PROJET ECOM

Quentin FAURE - Guillaume HAMMOUTI
Cenyo MEDEWOU - Bin SUN - Zilong ZHAO

PRÉSENTATION DU GROUPE

- Quentin FAURE : Chef de projet, architecture système
- Guillaume HAMMOUTI : Front-end
- Cenyo MEDEWOU : Scrum Master, Back-end
- Bin SUN : Front-end
- Zilong ZHAO : Back-end

GESTION D'ÉQUIPE

- Outils de communication : Slack
- Gestion des tâches : Trello

Boards

Trello

Sprint

ECom

Team Visible

TODO

Ajouter une fonctionnalité créer un compte

ZZ

Duplication de BD

ZZ

faire le produire-detail page

孙斌

Injection de charge (http://jmeter.apache.org ou http://gatling.io/#/)

QF

Add a card…

In progress

http://air.imag.fr/index.php/ECOM-RICM#Livrables

CM QF ZZ GH 孙斌

réalisé le logo du site

孙斌

Gestion des images avec Amazon S3

1

ZZ

Add a card…

Done

Techo Jenkins

2

CM

Commencer la base de donnée

ZZ

Faire le questionnaire

Sep 20

1

CM ZZ

Logging HAProxy

1

QF

Monitoring de performance (Graphana)

1

CM

Mettre en place Grafana, InfluxDB, Metrics avec Docker

Add a card…

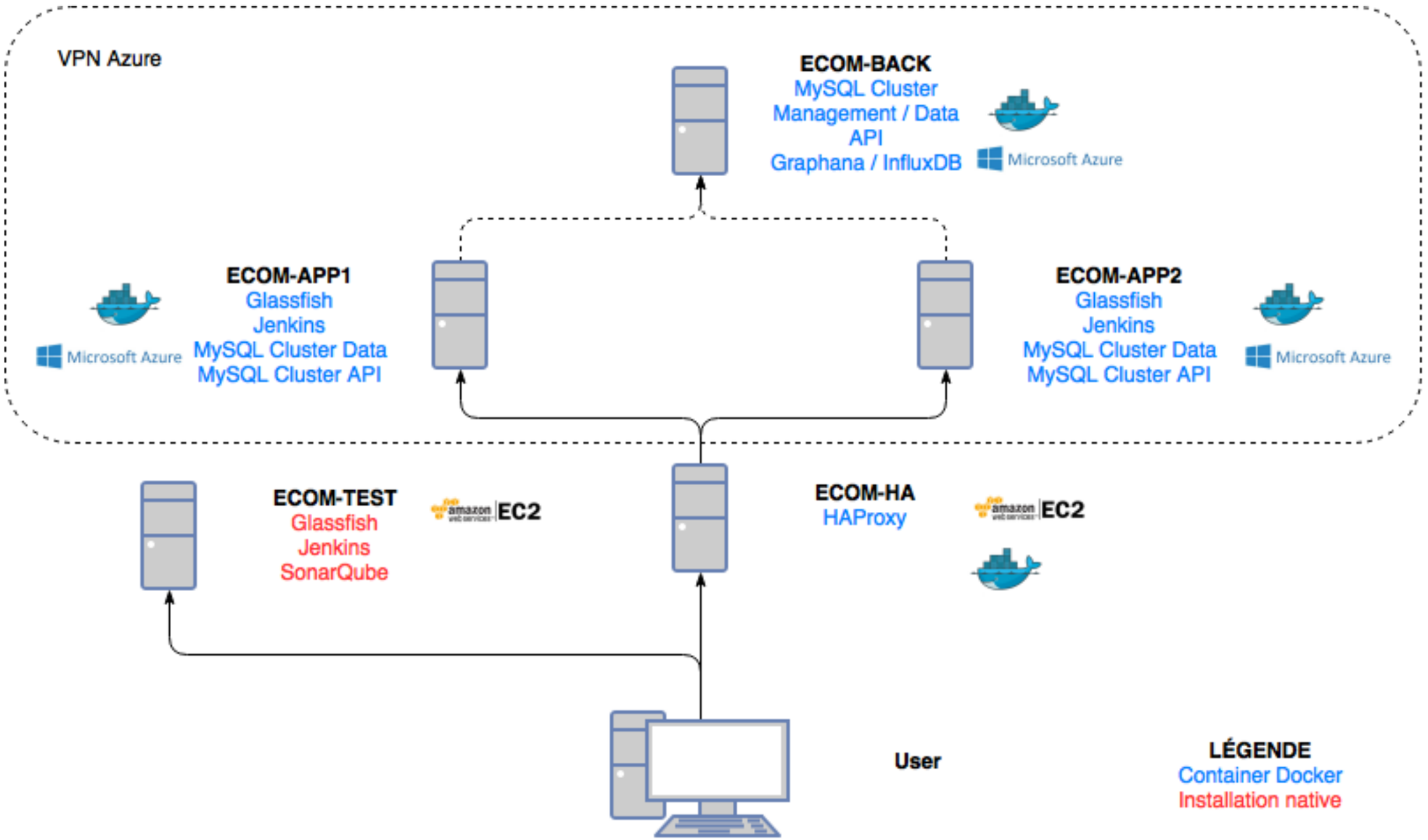
ARCHITECTURE SYSTÈME

- MySQL Cluster : réplication base de donnée
- 3 type de noeuds :
  - Management : Permet d'ajouter des noeuds
  - Data : Sauvegarde des données
  - API : Accès aux données

ARCHITECTURE SYSTÈME

- HAProxy : Load balancer
- « Point d’entrée » du site
- Réparti la charge sur 2 serveurs d’application

ARCHITECTURE SYSTÈME



ARCHITECTURE SYSTÈME

- Pourquoi Azure ?
- Permet de créer un VPN entre les machines
- VPN utilisé pour MySQL Cluster
- Plus de machines sur un compte

ARCHITECTURE SYSTÈME

- Serveur de test
- Premier serveur en place
- Installation native
- Déploiement avec Jenkins (chaque commit)

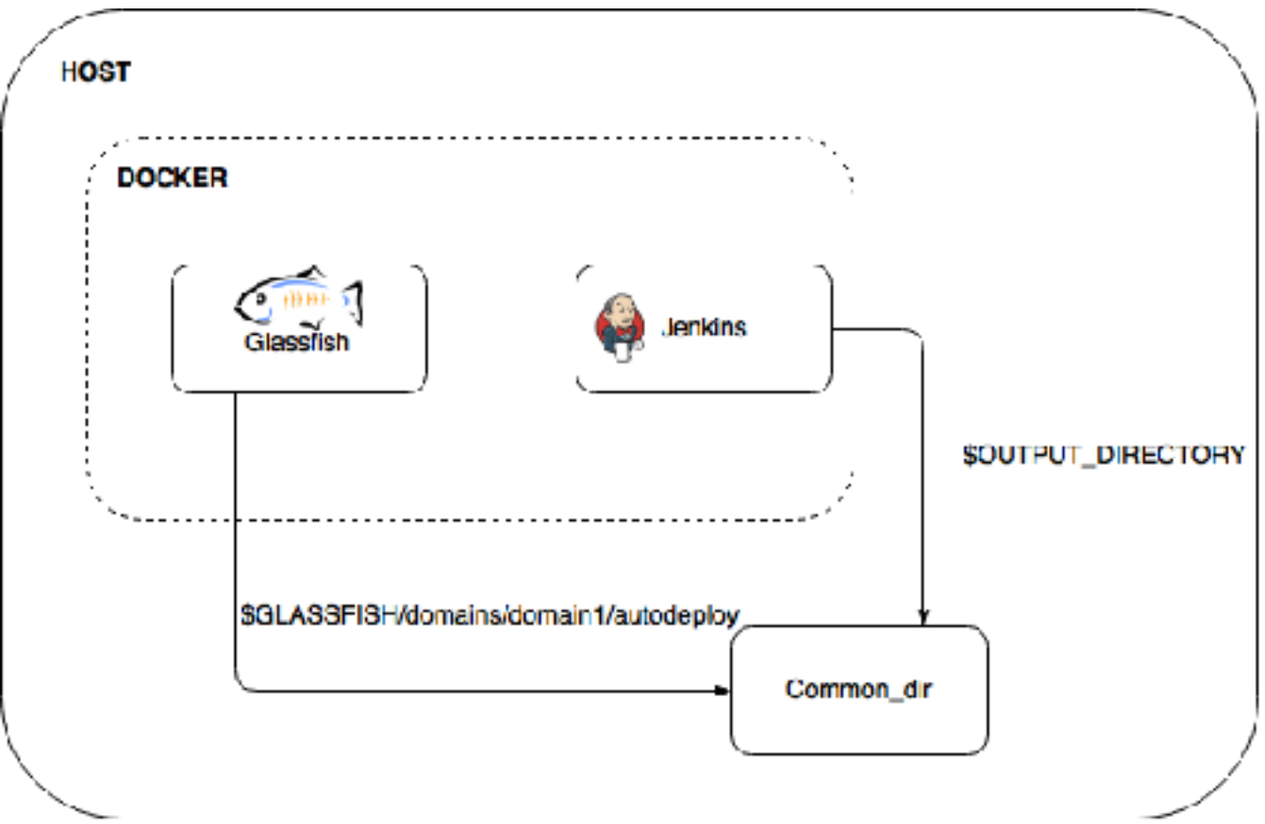
ARCHITECTURE SYSTÈME

- Serveurs d’application
- Mis en place avec Docker
- Déploiement avec Jenkins (versions stables)

ARCHITECTURE SYSTÈME

- Serveur « back »
- Noeud de management MySQL Cluster
- Conserve une copie de la BD
- Non exposé au client

ARCHITECTURE SYSTÈME



- Jenkins et Glassfish :
  - Compilation Jenkins
  - Déploiement direct sur Glassfish

ARCHITECTURE SYSTÈME

- Graphana et InfluxDB
    - Monitoring des serveurs d’application
    - Obtention des données avec Télégraf
- SornarQube : Qualité du code

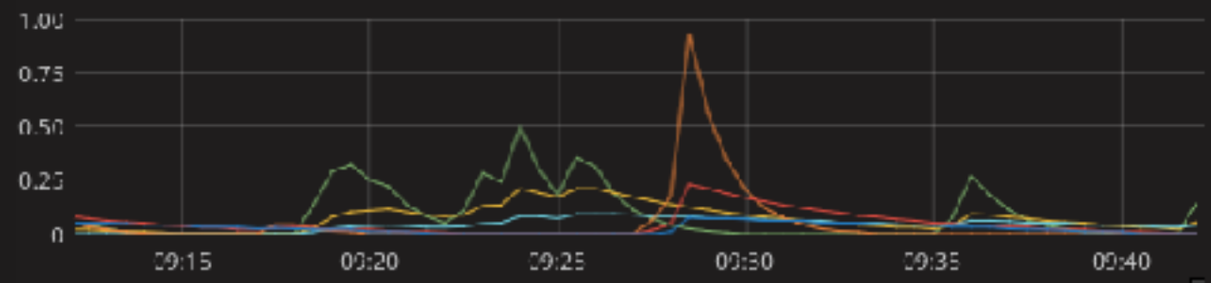
datasource | wadus | Server | ecom-app1 | ecom-app2 | Interval | auto

> SUMMARY

> Network

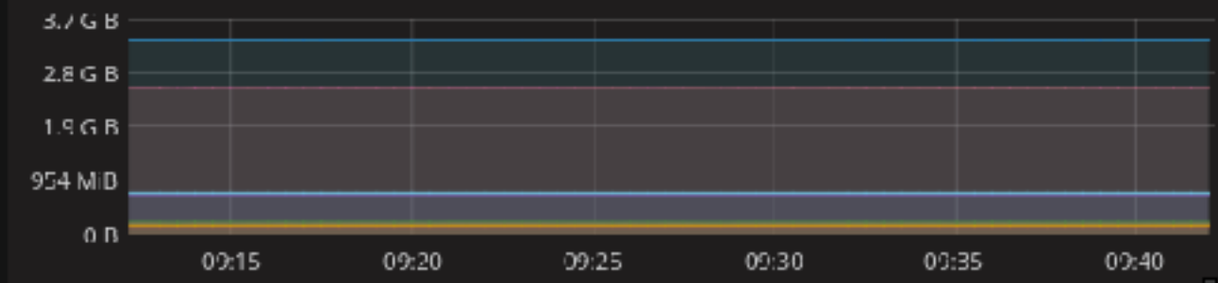
v SYSTEM

Load



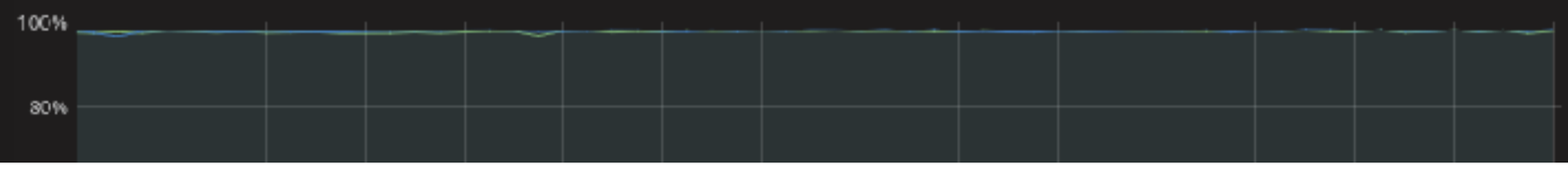
	min	max	avg	current
ecom-app1: load1	0	0.497	0.086	0.145
ecom-app1: load5	0	0.210	0.072	0.055
ecom-app1: load15	0	0.090	0.042	0.040
ecom-app2: load1	0	0.927	0.076	0
ecom-app2: load5	0	0.230	0.045	0

Memory usage



	min	max	avg	current
ecom-app2: total	3.359 GiB	3.359 GiB	3.359 GiB	3.359 GiB
ecom-app1: total	3.359 GiB	3.359 GiB	3.359 GiB	3.359 GiB
ecom-app1: used	2.578 GiB	2.543 GiB	2.539 GiB	2.539 GiB
ecom-app2: used	2.536 GiB	2.540 GiB	2.538 GiB	2.538 GiB
ecom-app1: cached	721 MiB	721 MiB	721 MiB	721 MiB

CPU



	min	max	avg	current
ecom-app2: idle	96%	98%	98%	98%
ecom-app1: idle	97%	98%	98%	98%
ecom-app1: system	1%	1%	1%	1%

BACK-END : GLASSFISH

- Création d'une JDBC-Connection-pool
- Création d'une SQLDataSource
- Création d'un fichier persistence.xml

BACK-END : BEANS

- 6 entity bean (généré depuis la BD) :
    - Admin, Category, Image, Product, User, Command
- Session bean : uniquement stateless

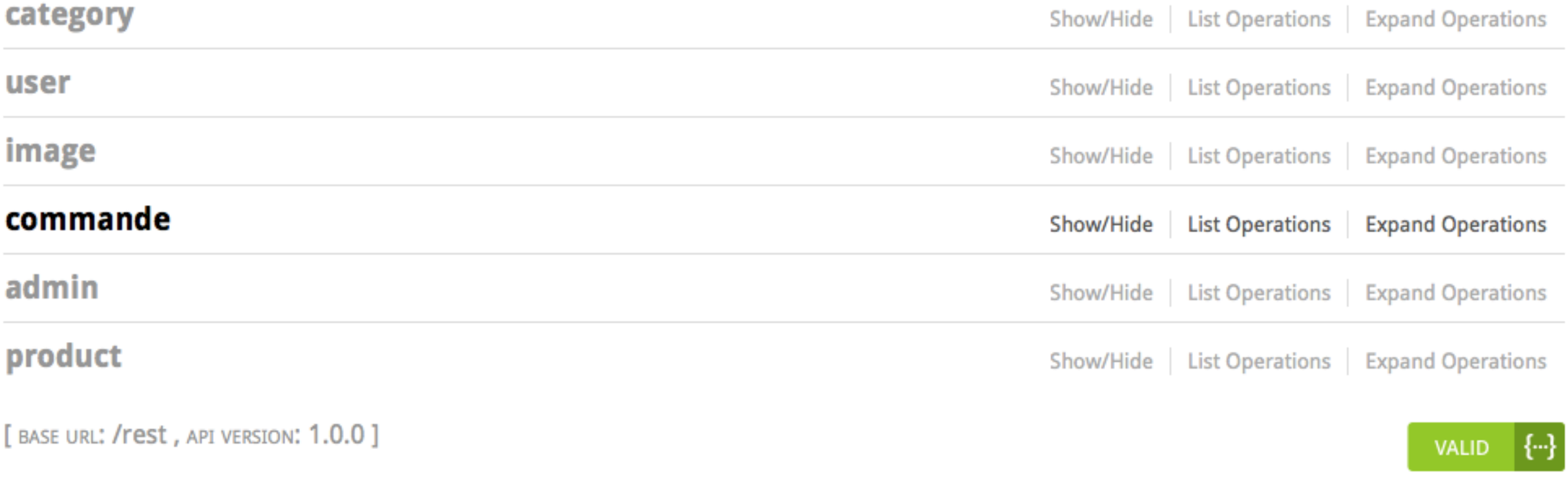
BACK-END : API REST

- Utilisation de Jersey, Jackson et JAXB
  - Jackson : Transformation des classe en JSON
  - Jersey : Création des Web Services Restful
  - JAXB : Transformation des classes en XML

BACK-END : SWAGGER

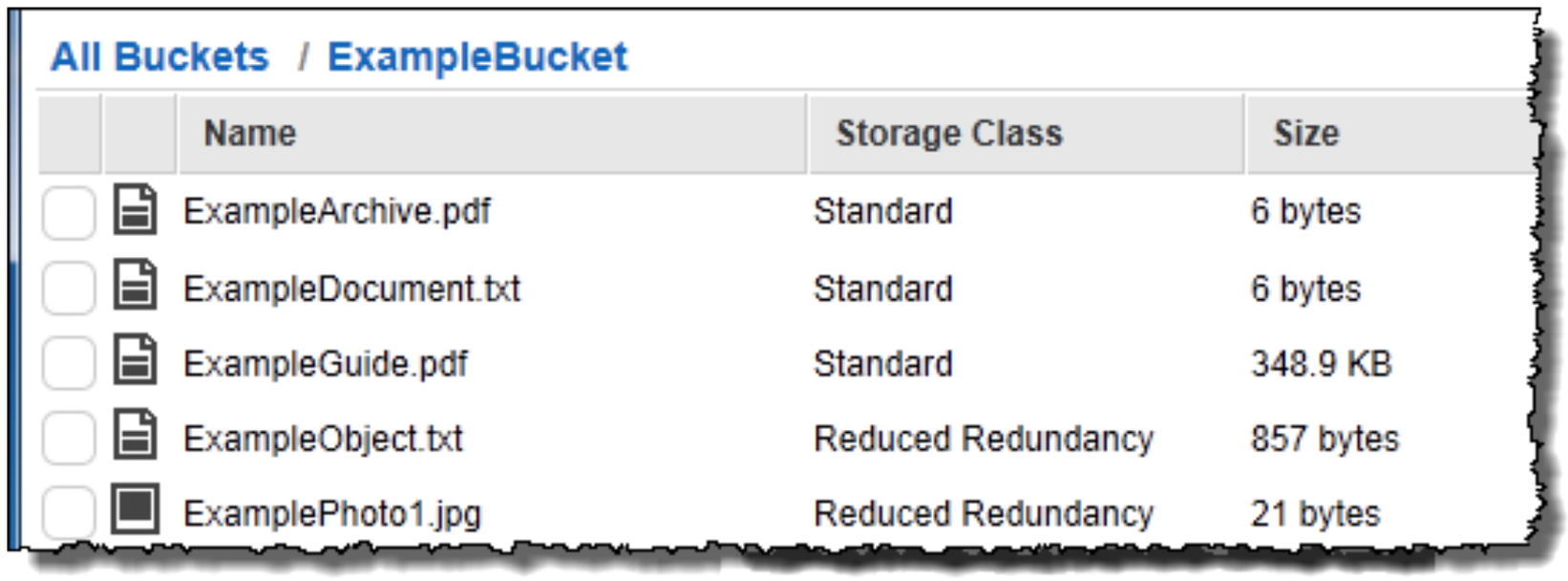
- Documentation de l'API rest
- Test sur l'API rest
- Accès BD et retour JSON

BACK-END : SWAGGER



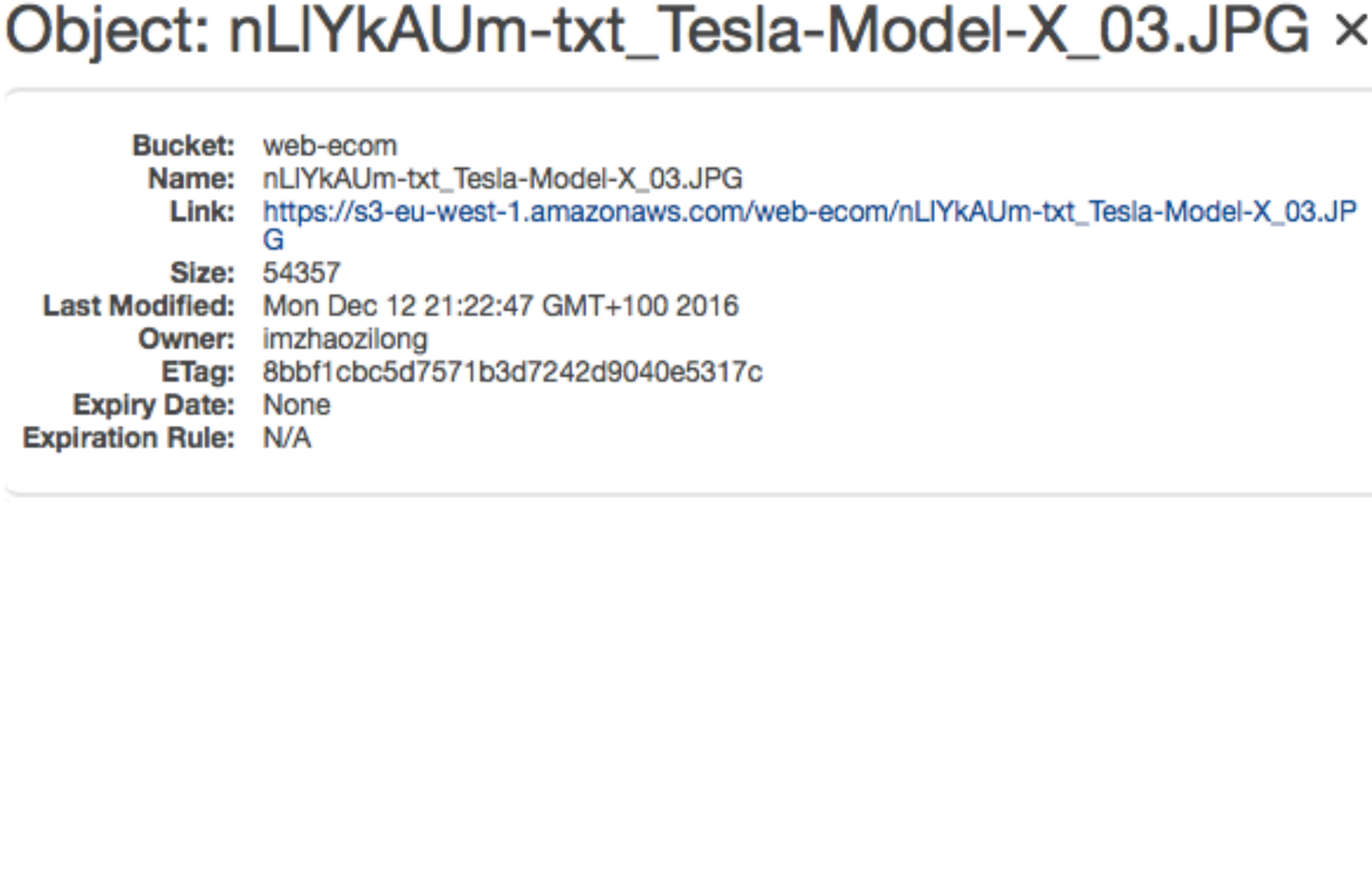
BACK-END : GESTION DES IMAGES

- Utilisation d'Amazon S3
- Notion de Bucket



BACK-END : GESTION DES IMAGES

- Liens Objet : liens vers l'image



Object: nLlYkAUm-txt_Tesla-Model-X_03.JPG ×

Bucket:	web-ecom
Name:	nLlYkAUm-txt_Tesla-Model-X_03.JPG
Link:	https://s3-eu-west-1.amazonaws.com/web-ecom/nLlYkAUm-txt_Tesla-Model-X_03.JPG
Size:	54357
Last Modified:	Mon Dec 12 21:22:47 GMT+100 2016
Owner:	imzhaozilong
ETag:	8bbf1cbc5d7571b3d7242d9040e5317c
Expiry Date:	None
Expiration Rule:	N/A

BACK-END : GESTION DES IMAGES

- Bucket Policy
- Accès au Bucket

Bucket Policy Editor

Policy for Bucket : "web-ecom"

Add a new policy or edit an existing bucket policy in the text area below


```
{
    "Version": "2012-10-17",
    "Id": "Policy1479309540626",
    "Statement": [
        {
            "Sid": "Stmt1479309526244",
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::web-ecom/*"
        }
    ]
}
```

BACK-END : GESTION DES IMAGES

- CORS : Controle d’accès http

CORS Configuration Editor

CORS Configuration for Bucket : "web-ecom"

Using CORS (Cross-Origin Resource Sharing) you can selectively allow web applicatic
bucket. Each CORS rule must contain the set of origins/domains and HTTP methods
the headers users can set in requests or access in responses and the duration the pr

Edit the existing CORS configuration for this bucket in the text area below.

```
<?xml version="1.0" encoding="UTF-8"?>
<CORSConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <CORSRule>
    <AllowedOrigin>*</AllowedOrigin>
    <AllowedMethod>POST</AllowedMethod>
    <AllowedMethod>GET</AllowedMethod>
    <AllowedMethod>PUT</AllowedMethod>
    <AllowedMethod>DELETE</AllowedMethod>
    <AllowedMethod>HEAD</AllowedMethod>
    <AllowedHeader>*</AllowedHeader>
  </CORSRule>
</CORSConfiguration>
```

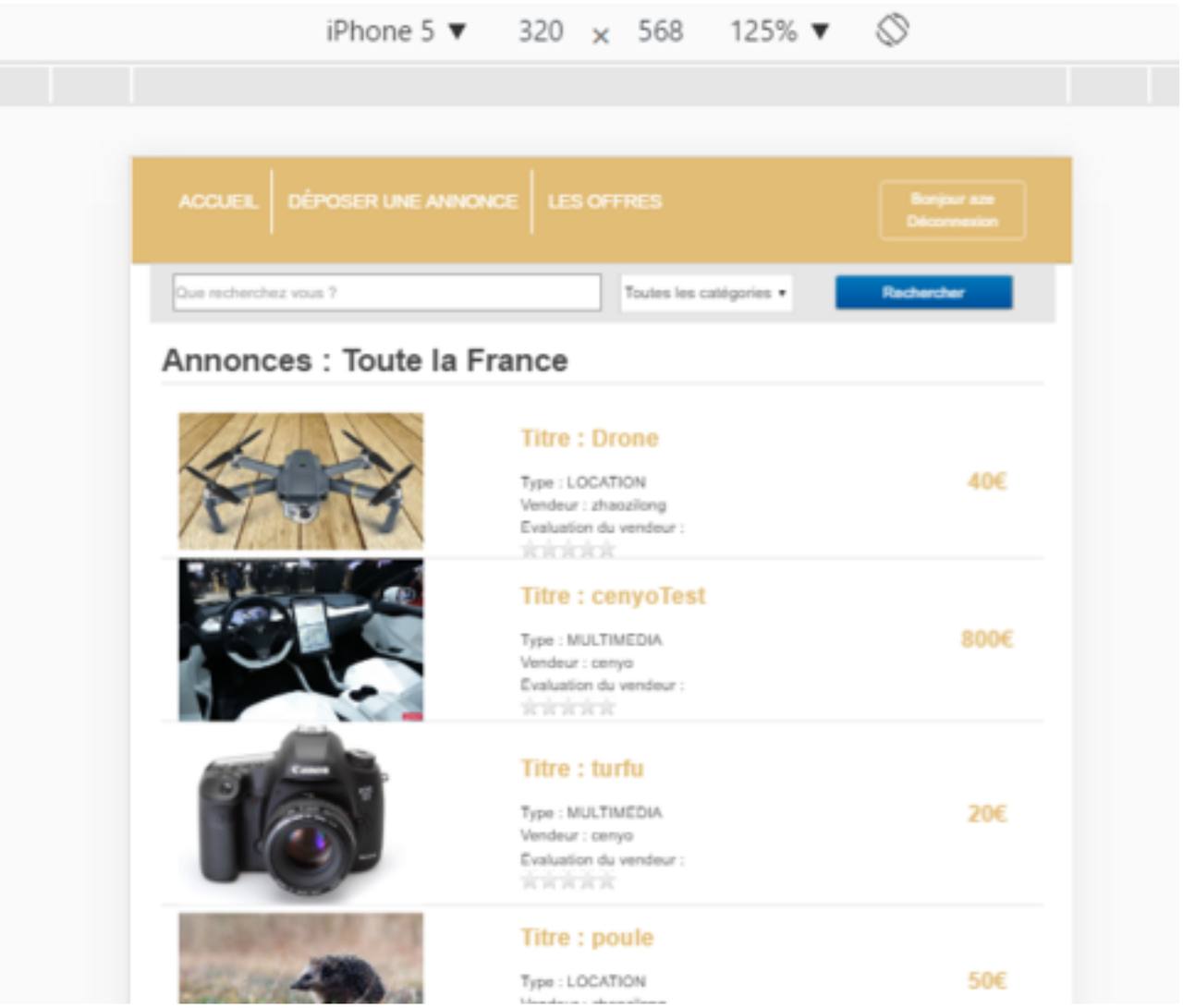
BACK-END : GESTION DES IMAGES

- SDK Amazon S3
- Connection avec Amazon S3
- Envois des images
- Création d’un lien vers l’image (sauvé BD)

BACK-END : E-MAILS

- Utilisation de l'API JavaMail
- Utilisation du SMTP de GMail

FRONT-END



- Design :
  - Homemade
  - MVC
    - HTML5 + CSS
- Bootstrap :
  - Mise en page
  - Responsive

FRONT-END

- jQuery :
  - Carte : jvectormap
  - Evaluation du vendeur
  - Mise en page

FRONT-END : ANGULAR JS

```
$scope.startUpload =function(product) {
    $scope.PictureNames = [];
    $scope.loading = true;
    for (var i = 0; i < $scope.images.length; i++) {
        $scope.upload($scope.images[i]);
    }

    var req = {
            method: 'POST',
            url: '/rest/product/createProduct',
            headers: {
                'Content-Type': 'application/json'
            },
            data: {
                "idProduct" : 0,
                "description" : product.description,
                "idUser" : user,
                "price" : product.prix,
                "title" : product.titre,
                "type" : product.category
            }
};

$http(req).then(function (data, status, headers, config) {

    if(data.status == 200){
        for (var i = 0; i < $scope.PictureNames.length; i++) {
            var url = "https://s3-eu-west-1.amazonaws.com/web-ecom/"+$scope.PictureNames[i];
            var reqIm = {
                    method: 'POST',
                    url: '/rest/image/createImage',
                    headers: {
                        'Content-Type': 'application/json'
                    },
                    data: {
                        "idImage": 0,
                        "idProduct" : data.data.idProduct,
                        "idUser" : user,
                        "imgUrl" : url
```

- Single Page
- Controler

DIFFICULTÉS RENCONTRÉES

- Niveau système
- Faible puissance des machines AWS
- Galéra Cluster
- Limites de Docker Cloud

DIFFICULTÉS RENCONTRÉES

- Niveau back-end
- Version de Glasshfish
- Version de Jackson
- Annotations

DIFFICULTÉS RENCONTRÉES

- Niveau front-end
- Passage à Angular JS
- Gestion des images

DIFFICULTÉS RENCONTRÉES

- Répartition des séances
- Nombre de technologies

CONCLUSION

- Beaucoup de nouvelle technologies vues
- Largement améliorable