Clean architecture

With Symfony

DG ECFIN R3 - Gilles Demaret
European Commission
February 2023

Introduction

Introduction

- Présentation de l’architecture sur laquelle se repose 3 de nos projets :

Introduction

- Présentation de l'architecture sur laquelle se repose 3 de nos projets :
  - CSR (Country Specific Recommendations)

Introduction

- Présentation de l'architecture sur laquelle se repose 3 de nos projets :
  - CSR (Country Specific Recommendations)
  - RRF (Recovery and Resilience Facility)

Introduction

- Présentation de l'architecture sur laquelle se repose 3 de nos projets :
  - CSR (Country Specific Recommendations)
  - RRF (Recovery and Resilience Facility)
  - RAFT (Submodule of RRF)

Pourquoi la clean architecture ?

- Pourquoi la clean architecture ? Pourquoi une architecture tout court ?

Pourquoi la clean architecture ?

- Pourquoi la clean architecture ? Pourquoi une architecture tout court ?
  - Pour palier à des problèmes récurrents sur des projets informatiques:

Pourquoi la clean architecture ?

- Pourquoi la clean architecture ? Pourquoi une architecture tout court ?
  - Pour palier à des problèmes récurrents sur des projets informatiques:
    - Code spaghetti

Pourquoi la clean architecture ?

- Pourquoi la clean architecture ? Pourquoi une architecture tout court ?
  - Pour palier à des problèmes récurrents sur des projets informatiques:
    - Code spaghetti
    - Code business éparpillé entre les controllers, les services, les managers et même les views

Pourquoi la clean architecture ?

- Pourquoi la clean architecture ? Pourquoi une architecture tout court ?
  - Pour palier à des problèmes récurrents sur des projets informatiques:
    - Code spaghetti
    - Code business éparpillé entre les controllers, les services, les managers et même les views
    - Des services ou manager qui ont trop de responsabilités

Pourquoi la clean architecture ?

- Pourquoi la clean architecture ? Pourquoi une architecture tout court ?
  - Pour palier à des problèmes récurrents sur des projets informatiques:
    - Code spaghetti
    - Code business éparpillé entre les controllers, les services, les managers et même les views
    - Des services ou manager qui ont trop de responsabilités
    - ...

Pourquoi la clean architecture ?

- Pourquoi la clean architecture ? Pourquoi une architecture tout court ?
  - Pour palier à des problèmes récurrents sur des projets informatiques:
    - Code spaghetti
    - Code business éparpillé entre les controllers, les services, les managers et même les views
    - Des services ou manager qui ont trop de responsabilités
    - ...
  - Pour faciliter l'évolution de nos projets, leur maintenance

Pourquoi la clean architecture ?

- Pourquoi la clean architecture ? Pourquoi une architecture tout court ?
  - Pour palier à des problèmes récurrents sur des projets informatiques:
    - Code spaghetti
    - Code business éparpillé entre les controllers, les services, les managers et même les views
    - Des services ou manager qui ont trop de responsabilités
    - ...
  - Pour faciliter l'évolution de nos projets, leur maintenance
  - Pour rendre notre métier plus intéressant et plus enrichissant

Qui ? Quand ?

- Qui ?

Qui ? Quand ?

- Qui ?
  - Présenté par Robert C. Martin (Uncle Bob) en 2012

Qui ? Quand ?

- Qui ?
  - Présenté par Robert C. Martin (Uncle Bob) en 2012
- Combine plusieurs variantes architecturales

Qui ? Quand ?

- Qui ?
  - Présenté par Robert C. Martin (Uncle Bob) en 2012
- Combine plusieurs variantes architecturales
  - Hexagonal architecture

Qui ? Quand ?

- Qui ?
  - Présenté par Robert C. Martin (Uncle Bob) en 2012
- Combine plusieurs variantes architecturales
  - Hexagonal architecture
  - Onion architecture

Qui ? Quand ?

- Qui ?
  - Présenté par Robert C. Martin (Uncle Bob) en 2012
- Combine plusieurs variantes architecturales
  - Hexagonal architecture
  - Onion architecture
  - Screaming architecture

Qui ? Quand ?

- Qui ?
  - Présenté par Robert C. Martin (Uncle Bob) en 2012
- Combine plusieurs variantes architecturales
  - Hexagonal architecture
  - Onion architecture
  - Screaming architecture
  - MVP (Model View Presenter)

Qui ? Quand ?

- Qui ?
  - Présenté par Robert C. Martin (Uncle Bob) en 2012
- Combine plusieurs variantes architecturales
  - Hexagonal architecture
  - Onion architecture
  - Screaming architecture
  - MVP (Model View Presenter)
  - Use Case driven approach

Qui ? Quand ?

- Qui ?
  - Présenté par Robert C. Martin (Uncle Bob) en 2012
- Combine plusieurs variantes architecturales
  - Hexagonal architecture
  - Onion architecture
  - Screaming architecture
  - MVP (Model View Presenter)
  - Use Case driven approach
  - SOLID

Objectif

Objectif

- L’objectif est:

Objectif

- L’objectif est:
  - De diviser le code en briques, chaque brique traitant d’un sujet.

Objectif

- L’objectif est:
  - De diviser le code en briques, chaque brique traitant d’un sujet.
  - Les briques sont elles-même divisées en couches.

Objectif

- L’objectif est:
  - De diviser le code en briques, chaque brique traitant d’un sujet.
  - Les briques sont elles-même divisées en couches.
  - Chaque brique comporte au moins une couche d’accès au business et une couche d’interfaçage permettant la connexion inter-briques.

Avantages

Avantages

- Avantages

Avantages

- Avantages
  - L’indépendance vis-à-vis des frameworks : les frameworks sont alors utilisés comme des outils plutôt que l’inverse

Avantages

- Avantages
  - L'indépendance vis-à-vis des frameworks : les frameworks sont alors utilisés comme des outils plutôt que l'inverse
  - L'indépendance vis-à-vis de l'interface utilisateur

Avantages

- Avantages
  - L'indépendance vis-à-vis des frameworks : les frameworks sont alors utilisés comme des outils plutôt que l'inverse
  - L'indépendance vis-à-vis de l'interface utilisateur
  - L'indépendance vis-à-vis de la base de données

Avantages

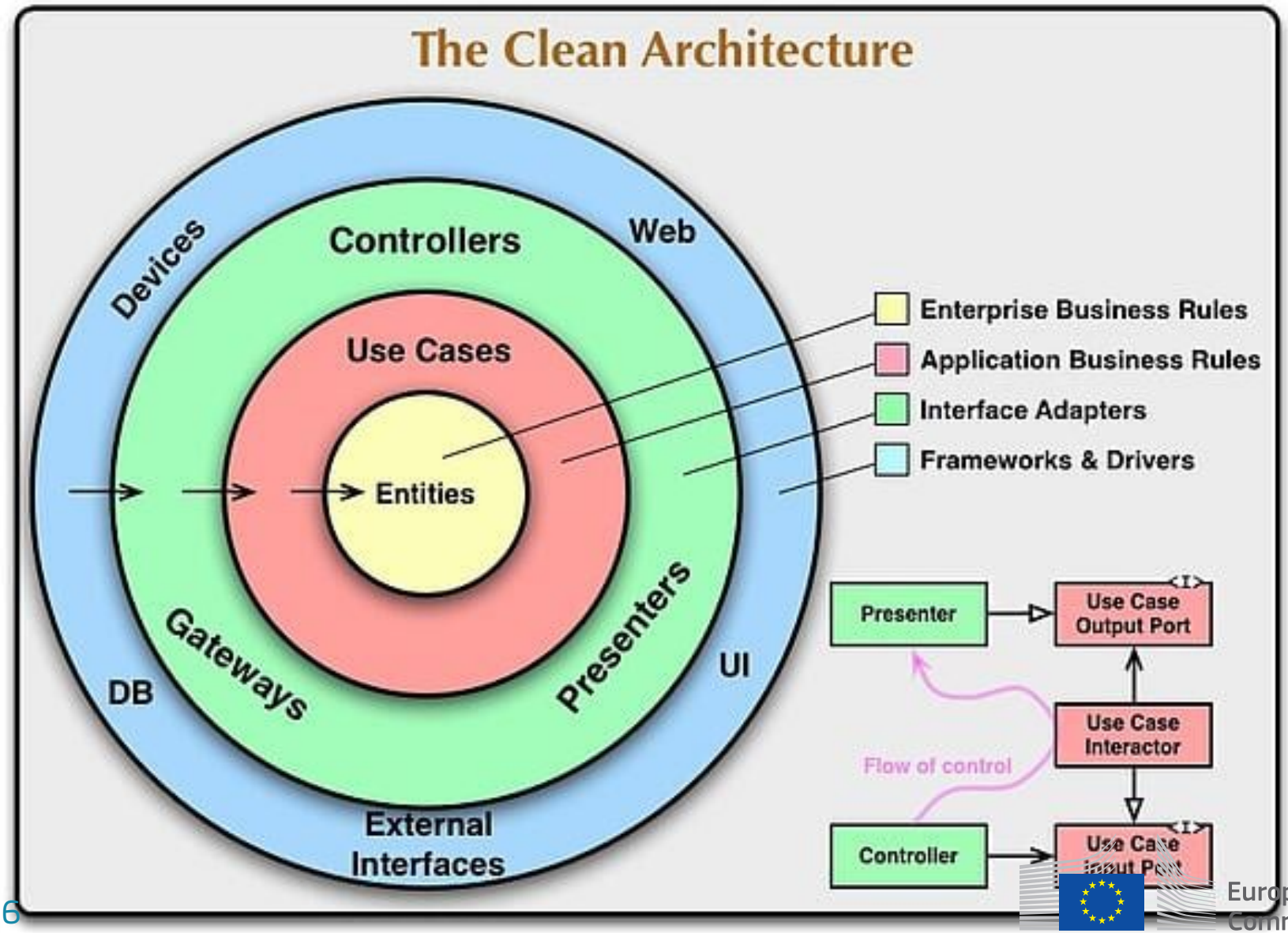
- Avantages
  - L'indépendance vis-à-vis des frameworks : les frameworks sont alors utilisés comme des outils plutôt que l'inverse
  - L'indépendance vis-à-vis de l'interface utilisateur
  - L'indépendance vis-à-vis de la base de données
  - L'indépendance vis-à-vis des services tiers

Avantages

- Avantages
  - L'indépendance vis-à-vis des frameworks : les frameworks sont alors utilisés comme des outils plutôt que l'inverse
  - L'indépendance vis-à-vis de l'interface utilisateur
  - L'indépendance vis-à-vis de la base de données
  - L'indépendance vis-à-vis des services tiers
  - La granularité des tests : il est facile de cibler précisément une couche / brique

Principe

Schéma d'uncle Bob



Le domaine

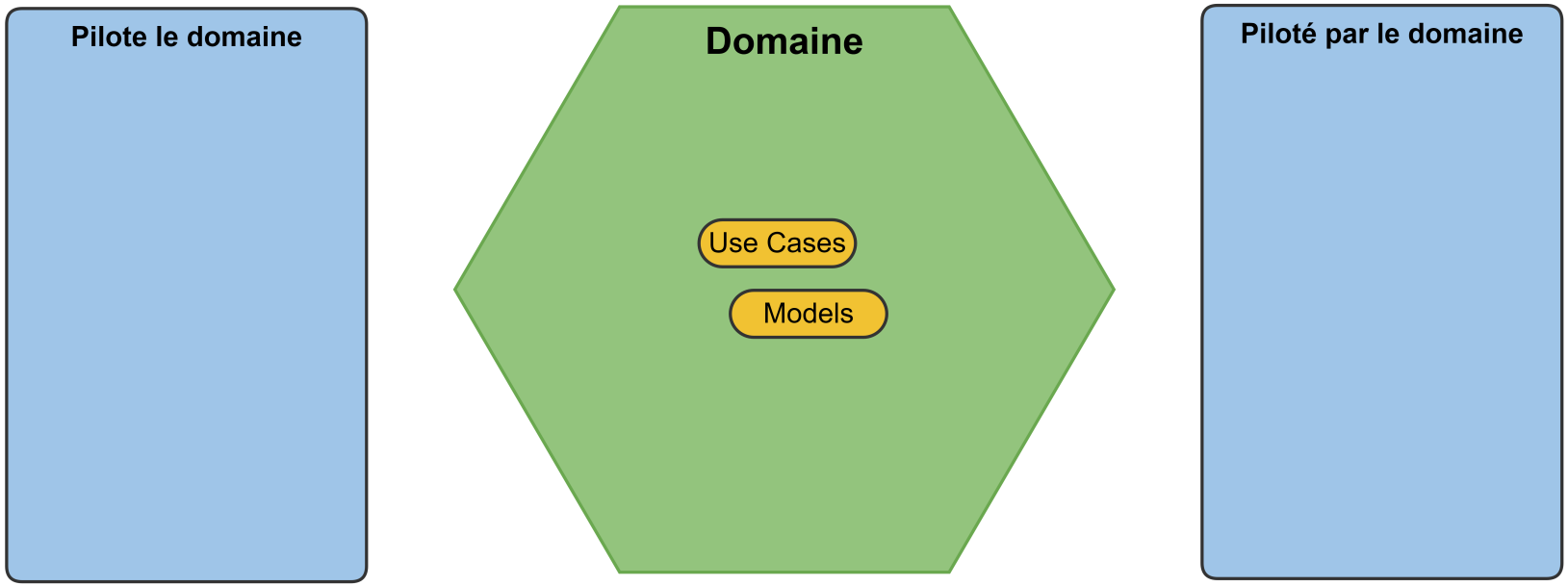


Figure 2: Clean architecture 02

Le domaine piloté et pilote

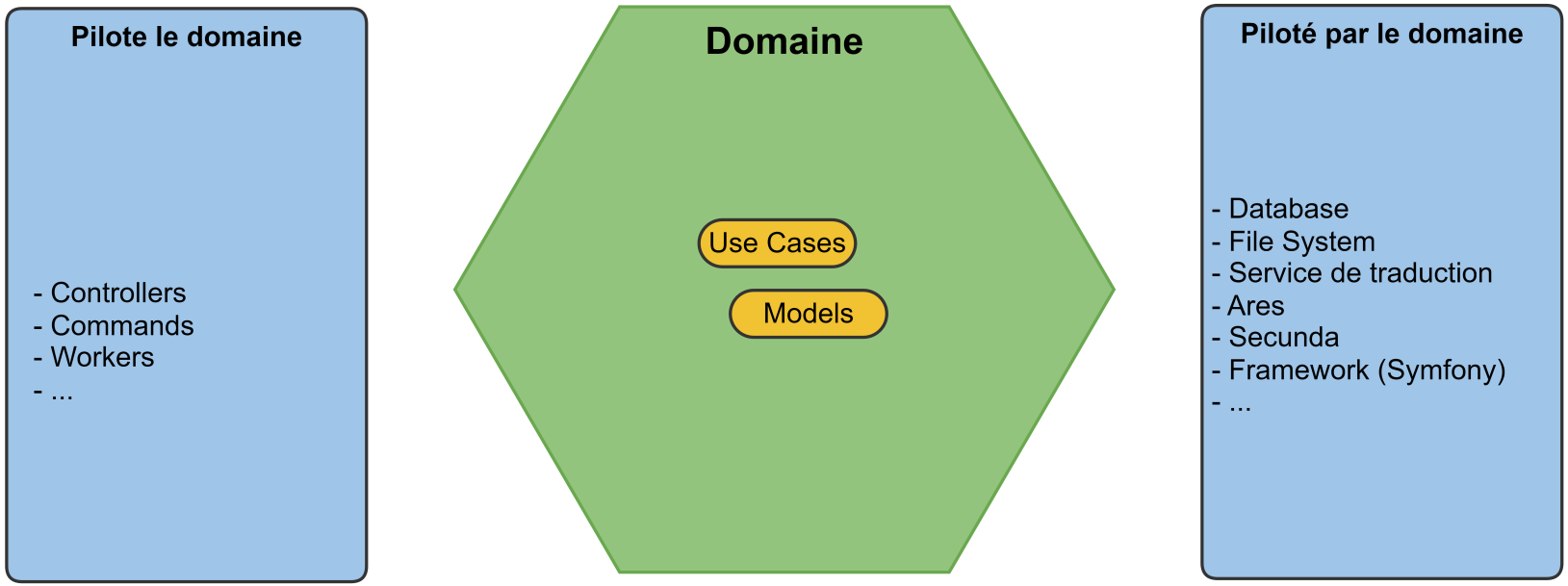


Figure 3: Clean architecture 03

Le domaine et le monde extérieur

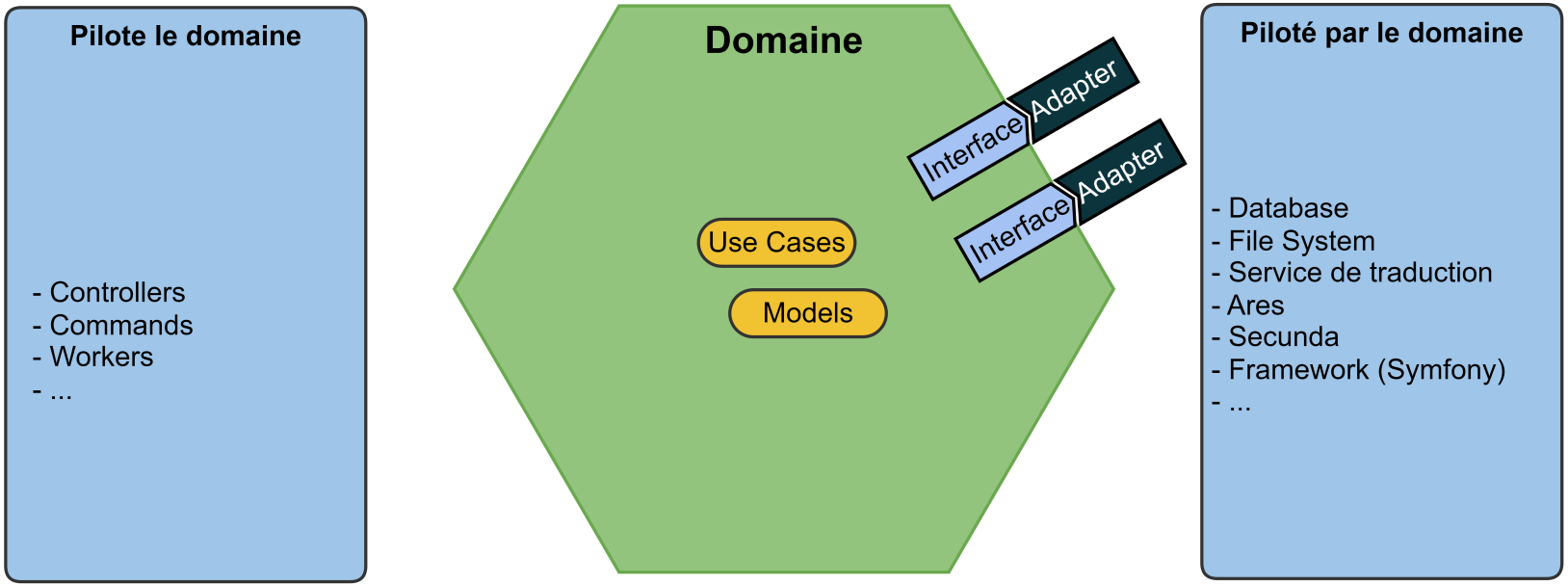


Figure 4: Clean architecture 04

Vue globale

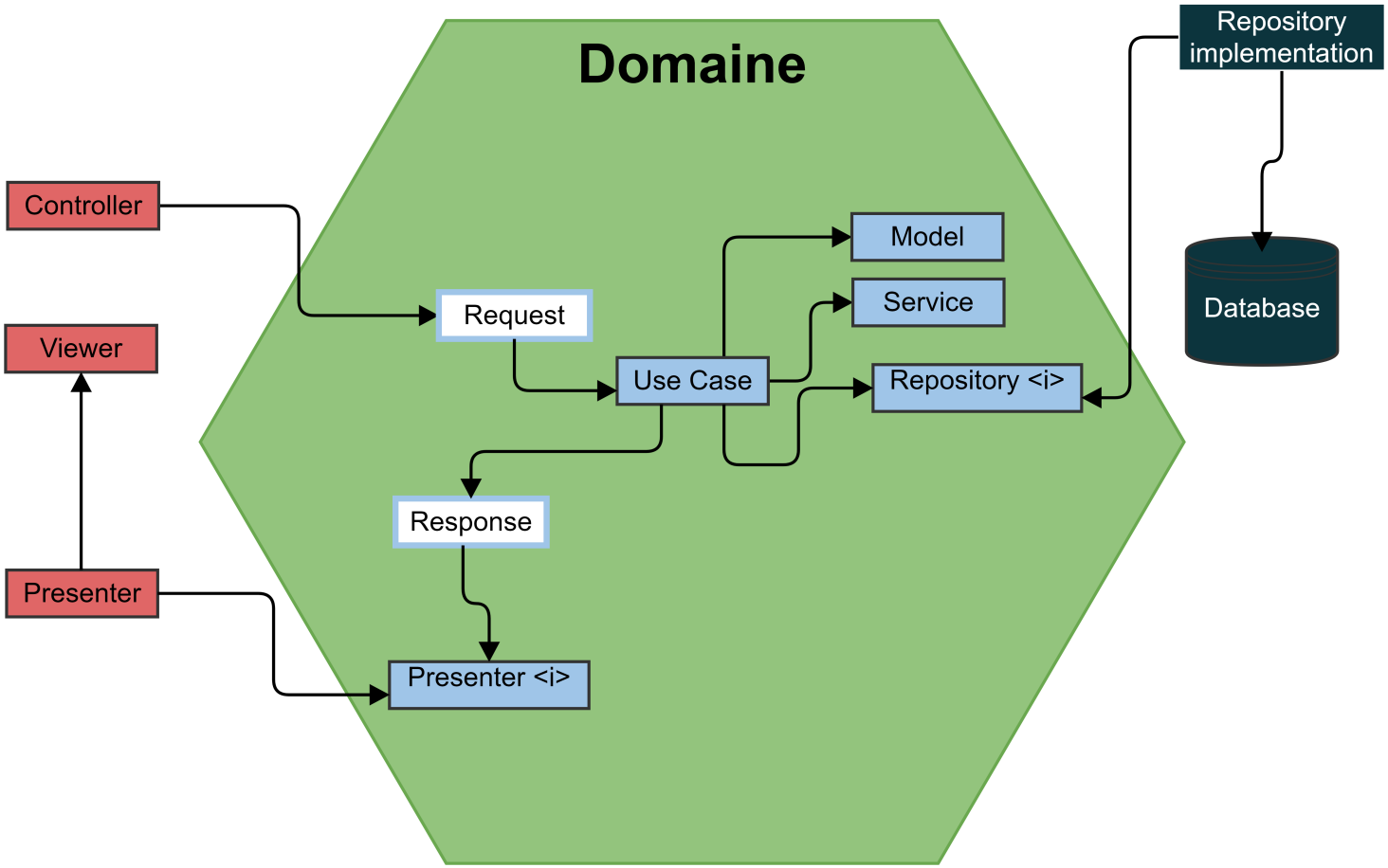


Figure 5: Clean architecture 05

Exemple

La request

```php
class SwitchPhaseToExecutionRequest implements RequestInterface
{
    public Audit $audit;
}
```

Le Use Case

```php
class SwitchPhaseToExecution extends AbstractUseCaseItem
{
    public function __construct(
        private SwitchPhaseToExecutionValidator $validator,
        private AuditPhaseRepositoryInterface $auditPhaseRepository
    ) {
    }

    private function execute(
        SwitchPhaseToExecutionRequest $request
        PresenterInterface $presenter
    ):void {
        ...
        $this->response->model = $request->audit;
        $this->presenter->present($this->response);
    }
}
```

La response

```php
class ApiItemResponse implements ResponseItemInterface
{
    /**
     * @var object
     */
    public $model;

    /**
     * @var array
     */
    public $errors = [];
}
```

Piloté par une commande

```php
class SwitchAuditPhaseToExecutionCommand extends Command
{
    public function __construct(
        private AuditRepository $auditRepository,
        private SwitchPhaseToExecution $useCase,
        private SwitchPhaseToExecutionCommandPresenter $presenter,
        private CommandViewer $viewer,
    ) {
        parent::__construct();
    }

    protected function execute(InputInterface $input, OutputInterface $output): int
    {
        $audits = $this->auditRepository->findAllOnPreparationPhase();

        ...

        foreach ($audits as $audit) {
            $useCaseRequest = new SwitchPhaseToExecutionRequest();
            $useCaseRequest->audit = $audit;

            try {
                $this->useCase->execute($useCaseRequest, $this->presenter);

                return $this->viewer->generateView($this->presenter->viewModel(), Command::SUCCESS);
            } catch (Throwable $exception) {
                return $this->viewer->generateView($this->presenter->viewModel(), Command::FAILURE);
            }
        }
    }
}
```

Conclusion

Conclusion

- Avantages ?

Conclusion

- Avantages ?
- Coût ? Inconvénients ?

Sources

Sources

- Clean Architecture: A Craftsman's Guide to Software Structure and Design: A Craftsman's Guide to Software Structure and Design (Robert C. Martin Series)
- https://youtu.be/LTxJFQ6xmzM (Présentation par Nicolas De Boose)
- https://www.adimeo.com/blog/forum-php-2019-clean-architecture
- https://www.adimeo.com/blog/forum-php-2019-developpement-pragmatique

Thank you