Identification de l'étudiant

- ZiadChaouche
- Ziad.chaouche.1@ens.etsmtl.ca
- AU41040
- Ziad.chaouche.1@ens.etsmtl.ca

Squelette pour un API simple dans Node, Express et TypeScript

Introduction

Ce squelette est proposé pour commencer les projets en LOG210. Il possède les qualités suivantes:

- il est simple pour les débutants en LOG210
  - il n'y a pas de framework pour le front-end ni pour la persistance, mais ça n'empêche pas d'ajouter ces dimensions.
  - il est seulement REST niveau 1, mais ça n'empêche pas de modifier l'API pour qu'il soit REST niveau 3.
- il est orienté objet (avec TypeScript)
- il contient des tests pour l'API (avec Jest et Supertest)
- il fait une séparation entre les couches présentation et domaine, selon la méthodologie de conception du cours LOG210 (Larman)
- il fournit une structure (en Bootstrap et Pug) permettant de gérer les connexions d'utilisateur et les vues selon le type d'utilisateur
- il fonctionne sur Windows 10 (et probablement d'autres systèmes d'exploitation avec Node)

D'où vient l'idée de base pour ce squelette?

Le code d'origine a été expliqué dans ce texte de blogue.

Dans le cadre du cours LOG210 de l'ÉTS, nous utilisons la méthodologie documentée par Craig Larman dans son livre *Applying UML and Patterns*. Ce livre documente beaucoup de principes avec des exemples en Java, qui n'est plus à la mode comme à l'époque où le livre a été écrit.

Pourtant, il est encore possible de suivre cette méthodologie avec des technologies modernes comme JavaScript, Node.js, surtout en utilisant TypeScript. Cependant, il n'est pas évident de trouver des exemples de ces technologies qui respectent les éléments clés de la méthodologie de Larman: la séparation des couches (présentation, domaine) avec les opérations système et les classes du domaine.

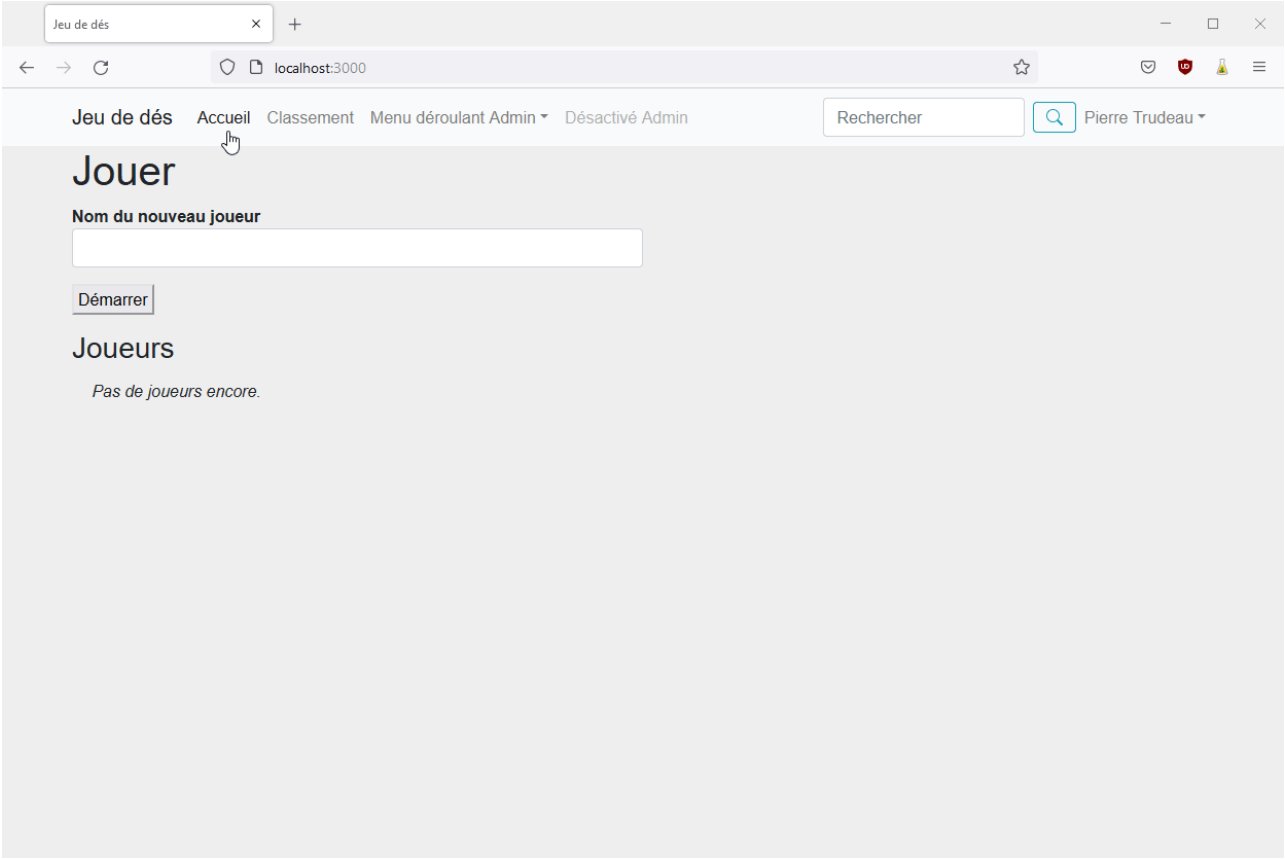
Ce squelette montre ces aspects importants, dans le contexte du *Jeu de dés*, qui est l'exemple utilisé dans le chapitre 1 du livre du cours. Nous avons modifié l'exemple pour le rendre un peu plus complexe (plusieurs opérations système). Les diagrammes (faits avec PlantUML) sont présentés plus bas dans la partie Artefacts.

L'éditeur Visual Studio Code est très utile, mais n'est pas nécessaire avec ce squelette.

Voulez-vous utiliser ce squelette?

1. (Créer une fork et) Cloner
2. Installer node
3. Installer les dépendances node - `npm install`
4. Compiler - `npm run build`
5. Lancer serveur de développement - `npm start`
6. Accéder à la page template de l'application - http://localhost:3000

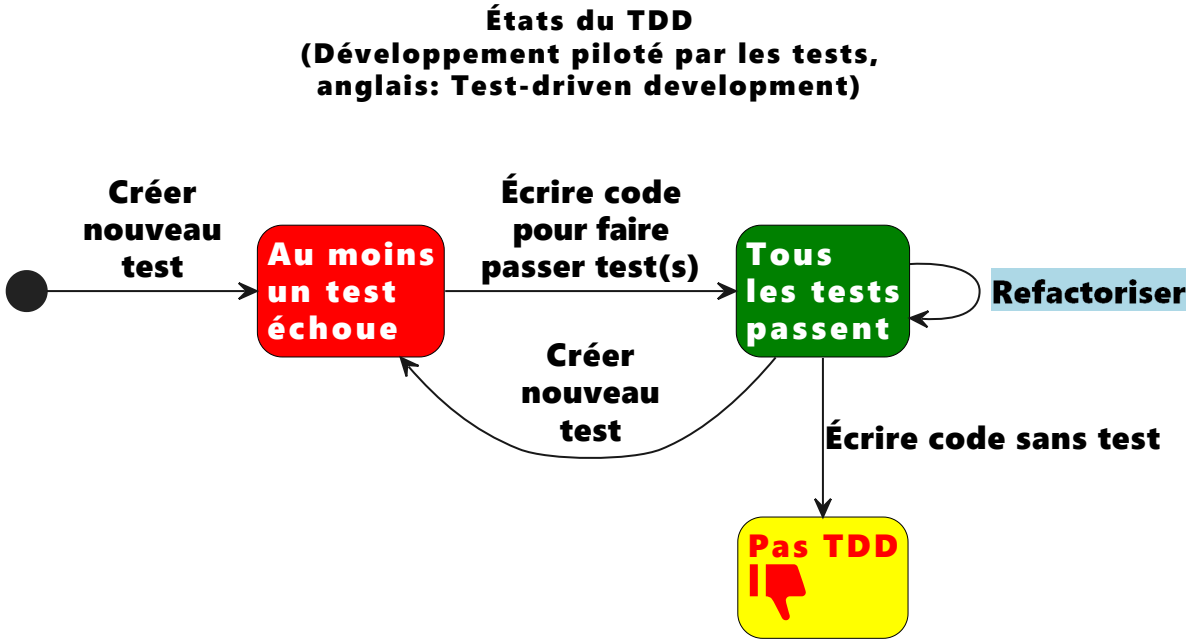
   ▶ Regarder exemple de la fonctionnalité



7. Lancer les tests (pas besoin de lancer le serveur d'abord) - `npm test`

Développement piloté par les tests (TDD)

▶ Plus de détails



Le développement piloté par les tests (Test-Driven Development, TDD) est une façon de développer des logiciels en commençant par les tests. Il y a plusieurs avantages de cette façon de faire et ce squelette supporte la méthodologie.

Le TDD suit un cycle particulier, comme vous pouvez voir à l'image plus haut:

1. Écrire un nouveau test
2. Exécuter le test (qui échouera)
3. Écrire juste assez de code pour faire passer le test
4. Refactoriser le code (et les tests) au besoin, et recommencer

> Il y a des tests pour tous les appels de l'API du serveur web, mais on devrait également faire des tests pour les autres classes (par exemple, des tests unitaires des classes du domaine).

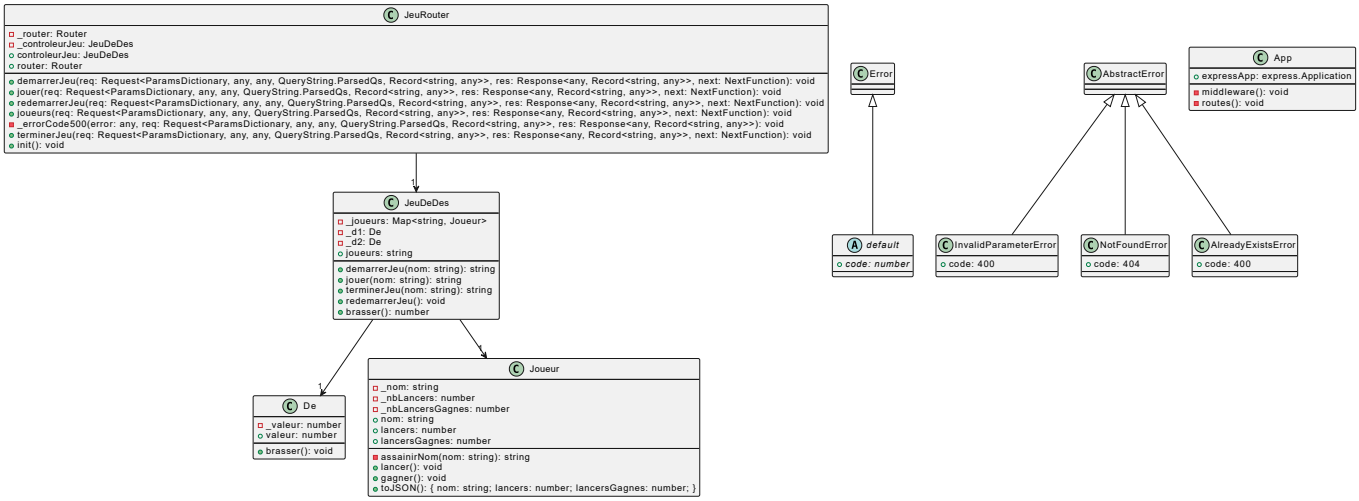
Support pour débogage

▶ Plus de détails

Ce squelette offre la possibilité de déboguer le code du serveur à l'aide de points d'arrêt placés à l'intérieur des fichiers TypeScript.

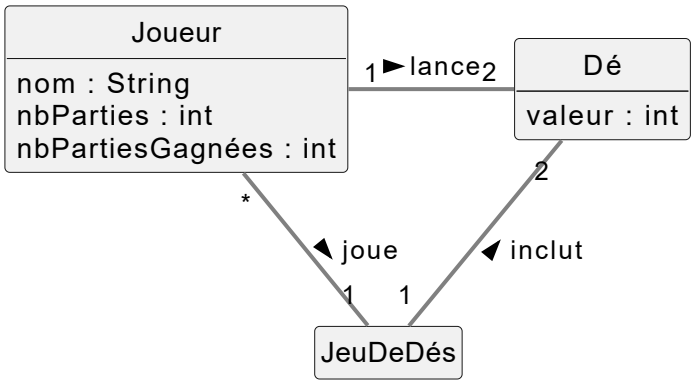
Voici comment il est possible de déboguer le projet à l'aide de différents environnements de développement.

Diagramme de classes logicielles



Modèle du domaine

Modèle du domaine (adapté du Jeu de dés du Ch. 1 de Larman)



DSS jouer (3 dés)

Welcome to PlantUML!

You can start with a simple UML Diagram like:

```
Bob->Alice: Hello
```

Or

```
class Example
```

You will find more information about PlantUML syntax on https://plantuml.com

(Details by typing `license` keyword)

RDCU jouer (3 dés)

Welcome to PlantUML!

You can start with a simple UML Diagram like:

```
Bob->Alice: Hello
```

Or

```
class Example
```

You will find more information about PlantUML syntax on https://plantuml.com

(Details by typing `license` keyword)

Débogage avec Visual Studio Code

VS Code offre la possibilité d'ajouter des configurations d'exécution à l'aide d'un fichier local. Ce fichier doit être nommé `launch.json` et être placé dans un dossier nommé `.vscode` à la racine du projet.

On peut utiliser ce fichier afin de créer des configurations d'exécution de débogage pour le projet. Un exemple de contenu pour ce fichier pourrait être :

```
{
    "version": "0.2.0",
    "configurations": [
        {
            "command": "npm start",
            "name": "Debug",
            "request": "launch",
            "type": "node-terminal"
        },
        {
            "command": "npm run start:watch",
            "name": "Debug:Watch",
            "request": "launch",
            "type": "node-terminal"
        }
    ]
}
```