DOCUMENT TECHNIQUE DE FORMATION SUR LA GRANDE DISTRIBUTION

V 2.2

Cas de l'enseigne

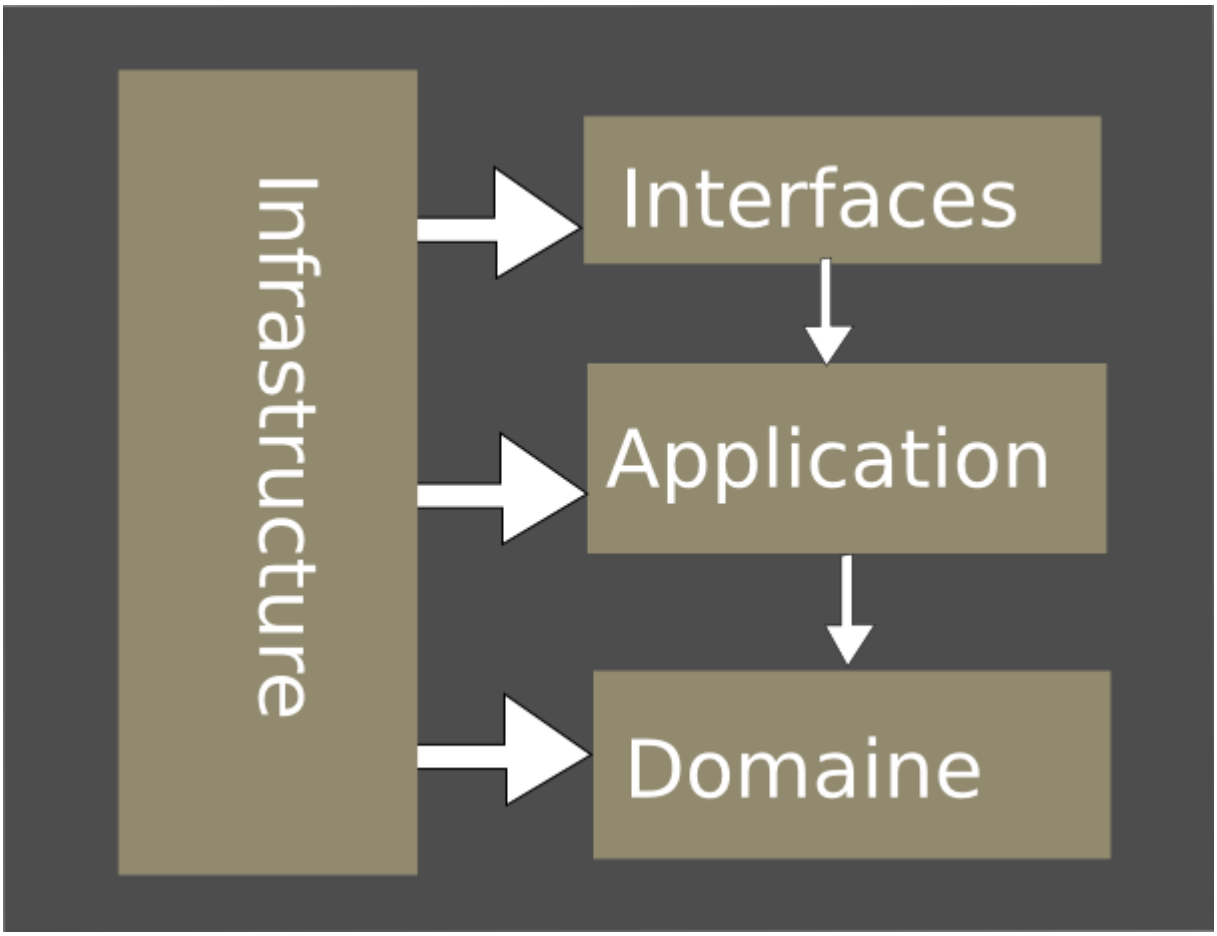
GRAND FRAIS

SOMMAIRE

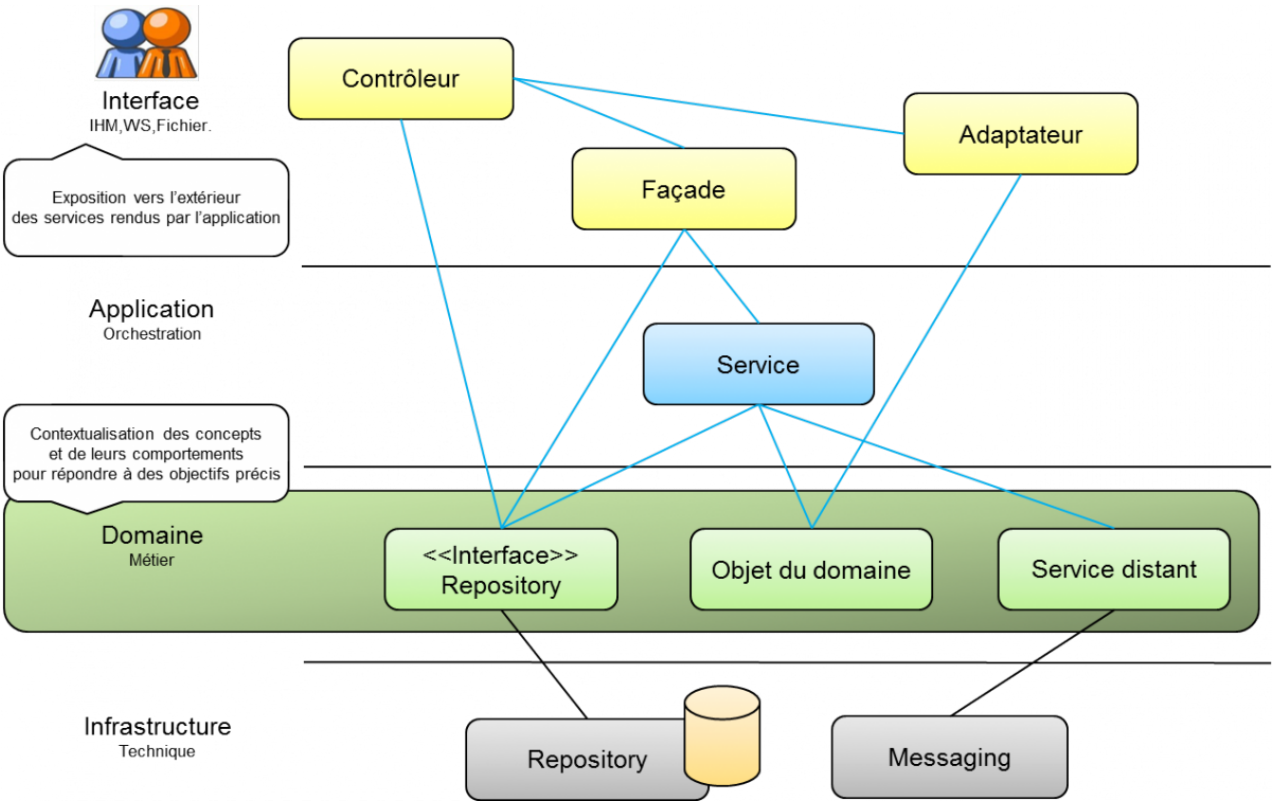
1 ENVIRONNEMENT TECHNIQUE

1.1 Architecture technique

L'approche choisie par GFIT dans la réalisation des applications SIGALE est une 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 :



1.1.1 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 front-end exposés aux autres applicatifs.

1.1.2 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

1.1.3 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).

1.1.4 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, …).

1.2 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 Sigale, ces outils peuvent être réparties en trois catégories :

1.2.1 Partie Front End

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

1.2.1.1 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/.

1.2.1.2 Angular 7

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.

1.2.2 Partie Back End

Ces outils permettent la mise en place des applications « serveur » sous forme d'API REST consommable par des clients.

1.2.2.1 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). C'est en ce sens que Spring est qualifié de conteneur « léger ».

1.2.2.2 JUnit

Créé par Kent Beck et Erich Gamma, JUnit est un framework de test unitaire pour le langage de programmation Java. Il permet d'implémenter le Test Driven Development 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 method, chaque test est indépendant.

1.2.3 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 Development Group. Il est à sa version 12.3 paru le 14 Mai 2020.

1.2.4 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.

1.3 Environnements de test

GFIT dispose de plusieurs environnements dédiés aux déploiements des applications SIGALE en fonction des besoins :

SIGALE PRODUCTION	SIGALE PRE-PROD	SIGALE INTEGRATION	SIGALE FORMATION	SIGALE DEMO	SIGALE FABRICATION
•Utilisé par le métier	•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	•Utilisé pour tous les tests •Lier aux logiciels de test externe •MAJ des données tous les dimanche soir •MAJ des données à la demande	•Utilisé pour formation, recette métier •Utilisé pour test en parallèle •MAJ des données à la demande	•Utilisé lors des démonstration de fin de sprint	•Utilisé pour les Test fonctionnels des développeurs •Utilisation d'un jeu de données réduit

1.4 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 né 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**

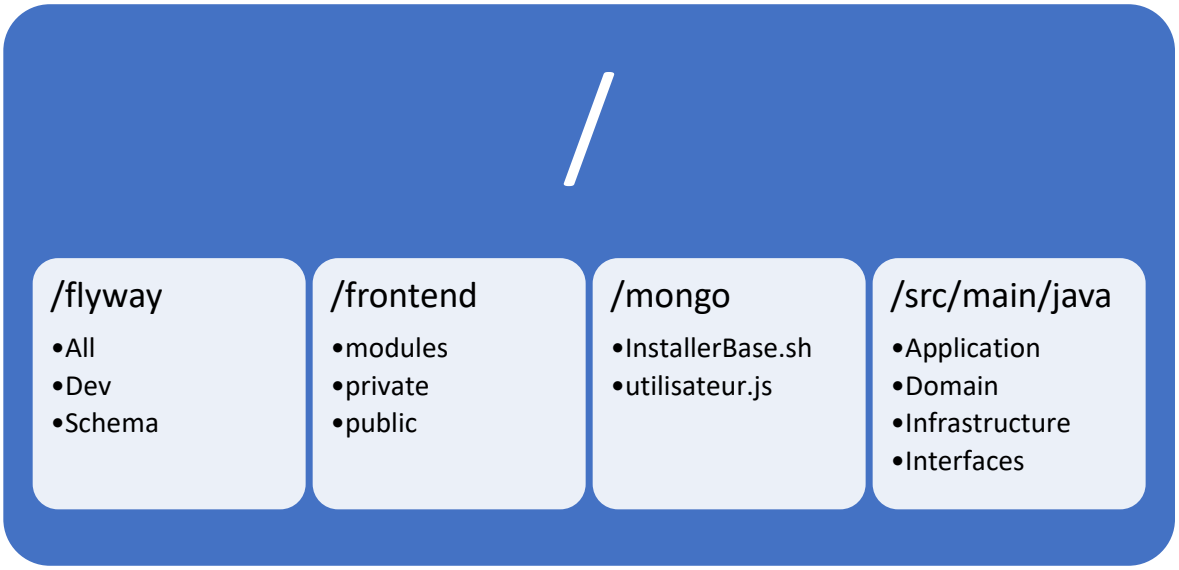
Le **Scrum Master** est responsable de la compréhension, de l'adhésion et de la mise en œuvre de la méthode Scrum qu'il maîtrise parfaitement. Il veille à ce que les principes et les valeurs de la méthodologie sont respectés. **C'est un facilitateur** qui aide à améliorer la communication au sein de 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.

2 PRESENTATION PROJET TYPE SIGALE

2.1 Description arborescence projet de base



/ : 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

- Private : Répertoire des fichier HTML et contrôleurs JS de l'application
- Public : Répertoire des fichiers de style
- Modules : Répertoire du projet Angular

/mongo : Répertoire contenant les fichiers relatifs aux utilisateurs de l'application

- installerBase.sh : programme de création de la base de données des utilisateurs
- utilisateur.js : fichier de configuration des utilisateurs

/src/main/java : Répertoire des fichiers backend

- Application : Ce package contient les VO et services des entités
- Domain : Ce package contient les entités et leurs repositories
- Infrastructure : Ce package contient l'implémentation des méthodes du repository de chaque entité
- Interfaces : Ce package contient l'ensemble des contrôleurs et des façades de l'application

2.2 Cohabitation Angular 7 / AngularJs

Le front-end des applications Sigale était géré au préalable par AngularJs. Il s'agit d'une librairie basée sur le langage de programmation JavaScript qui permet de dynamiser les pages web notamment en y ajoutant des évènements déclencheurs de traitements spécifiques grâce au caractère asynchrone de ce langage.

L'évolution rapide et prononcée du framework Angular a quasiment obligé les architectes des applications Sigale à intégrer cette technologie dans les projets sans toutefois faire disparaître AngularJs. C'est ainsi qu'est né le concept de « cohabitation Angular 7 / AngularJs ». Il s'agit de faire fonctionner ensemble deux technologies pratiquement similaires en termes de résultat, mais totalement différentes en termes d'architecture. Concrètement, il s'agit de gérer la logique front-end de certaine page avec AngularJs tandis que d'autres, parfois même des composants de la page AngularJs, sont quant à eux gérées par Angular.

Les détails de la mise en place de ce mécanisme sont présentés dans le *Document de Cohabitation de AngularJs et Angular 7 GFIT.*

2.3 Mécanisme de traduction des applications

L'expansion du groupe Grand Frais a permis l'exportation de ses enseignes hors des frontières de la France. Ainsi, nous Grand Frais se retrouve dans d'autres Pays tels que la Belgique, le Luxembourg ou encore l'Italie. La pénétration du marché italien a contraint les décisionnaires de GFIT à traduire les applications Sigale afin de les adapter aux nouveaux utilisateurs Italiens. D'un point de vue technique, 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. Cette traduction se fait grâce à l'application Sigale Traduction. Cette traduction est ensuite chargée dans la mémoire cache au démarrage de

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.

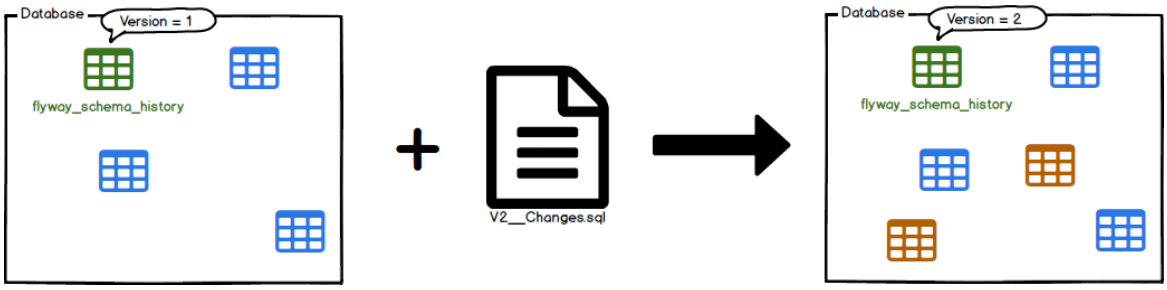
Le *document technique de traduction* donne plus de détails techniques sur le processus.

2.4 Gestionnaire de migration flyway

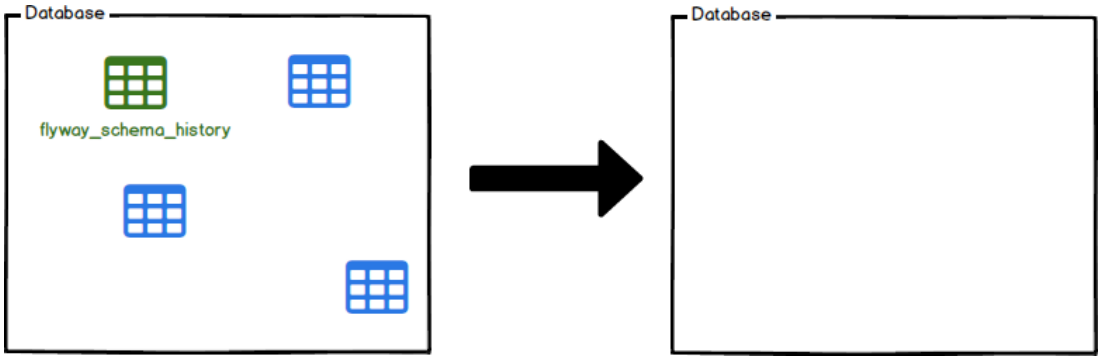
Dans un développent de logiciel en équipe, il est souvent utile d'utiliser un système de gestion de migrations de la base de données. Ce genre de gestion a l'avantage de fournir à chaque développeur exactement la même version de la base de données s'ils exécutent dans le même ordre les mêmes fichiers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

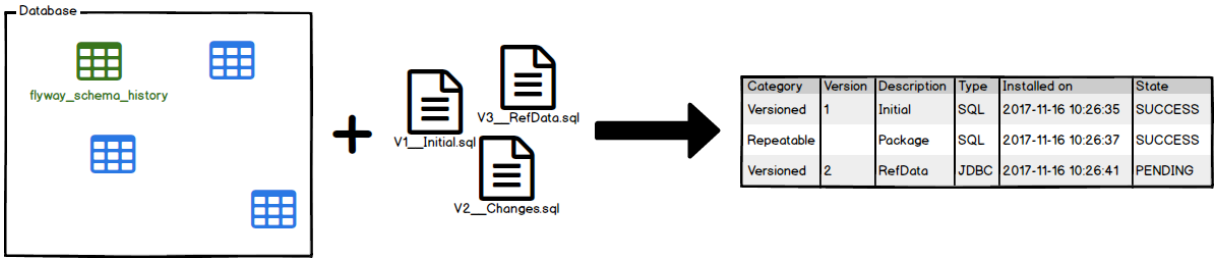
- **Migrate** : Migre le schéma vers la dernière version. Flyway créera automatiquement la table d'historique de schéma si elle n'existe pas.



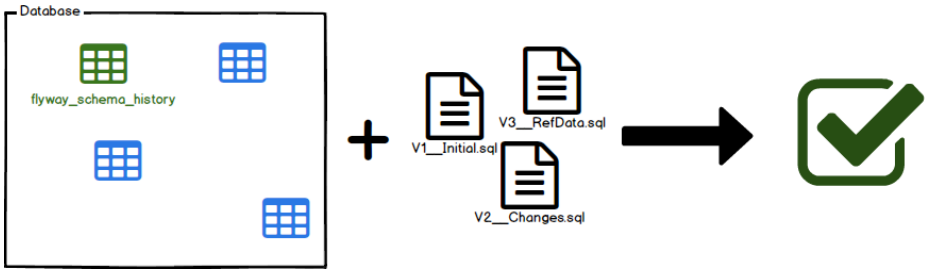
- **Clean** : Supprime tous les objets dans les schémas configurés.



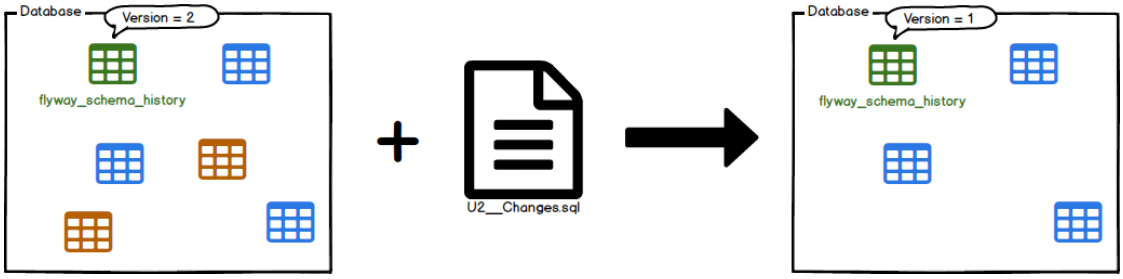
- **Info** : Affiche les détails et les informations d'état de toutes les migrations



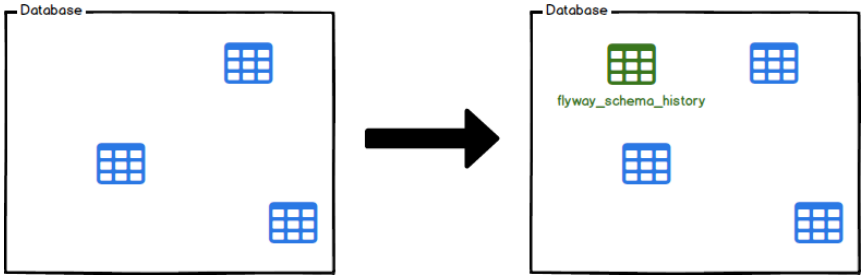
- **Validate** : Valide les migrations appliquées par rapport à celles disponibles.



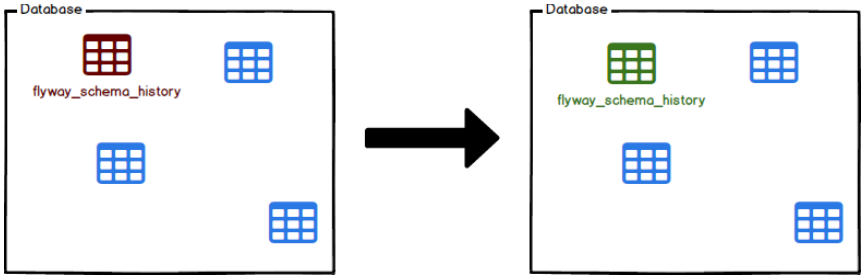
- **Undo** : Annule la migration la plus récemment appliquée.



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



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



2.5 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/.

3 CAS PRATIQUES

3.1 Niveau 0 : Installation des prérequis.

Les prérequis au niveau du Front End :

NodeJs

NPM

Angular 11

NG Prime

Les prérequis au niveau du Back End :

Spring et Spring Boot

JUnit

Les prérequis au niveau de la Base de données

PostgreSQL 11 & 12

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

3.2 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.
4. Inclure la bibliothèque PrimeNg dans l'application.
5. Afficher une page d'accueil avec une barre de menu comportant uniquement le menu Accueil (Utiliser PrimeNg)

3.3 Niveau 2 : Réaliser une interface suivant l'arborescence SIGALE

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.

3.4 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.
4. Faire une reprise de données de la table personne.
5. Ajouter un champ de type autocomplete au formulaire d'ajout/modification de personne.
6. Afficher une nouvelle colonne dans la liste des personnes contenant la désignation de leur département.
7. 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).

3.5 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.

3.6 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.