

Documents de stratégie de test



Projet allocation de lits d'hôpital pour les urgences

Table des matières

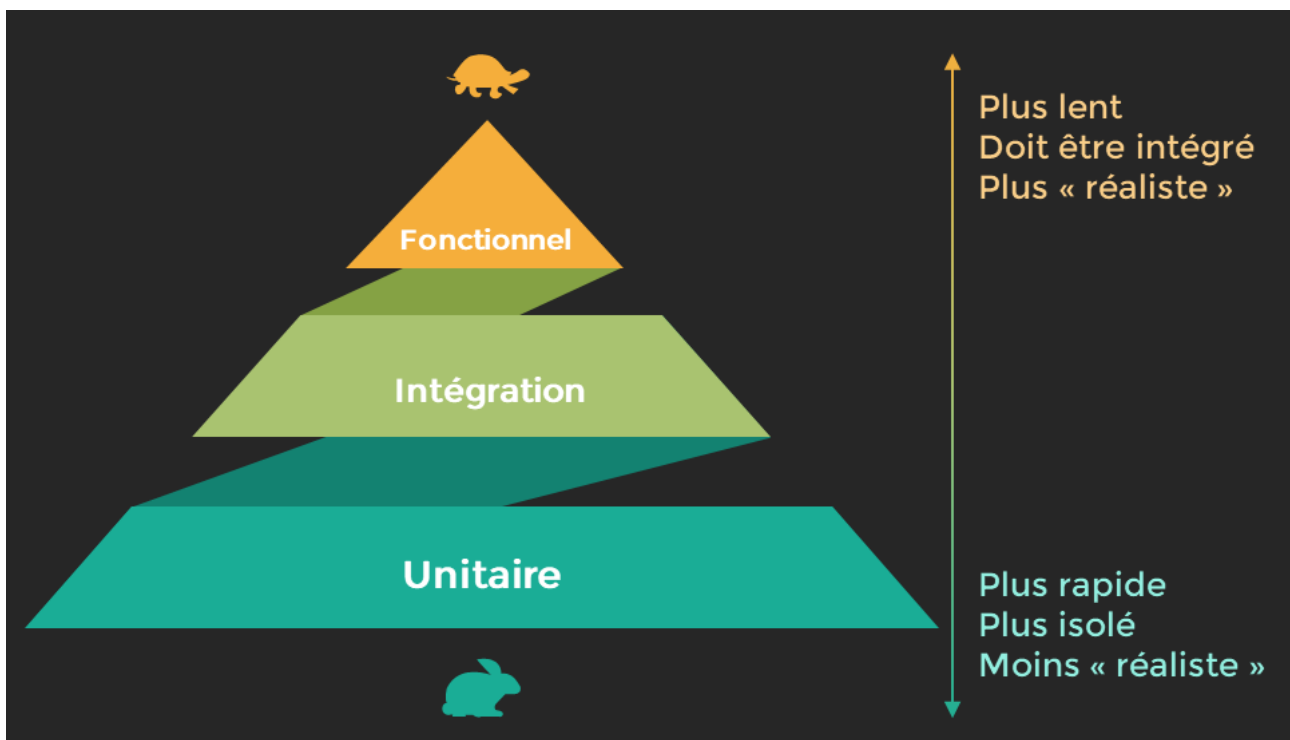
Introduction.....	3
Tests et environnement.....	4
Environnement de développement:.....	4
Bibliothèques et outils.....	4
JUnit 5.....	4
Apache JMeter.....	5
IntelliJ Coverage.....	5
Mockito.....	6
Analyseur de code : Code Climate.....	6
Tests unitaires.....	6
Tests d'intégration.....	8
Tests fonctionnels.....	9
Stress tests.....	10

Introduction

Ce document a pour objectif de donner une description la plus complète possible de la stratégie de test mis en place dans le cadre de la preuve de concept (PoC)

Il décrit les différents type de tests, leur portée et la méthodologie utilisé pour les mettre en place.

Enfin, il précisera les modalités d'automatisation des différents tests.



La stratégie de test fera intervenir les 3 types de tests présents dans la pyramide de test :

- Des tests fonctionnels
- Des tests d'intégrations
- Des tests fonctionnels

Tests et environnement

Environnement de développement:

Système d'exploitation : Ubuntu 21,10

Java : 11 'Temurin'

Interface de développement : IntelliJ IDEA Community Edition v2022.1.3

Maven : 4.0.0

SpringBoot : 2.7.0

Git : 2.32.0

Bibliothèques et outils

JUnit 5



JUnit 5 est un framework de test pour le langage Java. Il est intégré par défaut sur les environnements de développement (IDE)

Apache JMeter



Apache JMeter est une application libre permettant d'effectuer des tests de performance d'application ou de serveurs.

Il fonctionne en simulant le comportement de plusieurs utilisateurs agissant de manière simultanée sur une application.

Dans notre cas, il sera utilisé pour effectuer les stress tests de l'application.

IntelliJ Coverage

Element	Class, %	Method, %	Line, %
com	70% (7/10)	65% (27/41)	63% (76/119)
medhead	70% (7/10)	65% (27/41)	63% (76/119)
api	70% (7/10)	65% (27/41)	63% (76/119)
RunApi	100% (1/1)	0% (0/1)	50% (1/2)
service	100% (2/2)	80% (12/15)	75% (48/64)
HospitalService	100% (1/1)	83% (10/12)	88% (22/25)
GisOperations	100% (1/1)	66% (2/3)	66% (26/39)
repository	100% (1/1)	100% (1/1)	100% (2/2)
HospitalRepositoryCust	100% (0/0)	100% (0/0)	100% (0/0)
HospitalRepository	100% (0/0)	100% (0/0)	100% (0/0)
customProperty	100% (1/1)	100% (1/1)	100% (2/2)
model	50% (1/2)	80% (8/10)	61% (8/13)
Itineraire	0% (0/1)	0% (0/2)	0% (0/5)
Hospital	100% (1/1)	100% (8/8)	100% (8/8)
exceptions	33% (1/3)	33% (1/3)	33% (1/3)
NotYetImplemented	0% (0/1)	0% (0/1)	0% (0/1)
NoParametersEntered	0% (0/1)	0% (0/1)	0% (0/1)
NoHospitalFound	100% (1/1)	100% (1/1)	100% (1/1)
controller	100% (1/1)	45% (5/11)	45% (16/35)

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
com.medhead.api.service	<div><div></div></div>	29 %	<div><div></div></div>	22 %	20	32	45	66	11	21	0	2
com.medhead.api.model	<div><div></div></div>	13 %	<div><div></div></div>	0 %	36	47	10	14	11	22	1	2
com.medhead.api.controller	<div><div></div></div>	1 %	<div><div></div></div>	0 %	19	20	34	35	11	12	0	1
com.medhead.api.exceptions	<div><div></div></div>	0 %	<div><div></div></div>	n/a	3	3	6	6	3	3	3	3
com.medhead.api	<div><div></div></div>	37 %	<div><div></div></div>	n/a	1	2	2	3	1	2	0	1
com.medhead.api.repository	<div><div></div></div>	100 %	<div><div></div></div>	n/a	0	2	0	2	0	2	0	1
Total	749 of 934	19 %	83 of 88	5 %	79	106	97	126	37	62	4	10

Le code coverage, que ce soit via IntelliJ ou Jacoco, est un programme qui permet de mesurer le nombre de ligne de code qui sont couvertes par les tests.

Mockito



Tout comme Junit, Mockito est un framework de test pour les applications en Java.

Il permet de créer des « Mock », des objets fictifs qui vont intervenir dans le cadre des tests unitaires.

Analyseur de code : Code Climate

Showing 3 of 3 total issues

Method `getItineraireOSRM` has 31 lines of code (exceeds 25 allowed).

Consider refactoring.

OPEN

```
49     public static Iterable<Itineraire> getItineraireOSRM(float longCentre, float latCentre, It
50
51     String apiUrl = "https://api.openrouteservice.org/v2/directions/driving-car?api_key=";
52     String apiKey = "5b3ce3597851110001cf6248dc9db6fb880d46b7b63a907d42d826e5" ;
53     ArrayList<Itineraire> myItineraireList = new ArrayList<>();
```

Found in `src/main/java/com/medhead/api/service/GisOperations.java` - About 1 hr to fix

Similar blocks of code found in 2 locations. Consider refactoring.

OPEN

```
127     @GetMapping("/hospital/free/speciality/{spec}")
128     public Iterable<Hospital> getAllFreeBySpec(@PathVariable("spec") final String speciality)
129     ArrayList<Hospital> myHospitalList = (ArrayList<Hospital>) hospitalService.findByFreeb
130     if (!myHospitalList.isEmpty()){
131         return hospitalService.findByFreebedAndBySpecialities(speciality);
```

Found in `src/main/java/com/medhead/api/controller/HospitalController.java` and 1 other location - About 35 mins to fix

Similar blocks of code found in 2 locations. Consider refactoring.

OPEN

```
111     @GetMapping("/hospital/speciality/{spec}")
112     public Iterable<Hospital> getAllBySpec(@PathVariable("spec") final String speciality) thro
113     ArrayList<Hospital> myHospitalList = (ArrayList<Hospital>) hospitalService.findBySpeci
114     if (!myHospitalList.isEmpty()){
115         return hospitalService.findBySpecialities(speciality);
```

Found in `src/main/java/com/medhead/api/controller/HospitalController.java` and 1 other location - About 35 mins to fix

Tests unitaires

Les tests unitaires consistent en des tests d'unités plus restreintes du code.

Chaque fonctionnalité est testé de manière isolée pour vérifier que le comportement se déroule comme convenu.

Comme le montre la pyramide des tests, ils sont très rapides et simples, pour le programme de test, à effectuer.

Les tests unitaires ont été mis en place pour les méthodes déclarées manuellement dans la classe *HospitalRepositoryCustom* :

Test	Donnée	Attendu
findByFreebed()	Data.sql	Liste de 859 réponses
findBySpecialities()	Data.sql Spécialité imposée : <i>pharmacology</i>	Liste de 200 réponses
findByFreebedAndBySpecialities()	Data.sql Spécialité imposée : <i>pharmacology</i>	Liste de 132 réponses

La suite des tests unitaires sera ceux qui vont impacter la classe de service *HospitalService*

Pour ces tests, il ne sera pas utilisé les données du fichier *data.sql*, mais 3 objets de type *Hospital* :

```
Hopital1 : {
id = 1,
name = Cromwell Hospital,
longitude = -1.21137082993984222
latitude = 51.766518859863281,
spécialities : audio vestibular medicine,prosthodontics,endocrinology and diabetes mellitus,restorative
dentistry
Address = 164-178 Cromwell Road London SW5 0TU
Freebed = 145
}
Hopital2 : {
id = 1,
name = Princess Grace Hospital,
```

```
longitude = -0.15291328728199005
```

```
latitude = 51.495105743408203,
```

```
spécialités : pharmacology, acute internal medicine,oral and maxillo-facial surgery,paediatric dentistry,clinical neurophysiology
```

```
Address = 42-52 Nottingham Place London W1U 5NY
```

```
Freebed = 0
```

```
}
```

```
Hopital3 : {
```

```
id = 1,
```

```
name = Spire Tunbridge Wells Hospital,
```

```
longitude = 0.1881597638130188
```

```
latitude = 51.134513854980469,
```

```
spécialités : gastroenterology,oral medicine,restorative dentistry,additional dental specialties
```

```
Address = Tunbridge Wells Hospital Fordcombe Road Fordcombe Tunbridge Wells TN3 0RD
```

```
Freebed = 107
```

```
}
```

Test	Donnée	Attendu
getAllHospital()	Liste d'hopitaux	Liste de 3 réponses
getHospital()	Liste d'hopitaux	Le nom de l'hôpital avec l'ID 1
findByFreebed()	Liste d'hopitaux	Liste de 2 réponses
getAllHospitalInRange()	Liste d'hopitaux	Liste de 1 réponse

Tests d'intégration

Les tests d'intégration sont là pour vérifier que les différentes unités du code (classes, méthodes) fonctionnent correctement ensemble, comme prévu. Ils viennent après les tests unitaires, et sont plus fiables sur le bon fonctionnement de l'application finale, du fait de la vérification des interactions entre les unités.

Les tests d'intégrations ont été mis en place pour les différentes méthodes du contrôleur via les appels de l'API:

Test	Donnée	Attendu
/hospital	Data.sql	<ul style="list-style-type: none"> Liste de 1290 réponses Champ « name » de la première réponse : « Walton Community Hospital - Virgin Care »

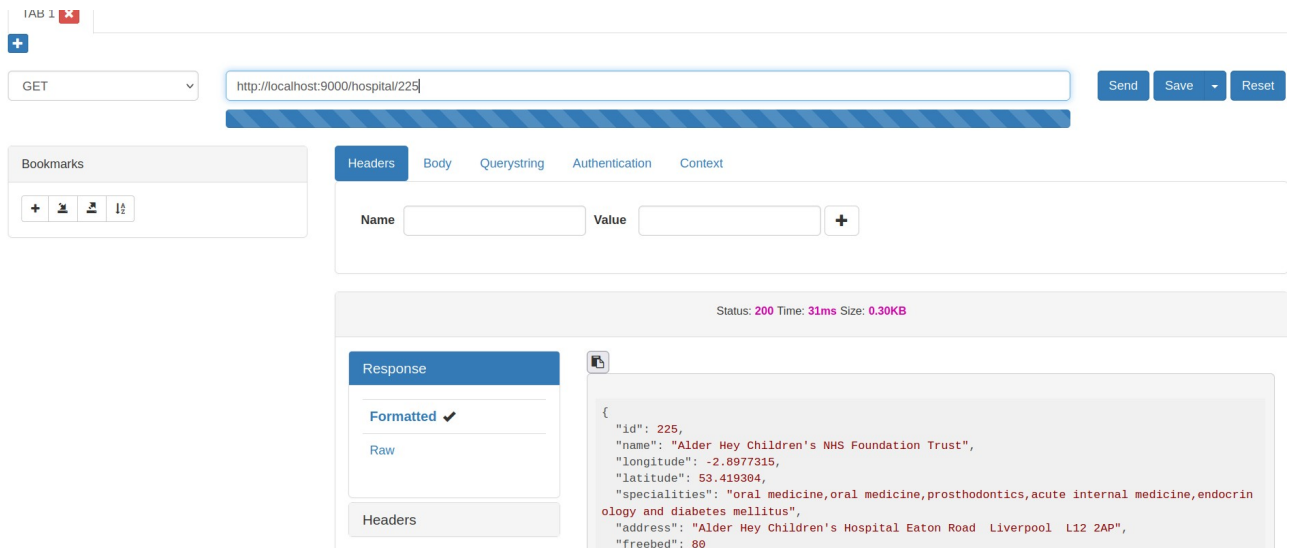
		<i>Services Ltd »</i> <ul style="list-style-type: none"> Champ « <i>address</i> » de la seconde réponse : « <i>Heathside Road Woking Surrey GU22 7HS</i> » Champ « <i>freebed</i> » de la troisième réponse : 83
/hospital/4	Data.sql Id imposé : 4	<ul style="list-style-type: none"> Champ « <i>Id</i> » : 4 Champ « <i>name</i> » : « <i>Bridgewater Hospital</i> » Champ « <i>address</i> » : « <i>120 Princess Road Manchester Greater Manchester M15 5AT</i> » Champ « <i>freebed</i> » : 0
/hospital/speciality/ pharmacology	Data.sql Spécialité imposée : pharmacology	<ul style="list-style-type: none"> Liste de 200 réponses Champ « <i>name</i> » de la première réponse : « <i>Bridgewater Hospital</i> » Champ « <i>address</i> » de la 7^e réponse : « <i>The Alexandra Hospital Mill Lane Cheadle Cheshire SK8 2PX</i> » Champ « <i>freebed</i> » de la 4^e réponse : 0
/hospital/free/ speciality/ pharmacology		<ul style="list-style-type: none"> Liste de 132 réponses Champ « <i>name</i> » de la première réponse : « <i>Nuffield Health</i> » Champ « <i>address</i> » de la 7^e réponse : « <i>Benslow Lane Hitchin Hertfordshire SG4 9QZ</i> » Champ « <i>freebed</i> » de la 4^e réponse : 29
/hospital/400258	Data.sql	Status : Not found
/nearest/-1.8/51.25/2/ pharmacology	Data.sql	Status : Not found

Tests fonctionnels

Les tests fonctionnels ont été réalisés sur navigateur avec le plugin pour le navigateur *Mozilla Firefox « Resting »*, version 1.4.0

Les différents tests ont été passés avec le démarrage de l'API dans l'environnement de développement (Intellij IDEA) ainsi que qu'après démarrage de l'image docker, ce qui a permis une validation du bon fonctionnement de celle-ci.

Ces tests ont couvert toutes les adresses présentes dans la classe *HospitalController* de l'API.



GET Send Save Reset

Bookmarks: + -

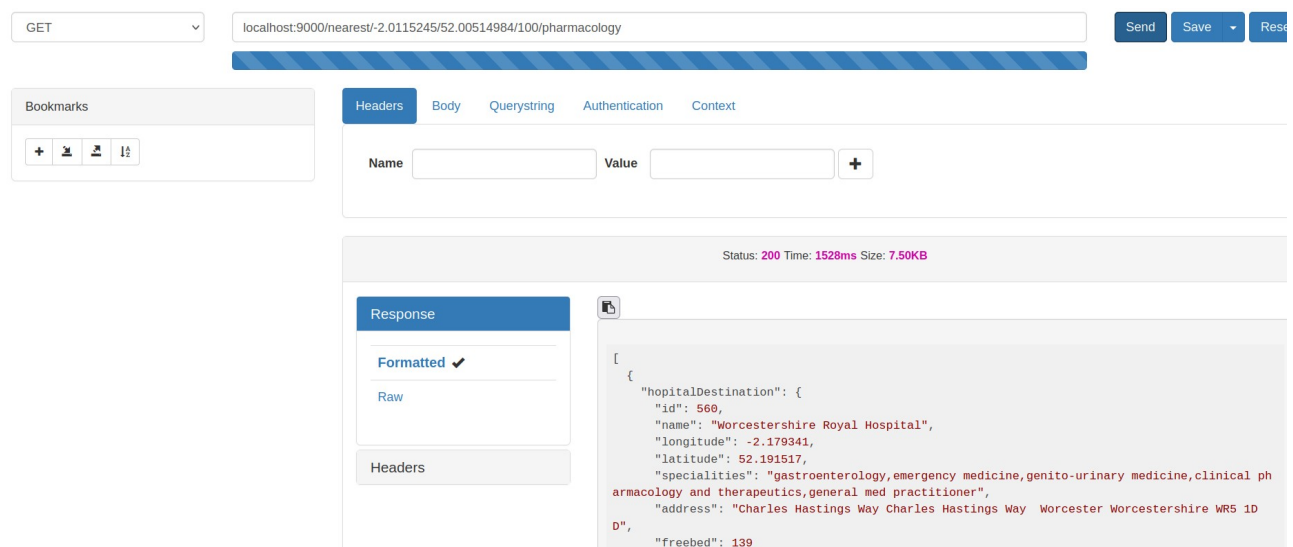
Headers Body Querystring Authentication Context

Name Value +

Status: 200 Time: 31ms Size: 0.30KB

Response: Formatted ✓ Raw

```
{
  "id": 225,
  "name": "Alder Hey Children's NHS Foundation Trust",
  "longitude": -2.8977315,
  "latitude": 53.419384,
  "specialities": "oral medicine,oral medicine,prosthodontics,acute internal medicine,endocrinology and diabetes mellitus",
  "address": "Alder Hey Children's Hospital Eaton Road Liverpool L12 2AP",
  "freedbed": 80
}
```



GET Send Save Reset

Bookmarks: + -

Headers Body Querystring Authentication Context

Name Value +

Status: 200 Time: 1528ms Size: 7.50KB

Response: Formatted ✓ Raw

```
[
  {
    "hospitalDestination": {
      "id": 560,
      "name": "Worcestershire Royal Hospital",
      "longitude": -2.179341,
      "latitude": 52.191517,
      "specialities": "gastroenterology,emergency medicine,genito-urinary medicine,clinical pharmacology and therapeutics,general med practitioner",
      "address": "Charles Hastings Way Charles Hastings Way Worcester Worcestershire WR5 1D",
      "freedbed": 139
    }
  }
]
```

Stress tests

Des tests de stress ont été mis en place en local avec l'outil JMeter.

Plusieurs scénarios ont été configurés, pour permettre de vérifier les contraintes de temps d'exécution, et ce pour vérifier entre autres le principe suivant : « *que nous obtenons un temps de*

réponse de moins de 200 millisecondes avec une charge de travail allant jusqu'à 800 requêtes par seconde, par instance de service »

Exemple pour le paramétrage suivant :

Propriétés du groupe d'unités	
Nombre d'unités (utilisateurs) :	800
Durée de montée en charge (en secondes) :	1
Nombre d'itérations :	<input type="checkbox"/> Infini 1

Libellé	# Echantillons	Moyenne	Médiane	90% centile	95% centile	99% centile	Min	Max	% Erreur	Débit	Ko/sec
GET Freebad R...	800	71	18	154	200	1032	1	1041	0,00%	503,5/sec	719,8
TOTAL	800	71	18	154	200	1032	1	1041	0,00%	503,5/sec	719,8

Les résultats nous montrent une tenue de la charge, en local, avec un respect de la réponse de 200 millisecondes ou moins pour une charge de 800 requêtes par seconde.

Cependant, l'appel effectué ne fait pas intervenir l'API externe. En effet, le service proposé dans la PoC faisant appel à un prestataire externe pour la sélection de l'hôpital le plus proche, un appel vers cet API peut provoquer un dépassement du temps de réponse. Celui-ci pouvant être amélioré en fonction de l'offre choisit.