D’une archi web MVC à une archi clean hexagonale

Hands-On “Happy Town”

Céline Gilet

Faisons connaissance

Céline Gilet

@celinegilet

- Tribu Software Craftsmanship à OCTO Technology
- Développeuse depuis + de 10 ans
- Conseil & accompagnement sur les pratiques de qualité logicielle
- Formation (Test Driven Development, Clean Code, Legacy Code..)

Objectifs de la session

- Les limites d'un design applicatif basé sur une organisation en couches techniques
- Les principes de l'architecture hexagonale et de la clean architecture pour simplifier la maintenabilité et l'évolutivité
- Mise en pratique sur une base de code
  - Isoler le coeur métier de tout le reste (outils, frameworks, briques d'infrastructure)
  - S'abstraire des accès à une base de données / un système de fichiers / un serveur de mail
- Les apports sur le quotidien de développeur

Programme du Hands-On “Happy Town”

- ❏ Prise en main du sujet
  - ❏ Installation et présentation du projet
  - ❏ Schéma de l’architecture actuelle

  15 min
- ❏ Maintenabilité et évolutivité du code
  - ❏ Revue collective du code
  - ❏ Limites actuelles

  15 min
- ❏ Vers une archi clean hexagonale
  - ❏ Clean Architecture / Hexagonale Architecture
  - ❏ Mise en pratique

  60 min
- ❏ Debrief et conclusions

Happy Town
Prise en main du sujet

Installation du projet

- https://github.com/celinegilet/happy-town
- Le “Pitch”

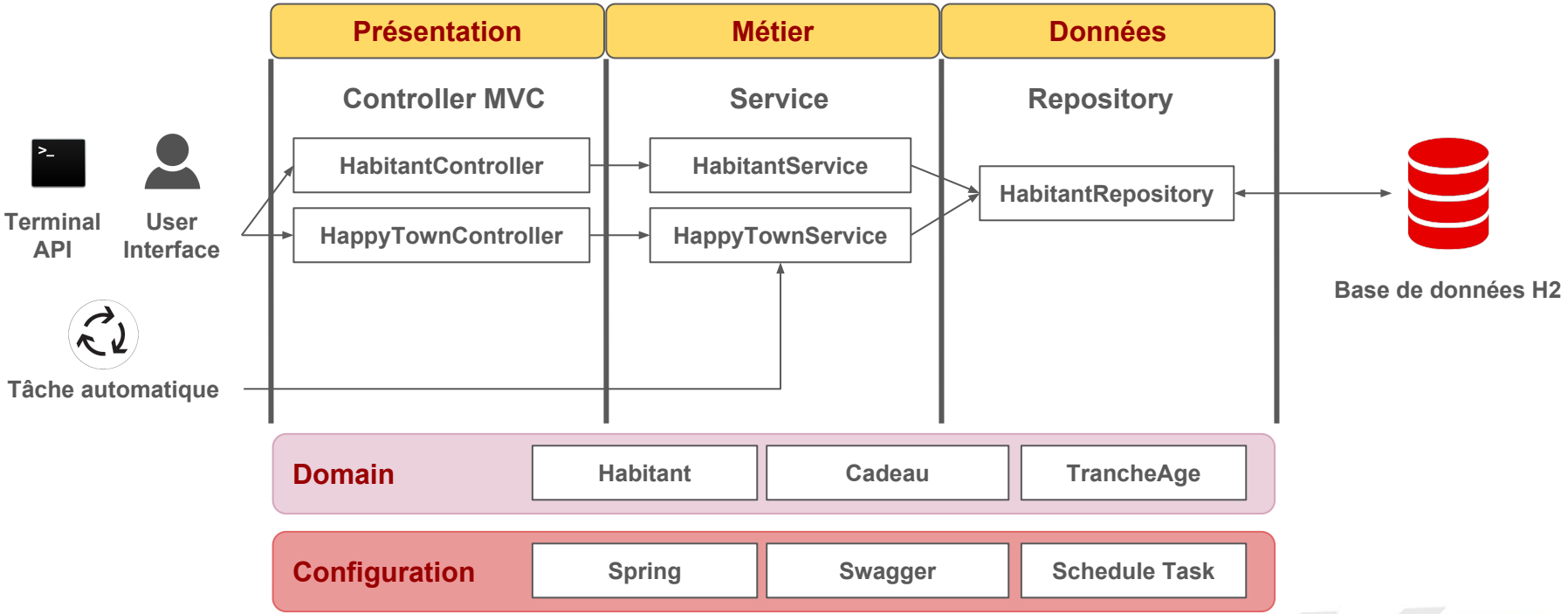
Pour accueillir dignement ses nouveaux habitants, le conseil municipal de *HappyTown* a décidé d’**offrir un cadeau** à **tous ses habitants** qui soufflent **leur première bougie dans la commune**.

Le rôle de notre application est donc :

- De **sélectionner les habitants éligibles** à l’obtention d'un cadeau (ils ont emménagés depuis plus de 1 an)
- Pour **chacun** des **habitants éligibles** :
  - **Trouver** le **cadeau approprié** en fonction de son **âge** : il y a des **cadeaux différents** par **tranche d’âge**
  - **Envoyer** un **mail** annonçant l'attribution du cadeau
- **Envoyer** un **mail récapitulatif** au service cadeau de la mairie avec **tous les cadeaux attribués de la journée**

Présentation du projet

Schéma de l'architecture actuelle : 3-Tiers



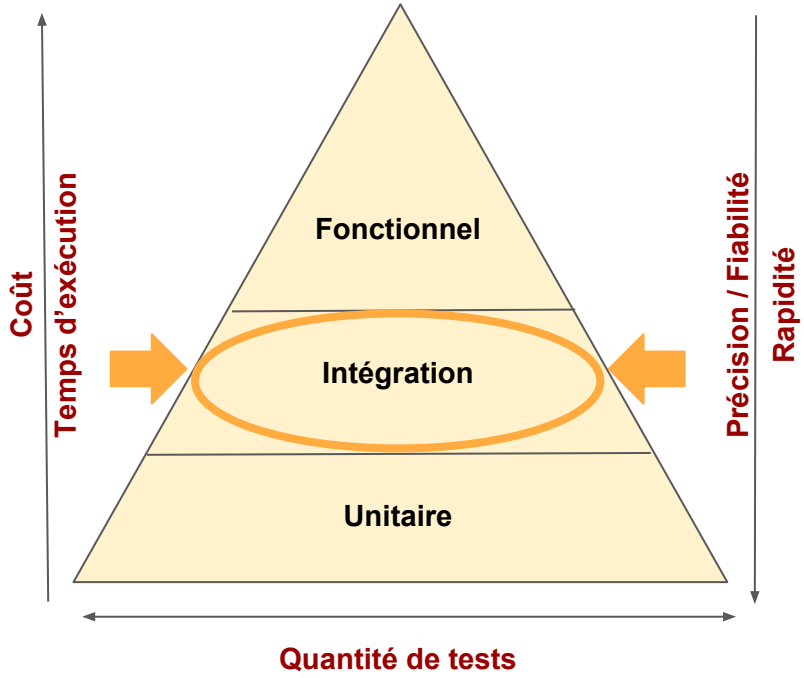
Maintenabilité et évolutivité du code

Revue collective du code

Limites actuelles

- Du code centré autour des frameworks
- Un découpage et une architecture par responsabilité technique (controller / service / repository)
- Le modèle métier est à la fois le modèle de stockage et le modèle de présentation
- Une perte de la logique métier et des services de l'application
- Un fort couplage et une adhérence aux composants d'infrastructure (serveur de mail, système de fichiers)
- Des difficultés à écrire des tests rapidement qui représentent le métier
- Des tests écrits à posteriori avec une stratégie de couverture de code (absence de TDD / design émergent)
- Évolutions de + en + difficile (en durée et complexité)

Limites actuelles



- Une pyramide de tests sans base
- Des tests avec un caractère aléatoire
- Une nécessité de démarrer un serveur de mail pour tester les règles métiers
- Une stratégie de tests basée sur le fonctionnement de frameworks

Vers une architecture clean hexagonale

Dans quel but ?

- La valeur d'une application réside dans ses cas d'utilisation et ses services métiers
- Isoler et protéger cette valeur des changements et évolutions techniques
- Le domaine métier d'une application n' évolue pas au même rythme que les éléments connexes (frameworks, base de données, infra…)
- Une prise en compte des aspects techniques à posteriori
- Pas de dispersion de la logique métier dans plusieurs couches
- Des tests ciblés sur une problématique précise : rapidité, fiabilité et robustesse
- Un découpage par responsabilité pour favoriser les évolutions et accélérer le cycle des déploiements

Clean architecture / Hexagonale architecture

Clean Architecture

- Uncle Bob Martin -

*The **center** of your application is **not the database**. Nor is it one or more of the **frameworks** you may be using.*
The center of your application is the use cases of your application

***L'élément clé** d'une application **ne réside pas** dans sa **base de données** et les **frameworks** utilisés.*
Les use-cases d'une application sont l'élément central

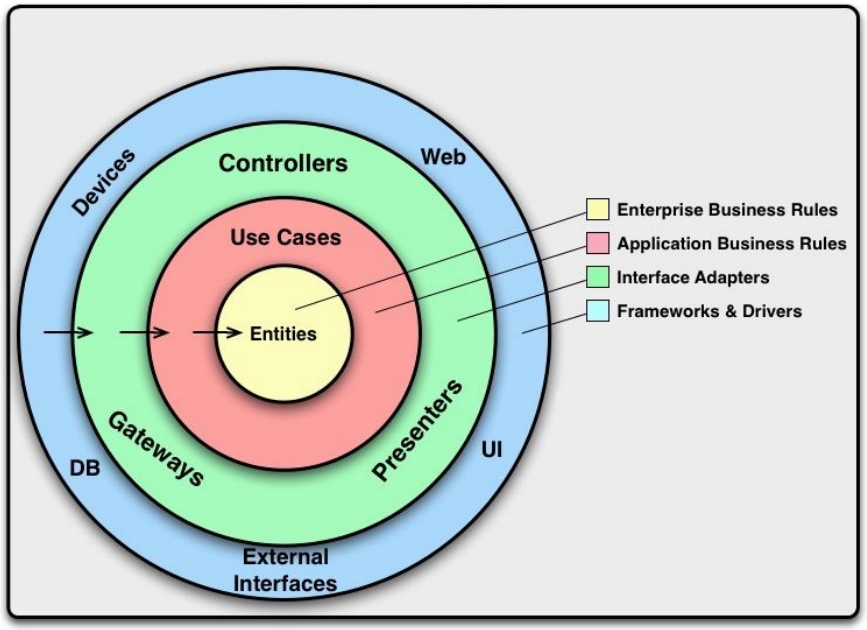
Hexagonale Architecture

- Alistair Cockburn -

Allow** an **application** to **equally be driven** by **users**, **programs**, **automated test** or **batch scripts**, and to be **developed** and **tested** in **isolation** from its eventual **run-time devices** and **databases

Permettre** à une **application** d'être **pilotée** aussi bien par des **utilisateurs** que par des **programmes**, des **tests automatisés** ou des **scripts batchs**, et d'être **développée** et **testée** en **isolation** de ses éventuels **systèmes d'exécution** et **bases de données

Clean architecture

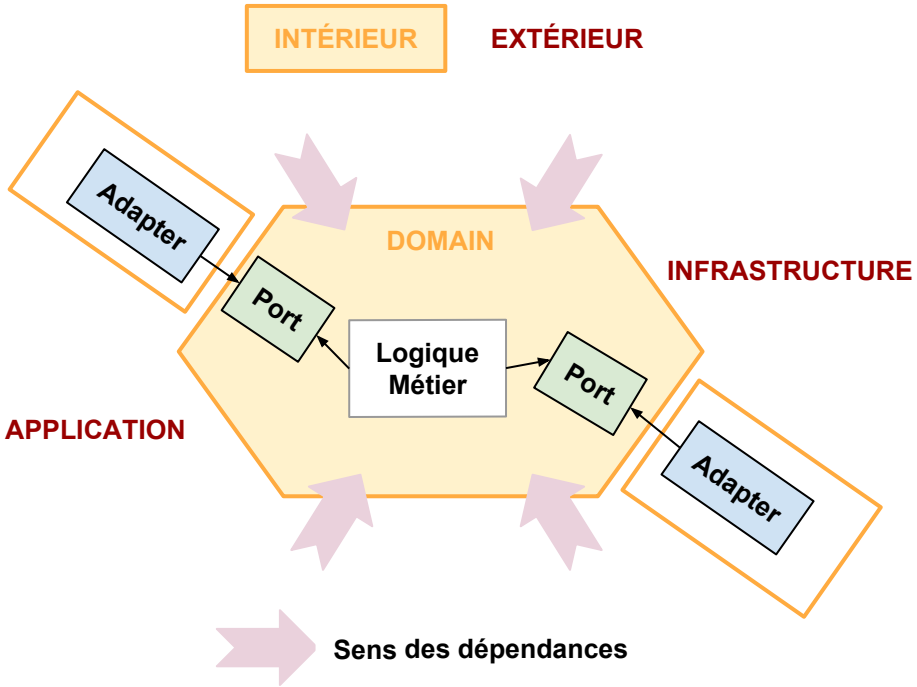


Source :
http://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html

Principes

- Un centre contenant la logique métier sans frameworks ni annotations
  - Entities - Objets du domaine
  - Use Cases - Services proposés par l'application
- Des points d'entrée (Entrypoints) pour déclencher les use cases : API Rest, interface graphique, jobs
- Des fournisseurs de données (DataProviders) pour récupérer et stocker les données : BDD, périphériques réseau, fichiers, systèmes externes
- Les éléments de configuration (Configuration)

Hexagonale architecture



Principes

- Intérieur vs Extérieur
- Découpage en 3 zones distinctes
  - Application - les moyens d'interactions pour piloter / déclencher le métier
  - Domain - la logique métier
  - Infrastructure - les besoins et dépendances nécessaires au métier (BDD, Systèmes extérieurs, File System)
- Sens des dépendances uniquement vers l'Intérieur : le Domain
- Isolation des frontières par des Ports et Adapters (Interfaces)

Mise en pratique

Métier et valeur de l'application

- Attribuer un cadeau aux habitants de HappyTown
- Récupérer les informations des habitants de HappyTown

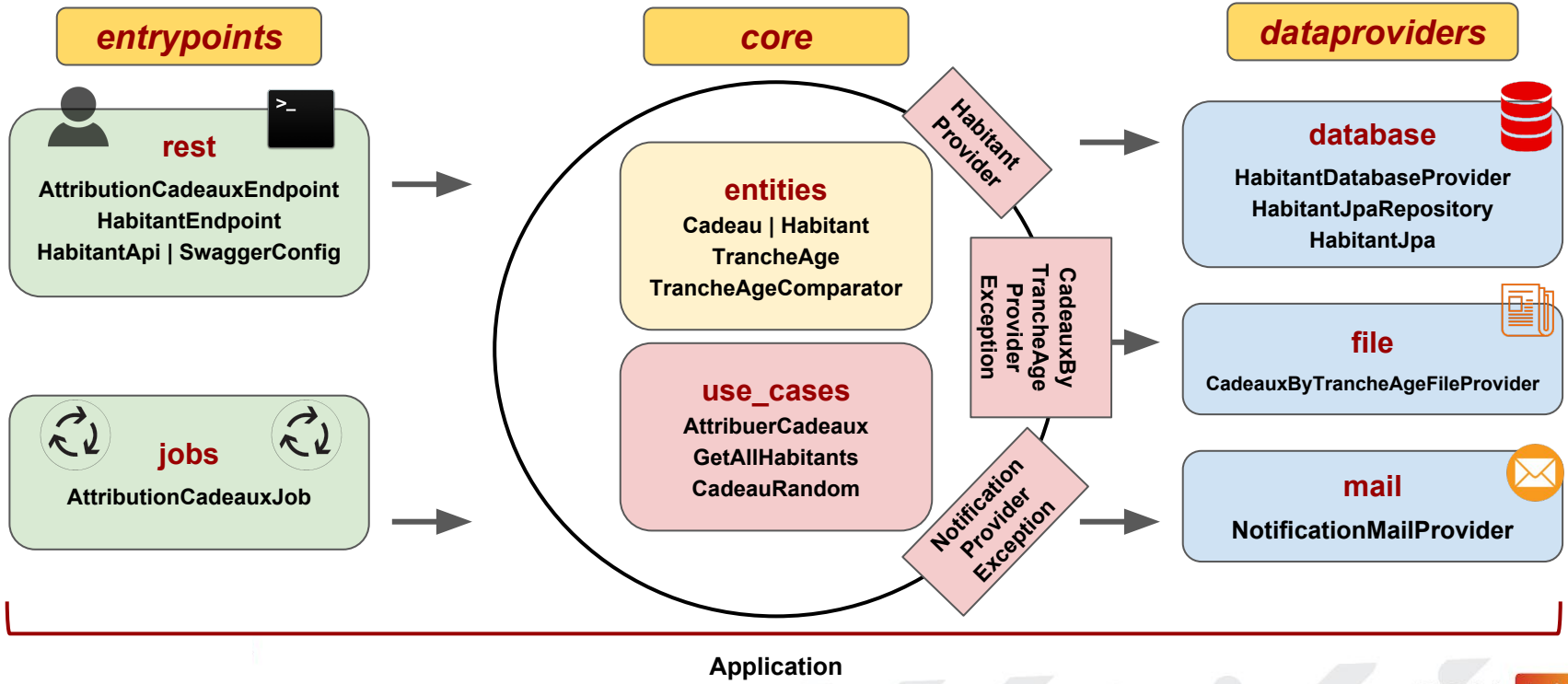
Déclencheurs

- Un terminal console (CURL endpoint)
- Une interface graphique (swagger-ui.html)
- Une tâche automatisée
- Des tests

Fournisseurs de données

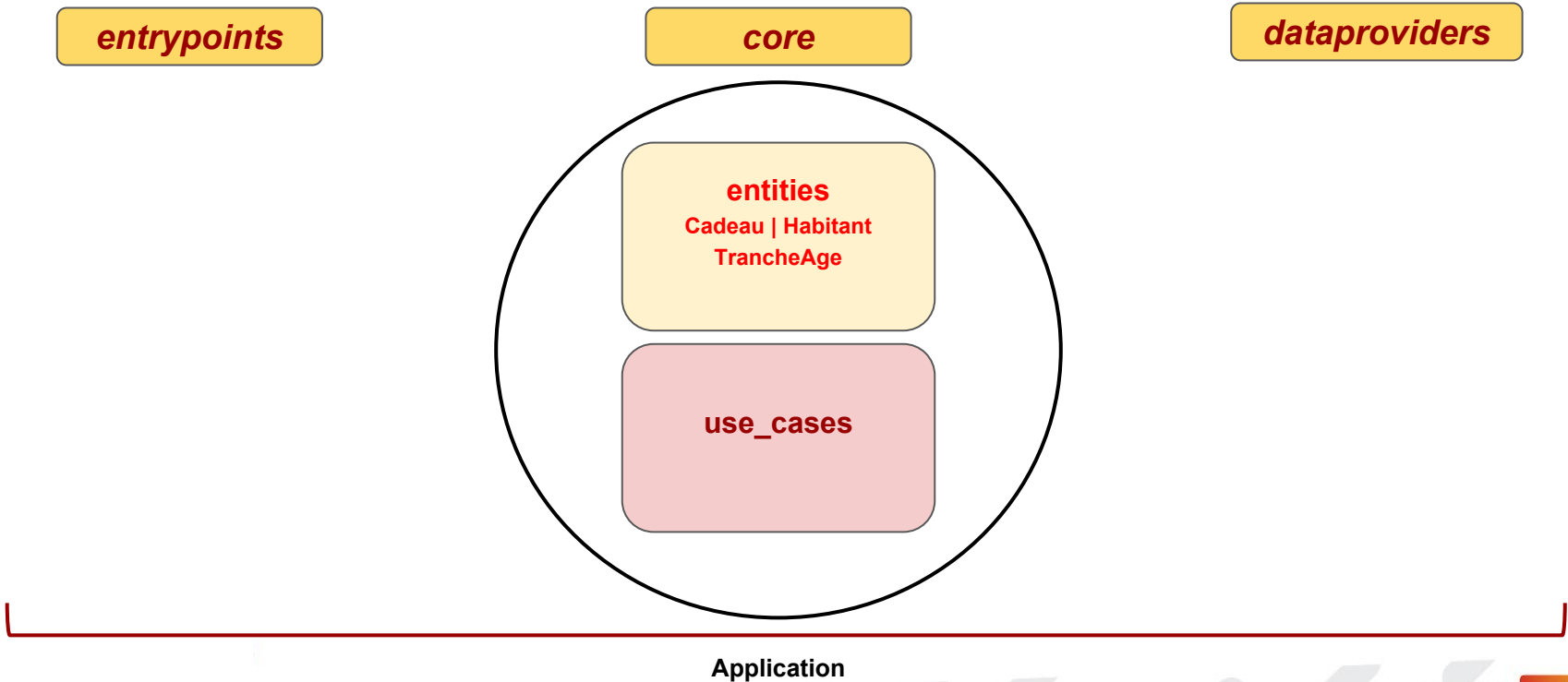
- Base de données H2 contenant les habitants
- Fichier contenant les cadeaux par tranche d'âge
- Serveur de mail

Clean archi : Schéma cible



Clean archi : Schéma en construction

Step 01



Clean archi : Réorganisation du code

Step 01

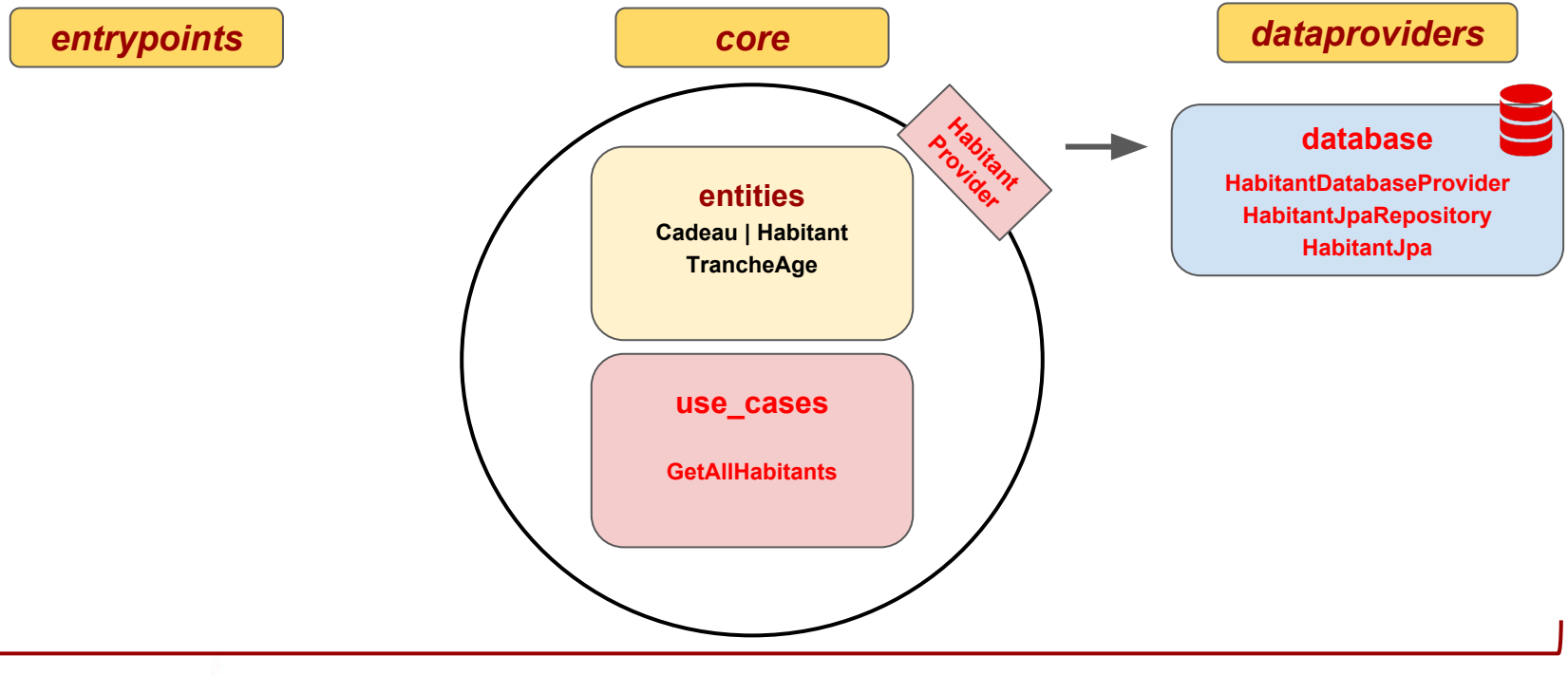
- com.happytown
  - configuration
    - ScheduleTasks
    - SwaggerConfig
  - controller
    - HabitantController
    - HappyTownController
  - domain
    - Cadeau
    - Habitant
    - TrancheAge
  - repository
    - HabitantRepository
  - service
    - HabitantService
    - HappyTownService
    - TrancheAgeComparator

- ❑ **Création de la nouvelle structure de packages**
  - ❑ **core (entities + use_cases)**
  - ❑ **entrypoints**
  - ❑ **dataproviders**
- ❑ **Déplacement des objets de domain vers entities**
- ❑ **Suppression des frameworks / annotations (Lombok, Javax Validation)**

- com.happytown
  - configuration
    - ScheduleTasks
    - SwaggerConfig
  - controller
    - HabitantController
    - HappyTownController
  - core
    - entities
      - Cadeau
      - Habitant
      - TrancheAge
    - use_cases
  - dataproviders
  - entrypoints
  - repository
    - HabitantRepository
  - service
    - HabitantService
    - HappyTownService
    - TrancheAgeComparator

Clean archi : Schéma en construction

Step 02



Clean archi : Récupération des habitants

Step 02

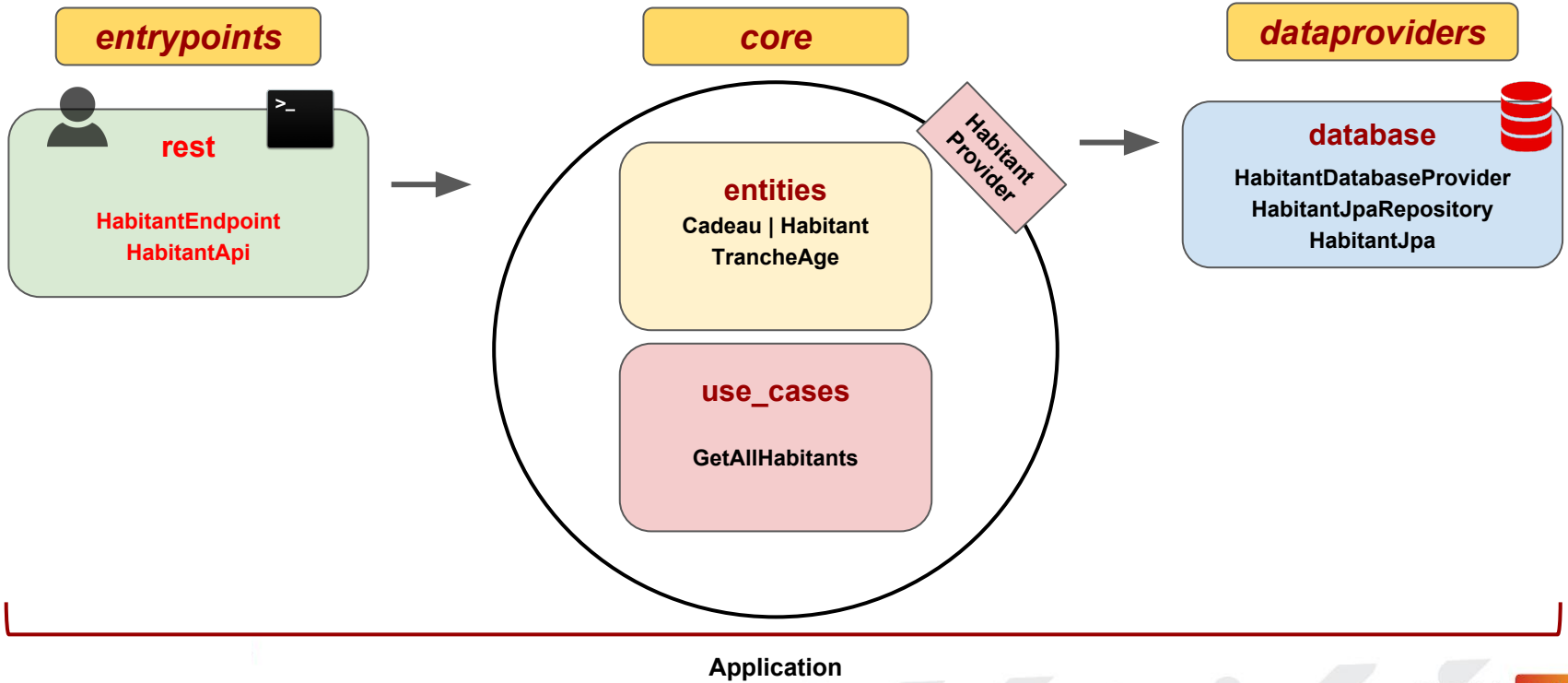
- com.happytown
  - configuration
    - ScheduleTasks
    - SwaggerConfig
  - controller
    - HabitantController
    - HappyTownController
  - core
    - entities
      - Cadeau
      - Habitant
      - TrancheAge
    - use_cases
  - dataproviders
  - entrypoints
  - repository
    - HabitantRepository
  - service
    - HabitantService
    - HappyTownService
    - TrancheAgeComparator

- ❑ **Ajout d'un use case pour récupérer tous les habitants : GetAllHabitants**
- ❑ **Nécessité d'avoir un nouveau fournisseur de données pour les habitants : HabitantProvider**
- ❑ **Implémentation du fournisseur de données des habitants en base de données : HabitantDatabaseProvider**
- ❑ **Séparation entre les objets du domaine métier : Habitant et le modèle de stockage : HabitantJpa**
- ❑ **Suppression du package repository**

- com.happytown
  - configuration
    - ScheduleTasks
    - SwaggerConfig
  - controller
    - HabitantController
    - HappyTownController
  - core
    - entities
      - Cadeau
      - Habitant
      - TrancheAge
    - use_cases
      - GetAllHabitants
      - HabitantProvider
  - dataproviders.database
    - HabitantDatabaseProvider
    - HabitantJpa
    - HabitantJpaRepository
  - entrypoints
  - service
    - HabitantService
    - HappyTownService
    - TrancheAgeComparator

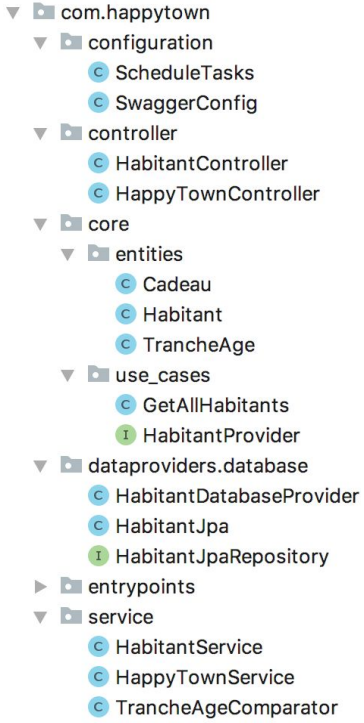
Clean archi : Schéma en construction

Step 03

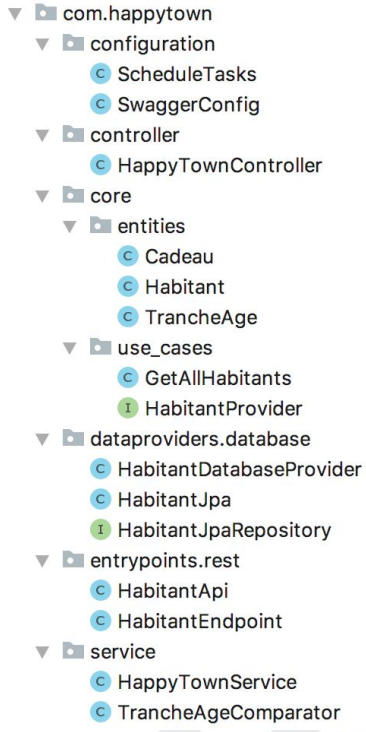


Clean archi : Visualisation des habitants

Step 03

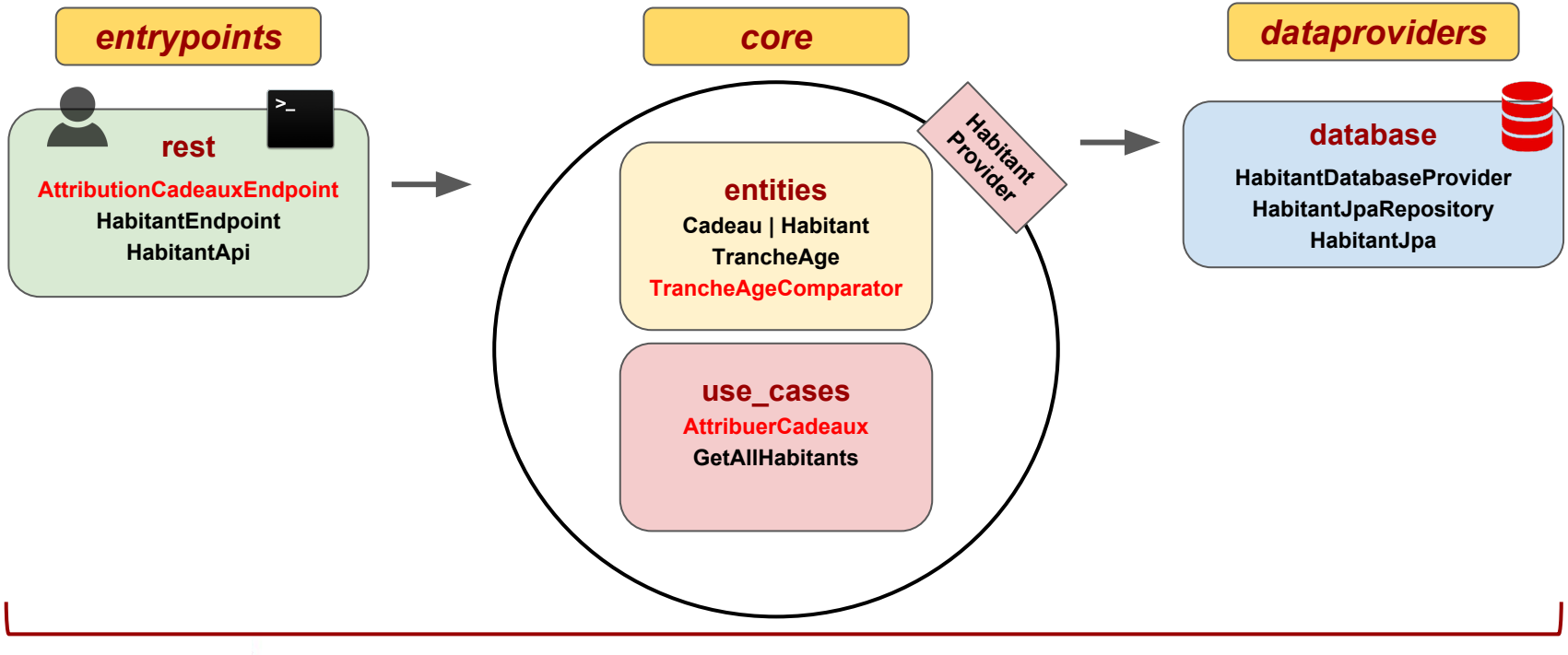


- **Ajout d'un entrypoint rest pour récupérer la liste des habitants : HabitantEndpoint**
- **Séparation entre les objets du domaine métier : Habitant et le modèle de présentation : HabitantApi**
- **Suppression de HabitantController et HabitantService**
- **L'objet Habitant est maintenant agnostique de tout framework technique (présentation, persistence)**



Clean archi : Schéma en construction

Step 04



Clean archi : Attribution des cadeaux

Step 04

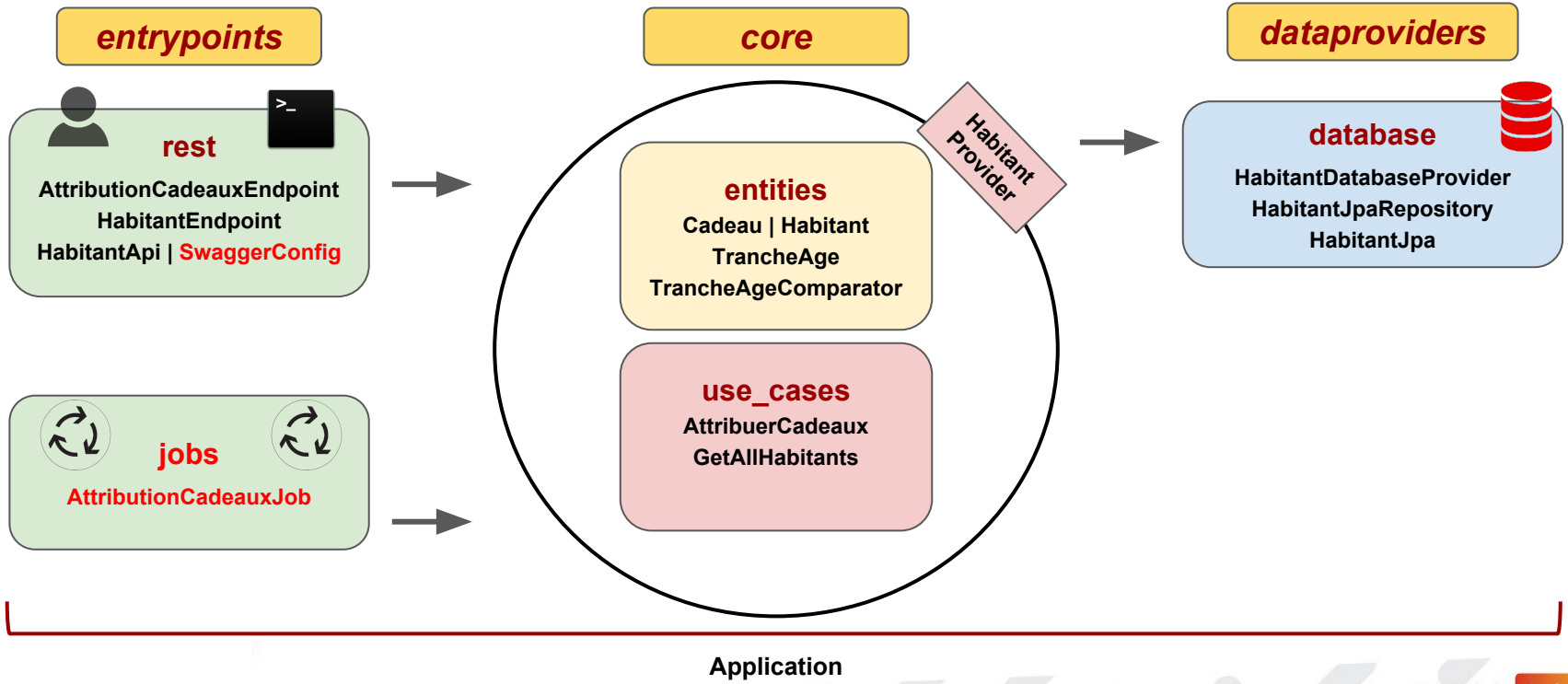
- com.happytown
  - configuration
    - ScheduleTasks
    - SwaggerConfig
  - controller
    - HappyTownController
  - core
    - entities
      - Cadeau
      - Habitant
      - TrancheAge
    - use_cases
      - GetAllHabitants
      - HabitantProvider
  - dataproviders.database
    - HabitantDatabaseProvider
    - HabitantJpa
    - HabitantJpaRepository
  - entrypoints.rest
    - HabitantApi
    - HabitantEndpoint
  - service
    - HappyTownService
    - TrancheAgeComparator

- **Ajout d'un use case pour attribuer les cadeaux : AttribuerCadeaux**
- **Ajout d'un entrypoint rest pour attribuer les cadeaux : AttributionCadeauxEndpoint**
- **Suppression de HappyTownService**
- **Le comparateur des tranches d'âge TrancheAgeComparator rejoint le package des entities**
- **Suppression du package de service**

- com.happytown
  - configuration
    - ScheduleTasks
    - SwaggerConfig
  - core
    - entities
      - Cadeau
      - Habitant
      - TrancheAge
      - TrancheAgeComparator
    - use_cases
      - AttribuerCadeaux
      - GetAllHabitants
      - HabitantProvider
  - dataproviders.database
    - HabitantDatabaseProvider
    - HabitantJpa
    - HabitantJpaRepository
  - entrypoints.rest
    - AttributionCadeauxEndpoint
    - HabitantApi
    - HabitantEndpoint

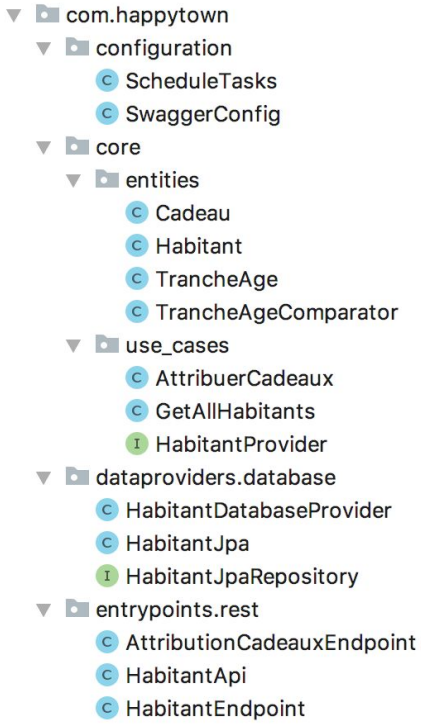
Clean archi : Job d'attribution de cadeaux

Step 05

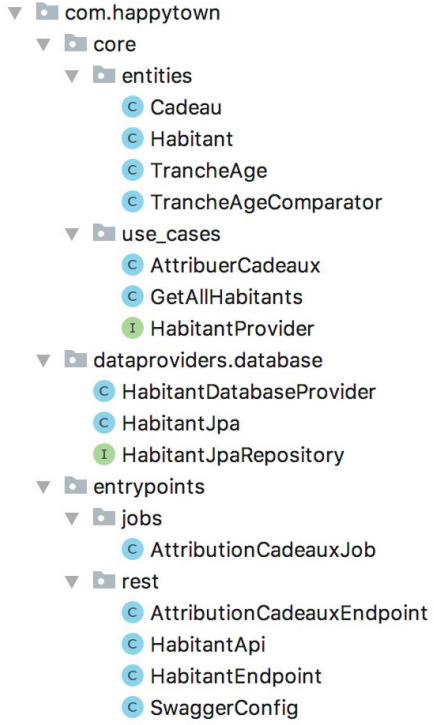


Clean archi : Attribution des cadeaux

Step 05

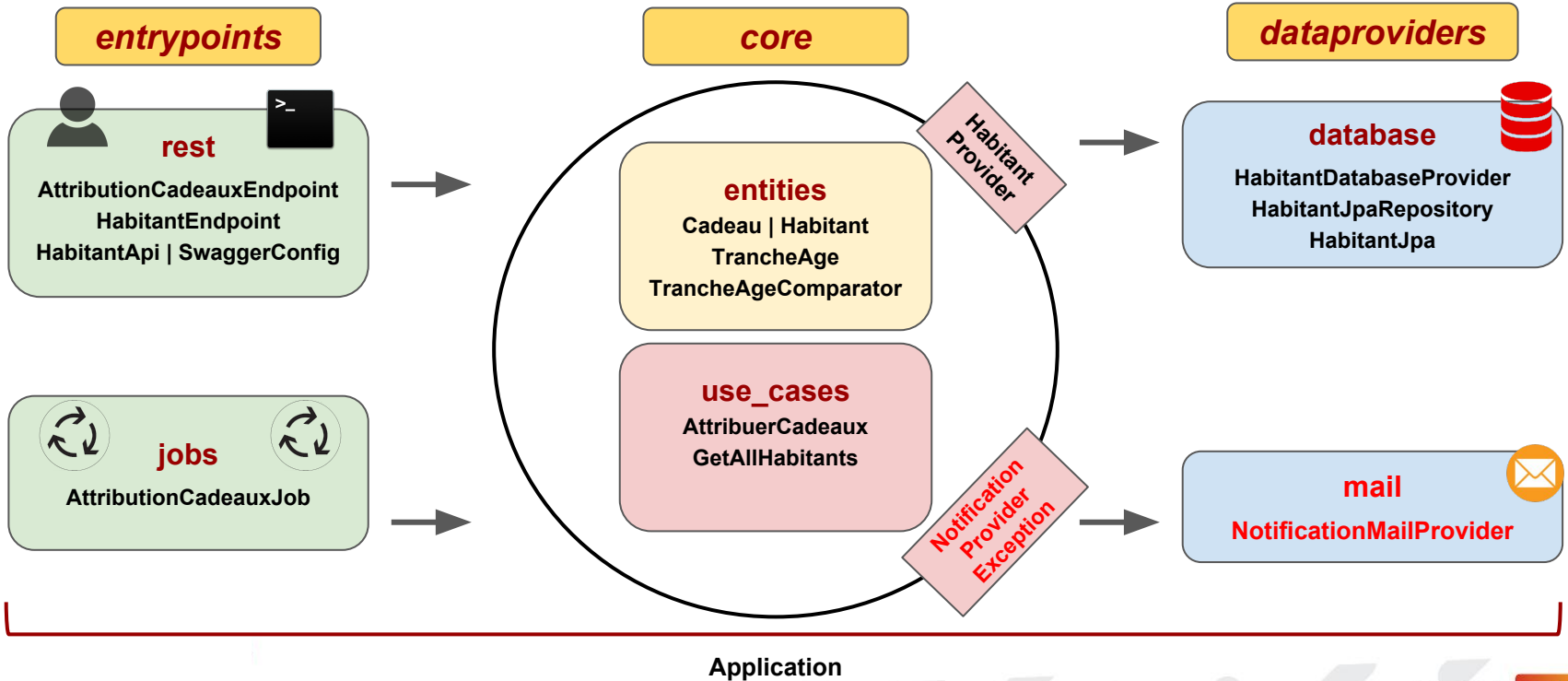


- **Ajout d'un entrypoint jobs pour la tâche automatique d'attribution de cadeaux : AttributionCadeauxJob**
- **Déplacement de la classe de configuration SwaggerConfig dans le package rest des entrypoints**
- **Suppression du package de configuration**
- **Mise en place d'un mécanisme de vérification du respect de la clean architecture (sens des dépendances) : tools/check-cleanArchi.sh**



Clean archi : Schéma en construction

Step 06



Clean archi : Notification par mail

Step 06

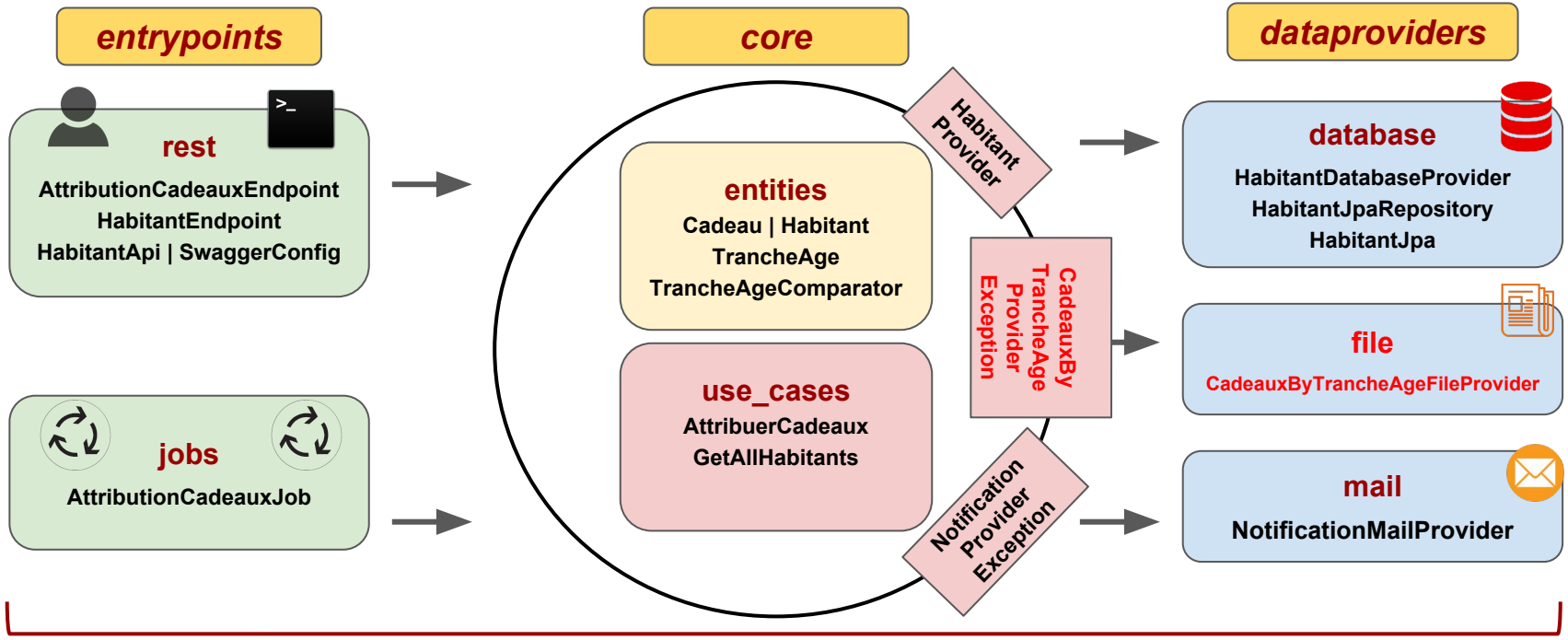
- com.happytown
  - core
    - entities
      - Cadeau
      - Habitant
      - TrancheAge
      - TrancheAgeComparator
    - use_cases
      - AttribuerCadeaux
      - GetAllHabitants
      - HabitantProvider
  - dataproviders.database
    - HabitantDatabaseProvider
    - HabitantJpa
    - HabitantJpaRepository
  - entrypoints
    - jobs
      - AttributionCadeauxJob
    - rest
      - AttributionCadeauxEndpoint
      - HabitantApi
      - HabitantEndpoint
      - SwaggerConfig

- ❑ **Sortie de la partie concernant les mails**
- ❑ **Nouveau Provider + Exception Métier : NotificationProvider + NotificationException**
- ❑ **Implémentation d'un mail provider : NotificationMailProvider avec les infos de config : smtpHost + smtpPort**
- ❑ **Suppression de la config du serveur de mail dans les use cases + entrypoints**
- ❑ **Tests unitaires du use case : AttribuerCadeaux (plus de lancement de serveur de mail) : rapidité, fiabilité**
- ❑ **Changement de la librairie de mail pour les tests unitaires : FakeSmtpRule**

- com.happytown
  - core
    - entities
      - Cadeau
      - Habitant
      - TrancheAge
      - TrancheAgeComparator
    - use_cases
      - AttribuerCadeaux
      - GetAllHabitants
      - HabitantProvider
      - NotificationException
      - NotificationProvider
  - dataproviders
    - database
      - HabitantDatabaseProvider
      - HabitantJpa
      - HabitantJpaRepository
    - mail
      - NotificationMailProvider
  - entrypoints
    - jobs
      - AttributionCadeauxJob
    - rest
      - AttributionCadeauxEndpoint
      - HabitantApi
      - HabitantEndpoint
      - SwaggerConfig

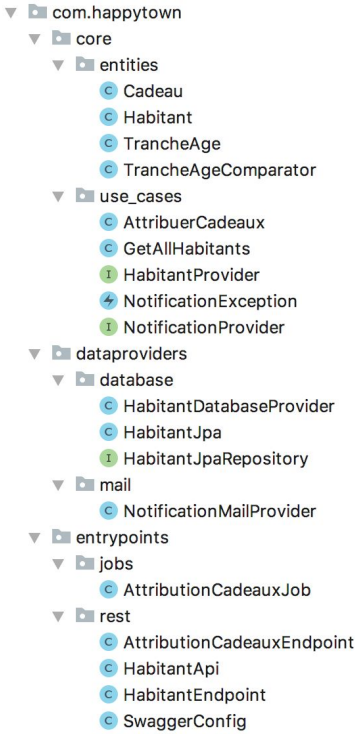
Clean archi : Schéma en construction

Step 07

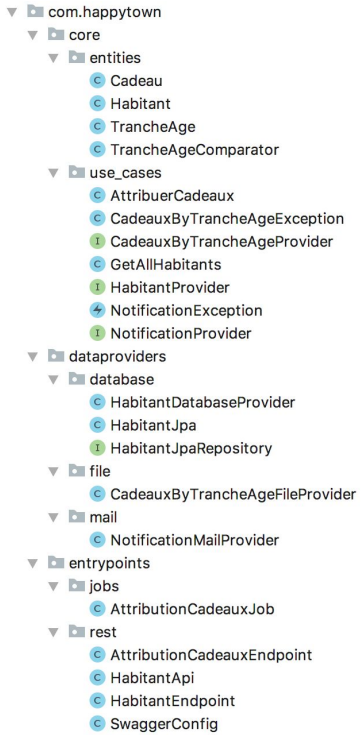


Clean archi : Récupération des cadeaux par tranche d'âge

Step 07

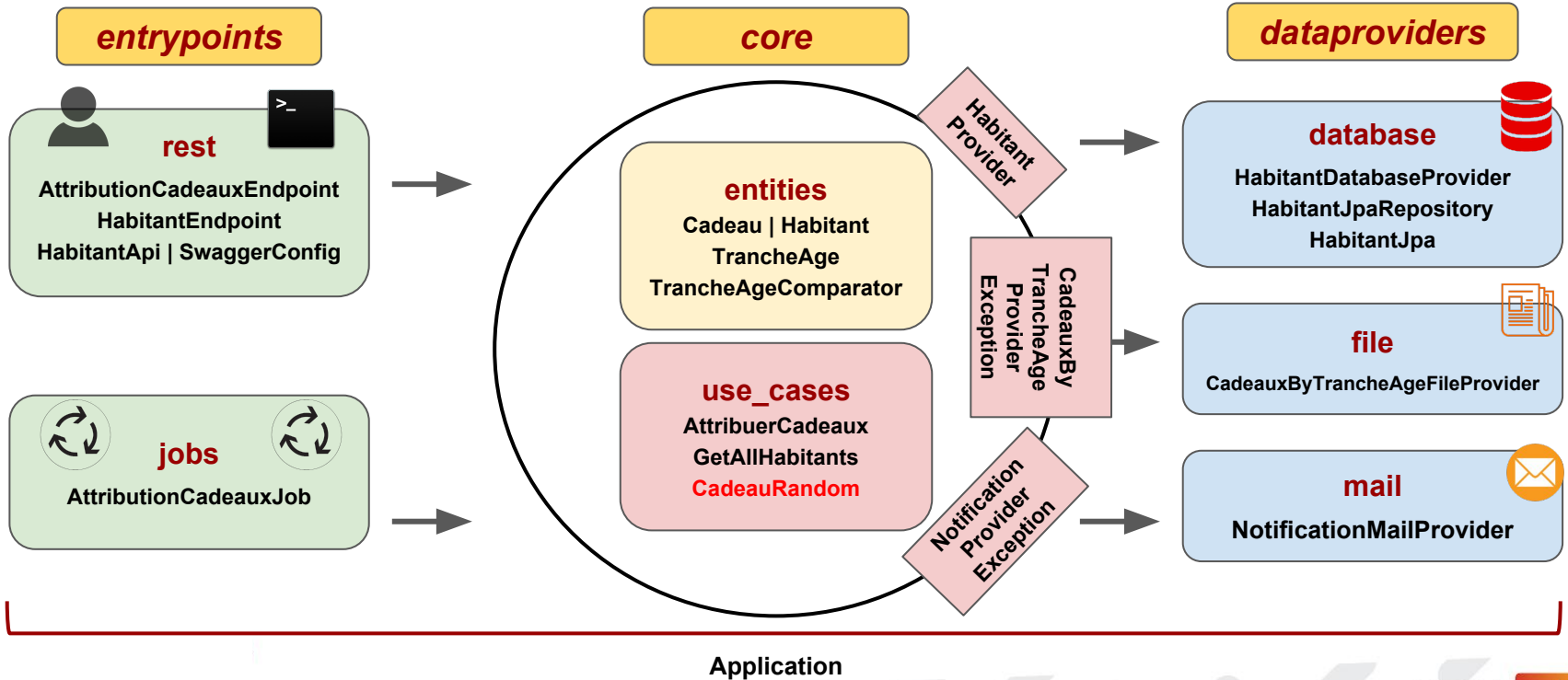


- **Sortie** de la partie concernant la récupération des cadeaux par tranche d'âge
- Nouveau Provider + Exception Métier : **CadeauxByTrancheAgeProvider** + **CadeauxByTrancheAgeException**
- Implémentation d'un file provider : **CadeauxByTrancheAgeFileProvider** avec les informations de config : **fileName**
- **Suppression** des infos de **config** du **fichier** dans les **use cases** + **entrypoints**
- **Suppression** des infos de **dateCourante** dans les **use cases** et les **entrypoints** avec **injection** d'une **Clock**



Clean archi : Schéma en construction

Step 08



Clean archi : Refactoring

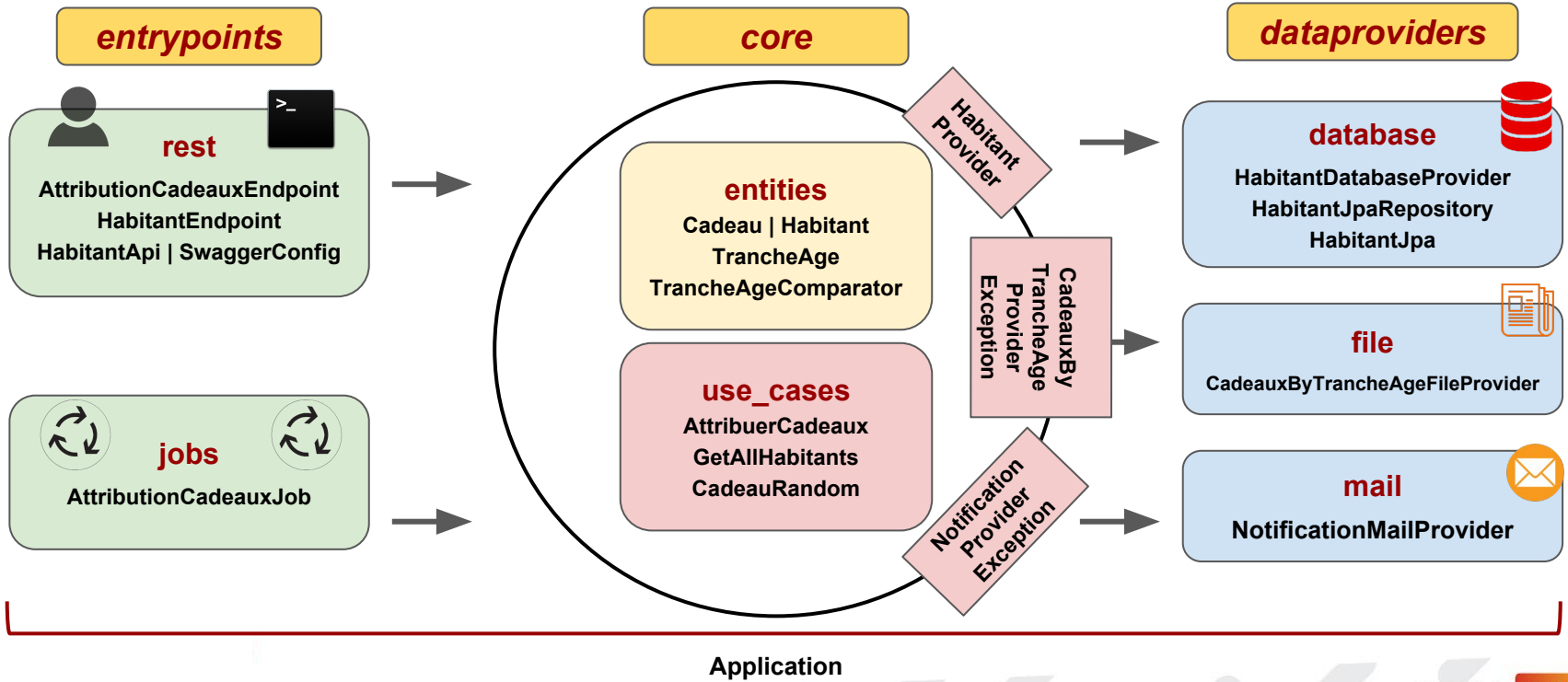
Step 08

- com.happytown
  - core
    - entities
      - Cadeau
      - Habitant
      - TrancheAge
      - TrancheAgeComparator
    - use_cases
      - AttribuerCadeaux
      - CadeauxByTrancheAgeException
      - CadeauxByTrancheAgeProvider
      - GetAllHabitants
      - HabitantProvider
      - NotificationException
      - NotificationProvider
  - dataproviders
    - database
      - HabitantDatabaseProvider
      - HabitantJpa
      - HabitantJpaRepository
    - file
      - CadeauxByTrancheAgeFileProvider
    - mail
      - NotificationMailProvider
  - entrypoints
    - jobs
      - AttributionCadeauxJob
    - rest
      - AttributionCadeauxEndpoint
      - HabitantApi
      - HabitantEndpoint
      - SwaggerConfig

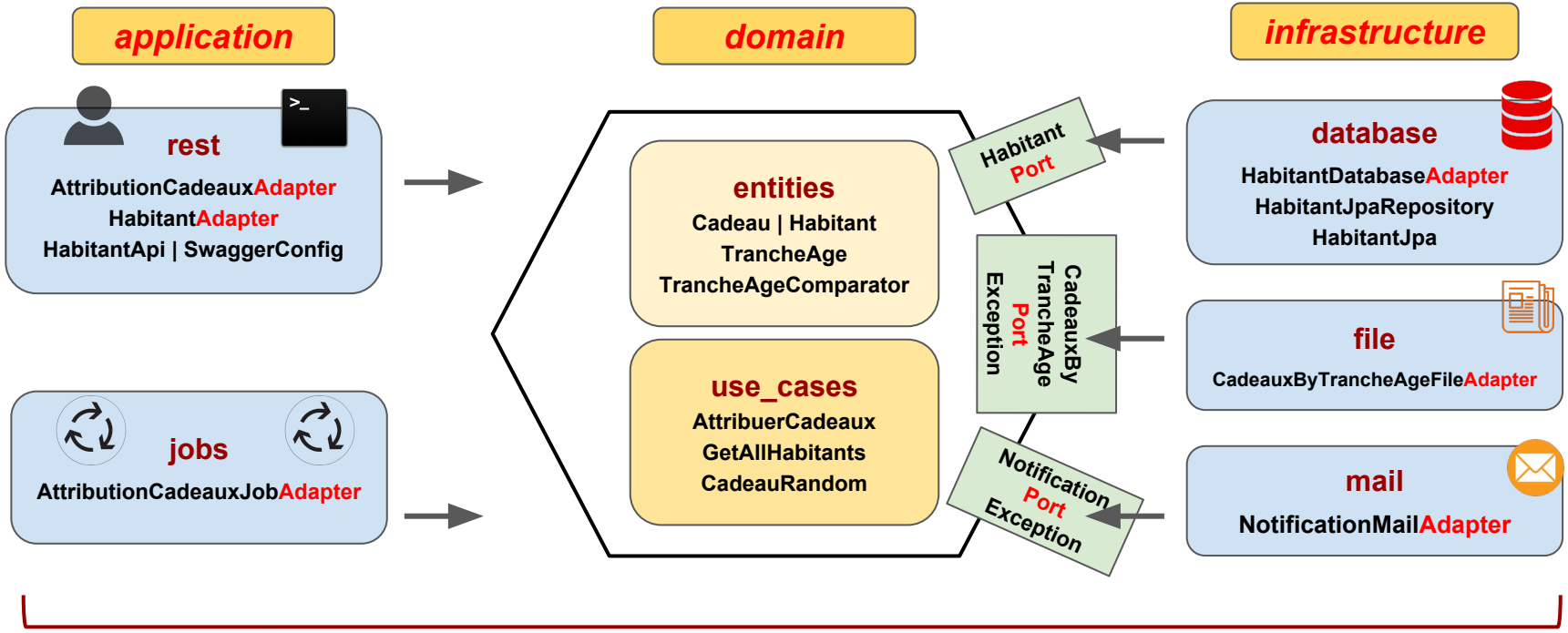
- Mise en place de **templates** de messages pour les **mails**
- **Ajout** d'un **use case** d'attribution de cadeau aléatoire : **CadeauRandom**
- Ajout de **fixtures** pour **simplifier** l'écriture des **tests**
- **Refactoring** divers

- com.happytown
  - core
    - entities
      - Cadeau
      - Habitant
      - TrancheAge
      - TrancheAgeComparator
    - use_cases
      - AttribuerCadeaux
      - CadeauRandom
      - CadeauxByTrancheAgeException
      - CadeauxByTrancheAgeProvider
      - GetAllHabitants
      - HabitantProvider
      - NotificationException
      - NotificationProvider
  - dataproviders
    - database
      - HabitantDatabaseProvider
      - HabitantJpa
      - HabitantJpaRepository
    - file
      - CadeauxByTrancheAgeFileProvider
    - mail
      - NotificationMailProvider
  - entrypoints
    - jobs
      - AttributionCadeauxJob
    - rest
      - AttributionCadeauxEndpoint
      - HabitantApi
      - HabitantEndpoint
      - SwaggerConfig

Clean archi : Schéma final



Hexagonale archi : Schéma final



Debrief et conclusions

- ✔ **Isolation** et **protection** du **métier**
- ✔ **Totale indépendance** vis à vis des **frameworks** avec un **métier clair** et **explicite**
- ✔ **Meilleure testabilité** orientée sur le **comportement métier**
- ✔ Une **pyramide** de **tests saine** pour les **évolutions** et la **maintenance**
- ✔ **Séparation** claire des **problèmes**

Discuter en équipe de votre choix d'architecture : clean architecture, hexagonale architecture ou un mix des deux ou une autre…

L'important est de redonner sa place au métier et de sortir des architectures à découpage technique

MERCI ! :)

Références

- https://github.com/celinegilet/happy-town
- https://blog.octo.com/architecture-hexagonale-trois-principes-et-un-exemple-dimplementation/
- https://github.com/damienbeaufils/clean-architecture-demo
- https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html
- https://github.com/mattia-battiston/clean-architecture-example