Eleven Labs

ARCHITECTURE HEXAGONALE

Pourquoi l'envisager pour votre application ?

FICHE D'INFORMATION

L'infrastructure d'une application, les bases de données, les appels API, de même que les frameworks évoluent constamment.

Mais ce n'est pas forcément le cas du besoin métier.

En partant de ce principe, quelle solution est envisageable afin de rendre une application maintenable dans le temps, d'avoir un besoin métier complètement agnostique des choix techniques de son implémentation ?

Pour visualiser l'architecture hexagonale, on la représente souvent par un hexagone avec en son coeur le Domain. La couche du dessus correspond à l'application et le tout est entouré de l'Infrastructure.

L'architecture hexagonale peut être la solution car elle tend à isoler une application en trois couches distinctes.

3 couches distinctes

Domain
Application
Infrastructure

Domain

Cette couche contient les entités existantes dans le domaine. Dans cette couche on ne se soucie pas de la manière d'exposer les données à l'utilisateur ou de la façon dont elles seront stockées. Les entrées/sorties se font à l'aide d'interfaces uniquement et respectent le D du principe SOLID avec l'inversion de dépendance. Pour cela il faut garder en tête que le métier ne doit dépendre de rien et donc ne jamais être lié ni à la couche Application, ni à la couche Infrastructure.

Application

C'est dans cette couche que l'Infrastructure va communiquer avec le Domain par le biais des interfaces. On considère ici que la couche Application connaît le Domain mais ne doit jamais être liée à l'Infrastructure. C'est là que pour des raisons métiers on peut enrichir, comparer, filtrer des éléments renvoyés par les interfaces du Domain sans se soucier de l'implémentation côté Infrastructure. Il est fréquent de parler dans cette couche de Handler mais également de Use Case métier. Les Use Cases métiers peuvent utiliser le pattern CQRS pour séparer plus finement les besoins métiers d'écriture de ceux de lecture.

Infrastructure

L'architecture hexagonale considère que le Domain se trouve au centre de l'application. De fait, la couche d'Infrastructure va pouvoir communiquer et invoquer les objets du Domain par le biais des interfaces. Attention, l'infrastructure connaît la couche Application mais non l'inverse ! De cette façon, si l'on souhaite changer d'API tierce, faire de la double écriture ou changer de modèle de données, il suffit de recréer un adapter dédié dans la couche Infrastructure en respectant l'interface du Domain.

Avantages et inconvénients

Avantages

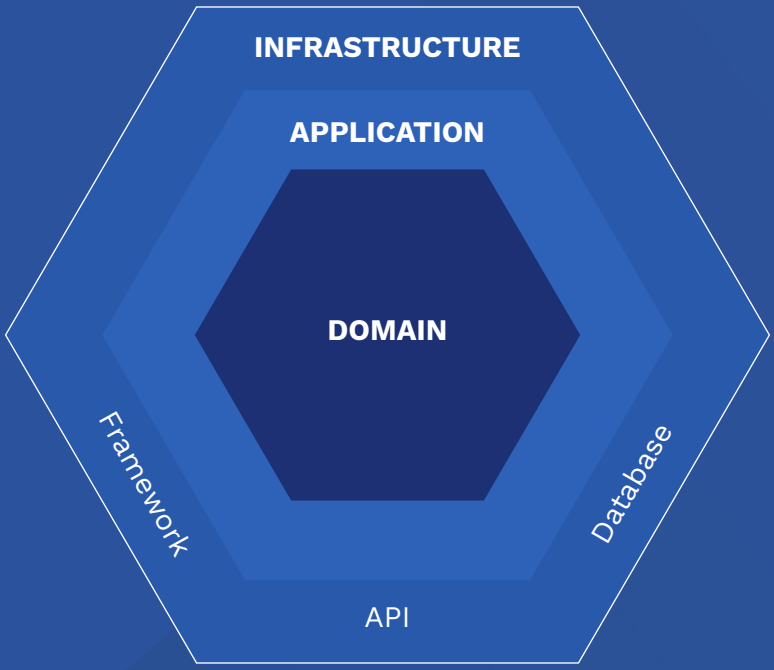
- Rend le besoin métier agnostique de son implémentation
- Technique
- Maintenable
- Évolutif
- Testable sans se soucier de l'Infrastructure
- Respecte le principe d'inversion de dépendance

Inconvénients

- Non adapté à une petite application web

> **Dans une application basée sur l'architecture hexagonale, il est moins pénible de changer de base de données.**

> **Lorsque par exemple on envisage de changer d'une base Oracle vers Cassandra ou lorsque que l'on souhaite faire une montée de version de Symfony, le coeur de métier étant le même dans la couche Domain, on ne se préoccupe que de l'Infrastructure.**


INFRASTRUCTURE
APPLICATION
DOMAIN
Framework
Database
API

Bien que l'on comprenne très rapidement son utilité, l'architecture hexagonale peut prêter à confusion au premier abord et **elle sera également déconseillée pour les besoins de petits projets.**

Cependant, en travaillant dans une architecture hexagonale, il est plus aisé d'envisager la vie long-termiste d'une application, de la maintenir, de la tester et de réduire le couplage avec ses sources de données et avec l'exposition à l'utilisateur. Les changements techniques sont réguliers, les problématiques de montée en charge et les choix de migrations sont envisagés de manière beaucoup plus sereine et paraissent moins pénibles.

Attention cela ne veut pas dire que le métier reste figé, au contraire, c'est une manière de capitaliser sur celui-ci et de l'améliorer.

L'architecture hexagonale peut fonctionner avec le DDD et avec le CQRS, bien que pour ce dernier il puisse être sujet à quelques adaptations.

MAIS AU FAIT

QUI SOMMES-NOUS ?

Eleven Labs est une société de conseil, spécialisée en création et réalisation de projets Web agiles de qualité.

Les experts interviennent sur des missions qu'ils choisissent, et elles sont axées sur du développement, de l'architecture, de la conduite de projet Agile, de l'audit et du conseil.

Au quotidien, tout est fait pour encourager la progression et l'épanouissement technique, à travers des plateformes dédiées à l'apprentissage (le blog, le codelabs, des accès à Udemy et egghead.io...), ou grâce à des événements internes (workshops, meetups, formations...) réguliers.

Nous travaillons pour des grandes marques comme :
Canal+ - Deezer - Société Générale - France TV - Bouygues Télécom - Oui Sncf - Allianz - Lagardère - Alstom - Leroy Merlin

NOS TECHNOLOGIES

Eleven Labs

15 Avenue de la Grande-Armée
75016 Paris

contact@eleven-labs.com

01 82 83 11 75