Construction d’une plateforme éducative avec Vert.x

Retour sur la conception et le développement d’un réseau social éducatif

Rafik Djedjig
Nabil Mansouri

VERT.X

© Tous droits réservés - opendigitaleducation.com

Open Digital Education

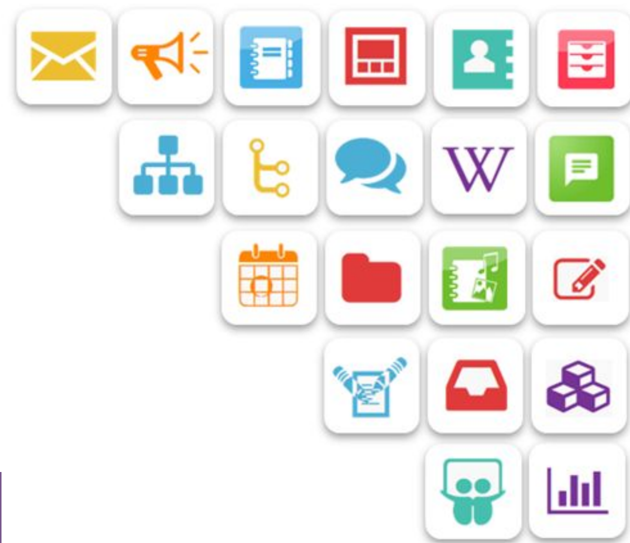
- le projet
- quelques chiffres
- architecture générale

© Tous droits réservés - opendigitaleducation.com

Des applications adaptées aux usages

Des applications variées et évolutives (pédagogie, communication, vie scolaire) et Open Source

Les enseignants choisissent les applications utiles à leurs projets

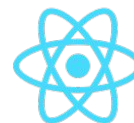


open digital education

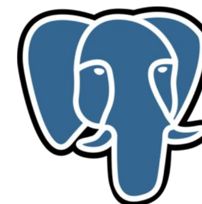
© Tous droits réservés - opendigitaleducation.com

Composants techniques majeurs

TypeScript



VERT.X

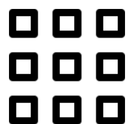


© Tous droits réservés - opendigitaleducation.com

Quelques chiffres

25

Développeurs

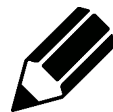


50

application & widgets

2 M

d’élèves inscrits



50k

classes

300M

connexions / an

22

pays

© Tous droits réservés - opendigitaleducation.com

CORE

- Edu Social Network
- App Registry
- Launcher
- Platform Conf
- Framework Edu

CORE APPS

- Admin
- Auth
- Workspace
- Notif
- Portal
- Feeder

EXTENSIONS

more than __50 apps and widgets (packaged as vertx2 modules)__ dedicated to educational field

ODE PLATFORM

SPRINGBOARD

assets, theme, confs, i18, overrides, help-doc, build

© Tous droits réservés - opendigitaleducation.com

Pourquoi Vertx

En 2012 on cherche un framework de développement réactive en Java pour développer une plateforme éducative open-source

- **Java**
- **Reactive Programming**
- **Modules (isolation de classloader)**
- **Modèle de programmation “direct”**

© Tous droits réservés - opendigitaleducation.com

Pourquoi Vert.x

- Java est une contrainte forte pour le projet Open ENT qu'on veut reconstruire
- En 2012 les alternatives avec du reactive programming sont :
  - Play Framework 2 : Version Java est très instable / pas de système de module isolé
  - Servlet 3 : Syntaxe très complexe / ne résout que le sujet du serveur HTTP
- avantages de l'approche Vert.x
  - très peu de dépendances
  - système de module avec classloader isolé et système de déploiement
  - implémentation HTTP qui ne masque pas le protocole
  - runtime très léger

© Tous droits réservés - opendigitaleducation.com

Construction avec Vert.x 2

Vert'x n'est pas un framework MVC. C'est une plateforme modulaire de reactive programming sur la JVM. En 2013 son écosystème est encore naissant

- **Framework web**
- **Module d'accès au données**
- **Serveur d'authentification (CAS, oAuth2)**

© Tous droits réservés - opendigitaleducation.com

Framework Web

- déclaration des API REST par annotation
- chaîne de filtres HTTP extensible pour la sécurité
- cookie
- i18n
- validation de schéma de réponse
- …..

Accès aux données

- Postgres
- Neo4j
- MongoDB
- GridFS
- Swift

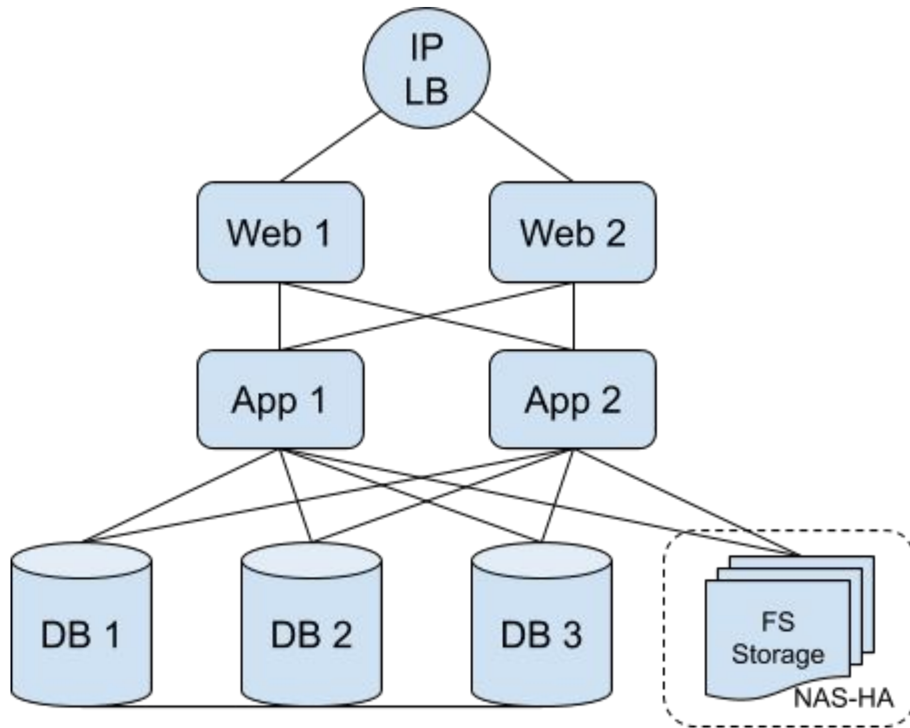
Serveur d'authentification

- CAS 2
- CAS 3
- oAuth 2

nos intentions :

- proposer un framework web conventionnel pour un développeur Java de 2013
- utiliser exclusivement vertx comme plateforme d'intégration et d'exécution

© Tous droits réservés - opendigitaleducation.com



- Web 1 et 2 : 2 CPU, 2 Go de Ram
- App 1 et 2 : 2 CPU, 4 Go de Ram
- DB : 4 CPU, 16 Go de Ram (surtout pour Neo4j)

Infra de production haute-disponibilité

© Tous droits réservés - opendigitaleducation.com

https://github.com/opendigitaleducation/entcore
https://github.com/opendigitaleducation/web-utils

https://github.com/opendigitaleducation/mod-postgresql
https://github.com/opendigitaleducation/mod-mongo-persistor

https://github.com/opendigitaleducation/mod-gridfs-persistor
https://github.com/opendigitaleducation/oauth2-server
https://github.com/opendigitaleducation/cas-server-async

https://github.com/opendigitaleducation/mod-zip
https://github.com/opendigitaleducation/mod-image-resizer

© Tous droits réservés - opendigitaleducation.com

Bilan avec Vert.x 2

- écosystème pauvre (on le savait !)
- **plateforme d'intégration de composants techniques très efficace** (isolation de classloader et isolation du "code bloquant" avec les worker modules)
- (en 2013/2014) les développeurs applicatifs ont surtout eu des difficultés à appréhender le reactive programming
- **Le projet Vert.x est super bien maintenu** (Breaking Change seulement en version majeure, super doc, base de code abordable)
- quelques petits soucis : pas de timeout sur les appels au BUS, instabilité du cluster Hazelcast, pas d'utilitaire pour manipuler le request.body

© Tous droits réservés - opendigitaleducation.com

Passage à Vert.x 3

On a passé une première étape de migration des API io.core.vertx et de rétro-compatibilité des modules Vert.x 2 (qu'on utilise massivement)

- **Rétro-compatibilité avec les modules Vert.x 2**
- **Fonctionnalités de Vertx 3 qu'on utilise réellement**

© Tous droits réservés - opendigitaleducation.com

vertx-service-launcher

Fichier de config

Vertx Service Launcher (verticle)

JAR

Unarchived JAR

NEXUS

VERTX

Nous avons fait une image docker du vertx service launcher

Ce qu'on utilise de vertx 3

- Les AsyncResult pour gérer nos valeurs de retours (bus…) => plus clean
- A vrai dire rien d'autre
  - toute la partie vertx web => web-util
  - tous les mod vertx2 => adapté

© Tous droits réservés - opendigitaleducation.com

Et Maintenant

En 2019

- passage à Java 11
- découpage en microservice
- passage à Vert.x 4

© Tous droits réservés - opendigitaleducation.com

Passage à Java 11

- Le type de class loader a changé (AppClassLoader au lieu de URLClassLoader)
    - La fonctionnalité d'isolation dont nous profitions sera obsolète
    - En réalité ce problème se pose depuis Java 9
- L'équipe Vertx a confirmé qu'il n'y aura pas de rétrocompatibilité
- Nous avons 50 apps qui utilisent ce mécanisme de déploiement
- Comment faire?
    - A priori des contournements existent mais il faudrait l'implémenter nous même
    - Peut être est il temps de changer notre architecture pour d'autres raisons

© Tous droits réservés - opendigitaleducation.com

Passage au Micro Service

Caractéristiques des micro-services

- Architecture décentralisé
- Responsabilité limitée (petite taille)
- Polyglot (aujourd’hui essentiellement au niveau BDD)
- Un support de communication (le bus vertx)
- Déploiement indépendant
- Processus automatisé de déploiement => notre infra met en place kubernetes

© Tous droits réservés - opendigitaleducation.com

Passage à Vertx 4

- On va profiter de cette migration Java / Micro Service pour passer à Vertx 4
  - A noter que nous utilisons Vertx 3.5.0 aujourd’hui
  - La dernière version à ce jour est la 3.7.1
- D’après la roadmap la migration devrait se faire sans trop de problèmes
  - Breaking change sur l’API “Future” (completer)
  - Changement sur le handler du HttpClient
- Pas nécessairement besoin des nouvelles fonctionnalités qui seront apportés

© Tous droits réservés - opendigitaleducation.com

Rétrospective et Perspective

- ce qu’on voulait faire
- ce qu’on a pas fait
- ce qu’on veut faire

© Tous droits réservés - opendigitaleducation.com

Ce qu'on voulait faire

- Contribuer à vertx
    - Vertx 2 fournissait des API plutôt bas niveau pour le web (client / server HTTP)
    - nous avons développé notre propre framework web
        - chaîne de filtre HTTP
        - annotations pour la sécurité
        - annotation type "Spring MVC" pour faire le binding entre méthode Java/HTTP
    - Vertx 3 a ensuite fournit la plupart de ces fonctionnalités dans vertx web

© Tous droits réservés - opendigitaleducation.com

Ce qu'on n'a pas fait (1/3)

- Exploiter pleinement les fonctionnalités de vertx 3
  - remplacer web-util par vertx web
  - ceci a coût de migration important
  - web-util fonctionne plutôt bien (très peu de commit sur ce répo)
  - les gains ne sont pas suffisamment important pour justifier ce coût
- Utilisation d'un ORM
  - Nous utilisons majoritairement des JSONObject / JSONArray pour modéliser nos données
  - Utilisation de classe Java dans de rare cas => en cas de traitement complexes sur les données
    - les classes modélisant notre modèle sont essentiellement en typescript (côté front)

© Tous droits réservés - opendigitaleducation.com

Ce qu’on n’a pas fait (2/3)

- Monitoring
  - Introduction des modules vertx permettant de collecter des “metrics” type “Dropwizard Metrix”
  - On est encore loin d’atteindre nos limites => pas besoin de metrics sur les perf pour le moment
- Migration de nos modules d’accès aux BDD
  - Nous avons “wrappé” des drivers synchrones
  - Vertx fournit des modules basés sur des drivers asynchrones
  - Le coût de migration peut être important car ces modules sont utilisés par toutes les app
  - On pourrait envisager de développer des adaptateurs pour assurer une rétro compatibilité

© Tous droits réservés - opendigitaleducation.com

Ce qu’on n’a pas fait (3/3)

- Multi langage (polyglot)
    - Vertx permet d’utiliser d’autres langages (JS…)
    - On aurait pu utiliser typescript pour notre back
        - on aurait pu partager du code avec notre front (model…)
    - Au lancement typescript n’était pas mature
        - JS est exclu car non typé
    - Nous craignons un overhead (coût supplémentaire lié à l’interprétation du JS dans Nashorn)
    - Nos équipes sont “Java friendly” => Java était donc la meilleur solution
- Utiliser les services proxy vertx pour accèder à des API du bus
    - Nous tirons directement l’implémentation depuis un module commons

© Tous droits réservés - opendigitaleducation.com

Ce qu’on veut faire (1/2)

- Mettre en place du temps réel
  - Vertx fournit une lib SockJS permettant de se connecter au bus via Websocket
  - Nécessité de tester les performances => nombreux utilisateurs simultanés sur nos plateformes
- Utilisation des API Stream / Future
  - Nous n’utilisons du tout les API Stream et très peu les API Future actuellement
    - RxJava pourrait être intéressant notamment pour notre “feeder”
  - On a dans certains cas du code “compliqué” lié à l’orchestration de fonction asynchrone (appels parallèles / séquentiel…)

© Tous droits réservés - opendigitaleducation.com

Ce qu’on veut faire (2/3)

- Mettre en place des tests automatisés
  - essentiellement des tests fonctionnels / intégration
  - en évitant un max l’utilisation de “mock” => pour augmenter la fiabilité des tests
    - pas mal de bugs proviennent des jeux de données
    - ou des échanges inter-modules (nécessité de tester ceux ci)
  - Problèmes principaux
    - les fonctionnalités exposés par d’autres app via le bus? démarrer tous les verticles?
    - avoir des jeux de données réalistes et représentatif des différents cas de figure

© Tous droits réservés - opendigitaleducation.com

Ce qu'on veut faire (2/3)

- Mise en place de logs "agrégés"
  - 1 requête HTTP => 1 groupe de log corrélé (inter module)
  - D'autant plus important lorsqu'on passera en micro service
  - Vertx (sur son blog) propose une "ELK" (LogStash, Elastic Search et Kibana)
  - Ou une solution "plus légère" => message du bus envoyé avec "DeliveryOption" et un header "correlationId"

© Tous droits réservés - opendigitaleducation.com

Bilan

- **Avantages**
- **Inconvénients**

© Tous droits réservés - opendigitaleducation.com

Avantages

- Liberté
- Nous avons de bonnes performances (comparé aux frameworks JEE conventionnels)
- Pas de serveur d'application (type tomcat)
- Programmation asynchrone type NodeJS
- Écosystème Vertx riche

Inconvénients

- Manque de notoriété
- approche boîte à outils moins accessibles qu'un framework intégré

© Tous droits réservés - opendigitaleducation.com

Merci de votre attention

© Tous droits réservés - opendigitaleducation.com