

L'Event Sourcing vu par un jeune Développeur

C'est dur...



La victime de cette expérience 🙌



Valmont PEHAUT-PIETRI



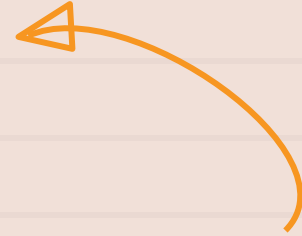
Valmonzo



@valmontpp



Introduction



Pourquoi ce sujet ?

Ou comment tout a commencé

Le contexte





01

Le désespoir



Mes connaissances

- Les bases de Symfony

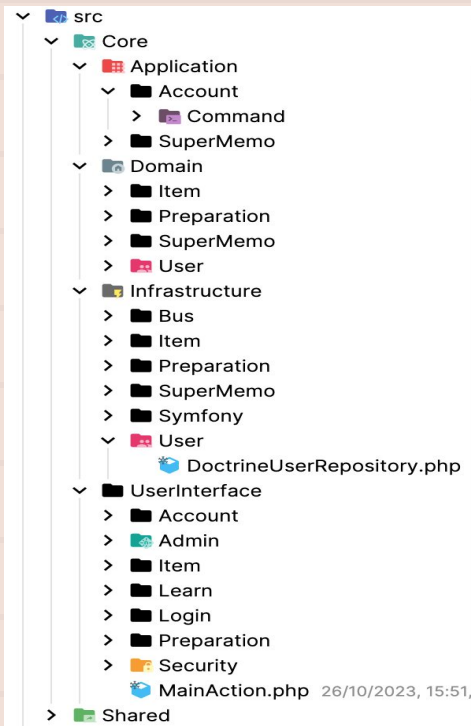
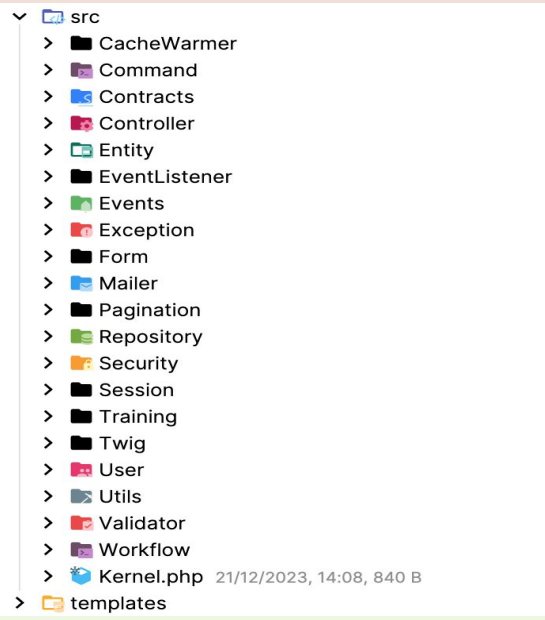
- 8 mois de PHP

- Quelques principes de base



Les premiers obstacles

VS



Les premiers obstacles

Domain Driven Design

Layering Architecture

CQRS



Mes premières impressions



- Le projet est illisible
- C'est compliqué pour rien
- Ça me donne l'impression d'être nul *



02



Dans le doute, essayons



La recette miracle de l'apprentissage



Matériel

1. Ordinateur
2. Internet
3. Cerveau
4. Motivation



Temps

Pour la vie

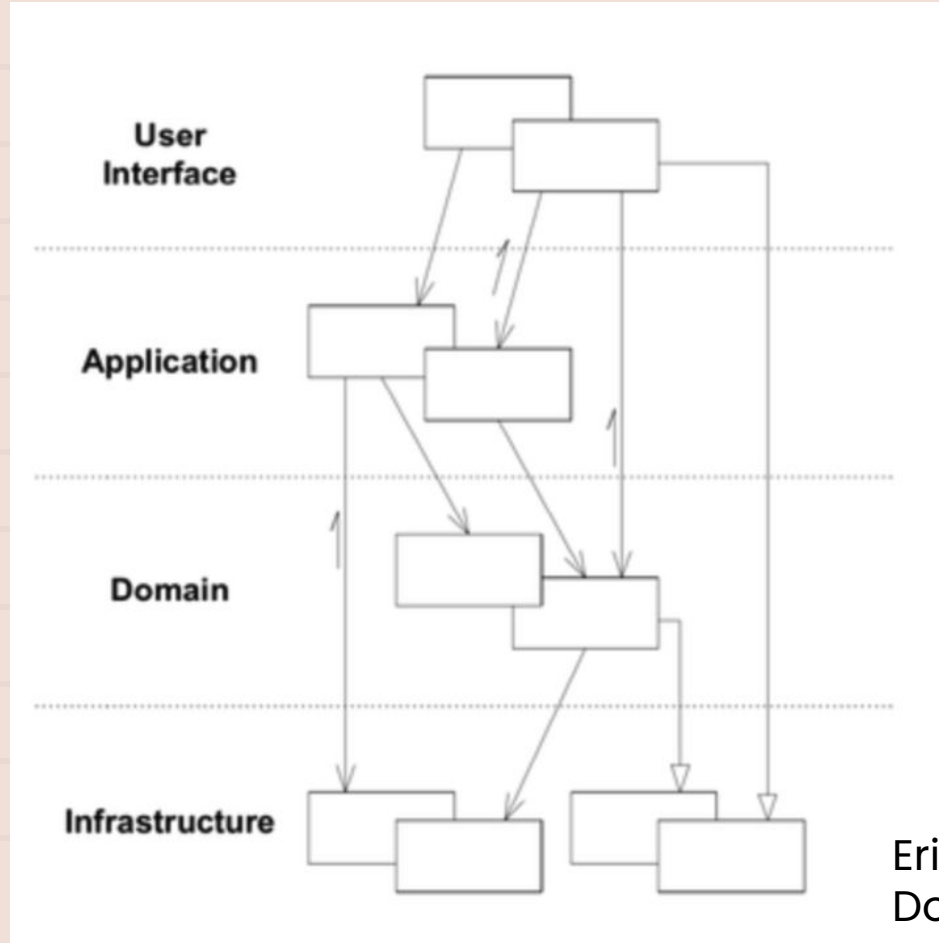
No	Étape	Description
1.	Théorie	DDD ? CQRS ? Event Sourcing ? Qu'est-ce que ça veut dire ?
2.	Comprehension	Regarder des conférences, de la doc etc..
3.	Application	Analyse du besoin, feature
4.	Analyse	Ma logique, mes choix
5.	Évaluation	Utilité, montée en compétence
6.	Optimisation	Optimisation de la feature, ajout etc...

En théorie

Domain Driven Design : Je veux que mon code reflète un maximum le métier

Layering Architecture : Plusieurs couches logiques, chaque couche ayant une responsabilité spécifique.





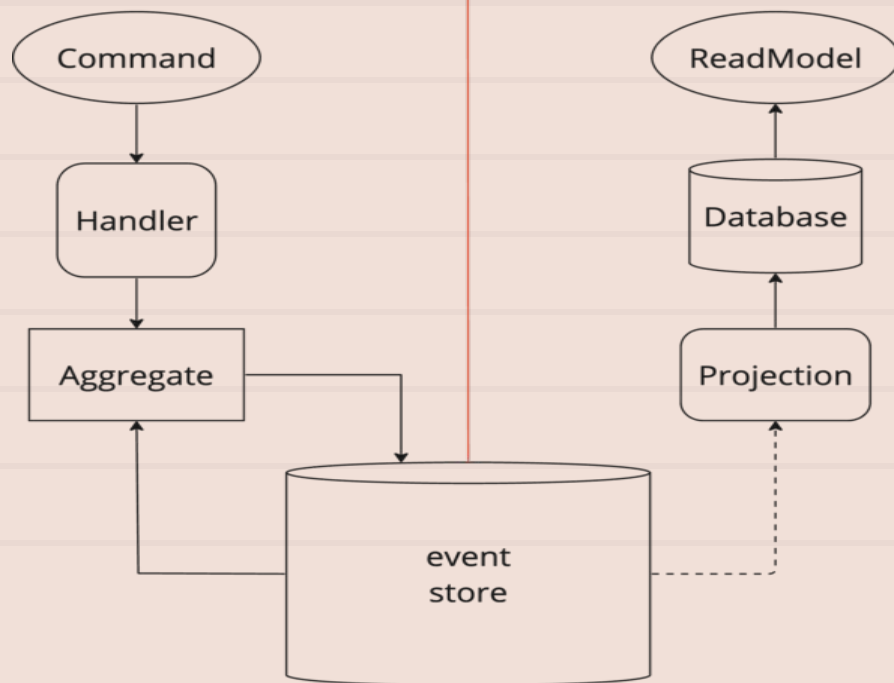
Eric Evans –
Domain Driven Design


En théorie

Command Query Responsibility Segregation (CQRS) : On sépare les opérations de lecture et d'écriture.

Event Sourcing : Je capture l'état d'un système sous forme d'événements, je reconstruit le modèle de lecture en rejouant ces événements *

Focus sur l'évent sourcing



No	Étape
1.	Théorie 
2.	Comprehension
3.	Application
4.	Analyse



Compréhension

BreizhCamp



Mes ch
08:35:38

API Platform providers

GET /books/cheapest



```
new GetCollection('/books/cheapest')
```

Operation

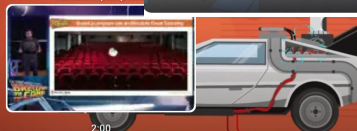
API PLATFORM
CONFERENCE



24:45 / 41:11

API PLA
CONF

⚙️ Faites glisser vers le haut
recherche plus préc



2:00

0:01 / 51:48

#DevovxFR



0:05 / 14:49

DDD ≠ RAD

“Domain Driven Design is **not** a fast way to build software.
However, it is able to ease your life when you have to deal with
complex business expectations.”

William Durand - DDD with Symfony 2: Making things clear

@matarId @challas_r



FORUM PHP
PARIS 2021

2:35 / 38:56



Compréhension

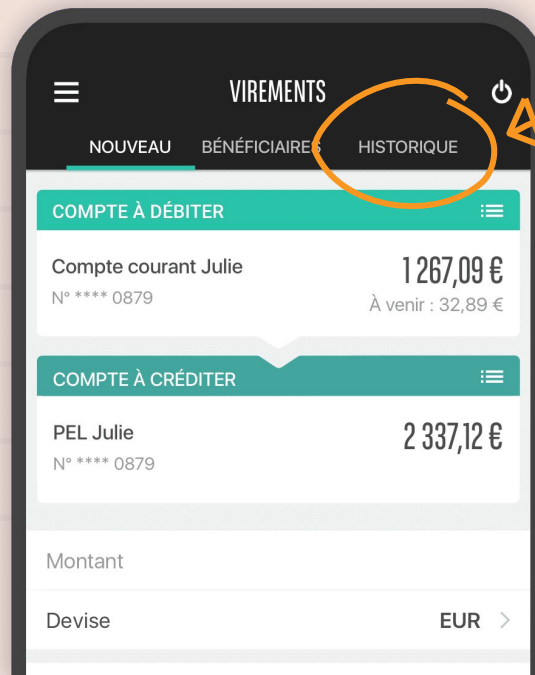
✓ Comprendre le vocabulaire en y mettant des **mots simples**

✓ Le mettre en place dans **la vie de tous les jours** (oui la vie est rempli d'événements !) permet de comprendre son utilité

✓ Lire des articles, regarder des conférences





Compréhension par l'exemple



Elle a déjà fait son virement



No	Étape
1.	Théorie 
2.	Comprehension 
3.	Application
4.	Analyse



Application

Les besoins : On veut enregistrer les événements liés à un User

La stack : Symfony, Doctrine, Behat



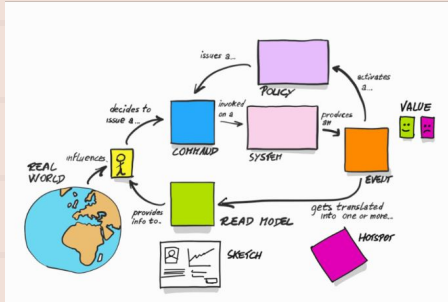
Application

“Définir les besoins”



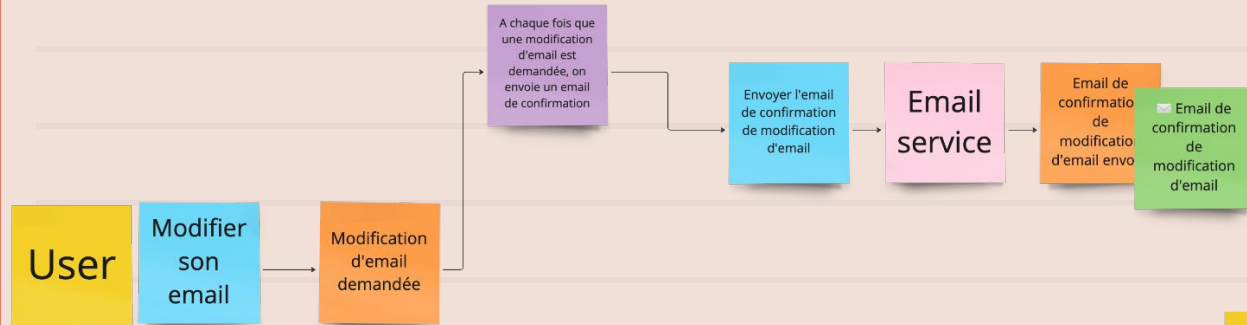
Application

"L'event storming"



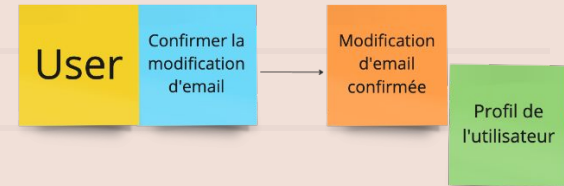
Application

"Focus sur une feature"



Feature:

As a User logged, I want to change my email.



Application

“Les outils à disposition”

```
final public static function reconstruct(  
    object $rootId,  
    iterable $events,  
): static {  
    $aggregate = new static($rootId);  
  
    foreach ($events as $event) {  
        $aggregate->apply($event);  
    }  
  
    return $aggregate;  
}
```



Application

“Les outils à disposition”

```
final protected function record(object $event): void
{
    $this->apply($event);

    $this->pendingEvents[] = [
        'time' => new DatePoint(),
        'aggregate_root_id' => $this->rootId,
        'aggregate_name' => $this->name,
        'aggregate_version' => $this->version,
        'event' => $event,
    ];
}
```



Application

“Les outils à disposition”

```
final public function apply(object $event): void
{
    ++$this->version;
    $this->applyEvent($event);
}

abstract private function applyEvent(object $event): void;
```



Application

“Les outils à disposition”

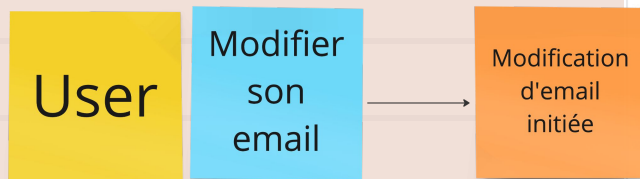
```
final public function releaseEvents(): array
{
    $releasedEvents = $this->pendingEvents;
    $this->pendingEvents = [];

    return $releasedEvents;
}
```



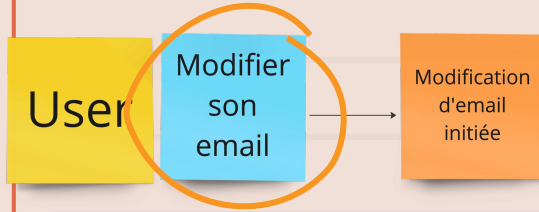
Application

"Les tests"



```
@domain
Scenario: Initiate Change USER email (authenticated user: Jérémy)
  Given the "Jérémy" USER has been registered:
    | firstname | Jérémy |
    | lastname  | FreeAgent |
    | email     | jeremy@example.org |
    | registrationDate | yesterday |
  When one initiate email change of the "Jérémy" USER:
    | email | jeremyfreeagent@example.org |
    | InitiateEmailChangeAt | now |
  Then success
  And the "Jérémy" USER email change should be initiated:
    | email | jeremyfreeagent@example.org |
```

Application "Domain"



```
final readonly class InitiateChangeEmail
{
    public function __construct(
        public UserId $id,
        public string $newEmail,
        public DateTime $initiateChangeEmailAt,
    ) {
    }
}
```



Application "Domain"



```
final class Account implements AggregateInterface
{
    use AggregateTrait;

    public function initiateChangeEmail(string $newEmail,
    DateTime $initiateEmailChangeAt): void
    {
        $this->record(new ChangeEmailInitiated($this->rootId(),
        $newEmail, $initiateEmailChangeAt));
    }
}
```

Application “Domain”

```
final readonly class ChangeEmailInitiated
{
    public function __construct(
        public UserId $id,
        public string $newEmail,
        public DateTime $changeEmailInitiatedAt,
    ) {
    }
}
```

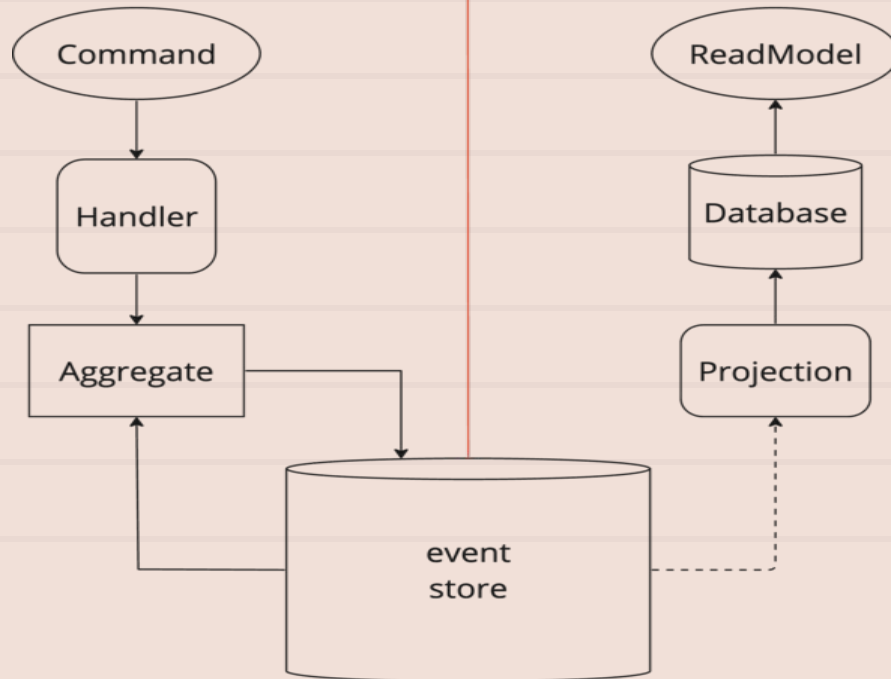

Application “Domain”

```
public function save(AggregateInterface $aggregate): void
{
    $releasedEvents = $aggregate->releaseEvents();

    $this->eventStore->store($releasedEvents);

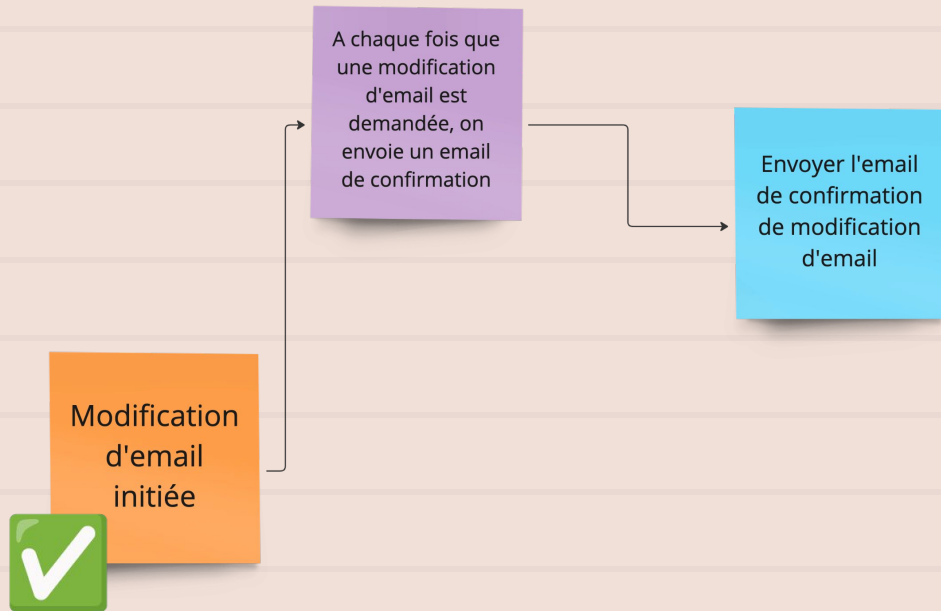
    $this->eventBus->dispatch($releasedEvents);
}
```

La partie écriture



Application




“Ça s’est produit”



Application “Donc je réagis”

A chaque fois que
une modification
d'email est
demandée, on
envoie un email
de confirmation

```
final readonly class WheneverChangeEmailInitiated
{
    public function __invoke(ChangeEmailInitiated $changeEmailInitiated) :
void
    {
        $this->asyncCommandBus->dispatch(
            new SendChangeEmailConfirmation(
                $changeEmailInitiated->id,
                $changeEmailInitiated->newEmail,
                $changeEmailInitiated->changeEmailInitiatedAt
            ));
    }
}
```

No	Étape
1.	Théorie 
2.	Comprehension 
3.	Application 
4.	Analyse



Analyse



L'intérêt de l'event sourcing



Il détient l'histoire et n'est pas modifiable



De la donnée pour vos KPIs, audits, logs



L'intérêt de l'event sourcing



Un projet scalable à souhait



Mais tout aussi facile à maintenir



03

**On s'est quand même bien
amusé**



**ANNONCER À MON FRÈRE QUE
JE NE SUIS PLUS DANS SON CAMP**



L'éternel débat

Est-ce adapté pour un.e débutant.e ?

Ne devrais-je pas consolider mes bases avant ?

Pourquoi on ne le voit pas en formation*?



3

Merci !

Vos feedbacks pleins
d'amour en scannant
ce code



CREDITS: This presentation template
was created by [Slidesgo](#), and
includes icons by [Flaticon](#), and
infographics & images by [Freepik](#)

Please keep this slide for attribution

