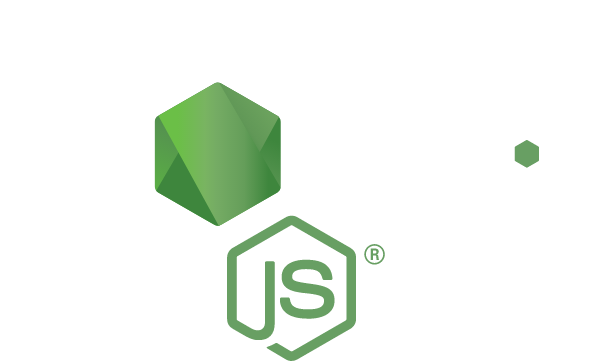
|  |  |
| --- | --- |
| **BTS SIO** | **BTS SIO (SLAM)** |
|  | **Bloc 3** |
|  |  |
| **Express et routage** | |



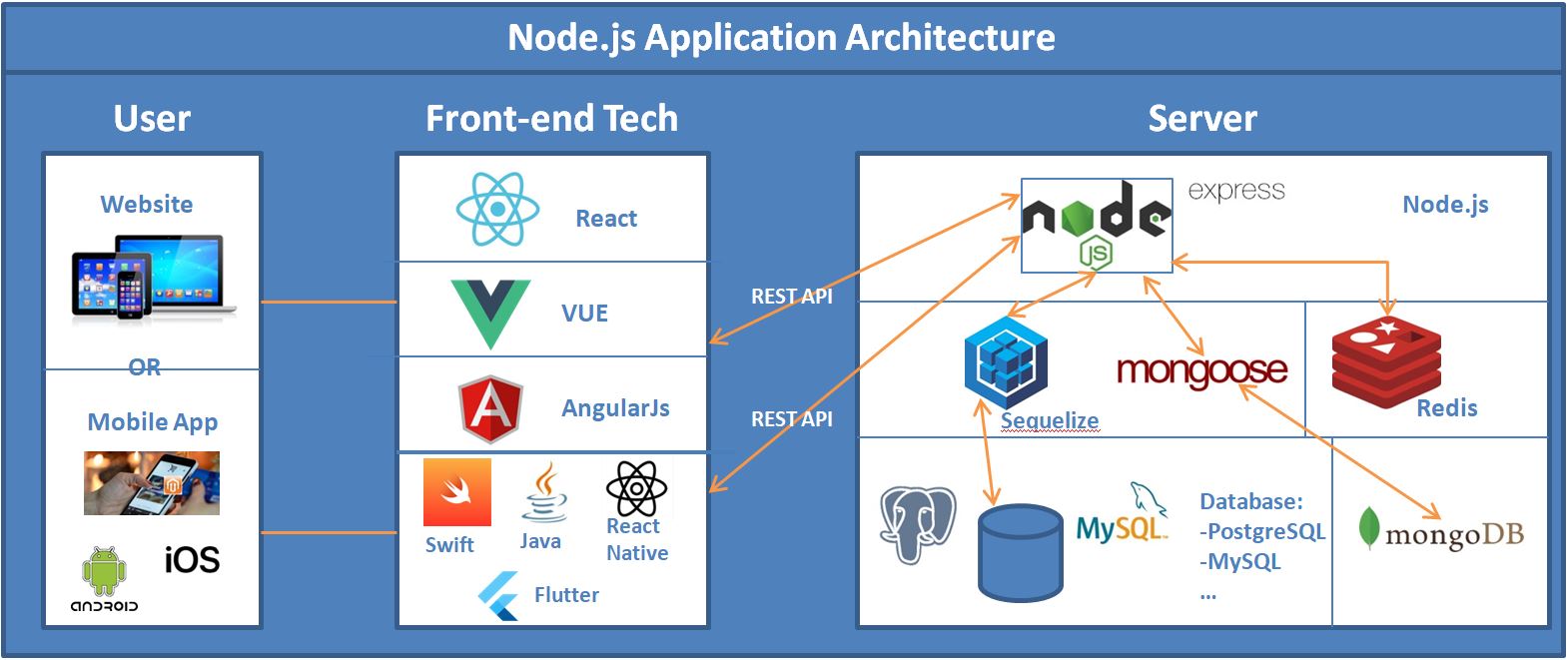
**Express**

|  |  |  |  |
| --- | --- | --- | --- |
| **date** | **révision** | | |
|  | Timothée Robert | | |
|  |  | | |
|  |  | | |
|  |  | | |
|  |  | | |
|  |  | | |
|  |  | | |
| **BTS SIO** | |  | **BTS SIO (SLAM)** |
|  | | |  |
| INTRODUCTION | | |  |

Express est une librairie importante de NodeJS, existante sous forme de framework d’application Web, permettant de fournir à l’écosystème NodeJS des outils puissant et robustes pour le développement d’applications Web et Mobiles.

# Architecture

## Architecture d’une application Node

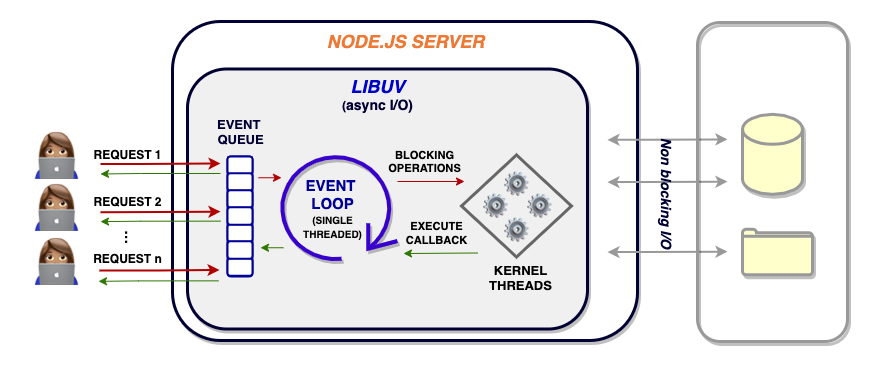


Ce schéma d’architecture globale représente à gauche les 2 principaux types d’application pouvant être développées en utilisant NodeJS avec la librairie Express.

Au milieu les technologies Front End, en l’occurrence ici sur le schéma 3 frameworks JavaScript (React, Vue et Angular), un Framework Apple, un Java …

A droite les technos sur le serveur, en l’occurrence NodeJS et Express complétées d’accès aux données (Sequelize est un ORM, Redis un serveur de cache).

## Fonctionnement du serveur NodeJS

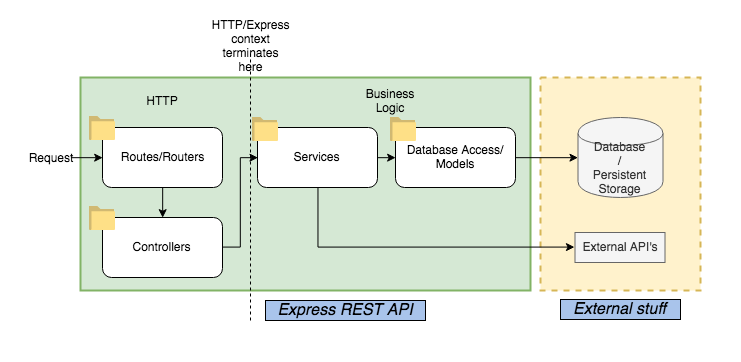


Ce schéma représente le fonctionnement de NodeJS : il est basé sur les évènements et non bloquant, dans un seul Thread. Les opérations bloquantes sont lancées de manière asynchrone et, si retour il y a, gérée via des Callbacks (fonctions de rappel).

Explications orales à fournir en complément (on y reviendra).

## Couches du framework Express de NodeJS

Les différentes couches de http/Express:



Une couche http/Express qui gère les routes auxquelles sont associées des contrôleurs, une couche Services et une couche d’accès aux données. Nous détaillerons ensuite.

# Le routage

## Introduction et définition

Le routage est un des aspects les plus importants d’un site Web ou d’un service Web.

Définition : le routage est le mécanisme par lequel des requêtes, spécifiées par une URL et une méthode (verbe http), sont dirigées (routées) vers le code qui les prend en charge.

Préliminaire : rappels - les verbes http

1. Quels sont les différents verbes HTTP ?
2. Quel verbe permet d’accéder à une ressource ?
3. Quel verbe permet de créer une nouvelle ressource ?
4. A quelle balise HTML est généralement associé le verbe de la question précédente ?

## Définition du routage dans Express

Documentation : méthode de routage dans le framework Express

Le routage pour un développeur Web a une signification significativement différent de celle qui prévaut pour les réseaux informatiques.

Dans NodeJS, une fois importé le middleware express, associé à une constante, on peut utiliser directement les méthodes get et post du middleware ou alors utiliser un gestionnaire de routage (Router) : <https://expressjs.com/fr/guide/routing.html>

## Un exemple complet

Sur PHPStorm, dans un projet NodeJS,

Etape 1 : il faut importer le module express.

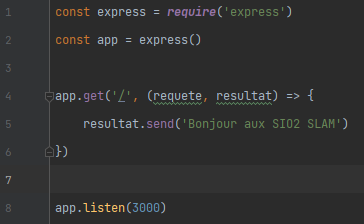
Les imports de librairie JavaScript dans l’écosystème NodeJS se font avec Npm (cf documentation sur l’installation de NodeJS).

Dans une fenêtre de Terminal (sous PHPStorm, View 🡪 Tool Windows 🡪 Terminal ou ALT+F12 ), exécutez la commande suivante :



Etape 2 :

créez un fichier routeintro.js , avec le code suivant :



Etape 3 :

Exécutez le et ensuite aller sur le navigateur à l’adresse <http://localhost:3000/>

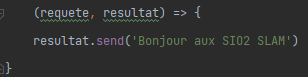
## Analyse

Ligne 1 : on importe le module express dans le programme , il est dorénavant référencé  par une constante que l’on nomme …. express

Ligne 2 : on crée une application Express que l’on affecte à une constante app. express() est une function de top niveau exportée par la librairie Express. <https://expressjs.com/en/5x/api.html#express>

Ligne 4 : on crée une route qui va gérer toutes les requêtes http de type GET vers le chemin / (racine de l’application Web) via une fonction de rappel , un callback

Ici notre callback est :



Remarque :

- l’objet requete (dans la doc généralement écrit req) représente la requête http et contient des propriétés pour la chaine de requête (query string), les paramètres, le body, les entêtes http etc

- l’objet resultat (dans la doc généralement écrit res) contient la réponse http renvoyée au client.

- resultat.send : envoie la réponse http au client. <https://expressjs.com/en/5x/api.html#res.send>

Les entêtes ainsi que le HEAD et le Content-type sont automatiquement alimentés

Ligne 8 : app.listen() attache l’application à un port, ici le 3000 et écoute pour des connections entrantes sur l’hôte et le port spécifiés.

<https://expressjs.com/en/5x/api.html#app.listen>

## Questions de recherche

- Dans quel répertoire l’installation des packages a-t-elle été effectuée ?

DANS NOTRE FICHIER ROOT DU PROJET ET DANS NODE-MODULES

- Quel fichier de configuration lié à NodeJS contient la liste des dépendances du projet ?

PACKAGE.JSON

- Qu’est-ce qu’une dépendance ?

LORSQU’UN LOGICIEL A BESOIN D’UN AUTRE LOGICIEL POUR FONCTIONNER

- Lorsque on met le projet sur GIT, a-t-on besoin de transférer les paquetages (dépendances) ?

OUI

- Quelle commande doit on généralement effectuer lorsqu’on installe un projet GIT NodeJS ? Pourquoi ?

Git clone (lien du github) =>

- A quoi sert la librairie express ?

## Utilitaire de hot reloading : nodemon

Pour le moment, à chaque fois qu’on effectue une modification, il faut :

* Arrêter le processus Nodejs
* Relancer le programme : node NomProgramme.js

Heureusement, il existe un package, nommé **nodemon**, qui permet de faire du hot reloading du serveur.

Etape 1 : installation du package

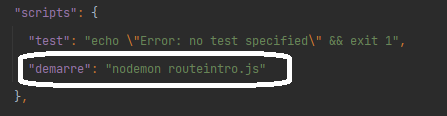
Dans une fenêtre de Terminal à la racine du projet : **npm install nodemon**

Etape 2 : ajout d’un script dans le fichier package.json

Etape 3 :

On démarre via la commande : **npm run nomDuScriptDansPackageDotJson** (et non plus par node nomProgrammeJavaScript)

Exemple : j’ajoute ici un script dans le fichier de configuration npm du projet nommé package.json, avec une entrée demarre : quand cette entrée est exécutée le programme routeintro.js est exécuté et de plus les modifications à ce programme sont automatiquement prises en compte



Pour démarrer , dans la fenêtre Terminal, on saisit alors :

**npm run demarre**

Cela permettra de démarrer le programme routeintro.js avec une fonctionnalité de hot reloading, très pratique en phase de développement.

## Exercices

Exercice n°1 : routage simple

Dans votre programme routeintro.js, ajoutez une route de type get qui au chemin ***accueil*** associe le message « Vous êtes arrivés sur la page d’accueil du site »

Exercice n°2 : routage et envoi d’un fichier HTML préexistant

Pour envoyer un fichier HTML existant vers le navigateur il faut simplement construire le chemin relatif du fichier.

* Tout d’abord on ajoute la librairie standard path

const path = ***require***('path');

Toujours dans le même programme, on va définir une route qui au chemin ***bienvenue*** renvoie le fichier bienvenue.html

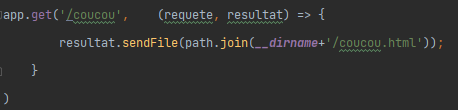
* Ajouter un fichier bienvenue.html avec un message de bienvenue
* Utiliser la fonction sendFile du resultat avec la fonction path.join

<https://expressjs.com/en/5x/api.html#res.sendFile>

<https://nodejs.org/api/path.html#pathjoinpaths>

\_\_dirname est une variable d’environnement qui indique le chemin absolu du répertoire contenant le fichier en cours d’exécution

Hint : si vous ne trouvez pas avec la documentation, ci-joint un exemple



Exercice n°3 : routage et ancre vers une route

* Ajoutez une autre route ***panorama*** qui renvoie un fichier planDuSite.html
* Dans ce fichier html, créez des liens hypertexte vers toutes les routes précédentes (accueil et bienvenue)