1.1 LES VARIABLES D'ENVIRONNEMENT

Il faut différencier le développement fait en environnement de test et lorsque l'application est en production.
Il est recommandé d'utiliser les variables d'environnement, qui seront différentes selon que le code est sur la machine du développeur ou sur le serveur.

1.1.1 Script d'exemple

Les variables d'environnement utiles commencent par process.env :

```
// config/config.js
'use strict'

// required environment variables
[
  'NODE_ENV',
  'PORT'
].forEach((name) => {
  if (!process.env[name]) {
    throw new Error(`Environment variable ${name} is missing`)
  }
})

// settings for the configuration
const config = {
  env: process.env.NODE_ENV,
  logger: {
    level: process.env.LOG_LEVEL || 'info',
    enabled: process.env.BOOLEAN ? process.env.BOOLEAN.toLowerCase() === 'true' : false
  },
  server: {
    port: Number(process.env.PORT)
  }
  // ...
}

module.exports = config
```

Il est fréquent de créer les variables d'environnement qui seront utilisées pour différencier où l'application fonctionnera, notamment, pour l'adresse du serveur de la base de données.

1.1.2 Mise en œuvre

Comme les variables d'environnement dépendent de l'environnement (et peuvent contenir des informations sensibles), il est recommandé de configurer le fichier **.gitignore** pour ne pas utiliser les fichiers .**env**.

Installer le gestionnaire de variables d'environnement :

```
npm install dotenv --save
```

il suffit ensuite d'inclure la ligne suivante dans le code principal :

```
require('dotenv').config();
```

et enfin, de créer un fichier .env à la racine du projet (sur chaque environnement de développement). Dans l'environnement de test, il ressemblera à ceci :

.env

```
NODE_ENV=development
DB_HOST=localhost
DB_USER=root
DB_PASSWORD=root
DB_NAME=testDB
```

Exemple pour l'environnement de production :

.env

```
NODE_ENV=production
DB_HOST=dbserver.interloop.org
DB_USER=admApp
DB_PASSWORD=apzoeiruty312$
DB_NAME=appDB
```

1.1.3 Intérêt

Outre le fait que les informations ne soient pas incluses directement dans le code, la variable `NODE_ENV` permet également de choisir de masquer certaines informations de débogage lorsque l'application plante (messages complets en développement et succinct en production).

Voici comment obtenir la liste des variables d'environnement dans une application :

```
console.log(process.env)
```

Et un extrait du résultat :

résultat

```
{
  ALLUSERSPROFILE: 'C:\\ProgramData',
  APPDATA: 'C:\\Users\\prRoumanet\\AppData\\Roaming',
  COMPUTERNAME: 'NCC-1701',
  ComSpec: 'C:\\WINDOWS\\system32\\cmd.exe',
  DriverData: 'C:\\Windows\\System32\\Drivers\\DriverData',
  FPS_BROWSER_APP_PROFILE_STRING: 'Internet Explorer',
  FPS_BROWSER_USER_PROFILE_STRING: 'Default',
  HOMEDRIVE: 'C:',
  HOMEPATH: '\\Users\\prRoumanet',
  JAVAFX_HOME: 'C:\\Program Files (x86)\\Java\\JavaFX',
  JAVA_HOME: 'C:\\Program Files (x86)\\Java\\jdk1.8.0_202',
  LANG: 'fr_FR.UTF-8',
  LOCALAPPDATA: 'C:\\Users\\prRoumanet\\AppData\\Local',
  LOGONSERVER: '\\\\NCC-1701',
  MOZ_PLUGIN_PATH: 'C:\\Program Files\\Tracker Software\\PDF Viewer\\Win32\\',
  NUMBER_OF_PROCESSORS: '4',
  OS: 'Windows_NT',
  Path: 'C:\\Program Files (x86)\\Common Files\\Oracle\\Java\\javapath;C:\\Program ' +
    'Files (x86)\\Intel\\iCLS Client\\;C:\\Program Files\\Intel\\iCLS ' +
    'Files\\Git\\cmd;C:\\Program Files ' +
    'Files\\nodejs;',
  PATHEXT: '.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JSE;.WSF;.WSH;.MSC;.CPL',
  PROCESSOR_ARCHITECTURE: 'AMD64',
  PROCESSOR_IDENTIFIER: 'Intel64 Family 6 Model 94 Stepping 3, GenuineIntel',
  PROCESSOR_LEVEL: '6',
  PROCESSOR_REVISION: '5e03',
  ProgramData: 'C:\\ProgramData',
  PROMPT: '$P$G',
  PUBLIC: 'C:\\Users\\Public',
  SESSIONNAME: 'Console',
  SystemRoot: 'C:\\WINDOWS',
  TEMP: 'd:\\Temp',
  TERM_PROGRAM: 'vscode',
  TERM_PROGRAM_VERSION: '1.41.0',
  TMP: 'd:\\temp',
  USERDOMAIN: 'NCC-1701',
  USERNAME: 'prRoumanet',
  USERPROFILE: 'C:\\Users\\prRoumanet',
  windir: 'C:\\WINDOWS'
}
```