DDD & Event Sourcing à la rescousse pour implémenter la GDPR

Speakers : Jérôme Avoustin (Saagie) et Guillaume Lours (Saagie)
Format : Conférence
Date : 19 avril 2018

Présentation de la GDPR

GDPR ou RGPD en français : Règlement Général sur la Protection des Données
Texte de référence concernant toutes les entreprises de l'UE et les entreprises mondiales collectant des données sur les citoyens européens.

Mise en œuvre pour quand ? initialement il y'a 2 ans, désormais pour le 25 mai 2018.
D'après Gartner, à peine 50% des entreprises EU seront en conformité fin 2018. Amendes jusqu'à 4% du CA de l'entreprise mais plafonné à 20 millions €.

Quelles sont les données concernées par la réglementation : toutes les données permettant d'identifier une personne physiquement : numéro de sécurité sociale, numéro d'identité, adresse IP ou même un groupement d'informations permettant d'identifier une personne.

Impacts :
- Notification de 72h pour prévenir la CNIL si faille détectée
- Nouveaux processus
- Architecture IT à mettre en conformité
- Nouveau poste : Data Protection Officer (DPO). Il se doit de tout documenter
- Nouveaux processus

Les problématiques :
- Restriction : mises-en place de droit d'accès.
- Audit : identifier qui a accès à la donnée, à quel moment (traçabilité)
- Registre des traitements : liste de tous les traitements effectués sur la donnée
- Consentement nécessaire pour pouvoir exploiter la donnée. La personne doit pouvoir révoquer son droit simplement
- Anonymisation nécessaire si besoin de statistiques pour garantir la vie privée
- Sécurité : garantir la sécurité des données privées. Il faut montrer qu'on a mis les moyens pour garantir la sécurité des données.
- Privacy by design : ne collecter que les données nécessaires au traitement métier. Ex de l'adresse postale pour une newsletter : interdit.

Anonymisation :
- Irréversible
- Utilisation d'algorithmes qu'il va falloir coupler entre eux

Pseudonymisation :
- Réversible
- Utilisé pour déléguer l'analyse des données à un tiers.

Etudes de cas

Application Saagie Data Governance

- Application BigData avec beaucoup de données
- Nombreux clients
- Cet outil se doit d'être conforme à la GDPR
- Beaucoup de DataSet

Etat des lieux :

- Application développée en 2 mois pour le salon Big Data 2017
- Spring Boot + Angular
- CRUD
- Un méga objet qui gère tout
- Découpage en couche
- Code basé sur le modèle de données de la base de données

Découpée en couches : UI -> API -> Services -> Data Access -> Database
Utilisation de CRUD pour accéder à la base

En route vers le DDD

Que faire ? Utiliser le **Domain Driven Design** (DDD)

Objectifs :

- faire collaborer les sachant fonctionnels
- utiliser le modèle métier et pas celui de la base

Travailler sur le Ubiquitous Language

- Travailler autour d'un langage commun partagé par tous
- Tous les acteurs doivent avoir la même définition (Article, Produit)

Plusieurs personnes : personnes de la compta, du métier, des achats, de la logistique
Ils parlent tous de Produits.
Le modèle unique est un vrai problème.
On se retrouve avec des God classes.

Le modèle qu'on utilise dépend du contexte dans lequel on se situe. Le Produit va avoir une représentation différente si on discute avec les achats ou le marketing.
La même réalité peut avoir une projection différente. C'est ce qu'on appelle les **Bounded Context** en DDD.

Dans les contextes où le métier a beaucoup d'importance, le CRUD n'est pas approprié.
En DDD, le modèle est au centre et ne dépend de rien d'autre.

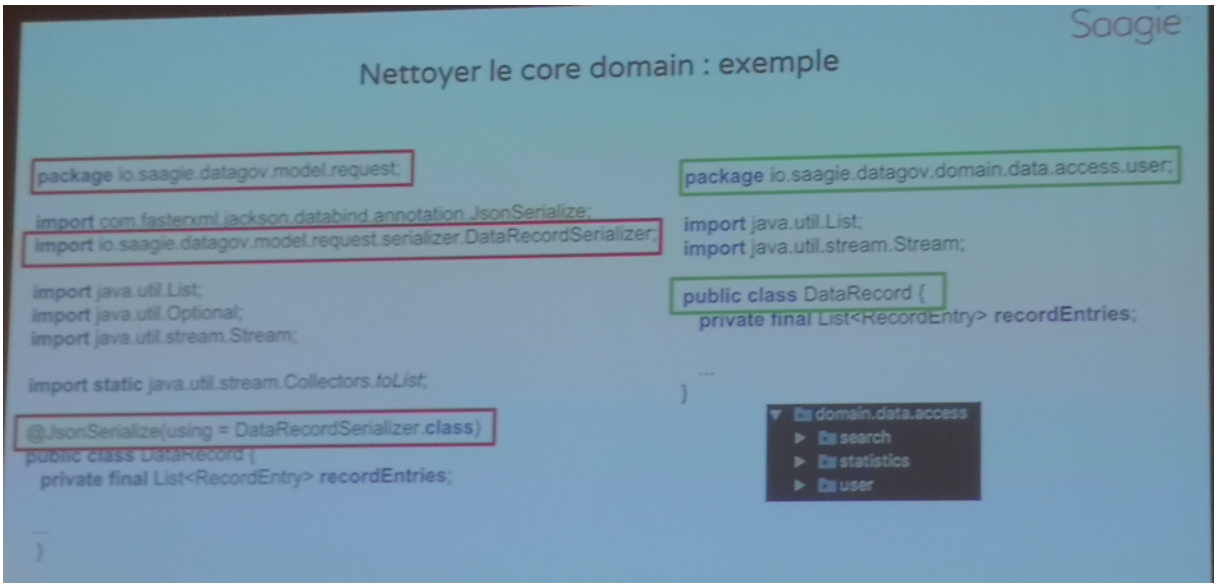
Dans le DDD, on se focus sur les évènements. Le domaine est piloté et chorégraphié par les évènements. Ces évènements vont avoir une représentation dans le code.

Ces évènements permettent de faire du contrôle de cohérence et faire de la traçabilité.
Les évènements prennent la forme de Value Objects (immutables).
Exemples : OrderShipped, ProductPurchased

Amène à une architecture de type Event Driven : Chrography over Orchestration

Outil utilisé pour explorer le domaine métier : l'Event Storming. Utilisé 2 fois : pour refactorer l'application puis comprendre la GDPR. Identification des impacts sur le fonctionnement actuel.

Nettoyer le core domain : Hexagonal Architecture
On supprime les données de sérialisation JSON et on renomme le package.
Les annotations sont renvoyées au code d'Infra. Même chose pour les @Service, @Autowired. Objectif : les classes du Domaine ne doivent avoir aucune adhérence avec un framework tiers.
Utilisation du pattern DataQuery qui n'est pas un DAO => plus de dépendance au framework. Utilisation d'un adapteur pour faire de l'inversion de dépendance.



Nettoyer le core domain : sans tout casser

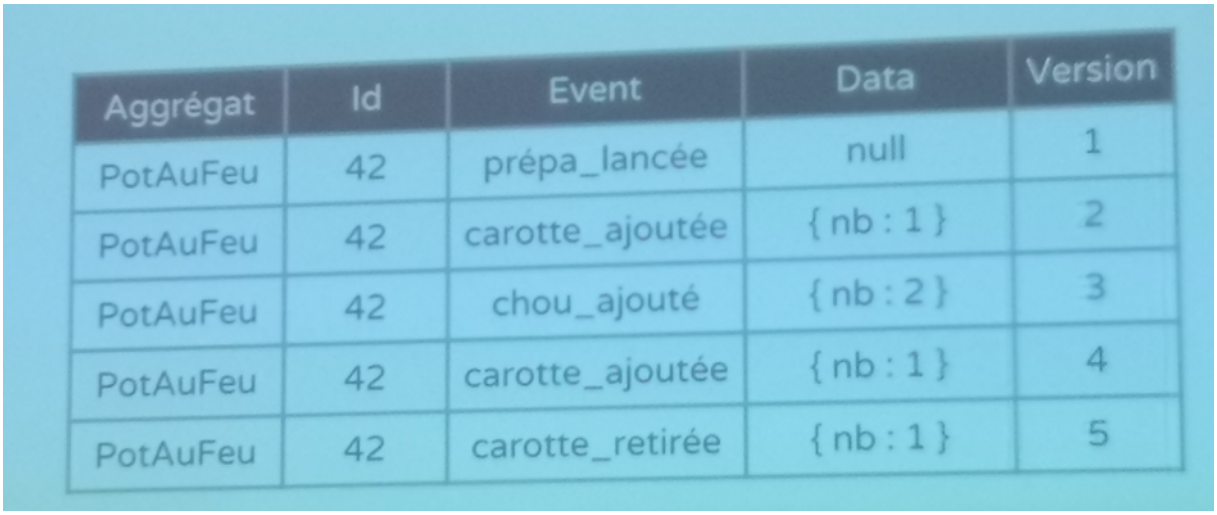
```
@Configuration
public class WebConfiguration extends WebMvcConfigurerAdapter {

  @Bean
  public MappingJackson2HttpMessageConverter mappingJackson2HttpMessageConverter() {
    ObjectMapper mapper = new ObjectMapper();
    mapper.registerModule(new JavaTimeModule());
    mapper.registerModule(new VavrModule());
    mapper.registerModule(dataRecordModule());
    ...
    return new MappingJackson2HttpMessageConverter(mapper);
  }
  ...
  private Module dataRecordModule() {
    SimpleModule module = new SimpleModule();
    module.addSerializer(DataRecord.class, new DataRecordSerializer());
    return module;
  }
}
```

```
package io.saagie.datagov.infra.adapter;

...

@Repository
public class DataQueryRepository implements DataQuery {

  @Autowired
  private SaagieBotDao saagieBotDao;

  @Autowired
  private QueryDao queryDao;

  @Override
  public QueryResult sampleQuery(String query) {
    return this.queryDao.testQuery(query);
  }

  @Override
  public QueryResult executeQuery(String query) {
    return this.queryDao.executeQuery(query);
  }

  @Override
  public CompactDataset executeQueryAsCompactDataset(String query) {
    return this.queryDao.executeQueryToCompactDataset(query);
  }
...
}
```

Prêt pour l'Event Sourcing ?

Event Sourcing :

- Persistance de l'historique des évènements
- Les évènements deviennent la source de vérité, pour reconstituer l'état
    - Initialement, les évènements étaient poussés dans un bus. Un handler les poussent dans une base Mongo.
    - Permet de connaître l'ordre et l'heure des évènements
- Source de donnée immutable (append only)



Aggrégat	Id	Event	Data	Version
PotAuFeu	42	prépa_lancée	null	1
PotAuFeu	42	carotte_ajoutée	{ nb : 1 }	2
PotAuFeu	42	chou_ajouté	{ nb : 2 }	3
PotAuFeu	42	carotte_ajoutée	{ nb : 1 }	4
PotAuFeu	42	carotte_retirée	{ nb : 1 }	5

Apports :

- Intéressant en termes de testabilité : assertions plus simples à écrire
- Versioning des aggrégats
- Debugging et reproductibilité => snapshot de l'event store du testeur
- Rollback facilité : on applique un événement contraire
- Traçabilité gratuite
- Récomposition des états par projection

Peut-on tout event sourcer ?

- Pas si simple
- Les évènements sont gardés à vie
- Réflexion nécessaire lorsque les aggrégats changent
- Les données personnelles ne doivent pas être conservées dans l'Event Store ou bien doivent être chiffrées avec une clé unique par personne (possibilité de « perdre » la clé sur demande de l'utilisateur)