

Manuel T00

Sommaire :

1. **Instructions :**

2. **Organisation, présentation du Code,des classes :**

3. **Technologies Utilisées**

4. **Commentaires**

5. **Documentation**

1) Instructions :

Ajout de fichier DMN :

Étape n°1 :

- Sélectionner le fichier DMN et le mettre dans le dossier backend/src/main/resources/dmn

Étape n°2 :

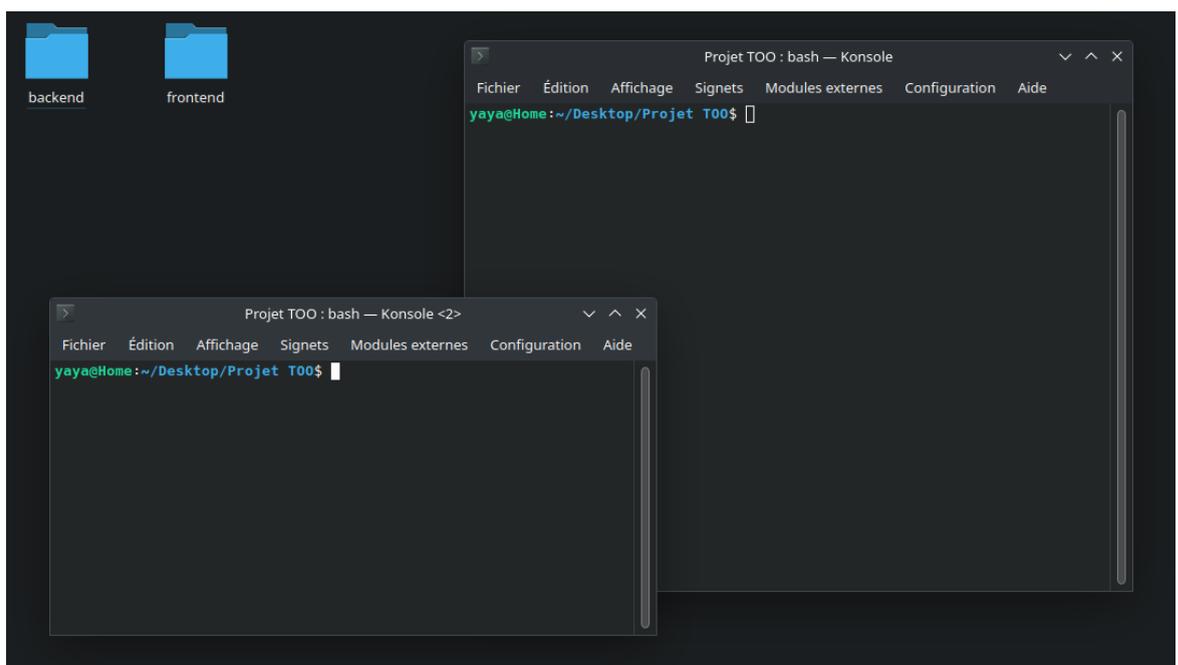
- Renommer le fichier de sorte que le début de chaque mots soit en majuscule. Exemple :

testfichierpremier.dmn -> TestFichierPremier.dmn

Initialization :

Étape n° 1 :

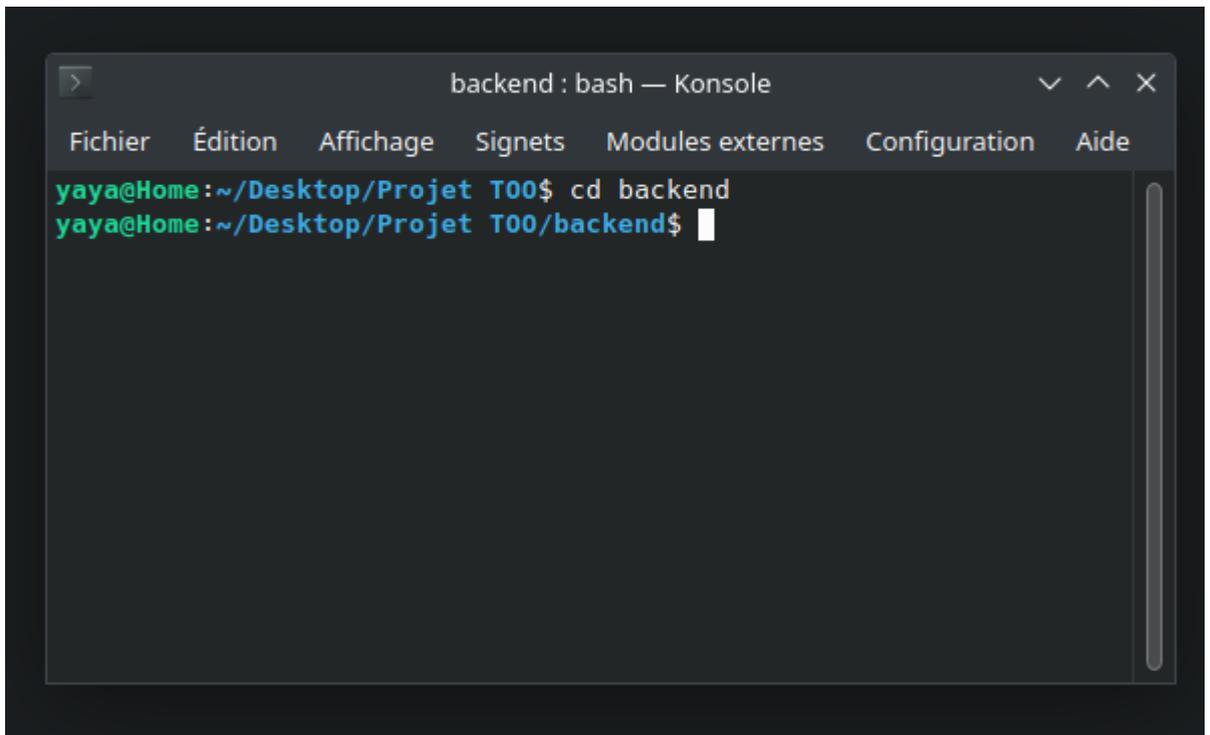
- Se positionner dans le fichier du projet.
- Ouvrir deux terminals



Étape n°2 :

- Dans le premier, se rendre dans le dossier back-end :

`cd back-end`



The image shows a terminal window titled "backend : bash — Konsole". The window has a menu bar with options: "Fichier", "Édition", "Affichage", "Signets", "Modules externes", "Configuration", and "Aide". The terminal content shows the following commands and output:

```
yaya@Home:~/Desktop/Projet T00$ cd back-end
yaya@Home:~/Desktop/Projet T00/backend$
```

- Puis, rentrer la commande suivante :

`mvn spring-boot:run`

Étape n°4 :

- Puis, entrer la commande suivante :

npm install

```

frontend : bash — Konsole
Fichier  Édition  Affichage  Signets  Modules externes  Configuration  Aide
yaya@Home:~/Desktop/Projet T00$ cd frontend
yaya@Home:~/Desktop/Projet T00/frontend$ npm install

added 619 packages, and audited 620 packages in 10s

196 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
yaya@Home:~/Desktop/Projet T00/frontend$

```

Étape n°5 :

- Puis écrivez la commande suivante :

npm run build

```

yaya@Home:~/Desktop/Projet T00/frontend$ npm run build
> frontend@0.0.1 build
> astro check && astro build

19:51:31 [check] Getting diagnostics for Astro files in /home/yaya/Desktop/Projet T00/frontend...
WARNING: Checking '.jsx' and '.tsx' files is temporarily disabled due to an issue in the Astro language server and TypeScript. See https://github.com/wwithastro/language-tools/issues/727 for more details. In the meantime, such files can be checked using `tsc --noEmit`.
Result (6 files):
- 0 errors
- 0 warnings
- 0 hints

19:51:38 [build] output: "static"
19:51:38 [build] directory: /home/yaya/Desktop/Projet T00/frontend/dist/
19:51:38 [build] collecting build info...
19:51:38 [build] ✓ completed in 18ms
19:51:38 [build] Building static entrypoints...
19:51:40 [build] ✓ completed in 2.18s

building client (vite)
19:51:44 [vite] ✓ 1470 modules transformed.
19:51:45 [vite] dist/_astro/client_KGhndtB.js 1.40 kB | gzip: 0.75 kB
19:51:45 [vite] dist/_astro/index_ofGy_RFE.js 99.43 kB | gzip: 33.82 kB
19:51:45 [vite] dist/_astro/index_go4T2lKB.js 141.31 kB | gzip: 45.47 kB
19:51:45 [vite] ✓ built in 4.77s

building static content
19:51:45 = src/pages/index.astro
19:51:45 | /index.html (+38ms)
19:51:45 ✓ completed in 269ms

19:51:45 [build] 1 page(s) built in 7.26s
19:51:45 [build] Complete!
yaya@Home:~/Desktop/Projet T00/frontend$

```

(Il se peut qu'il y est à refaire plusieurs fois cette commande...)

Étape n°6 :

- Enfin, écrivez la commande :

npm run preview

```
yaya@Home:~/Desktop/Projet T00/frontend$ npm run preview
> frontend@0.0.1 preview
> astro preview

astro v4.0.7 ready in 26 ms
| Local   http://localhost:4321/
| Network use --host to expose
```

- Ouvrir le lien obtenu. (Qui se trouve sous la forme de <http://localhost:4321/>).



Choisissez votre DMN à évaluer

DMN

Étape n°7 :

- Choisir le dmn à évaluer :

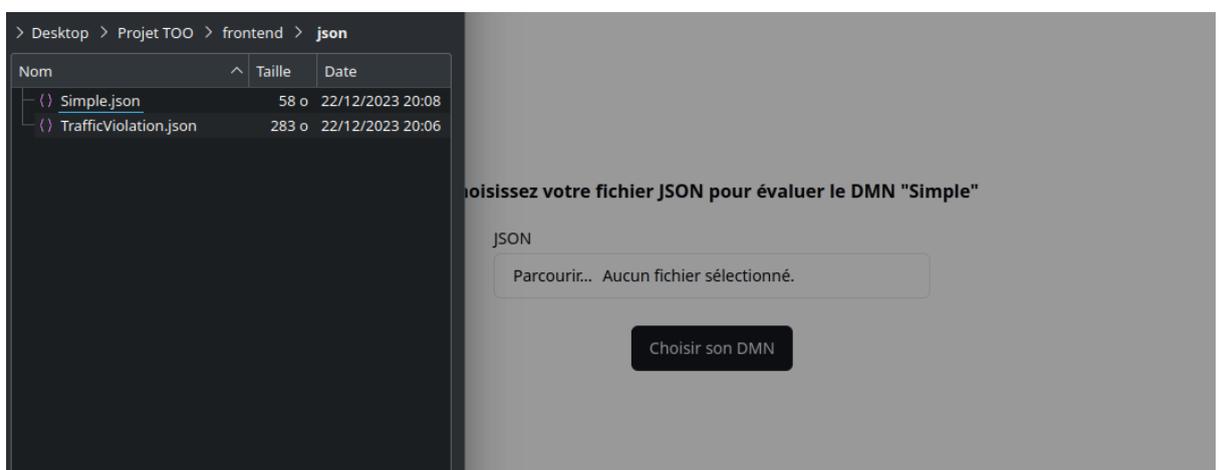
Choisissez votre fichier JSON pour évaluer le DMN "Simple"

JSON

Parcourir... Aucun fichier sélectionné.

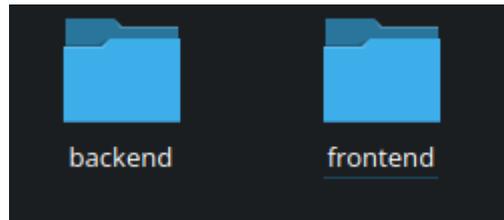
Choisir son DMN

- Sélectionner le fichier JSON correspondant qui se trouve dans le fichier frontend/json/



2) Organisation, présentation du code, des classes :

Initialement, nous sommes confrontés à la structure suivante :



Dans ce contexte, seul l'aspect back-end retiendra notre attention. En examinant de manière approfondie, voici la hiérarchie du back-end :

```
yaya@Home:~/Desktop/Projet T00/backend$ tree -d
├── src
│   ├── main
│   │   ├── java
│   │   │   ├── org
│   │   │   │   └── too
│   │   ├── resources
│   │   │   ├── dmn
│   │   │   └── META-INF
│   └── test
│       ├── java
│       │   ├── org
│       │   └── too
│       └── resources
├── target
│   ├── classes
│   │   ├── dmn
│   │   ├── http_58_47_47www_46signavio_46com_47dmn_471_461_47diagram_47cb7e33e39ee644da9a4bb48b1cc74e65_46xml
│   │   ├── https_58_47_47github_46com_47kiegroup_47drools_47kie_45dmn_47__A4BCA8B8_45CF08_45433F_4593B2_45A2598F19ECFF
│   │   ├── META-INF
│   │   └── resources
│   ├── org
│   │   ├── drools
│   │   │   └── project
│   │   │       └── model
│   │   ├── kie
│   │   │   ├── kogito
│   │   │   └── app
│   │   └── too
│   ├── generated-sources
│   │   ├── annotations
│   │   ├── kogito
│   │   │   ├── http_58_47_47www_46signavio_46com_47dmn_471_461_47diagram_47cb7e33e39ee644da9a4bb48b1cc74e65_46xml
│   │   │   ├── https_58_47_47github_46com_47kiegroup_47drools_47kie_45dmn_47__A4BCA8B8_45CF08_45433F_4593B2_45A2598F19ECFF
│   │   │   └── org
│   │   │       ├── drools
│   │   │       │   └── project
│   │   │       │       └── model
│   │   │       ├── kie
│   │   │       │   ├── kogito
│   │   │       │   └── app
│   └── generated-test-sources
│       └── test-annotations
```

```

target
├── resources
├── classes
│   ├── dmn
│   ├── http_58_47_47www_46signavio_46com_47dmn_471_461_47diagram_47cb7e33e39ee644da9a4bb48b1cc74e65_46xml
│   ├── https_58_47_47github_46com_47kiegroup_47drools_47kie_45dmn_47__A4BCA8B8_45CF08_45433F_4593B2_45A2598F19ECFF
│   ├── META-INF
│   └── resources
├── org
│   ├── drools
│   │   └── project
│   │       └── model
│   ├── kie
│   │   └── kogito
│   │       └── app
│   └── too
├── generated-sources
│   ├── annotations
│   ├── kogito
│   │   ├── http_58_47_47www_46signavio_46com_47dmn_471_461_47diagram_47cb7e33e39ee644da9a4bb48b1cc74e65_46xml
│   │   └── https_58_47_47github_46com_47kiegroup_47drools_47kie_45dmn_47__A4BCA8B8_45CF08_45433F_4593B2_45A2598F19ECFF
│   └── org
│       ├── drools
│       │   └── project
│       │       └── model
│       ├── kie
│       │   └── kogito
│       │       └── app
├── generated-test-sources
│   └── test-annotations
├── maven-status
│   └── maven-compiler-plugin
│       ├── compile
│       │   ├── default-compile
│       │   └── default-compile-1
│       ├── testCompile
│       └── default-testCompile
├── test-classes
│   └── org
│       └── too

```

53 directories
yaya@Home:~/Desktop/Projet T00/backend\$

Nous allons à présent examiner en détail la structure hiérarchique du projet en montrant brièvement les divers composants.

```

yaya@Home:~/Desktop/Projet T00/backend$ tree
.
├── pom.xml
├── README.md
├── src
│   ├── main
│   │   ├── java
│   │   │   ├── org
│   │   │   │   └── too
│   │   │   │       ├── CorsConfig.java
│   │   │   │       ├── GetAllDMNNames.java
│   │   │   │       └── KogitoApplication.java
│   │   └── resources
│   │       ├── application.properties
│   │       ├── dmn
│   │       │   ├── Simple.dmn
│   │       │   └── TrafficViolation.dmn
│   │       └── META-INF
│   └── test
│       ├── java
│       │   ├── org
│       │   │   └── too
│       │   │       ├── GreetingsTest.java
│       │   │       └── TrafficViolationTest.java
│       └── resources
├── target
│   ├── classes
│   │   ├── application.properties
│   │   ├── dmn
│   │   │   ├── Simple.dmn
│   │   │   ├── TrafficViolation.dmn
│   │   ├── http_58_47_47www_46signavio_46com_47dmn_471_461_47diagram_47cb7e33e39ee644da9a4bb48b1cc74e65_46xml
│   │   ├── Simple.dmn_nologic
│   │   ├── SimpleResource.class
│   │   ├── https_58_47_47github_46com_47kiegroup_47drools_47kie_45dmn_47__A4BCA8B8_45CF08_45433F_4593B2_45A2598F19ECFF
│   │   ├── TrafficViolation.dmn_nologic
│   │   └── TrafficViolationResource.class
│   ├── META-INF
│   │   └── resources
│   │       └── dmnDefinitions.json
│   └── org
│       └── drools

```

```

org
├── drools
│   ├── project
│   │   └── model
│   │       ├── ProjectModel.class
│   │       └── ProjectRuntime.class
│   └── kie
│       ├── kogito
│       │   └── app
│       │       ├── Application.class
│       │       ├── ApplicationConfig.class
│       │       ├── ConfigBean.class
│       │       ├── DecisionConfig.class
│       │       ├── DecisionModelResourcesProvider.class
│       │       ├── DecisionModels.class
│       │       ├── GlobalObjectMapper$1.class
│       │       ├── GlobalObjectMapper.class
│       │       ├── RuleConfig.class
│       │       └── RuleUnits.class
│       └── too
│           ├── CorsConfig.class
│           ├── GetAllDMNNames.class
│           └── KogitoApplication.class
└── generated-sources
    ├── annotations
    │   └── kogito
    │       ├── http_58_47_47www_46signavio_46com_47dmn_471_461_47diagram_47cb7e33e39ee644da9a4bb48b1cc74e65_46xml
    │       │   └── SimpleResource.java
    │       ├── https_58_47_47github_46com_47kiegroup_47drools_47kie_45dmn_47__A4BCA8B8_45CF08_45433F_4593B2_45A2598F19ECFF
    │       │   └── TrafficViolationResource.java
    │       └── org
    │           ├── drools
    │           │   ├── project
    │           │   │   └── model
    │           │   │       ├── ProjectModel.java
    │           │   │       └── ProjectRuntime.java
    │           └── kie
    │               ├── kogito
    │               │   └── app
    │               │       ├── ApplicationConfig.java
    │               │       ├── Application.java
    │               │       └── ConfigBean.java
    └── kogito
        ├── http_58_47_47www_46signavio_46com_47dmn_471_461_47diagram_47cb7e33e39ee644da9a4bb48b1cc74e65_46xml
        │   └── SimpleResource.java
        ├── https_58_47_47github_46com_47kiegroup_47drools_47kie_45dmn_47__A4BCA8B8_45CF08_45433F_4593B2_45A2598F19ECFF
        │   └── TrafficViolationResource.java
        └── org
            ├── drools
            │   ├── project
            │   │   └── model
            │   │       ├── ProjectModel.java
            │   │       └── ProjectRuntime.java
            └── kie
                ├── kogito
                │   └── app
                │       ├── ApplicationConfig.java
                │       ├── Application.java
                │       ├── ConfigBean.java
                │       ├── DecisionConfig.java
                │       ├── DecisionModelResourcesProvider.java
                │       ├── DecisionModels.java
                │       ├── GlobalObjectMapper.java
                │       ├── RuleConfig.java
                │       └── RuleUnits.java

```

```

kogito
├── http_58_47_47www_46signavio_46com_47dmn_471_461_47diagram_47cb7e33e39ee644da9a4bb48b1cc74e65_46xml
│   └── SimpleResource.java
├── https_58_47_47github_46com_47kiegroup_47drools_47kie_45dmn_47__A4BCA8B8_45CF08_45433F_4593B2_45A2598F19ECFF
│   └── TrafficViolationResource.java
└── org
    ├── drools
    │   ├── project
    │   │   └── model
    │   │       ├── ProjectModel.java
    │   │       └── ProjectRuntime.java
    └── kie
        ├── kogito
        │   └── app
        │       ├── ApplicationConfig.java
        │       ├── Application.java
        │       ├── ConfigBean.java
        │       ├── DecisionConfig.java
        │       ├── DecisionModelResourcesProvider.java
        │       ├── DecisionModels.java
        │       ├── GlobalObjectMapper.java
        │       ├── RuleConfig.java
        │       └── RuleUnits.java
└── generated-test-sources
    ├── test-annotations
    ├── maven-status
    │   └── maven-compiler-plugin
    │       ├── compile
    │       │   ├── default-compile
    │       │   │   ├── createdFiles.lst
    │       │   │   └── inputFiles.lst
    │       │   └── default-compile-1
    │       │       ├── createdFiles.lst
    │       │       └── inputFiles.lst
    │       └── testCompile
    │           └── default-testCompile
    │               ├── createdFiles.lst
    │               └── inputFiles.lst
    └── test-classes
        ├── org
        │   └── too
        │       ├── GreetingsTest.class
        │       └── TrafficViolationTest.class

```

53 directories, 54 files

3) Technologies utilisées :

Drools :

a) Présentation :

- ❖ Drools est un moteur de règles open source qui permet la gestion et l'exécution de règles métier complexes. Il est étroitement lié à la spécification DMN (Decision Model and Notation), offrant une prise en charge native de la modélisation, de l'évaluation, et de l'automatisation des décisions au sein de systèmes informatiques.

Java :

a) Présentation :

- ❖ Java, un langage de programmation polyvalent, joue un rôle clé dans l'intégration de DMN (Decision Model and Notation) pour la gestion efficace des règles métier dans le développement d'applications Java. En tant que composant essentiel, Java permet l'implémentation robuste des modèles de décision, renforçant la logique métier au cœur des applications Java.

Typescript :

a) Présentation :

- ❖ Typescript est un langage de programmation permettant de rendre plus facile et plus fiable l'écriture de code en JavaScript

pour des applications de grande ampleur. TypeScript apporte le typage statique au développement web, améliorant la robustesse des applications. Son utilisation avec DMN (Decision Model and Notation) offre une vérification statique des modèles de décision, renforçant la fiabilité et la maintenabilité du code métier.

Shadcn ui :

a) Présentation :

- ❖ Shadcn UI, conçu par Shadcn, se distingue en tant que collection de composants réutilisables, offrant une approche distincte des bibliothèques ou frameworks classiques. Il s'agit en effet d'une collection de composants réutilisables. Construit sur Tailwind CSS et Radix UI, il prend en charge divers frameworks.

b) Utilisation et explication :

- ❖ Nous avons décidé d'utiliser Shadcn pour les raisons suivantes :
 - Shadcn propose une plus grande polyvalence avec des exemples de codes efficaces.
 - Shadcn gère Tailwind.
 - Il permet de modifier le code, nous pouvons donc personnaliser ces bouts de code.

Nous avons donc dû cependant utiliser du typescript, tailwind, etc... afin de pouvoir profiter au maximum des avantages de Shadcn.

.

Astro :

a) Présentation :

- ❖ Astro est un framework web moderne axé sur les performances, offrant un rendu rapide côté serveur et côté client pour des applications web réactives et performantes. Il optimise la charge initiale des pages tout en permettant le développement avec les technologies préférées des développeurs.

b) Utilisation et explication :

- ❖ Nous avons choisi d'utiliser Astro puisque nous avons déjà quelques notions de bases dessus du à notre participation à la nuit de l'informatique. De plus, il est rapide, modulable et simple d'utilisation.

Spring Boot :

a) Présentation :

- ❖ Spring Boot, un framework Java basé sur Spring, simplifie le développement d'applications en offrant une configuration par défaut et une gestion facile des dépendances. Intégré à Drools, qui prend en charge DMN (Decision Model and Notation), Spring Boot facilite l'implémentation de règles métier dynamiques et leur intégration transparente dans les applications.

b) Utilisation et explication :

- ❖ Nous avons choisi Spring Boot pour gérer les fichiers DMN en raison de son intégration fluide avec Java, de sa facilité de configuration et de sa gestion des dépendances. De plus, sa modularité facilite le développement d'applications basées sur la modélisation de décisions.

Tailwind :

a) Présentation :

- ❖ Tailwind CSS est un outil de création de styles CSS utilisé par Shadcn UI. Plutôt qu'une bibliothèque de composants, Shadcn UI exploite Tailwind CSS pour créer une collection de composants réutilisables. Basé sur une approche utility-first, Tailwind offre une flexibilité considérable dans la conception de l'interface utilisateur.

Kogito :

a) Présentation :

- ❖ Kogito est une initiative de développement de la communauté Red Hat qui propose un écosystème complet pour simplifier la modélisation, l'exécution, et la gestion de services métier et de décision dans des environnements cloud-native. Intimement lié à la librairie Drools, Kogito utilise les capacités de Drools pour la gestion de règles métier et

d'automatisation de décisions, offrant ainsi une solution complète qui intègre la modélisation de processus métier (BPMN) et de décisions (DMN) avec des fonctionnalités cloud-native, le tout dans un ensemble cohérent et extensible.

b) Utilisation et explication :

- ❖ Le choix de Kogito s'est fait en raison de sa modélisation visuelle simplifiée avec le standard DMN, favorisant la collaboration avec les parties prenantes métier. La flexibilité offerte avec des frameworks Java tel que Spring Boot a été décisive pour traiter le projet. Les outils de modélisation DMN intégrés offrent une expérience visuelle, améliorant l'efficacité dans la création et la maintenance des modèles de décision. C'est donc pour son approche et sa gestion des fichiers DMN que nous avons choisi cet écosystème.
- ❖ Dans notre projet, nous utilisons Kogito pour évaluer et importer les fichiers DMN. En somme, pour gérer les fichiers DMN.

Maven

a) Présentation :

- ❖ Apache Maven est un outil de gestion de projet de développement logiciel. Il simplifie la construction, le reporting et la gestion des dépendances dans les projets Java. Maven utilise un modèle de projet déclaratif et fournit des cycles de vie prédéfinis pour les phases de build.

4) Commentaires :

- Nous avons délibérément opté pour le développement d'une interface graphique web dans le but d'améliorer la lisibilité des résultats générés par l'application Java. Grâce à l'aide de certains étudiants en informatique, nous avons pu élaborer une interface utilisateur primaire, conçue pour offrir une visibilité accrue et une appréciation globale plus poussée. Cette approche, que nous avons choisie, vise à rendre l'expérience utilisateur plus intuitive et à optimiser la compréhension des données fournies par l'application.

5) Documentation :

Astro :

- <https://github.com/withastro>

Kogito :

- <https://github.com/apache/incubator-kie-kogito-runtimes>
- <https://kogito.kie.org/get-started/>

Drools:

- <https://github.com/kiegroup/drools>
- <https://www.drools.org/learn/documentation.html>

- <https://www.baeldung.com/drools>

Shadcn ui :

- <https://github.com/shadcn-ui/ui>

Java :

- <https://docs.oracle.com/en/java/>
- <https://www.jmdoudoux.fr/java/dej/chap-javadoc.htm>

Spring Boot :

- <https://github.com/spring-guides/gs-spring-boot>
- <https://medium.com/javarevisited/spring-boot-drools-rule-engine-example-965eea437ee9>