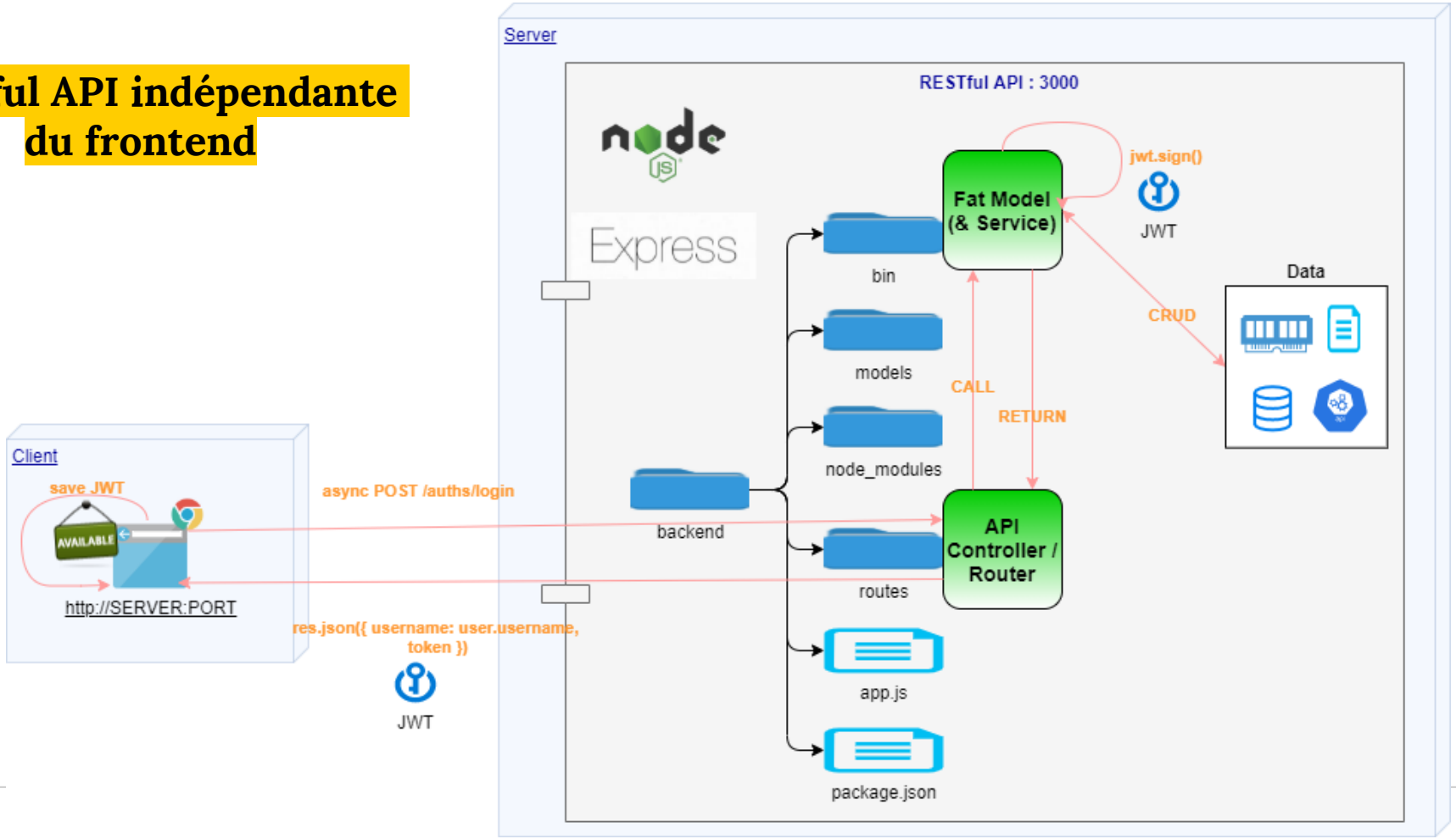
SPA monolithique sous Express : chargement du frontend



Frontend indépendant de la RESTful API



RESTful API indépendante du frontend



Plusieurs architectures de SPAs

- SPA monolithique :
  - RESTful API avec Serveur de fichiers statiques
  - Frontend déployé dans la RESTful API
  - Même serveur pour backend & frontend
  - => même ports pour Frontend & Backend
- SPA dont frontend est indépendant du backend
  - Frontend avec Serveur de fichier statiques
  - 2 serveurs, 1 pour le frontend, 1 pour le backend
  - => Ports différents pour Frontend & Backend

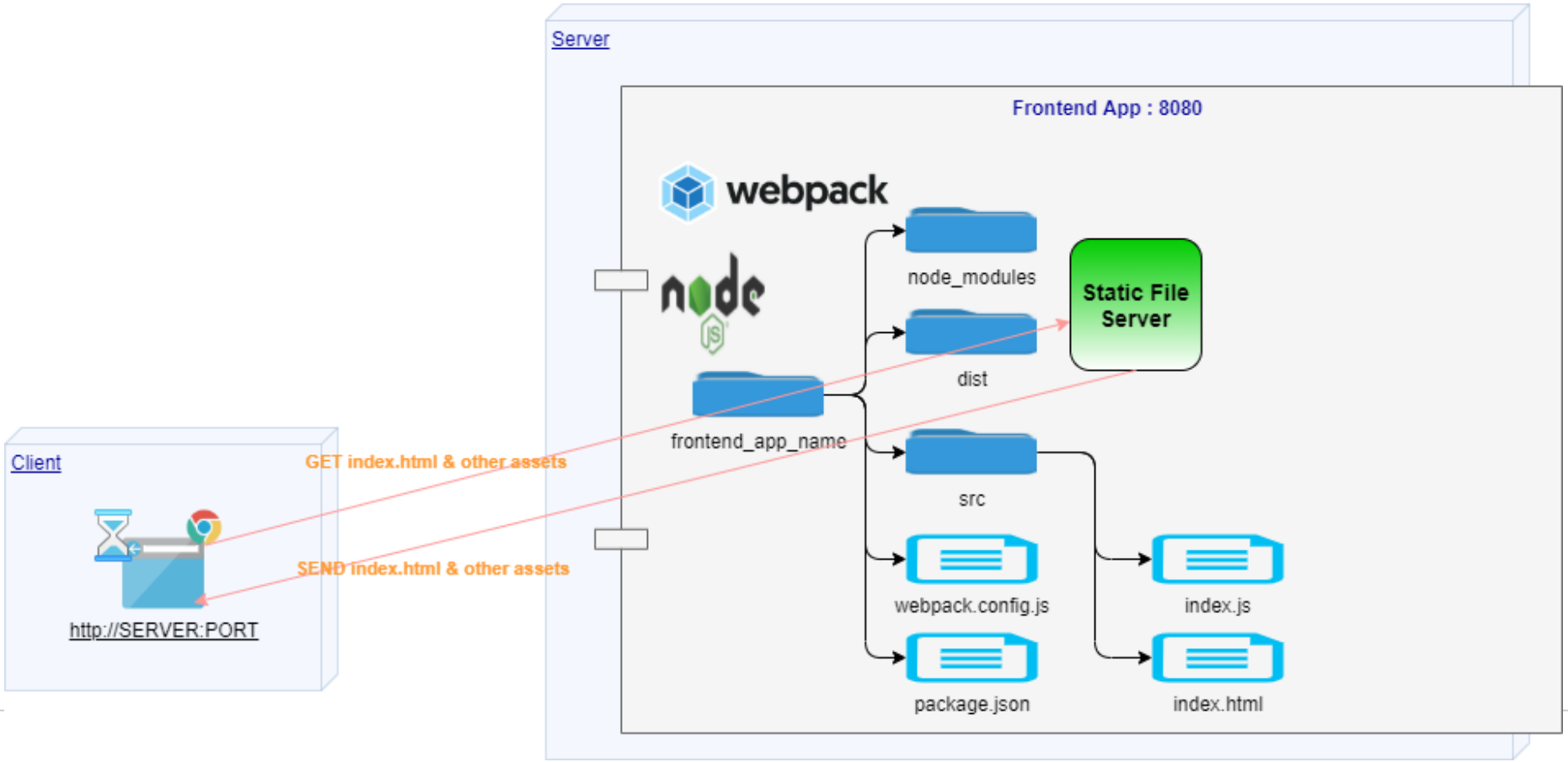
SPA monolithique
VS
SPA frontend / backend indépendants

- Qu'est-ce qui vous semble le plus intéressant ?

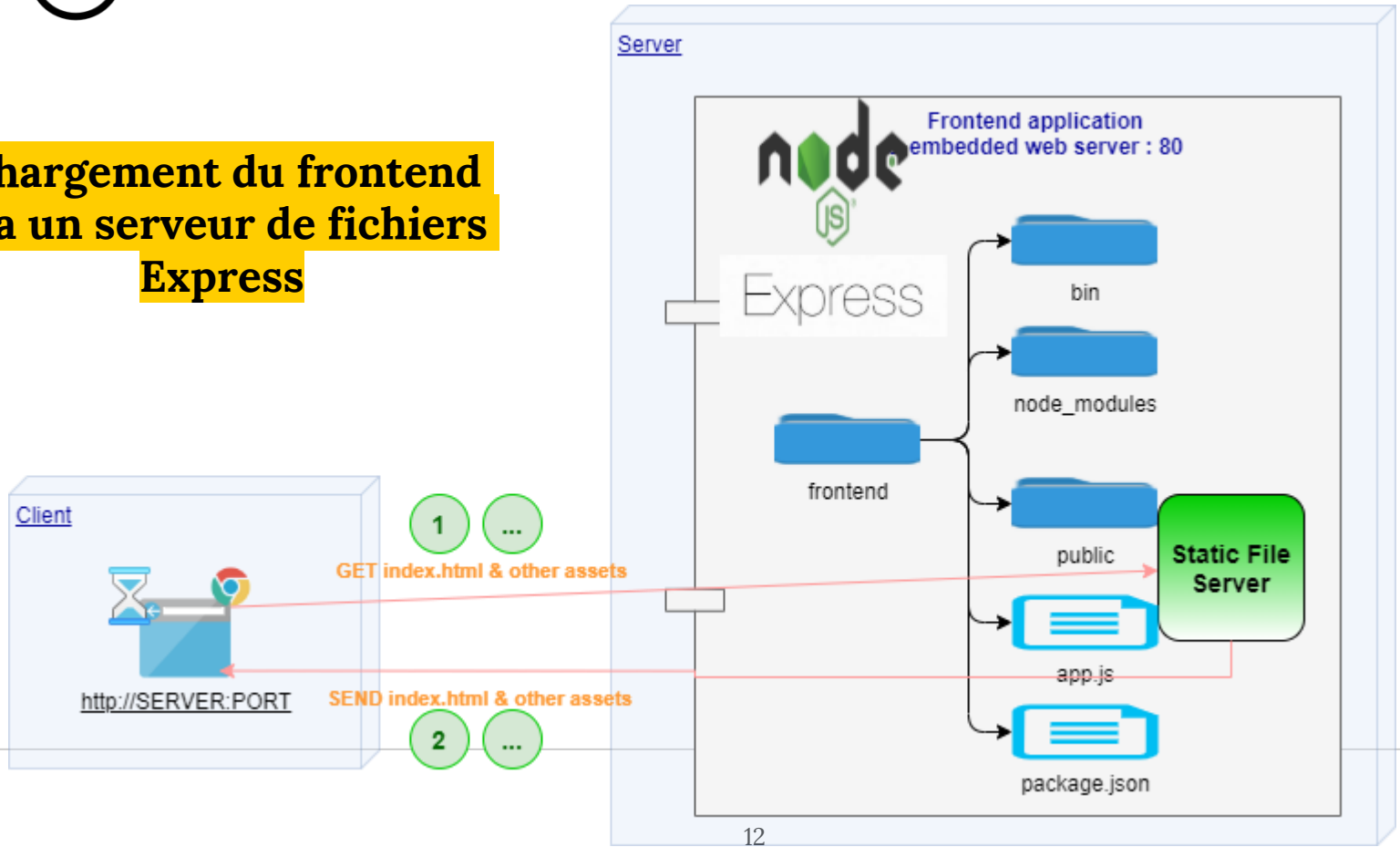
Création d'une SPA avec frontend indépendant du backend

Que se passe-t-il si mon frontend interroge une API externe ?
Pourquoi ça ne marche pas...

Architecture pour ce cours : frontend indépendant via Webpack

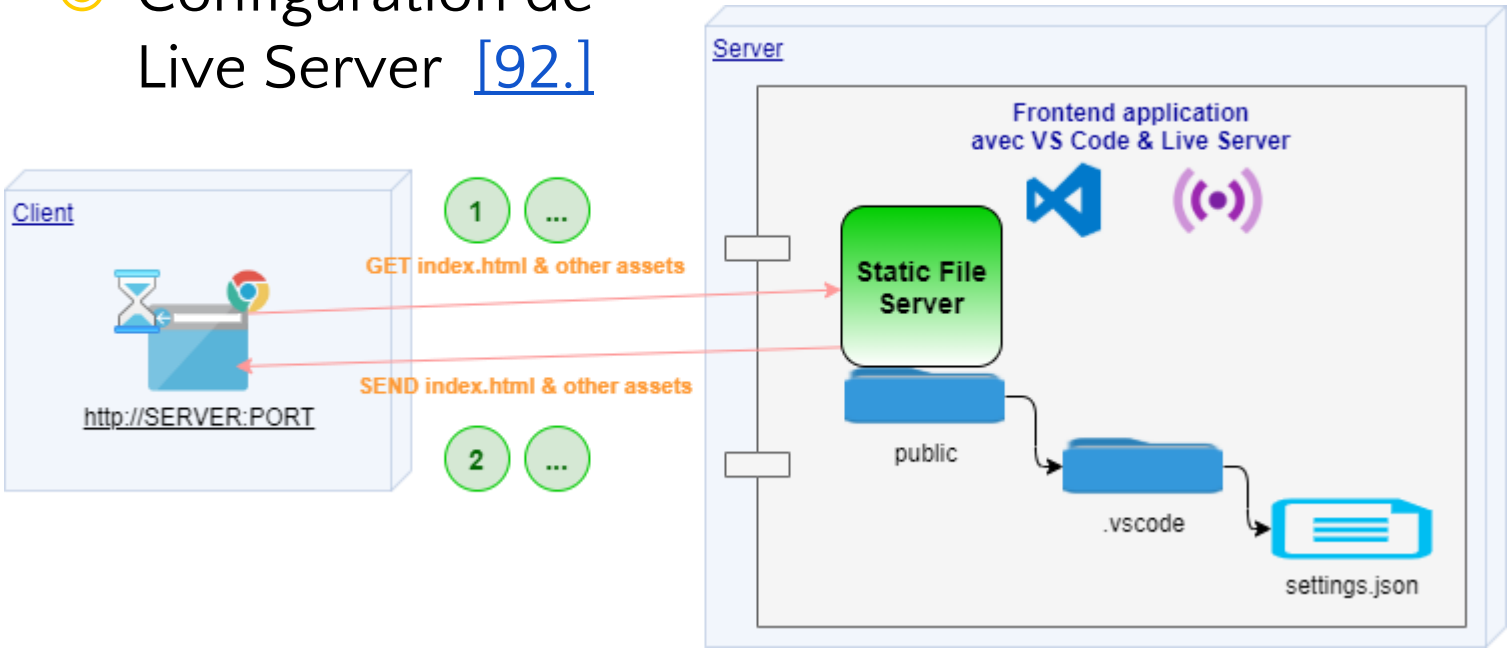


Chargement du frontend via un serveur de fichiers Express



Utilisation de VS Code et Live Server pour votre frontend

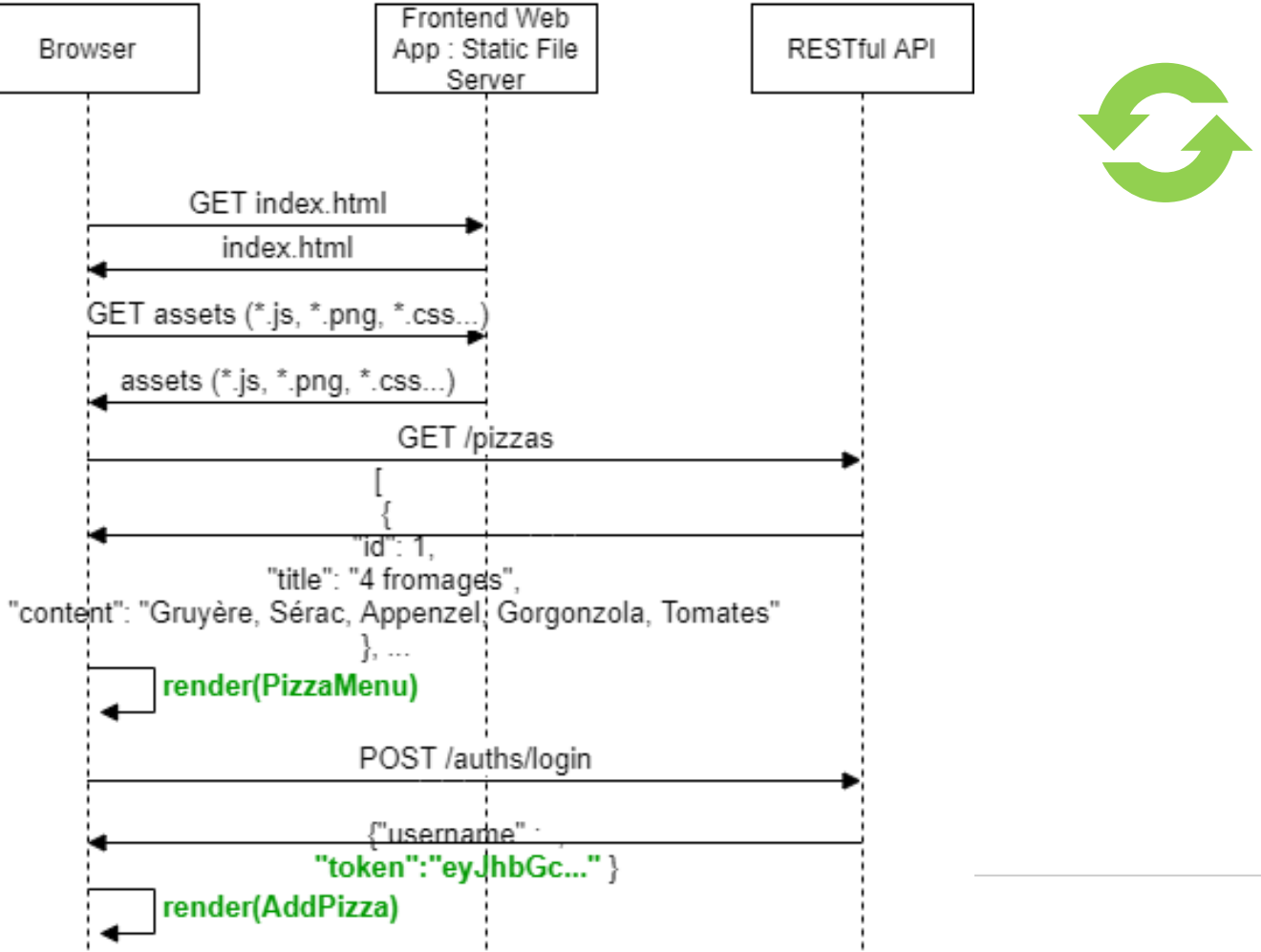
- Configuration de Live Server [92.]



Gestion des communications d'une SPA

Comment est-ce que notre frontend va communiquer avec notre Web API ?

Comment communiquer pour obtenir une liste de pizzas ?



Quelles protocoles / techniques principales pour communiquer avec une SPA ?

- AJAX :
  - **A**synchronous **Ja**vaScript and **X**ML
  - Requête HTTP asynchrone
  - Transport de données via XML (autrefois) ou JSON
  - Combinaison de technologies (HTML/CSS, DOM, JSON ou XML, XMLHttpRequest, JS) pour réaliser une application web asynchrone
- Websockets : technologie de communication temps-réel client/serveur bidirectionnelle

	Support			Features				
	Chrome & Firefox[1]	All Browsers	Node	Concise Syntax	Promises	Native[2]	Single Purpose[3]	Formal Specification
XMLHttpRequest	✓	✓				✓	✓	✓
Node HTTP			✓			✓	✓	✓
fetch()	✓			✓	✓	✓	✓	✓
Fetch polyfill	✓	✓		✓	✓		✓	✓
node-fetch			✓	✓	✓		✓	✓
isomorphic-fetch	✓	✓	✓	✓	✓		✓	✓
superagent	✓	✓	✓	✓			✓	
axios	✓	✓	✓	✓	✓		✓	
request			✓	✓			✓	
jQuery	✓	✓		✓				
reqwest	✓	✓	✓	✓	✓		✓	

[1] **Chrome & Firefox** are listed separately because they support `fetch()`: caniuse.com/fetch [2] **Native:** Meaning you can just use it - no need to include a library. [3] **Single Purpose:** Meaning this library or technology is ONLY used for AJAX / HTTP communication, nothing else.

Comparaison de librairies AJAX/HTTP [73.]

Appel asynchrone d'une opération d'une Web API

- Quelle librairie AJAX (environ 20 ans d'âge) utiliser pour ce cours ?
- Standard actuel : **Fetch API**
- Anciennement :
  - **XMLHttpRequest**
  - **$.ajax()**

Fetch : requête asynchrone vers une API

Fetch API [74.]

```
fetch("/api/users")
    .then((response) => {
      if (!response.ok)
        throw new Error(
          "Error code : " + response.status + " : " + response.statusText
        );
      return response.json();
    })
    .then((data) => onUserList(data)) // build and render the user list
    .catch((err) => onError(err));
```

Fetch : requête asynchrone vers une API

Fetch API [74.]

```
try {
    // hide data to inform if the pizza menu is already printed
    const response = await fetch("/api/pizzas"); // fetch return a promise
    if (!response.ok) {
      // status code was not 200, error status code
      throw new Error("fetch error : " + response.status + " : "
                      + response.status Text);}
    const pizzas = await response.json(); // json() returns a promise
    // generate a table and attach it to #page div …
  } catch (error) {
    console.error("pizzaView::error", error);
  }
```

Fetch : requête asynchrone vers une API

Fetch API [74.]

```
fetch("/api/users/", {
    method: "POST", // *GET, POST, PUT, DELETE, etc.
    body: JSON.stringify(user), // body data type must match "Content-Type" header
    headers: {
      "Content-Type": "application/json",
    },
  })
    .then((response) => { // deal with errors
        return response.json();
    })
    .then((data) => onUserRegistration(data)) // re-render navBar and go to /list
    .catch((err) => onError(err));
```

JQuery : requête asynchrone vers une API

- $.ajax() [75.]

```
$.ajax({
  type: "post",
  url: "/auths/login",
  data: { username: $("#username").val(), password: $("#password").val() },
  dataType: "json",
  success: function (response) {
    // Do something : if dataType was specified to “json”, response has already been
    // parsed to an Object. Else : reponse=JSON.parse(response);
  },
  error: function name(err, status, message) { // Do something in case of error
  },
});
```

SPA avec frontend indépendant de la RESTful API

- DEMO : Création d'une SPA pour une pizzeria : Part 0 : Intégration de la RESTful API de la pizzeria et problème de CORS

  Frontend sur le port 8080 et RESTful API sur le port 3000

```
Access to fetch at 'http://localhost:3000/pizzas' from origin 'http://localhost:8080' has been blocked by CORS policy: No 'Access-Control-Allow-Origin' header is present on the requested resource. If an opaque response serves your needs, set the request's mode to 'no-cors' to fetch the resource with CORS disabled.                    localhost/:1

▶GET http://localhost:3000/pizzas net::ERR_FAILED                    index.js?b635:65
```