DIGITALVIEW

Document technique de formation

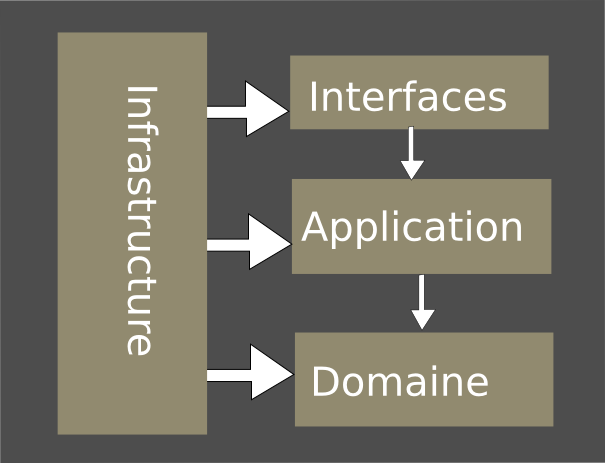
**V 1.0**

1. [ENVIRONNEMENT TECHNIQUE 3](#_bookmark0)
   1. [Architecture technique 3](#_bookmark1)
      1. [Interface 4](#_bookmark2)
      2. [Application 4](#_bookmark3)
      3. [Domaine 5](#_bookmark4)
      4. [Infrastructure 5](#_bookmark5)
   2. [Outils de développement 6](#_bookmark6)
      1. [Partie Front End 6](#_bookmark7)
      2. [Partie Back End 6](#_bookmark8)
      3. [Base de données : PostgreSQL 7](#_bookmark9)
      4. [Docker 7](#_bookmark9)
      5. [Gestionnaire de codes sources : GIT (Serveur) et GitLab (Client) 7](#_bookmark10)
   3. [Environnements de test 7](#_bookmark11)
   4. [Gestion des équipes projet agiles : JIRA 8](#_bookmark12)
   5. [Gestion des prises de note collaboratif pour les équipes agiles : Miro 8](#_bookmark12)
2. [PRESENTATION PROJET TYPE DIGITALVIEW 11](#_bookmark13)
   1. [Description arborescence projet de base 11](#_bookmark14)
   2. [Mécanisme de traduction des applications 12](#_bookmark16)
   3. [Gestionnaire de migration flyway 13](#_bookmark17)
   4. [Mapping avec Mapstruct 15](#_bookmark18)
3. [CAS PRATIQUES 16](#_bookmark19)
   1. [Niveau 0 : Installation des prérequis. 16](#_bookmark20)
   2. [Niveau 1 : Appréhender l’environnement de développement 16](#_bookmark21)
   3. [Niveau 2 : Réaliser une interface suivant l’arborescence DIGITALVIEW 17](#_bookmark22)
   4. [Niveau 3 : Gérer les critères de recherche 17](#_bookmark23)
   5. [Niveau 4 : Gérer les tests unitaires 17](#_bookmark24)
   6. [Niveau 5 : Prise en main de l’architecture 18](#_bookmark25)



# Architecture technique

L’approche choisie par DIGITALVIEW dans la réalisation de ces applications est l’approche orienté domaine, on peut parler de Domain Driven Design (DDD). Nous nous baserons sur différentes couches qui sont décrites ci-dessous.



La différence majeure par rapport à une architecture en couches « classique » sera la présence de code métier au sein des entités, le modèle de domaine n’est plus considéré comme un modèle anémique (*Un domaine anémique est un domaine dont les classes contiennent essentiellement des propriétés*). Ci-après, le détail de l’architecture applicative :

Une image contenant texte, capture d’écran, diagramme, Police

Description générée automatiquement

## Interface

La couche d’interface contient les IHM et les points d’entrée vers le métier. La couche

Interfaces portera les responsabilités suivantes :

* Exposition de point d’entrée vers le métier par le biais de façade
* Utilisation des points d’entrée métiers par les IHM ou les frontend exposés aux autres applicatifs.

## Application

La couche application contient les services du domaine de l’application qui ne sont pas de la responsabilité d’un objet du domaine. La couche application portera les responsabilités suivantes :

* Validation technique et fonctionnelle des objets du domaine
* Point d’entrée vers un processus métier complexe mettant en œuvre plusieurs objets

Du domaine.

* Orchestration des opérations du domaine

## Domaine

La couche domaine contient tous les objets du domaine. Ces objets seront les entités persistantes et contiendront des méthodes permettant de leur faire jouer leur rôle métier. La couche domaine portera les responsabilités suivantes :

* Opération relevant du domaine
* Définition des services d’accès aux données
* Gestion du cycle de vie des objets du domaine (création et persistance).

## Infrastructure

La couche d’infrastructure contient les objets traitant avec les couches basses de l’applicatif.

La couche infrastructure portera les responsabilités suivantes :

* Implémentations des services d’accès aux données
* Implémentations des services d’envoi de données vers l’extérieur
* Implémentations des services d’export de données (PDF, office, …).

# Outils de développement

Les outils de développement représentent un ensemble de technologies, logicielles pour la plupart, permettant à l’équipe de développement de transformer la modélisation en application concrète utilisable par le métier. Dans le cadre des applications DigitalView, ces outils peuvent être réparties en trois catégories :

## Partie Front End

Ces outils permettent la mise en place des applications « client » utilisées par les utilisateurs finaux.

* + - 1. **Node.js**

Node.js est une plateforme logicielle libre en JavaScript permettant de faire du développement web évènementiel. Créé le 27 mai 2009 par Ryan Lienhart Dahl, Node.js était à sa version 14.4.0 le 02 juin 2020. Node.js englobe plusieurs modules de façon native dont http permettant de se passer d’un autre serveur web lors du déploiement d’une application node.js. Toute la documentation de Node.js est disponible sur le site officiel <https://nodejs.org/>.

* + - 1. **Angular**

Angular est un Framework open source côté client permettant de dynamiser des applications web. Il s’agit d’une réécriture plus complète du Framework AngularJs. Il est développé Google et la communauté. Il était à sa version 10.0.1 paru le 26 juin 2020.

* + - 1. **Flutter**

Flutter est un Framework Open Source développé par Google. Sa première version Alpha a vu le jour en 2017. Depuis sa création, Flutter a été de plus en plus utilisée par les développeurs, et se place aujourd'hui parmi les technos de premier choix pour le développement d'applications mobiles.

## Partie Back End

Ces outils permettent la mise en place des applications « serveur » sous forme d’API REST

Consommable par des clients.

* + - 1. **Spring**

Spring est un Framework Java facilitant la mise en place et les tests de l’application. Il a été développé en 2003 par Pivotal Software et était à sa version 5.2.7 le 09 juin 2020. Selon Erik Gollot, Spring est un conteneur dit « léger » : « Spring est effectivement un conteneur dit

« Léger », c’est-à-dire une infrastructure similaire à un serveur d’application J2EE. Il prend donc en charge la création d’objets et la mise en relation d’objets par l’intermédiaire d’un fichier de configuration qui décrit les objets à fabriquer et les relations de dépendances entre ces objets. Le gros avantage par rapport aux serveurs d’application est qu’avec Spring, les classes n’ont pas besoin d’implémenter une quelconque interface pour être prises en charge

Par le Framework (au contraire des serveurs d’application J2EE et des [EJBs](https://fr.wikipedia.org/wiki/Enterprise_JavaBeans)). C’est en ce sens

Que Spring est qualifié de conteneur « léger ».

* + - 1. **JUnit**

Créé par [Kent Beck](https://fr.wikipedia.org/wiki/Kent_Beck) et [Erich Gamma](https://fr.wikipedia.org/wiki/Erich_Gamma), JUnit est un [Framework](https://fr.wikipedia.org/wiki/Framework) de [test unitaire](https://fr.wikipedia.org/wiki/Test_unitaire) pour le langage de programmation [Java](https://fr.wikipedia.org/wiki/Java_(langage)). Il permet d’implémenter le Test Driven Développent en Java.

Il existe deux types de test :

* + - * + TestCase (cas de test) sont des classes contenant un certain nombre de méthodes de tests. Un TestCase sert généralement à tester le bon fonctionnement d'une classe.
        + TestSuite permet d'exécuter un certain nombre de TestCase déjà définis.

NB : Dans un TestCase il n'y a pas de main méthode, chaque test est indépendant.

## **Base de données : PostgreSQL**

PostgreSQL est un système de gestion de base de données relationnelles et objets open source. Créé en 1996 par Michael Stonebraker, il est développé par la communauté PostgreSQL Global Développent Group. Il est à sa version 12.3 paru le 14 mai 2020.

* + 1. **Docker**

Docker est une plate-forme logicielle qui vous permet de concevoir, tester et déployer des applications rapidement. Docker intègre les logiciels dans des unités normalisées appelées [conteneurs](https://aws.amazon.com/fr/containers/), qui rassemblent tous les éléments nécessaires à leur fonctionnement, dont les bibliothèques, les outils système, le code et l'environnement d'exécution. Avec Docker, vous pouvez facilement déployer et dimensionner des applications dans n'importe quel environnement, avec l'assurance que votre code s'exécutera correctement.

## **Gestionnaire de codes sources : GIT (Serveur) et GitLab (Client)**

Git est un logiciel libre de versions décentralisé créé par Linus Torvald et s’avère être le plus populaire des logiciels de versions avec plus de douze millions d’utilisateurs. Paru en 2005, il était à sa version 2.27.0 le 1er juin 2020. Il est implémenté sous plusieurs forme dont le plus connu sont GitHub et GitLab.

# **Environnements de test**

DIGITALVIEW dispose de plusieurs environnements dédiés aux déploiements de ces applications en fonction des besoins :

* Utilisé par le

Métier

DIGITALVIEW PRODUCTION

* Utilisé par le support ou test rapide
* Utilisé à

L’installation des versions avant MEP

* MAJ des données tous les jours
* MAJ des données à la demande

DIGITALVIEW PRE- PROD

* Utilisé pour

Tous les tests

* Lier aux logiciels de test externe
* MAJ des données tous les dimanche soir
* MAJ des données à la demande

DIGITALVIEW INTEGRATION

# **Gestion des équipes projet agiles : JIRA**

La méthodologie Agile s'oppose généralement à la méthodologie traditionnelle waterfall. Elle se veut plus souple et adaptée, et place les besoins du client au centre des priorités du projet. A l'origine, cette approche a été créée pour les projets de développement informatique mais est utilisée aujourd’hui dans de nombreux autres domaines d’activités. L’idée est née suite à l'observation d'un taux d’échec élevé des projets dans les années 1990. C’est ainsi que 17 experts en développement logiciel se réunissent aux Etats-Unis en 2001 afin de mettre en commun leurs méthodes respectives. Le « Manifeste Agile » (*Agile Manifesto* en anglais) naît de cette rencontre et détermine les valeurs et les principes fondamentaux de la méthode

Agile. Ce manifeste prône les 4 valeurs fondamentales de la méthodologie :

* **L'équipe**, soit des individus et des interactions plutôt que des processus et des outils ;
* **L’application**, c'est-à-dire des fonctionnalités opérationnelles plutôt que de la documentation exhaustive ;
* **La collaboration** avec le client plutôt que la contractualisation des relations ;
* **L’acceptation du changement** plutôt que le suivi d'un plan.

De ces valeurs découlent les 12 principes généraux suivants :

* Satisfaire le client en priorité
* Accueillir favorablement les demandes de changement
* Livrer le plus souvent possible des versions opérationnelles de l’application
* Assurer une coopération permanente entre le client et l’équipe projet
* Construire des projets autour de personnes motivées
* Privilégier la conversation en face à face
* Mesurer l’avancement du projet en matière de fonctionnalité de l’application
* Faire avancer le projet à un rythme soutenable et constant
* Porter une attention continue à l’excellence technique et à la conception
* Faire simple
* Responsabiliser les équipes
* Ajuster à intervalles réguliers son comportement et ses processus pour être plus efficace La méthodologie estime qu’il est contre-productif de **planifier entièrement le projet dans les moindres détails avant de le développer**. Ainsi, la méthode Agile recommande de se fixer des objectifs à court terme. Le projet est donc divisé en plusieurs sous-projets. Une fois l'objectif atteint, on passe au suivant jusqu'à l'accomplissement de l'objectif final.

Autre point important : **la méthode Agile repose sur une relation privilégiée entre le client et l'équipe projet**. La satisfaction du client étant la priorité, l'implication totale de l'équipe et sa réactivité face aux changements du client comme aux imprévus sont nécessaires.

Dans le cadre de ce projet, nous utilisons la méthode SCRUM (qui signifie mêlée) qui est la plus utilisée des méthodes agiles. Considéré comme un cadre (Framework en anglais) de gestion de projet, Scrum se compose de plusieurs éléments fondamentaux :

* Rôles,
* Evénements,
* Artefacts,
* Règles.

Il s'agit d'une approche empirique (c'est-à-dire qui se base sur l'expérience), dynamique et participative de la conduite du projet. Au rugby, la mêlée est une phase indispensable car elle permet au jeu de repartir sur d'autres bases. Même chose pour Scrum : l'équipe se réunit quotidiennement lors d'une réunion de synchronisation, appelée mêlée quotidienne, afin de suivre l'avancement du projet.

L'équipe Scrum est **auto-organisée et pluridisciplinaire**, c'est-à-dire qu'elle choisit

La meilleure façon d’accomplir son travail et qu'elle possède toutes les compétences nécessaires à l'accomplissement du projet. La flexibilité, la créativité et la productivité de l'équipe sont ainsi optimisées.

L'équipe Scrum se compose de :

### Scrum Master

* **Product Owner** (ou propriétaire du produit en français)

### Equipe de développement

Le **Scrum Master** est responsable de la compréhension, de l'adhésion et de la mise en œuvre de la méthode Scrum qu'il maîtrise parfaitement. Il veille à ce que les principes et les valeurs de la méthodologie sont respectés. **C'est un facilitateur** qui aide à améliorer la communication au sein de l’équipe et cherche à maximiser la productivité et le savoir-faire de celle-ci. Il est considéré comme le coach de l'équipe de développement.

Le **Product Owner** porte la vision du produit à réaliser. Il travaille en interaction avec l’équipe de développement qui doit suivre ses instructions. C'est lui qui établit la priorité des fonctionnalités à développer ou à corriger, et qui valide les fonctionnalités terminées. Il est responsable de la gestion du product backlog (ou carnet de produit en français).

L'**équipe de développement** est chargée de transformer les besoins définis par le Product

Owner en fonctionnalités utilisables. Elle est pluridisciplinaire et possède toutes les compétences nécessaires pour réaliser le projet, sans faire appel à des prestations externes. Parmi ses membres, on trouve un architecte, un développeur, un testeur, etc. La taille idéale de l'équipe de développement est de 3 à 9 personnes. Il n'y a pas de notion de hiérarchie, toutes les décisions sont prises ensemble.



# Description arborescence projet de base

Une image contenant texte, Police, capture d’écran, nombre

Description générée automatiquement

**/** : Répertoire racine du projet

**/flyway** : Répertoire contenant les fichiers de migrations de la base de données

* + - All : Répertoire des fichiers de migrations générales
    - Dev : Répertoire des données de développement uniquement
    - Schema : Répertoire des fichiers de migrations du schéma de la base de données

**/frontend** : Répertoire des fichiers du frontend

* + - Guard : Répertoire de vérification de l’authentification et redirection des utilisateurs (protection des routes de l’application)
    - Modules : Répertoire des différents modules de l’application (administration, tableau de bord etc…)
    - Services : Répertoire des services partagé entre les différents modules (ressources API, interceptor, validator, web socket etc…)
    - Model : Répertoire des différents model de l’application (utilisateur, département, article etc.…)
    - Shared : Répertoire central des modules partagé

**/flutter** : Répertoire des fichiers du projet flutter

* + - Assets : Répertoire des fichiers (png, PDF etc..) de l’application
    - Ecrans : Répertoire des différents pages (accueil, profil, services etc..) de l’application
    - Model : Répertoire des différents model de l’application (utilisateur, département, article etc…)
    - Services : Répertoire des services partagé entre les différents Ecrans (ressources API, etc…)
    - Widgets : Répertoire des différents widgets (Button, Input, TextArea etc…) des écrans
    - Route : Répertoire des différents points d’entrée des écrans
    - Shared : Répertoire centralisé des services partagé et des serveurs API (message d’erreur, url-server et…)

**/src/main/java/ci.digitalview/projet** : Répertoire des fichiers backend

* + - Controller : Ce package contient les différentes méthodes d’accès aux ressources (API) de l’application. C’est le point d’entrée vers le métier.
    - Façade : Ce package contient l'implémentation des méthodes du repository de chaque entité et des services
    - Service : Ce package contient l'implémentation des méthodes du repository de chaque entité
    - Repository : Ce package contient les interfaces et méthodes d’accès aux données de la base de données (Objets persistants)
    - Security : Ce package contient les entités et services d’authentification à l’application
    - Entity : Ce package contient les entités et leurs propriétés
    - DTO : Ce package contient les objets de transfert de données des entités

# Mécanisme de traduction des applications

Il y a deux types de traduction : la traduction statique et la traduction dynamique.

La traduction statique consiste à utiliser des clés de traduction identiques et des valeurs de ses clés différentes selon la langue. Il ne reste qu’à identifier la langue sélectionnée pendant la session de l’utilisateur et d’afficher les mots appropriés. Ce type de traduction est utilisé pour les libellés.

La traduction dynamique quant à elle concerne les informations de la base données. Il est donc nécessaire d’identifier avant tout les données traduisibles. L’étape consiste à traduire ces données dans chaque langue.

L’application. Chaque requête contenant une donnée traduisible verra son résultat affecté par

La bonne traduction de l’information selon la langue sélectionnée par l’utilisateur.

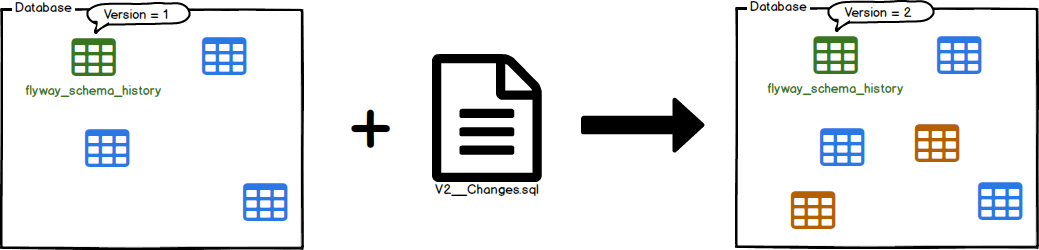
Le *document technique de traduction* donne plus de détails techniques sur le processus.

# Gestionnaire de migration flyway

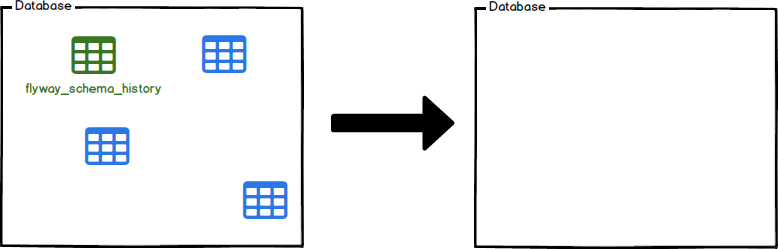
Dans un développent de logiciel en équipe, il est souvent utile d’utiliser un système de gestion de migrations de la base de données. Ce genre de gestion a l’avantage de fournir à chaque développeur exactement la même version de la base de données s’ils exécutent dans le même ordre les mêmes fichiers de migration.

Flyway est un gestionnaire de migration couramment utilisé avec les applications Spring Boot. Il est privilégié pour sa simplicité. Il utilise le langage SQL et est fondé sur sept (7) commandes de base :

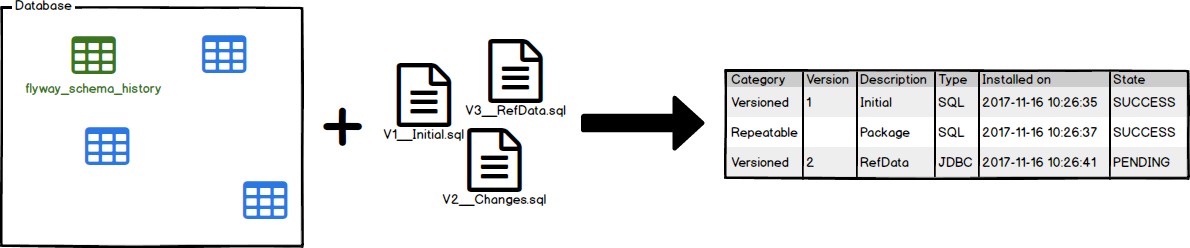
* + - **Migrate** : Migre le schéma vers la dernière version. Flyway créera automatiquement la table d'historique de schéma si elle n'existe pas.



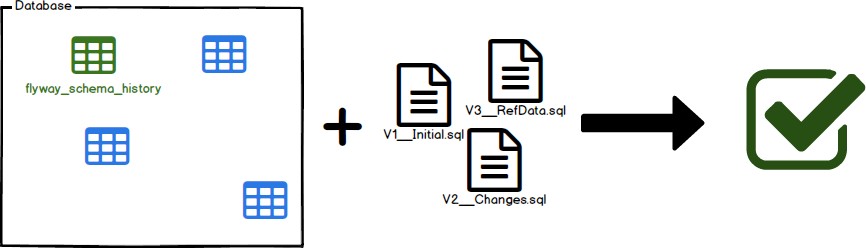
* + - **Clean** : Supprime tous les objets dans les schémas configurés.



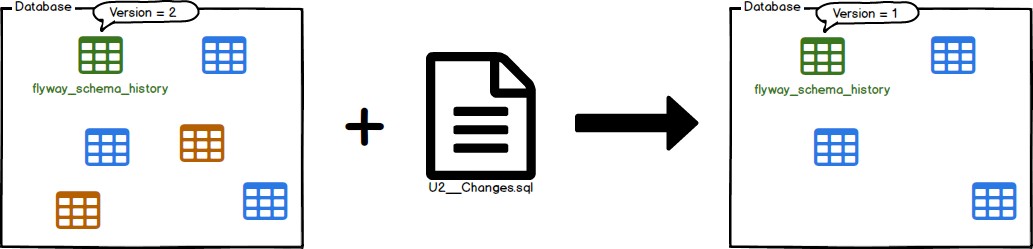
* + - **Info** : Affiche les détails et les informations d'état de toutes les migrations



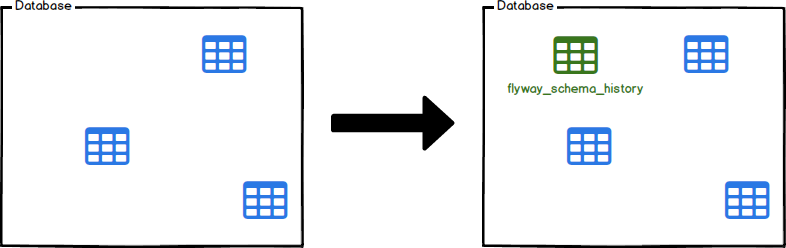
* + - **Validate** : Valide les migrations appliquées par rapport à celles disponibles.



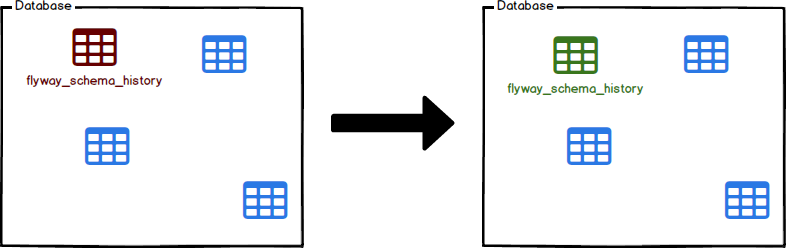
* + - **Undo** : Annule la migration la plus récemment appliquée.



* + - **Baseline** : Crée une version de migration de la base de données dans son état actuel.



* + - **Repair** : Répare la table d'historique de schéma.



# Mapping avec Mapstruct

Dans une architecture en couche, les données peuvent être manipulées sous diverses formes telles qu’entité, VO (Value Object) ou DTO (Data Transfert Object). Il est pratique d’utiliser une bibliothèque de mapping pour faciliter la transformation d’un type d’objet à l’autre. Ce mapping n’est pas utilisé dans la plupart des Applications Sigale. L’application Sigale Qualité a été la première à mettre en œuvre technologie. Ce mapping est fait avec la bibliothèque Mapstruct. Il s’agit est un générateur de code qui simplifie considérablement l'implémentation des mappages entre les types de bean Java basés sur une approche de convention sur la configuration. La documentation officielle est disponible à l’adresse suivante https://mapstruct.org/.



# Niveau 0 : Installation des prérequis.

### Les prérequis au niveau du Front End :

### IDE (VSode, IntelliJ)

NodeJs NPM

Angular (dernière version)

NG Prime

Flutter

Android Studio

### Les prérequis au niveau du Back End :

Spring et Spring Boot JUnit

### Les prérequis au niveau de la Base de données

PostgreSQL 13 & 14

### Environnement et outils de développement

IntelliJ IDEA Git **Tutoriels**

Nous pouvons recommander ces cours pour l’apprentissage de Angular. Ils ont fait leurs preuves dans l’apprentissage de certains nouveaux https://[www.youtube.com/playlist?list=PLl3CtU4THqPYNPElacumSgXXdzG0eSwjh](http://www.youtube.com/playlist?list=PLl3CtU4THqPYNPElacumSgXXdzG0eSwjh)

# Niveau 1 : Appréhender l’environnement de développement

### TP 1

* + 1. Créer un projet Spring Boot.
    2. Inclure les bibliothèques Flyway et Mapstruct.
    3. Créer un projet Angular dans un dossier nommé frontend situé à la racine de

L’application Spring Boot.

* + 1. Inclure la bibliothèque PrimeNg dans l’application.
    2. Afficher une page d’accueil avec une barre de menu comportant uniquement le menu Accueil (Utiliser PrimeNg)

# Niveau 2 : Réaliser une interface suivant l’arborescence DIGITALVIEW

### TP 2

* + 1. Créer la table personne avec les attributs nom, prénoms et âge (Utiliser flyway)
    2. Ajouter cinq (5) occurrences de personnes dans la table (Utiliser flyway)
    3. Créer les couches contrôleur, service et repository.
    4. Créer l’entité et le VO personne

### TP 3

Mettre en place le CRUD de personne.

### TP 4

1. Ajouter le menu personne à la barre de menu.
2. Afficher la liste des utilisateurs.
3. Mettre en place un formulaire d’ajout et de modification de l’utilisateur.
4. Ajouter des boutons de modification et de suppression de personne.

# Niveau 3 : Gérer les critères de recherche

* + 1. Ajouter une table département possédant les attributs code et désignation.
    2. Ajouter 3 lignes de données dans la table département.
    3. Ajouter une contrainte d’intégrité fonctionnelle entre les tables personne et

Département.

* + 1. Faire une reprise de données de la table personne.
    2. Ajouter un champ de type autocomplete au formulaire d’ajout/modification de personne.
    3. Afficher une nouvelle colonne dans la liste des personnes contenant la désignation de leur département.
    4. Ajouter un tri sur la colonne nom et un filtre sur la colonne âge (mineur en dessous de 18 ans et majeur ayant 18 ans et plus).

# Niveau 4 : Gérer les tests unitaires

### TP 5

Soit la classe suivante :

public class Calcul { private float a ; private float b ;

public Calcul(float a, float b) { this.a = a;

this.b = b;

}

public float additionner(float a, float b) { return a + b;

}

public float soustraire(float a, float b) { return a - b;

}

public float multiplier(float a, float b) { return a \* b;

}

public float diviser(float a, float b) throws Exception { if(b != 0){

return a + b;

}

throw new Exception();

}

public float carre(float a) { return multiplier(a, a);

}

public float identiteRemarquable(float a, float b) { float a2 = multiplier(a, a);

float b2 = multiplier(b, b); float ax2 = multiplier(2, a);

return additionner(additionner(a2, multiplier(ax2, b)), b2);

}

}

Ecrire les tests unitaires liés à chaque cas d’utilisations.

### TP 6

Ecrire les tests unitaires sur toutes les fonctions des services de l’application Spring Boot.

### TP 7

Ecrire les tests unitaires sur toutes les fonctions du composant du menu personne.

# Niveau 5 : Prise en main de l’architecture

### TP 8

Cloner, démarrer et se connecter aux applications Sigale Achats, Pricing, Facturation, Société, Qualité et Dons.

Prendre un ticket de la backlog.