



atsuhikoMochizuki

PIZZAHUTTE

Création d'une application Javascript FullStack complète



Administration d'une interface clientèle WEB dédiée à une infrastructure de type Pizzeria



Projet individuel

Sous la direction de Rossi Oddet

Avril 2023

git clone <https://github.com/atsuhikoMochizuki/pizzahute.git>



Table des matières

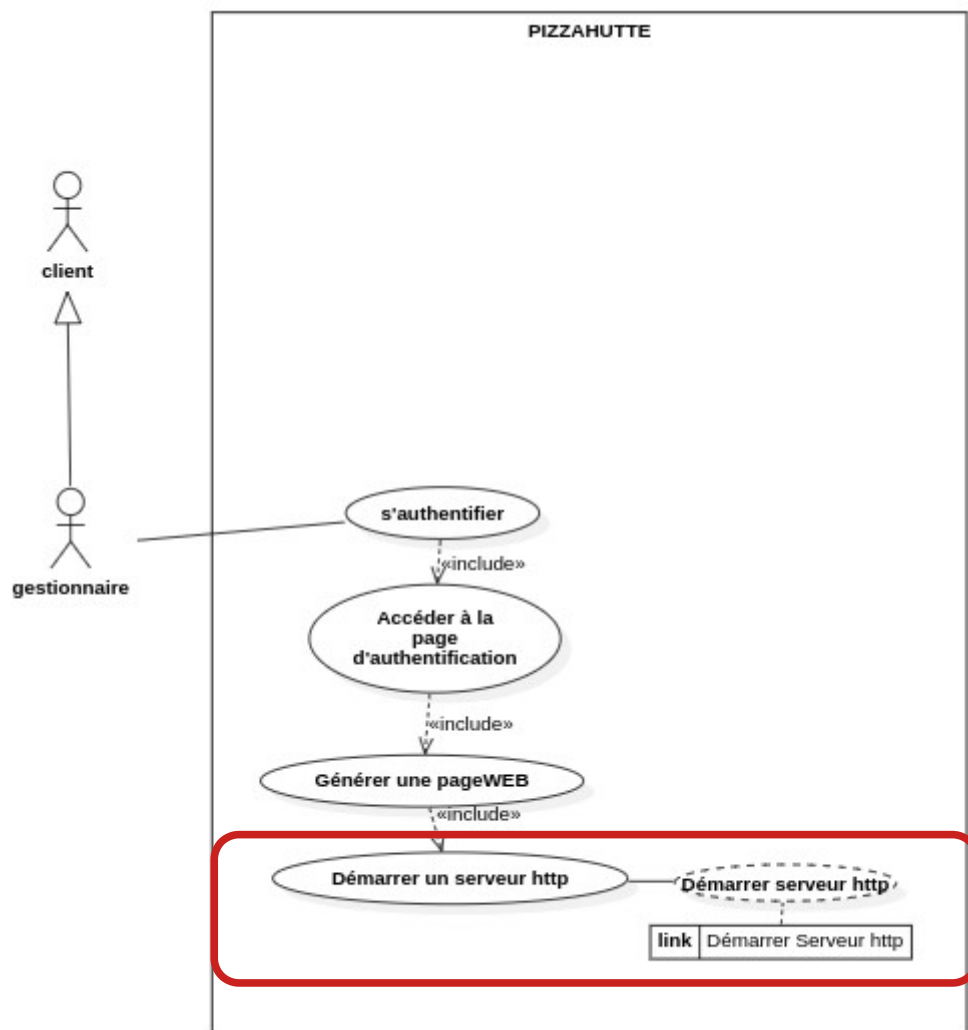
1. ANALYSE.....	4
1.1. Diagramme des cas d'utilisation.....	4
1.2. Découpage fonctionnel.....	5
1.2.1. FPA-4 : S'authentifier.....	6
1.2.1.1. FPA-3 Générer la page d'authentification.....	6
1.2.1.1.1. FP1-2 : FPGénérer une page WEB.....	6
1.2.1.1.1.1. FPA-1 : Démarrer un serveur HTTP.....	6
2. MODELISATION DE L'APPLICATION.....	7
2.1. FPA-1 : Implémenter un serveur HTTP.....	7
2.1.1. Premier Jet.....	8
2.1.1.1. Diagramme de séquence.....	8
2.1.1.2. Diagramme des classes participantes.....	8
2.1.2. Second jet.....	9
2.1.2.1. Diagramme de séquence corrigé et approfondi.....	9
2.1.2.2. Déduction d'un diagramme des classes participantes beaucoup plus réaliste.....	12
2.2. FPA-2 : Générer une page WEB.....	13
2.3. FPA-3 : Générer la page d'authentification.....	14
2.3.1. Reprise nécessaire des fondamentaux de l'outil Bootstrap.....	14
2.3.2. En résumé : ce qu'il est fondamental de saisir du système Bootstrap, comment penser cet outil puissant et responsive.....	15
2.4. FPA-4 : S'authentifier.....	18
3. CHOIX DES OUTILS TECHNOLOGIQUES.....	19
3.1. Choix de l'infrastructure.....	20
3.1.1. Génération des pages côté serveur.....	20
3.1.2. Choix du framework NodeJS pour l'implémentation du serveur en Javascript.....	20
3.1.2.1. Surcouche applicative Express.....	21
3.1.2.2. Génération des pages : moteur de vue PUG.....	21
3.1.3. Mise page du front.....	21
3.1.3.1. Choix de la librairie Bootstrap pour optimiser le temps de développement et le responsive du front.....	21
3.1.3.2. Système de gestion de base de données relationnelles : MySQL.....	22
3.1.3.3. Outils de scripting : BASH.....	22
3.1.4. Design.....	22
3.1.5. Charte graphique, logos : app.logo.com.....	22
3.1.6. Outils de gestion Projets : GitHub.....	22
3.1.7. Outils de modélisation : méthode UML.....	22
4. ORGANISATION ET PLANIFICATION DU PROJET.....	23
4.1. Découpage des tâches sous la forme KANBAN.....	23
5. MISE EN PLACE DU DEVELOPPEMENT.....	25
5.1. Mise en place de l'environnement de travail.....	26
5.1.1. Structuration de l'arborescence du projet.....	26
5.1.2. Création du dépôt github et clonage du projet vierge.....	26
5.1.3. Création du projet node via npm.....	26

5.1.3.1. installation des dépendances.....	26
5.1.3.2. Paramétrage du fichier de configuration package.json.....	26
5.1.3.2.1.1. Affectation de nodemon à la commande start de l'objet script.....	26
5.1.3.2.2. Prise en charge des modules ECMAScript (import, export).....	26
5.1.4. Consignation de la procédure dans un script BASH. Permettra de générer immédiatement le projet en cas de « pépin ».....	27
5.1.5. Création pour l'occasion d'une petite application le MochizukiServerProjectGenerator	28
6. CODAGE DE L'APPLICATION.....	29
6.1. Pf1 : Démarrer un serveur.....	30
6.1.1. Script de contrôle principal Www.js.....	30
6.1.2. Interface de gestion de l'application Express app.js.....	31
6.1.3. Script de routage de la page d'accueil authentication.js.....	32
6.1.4. Script de routage de la page d'accueil client users.js.....	33
7. Annexes.....	33
7.1. Trop de temps perdu par une compréhension incomplète de l'outil Bootstrap. Prenons le temps de remédier à ça.....	33
7.1.1. Donner la capacité à un élément de pouvoir modifier ses dimensions afin de remplir l'espace disponible dans son conteneur : la propriété Flex.....	33
7.1.2. Les concepts de base des conteneurs flexibles, les FLEXBOX.....	36
7.1.3. Les deux axes des boîtes flexibles flexbox.....	36
7.1.3.1. L'axe principal.....	37
7.1.3.2. L'axe secondaire (cross axis).....	38
7.1.4. Créer un conteneur flexible sur plusieurs lignes avec flex-wrap.....	39
7.1.5. La propriété raccourcie flex-flow.....	40
7.1.6. Alignement, justification et distribution de l'espace disponible entre les éléments.....	40
7.1.6.1. align-items.....	40
7.1.6.2. justify-content.....	41
7.1.7. Saisir pleinement l'outil Bootstrap, un investissement qui sera rapidement payant à court terme : allons-y.....	42
7.1.7.1. Le système de grilles de Bootstrap.....	42
7.1.7.2. Les containers (.container).....	43
7.1.7.3. Les lignes (.row).....	43
7.1.7.4. Les colonnes(.col).....	44
7.1.7.4.1. Alignement vertical.....	45
7.1.7.5. alignement horizontal.....	46
7.1.7.6. Les gouttières (.g).....	47
7.1.7.6.1. Gouttières horizontales.....	48
7.1.7.6.2. Gouttières verticales.....	48
7.1.7.6.3. Gouttières verticales ET horizontales.....	48
7.1.8. Utilitaires pour la mise en page.....	49
7.1.8.1. Modifier l'affichage.....	49
7.1.8.1.1. Les options FlexBox.....	49
7.1.9. Typographie.....	50
7.1.9.1. Les titres.....	50
7.1.9.2. En tête.....	52
Éléments de texte en ligne.....	52
7.1.10. Les listes.....	53

7.1.11. Les couleurs dans bootstrap 5.....	54
--	----

1. ANALYSE

1.1. Diagramme des cas d'utilisation



1.2. Découpage fonctionnel

La visualisation d'un diagramme de cas d'utilisation nous donne l'avantage d'avoir le recul maximal sur l'application. Cette hauteur de vue nous donne l'avantage incontestable de pouvoir appréhender l'application dans sa globalité, ce qui nous met toutes les chances de côté pour implémenter une solution robuste, modulaire et maintenable dans le temps.

Nous allons à partir du diagramme des cas d'utilisation descendre de façon granulaire dans les fonctionnalités à développer pour atteindre les objectifs métiers de l'application.

Une fois ce découpage effectué, le développement se fera dans le sens inverse.

Nous implémenterons les briques de base pour petit construire les fonctionnalités exprimées dans le cahier des charges.

1.2.1. FPA-4 : S'authentifier

1.2.1.1. FPA-3 *Générer la page d'authentification*

1.2.1.1.1. FP1-2 :FP Générer une page WEB

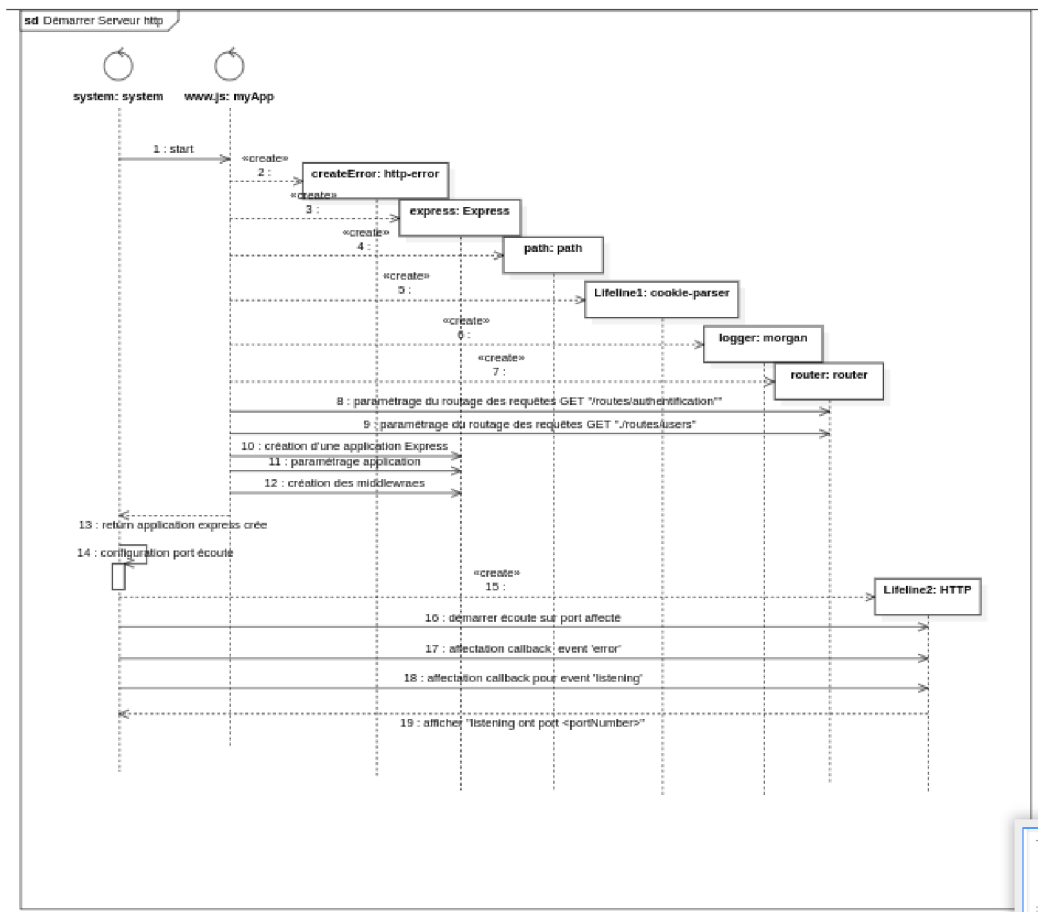
1.2.1.1.1.1. FPA-1 : *Démarrer un serveur HTTP*

2. MODELISATION DE L'APPLICATION

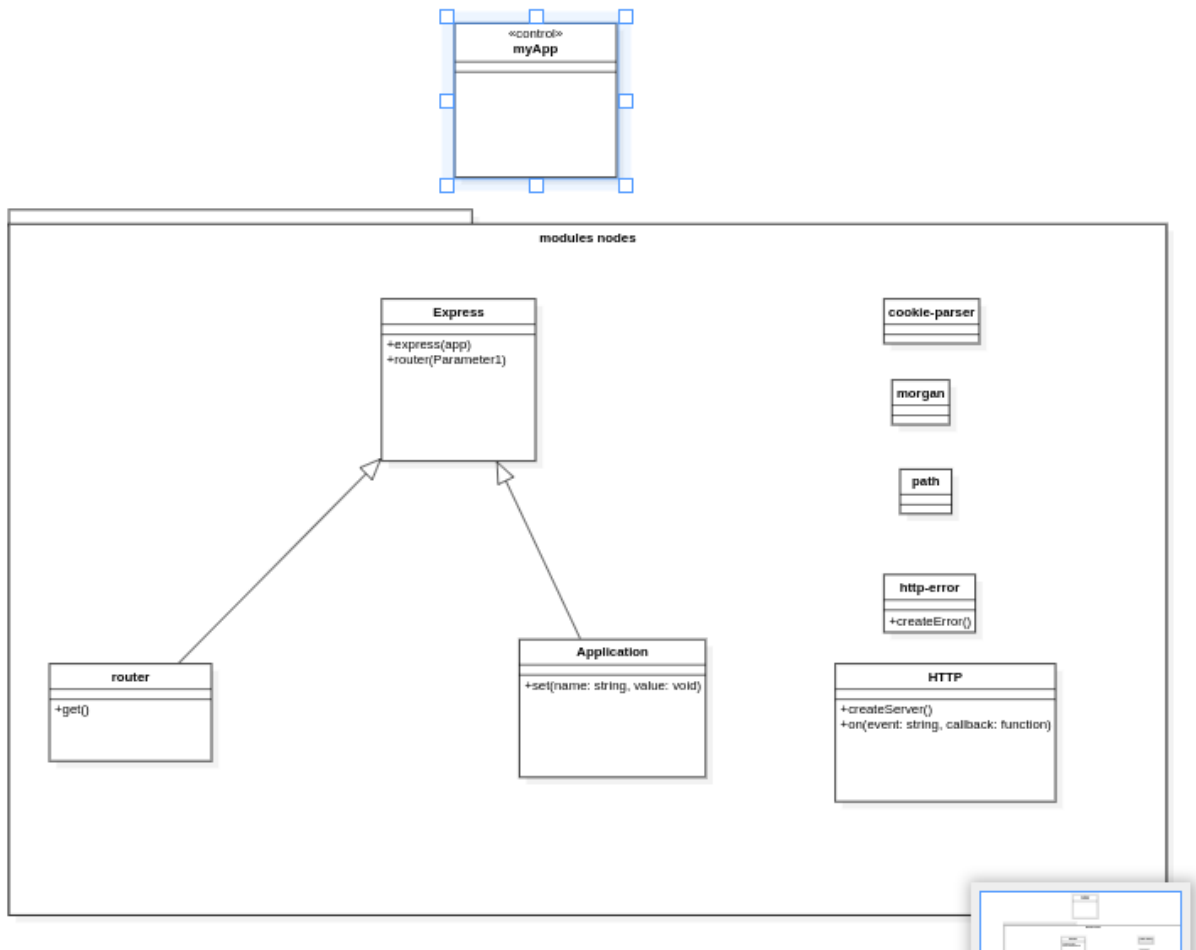
2.1. FPA-1 : Implémenter un serveur HTTP

2.1.1. Premier Jet

2.1.1.1. Diagramme de séquence



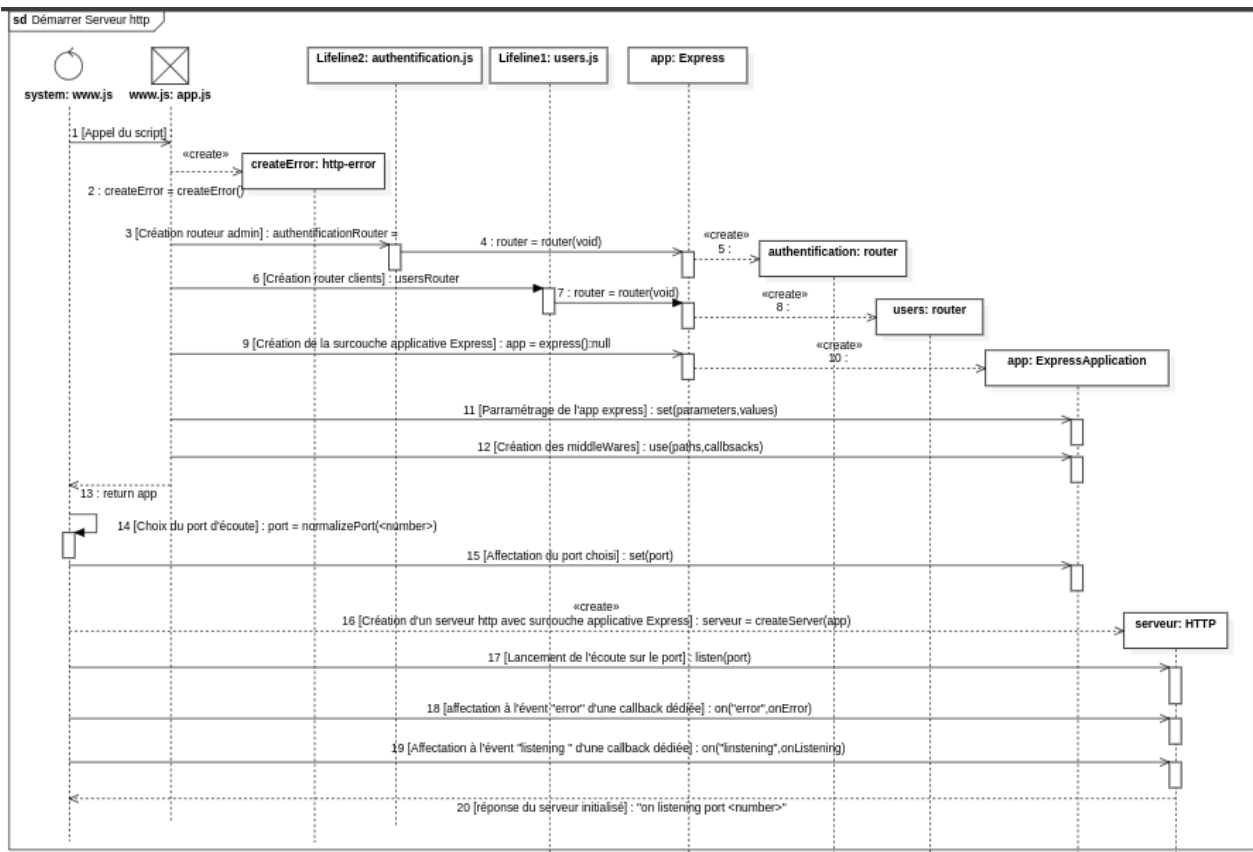
2.1.1.2. Diagramme des classes participantes

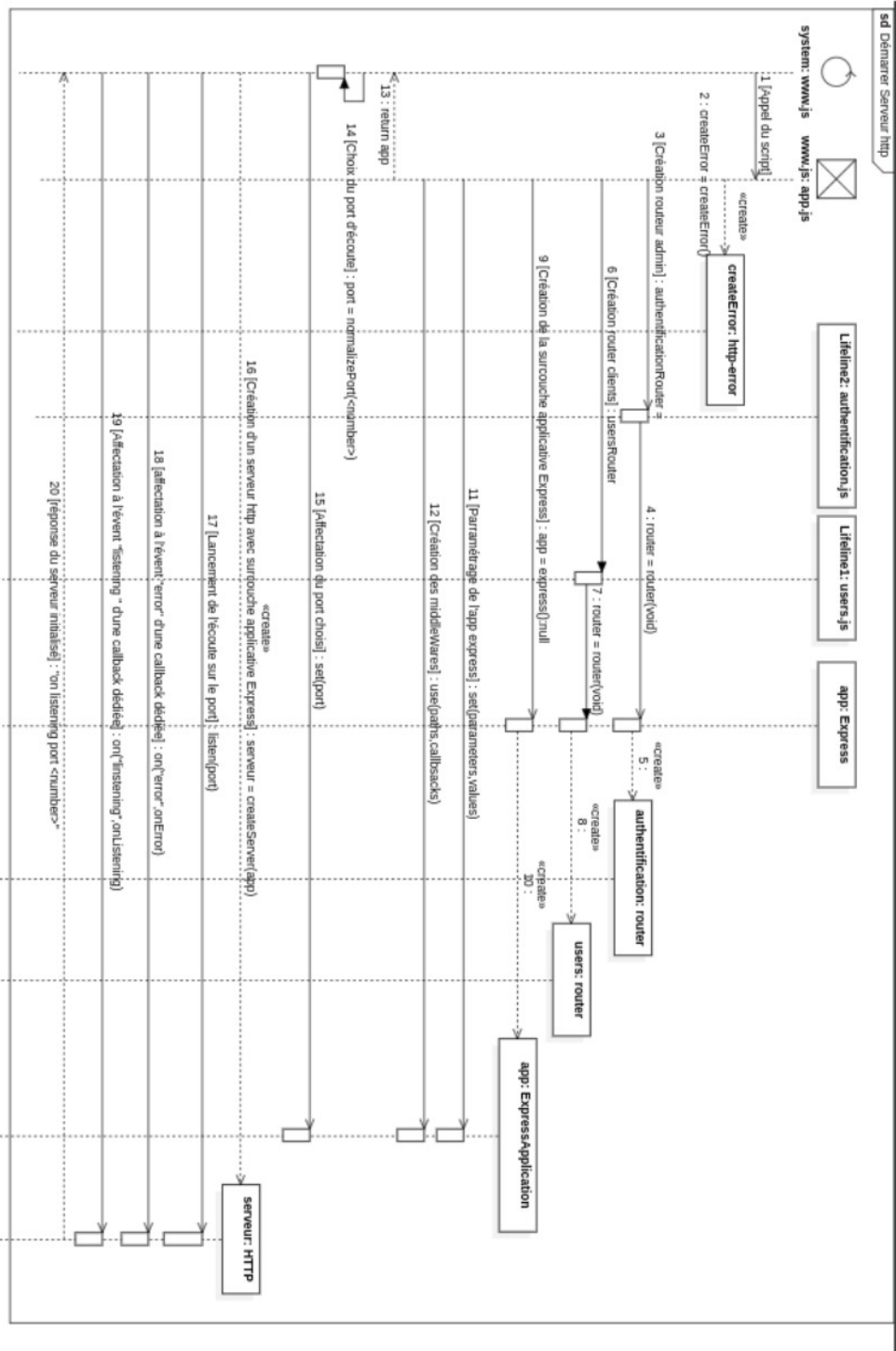


2.1.2. Second jet

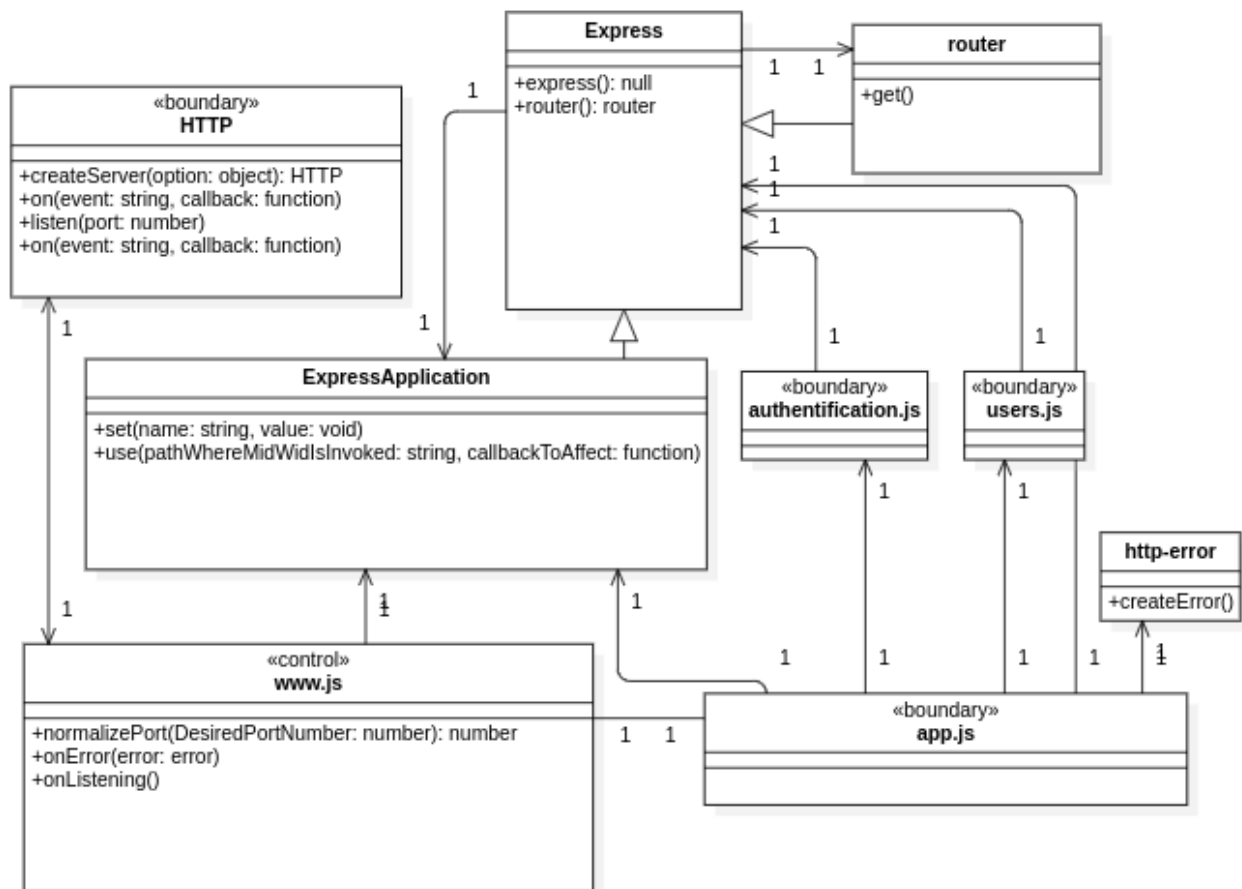
Nous allons affecter les méthodes des classes aux différentes interactions entre les objets de l'application

2.1.2.1. *Diagramme de séquence corrigé et approfondi*





2.1.2.2. Dédution d'un diagramme des classes participantes beaucoup plus réaliste



2.2. FPA-2 : Générer une page WEB

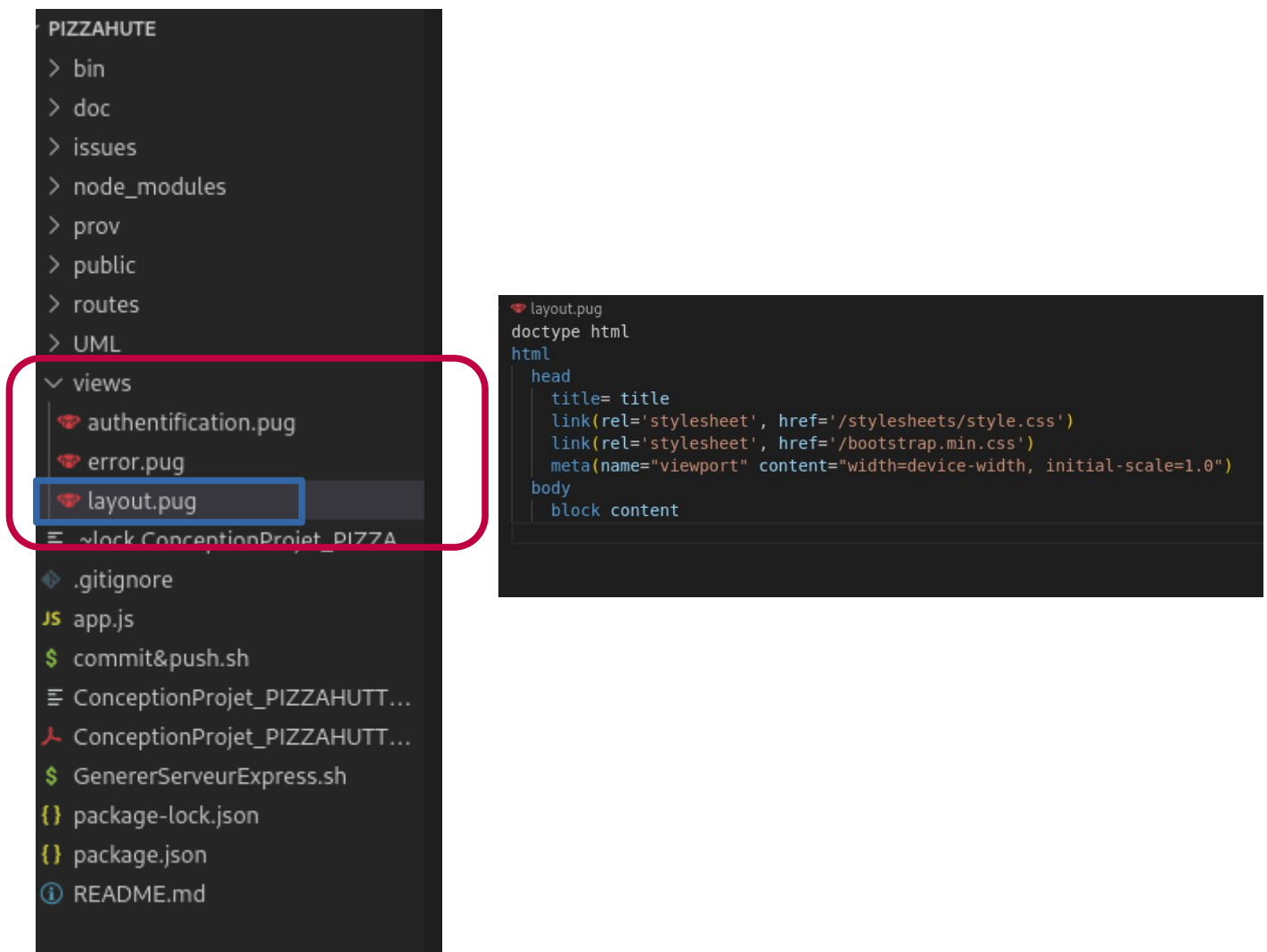
L'implémentation de cette partie nous fait découvrir la puissance de la surcouche applicative Express.

Notre serveur est paramétré et les routeurs sont en attente.

Il nous faut simplement afficher la page.

Nous utilisons le moteur de vue PUG.

Nous n'aurons donc dans notre application qu'à décrire, à l'aide de la syntaxe très intuitive et rapide de PUG, les éléments de la page dans la vue prévue à cet effet.



Express s'occupe alors de tous.

2.3. FPA-3 : Générer la page d'authentification

Nous allons ici commencer à coder notre front. L'utilisation de pug va permettre une implémentation rapide de nos interfaces.

Le choix de Bootstrap a également fait dans cette visée.

2.4. FPA-4 : S'authentifier

3. CHOIX DES OUTILS TECHNOLOGIQUES

3.1. Choix de l'infrastructure

3.1.1. Génération des pages côté serveur

Les pages seront générées dynamiquement côté serveur. Ceci va nous permettre :

- D'approfondir les bases de la programmation côté serveur
- Générer les pages de façon dynamique

- Sécuriser le site
- Administrer la base de données

3.1.2. Choix du framework NodeJS pour l'implémentation du serveur en Javascript

L'application sera de type pages générées dynamiquement par le serveur.

Nous utiliserons NodeJs pour implémenter le serveur.

Node.js est une plateforme logicielle libre en JavaScript, orientée vers les applications réseau évènementielles hautement concurrentes qui doivent pouvoir monter en charge.

Elle utilise la machine virtuelle V8, la bibliothèque libuv pour sa boucle d'évènements, et implémente sous licence MIT les spécifications CommonJS.

Parmi les modules natifs de Node.js, on retrouve http qui permet le développement de serveur HTTP. Ce qui autorise, lors du déploiement de sites internet et d'applications web développés avec Node.js, de ne pas installer et utiliser des serveurs webs tels que Nginx ou Apache.

Concrètement, Node.js est un environnement bas niveau permettant l'exécution de JavaScript côté serveur.

Nous utiliserons également et en surcouche la librairie Express de Node qui va nous permettre une gestion du serveur plus aisée et plus robuste.

3.1.2.1. Surcouche applicative Express

Express est une infrastructure d'applications Web Node.js minimaliste et flexible qui fournit un ensemble de fonctionnalités robuste pour les applications Web et mobiles.

Grâce à une foule de méthodes utilitaires HTTP et de middleware mise à votre disposition, la création d'une API robuste est simple et rapide.

Express apporte une couche fine de fonctionnalités d'application Web fondamentales, sans masquer les fonctionnalités de Node.js

3.1.2.2. Génération des pages : moteur de vue PUG

3.1.3. Mise page du front

3.1.3.1. Choix de la librairie Bootstrap pour optimiser le temps de développement et le responsive du front

Bootstrap est une collection d'outils utiles à la création du design (graphisme, animation et interactions avec la page dans le navigateur, etc.) de sites et d'applications web. C'est un ensemble qui contient des codes HTML et CSS, des formulaires, boutons, outils de navigation et autres éléments interactifs, ainsi que des extensions JavaScript en option. C'est l'un des projets les plus populaires sur la plateforme de gestion de développement GitHub.

3.1.3.2. Système de gestion de base de données relationnelles : MySQL

3.1.3.3. Outils de scripting : BASH

Nous consignerons en particulier certaines procédures dans des scripts bash dans un but de réimplémentation et d'automatisation.

3.1.4. Design

3.1.5. Charte graphique, logos : app.logo.com

3.1.6. Outils de gestion Projets : GitHub

Le projet sera administré sous une approche SCRUM à l'aide des outils de Github, qui sera également notre hébergeur de dépôt.

3.1.7. Outils de modélisation : méthode UML

Nous utiliserons pour cela le logiciel multi-plateforme StarUML, très performant.

4. ORGANISATION ET PLANIFICATION DU PROJET

4.1. Découpage des tâches sous la forme KANBAN

View 1+ New View

Filter by keyword or by field

32

Todo

This item hasn't been started

pizzahute #3

USA003 - Utilisateur - CRUD

pizzahute #4

USA004 - Pizza - CRUD

high

pizzahute #2

USA002 - Menu

high

pizzahute #5

USA005 - Livreur - CRUD

high

pizzahute #6

USA006 - Client - CRUD

high

pizzahute #7

USA007 - Commande - CRUD

+ Add item

1

In Progress

This is actively being worked on

pizzahute #1

USA001 - Authentification - Page Login

medium

+ Add item

1

Done

This has been completed

pizzahute #34

structuration du projet GIT

+ Add item

PIZZAHUTTE
Création d'une application Javascript FullStack complète

19

5. MISE EN PLACE DU DEVELOPPEMENT

5.1. Mise en place de l'environnement de travail

5.1.1. Structuration de l'arborescence du projet

5.1.2. Création du dépôt github et clonage du projet vierge

5.1.3. Création du projet node via npm

\$ npm init -y

5.1.3.1. *installation des dépendances*

```
npm install nodemon --save-dev  
npm install bootstrap --save  
npm install socket.io --save  
npm install body-parser --save
```

5.1.3.2. *Paramétrage du fichier de configuration package.json*

5.1.3.2.1.1. *Affectation de nodemon à la commande start de l'objet script*

```
"start": "nodemon ./bin/www"
```

5.1.3.2.2. *Prise en charge des modules ECMAScript (import, export)*

```
"type": "module"
```

5.1.4. Consignation de la procédure dans un script BASH. Permettra de générer immédiatement le projet en cas de « pépin »

Les étapes que nous avons listées précédemment sont à présent listées dans un script bash pour éviter à retaper toutes les instructions sur d'autres projets.

```
#!/usr/bin/bash
clear
cat << EOF
```



 G n rateur de projet

 version 1.0

 by Atsuhiko Mochizuki

EOF

```
read -p "Entrez le nom de l'application svp:" uservar
echo "[ ]Génération de l'application $uservar..."
echo "[ ]Moteur de vue:PUG"
echo "[ ]Génération de CSS :sass"
echo "[ ]Création d'un .gitignore"

express --pug -css sass --view pug --git $uservar
cd $uservar

echo "[ ]Installation des dépendances..."
echo "[ ]      --nodemon"
npm install nodemon --save-dev
echo "[ ]      --bootstrap"
npm install bootstrap --save
echo "[ ]      --socket.io"
npm install socket.io --save
echo "[ ]      --body-parser"
npm install body-parser --save

echo "[ ]Test du serveur"
echo -e "\033[31m [ ]Serveur en attente : veuillez entrer dans un navigateur 'localhost:3000'\n\033[1;32m"
DEBUG=$uservar:* npm start
```

5.1.5. Création pour l'occasion d'une petite application le MochizukiServerProjectGenerator

Pour étendre l'idée du script Bash déroulé précédemment, nous généralisons la procédure.

A savoir le script va générer en plus

- un dossier office structuré clé-en-main , avec une mise en page standardisée, prêt à la rédaction
- Une page de d'accueil du serveur clé-en-main
- Des logos

En résumé :

- Le temps de démarrage du projet est réduit à néant.

En un clic :

- Le serveur est généré et démarré avec une page d'accueil et d'authentification.
- Les vues PUG sont routées et prêtes à la rédaction
- Le dossier de conception est prêt :
 - mise en page OK
 - Plan d'analyse, de conception et de développement structuré
 - les logos, pieds de page, titres, polices, tables des matières sont prêtes à l'emploi.
- Le deuxième est qu'il va permettre de solidifier une méthode de conception qui pourra s'améliorer au fil du temps, à partir d'une base solide.
- Le dernier avantage est qu'il va permettre de standardiser la forme de nos projets dans le dépôt GitHub, qui va devenir une vitrine professionnelle pour la future activité professionnelle en sortie de formation.

6. CODAGE DE L'APPLICATION

6.1. Pf1 : Démarrer un serveur

6.1.1. Script de contrôle principal Www.js

```
#!/usr/bin/env node

/**
 * Module dependencies.
 */

var app = require("../app");
var debug = require("debug")("pizzahutte:server");
var http = require("http");

/**
 * Get port from environment and store in Express.
 */

var port = normalizePort(process.env.PORT || "2000");
app.set("port", port);

/**
 * Create HTTP server.
 */

var server = http.createServer(app);

/**
 * Listen on provided port, on all network interfaces.
 */

server.listen(port);
server.on("error", onError);
server.on("listening", onListening);

/**
 * Normalize a port into a number, string, or false.
 */

function normalizePort(val) {
  var port = parseInt(val, 10);

  if (isNaN(port)) {
    // named pipe
    return val;
  }

  if (port >= 0) {
    // port number
    return port;
  }
}
```

```

    return false;
}

/**
 * Event listener for HTTP server "error" event.
 */

function onError(error) {
    if (error.syscall !== "listen") {
        throw error;
    }

    var bind = typeof port === "string" ? "Pipe " + port : "Port " + port;

    // handle specific listen errors with friendly messages
    switch (error.code) {
        case "EACCES":
            console.error(bind + " requires elevated privileges");
            process.exit(1);
            break;
        case "EADDRINUSE":
            console.error(bind + " is already in use");
            process.exit(1);
            break;
        default:
            throw error;
    }
}

/**
 * Event listener for HTTP server "listening" event.
 */

function onListening() {
    var addr = server.address();
    var bind = typeof addr === "string" ? "pipe " + addr : "port " + addr.port;
    debug("Listening on " + bind);
}

```

6.1.2. Interface de gestion de l'application Express app.js

```

var createError = require("http-errors");
var express = require("express");
var path = require("path");
var cookieParser = require("cookie-parser");
var logger = require("morgan");

var authenticationRouter = require("./routes/authentication.js");
var usersRouter = require("./routes/users");
const exp = require("constants");

var app = express();

// view engine setup
app.set("views", path.join(__dirname, "views"));
app.set("view engine", "pug");

app.use(logger("dev"));
app.use(express.json());
app.use(express.urlencoded({ extended: false }));
app.use(cookieParser());
app.use(
    "/css",
    express.static(
        path.join(__dirname, "node_modules", "bootstrap", "dist", "css")
    )
);

```

```

    )
  );
  app.use(express.static(path.join(__dirname, "public")));
  app.use(
    express.static(
      path.join(__dirname, "node_modules", "bootstrap", "dist", "css")
    )
  );
  /*routes*/
  app.use("/", authenticationRouter);
  app.use("/users", usersRouter);

  // catch 404 and forward to error handler
  app.use(function (req, res, next) {
    next(createError(404));
  });

  // error handler
  app.use(function (err, req, res, next) {
    // set locals, only providing error in development
    res.locals.message = err.message;
    res.locals.error = req.app.get("env") === "development" ? err : {};

    // render the error page
    res.status(err.status || 500);
    res.render("error");
  });

  module.exports = app;

```

6.1.3. Script de routage de la page d'accueil authentication.js

```

var express = require("express");

var router = express.Router();

/* GET home page. */
router.get("/", function (req, res, next) {
  res.render("authentication", { title: "Pizzahutte" });
});

module.exports = router;

```

6.1.4. Script de routage de la page d'accueil client users.js

```
var express = require('express');
var router = express.Router();

/* GET users listing. */
router.get('/', function(req, res, next) {
  res.send('respond with a resource');
});

module.exports = router;
```

6.2. FPA-2 : Générer une page WEB

6.2.1. La puissance de l'infrastructure d'application WEB NodeJs EXPRESS

Nous nous reposerons ici **sur toute la puissance de Express**.

Le codage de app.js que les routeurs sont des middlewares qui font être lancés lorsque la demande chargement de la page dédié sera demandée côté client :

```
app.use("/", indexRouter);  
app.use("/users", usersRouter);
```

Nous avons pour l'instant *deux routeurs* qui sont configurés, **admin** et **client** et prêts à générer les pages lorsque le client en fait la demande via des requêtes de son navigateur.

Observons la forme du router de l'administrateur de la pizzeria (index.js) :

```
// Dependencies  
var express = require("express");  
var router = express.Router();  
  
//GET request  
router.get("/", function (req, res, next) {  
  res.render("index", {  
    title:  
      "PIZZAHUTTE-admin",  
  });  
});  
  
module.exports = router;
```

Nous renvoyons ici le rendu HTML de la vue index via la fonction de rappel.

Cette fonction utilise la méthode render() de la classe Application de l'API express (depuis notre objet). Cette dernière va ainsi permettre de pouvoir générer la page côté navigateur.

Le codage de la page d'authentification de fera donc ici dans le fichier index.

6.2.2. Organisation des vues

Nous avons choisi d'utiliser PUG dans notre application pour la génération de nos vues :

```
app.set("view engine", "pug");
```

Penchons nous un instant pour comprendre ce que peut nous apporter cet outil.

6.2.2.1. Le moteur de vue PUG

Pug est un **outil de templatage** qui permet de **générer du code HTML en compilant via une fonction Javascript**.

L'objectif de cet outil est de **se débarrasser des inconvénients que peut rencontrer un développeur front-end**. Il est utilisé en complément à **Node.js**.

Pug permet l'utilisation de **variables**. En compilant du code source Pug, ces dernières sont remplacées par les valeurs réelles, puis le résultat est envoyé au client dans une chaîne HTML.

6.2.2.1.1. Avantages

- La **simplification du code HTML**.
- L'utilisation de balises n'est plus nécessaire, grâce à un système d'indentations.
- Les classes et les id sont définis par des raccourcis, respectivement "." et "#".
- Le code obtenu est plus clair et plus agréable à lire.
- Le langage Pug est peu éloigné du Javascript, ce qui permet l'injection de code Javascript dans les fichiers Pug.
- A l'inverse du HTML *ou aucune indication n'est donnée en cas d'erreur* (cela sera visible uniquement à l'affichage de la page), le compilateur de Pug **repère directement les éventuelles erreurs**.
- Possibilité de définir des variables, d'ajouter de petits éléments de code comme des éléments de logique de base (if, each, unless...). On peut en outre écrire du JS directement depuis le fichier .pug, qui sera converti en page HTML.

6.2.2.1.2. Inconvénients

- l'utilisation d'indentations à la place du balisage peut s'avérer à double tranchant : une erreur d'indentation peut être fatale.
- peu d'IDEs prennent en charge le langage Pug : une demi-douzaine d'IDE seulement acceptent Pug, les plus connus étant Web Storm, PHP Storm et IntelliJ. Cela implique qu'il n'y a ni colorisation ni autosuggestions.
- Pug est incompatible avec le code HTML, c'est-à-dire qu'il est impossible de copier du code déjà existant. Cela pose évidemment un problème dans le cadre du refactoring, ou pour l'utilisations de parties de code HTML existants.
- Comme tous les langages, il faut passer par une étape d'installation et de familiarisation avec Pug. Cela s'avérer utile sur le long terme, mais il faut bien se poser la question du rapport temps passé à apprendre le langage par rapport au temps gagné sur le développement. La rentabilité augmente notamment avec la taille du projet et la diversité des pages HTML à écrire.

Les vues de notre projets seront organisées de la façon suivante :

- **index.pug**

Contiendra le contenu de la page d'authentification de l'administrateur de la pizzeria

- **users.js**

Contiendra le contenu de la page d'accueil du site

- **layout.pug**

Nous définirons dans cette vue un template de head qui sera appliqué à toutes les pages du site via la fonction Pug extends

6.3. FPA-3 Générer la page d'authentification de l'administrateur

6.3.1. Reprise nécessaire des fondamentaux de l'outil Bootstrap.

De nombreux points de frictions apparaissent rapidement avec Bootstrap, mettant en évidence une *incompréhension considérable sur la manière de penser et utiliser cet outil*, qui doit pourtant de coder très vite et responsive.

Nous allons donc sacrifier un temps considérable du projet à reprendre l'outil à la base, pour en tirer tous les bénéfices qu'il peut apporter en tant que développeur Front.

Les bénéfices en seront très profitables pour la suite de l'évolution du panels d'outils efficaces à maîtriser impérativement.

6.3.1.1. *En résumé : ce qu'il est fondamental de saisir du système Bootstrap, comment penser cet outil puissant et responsive.*(apporter précisions)

Le site de Bootstrap contient une documentation très bien faite, il n'est pas utile de retenir les noms des propriétés, qui, par ailleurs sont très intuitives et se mémorisent rapidement avec la pratique.

Ce qui est important de saisir, c'est plutôt la manière de penser l'outil. Voici ce que nous en avons pu en déduire après une étude approfondie de la documentation :

On utilise des **grilles flexbox pour agencer la page.**

Une grille Bootstrap est **hiérarchisée** de la manière suivante :

- Le **container** est l'**élément parent le plus haut dans la hiérarchie**. Il va **centrer et remplir horizontalement le contenu**, en **s'adaptant aux différents types de support**, selon six niveaux de taille d'écran.

Il enveloppe des rangées ou lignes ou rows.

Pour que le container utilise *toute la largeur de l'écran*, on le suffixe par **-fluid**.

Lorsque l'espace pour contenir les éléments **n'est plus suffisant**, il est possible **d'envoyer les éléments à la ligne**, afin qu'ils ne soient pas coupés à l'affichage, à l'aide de la propriété **wrap** :

- Nota : Il faudra dans ce cas préciser la propriété d-flex :
 - d-flex flex-nowrap
 - d-flex flex-wrap
 - flex-wrap-reverse.
- **Une row**, enveloppée dans un container, va permettre, elle-même, **d'envelopper des colonnes** (12 col par row).

Ce dans le but de pouvoir :

- **Aligner les colonnes qu'elle contient**, c'est à dire à *quelles position dans sa hauteur ces colonnes vont s'aligner*, à l'aide de la propriété **align-items ***:
 - **align-items-start**
 - **align-items-center**
 - **align-items-end**
- **Répartir l'espace horizontal entre les col qu'elle contient**, à l'aide de la propriété **justify-content-*** :
 - **justify-content-start**
 - **justify-content-center**
 - **justify-content-end**
 - **justify-content-around**
 - **justify-content-between**
 - **justify-content-evenly**
- Les **colonnes** (col), enveloppées par la row, vont permettre in fine de **placer notre contenu**.

A la manière de la propriété align-items-* pour les row, **align-self-*** permet le **même mécanisme à échelle individuelle** pour chaque colonne dans la row :

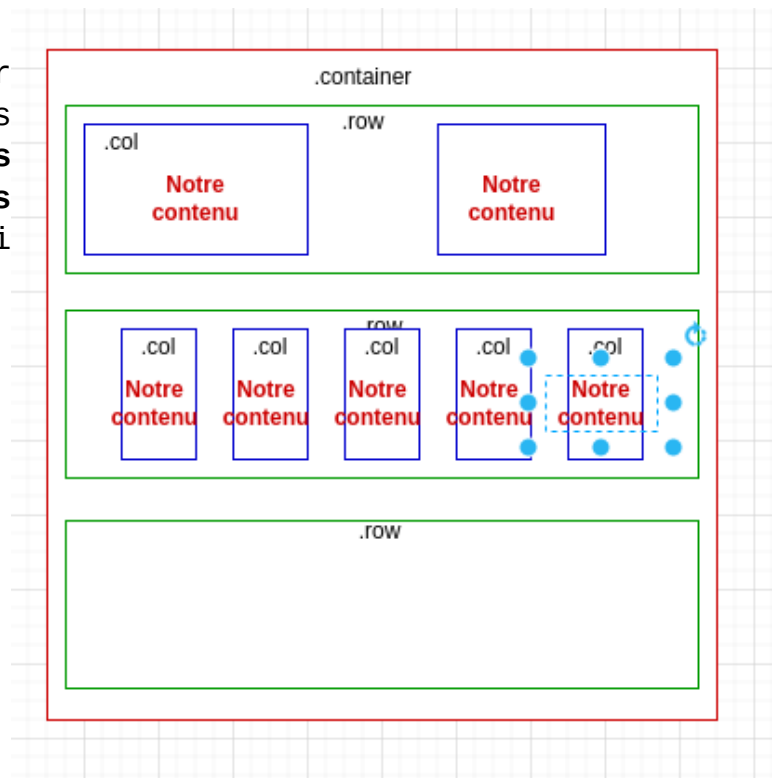
- **col align-self-start**
- **col align-self-center**
- **col align-self-end**

Nota : attention de bien faire attention que *align-self-** est une propriété dédiéees aux col.

Ce petit exemple montre comment disposer les éléments facilement avec bootstrap :

```
<div class="container text-center">
  <div class="row align-items-center justify-content-between">
    <div class="col align-self-center">
      1 of 2
    </div>
    <div class="col align-self-start">
      2 of 2
    </div>
  </div>
  <div class="row">
    <div class="col">
      1 of 3
    </div>
    <div class="col">
      2 of 3
    </div>
    <div class="col">
      3 of 3
    </div>
  </div>
</div>
```

Nota : Je remarque que sur le site de Bootstrap tous les exemples montrent **des classes associées à des div**. Nous procéderons ainsi à l'avenir.



6.3.2. Codage du head commun aux vues (layout .pug)

6.3.2.1. Particularités rencontrées

6.3.2.2. Liaison des dépendances

Elles sont réalisées à l'aide de la balise `link`, où l'attribut `rel` permet de spécifier la nature de la relation entre les ressources.

- Nos feuilles de styles posséderont l'attribut 'stylesheet'
- Nos scripts js l'attribut 'javascripts'
- Notre favicon 'shortcut icon'

Si elles sont locales, nos dépendances seront placées dans le dossier 'public' du projet.

Nous créons donc un raccourci vers ce dossier, dans notre application Express (`app.js`) :

```
app.use(express.static(path.join(__dirname, "public")));
```

Il pourra ainsi être utilisé dans nos middlewares.

Nota : Nous créons aussi, mais avec `app.set`, le chemin vers le dossier des vues

```
app.set("views", path.join(__dirname, "views"));
```

Nous pouvons à présent lier nos dépendances de la façon suivante :

```
link(rel='stylesheet', href='/stylesheets/style.css')
```

Afin de garder le même schéma, nous avons placés nos bibliothèques bootstrap dans le dossier public.

6.3.2.3. Code source de layout.pug

```
doctype html
html
  head
    title= title
    link(rel='stylesheet', href='/stylesheets/style.css')
    link(rel="stylesheet", href="https://fonts.googleapis.com/css2?
family=Gloria+Hallelujah&display=swap")
    link(rel='stylesheet', href='/stylesheets/bootstrap.min.css')
    link(rel='stylesheet', href='/stylesheets/bootstrap.css.map')
    link(rel='stylesheet', href='/stylesheets/bootstrap.min.css.map')
    link(rel='javascripts', href='/javascripts/bootstrap.min.js')
    link(rel="shortcut icon", href="/images/favicon.ico")
  body
    block content
```

6.3.3. Codage de la page d'authentification de l'administrateur(index.pug)

```
extends layout
block content
  .container.text-center
    img(src="./images/logos/logo.png" class="text-center mb-2")
    h1 PIZZAHUTTE
    .row.align-items-center
    .row.mt-3
    form
      .mb-3
      .row
      .col-3.bg-white
        input#exampleInputEmail1.form-control(type='email', aria-
describedby='emailHelp' class="col")
      .col-3.bg-white
        label.form-label(for='exampleInputEmail1' class="col") Email
      .mb-3
      .row
      .col-3.bg-white
        input#exampleInputPassword1.form-control(type='password' class='col')
      .col-3.bg-white
        label.form-label(for='exampleInputPassword1') Mot de passe
      .mb-3
      a(href='#')
        button.btn.btn-primary.mb-2(type='submit') Se connecter
      a.text-decoration-none(href='#')
        p mot de passe oublié ?
      .mt-5
      a.text-decoration-none.mt-2(href='https://github.com/atsuhikoMochizuki')
        img(src="./images/logos/github_logo_xsmall.png" alt="")
        p https://github.com/atsuhikoMochizuki
        p @Atsuhiko_Mochizuki - 2023 - Tous droits réservés
```

6.3.4. Aperçu côté Front



Email

Mot de passe

Se connecter

[mot de passe oublié ?](#)



<https://github.com/atsuhikoMochizuki>

@Atsuhiko_Mochizuki - 2023 - Tous droits réservés

6.4. FPA4 : s'authentifier

7. Annexes

7.1. Trop de temps perdu par une compréhension incomplète de l'outil BootStrap. Prenons le temps de remédier à ça.

7.1.1. Donner la capacité à un élément de pouvoir modifier ses dimensions afin de remplir l'espace disponible dans son conteneur : la propriété Flex.

La propriété flex permet de rendre un élément **flexible**, c'est à dire la faculté de **pouvoir modifier ses dimensions afin de remplir l'espace disponible de son conteneur**.

En effet, *quand il reste de l'espace restant*, si on **préfère que les éléments s'étirent pour l'occuper**,

il nous faut une méthode pour distribuer cet espace parmi les éléments.

3 propriétés à affecter aux éléments permettent de pouvoir gérer ceci

- **flex-basis**
- **flex-grow**
- **flex-shrink**

Avec ces propriétés, les éléments deviennent flexibles, ils peuvent être **étirés** ou **réduits** afin **de ne pas dépasser de leur conteneur**.

Pour autant ce sont des outils bas-niveau que dans la pratique, on utilise rarement et nous n'allons donc pas rentrer dans les détails ici.

On utilise beaucoup plus la propriété raccourcie **flex**.

Cette propriété permet également d'utiliser **des valeurs synthétiques qui couvrent la majorité des scénarios**.

Vous verrez souvent ces valeurs utilisées dans les tutoriels et, dans de nombreux cas, celles-ci suffiront :

- **auto**

L'élément **affublé de la propriété flex possédant elle-même la valeur « auto »** est dimensionné selon ses propriétés width et height

MAIS :

- **peut grandir pour absorber l'espace libre disponible dans le conteneur flexible**
- **rétrécir à sa taille minimale pour rentrer dans le conteneur.**

- **Initial**

L'élément **affublé de la propriété flex possédant elle-même la valeur « initial »** est dimensionné selon ses propriétés *width* et *height*.

Si besoin :

- l'élément **rétrécit à sa taille minimale pour rentrer dans le conteneur mais il ne grandira pas s'il y a de l'espace disponible dans ce conteneur.**

- **None**

L'élément **affublé de la propriété flex possédant la valeur elle-même la valeur « none »** est dimensionné selon ses propriétés *width* et *height*.

Il n'est pas flexible

Il ne peut ni rétrécir ni grandir selon l'espace du conteneur flexible.

On peut également mettre :

- **un nombre positif sans unité**

Par défaut, les éléments flexibles ne rétrécissent pas en dessous en dessous de la taille minimale du conteneur.

Pour modifier ce comportement il faudra paramétrer *min-width* ou *min-height*.

- **une valeur de largeur valide (width) :**

ex :

- `flex: 10em;`
- `flex: 30px;`

- flex: content;
- flex : auto ;
- flex : initial

Et on en restera là pour l'instant.

On peut trouver sur ce lien <https://developer.mozilla.org/fr/docs/Web/CSS/flex>, un exemple interactif permettant de visualiser instantanément ces concepts.

Il est important de bien comprendre que la propriété flex s'applique à l'élément, dans sa façon de gérer l'espace disponible dans son conteneur.

Nous avons, pour compléter notre panoplie, des containers flexibles qui vont nous permettre de facilement organiser spatialement les éléments au sein d'une zone visuelle à traiter.

On les appelle les **flexbox**. **Ce sont des boîtes flexibles.**

7.1.2. Les concepts de base des conteneurs flexibles, les FLEXBOX

Le module des boîtes flexibles, aussi appelé « **flexbox** », a été conçu comme un **modèle de**

- **DISPOSITION.**

Nous travaillons ici de façon unidimensionnelle. Une flexbox gère une dimension à la fois : **Elle est soit une colonne, soit une ligne.**

- **RÉPARTITION DE L'ESPACE**

Les flexbox vont permettre :

- d'aligner les éléments qu'elle contiennent.
- de distribuer l'espace entre ces derniers.

7.1.3. Les deux axes des boîtes flexibles flexbox

Lorsqu'on travaille avec les boîtes flexibles, **deux axes** interviennent :

- **l'axe principal** (main axis en anglais)

Il est défini par la propriété **flex-direction**

- **l'axe secondaire** (cross axis en anglais).

C'est l'axe qui est **perpendiculaire à l'axe principal**.

Tout ce que nous manipulons avec les boîtes flexibles fera référence à ces axes.

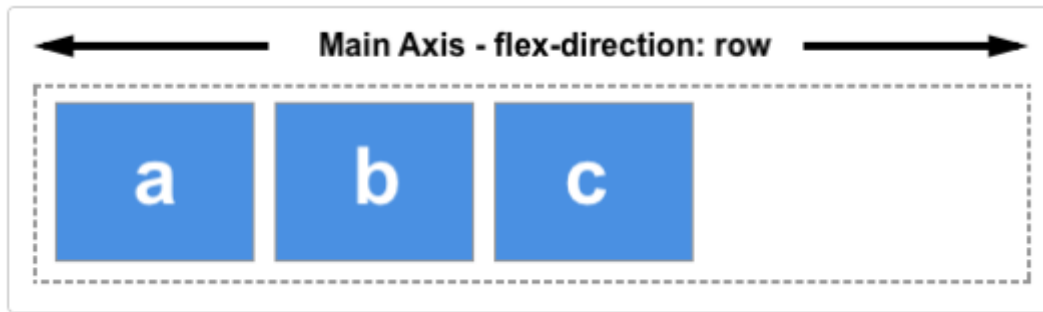
7.1.3.1. *L'axe principal*

L'axe **principal** est défini par la propriété **flex-direction** qui peut prendre quatre valeurs :

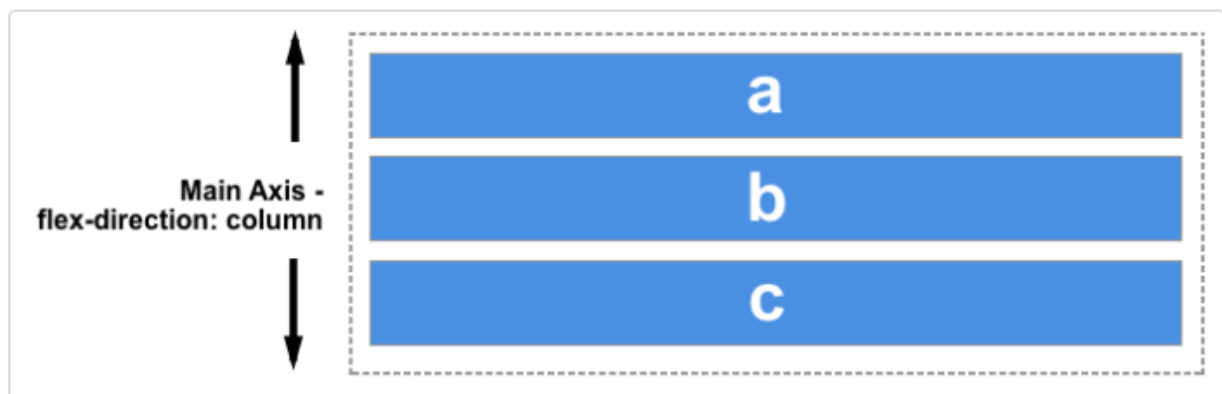
- **row**
- **row-reverse**
- **column**
- **column-reverse**
-

Si on choisit la valeur row ou row-reverse, l'axe principal sera aligné avec la direction « en ligne » (inline direction).

C'est la direction logique qui suit le sens d'écriture du document.

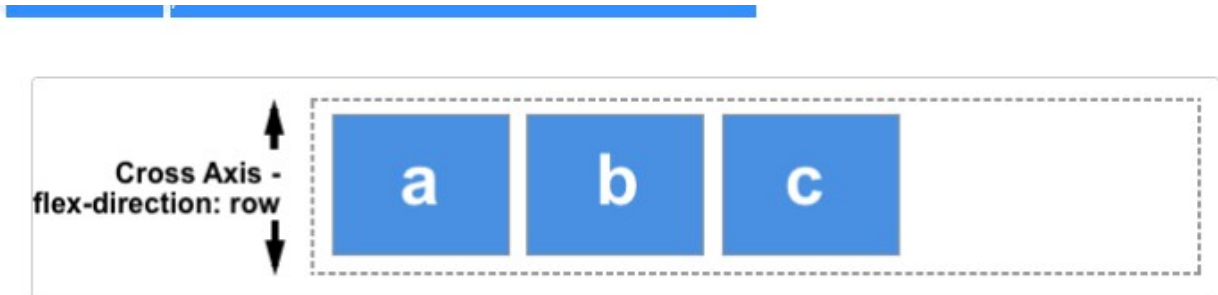


Si on choisit la valeur column ou column-reverse, l'axe principal suivra la direction de bloc (block direction) et progressera le long de l'axe perpendiculaire au sens d'écriture.

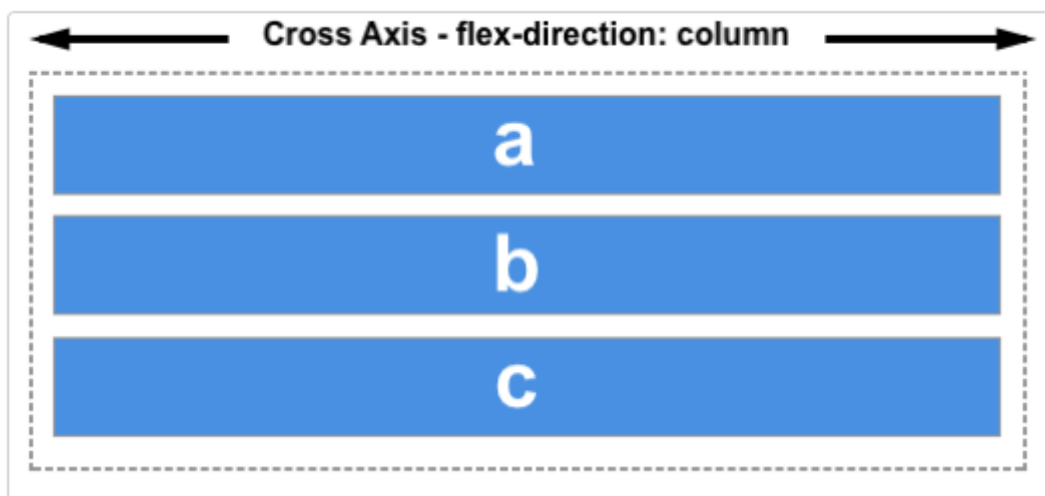


7.1.3.2. L'axe secondaire (cross axis)

L'axe secondaire est **perpendiculaire à l'axe principal**. Ainsi, si *flex-direction* vaut *row* ou *row-reverse*, l'axe secondaire suivra l'axe des colonnes.



Si l'axe principal est *column* ou *column-reverse*, l'axe secondaire suivra celui des lignes (horizontales).



Comprendre les liens entre les différents axes est crucial lorsqu'on commence à aligner/justifier des éléments flexibles sur un axe ou l'autre grâce aux fonctionnalités et propriétés des boîtes flexibles.

7.1.4. Créer un conteneur flexible sur plusieurs lignes avec flex-wrap

Bien que le modèle des boîtes flexibles soit organisé sur une dimension, **il est possible d'organiser les éléments flexibles afin que ceux-ci s'étendent sur plusieurs lignes ou colonnes (plutôt que de dépasser).**

Lorsque c'est le cas, chaque nouvelle ligne ou colonne **agit comme un nouveau conteneur flexible**. La distribution de l'espace sur cette ligne/colonne ne tiendra pas compte des autres lignes/colonnes.

Pour obtenir ce « **passage à la ligne** »,

on ajoute la propriété flex-wrap avec la valeur wrap

Désormais, *si les éléments sont trop grands pour tenir sur une seule ligne, ils passeront sur une autre ligne.*

Si en revanche on modifie la valeur avec **nowrap** (qui correspond à la valeur initiale), **les éléments seront rétrécis pour tenir sur une ligne** (car les valeurs initiales des boîtes flexibles permettent aux éléments d'être ainsi redimensionnés).

Si on utilise nowrap et que les éléments ne peuvent pas être redimensionnés (ou pas suffisamment), cela causera un *dépassement*.

7.1.5. La propriété raccourcie flex-flow

Il est possible de synthétiser les propriétés

- flex-direction
- flex-wrap

avec la propriété raccourcie **flex-flow** :

- **La première valeur de cette propriété sera utilisée pour flex-direction**
- **la seconde pour flex-wrap.**

Exemple :

```
.box {  
    display: flex;  
    flex-flow: row wrap;  
}
```

7.1.6. Alignement, justification et distribution de l'espace disponible entre les éléments

Comme nous l'avons vu, Une fonctionnalité majeure des boîtes flexibles est de permettre l'alignement et la justification des éléments le long des axes principal et secondaire tout en distribuant l'espace entre les éléments flexibles.

7.1.6.1. *align-items*

La propriété **align-items** permet **d'aligner les éléments le long de l'axe secondaire**.

La valeur initiale de cette propriété est **stretch**, ce qui explique pourquoi, par défaut, les éléments flexibles sont étirés sur l'axe perpendiculaire afin d'avoir la même taille que l'élément le plus grand dans cet axe (qui définit la taille du conteneur sur cet axe).

Les valeurs de align-items sont les suivantes :

- **stretch**
- **flex-start**
- **flex-end**
- **center**

7.1.6.2. *justify-content*

La propriété **justify-content** est utilisée afin

d'aligner les éléments le long de l'axe principal (dont la direction définie par flex-direction).

Les valeurs possibles sont :

- **flex-start**

C'est la valeur par défaut. Elle place les éléments à partir de la ligne de début du conteneur sur l'axe principal.

- **flex-end**

permet de les placer vers la fin

- **center**

permet de les centrer le long de l'axe principal.

- **space-between**

permet de répartir l'espace disponible de façon égale entre chaque élément.

- **space-around**

L'espace sera réparti autour des éléments, y compris au début et à la fin.

Nota : il y aura alors un demi espace à la fin et au début.

- **Space-evenly.**

Si on souhaite que l'espace soit également réparti et qu'il y ait un espace entier au début et à la fin.

7.1.7. Saisir pleinement l'outil Bootstrap, un investissement qui sera rapidement payant à court terme : allons-y

7.1.7.1. Le système de grilles de Bootstrap

La *puissante* **grille flexbox** permet de construire des mises en page de toutes formes et tailles *grâce à* **un système de douze colonnes**, six niveaux réactifs par défaut, des variables et mixins Sass et des dizaines de classes prédéfinies.

Le système de grille de Bootstrap utilise **une série de conteneurs, de lignes et de colonnes pour mettre en page et aligner le contenu.**

Il est construit avec flexbox et est entièrement réactif.

La grille prend en charge **six points de rupture réactifs** :

- Extra small (xs)
- Small (sm)
- Medium (md)
- Large (lg)
- Extra large (xl)
- Extra extra large (xxl)

Comme indiqué ci-dessus, chacun de ces points d'arrêt possède son propre conteneur, un préfixe de classe unique et des modificateurs. Voici comment la grille change en fonction de ces points d'arrêt :

	xs <576px	sm ≥576px	md ≥768px	lg ≥992px	xl ≥1200px	xxl ≥1400px
Container <i>max-width</i>	None (auto)	540px	720px	960px	1140px	1320px
Class prefix	<i>.col-</i>	<i>.col-sm-</i>	<i>.col-md-</i>	<i>.col-lg-</i>	<i>.col-xl-</i>	<i>.col-xxl-</i>
# of columns	12					
Gutter width	1.5rem (.75rem on left and right)					
Custom gutters	Yes					
Nestable	Yes					
Column ordering	Yes					

7.1.7.2. Les containers (.container)

Le type **container** est un élément **fondamental** de Bootstrap.

Il **contient**, **rembourse** et **aligne** votre le contenu **selon l'appareil** (smartphone, tablette, desktop...) ou le viewport de navigateur.

Par défaut, il est **réactif à largeur fixe** : sa **largeur maximale change à chaque point point de rupture**.

Par défaut, un container contient des **marges**, qu'il est **possible d'enlever en le suffixant avec -fluid**. Il prendra alors toute la largeur du contenu.

Les conteneurs centrent et remplissent horizontalement le contenu.

On utilise:

- **.container** pour une largeur en pixels réactive
- **.container-fluid** pour une largeur de 100 % sur tous les écrans et appareils,
- **ou un conteneur réactif** (par exemple, **.container-md**) pour une combinaison de largeurs fluides et en pixels.

7.1.7.3. Les lignes (.row)

Les lignes sont des enveloppes pour les colonnes.

Chaque colonne dispose d'un remplissage horizontal (appelé **gouttière**) qui permet de **contrôler l'espace entre les colonnes**.

7.1.7.4. Les colonnes(.col)

Les colonnes sont incroyablement flexibles.

Il existe **12 modèles de colonnes par ligne**, ce qui permet de créer différentes combinaisons d'éléments qui s'étendent sur un nombre illimité de colonnes.

Les classes de colonnes indiquent le nombre de colonnes de modèle à couvrir (par exemple, col-4 couvre quatre colonnes).

Les largeurs sont définies en **pourcentages** afin d'avoir toujours le **même dimensionnement relatif**.

Les colonnes s'appuient sur l'architecture flexbox de la grille.

Cette dernière permet de **modifier des colonnes individuelles et des groupes de colonnes au niveau de la ligne**.

Lors de la création de grilles, tout le contenu est placé dans des colonnes.

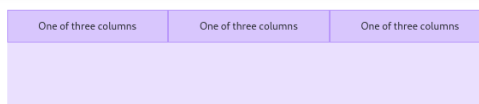
La hiérarchie de la grille Bootstrap va du conteneur à la ligne, à la colonne et à votre contenu.

```
<div class="container text-center">
  <div class="row">
    <div class="col">
      1 of 2
    </div>
    <div class="col">
      2 of 2
    </div>
  </div>
  <div class="row">
    <div class="col">
      1 of 3
    </div>
    <div class="col">
      2 of 3
    </div>
    <div class="col">
      3 of 3
    </div>
  </div>
</div>
```

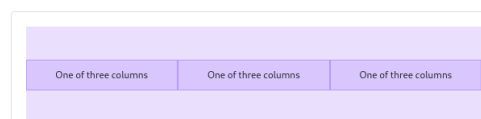
7.1.7.4.1. Alignement vertical

On modifie l'alignement vertical à l'aide de l'une des classes responsive align-items-*

```
<div class="container text-center">
  <div class="row align-items-start">
    <div class="col">
      One of three columns
    </div>
    <div class="col">
      One of three columns
    </div>
    <div class="col">
      One of three columns
    </div>
  </div>
</div>
```



```
<div class="container text-center">
  <div class="row align-items-center">
    <div class="col">
      One of three columns
    </div>
    <div class="col">
      One of three columns
    </div>
  </div>
```



```

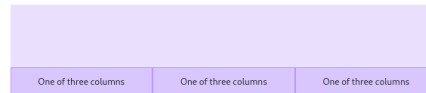
    </div>
    <div class="col">
      One of three columns
    </div>
  </div>
</div>

```

```

<div class="container text-center">
  <div class="row align-items-end">
    <div class="col">
      One of three columns
    </div>
    <div class="col">
      One of three columns
    </div>
    <div class="col">
      One of three columns
    </div>
  </div>
</div>

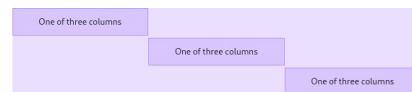
```



```

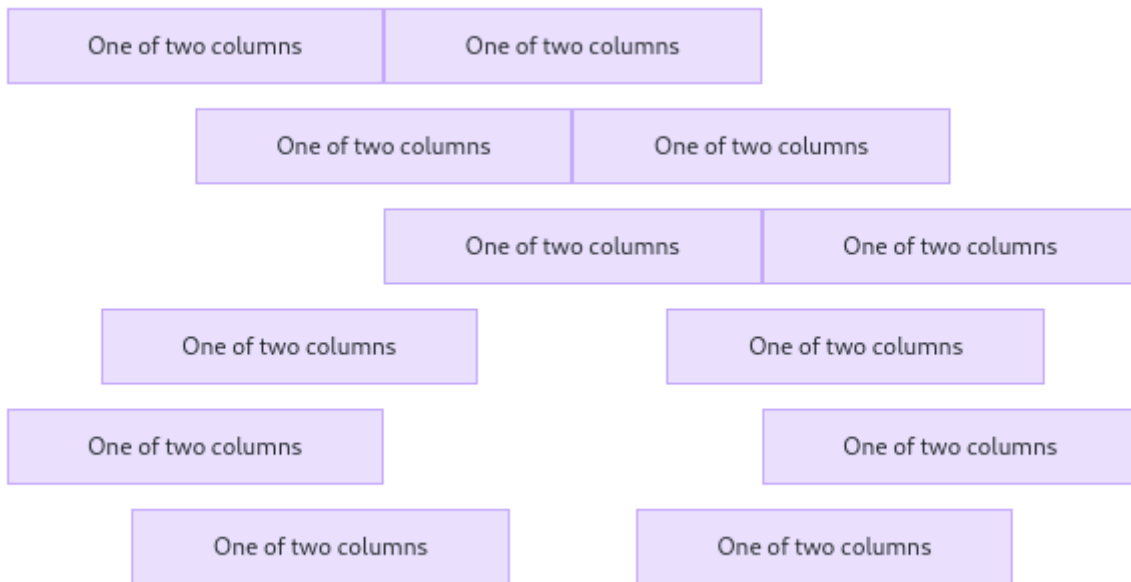
<div class="container text-center">
  <div class="row">
    <div class="col align-self-start">
      One of three columns
    </div>
    <div class="col align-self-center">
      One of three columns
    </div>
    <div class="col align-self-end">
      One of three columns
    </div>
  </div>
</div>

```



7.1.7.5. *alignement horizontal*

On modifie l'alignement horizontal à l'aide de l'une des classes réactives justify-content-*.



```

<div class="container text-center">
  <div class="row justify-content-start">
    <div class="col-4">
      One of two columns
    </div>
    <div class="col-4">
      One of two columns
    </div>
  </div>
  <div class="row justify-content-center">
    <div class="col-4">
      One of two columns
    </div>
    <div class="col-4">
      One of two columns
    </div>
  </div>
  <div class="row justify-content-end">
    <div class="col-4">
      One of two columns
    </div>
    <div class="col-4">
      One of two columns
    </div>
  </div>
  <div class="row justify-content-around">
    <div class="col-4">
      One of two columns
    </div>
    <div class="col-4">
      One of two columns
    </div>
  </div>
  <div class="row justify-content-between">
    <div class="col-4">
      One of two columns
    </div>
  </div>

```

```

</div>
<div class="col-4">
  One of two columns
</div>
</div>
<div class="row justify-content-evenly">
  <div class="col-4">
    One of two columns
  </div>
  <div class="col-4">
    One of two columns
  </div>
</div>
</div>

```

7.1.7.6. Les gouttières (.g)

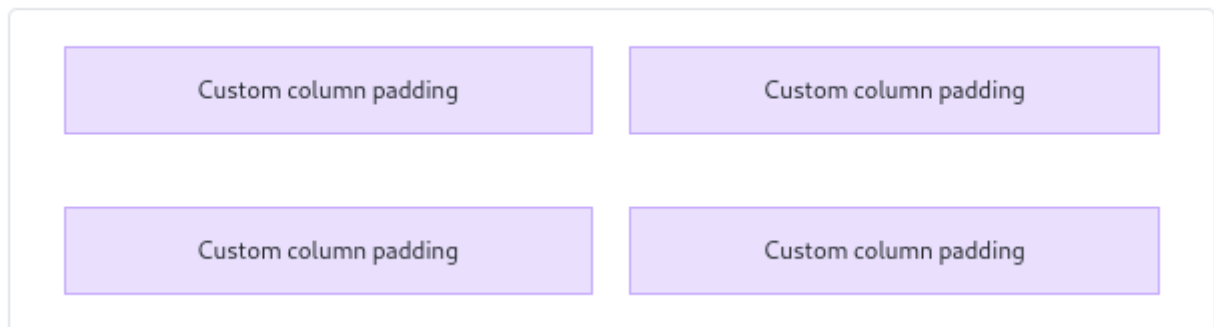
Les *gouttières* sont le **rembourrage d'espace entre les colonnes**, utilisé pour espacer et aligner le contenu de manière réactive dans le système de grille Bootstrap.

Les gouttières sont les espaces entre le contenu des colonnes, créés par le remplissage horizontal.

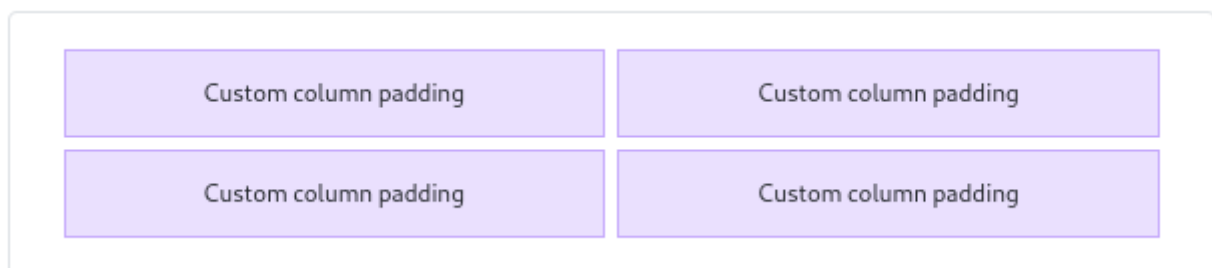
7.1.7.6.1. Gouttières horizontales



7.1.7.6.2. Gouttières verticales



7.1.7.6.3. Gouttières verticales ET horizontales



7.1.8. Utilitaires pour la mise en page

Pour accélérer le développement d'applications mobiles et réactives, Bootstrap comprend des dizaines de classes d'utilitaires permettant d'afficher, de masquer, d'aligner et d'espacer le contenu.

7.1.8.1. Modifier l'affichage

On utilise `display` pour basculer de manière réactive les valeurs courantes de la propriété `display`.

7.1.8.1.1. Les options FlexBox

Bootstrap est construit avec flexbox, **mais** l'affichage de tous les éléments **n'a pas été modifié en `display : flex`**, car cela ajouterait *de nombreuses surcharges inutiles et modifierait de manière inattendue des comportements clés du navigateur*.

La plupart de nos composants sont construits avec la **fonction flexbox activée**.

Si vous devez ajouter `display : flex` à un élément, faites-le avec **`.d-flex`** ou l'une des variantes réactives (par exemple, `.d-sm-flex`). Vous aurez besoin de cette classe ou de

cette valeur d'affichage pour permettre l'utilisation de nos utilitaires flexbox supplémentaires pour le dimensionnement, l'alignement, l'espacement, etc.

7.1.9. Typographie

7.1.9.1. Les titres

`.h1` through `.h6` classes are also available, for when you want to match the font styling of a heading but cannot use the associated HTML element.

h1. Bootstrap heading

h2. Bootstrap heading

h3. Bootstrap heading

h4. Bootstrap heading

h5. Bootstrap heading

h6. Bootstrap heading

HTML



```
<p class="h1">h1. Bootstrap heading</p>
<p class="h2">h2. Bootstrap heading</p>
<p class="h3">h3. Bootstrap heading</p>
<p class="h4">h4. Bootstrap heading</p>
<p class="h5">h5. Bootstrap heading</p>
<p class="h6">h6. Bootstrap heading</p>
```


Les éléments d'en-tête traditionnels sont conçus pour fonctionner au mieux dans le contenu de votre page. Lorsque vous avez besoin d'un titre qui se démarque, envisagez d'utiliser un titre d'affichage, c'est-à-dire un style de titre plus grand et légèrement plus expressif.

Display 1

Display 2

Display 3

Display 4

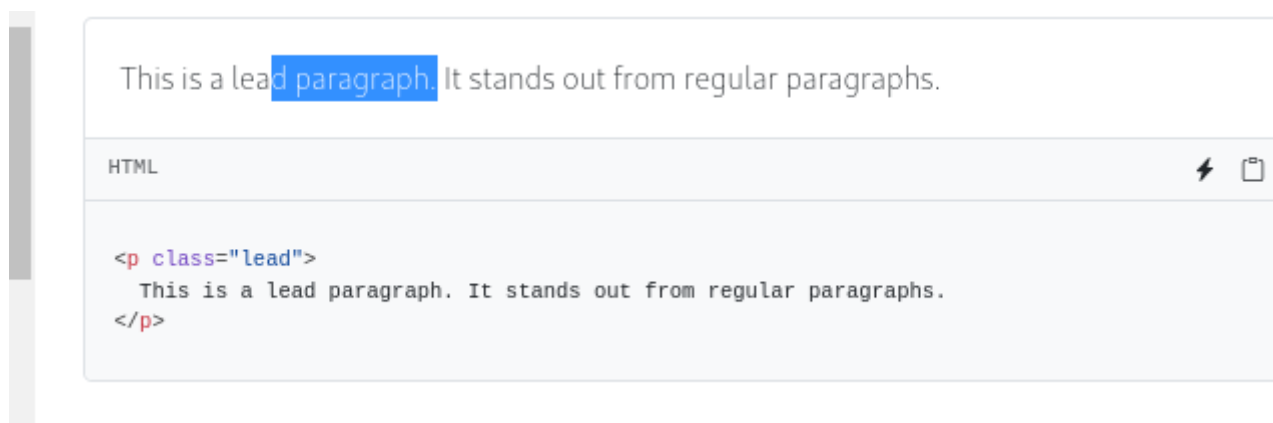
Display 5

Display 6

```
<h1 class="display-1">Display 1</h1>
<h1 class="display-2">Display 2</h1>
<h1 class="display-3">Display 3</h1>
<h1 class="display-4">Display 4</h1>
<h1 class="display-5">Display 5</h1>
<h1 class="display-6">Display 6</h1>
```

7.1.9.2. En tête

Faites ressortir un paragraphe en ajoutant un point d'orgue.



Éléments de texte en ligne

Style pour les éléments HTML5 en ligne les plus courants.

You can use the mark tag to highlight text.

~~This line of text is meant to be treated as deleted text.~~

~~This line of text is meant to be treated as no longer accurate.~~

This line of text is meant to be treated as an addition to the document.

This line of text will render as underlined.

This line of text is meant to be treated as fine print.

This line rendered as bold text.

This line rendered as italicized text.

```

<p>You can use the mark tag to <mark>highlight</mark> text.</p>
<p><del>This line of text is meant to be treated as deleted text.</del></p>
<p><s>This line of text is meant to be treated as no longer accurate.</s></p>
<p><ins>This line of text is meant to be treated as an addition to the
document.</ins></p>
<p><u>This line of text will render as underlined.</u></p>
<p><small>This line of text is meant to be treated as fine print.</small></p>
<p><strong>This line rendered as bold text.</strong></p>
<p><em>This line rendered as italicized text.</em></p>

```

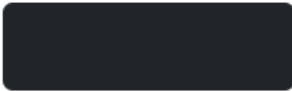


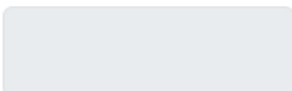

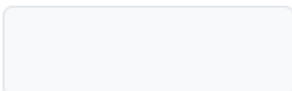

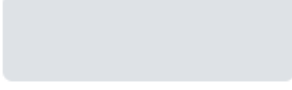
7.1.10. Les listes

```

<ul class="list-unstyled">
  <li>This is a list.</li>
  <li>It appears completely unstyled.</li>
  <li>Structurally, it's still a list.</li>
  <li>However, this style only applies to immediate child elements.</li>
  <li>Nested lists:
    <ul>
      <li>are unaffected by this style</li>
      <li>will still show a bullet</li>
      <li>and have appropriate left margin</li>
    </ul>
  </li>
  <li>This may still come in handy in some situations.</li>
</ul>

```

7.1.11. Les couleurs dans bootstrap 5

La description	Échange	Variables
Corps — Premier plan par défaut (couleur) et arrière-plan, y compris les composants.		<code>--bs-body-color</code> <code>--bs-body-color-rgb</code>
		<code>--bs-body-bg</code> <code>--bs-body-bg-rgb</code>
Secondaire — Utiliser le <code>color</code> option pour un texte plus léger. Utiliser le <code>bg</code> option pour les séparateurs et pour indiquer les états des composants désactivés.		<code>--bs-secondary-color</code> <code>--bs-secondary-color-rgb</code>
		<code>--bs-secondary-bg</code> <code>--bs-secondary-bg-rgb</code>
Tertiaire — Utiliser le <code>color</code> option pour un texte encore plus léger. Utiliser le <code>bg</code> option pour styliser les arrière-plans pour les états, les accents et les puits en vol stationnaire.		<code>--bs-tertiary-color</code> <code>--bs-tertiary-color-rgb</code>
		<code>--bs-tertiary-bg</code> <code>--bs-tertiary-bg-rgb</code>
Accent — Pour un texte à contraste plus élevé. Ne s'applique pas aux arrière-plans.		<code>--bs-emphasis-color</code> <code>--bs-emphasis-color-rgb</code>
Bordure — Pour les bordures, séparateurs et règles des composants. Utilisation <code>--bs-border-color-translucent</code> mélanger avec des arrière-plans avec un <code>rgba()</code> valeur.		<code>--bs-border-color</code> <code>--bs-border-color-rgb</code>

Primaire — Couleur du thème principal, utilisée pour les hyperliens, les styles de mise au point et les états actifs des composants et des composants.



--bs-primary
--bs-primary-rgb



--bs-primary-bg-subtle



--bs-primary-border-subtle

Texte

--bs-primary-text-emphasis

Succès — Couleur du thème utilisée pour les actions et informations positives ou réussies.



--bs-success
--bs-success-rgb



--bs-success-bg-subtle



--bs-success-border-subtle

Texte

--bs-success-text-emphasis

Danger — Couleur du thème utilisée pour les erreurs et les actions dangereuses.



--bs-danger
--bs-danger-rgb



--bs-danger-bg-subtle



--bs-danger-border-subtle

Texte

--bs-danger-text-emphasis

Avertissement — Couleur du thème utilisée pour les messages d'avertissement non destructifs.



`--bs-warning`
`--bs-warning-rgb`



`--bs-warning-bg-subtle`



`--bs-warning-border-subtle`

Texte

`--bs-warning-text-emphasis`

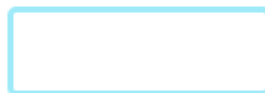
Info — Couleur du thème utilisée pour le contenu neutre et informatif.



`--bs-info`
`--bs-info-rgb`



`--bs-info-bg-subtle`

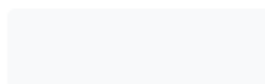


`--bs-info-border-subtle`

Texte

`--bs-info-text-emphasis`

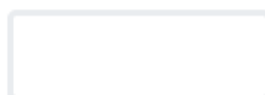
Lumière — Option de thème supplémentaire pour des couleurs moins contrastées.



`--bs-light`
`--bs-light-rgb`



`--bs-light-bg-subtle`



`--bs-light-border-subtle`

Texte

`--bs-light-text-emphasis`

Sombre — Option de thème supplémentaire pour des couleurs contrastées plus élevées.



--bs-dark
--bs-dark-rgb



--bs-dark-bg-subtle



--bs-dark-border-subtle

Texte

--bs-dark-text-emphasis

7.1.12.

7.1.13.