

Architecture monolithique 2.0, Microservices obsolètes ?

MODULITH



Présentation

NGOS Simon Pascal

FOUOMENE PEWO Daniel Rene

PLAN

- Domaine fonctionnel de l'application Extranet OneLogic-HR
- L'architecture monolithe
- L'architecture microservice
- L'architecture monolithe 2.0 (Modulith)
- Cas pratique SPRING MODULITH
- Conclusion
- Références

Domaine fonctionnel de l'application Extranet-OneLogic HR

CRA

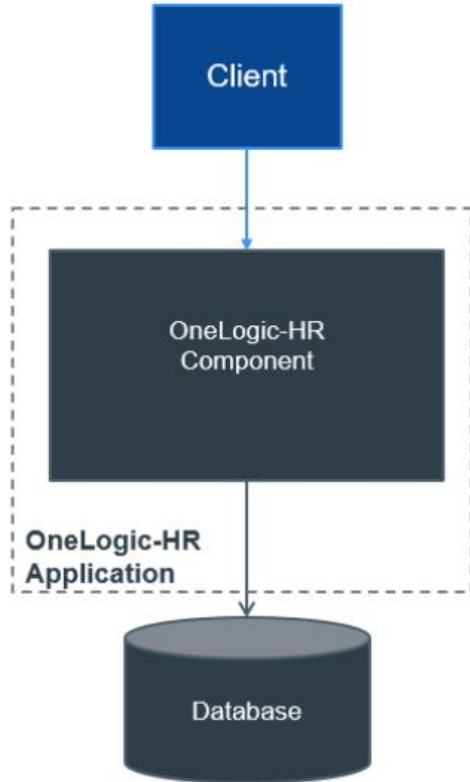
Employé

Congés

Salaire

Notification
Mail/SMS/Push

Monolithique



➤ src/main/java

- ☐ ...afrinnov.onelogic.hr
- ☐ ...afrinnov.onelogic.hr.**domain**
- ☐ ...afrinnov.onelogic.hr.**persistence**
- ☐ ...afrinnov.onelogic.hr.**service**
- ☐ ...afrinnov.onelogic.hr.**web**



Monolithique (les plus...)

❑ **Une grande simplicité de mise en œuvre :**

❑ **Des tests rapides et simplifiés :**

Les tests de bout en bout peuvent être effectués plus rapidement qu'avec une app distribuée, en local par exemple pour la déboguer ;

❑ **Déploiement facile:**

Il suffit de copier-coller un seul fichier exécutable ou répertoire sur un serveur ;

❑ **Moins de latence :**

Les appels sont locaux, les temps de traitement et de réponse sont ainsi réduits ;

❑ **Des intégrations simplifiées :**

Les frameworks, les bibliothèques ou les scripts peuvent être ajoutés rapidement.



- En 2019 - 238 millions – 120 Employés
- COVID-Déc 2021- 60 millions
- En 2022 – 500 Employés,
- 283 Devs (Nantes, Lyon, Berlin et Paris)

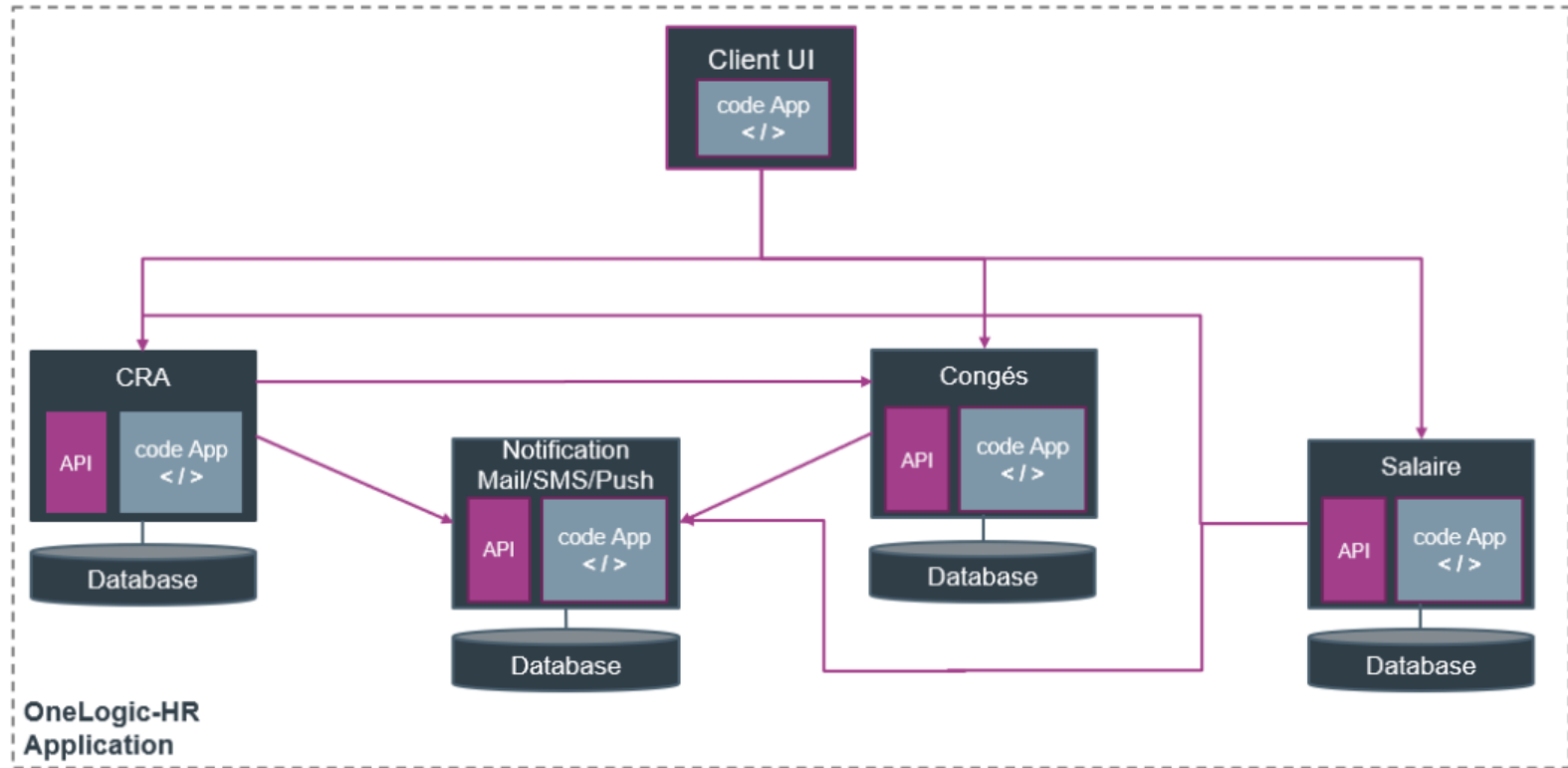
Monolithique (les moins...)

- ❑ **Un potentiel manque de fiabilité :**
Une erreur peut affecter la disponibilité de l'ensemble de l'app
- ❑ **Des redéploiements intégraux :**
Un changement mineur >>> le redéploiement de l'ensemble
- ❑ **Maintenance compliquée :**
Les applications deviennent souvent volumineuses et complexes
- ❑ **Difficultés pour travailler simultanément :**
Nombreux conflits lors des modifications en même temps du code source
- ❑ **Pas de Flexibilité technologique**
Obstacle à l'adoption de nouvelle technologie



Les équipes commencent à sentir le poids de la charge cognitive, tant le volume de code, sa complexité et les efforts de maintien de la cohérence au sein du monolithe sont importants.

Microservices



Communication  service to service

Microservices(les plus...)

☐ **Agilité :**

Promouvoir des méthodes de travail Agile avec de petites équipes qui déploient fréquemment.

☐ **Fiabilité élevée :**

Déployer des changements pour un service spécifique, sans risquer de paralyser l'ensemble de l'app.

☐ **Plus facile à maintenir et à faire évoluer :**

Conteneurs gèrent leurs propres modèles et données,

☐ **Flexibilité technologique:**

Permet aux équipes de sélectionner les outils souhaités,

☐ **Réutilisabilités des services dans différents logiciels et programmes**



- 220 millions d'abonnés dans le monde.
- Plus de 200 pays
- ZUUL (Gateway Service)

Microservices (les moins...)

❑ Applications peuvent devenir rapidement complexes :

- ❑ Gestion de la communication entre les différents
- ❑ Du débogage
- ❑ Des tests d'intégrations

❑ Coûts d'infrastructure exponentiels :

Augmente les coûts d'exploitation à chaque nouveau service

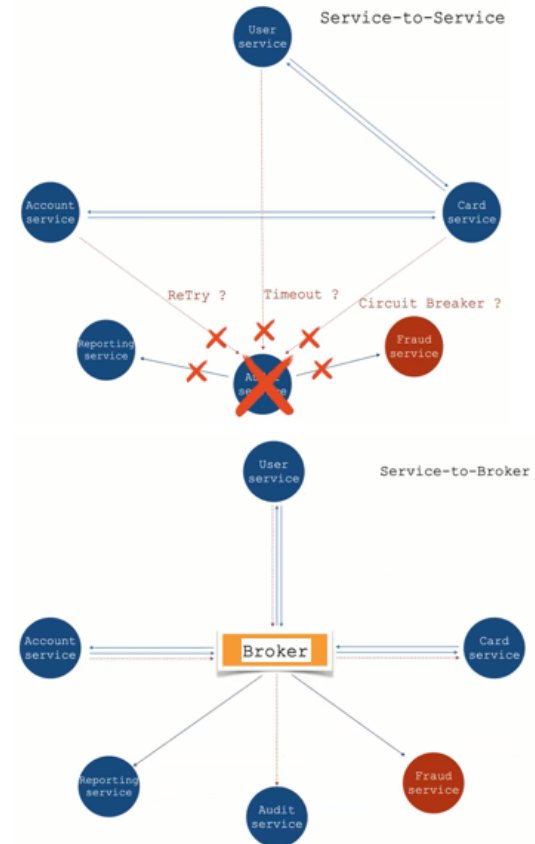
- ❑ déploiement
- ❑ infrastructure d'hébergement
- ❑ outils de surveillance
- ❑

❑ Défis de débogage :

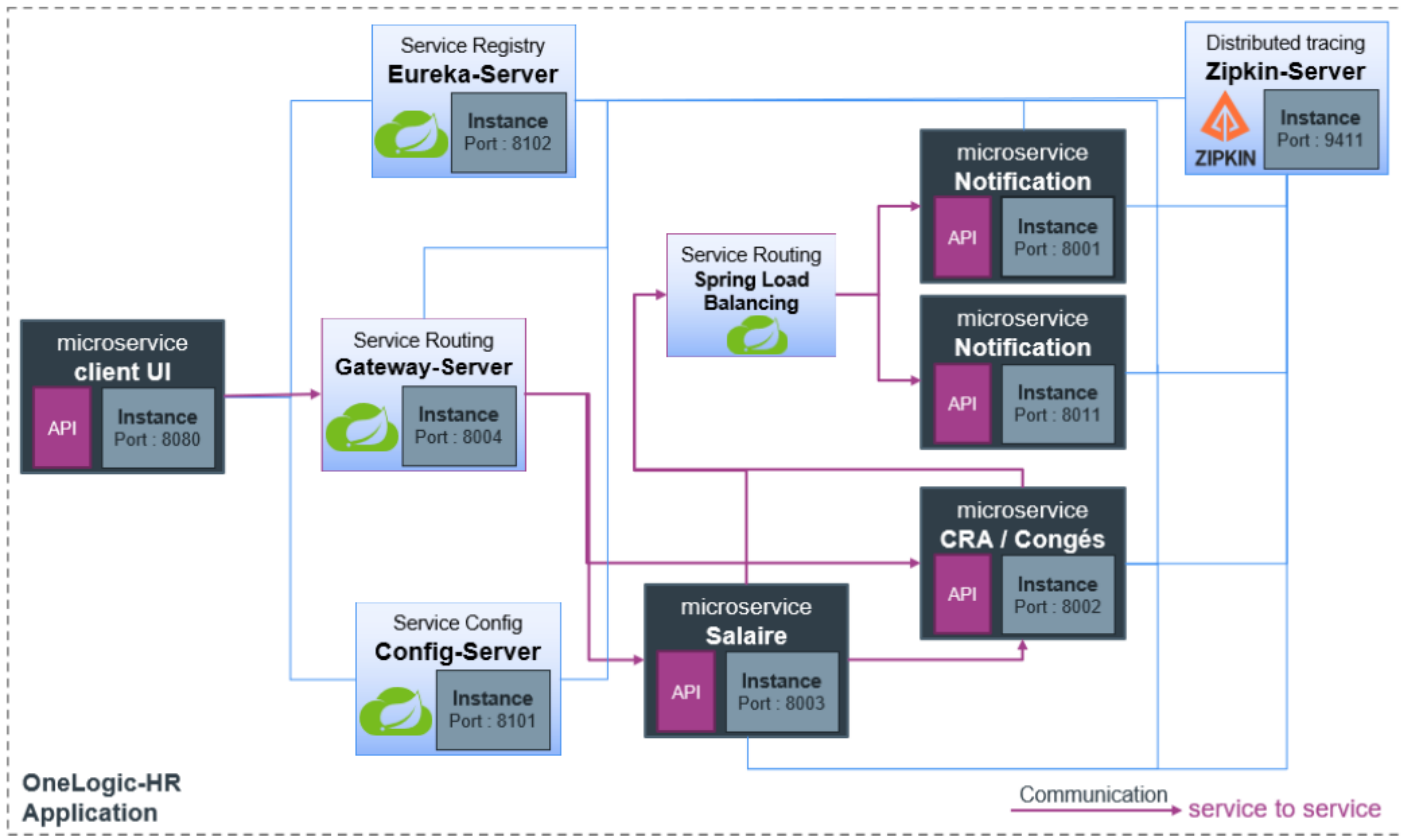
Un seul processus métier peut s'exécuter sur plusieurs machines, ce qui complique encore le processus.

❑ Frais organisationnels supplémentaires

Communication et de collaboration pour coordonner les mises à jour entre les équipes



Microservices (Implémentation avec Spring Cloud)

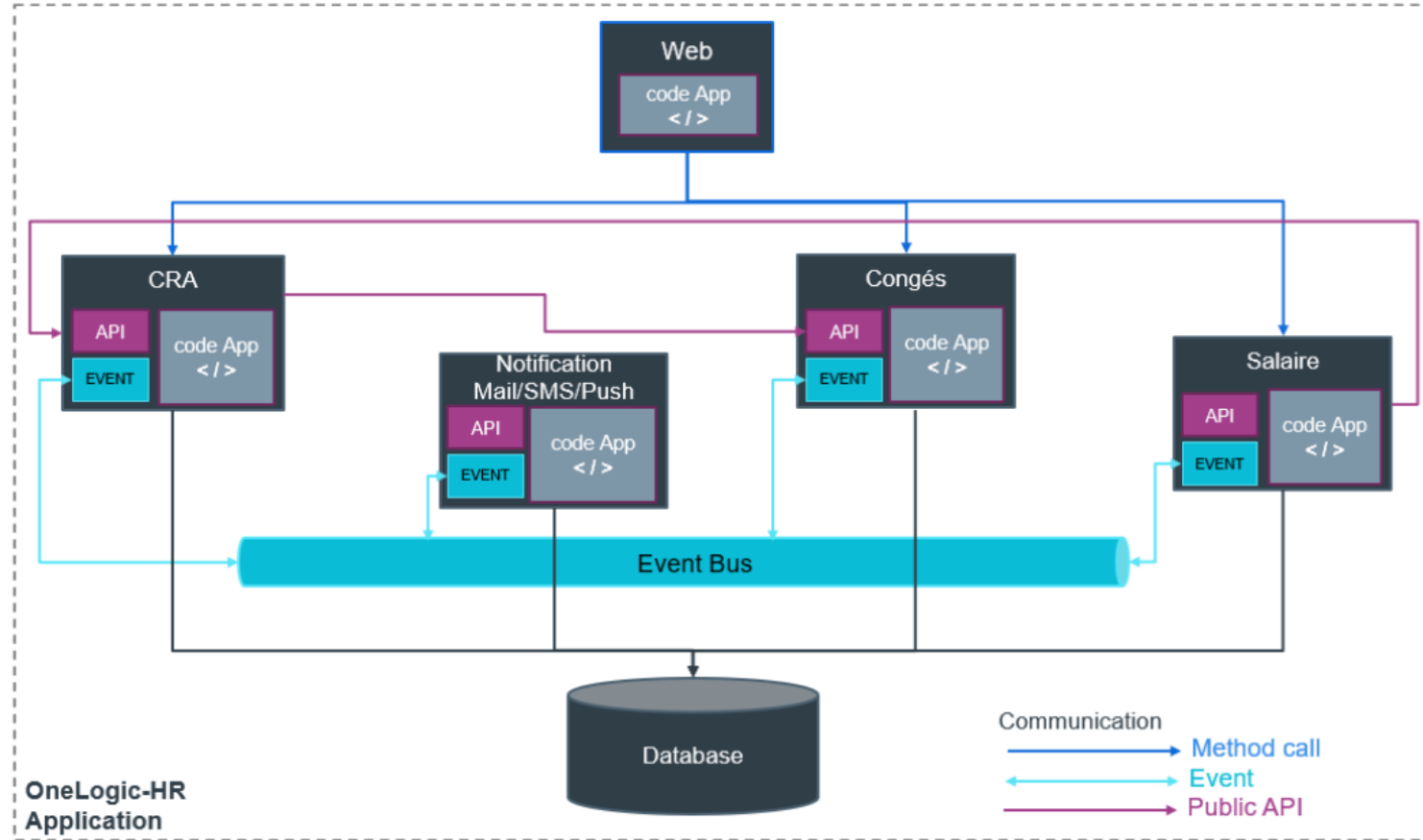


Quelle architecture choisir pour créer un nouveau produit ?

Si vous êtes séduits à la fois :

- La simplicité du monolithe,
- La modularité des microservices,
- L'isolation du code métier, de l'architecture Hexagonale ou du Domain-Driven-Design,
- Le côté asynchrone de l'Event-Driven

Monolithique 2.0 (Modulith)



Monolithique 2.0 (Modulith)

➤ src/main/java

- ❑ ...afrinnov.onelogic.hr
 - ❑ ...afrinnov.onelogic.hr.cra
 - ❑ ...cra.domaine
 - ❑ ...cra.event
 - ❑ ...cra.api
 - ❑ ...cra.service
 - ❑ ...cra.repository
- ❑ ...afrinnov.onelogic.hr.salaire
 - ❑ ...salaire.domaine
 - ❑ ...salaire.event
 - ❑ ...salaire.api
 - ❑ ...salaire.service
 - ❑ ...salaire.repository
- ❑ ...afrinnov.onelogic.hr.conges
 - ❑ ...conges.domaine
 - ❑ ...conges.event
 - ❑ ...conges.api
 - ❑ ...conges.service
 - ❑ ...conges.repository
- ❑ ...afrinnov.onelogic.hr.employe
 - ❑ ...
- ❑ ...afrinnov.onelogic.hr.notification
 - ❑ ...
- ❑ ...afrinnov.onelogic.hr.web
 - ❑ ...

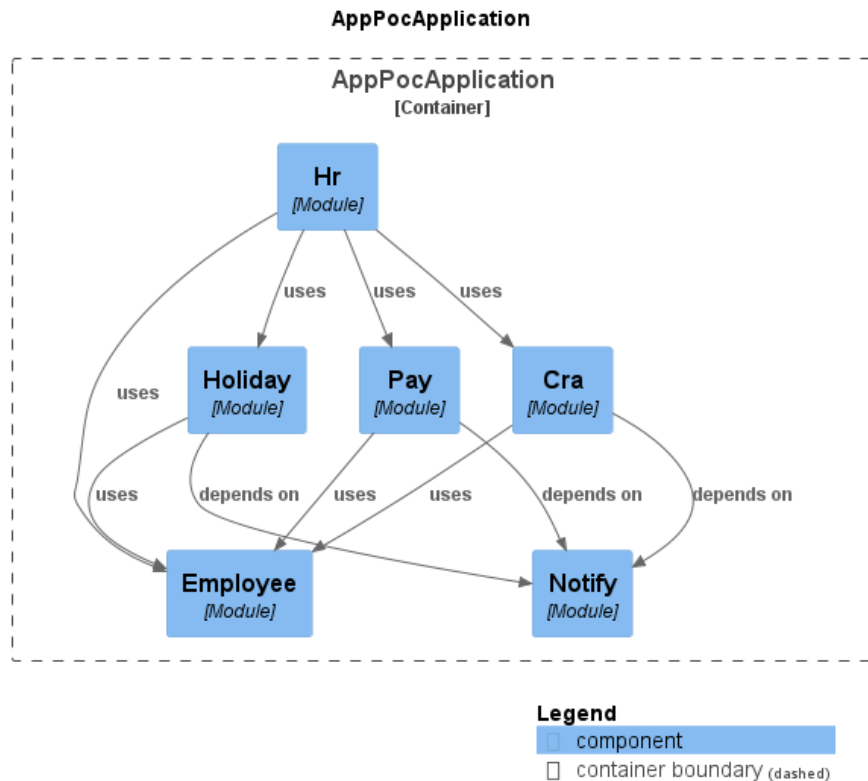


Spring Modulith

- ❑ Spring modulith est un outil basé sur Spring Framework, le site officiel : <https://spring.io/projects/spring-modulith>
- ❑ Ce framework impose au développeur de mieux organiser son application.
- ❑ L'outil fournit un système de contrôle des règles en utilisant un test d'intégration.
- ❑ Une routine est mise à la disposition de développeur pour générer une vue d'ensemble des modules.



Spring Modulith



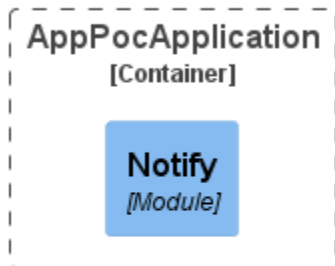
Aperçue générale des modules de l'application.

Les composants sont matérialisés au niveau du code par les packages.



- https://github.com/ngos2024/monolith_next_gen.git

Spring Modulith

Notify



Legend

-  component
-  container boundary (dashed)

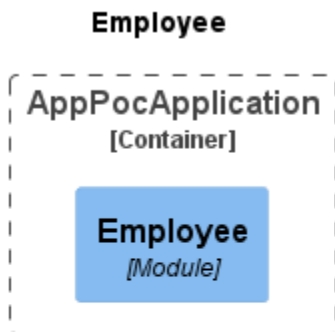
Composant **Notify**.

Dans ce package nous mettons en place les capacités d'envoyer des mails, des SMS ou tout autres signaux



Base package

`org.afrinnov.onelogic.hr.notify`

Spring Modulith



Legend

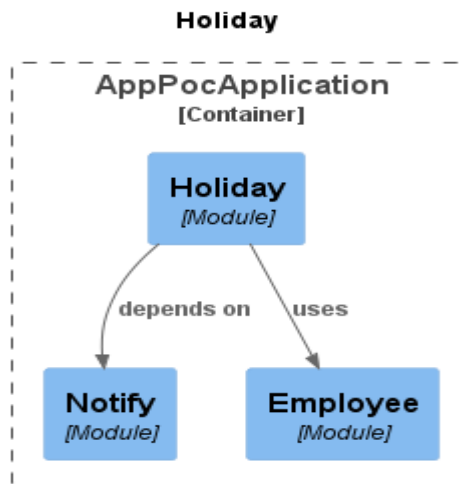
-  component
-  container boundary (dashed)

Composant **Employee**.



Dans ce package Nous avons toutes les fonctionnalités de gestion d'un employé (Génération des matricules, enregistrement et modification des informations d'un employé)

| | |
|-------------------|--|
| Base package | <code>org.afrinnov.onelogic.hr.employee</code> |
| Spring components | <i>Services</i> <ul style="list-style-type: none"><code>o.a.o.h.e.service.EmployeeService,</code> <code>o.a.o.h.e.spi.EmployeeApiService</code> (via <code>o.a.o.h.e.service.adapter.H2EmployeeService</code>) |

Spring Modulith



Legend

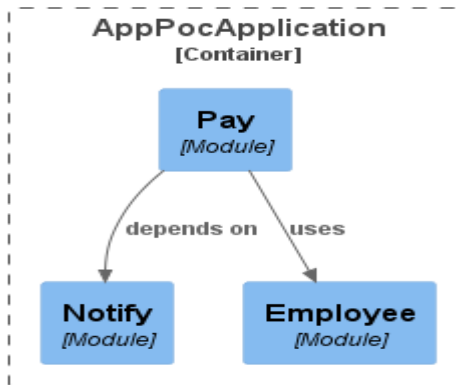
-  component
-  container boundary (dashed)

Composant **Holiday**.



Dans ce package Nous avons toutes les fonctionnalités de gestion des congés d'un employé (Validation, enregistrement, planification) et envoi des notifications

| | |
|-------------------|---|
| Base package | <code>org.afrinnov.onelogic.hr.holiday</code> |
| Spring components | <i>Services</i> <ul style="list-style-type: none"><code>o.a.o.h.h.service.HolidayService</code>, <code>o.a.o.h.h.spi.HolidayApiService</code> (via <code>o.a.o.h.h.service.adapter.H2HolidayService</code>) |
| Bean references | <ul style="list-style-type: none"><code>o.a.o.h.e.spi.EmployeeApiService</code> (in <code>Employee</code>) |

Pay



Legend

-  component
-  container boundary (dashed)

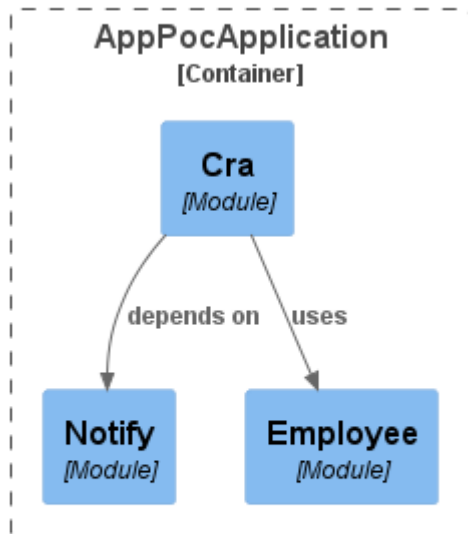
Spring Modulith

Composant **Pay**.

Dans ce package Nous avons toutes les fonctionnalités de gestion de la paie d'un employé (Validation, Évaluation) et envoi des notifications

| | |
|-------------------|---|
| Base package | <code>org.afrinnov.onelogic.hr.pay</code> |
| Spring components | <i>Services</i> <ul style="list-style-type: none"><code>o.a.o.h.p.service.PayService</code>, <code>o.a.o.h.p.spi.PayApiService</code> (via <code>o.a.o.h.p.service.adapter.H2PayService</code>) |
| Bean references | <ul style="list-style-type: none"><code>o.a.o.h.e.spi.EmployeeApiService</code> (in <code>Employee</code>) |

Cra



Legend

- component
- container boundary (dashed)

Spring Modulith

Composant **Cra**.

Dans ce package Nous avons toutes les fonctionnalités de gestion de cra d'un employé (Validation, Saisie) et envoi des notifications

| | |
|-------------------|--|
| Base package | <code>org.afrinnov.onelogic.hr.cra</code> |
| Spring components | <p><i>Services</i></p> <ul style="list-style-type: none"><code>o.a.o.h.c.service.CraService</code>, <code>o.a.o.h.c.service.CraAlertService</code>, <code>o.a.o.h.c.spi.CraApiService</code> (via <code>o.a.o.h.c.service.adapter.H2CraService</code>) |
| Bean references | <ul style="list-style-type: none"><code>o.a.o.h.e.spi.EmployeeApiService</code> (in Employee) |

Hr

AppPocApplication
[Container]

Hr

[Module]

uses

uses

uses

Holiday

[Module]

Pay

[Module]

Cra

[Module]

uses

uses

uses

uses

Employee

[Module]

Legend

□ component

□ container boundary (dashed)

Spring Modulith

Composant **Hr**.

Module d'orchestration dans la plateforme des ressources humaines

Calculer la paie requiert les modules

Employee, Holiday et Cra.

Saisir le cra requiert la vérification des holiday

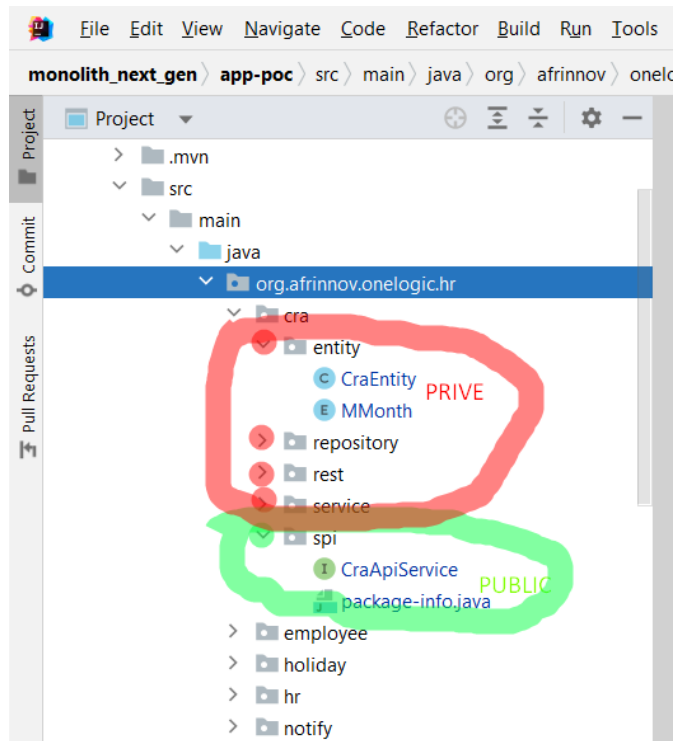
Base package

org.afrinnov.onelogic.hr.hr

Bean references

- o.a.o.h.e.spi.EmployeeApiService (in Employee)
- o.a.o.h.c.spi.CraApiService (in Cra)
- o.a.o.h.p.spi.PayApiService (in Pay)
- o.a.o.h.h.spi.HolidayApiService (in Holiday)

Spring Modulith



- ❑ Le package **cra** est du point de vue Spring Modulith un domain.
- ❑ Le contenu du **module cra** est par **défaut privé**, c'est-à-dire non accessible en dehors du package.
- ❑ par contre le contenu du sous package **cra.spi** est accessible à l'extérieur, ainsi l'interface **CraApiService** est public

Spring Modulith

← → ↻ localhost:8181/swagger-ui/index.html#

cra-controller

- PUT** /api/cra/saisir
- POST** /api/cra/saisir

pay-controller

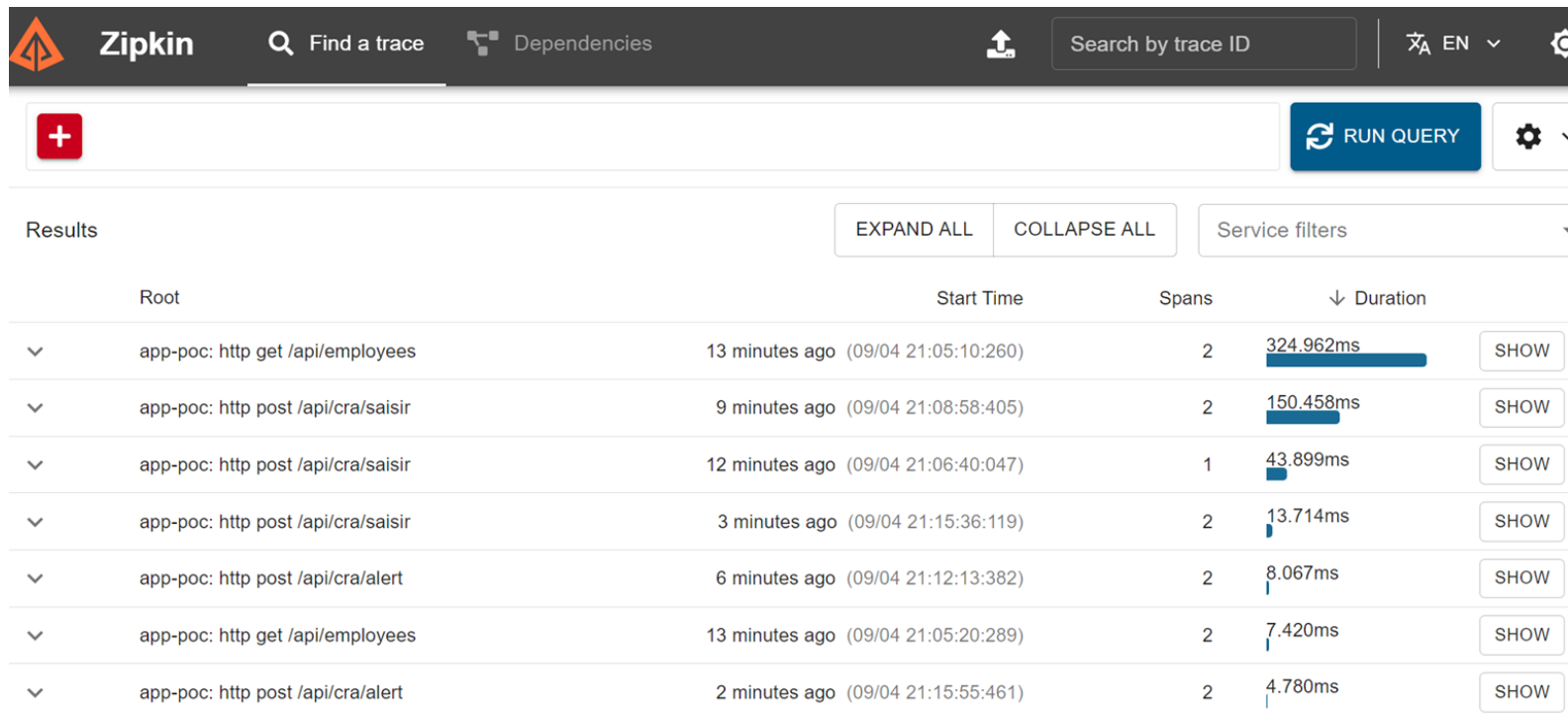
- POST** /api/pay
- POST** /api/pay/validate

holiday-controller

- POST** /api/holiday
- POST** /api/holiday/reject
- POST** /api/holiday/accepted

cra-alert-controller

Spring Modulith



Conclusion

Bien que les deux approches cherchent à découpler les fonctionnalités en modules distincts,

La différence clé entre une **architecture monolithe 2.0** et une **architecture Micoservice**

- ❑ réside donc dans le lieu de déploiement des modules,
- ❑ et les mécanismes de communication entre eux.

Si les modules sont *conçus* pour être déployés indépendamment les uns des autres,

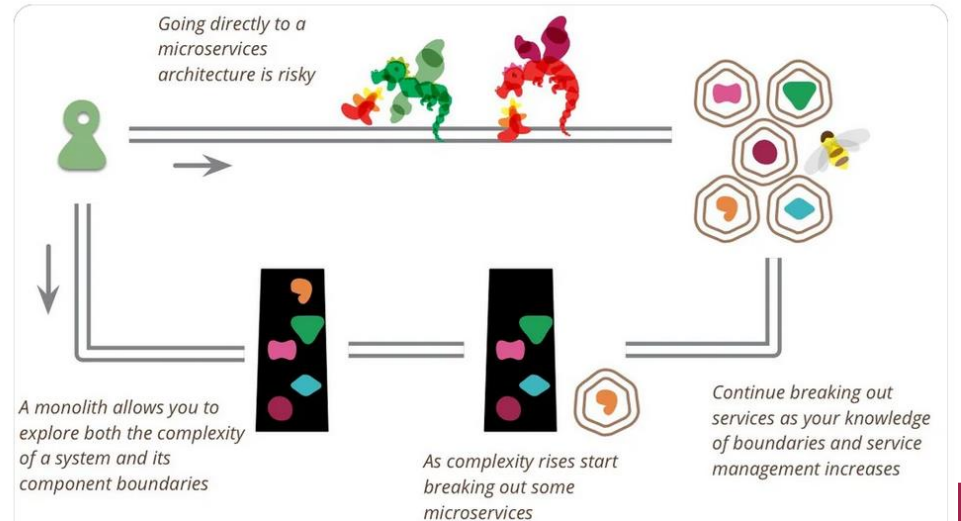
cela signifie que nous avons besoin d'un environnement d'exécution susceptible de permettre le remplacement à chaud des éléments au sein d'un système centralisé.

Spring Modulith encourage l'utilisation des événements applicatifs de **Spring Framework** pour la communication entre eux.

Pour les garder aussi découplés que possible les uns des autres, la publication et la consommation d'événements internes comme principal moyen d'interaction est une pratique architecturale qui a de l'avenir !

En effet, sur cette base, il est d'autant plus facile de basculer vers une architecture distribuée (Event Driven) en s'appuyant sur les **meilleurs brokers du marché**.

Martin Fowler: Monolith First



Références

- ❑ <https://www.atlassian.com/fr/microservices/microservices-architecture/microservices-vs-monolith>
- ❑ <https://fr.slideshare.net/fouomene/migration-dune-architecture-microservice-vers-une-architecture-eventdriven-microservice-avec-kafka>
- ❑ <https://www.wenvision.com/modular-monolith-vs-microservices-architecture>
- ❑ <https://spring.io/projects/spring-modulith>
- ❑ https://github.com/ngos2024/monolith_next_gen.git
- ❑ <https://fr.wikipedia.org/wiki/Doctolib>
- ❑ <https://fr.wikipedia.org/wiki/Netflix>
- ❑ <https://extranet.onelogic.fr/>

Questions?
Réponses!