

# AcccelEngine

## - User Guide -

Niveau de confidentialité	Document Confidentiel
Dernière mise à jour	02/02/2021
Destinataires	Réservé à un usage strictement interne

### Rédaction et modifications

Version actuelle : 0.1

Version	Date	Rédacteur	Description
0.1	04/12/2020	Khaled HAMMROUNI Mohamed Edam MEFTEH Haithem Ben Said	

## Mentions légales

Étant donné le caractère confidentiel de tout ou partie de ce document et de façon à assurer la protection contre un emploi intempestif ou une divulgation non autorisée à des tiers :

- La permission est accordée en vertu du présent accord de télécharger les documents détenus par de la Société Discovery et d'utiliser les informations contenues dans ce document uniquement dans la réalisation des projets de la Société Discovery.
- Vous s'engage à garder strictement confidentiel et à ne pas divulguer ou communiquer à des tiers, par quelque moyen que ce soit, toutes les informations contenues dans ce document.
- Vous s'engage à ne faire aucune copie, partielle ou complète, de ce document, de ses annexes et de tous documents s'y rapportant ou des documents ultérieurs et à retourner tous les documents reçus sur simple demande de la part de la Société Discovery.

## Table des matières

<b>1</b>	<b>INTRODUCTION .....</b>	<b>5</b>
<b>2</b>	<b>ARCHITECTURE (CLEAN ARCHITECTURE) .....</b>	<b>5</b>
2.1	Backend .....	5
2.1.1	Domain-driven design.....	5
2.1.2	Hexagonal Architecture .....	6
2.2	Frontend.....	7
2.2.1	Accelengine modules .....	7
2.2.2	Lazy-loading .....	7
<b>3</b>	<b>DEVELOPPENT TOOLS.....</b>	<b>7</b>
3.1	Eclipse.....	7
3.2	Gradle .....	8
3.3	Spring Tools .....	9
3.4	Lombok .....	9
3.5	Visual Studio Code.....	9
3.6	DBeaver .....	9
3.7	Git.....	10
<b>4</b>	<b>INSTALLATION D'ENVIRONNEMENT .....</b>	<b>10</b>
4.1	Base de données .....	10
4.1.1	PostgreSQL .....	10
4.1.2	DBeaver .....	11
4.2	Récupération de code .....	12
4.3	Backend .....	12
4.4	Frontend.....	13
<b>5</b>	<b>EXECUTION DE L'APPLICATION .....</b>	<b>15</b>
5.1	Configuration de l'application .....	15
5.2	Exécution backend.....	15
5.3	Exécution frontend .....	16
<b>6</b>	<b>GUIDES DE DEVELOPPEMENT .....</b>	<b>16</b>
6.1	Backend .....	16
6.1.1	Entity.....	16
6.1.2	Exceptions .....	16
6.1.3	Authorization.....	17
6.1.4	Events.....	17
6.1.5	Batch.....	18
6.1.6	Job .....	18
6.1.7	Send Email .....	19
6.1.8	Webservices externes .....	19

6.1.9	LoggerUtil .....	20
6.1.10	SecurityUtil.....	20
6.1.11	FileUtil .....	20
6.1.12	DateTimeUtil .....	20
6.1.13	Module .....	20
6.1.14	CRUD .....	20
6.1.15	Criteria (rsq)l .....	21
6.1.16	Pageable.....	21
6.2	Frontend.....	22
6.2.1	Architecture application front.....	22
6.2.2	Entity .....	22
6.2.3	Component .....	22
6.2.4	Service .....	23



**Entités (Enterprise Business Rules)**

Les entités incluent des règles métier d'une entreprise. Ils représentent des entités qui sont à la base de sa zone d'opération. Ce sont les composants avec le plus haut niveau d'abstraction.

**Cas d'utilisation (Application Business Rules)**

Les cas d'utilisation sont indiqués en tant que règles métier d'application. Chaque élément de cette couche fournit une interface avec la couche externe et agit comme un hub qui communique avec d'autres parties du système. Ils sont responsables de l'exécution complète des cas d'utilisation.

**Adaptateurs (Interface adapters)**

Cette couche se compose de la frontière entre les règles métier d'application et les outils qui lui permettent d'interagir avec le monde extérieur, comme les bases de données et les interfaces graphiques. Les éléments de cette couche agissent comme des médiateurs, recevant des données d'une couche et les transmettant à l'autre, adaptant les données selon les besoins.

**Infrastructure (Frameworks and Drivers)**

Les outils que votre système utilise pour communiquer avec le monde extérieur composent la couche la plus externe. Nous n'écrivons généralement pas de code dans cette couche, qui inclut des bibliothèques (UI, database, frameworks, devices).

La couche infrastructure prend en charge les interactions entre les autres couches via le **framework Spring Boot**.

**2.1.2 Hexagonal Architecture**

L'objectif principal de L'architecture hexagonale (ou ports et adaptateurs) est d'éviter les pièges structurels connus dans la conception de logiciels. Telles que la pollution du code de l'interface utilisateur avec la logique métier ou des dépendances indésirables entre les couches.



Un port est un point d'entrée/sortie vers/depuis l'application. Un port est une interface.

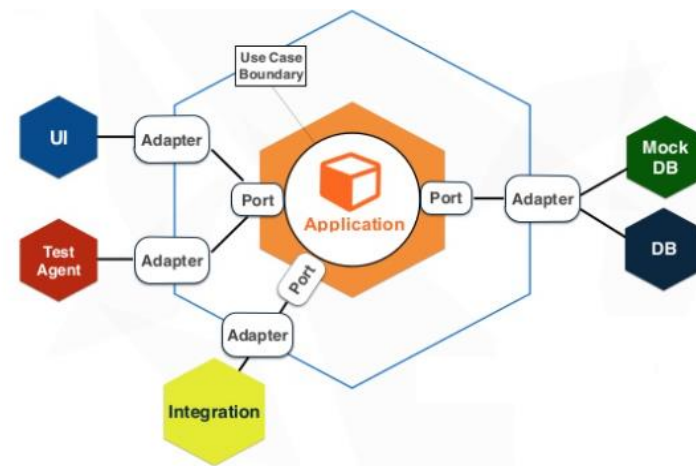
L'adapter forme la frontière entre la couche domaine et la couche Infrastructure, un adapter est une implémentation d'une interface port

**Exemple de port d'entrée :**

Sont des moyens d'interagir avec l'application, et impliquent généralement un mécanisme de livraison (par exemple, API REST, tâches planifiées, interface graphique, autres systèmes)

**Exemple de port de sortie :**

Récupérez et stockez des données depuis et vers un certain nombre de sources (base de données, périphériques réseau, système de fichiers, tiers, etc.)



## 2.2 Frontend

### 2.2.1 Accelengine modules

Un module est un mécanisme pour regrouper des composants, des directives et des services qui sont liés, de manière à pouvoir être combinés avec d'autres modules pour créer une application. Une application angular peut être considérée comme un puzzle où chaque pièce (ou chaque module) est nécessaire pour pouvoir voir l'image complète.

### 2.2.2 Lazy-loading

En configurant l'intégralité du Routing de l'application dans le module AppRoutingModuleModule, on serait amené à **importer tous les modules de l'application avant son démarrage**. A titre d'exemple, plus l'application sera riche, plus la page d'accueil sera lente à charger par effet de bord.

Pour éviter ces problèmes de "scalability", **Angular permet de charger les modules à la demande (Lazy Loading)** afin de ne pas gêner le chargement initial de l'application.

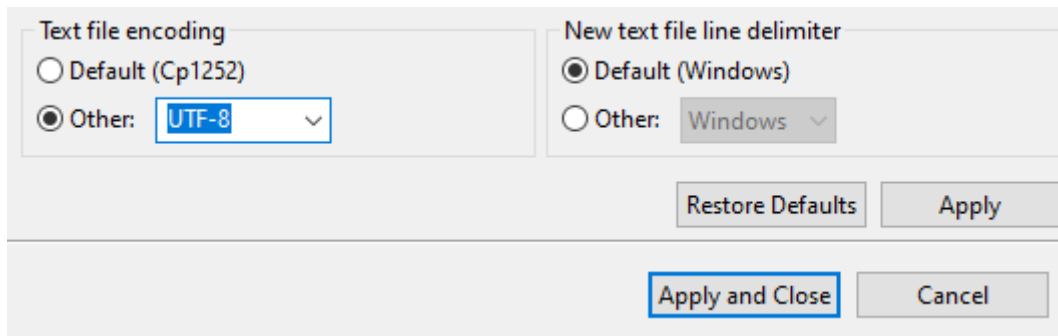
## 3 Développent Tools

### 3.1 Eclipse

Eclipse est l'IDE le plus populaire utilisé par les développeurs Java dans la programmation informatique. Il est utilisé pour développer des applications non seulement en Java mais également dans d'autres langages de programmation.

Définition du codage de texte par défaut pour Eclipse

Window -> Preferences -> General -> Workspace : Text file encoding



### Configuration des formateurs

Allez dans Eclipse -> Preferences -> Java -> Code Style -> Formatter

Cliquez sur Importer et choisissez le fichier formatter.xml dans le répertoire backend du projet

cliquez sur OK

### Enregistrer la configuration des actions

Allez dans Eclipse -> Preferences -> Java -> Editor -> Save Actions

Choisissez « Perform the selected actions on save »

Choisissez « Format source code »

Choisissez « Format all lines »

Choisissez « Organize imports »

Cliquez sur OK

### Workspace

Allez dans Eclipse -> Preferences -> General -> Workspace

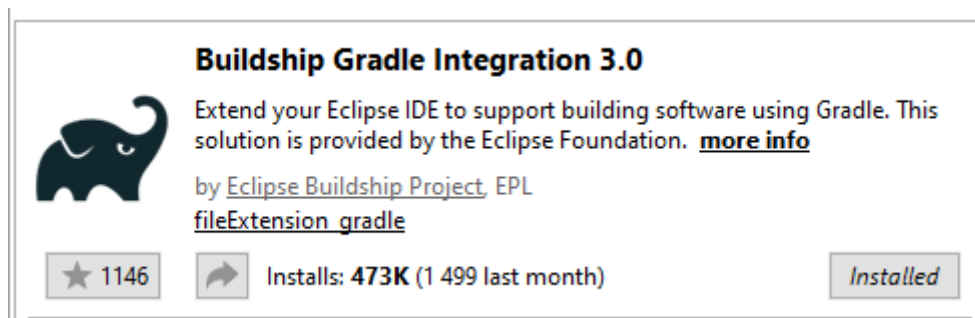
Définir le codage de fichier texte sur « Other: UTF-8 »

Définir New text file line delimiter to Unix

Cliquez sur OK

## 3.2 Gradle


Pour obtenir le support complet de Gradle dans eclipse, vous devez installer le plugin buildship.





### 3.3 Spring Tools

Pour obtenir une prise en charge complète de Spring dans eclipse, vous devez installer le plugin Spring Tools

**Spring Tools 3 (Standalone Edition) 3.9.13.RELEASE**

Spring Tools 3 (Standalone Edition) The Spring Tools 3 contain the previous generation Spring tooling for Eclipse, mostly focused on working with Spring apps... [more info](#)

by [VMware](#), EPL  
[J2EE spring](#) [Spring IDE](#) [Cloud jee](#)

★ 55

Installs: **120K** (9 055 last month)

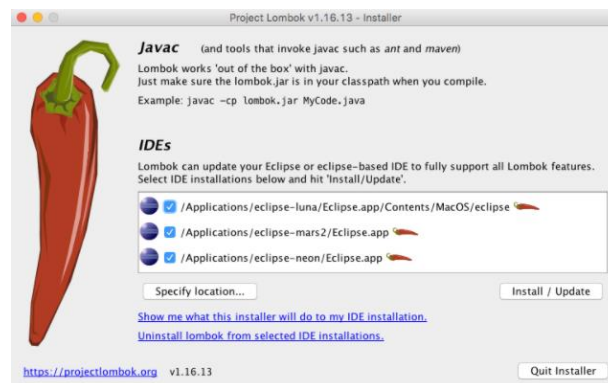
Installed

### 3.4 Lombok

Il s'agit d'une bibliothèque qui va générer pour vous, en respectant de nombreuses bonnes pratiques :

- constructeurs
- getters / setters
- equals / hashCode
- modificateurs d'accès (private, protected, etc.)

L'éditeur Eclipse est compatible avec lombok : double-cliquez sur lombok.jar. Cela lance le programme d'installation qui trouvera eclipse et propose d'installer lombok.



### 3.5 Visual Studio Code

L'éditeur Visual Studio Code prend en charge Angular, l'installer est disponible gratuitement pour Windows, OSX et les distributions Linux. Vous pourrez le télécharger à l'adresse suivante :

<https://code.visualstudio.com>

### 3.6 DBeaver

DBeaver est un outil de base de données universel gratuit et open source pour les développeurs et les administrateurs de bases de données. Vous pourrez le télécharger à l'adresse suivante :

<https://dbeaver.io/download/https://dbeaver.io/download/>

### 3.7 Git

Git est un système de contrôle de version distribué gratuit et open source conçue pour gérer tout, des petits projets aux très grands projets avec rapidité et efficacité.

- main : contient la version stable de l'application
- Develop : contient la version en cours de test
- feature : contient les fonctions en cours de développement

git clone	Clone le repo dans un nouveau dossier (workspace local)
git branch -a	La liste des branches (projets)
git status	Donner la liste des programmes modifiés
git add .	Ajouter la liste des programmes afin de les enregistrer dans la version courante
git commit -m "message de commit"	Enregistrer la liste des programmes modifiés dans le repo local en lui attribuant un commentaire
git pull	Télécharger et intégrer les modifications de la repo origine (dans le serveur)
git push	Charger le contenu du repo local vers le repo origine (dans le serveur)
git flow feature start NOM	Le workflow Gitflow définit un modèle de création de branche strict conçu autour de la livraison de projet

## 4 Installation d'environnement

### 4.1 Base de données

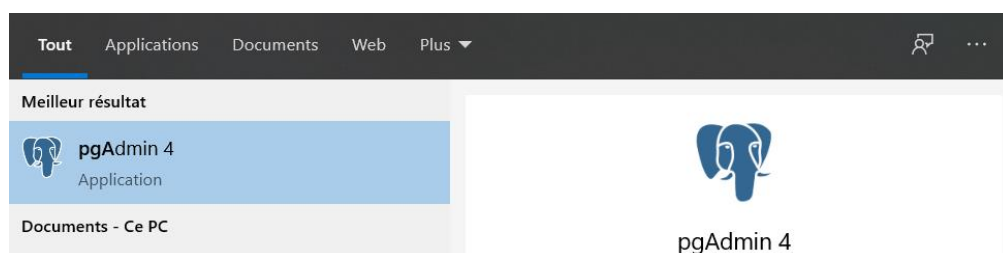
#### 4.1.1 PostgreSQL

Vous devrez installer et configurer la base de données PostgreSQL. Vous pourrez le télécharger à l'adresse suivante <https://www.postgresql.org/download/>

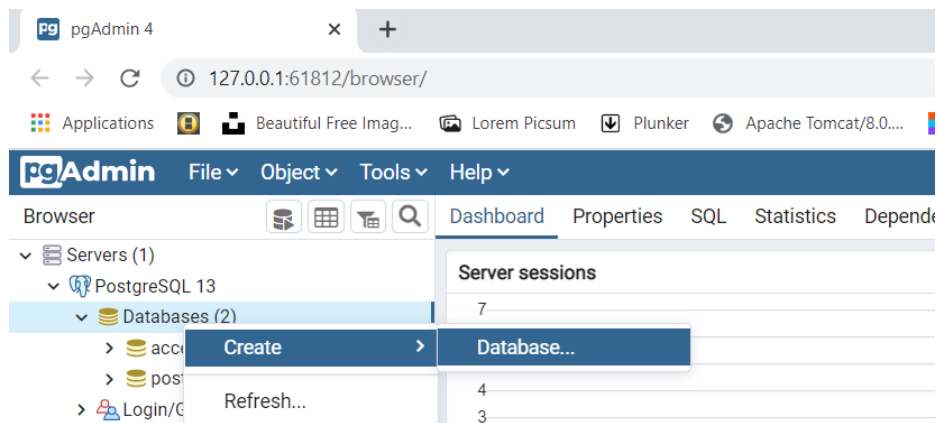
N'oubliez pas de configurer vos propriétés de connexion en conséquence dans les fichiers

/backend/src/main/resources/application - \*. Yml

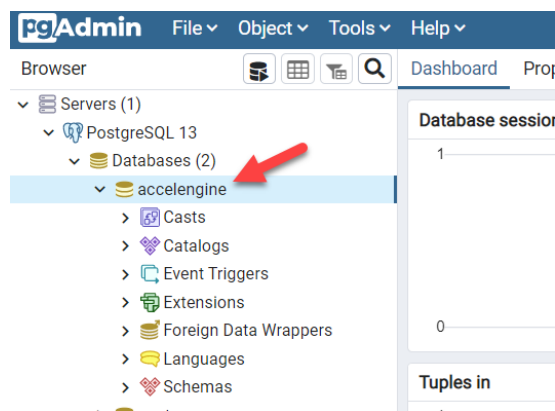
Après installation, lancez l'application "pgAdmin"



L'application s'ouvre dans le navigateur. Vous devez créer une nouvelle base qui s'appelle "accelengine"

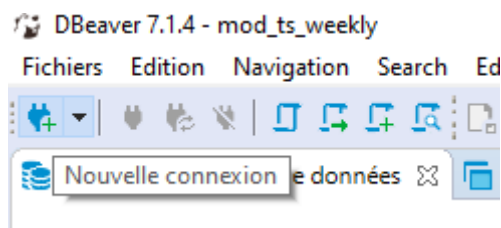


La base est créée :



## 4.1.2 DBeaver

Ajouter une nouvelle connexion



Sélectionner PostgreSQL standard driver



Ajouter la base de données déjà créée

Général
PostgreSQL
Propriétés du pilote
SSH
Proxy
SSL

Server
Host: localhost Port: 5432
Database: accelengine

Authentication
Authentication: Database Native Database native authentication
Nom d'utilisateur : postgres
Mot de passe :  ☒ Save password locally

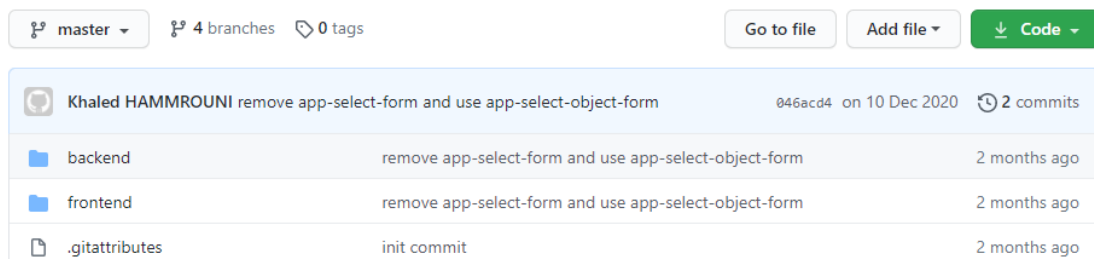
Advanced
Local Client: PostgreSQL 12

*?* You can use variables in connection parameters.
Connection details (name, type, ...)

Driver name: PostgreSQL
Edit Driver Settings

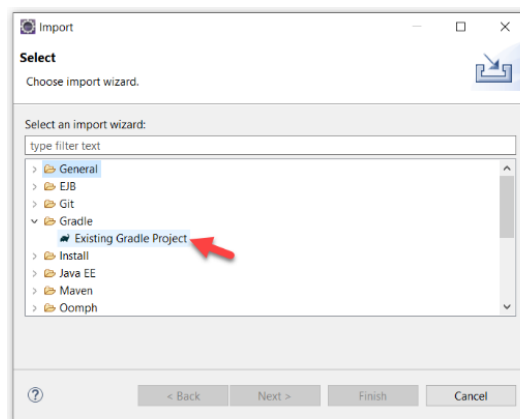
## 4.2 Récupération de code

Github C'est une plateforme de partage de code source des applications et de gestion des repositories Git. (Vous devez créer un compte)

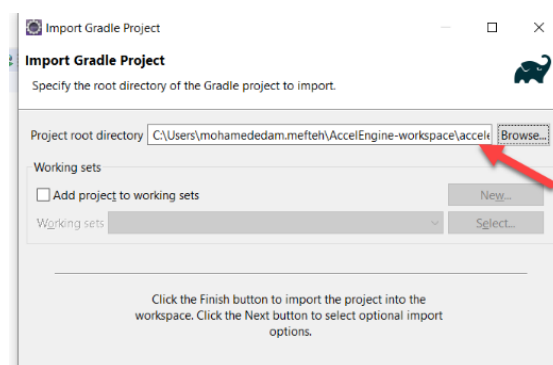


## 4.3 Backend

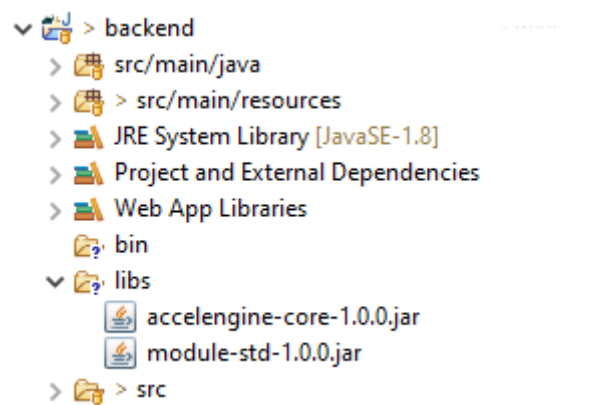
Après avoir récupérer le code il faut l'importer dans Eclipse : Importez votre projet en tant que projet Gradle : Sélectionnez Fichier -> Importer  
Choisissez le projet Gradle



Sélectionnez le répertoire racine de projet et Cliquez sur Suivant et terminez l'assistant



Téléchargez les bibliothèques standard AccelEngine (lien téléchargement jars a communiqué) et copiez-les dans :  
 \accelengine\backend\libs



Si la version des jars est modifiée, vous devez modifier l'importation dans la configuration gradle "build.gradle" du projet backend :

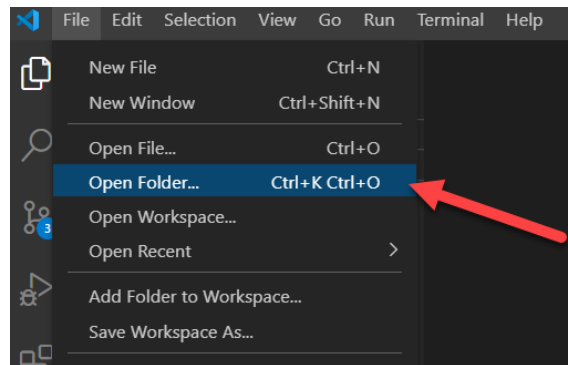
```
compile fileTree(dir: '../libs', include: ['accelengine-core-1.0.0.jar', 'module-std-1.0.0.jar'])
```

## 4.4 Frontend

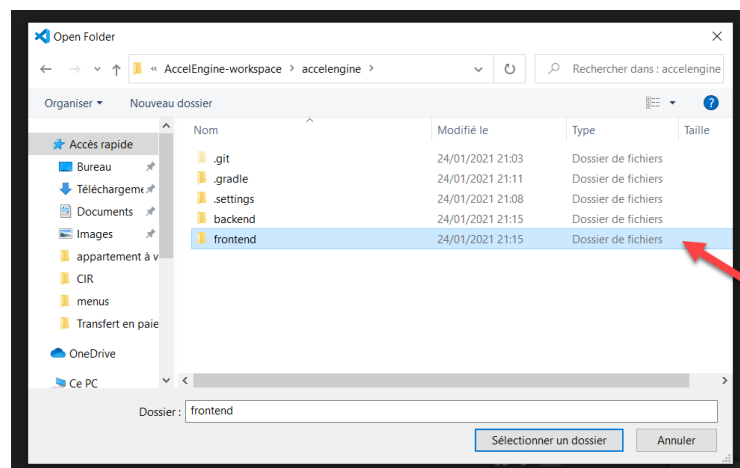
Installation NodeJS lien de téléchargement : <https://nodejs.org/en/> )

Install Angular CLI : lancer la commande "npm install -g @angular/cli" dans l'invite de commande CMD

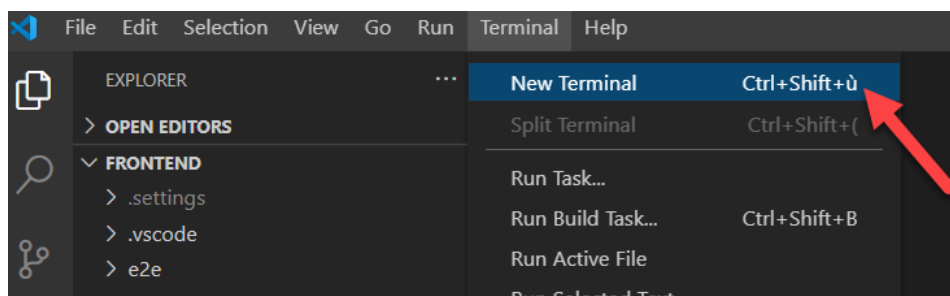
Ouvrir le Visual Studio Code et ouvrir le dossier racine du projet :



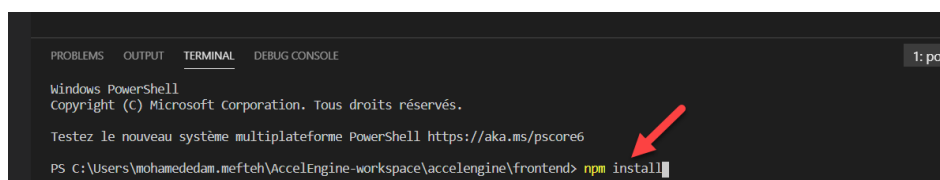
Sélectionner le dossier frontend du projet



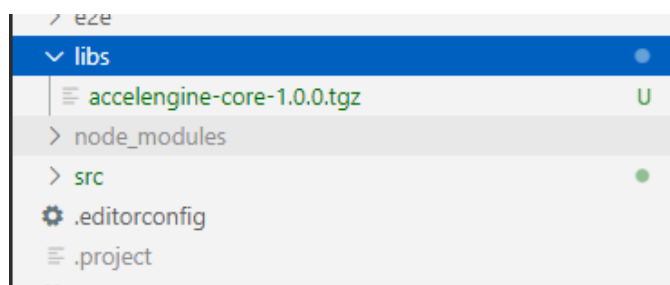
Ouvrir un terminal dans la VSCODE



Lancer la commande "npm install" pour installer les dépendances de projet



Téléchargez les bibliothèques standard AccelEngine (lien téléchargement tgz a communiqué) et copiez-les dans :  
\\accelengine\\frontend\\libs



Lancer la commande « `npm install libs\accelengine-core-1.0.0.tgz` » pour installer les dépendances standard AccelEngine, vérifier dans package.json :

```
"@types/sockjs-client": "1.5.0",
"accelengine-core": "file:C:/libs/accelengine-core-1.0.0.tgz",
"bootstrap": "4.6.0",
```

## 5 Exécution de l'application

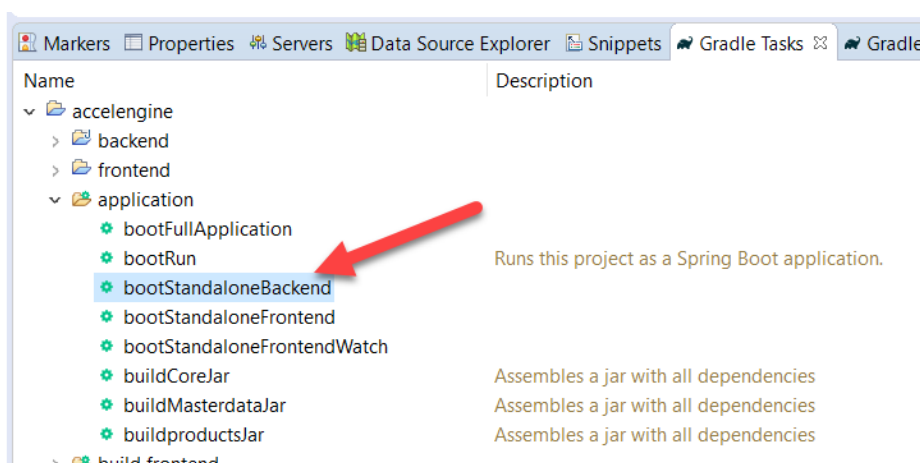
### 5.1 Configuration de l'application

Par défaut, Accelengine utilise le profil « dev ».

Propriétés de l'application standard Spring Boot	/backend/src/main/resources/application.yml
Configuration spécifique à AccelEngine	/backend/src/main/resources/accelengine.config
Configuration par profil	/backend/src/main/resources/application-dev.yml /backend/src/main/resources/application-prod.yml
Utilisateurs créés lors du démarrage de l'application	/backend/src/main/resources/accounts.csv
Menus créés lors du démarrage de l'application	/backend/src/main/resources/menus.csv

### 5.2 Exécution backend

Dans Eclipse, lancer la tâche gradle `bootStandaloneBackend`

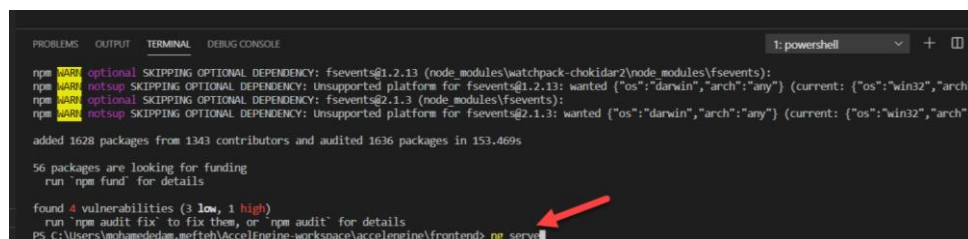


## Taches Gradle

bootRun	Démarrer l'application Spring boot
bootStandaloneBackend	Démarrez uniquement la partie Backend
buildFrontendToSpring	Compilez et intégrez le Frontend dans l'application Spring Boot
bootStandaloneFrontend	Démarrez uniquement la partie Frontend

## 5.3 Exécution frontend

Dans le terminal VSCODE, lancer la commande "ng serve" ou "ng s"



```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE 1: powershell
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.13 (node_modules\watchpack-chokidar2\node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.13: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@2.1.3 (node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@2.1.3: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":
added 1628 packages from 1343 contributors and audited 1636 packages in 153.469s
56 packages are looking for funding
  run `npm fund` for details
found 4 vulnerabilities (3 low, 1 high)
  run `npm audit fix` to fix them, or `npm audit` for details
PS C:\Users\Mohamedadaw.mefteh\AccelEngine-workspace\accelengine\frontend> ng serve
  
```

## 6 Guides de développement

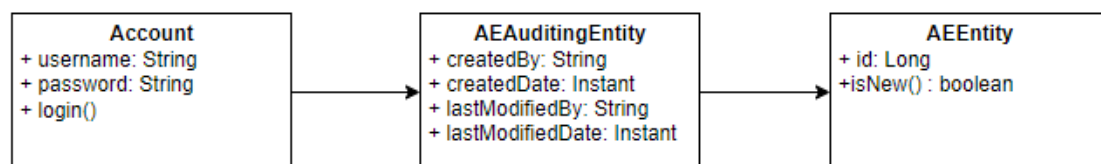
### 6.1 Backend

#### 6.1.1 Entity

AEEntity : classe générique inclut les propriétés utilisées par tous les objets

AEAuditingEntity : classe générique inclut les propriétés surveillance et traçabilité

⇒ Qui a créé et quand, qui a modifié et quand



AEList **TODO**

#### 6.1.2 Exceptions

Toutes les exceptions de l'application doivent être étendues à partir de l'exception standard **AEEException**, exemple :

AEBusinessException : exception dans une règle de gestion

AETechnicalException : exception technique



### 6.1.3 Authorization

Ce module et développer pour augmente le modèle de sécurité de la sécurité basée sur le code à la sécurité basée sur l'utilisateur

Chaque Entités (Enterprise Business Rules) représenter comme ressources, les autorisations sont au cœur de l'autorisation ; ils contrôlent l'accès (les actions) aux ressources.

#### Action standard

- Lire
- Créer
- Mettre à jour
- Supprimer
- Toutes les actions

Nous pouvons accorder un ou plusieurs droits (Permissions) de contrôle d'accès pour chaque utilisateur.

#### Cette configuration se fait dans le Module Installer

Création d'actions personnalisées :

```
Action revisit = actionInput.createNewDataAndGet(new Action("REVISIT", "Revisiter"));
```

Création de Permission et affectation des Actions :

```
manger_permissions.add(new Permission(Document, validation));
```

Création de rôle et affectation des Permissions :

```
roleInput.createNewDataAndGet(new Role("Manger", manger_permissions));
```

#### Ce contrôle se fait dans les UseCase

Contrôler si l'utilisateur actuel a le droit d'effectuer une action sur un document :

```
authorization.assertCan(Weekly.class, "REVISIT");
```

### 6.1.4 Events

Un événement est la manière dont un composant permet à un auditeur de savoir que quelque chose s'est passé.

Nous devons fournir un mécanisme pour enregistrer et désenregistrer les écouteurs d'événements :

```
@AEEvent
@Slf4j
public class AccountLoginEventListener {
    @PostConstruct
    public void init() {
        AEEventRunner.getInstance().register(this);
    }
}
```

Le message contient toutes les informations nécessaires à un auditeur pour comprendre ce qui s'est passé :

```
public class LoginMessage {
    @Getter
    public static class AccountLogin implements AEEventMessage {
        private String account;

        public AccountLogin(String account) {
            this.account = account;
        }
    }
}
```

Pour écouter un événement, un écouteur doit utiliser l'annotation `@AEListener` et lorsqu'un événement se produit, le standard appelle la méthode appropriée :

```
@AEListener
public void handle(LoginMessage.AccountLogin data) {
    log.debug("AccountLogin ok ");
}
```

### 6.1.5 Batch

Le traitement en batch consiste à exécuter des jobs répétitifs contenant des volumes importants de données. La méthode par lot (batch) permet aux utilisateurs de traiter des données avec peu ou pas d'intervention de leur part.

Initialisation de batch et déclaration des paramètres batch :

```
BatchParameter batchParameter = new BatchParameter();
batchParameter.getParameters().put("key1", "value1");
batchParameter.getParameters().put("key2", "value2");
```

Lancement de batch :

```
this.batchInput.startBatch(TestBatch.BATCH_NAME, batchParameter);
```

- beforeBatch : récupération des paramètres et collecte de données
- execute : le traitement principal du batch
- stopBatch : le traitement à effectuer pour arrêter le traitement en cours
- failedBatch : le traitement à effectuer si le batch se termine par une erreur
- afterBatch : le traitement à effectuer si le batch se termine correctement

### 6.1.6 Job

Un nouveau Job est créé en implémentant l'interface `AEJobDetail`.

La méthode `init()` inclut la création de la configuration de la job

```
@PostConstruct
public void init() {
    Job job = new Job();
    job.setName("Test");
    job.setGroupName("Group");
    job.setClassName(this.getClass().getName());
    job.setCronExpression("0 * * ? * *");
    job.setStatus(true);
    schedulerJobInput.scheduleNewJob(job);
}
```

La méthode `executeJob()` inclut la logique métier qui sera exécutée.

```
protected void executeJob(JobExecutionContext context) throws JobExecutionException {  
    log.info("SampleCronJob Start.....");  
    log.info("SampleCronJob End.....");  
}
```

### 6.1.7 Send Email

Envoyer des mails à l'aide de SMTP, depuis de simples messages textes jusqu'à de complexes messages HTML avec pièces jointes. Les configurations sont initialisées par deux méthodes :

#### Configurations local:

`AEEEmail(String username, String password, String host, String port)`

#### Configurations dans l'application :

Menu -> Administration -> Application et dans rubrique « Application »

Email	<input type="text"/>
Email - Nom d'utilisateur	<input type="text"/>
Email - Mot de passe	<input type="password"/>
Email - Hôte	<input type="text"/>
Email - Port	<input type="text"/>

Initialisation et connexion au serveur de messagerie :

```
AEEEmail mail = new AEEEmail();
```

Initialisation information et envoi d'email :

```
AEMessage msg = new AEMessage();
```

```
msg.setFrom("From");  
msg.setFromName("FromName");  
msg.setTo("email");  
msg.setSubject("Subject");  
msg.setBody("Body");  
  
msg.getAttachments().add("D:\\aaa.txt");  
  
mail.sendMail(msg);
```

### 6.1.8 Webservices externes

**TODO**

### 6.1.9 LoggerUtil

Initialisation de la fichier log :

```
LoggerUtil logger = new LoggerUtil(dirPath, domain, fileName);
```

Type des message log

- addTitleMessage
- addInfoMessage
- addErrorMessage
- addWarningMessage

Pour calculer le temps d'exécution d'un bloc de traitement :

timeEnd() => Renvoie le temps passé depuis l'exécution de timeStart()

### 6.1.10 SecurityUtil

getCurrentAccount : Obtenez l'utilisateur actuel

getCurrentAccountLogin : Obtenez le login de l'utilisateur actuel

passwordEncode Encoder le mot de passe

passwordMatches Vérifiez que le mot de passe codé correspond au mot de passe soumis

### 6.1.11 FileUtil

Contient des fonctions qui vous permettent de manipuler des dossiers et des fichiers.

### 6.1.12 DateTimeUtil

Contient des fonctions qui vous permettent de manipuler des dates.

### 6.1.13 Module

#### TODO

- Installer
- Setting
- Role/Action

### 6.1.14 CRUD

Plusieurs classes peuvent être étendues, dont les définitions des fonctions de base de CRUD

- **AECrudApi** : les fonctions CRUD de webservice REST
- **AECrudInputPort** : interface des fonctions entrent ver les usecases CRUD
- **AECrudOutputPort** : interface des fonctions sortent des usecases CRUD
- **AECrudUseCase** : les fonctions usecases CRUD

### 6.1.15 Criteria (rsql)

RSQL est un langage de requête pour le filtrage paramétré des entrées dans les API RESTful. Il est basé sur FIQL (Feed Item Query Language), une syntaxe conviviale pour les URI permettant d'exprimer des filtres sur les entrées d'un flux Atom. FIQL est idéal pour une utilisation dans URI; il n'y a pas de caractères dangereux, le codage d'URL n'est donc pas nécessaire. D'un autre côté, la syntaxe de FIQL n'est pas très intuitive et le codage d'URL n'est pas toujours très important, donc RSQL fournit également une syntaxe plus conviviale pour les opérateurs logiques et certains des opérateurs de comparaison.

Dans notre solution, les paramètres de filtrage sont passés dans un `@RequestBody` de type `AESearchCriteria`. Cet objet est constitué essentiellement d'une liste des `AECriteria` avec laquelle on construira la requête de filtrage.

Chaque `AECriteria` est construit :

- Un code qui correspond à un attribut de l'entité cible
- Un opérateur de comparaison
- Une valeur

Exemple d'utilisation :

```
public AEList<T> findByCriteria(AESearchCriteria searchRequest) {  
    Node rootNode = new RSQLParser().parse(searchRequest.searchRequestBuilder());  
    Specification<T> spec = rootNode.accept(new CustomRsqlVisitor<T>());  
    AEList<T> list = new AEList<>();  
    list.setDatas(jpaRepository.findAll(spec));  
    return list;  
}
```

### 6.1.16 Pageable

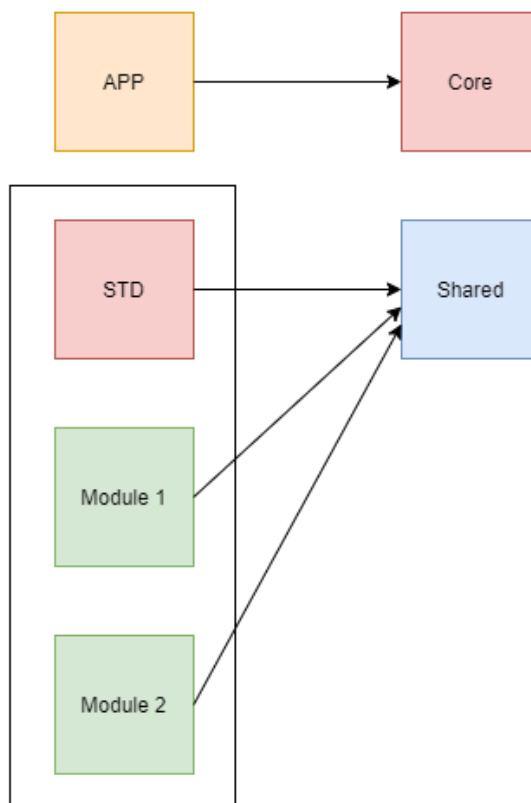
`AEPagable` est utilisée pour gérer la pagination coté serveur. Elle est constituée du numéro de la page et le nombre des données par page et qui sert à construire un objet de type `"org.springframework.data.domain.Pageable"` afin

Exemple d'utilisation :

```
public AEList<T> findAllPageable(AEPagable pageable) {  
    Page<T> datas = jpaRepository.findAll(pageable.getSpringPageable());  
    return new AEList<>(datas.getContent(), datas.getTotalPages(), datas.getTotalElements());  
}
```

## 6.2 Frontend

### 6.2.1 Architecture application front



### 6.2.2 Entity

AEEntity : classe générique inclut les propriétés utilisées par tous les objets

AEAuditingEntity : classe générique inclut les propriétés surveillance et traçabilité

⇒ Qui a créé et quand, qui a modifié et quand

AEList **TODO**

### 6.2.3 Component

Dans le standard, nous avons de nombreuses classes (Component) avec des fonctions prédéfinies pour faciliter le développement et éviter de réécrire des fonctions réutilisables.

**BaseComponent** : classe de base de tous les Components

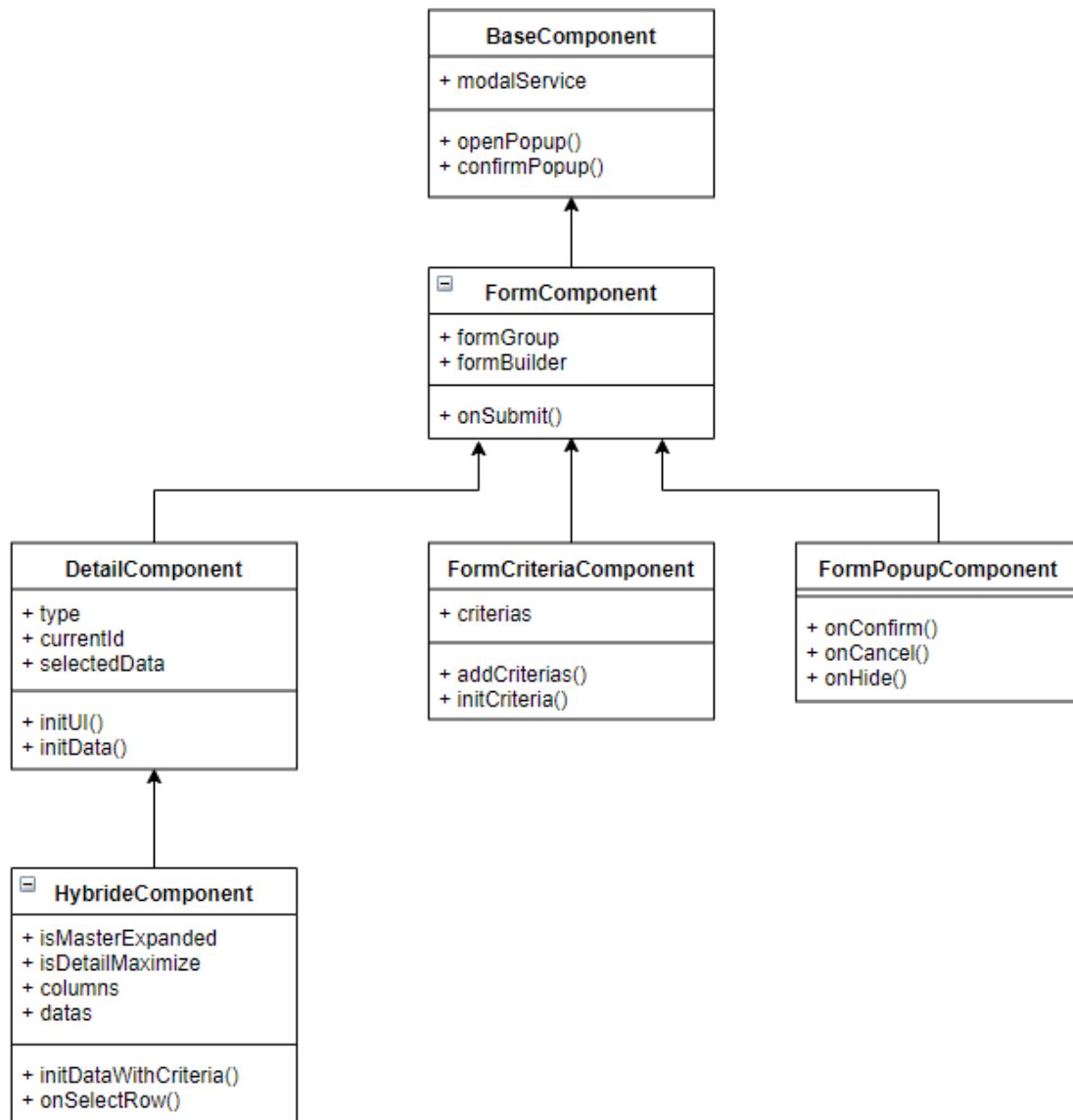
**FormComponent** : gérer les Form Angular (FormGroup, FormBuilder)

**FormCriteriaComponent** : utiliser pour ajouter une coche de filtre standard

**FormPopupComponent** : gérer les Form Modal Angular

**DetailComponent** : formulaire : affichage, mode (lecture ou mise à jour), validation, sauvegarde

**HybrideComponent** : gérer les tables des données



## 6.2.4 Service

**CrudAPIService** : on peut étendre cette classe incluant la définition des fonctions de base de CRUD, cette classe est complémentaire de l'interface **AECrudApi** de la partie back