Atsuhiko Mochizuki

PIZZAHUTTE

Création d’une application Javascript FullStack complète

Administration d’une interface clientèle WEB dediée à une infrastructure de type Pizzéria

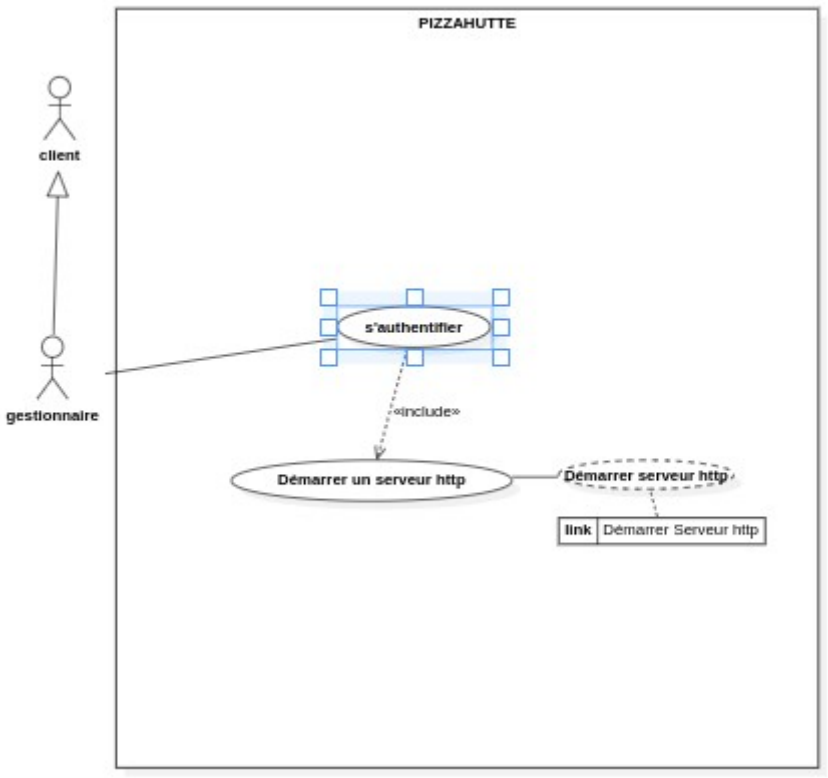
Projet individuel
Sous la direction de Rossi Oddet

diginamic FORMATION

Table des matières

1. ANALYSE ET CONCEPTION

1.1. Diagramme des cas d’utilisation



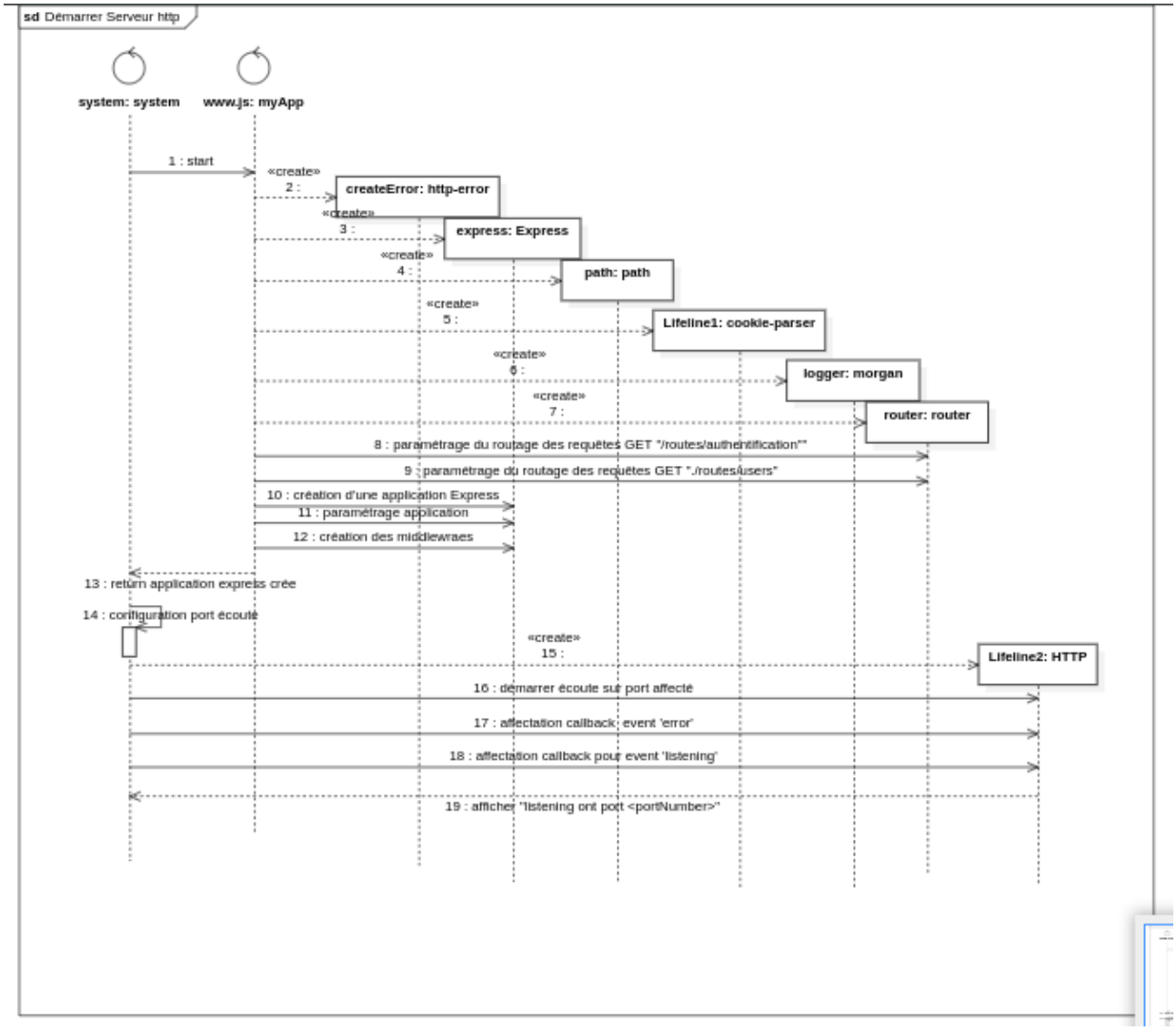
1.2. Analyse pas à pas des cas d'utilisation

1.2.1.S'authentifier

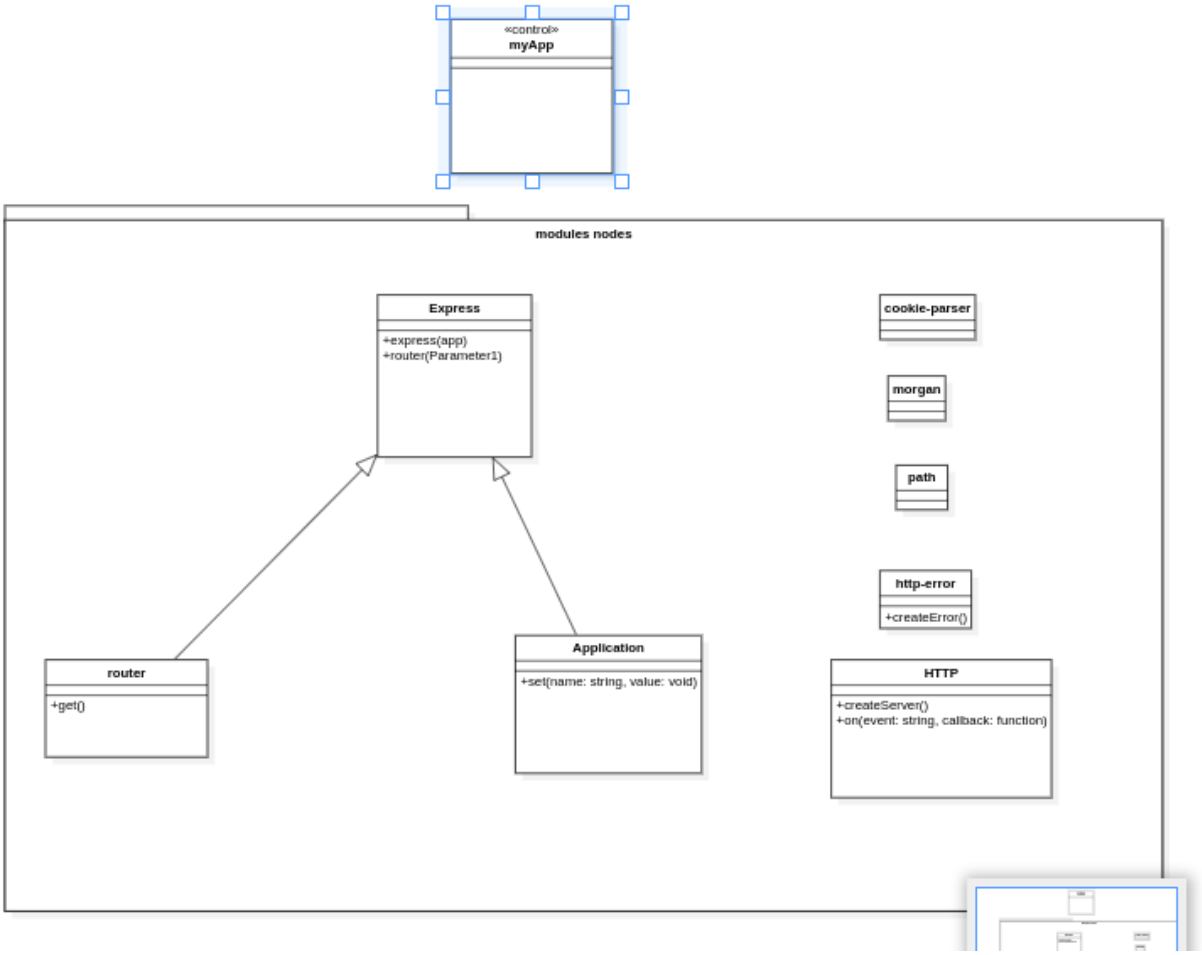
1.2.1.1. Démarrer un serveur HTTP

1.2.1.1.1. Premier jet

Diagramme de séquence



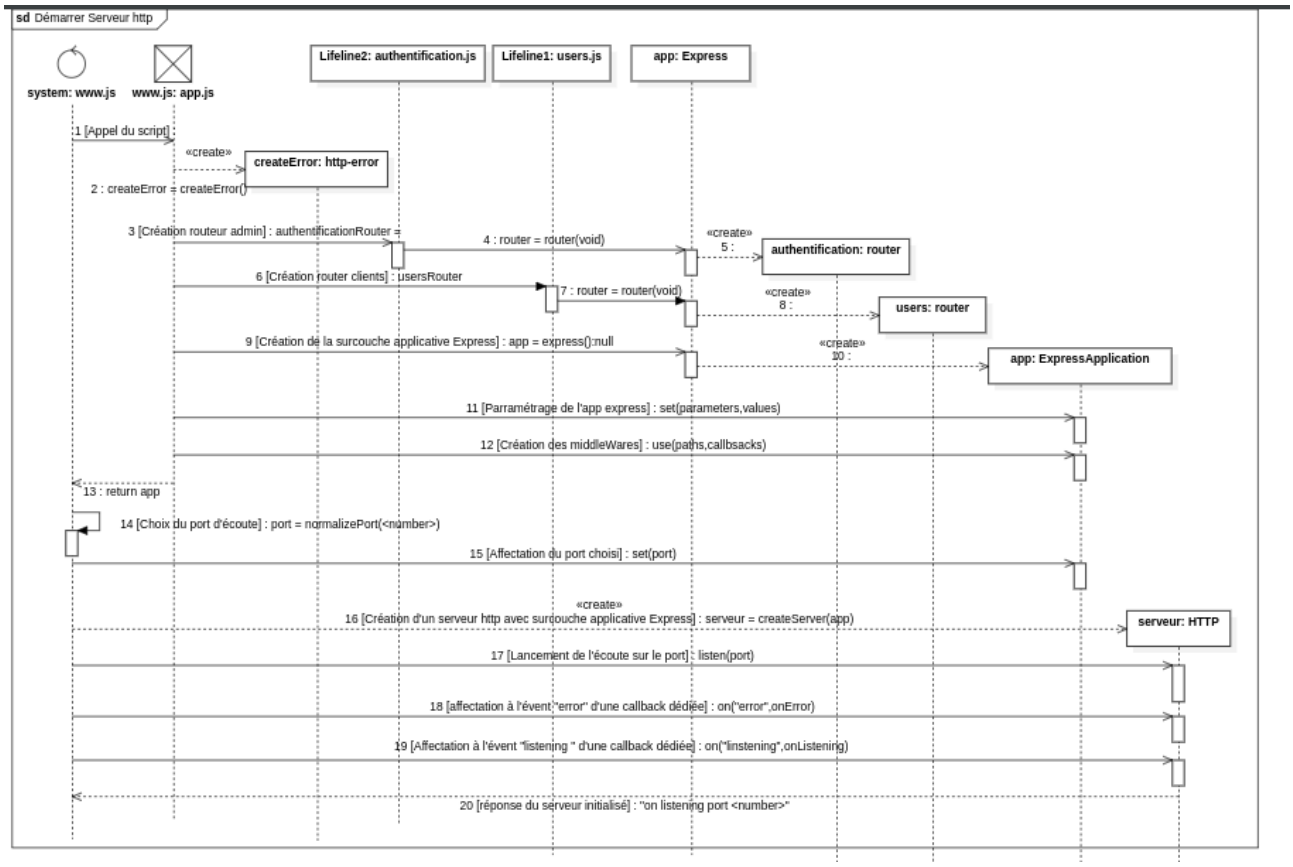
Déduction du diagramme des classes participantes

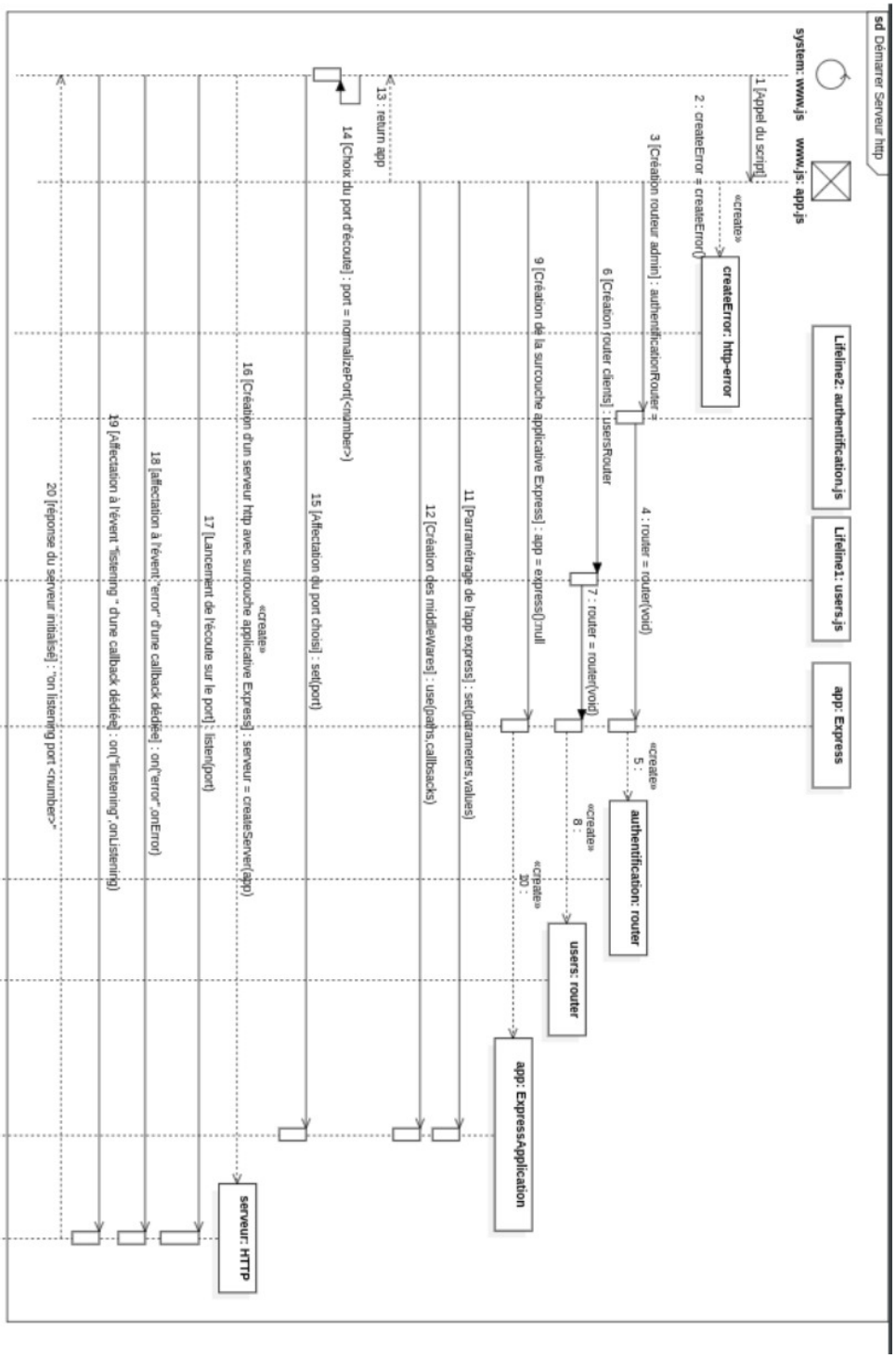


1.2.1.1.2. Second jet

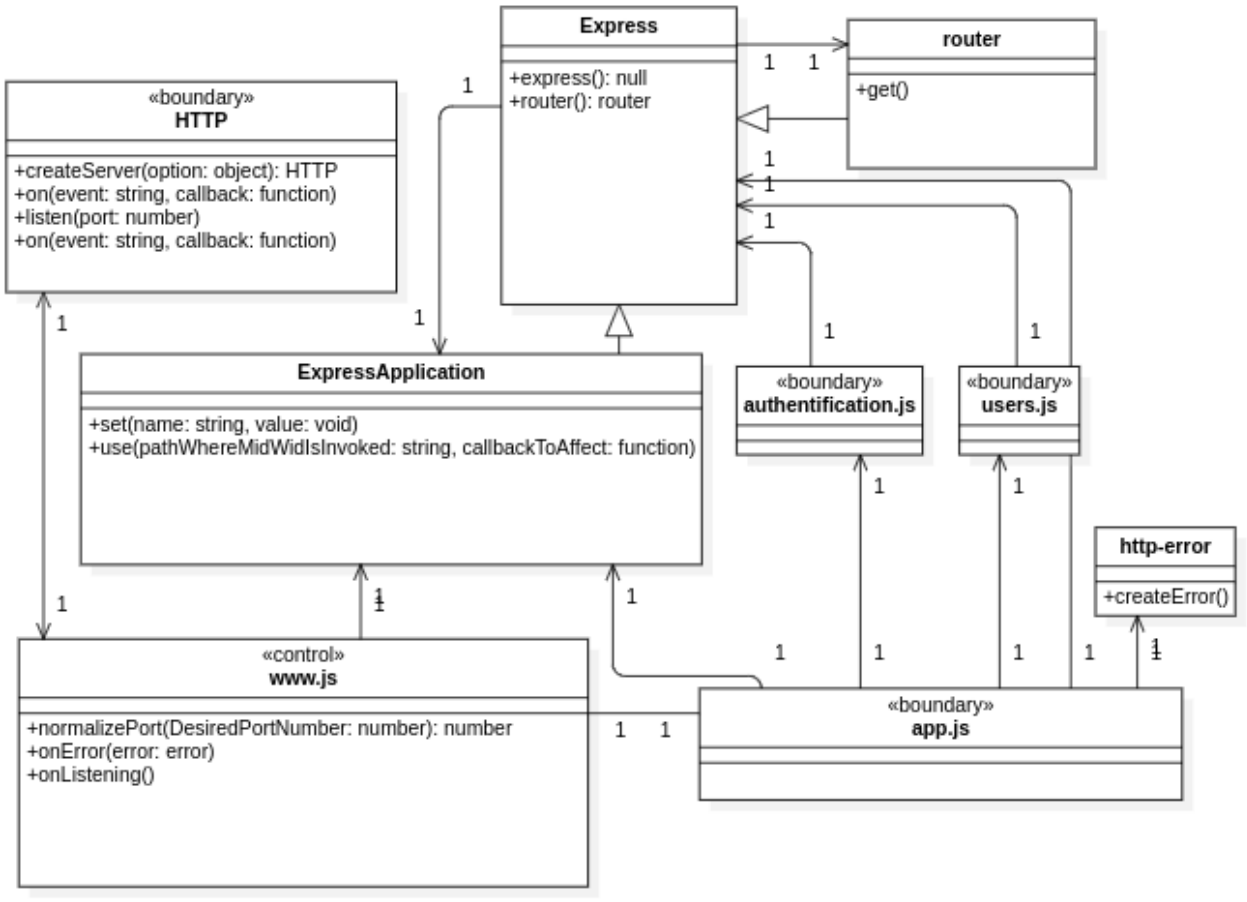
Nous allons affecter les méthodes des classes aux différentes interactions entre les objets de l’application

Diagramme de séquence corrigé et approfondi




sd Démarrer Serveur http
system: www.js
www.js: app.js
Lifeline2: authentification.js
Lifeline1: users.js
app: Express
1 [Appel du script]
«create»
createError: http-error
2 : createError = createError()
3 [Création routeur admin] : authentificationRouter =
4 : router = router(void)
«create»
5 :
authentification: router
6 [Création routeur clients] : usersRouter
7 : router = router(void)
«create»
8 :
users: router
9 [Création de la surcouche applicative Express] : app = express():null
«create»
10 :
app: Express-Application
11 [Parramétrage de l'app express] : set(parameters,values)
12 [Création des middleWares] : use(paths,callbacks)
13 : return app
14 [Choix du port d'écoute] : port = normalizePort(<number>)
15 [Affectation du port choisi] : set(port)
«create»
16 [Création d'un serveur http avec surcouche applicative Express] : serveur = createServer(app)
serveur: HTTP
17 [Lancement de l'écoute sur le port] : listen(port)
18 [affectation à l'event "error" d'une callback dédiée] : on("error",onError)
19 [Affectation à l'évent "listening" d'une callback dédiée] : on("listening",onListening)
20 [réponse du serveur initialisé] : "on listening port <number>"

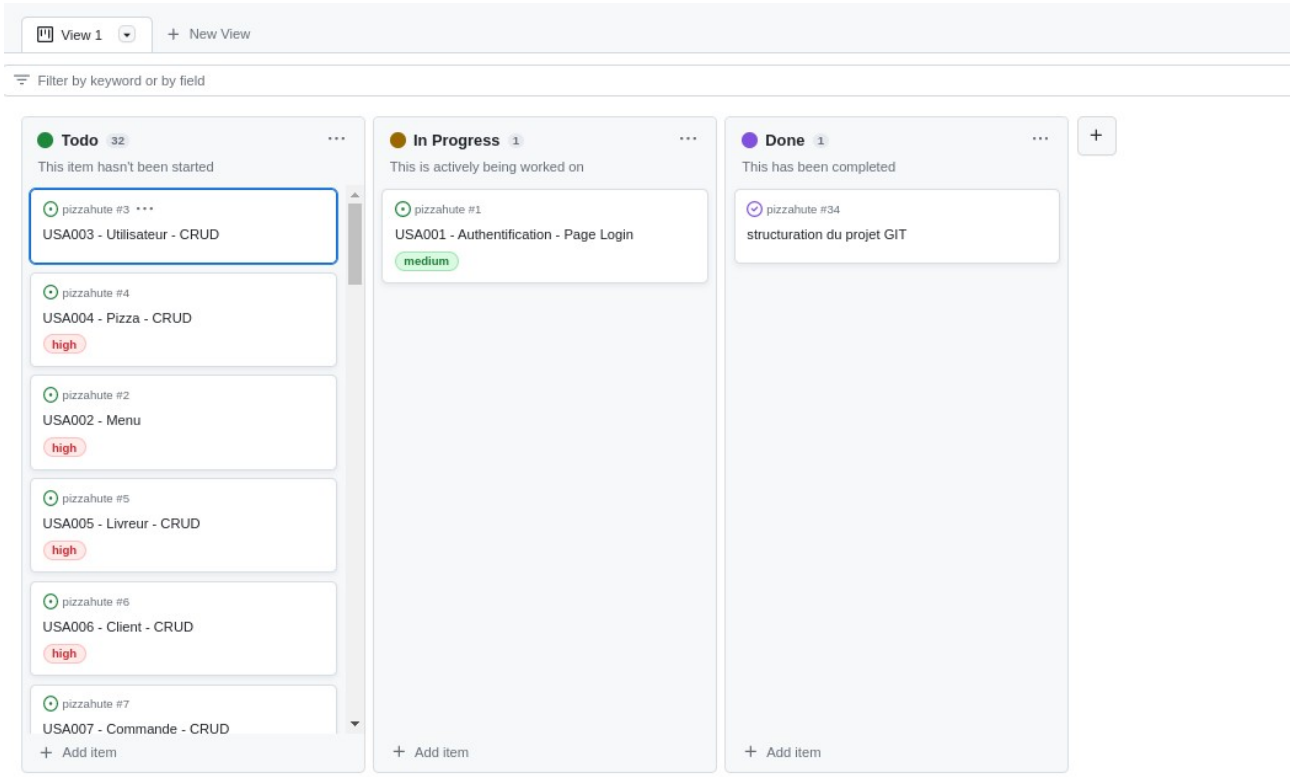
Déduction du diagramme des classes participantes



2. CHOIX DES OUTILS TECHNOLOGIQUES

3. ORGANISATION ET PLANIFICATION DU PROJET

3.1. Découpage des tâches sous la forme KANBAN


View 1
New View
Filter by keyword or by field
Todo 32
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
In Progress 1
This is actively being worked on
pizzahute #1
USA001 - Authentification - Page Login
medium
Add item
Done 1
This has been completed
pizzahute #34
structuration du projet GIT
Add item

4. MISE EN PLACE DU DEVELOPPEMENT

4.1. Mise en place de l’envirronement de travail

4.1.1. Structuration de l’arborescence du projet

4.1.2. Création du dépôt github et clonage du projet vierge

4.1.3. Création du projet node via npm

$ npm init -y

4.1.3.1. installation des dépendances

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

4.1.3.2. Paramétrage du fichier de configuration package.json

Affectation de nodemon à la commande start de l’objet script

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

4.1.3.2.2. Prise en charge des modules ECMAscript (import, export)

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

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

Les étapes que nous avons listées précedemment 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

  _____ _____ _____________      _    _ _    _ _______ _______ ______
 |  __ \_   _|___  /___  /   /\   | |  | | |  | |__   __|__   __|  ____|
 | |__) || |    / /   / /   /  \  | |__| | |  | |  | |     | |  | |__
 |  ___/ | |   / /   / /   / /\ \ |  __  | |  | |  | |     | |  |  __|
 | |    _| |_ / /__ / /__ / ____ \| |  | | |__| |  | |     | |  | |___
 |_|   |_____/_____/_____/_/    \_\_|  |_|\____/   |_|     |_|  |______|
 Générateur 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'\
033[1;32m"
DEBUG=$uservar:* npm start
```

5. CODAGE DE L’APPLICATION

5.1. Pf1 : Démarrer un serveur

5.1.1. 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);
}
```

5.1.2. 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 authentificationRouter = require("./routes/authentification.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("/", authentificationRouter);
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;
```

5.1.3.Script de routage de la page d’accueil authentification.js

```
var express = require("express");

var router = express.Router();



/* GET home page. */

router.get("/", function (req, res, next) {

  res.render("authentification", { title: "Pizzahutte" });

});



module.exports = router;
```

5.1.4. 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. A mettre en place

6.1.1. Vues à générer par le moteur de vue Pug

6.1.1.1. Layout.pug

```
doctype html
html
  head
    title= title
    link(rel='stylesheet', href='/stylesheets/style.css')
    link(rel='stylesheet', href='/bootstrap.min.css')
    meta(name="viewport" content="width=device-width, initial-scale=1.0")
  body
    block content
```

6.1.1.2. Authentification .pug

```
extends layout

block content
  header
    div(class="mx-auto mb-5")
      img(src="images/logo-no-background-small.png" class="img-fluid" alt="")
      h1(class="h1 text-center text-secondary") PIZZAHUTTE
  main
    div(class="container text-center")
      form(class="m-auto")
        // Email input
        .form-outline.mb-4
          input#form2Example1.form-control(type='email')
          label.form-label(for='form2Example1') Email
        // Password input
        .form-outline.mb-4
          input#form2Example2.form-control(type='password')
          label.form-label(for='form2Example2') Mot de passe
        // 2 column grid layout for inline styling
        div.d-flex.flex-row
        button.btn.btn-primary.btn-block.mp-10.text-center(type='button') Se
connecter
        .text-center.mt-3
        a.text-center(href='#!') Mot de passe oublié ?
```

6.1.1.3. Error.pug

```
extends layout

block content
  h1= message
  h2= error.status
  pre #{error.stack}
```