Nest.js

- Création du projet Nest.js en TypeScript
- Ajouter des configurations - VSCode
- Outils supplémentaires

Création du projet Nest.js en TypeScript

> Nest v10 - `^10.0.0`
> RxJS v7 - `^7.8.1`

Projet Nest.js

La création du projet se fait en une seule ligne de commande :

```
nest new -l TypeScript -p npm --strict project-name
```

Cette ligne va créer tous les fichiers de base d'un projet Nest.js dans le dossier `project-name`. A partir de maintenant, tout sera fait en étant dans le dossier du projet : `project-name`.

> S'il faut créer le projet dans le dossier courant et pas dans un sous-dossier, il faut ajouter cet argument à la commande : `--directory .`

Pour tester que tout c'est bien passé, il suffit de lancer la commande suivante puis d'ouvrir un navigateur et de taper : `localhost:3000`.

```
npm run start:dev
```

Si le résultat est celui-ci, alors l'installation basique de Nest.js à fonctionné :

Hello World!

Ajouter des configurations - VSCode

> Cette page va lister quelques configurations pouvant être utiles en cas d'utilisation de VSCode en tant qu'IDE (*Integrated Development Environment*)

VSCode

Il y a 3 fichiers de VSCode qui peuvent être modifiés/créés dans le dossier `.vscode` : `extensions.json`, `launch.json` et `settings.json`

extensions.json

```
{
  "recommendations": [
    "pkief.material-icon-theme",
    "esbenp.prettier-vscode",
    "sonarsource.sonarlint-vscode",
    "ryanluker.vscode-coverage-gutters",
    "orta.vscode-jest",
    "pmneo.tsimporter",
    "archsense.architecture-view-nestjs"
  ]
}
```

settings.json

```
{
  "javascript.preferences.importModuleSpecifier": "relative",
  "typescript.preferences.importModuleSpecifierEnding": "minimal",
  "typescript.preferences.importModuleSpecifier": "relative",
  "javascript.preferences.importModuleSpecifierEnding": "minimal",
  "js/ts.implicitProjectConfig.module": "ES2022",
  "editor.formatOnSave": true,
```

```
  "editor.codeActionsOnSave": {
    "source.organizeImports": "always"
  },
  "files.associations": {
    "*.prettierrc": "json"
  },
  "jest.useDashedArgs": true,
  "jest.coverageFormatter": "GutterFormatter",
  "jest.runMode": "on-demand",
  "coverage-gutters.showLineCoverage": true,
  "coverage-gutters.showGutterCoverage": false,
  "coverage-gutters.showRulerCoverage": true
}
```

Node

package.json

Il faut installer :

- `npm i @types/node`
- `npm i -D ng-mocks webpack-bundle-analyzer`

Puis il faut ajouter, dans les scripts ;

```
"build": "nest build",
"format": "prettier --write \"src/**/*.ts\" \"test/**/*.ts\"",
"start": "nest start",
"start:dev": "npm start -- --watch",
"start:debug": "npm start:dev -- --debug",
"start:prod": "node dist/main",
"lint": "eslint \"{src,apps,libs,test}/**/*.ts\" --fix",
"test": "jest --verbose",
"test:watch": "npm test -- --watch",
"test:cov": "npm test -- --coverage --coverageReporters=lcov --coverageReporters=text --
coverageReporters=text-summary --coverageReporters=html",
"test:ci": "jest --ci",
"test:debug": "node --inspect-brk -r tsconfig-paths/register -r ts-node/register node_modules/.bin/jest --
runInBand",
"test:e2e": "jest --config ./test/jest-e2e.json",
```

```
"prepare": "husky",
"postversion": "node set-sonar-version.js && git add sonar-project.properties && git commit -m \"Updating project version\" && git push --tags && git push --all -n"
```

TypeScript Config

Dans le fichier `tsconfig.json`, il est possible et recommandé d'ajouter cette ligne dans la catégorie `compilerOptions`

```
"esModuleInterop": true,
"strict": true
```

Prettier

Il faut, pour le configurer, placer un fichier `.prettierrc` à la racine du projet. Voilà un exemple de ce qu'il pourrait contenir ;

```
{
  "singleQuote": true,
  "arrowParens": "avoid",
  "proseWrap": "always",
  "htmlWhitespaceSensitivity": "ignore",
  "bracketSameLine": true,
  "printWidth": 120,
  "useTabs": false,
  "jsxSingleQuote": true,
  "trailingComma": "all",

  "singleAttributePerLine": false,
  "semi": true,
  "tabWidth": 2,
  "bracketSpacing": true,
  "embeddedLanguageFormatting": "auto",
  "endOfLine": "lf",
  "insertPragma": false,
  "quoteProps": "as-needed",
  "requirePragma": false,
  "vueIndentScriptAndStyle": false
```

```
}
```

ESLint

Il faut ajouter dans le fichier `eslint.config.mjs`, dans les règles, celles-ci :

```
'@typescript-eslint/no-unsafe-call': 'warn',
'prettier/prettier': [
  'error',
  {
    endOfLine: 'auto',
  },
],
```

Outils supplémentaires

BCrypt.js

```
npm i bcryptjs
npm i -D @types/bcryptjs
```

ClassTransformer / ClassValidator

```
npm i class-transformer class-validator
```

Passport

```
npm i passport @nestjs/passport
```

MySQL

```
npm i mysql2
```

Swagger

```
npm i swagger-ui-express @nestjs/swagger
```

TypeORM

```
npm i typeorm @nestjs/typeorm
```

Winston

```
npm i winston
```

LintStaged

```
npm i -D lint-staged
```

JestJunit

```
npm i -D jest-junit
```

Husky

```
npm i -D husky
```