

2022

Plan de Test

POC Emergency Responder Subsystem



Magalie Morteau
Architecte Logiciel
MedHead
20/01/2022

Table of contents

| | | |
|-------|---------------------------------------|----|
| 1 | Objectif du document..... | 2 |
| 2 | Objectifs global..... | 3 |
| 3 | Périmètre..... | 4 |
| 4 | Caractéristiques à tester..... | 6 |
| 5 | Environnement et outils de test | 7 |
| 6 | Cas de test..... | 8 |
| 6.1 | Tests unitaires | 8 |
| 6.2 | Tests d'intégration..... | 9 |
| 6.3 | Tests d'acceptation..... | 10 |
| 6.4 | Tests de performance..... | 16 |
| 6.5 | Tests de qualité..... | 18 |
| 7 | Annexe..... | 19 |
| 7.1 | Résultats des tests..... | 19 |
| 7.1.1 | Environnement de DEV | 19 |
| 7.1.2 | Environnement de TEST..... | 24 |
| 7.2 | Exemples d'erreur..... | 28 |
| 7.2.1 | Sous GitHub | 28 |
| 7.2.2 | Sous SonarCloud..... | 30 |

1 Objectif du document

Ce document constitue le plan de test réalisé dans le cadre du POC pour la gestion du système d'intervention d'urgence en temps réel, ERS (Emergency Responder Subsystem).

Ce plan de test est basé sur le document de « Stratégie de test » détaillant le plan d'action pour mener les tests.

Il détaille le périmètre, les phases, les cas de test, l'environnement, les outils, les jeux de données et les résultats d'exécution des différents tests.

Plans de test comme outils de communication des exigences

- Les livrables avec des plans de test auto-documentés sont préférables aux plans de test documentés en externe.
- La preuve de concept doit comporter des plans de test décrivant comment le produit doit se comporter.
- Les plans de test doivent utiliser des scénarios **BDD** (cf. Principe C3) pour décrire les critères d'acceptation métier qui sont dans la portée.
- Les plans de test doivent utiliser le langage commun de l'entreprise et être compréhensibles par les partenaires techniques et non techniques.

Pour rédiger les cas de test **BDD** pour un Use Case on utilisera la syntaxe du modèle Gherkin avec la formule Given-When-Then.

The **Given** scenario + **When** an action takes place + **Then** this should be the outcome.

BDD : Business-Driven-Development

Ce plan de test pourra être utilisé pour la réalisation d'autres preuves de concept.

2 Objectifs global

L'objectif global à atteindre pour les tests manuels et les tests d'automatisation pour ce POC de demande d'intervention pour les urgences médicales est la réussite de l'exécution des tests. Permettant de vérifier que le sous-système ERS de la plateforme médical de MedHead sera hautement optimisé et que ce système aura des niveaux élevés de tolérance aux pannes.

L'objectif des tests est de trouver autant de défauts logiciels que possible ; s'assurer que le logiciel testé est exempt de bogues avant sa publication afin de rassurer les parties prenantes du consortium.

S'assurer que les bugs/problèmes seront identifiés et corrigés avant chaque mise en production mais également de façon précoce avec une approche en TDD.

S'assurer que l'application sous test est conforme aux exigences fonctionnelles et non fonctionnelles.

S'assurer que l'application sous test répond aux spécifications de qualité définies dans les principes d'architecture et réponds aux normes spécifiques au domaine médicale.

S'assurer que le service fonctionnera correctement, même pendant les périodes de pic d'activité, qu'il sera en capacité de fournir une allocation de lits en temps opportun.

S'assurer que les temps de réponse, l'évolutivité, la tolérance aux pannes des systèmes hospitaliers auxiliaires et la résilience sous charge.

3 Périmètre

Pour ce POC permettant d'apporter une réponse rapide aux craintes des parties prenantes, nous allons restreindre le périmètre des tests au strict nécessaire. Ci-dessous la liste du périmètre de notre plan de test et celle des éléments hors périmètre.

Dans le périmètre :

Les fonctionnalités à tester :

- Fournir en temps réel les coordonnées de géo localisation et les consignes de l'hôpital le plus proche.
- Fournir l'historique du journal d'intervention des urgences médicales.
- Fournir le détail d'une intervention en urgence médicale.
- Fournir la liste des hôpitaux
- Retourner le détail d'un hôpital
- Fournir la liste des lits disponibles par spécialisation et hôpital
- Fournir la liste des lits disponibles par hôpital pour une spécialisation donnée.
- Calculer la distance entre deux points données.

Les composants à tester :

- Emergency : pour la gestion des urgences médicales
- HospitalPathology : pour la gestion des spécialisation par hôpital
- Hospital : pour la gestion des hôpitaux
- DistanceUtils : pour le calcul des distances

Les types de composants Java à tester :

- Tous les Controller
- Les principaux Services

Les contraintes techniques à tester :

Temps de réponse pour une demande d'urgence médicale isolée.

Jeux de tests

- Jeux de données restreint dans la zone 24 soit la région « Centre Val de Loire » pour laquelle on a récupéré des données réelles de la liste des hôpitaux avec leur adresse et les coordonnées de géo localisation (latitude, longitude).
- Liste des pathologies référencés selon les « Données de référence sur les spécialités NHS ».
- Liste des services médicaux par hôpital est fictive afin de tester les cas de figures les plus pertinents.
- La liste des patients (nom, prénom, âge, genre) est fictive.
- Les adresses et la géo localisation correspondent à des adresses réelles permettant d'effectuer des tests pertinents. Nous avons choisi des adresses d'établissement publics (hôpitaux, écoles, mairie, ...) afin de garantir la confidentialité des données.

Les environnements de test :

Environnement de développement avec une base de données en mémoire H2.

Environnement de test avec une base de données PostgreSQL.

Hors périmètre :

Les fonctionnalités:

La récupération des données de géo localisation du lieu d'intervention en fonction de son adresse, retournant la latitude et la longitude.

L'initialisation de la demande d'intervention pour une urgence médicale. On considère que celle-ci a déjà été effectuée en amont, lors de l'appel l'intervenant du service de régulation collecte les données du patient (nom, prénom, âge, adresse, géo localisation).

Les composants:

Dans le cadre de ce POC on ne testera pas le composant Java « Patient ». En effet, les données du patient étant directement fournies lors des appels des services.

Les types de composants Java:

Dans le cadre de ce POC on ne testera pas les composants Java de type Model et Repository.

Les contraintes techniques:

Les tests ne porteront pas sur les contraintes techniques de type sécurité, tests de montée en charge et tests de stress étant donné que les éléments pour l'architecture cible n'ont pour l'instant pas été définis et/ou implémentés (base de données cibles, serveur distribué, espace dans un Cloud privé, API Gateway ...).

Jeux de données :

- Pas de jeux de données sur l'historique des patients
- Pas de jeux de données sur le planning du personnel médical

Les environnements:

Dans le cadre de cette première version du POC ERS, on ne dispose pas d'environnement de recette, ni d'interface utilisateur permettant de tester notre API de bout en bout, ni d'effectuer les tests de performance, de charge et de stress et de sécurité.

4 Caractéristiques à tester

Liste des fonctionnalités et les composants responsables de son implémentation.

| ID | Fonctionnalités | |
|-----|--|---|
| F1 | Retourne en temps réel les coordonnées de géo localisation et les consignes de l'hôpital le plus proche. | E |
| F2 | Retourne l'historique du journal d'intervention des urgences médicales. | E |
| F3 | Retourne le détail d'une intervention en urgence médicale. | E |
| F4 | Retourne la liste des hôpitaux de la zone d'intervention ayant des lits disponibles dans un service demandé. | B |
| F5 | Réserve un lit dans un hôpital et un service. | B |
| F6 | Retourne la liste des lits disponibles par spécialisation et hôpital. | B |
| F7 | Retourne la liste des lits disponibles par hôpital pour une spécialisation donnée. | B |
| F8 | Retourne la distance en kms entre 2 points géo localisés avec leurs latitudes et longitudes. | D |
| F9 | Retourne la liste des hôpitaux | H |
| F10 | Retourne le détail d'un hôpital | H |

E : **E**mergency **B** : Hospital Pathology with **B**ed Availability

H : **H**ospital **D** : **D**istance

Liste des contraintes techniques.

| ID | Contraintes | |
|----|--|---------------|
| C1 | Retourne en temps réel les coordonnées de géo localisation et les consignes de l'hôpital le plus proche en moins de 200ms. | Performance |
| C2 | Réutilisabilité du code et des techniques | Documentation |
| C3 | Découpage en micro service permettant de tester individuellement chaque service de l'architecture. | Architecture |
| C4 | API Restful | Architecture |
| C5 | Intégration et déploiement continu en moins de 5 minutes | Architecture |

5 Environnement et outils de test

Les tests peuvent être lancés manuellement ou de façon programmée (automatique).

| Environnement de test | Type de test | Outils | Responsable |
|-----------------------|---|--|--------------|
| DEV | Tests unitaires Tests d'intégration Tests de qualité (code) | JUnit / Eclipse ou Maven Workflow/Github Actions Jacoco / SonarCloud | Développeurs |
| TEST | Tests d'intégration | Collection / Postman | Développeurs |
| RECETTE | Test d'acceptation de bout en bout | Cucumber ; Selenium | Testeurs, QA |

Sources de données

En fonction de l'environnement de test, la source de données n'est pas la même.

H2 en mémoire pour la DEV et PostgreSQL pour les autres environnements.

Les jeux de données

H2 : initialisés dans le fichier [data.sql](#)

PostgreSQL : source de données *erspoc* plus de détail dans le fichier [properties](#)

Ressources virtuelles

Pour les tests lancés automatiquement dans le pipeline CI/CD sous Github Actions.

Serveur virtuel de test Linux : `ubuntu-20.04`

6 Cas de test

6.1 Tests unitaires

Pour notre API les tests unitaires vont correspondre à tester les composants individuellement soient les classes Java.

Outils : JUnit/Eclipse ou JUnit/Maven.

Exemple de script de test unitaire avec: **DistanceUtilsTest**

```
@SpringBootTest
@Tag("UnitTest")
class DistanceUtilsTest {

    @Test
    @DisplayName("Given: les latitudes et longitudes des points A et B "
        + "When: calcule la distance en kms entre les points A et B selon la formule Haversine "
        + "Then: est de 11kms")
    void testCalcDistance() throws Exception {
        // Given
        final double latA = 47.8106061;
        final double longA = 0.9128109;
        final double latB = 47.7941772;
        final double longB = 1.0630365;

        // When
        long result = (long) DistanceUtils.calcDistance(latA, longA, latB,
            longB);

        // Then
        assertThat(result).isEqualTo(11);
    }
}
```

Exemple script de de test unitaire avec : **EmergencyControllerTest**

```
// When
mockMvc.(post("/emergencies")
    .(MediaType.APPLICATION_JSON).(jsonBody))
// Then
.andExpect(status().isOk()) ;
```

6.2 Tests d'intégration

Pour notre API les tests d'intégration vont correspondre à tester l'intégration de toutes les classes Java entre elles ainsi que les échanges avec les composants externes soit pour notre POC avec la base de données PostgreSQL (CRUD).

- JUnit/(Eclipse ou Maven) + H2 (CRUD)

Exemple de script de test d'intégration avec **EmergencyControllerTest**

```
@Test
@DisplayName("Given: id=1 dans le journal d'intervention des urgences "
    + "When: consulte le log "
    + "Then: concerne la patiente Magalie")
void testGetEmergency() throws Exception {
    // Given : id = 1
    // When
    mockMvc.perform(get("/emergencies/{id}", 1))
    // Then
    .andExpect(status().isOk())
    .andExpect(jsonPath("$.patientFirstName").value("Magalie"));
}
```

- Postman / Collections / PostgreSQL (CRUD)

Exemple de script de test d'intégration avec Postman pour de la requête :

GET /emergencies/{id} qui retourne le détail d'une intervention en urgence médicale. id = 1

The screenshot shows the Postman interface for a GET request to `http://localhost:9001/emergencies/1`. The 'Tests' tab is active, displaying a JavaScript test script. Below the script, the 'Test Results (3/3)' tab is active, showing three passed tests.

GET `http://localhost:9001/emergencies/1`

Params Authorization Headers (6) Body Pre-request Script **Tests** Settings

```
1 pm.test("Status code is 200", function () {
2   pm.response.to.have.status(200);
3 });
4 pm.test("Content-Type header is application/json", () => {
5   pm.expect(pm.response.headers.get('Content-Type')).to.eql('application/json');
6 });
7 pm.test("Found patient Magalie when Emergency log id = 1", function () {
8   var jsonData = pm.response.json();
9   pm.expect(jsonData.patientFirstName).to.include("Magalie");
10 });
11
```

Body Cookies Headers (5) **Test Results (3/3)**

All Passed Skipped Failed

- PASS** Status code is 200
- PASS** Content-Type header is application/json
- PASS** Found patient Magalie when Emergency log id = 1

6.3 Tests d'acceptation

Pour notre API nous ne disposons pas d'interface utilisateur ni d'environnement de recette (architecture cible), nous ne pouvons donc pas réaliser les tests E2E. Nous allons cependant tester la principale fonctionnalité de notre API dans le cadre de notre POC. Soit :

| | |
|-------------------|---|
| F1 | Retourne en temps réel les coordonnées de géo localisation et les consignes de l'hôpital le plus proche de la zone d'intervention pour lequel une réservation est également faite dans le service approprié si disponible. |
| User Story | En tant qu'intervenant au service de traitement des urgences médicales, je veux les coordonnées de géo localisation de l'hôpital le plus proche du lieu où se trouve un patient et ayant de la place dans un service spécialisé. |

Liste des scénarios BDD éligibles à l'automatisation pour la fonctionnalité F1

| | |
|---------------------|---|
| Scénario S11 | Retourne l'hôpital le plus proche du patient localisé à Lunay souffrant d'une pathologie cardiaque, soit l'hôpital de Blois avec une réservation dans le service « Cardiologie». |
| GWT | GIVEN un patient ayant un probleme cardiaque, habitant à Lunay dans la région Centre Val de Loire AND l'hôpital le plus proche ayant des lits disponibles en Cardiologie est l'hôpital de Blois WHEN j'enclenche le service ERS, pour une place en cardio le plus proche de Lunay THEN je récupère en moins de 200ms, les coordonnées de géo localisation de l'hôpital Blois AND les consignes pour le trajet et pour le patient AND le système a réservé un lit pour ce patient dans le service « Cardiologie » de l'hôpital de Blois |

| | |
|---------------------|--|
| Scénario S12 | Retourne l'hôpital le plus proche du patient localisé à Lunay souffrant d'une pathologie respiratoire, soit l'hôpital de Vendôme avec une réservation dans le service « Médecine d'urgence » car aucun autre hôpital dans la zone d'intervention n'a de lits disponibles en médecine respiratoire. |
| GWT | GIVEN un patient ayant un probleme respiratoire, habitant à Lunay dans la région Centre Val de Loire AND l'hôpital le plus proche ayant des lits disponibles en « Médecine d'urgence » est l'hôpital de Vendôme car il n'y a plus de place en « Médecine respiratoire » dans les hôpitaux de la région. WHEN j'enclenche le service ERS, pour une place en « Médecine respiratoire » le plus proche de Lunay THEN je récupère en moins de 200ms, les coordonnées de géo localisation de l'hôpital de Vendôme AND les consignes pour le trajet et pour le patient AND le système a réservé un lit pour ce patient dans le service par défaut soit « Médecine d'urgence » de l'hôpital de Vendôme |

| | |
|--------------|---|
| Scénario S13 | Retourne l'hôpital le plus proche du patient localisé à Saint-Calais en Pays de la Loire, souffrant d'une urgence respiratoire, mais aucun lit n'est disponible dans tous les hôpitaux de la zone d'intervention, alors un message d'alerte est retourné. |
| GWT | <p>GIVEN un patient ayant un probleme respiratoire, habitant à Saint-Calais en Pays de la Loire</p> <p>AND aucun lit n'est disponible dans tous les hôpitaux de la zone d'intervention</p> <p>WHEN j'enclenche le service ERS, pour une place en « Médecine respiratoire » le plus proche de Saint-Calais</p> <p>THEN je récupère en moins de 200ms, un message d'alerte indiquant qu'aucun lit n'a été trouvé</p> <p>AND m'invitant à faire une demande dans une zone voisine</p> |

- JUnit/(Eclipse ou Maven) + H2 (CRUD)

Exemple de script de test avec le scénario S11 de la fonctionnalité F1 avec

EmergencyControllerTest

```
@Test
@DisplayName("Given: urgence cardiologie à Lunay When: ERS Then: dispo en Cardio à Blois")
void test1CreateEmergency() throws Exception {

    // Given
    final String jsonBody = "{" + "\"idZone\":24," + "\"idResponder\":1,"
        + "\"idPatient\":1," + "\"patientFirstName\":\"Magalie\","
        + "\"patientLastName\":\"Mortreau\","
        + "\"patientGender\":\"F\"," + "\"patientAge\":50,"
        + "\"patientAddress\":\"20 rue du Progrès 41360 Lunay \","
        + "\"patientLatitude\":47.8106061,"
        + "\"patientLongitude\": 0.9128109," + "\"idPathology\":70,"
        + "\"dtStart\":null}";

    // When
    mockMvc.perform(post("/emergencies")
        .contentType(MediaType.APPLICATION_JSON).content(jsonBody))
        // Then
        .andExpect(status().isOk())
        .andExpect(jsonPath("$.hospitalName").value(Matchers
            .containsStringIgnoringCase("CH BLOIS SIMONE VEIL")));
}
```

Exemple de script de test avec le scénario S12 de la fonctionnalité F1, avec:

EmergencyControllerTest

```
@Test
@DisplayName("Given: urgence respiratoire à Lunay When: ERS Then: dispo en Urgence à Vendôme")
void test2CreateEmergency() throws Exception {

    // Given
    final String jsonBody = "{" + "\"idZone\":24," + "\"idResponder\":1,"
        + "\"idPatient\":2," + "\"patientFirstName\":\"Maurice\","
        + "\"patientLastName\":\"Moss\"," + "\"patientGender\":\"M\","
        + "\"patientAge\":50,"
        + "\"patientAddress\":\"20 rue du Progrès 41360 Lunay \","
        + "\"patientLatitude\":47.8106061,"
        + "\"patientLongitude\":0.9128109," + "\"idPathology\":41,"
        + "\"dtStart\":null}";

    // When
    mockMvc.perform(post("/emergencies")
        .contentType(MediaType.APPLICATION_JSON_VALUE)
        .content(jsonBody))

        // Then
        .andExpect(status().isOk())
        .andExpect(jsonPath("$.hospitalName")
            .value("CH VENDOME - MONTOIRE"))
        .andExpect(jsonPath("$.hospitalServiceName")
            .value(Matchers.containsStringIgnoringCase("urgence")));
}
```

Exemple de script de test avec le scénario S13 de la fonctionnalité F1, avec:

EmergencyControllerTest

```
@Test
@DisplayName("Given: urgence respiratoire à Saint-Calais "
    + "When: ERS Then: ALERTE aucun hôpital dispo dans la région")
void test3CreateEmergency() throws Exception {

    // Given
    final String jsonBody = "{" + "\"idZone\":52," + "\"idResponder\":2,"
        + "\"idPatient\":3," + "\"patientFirstName\":\"Roy\","
        + "\"patientLastName\":\"Trenneman\","
        + "\"patientGender\":\"M\"," + "\"patientAge\":36,"
        + "\"patientAddress\":\"Rue Amédée-Savidan 72120 Saint-Calais\","
        + "\"patientLatitude\":47.9211271,"
        + "\"patientLongitude\":0.7429723," + "\"idPathology\":41,"
        + "\"dtStart\":null}";

    // When
    mockMvc.perform(post("/emergencies")
        .contentType(MediaType.APPLICATION_JSON).content(jsonBody))

        // Then
        .andExpect(status().isOk())
        .andExpect(jsonPath("$.hospitalName").isEmpty())
        .andExpect(jsonPath("$.instructions")
            .value(Matchers.containsStringIgnoringCase("Alerte")));
}
```

- Postman / Collections / PostgreSQL (CRUD)

Exemple de cas de test avec le scénario S11 de la fonctionnalité F1, avec Postman pour la requête: POST /emergencies

"Given: urgence cardiologie à Lunay When: ERS Then: dispo en Cardio à Blois"

The screenshot displays the Postman interface for a POST request to `http://localhost:9001/emergencies`. The request body is a JSON object representing an emergency case. Below the request, the 'Tests' tab is active, showing a script with four assertions. The 'Test Results' section at the bottom indicates that all four tests passed successfully.

Request Details:

- Method: POST
- URL: `http://localhost:9001/emergencies`
- Body Type: JSON

Request Body (JSON):

```

{
  "idZone": 24,
  "idResponder": 1,
  "idPatient": 2,
  "patientFirstName": "Maurice",
  "patientLastName": "Moss",
  "patientGender": "M",
  "patientAge": 50,
  "patientAddress": "20 rue du Progrès 41360 Lunay",
  "patientLatitude": 47.8106061,
  "patientLongitude": 0.9128109,
  "idPathology": "70",
  "dtRequest": "2022-02-07"
}

```

Test Script:

```

1 pm.test("Status code is 200", function () {
2   pm.response.to.have.status(200);
3 });
4 pm.test("Content-Type header is application/json", () => {
5   pm.expect(pm.response.headers.get('Content-Type')).to.eql('application/json');
6 });
7 pm.test("Hospital found in Blois", function () {
8   var jsonData = pm.response.json();
9   pm.expect(jsonData.hospitalName).to.include("BLOIS");
10 });
11 pm.test("Response time is less than 200ms", function () {
12   pm.expect(pm.response.responseTime).to.be.below(200);
13 });

```

Test Results (4/4):

- PASS** Status code is 200
- PASS** Content-Type header is application/json
- PASS** Hospital found in Blois
- PASS** Response time is less than 200ms

Exemple de cas de test avec le scénario S12 de la fonctionnalité F1, avec Postman pour la requête: POST /emergencies

"Given: urgence respiratoire à Lunay When: ERS Then: dispo en Urgence à Vendôme"

POC API Emergency Nearest Hospitals / ERS Lunay Respi

POST http://localhost:9001/emergencies

Params Authorization Headers (9) Body Pre-request Script T

none form-data x-www-form-urlencoded raw binary

```
1 {
2   ....
3   ...."idZone": 24,
4   ...."idResponder": 1,
5   ...."idPatient": 2,
6   ...."patientFirstName": "Maurice",
7   ...."patientLastName": "Moss",
8   ...."patientGender": "M",
9   ...."patientAge": 50,
10  ...."patientAddress": "20 rue du Progrès 41360 Lunay",
11  ...."patientLatitude": 47.8106061,
12  ...."patientLongitude": 0.9128109,
13  ...."idPathology": 41,
14  ...."dtRequest": "2022-02-07"
15 }
16
17 }
```

```
1 pm.test("Status code is 200", function () {
2   pm.response.to.have.status(200);
3 });
4 pm.test("Content-Type header is application/json", () => {
5   pm.expect(pm.response.headers.get('Content-Type')).to.eql('application/json');
6 });
7 pm.test("Hospital found in Vendôme", function () {
8   var jsonData = pm.response.json();
9   pm.expect(jsonData.hospitalName).to.include("VENDOME");
10 });
11 pm.test("Service name is Urgence", function () {
12   var jsonData = pm.response.json();
13   pm.expect(jsonData.hospitalServiceName).to.include("urgence");
14 });
15
16 pm.test("Response time is less than 200ms", function () {
17   pm.expect(pm.response.responseTime).to.be.below(200);
18 });
19 }
```

Body Cookies Headers (5) Test Results (5/5)

All Passed Skipped Failed

PASS Status code is 200

PASS Content-Type header is application/json

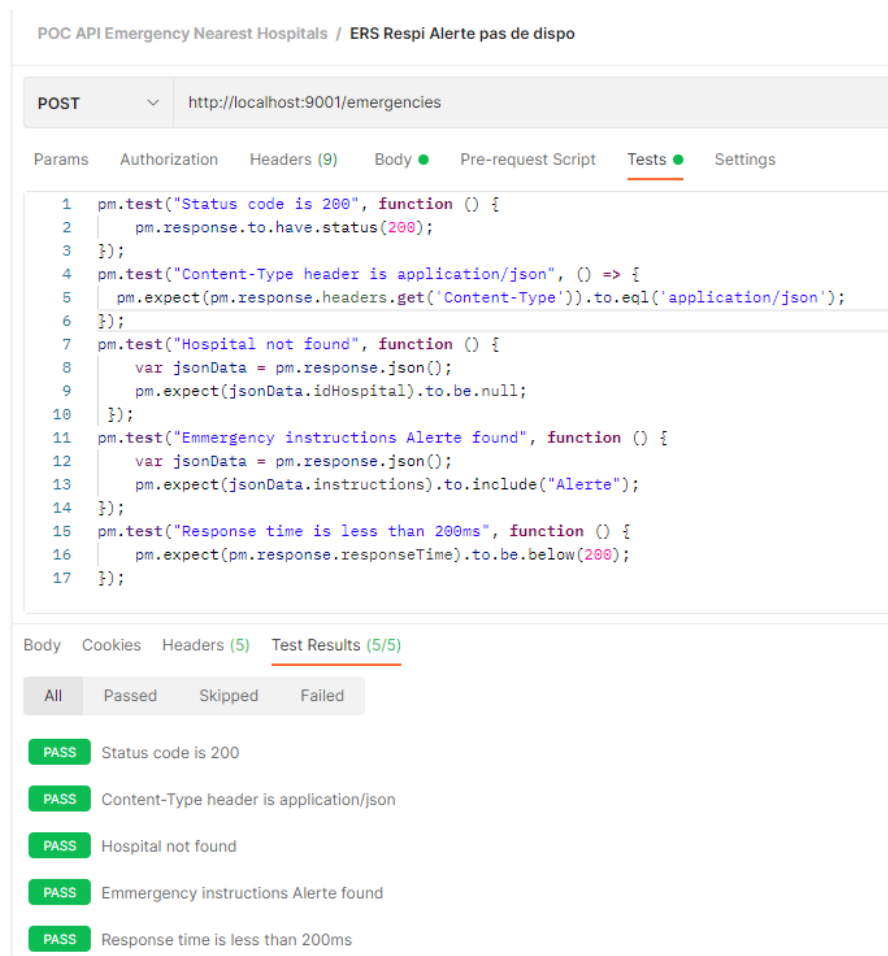
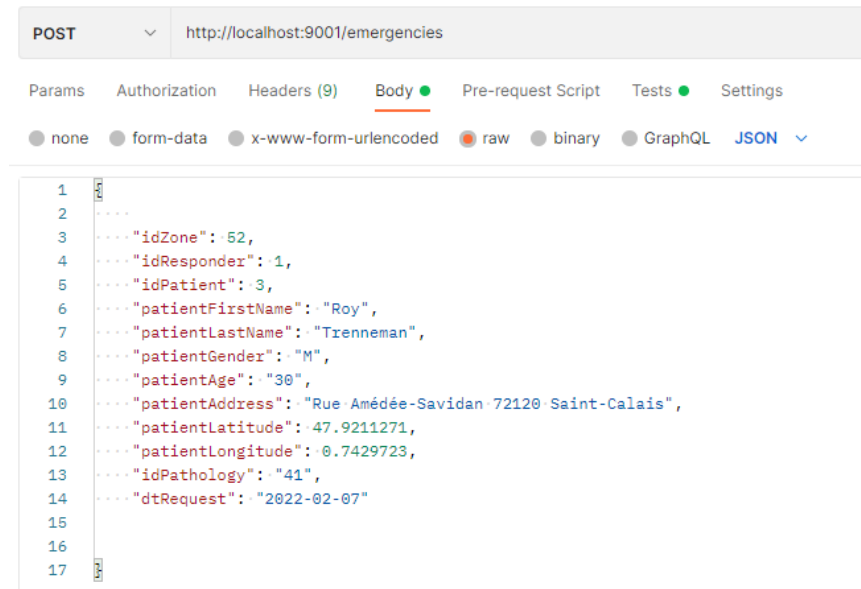
PASS Hospital found in Vendôme

PASS Service name is Urgence

PASS Response time is less than 200ms

Exemple de cas de test avec le scénario S13 de la fonctionnalité F1, avec Postman pour la requête: POST /emergencies

"Given: urgence respiratoire à Saint-Calais When: ERS Then: ALERTE aucun hôpital dispo dans la région"



6.4 Tests de performance

Pour notre API nous ne disposons pas à ce jour d'environnement de recette (architecture cible), nous ne pouvons donc pas réaliser les tests de performance, ni les tests de charge et de stress. Nous allons cependant créer les cas de test pour la performance qu'il suffira de reproduire dans l'environnement cible.

- Postman / Collections / PostgreSQL (CRUD)

Exemple de cas de test de temps d'exécution, avec Postman pour la requête:
POST /emergencies

When 1 user call the API
the average response time should be below 200ms
and no errors should occur

Soit le test suivant en javascript :

```
pm.test("Response time is less than 200ms", function () {  
    pm.expect(pm.response.responseTime).to.be.below(200);  
});
```

Exemple de cas de test de charge, avec Postman pour la requête: POST /emergencies

When 800 user call the API within 1s
the average response time should be below 200ms
and no errors should occur

Soit l'exécution de de la collection suivante, exécutant 32 itérations de 25 appels simultanés de la requête pour demande d'intervention médicale urgente.

✓ POST ERS Lunay Cardio 1

✓ POST ERS Lunay Neuro 1

✓ POST ERS Lunay Pédiatrie 1

✓ POST ERS Lunay Trauma 1

✓ POST ERS Respi Alerte 1

✓ POST ERS Lunay Cardio 2

✓ POST ERS Lunay Neuro 2

✓ POST ERS Lunay Pédiatrie 2

✓ POST ERS Lunay Trauma 2

✓ POST ERS Respi Alerte 2

✓ POST ERS Lunay Cardio 3

✓ POST ERS Lunay Neuro 3

✓ POST ERS Lunay Pédiatrie 3

✓ POST ERS Lunay Trauma 3

✓ POST ERS Respi Alerte 3

✓ POST ERS Lunay Cardio 4

✓ POST ERS Lunay Neuro 4

✓ POST ERS Lunay Pédiatrie 4

✓ POST ERS Lunay Trauma 4

✓ POST ERS Respi Alerte 4

✓ POST ERS Lunay Cardio 5

✓ POST ERS Lunay Neuro 5

✓ POST ERS Lunay Pédiatrie 5

✓ POST ERS Lunay Trauma 5

✓ POST ERS Respi Alerte 5

Iterations32

Delay0ms

Data

Select File

☒ Save responses ⓘ

☐ Keep variable values ⓘ

☐ Run collection without using stored cookies

☐ Save cookies after collection run ⓘ

Run POC API Emergency Loadin...

6.5 Tests de qualité

Pour notre API nous allons audité sa qualité avec les outils : Jacoco / SonarCloud.

Le repository GitHub de notre API est référencé dans SonarCloud qui auditera le code.

L'audit de code est automatiquement déclenché lors d'un Merge Request et d'un Pull Request.

Cette vérification de code fait partie du processus CI/CD définit dans le pipeline maven.yml sous GitHub Actions.

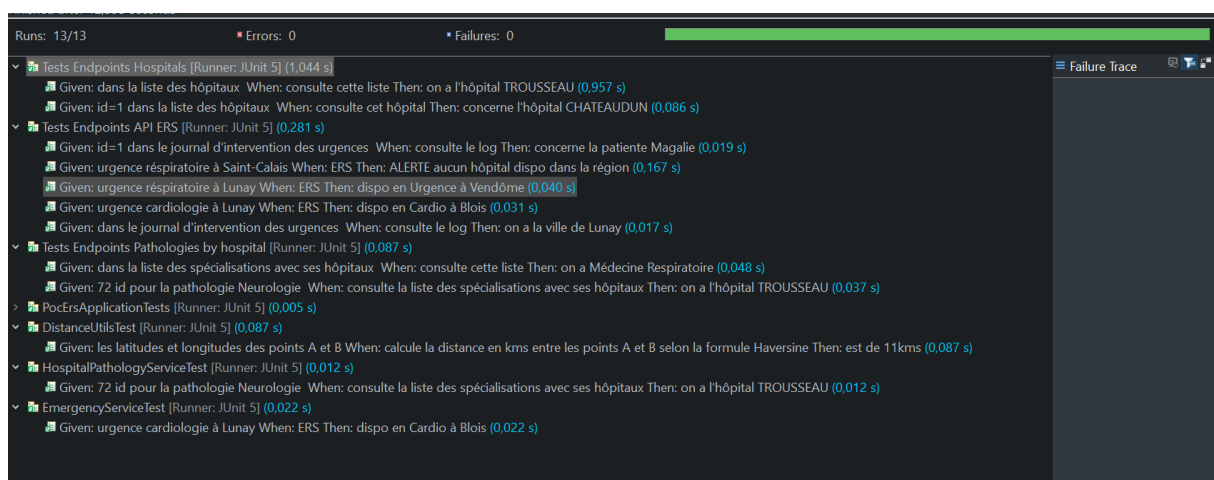
7 Annexe

7.1 Résultats des tests

Dans ce chapitre on listera les rapports d'exécution des test produits pendant l'intégration continue afin de démontrer les comportements livrés (dans le contexte d'un développement BDD).

7.1.1 Environnement de DEV

7.1.1.1 Sous Eclipse



7.1.1.2 Sous Maven

```
2022-02-14 16:51:38.887 INFO 2148 --- [main] c.m.e.c.EmergencyControllerTest : Starting EmergencyControllerTest using Java 17.0
2022-02-14 16:51:38.890 INFO 2148 --- [main] c.m.e.c.EmergencyControllerTest : No active profile set, falling back to default
2022-02-14 16:51:48.709 INFO 2148 --- [main] o.s.b.a.h2.H2ConsoleAutoConfiguration : H2 console available at '/h2-console'. Database available at 'jdbc:h2:mem:202a7d39-fb17-4a40-97b5-c56802dc38db'
2022-02-14 16:51:49.617 INFO 2148 --- [main] c.m.e.c.EmergencyControllerTest : Started EmergencyControllerTest in 11.674 seconds
[INFO] Tests run: 5, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 14.921 s - in com.medhead.ers.controller.EmergencyControllerTest
[INFO] Running com.medhead.ers.controller.HospitalControllerTest
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.046 s - in com.medhead.ers.controller.HospitalControllerTest
[INFO] Running com.medhead.ers.controller.HospitalPathologyControllerTest
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.084 s - in com.medhead.ers.controller.HospitalPathologyControllerTest
[INFO] Running com.medhead.ers.PocErsApplicationTests
:: Spring Boot :: (v2.6.2)

2022-02-14 16:51:51.641 INFO 2148 --- [main] com.medhead.ers.PocErsApplicationTests : Starting PocErsApplicationTests using Java 17.0
2022-02-14 16:51:51.642 INFO 2148 --- [main] com.medhead.ers.PocErsApplicationTests : No active profile set, falling back to default
2022-02-14 16:51:53.188 INFO 2148 --- [main] o.s.b.a.h2.H2ConsoleAutoConfiguration : H2 console available at '/h2-console'. Database available at 'jdbc:h2:mem:202a7d39-fb17-4a40-97b5-c56802dc38db'
2022-02-14 16:51:53.683 INFO 2148 --- [main] com.medhead.ers.PocErsApplicationTests : Started PocErsApplicationTests in 2.095 seconds
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 2.126 s - in com.medhead.ers.PocErsApplicationTests
[INFO] Running com.medhead.ers.service.EmergencyServiceTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.284 s - in com.medhead.ers.service.EmergencyServiceTest
[INFO] Running com.medhead.ers.service.HospitalPathologyServiceTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.019 s - in com.medhead.ers.service.HospitalPathologyServiceTest
[INFO] Running com.medhead.ers.utils.DistanceUtilsTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.012 s - in com.medhead.ers.utils.DistanceUtilsTest
[INFO] Results:
[INFO] Tests run: 13, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] --- jacoco-maven-plugin:0.8.7:report (report) @ ers ---
[INFO] Loading execution data file C:\Users\Mag\git\ers\target\jacoco.exec
[INFO] Analyzed bundle 'ers' with 15 classes
[INFO] BUILD SUCCESS
[INFO] Total time: 24.215 s
[INFO] Finished at: 2022-02-14T16:51:55+01:00
[INFO]
```

7.1.1.3 Sous GitHub

github.com/Mamak2020/poc-api-ers/runs/5197690236?check_suite_focus=true

Summary

Jobs

- Tests
- Build
- SonarCloud analysis
- Deploy

Tests

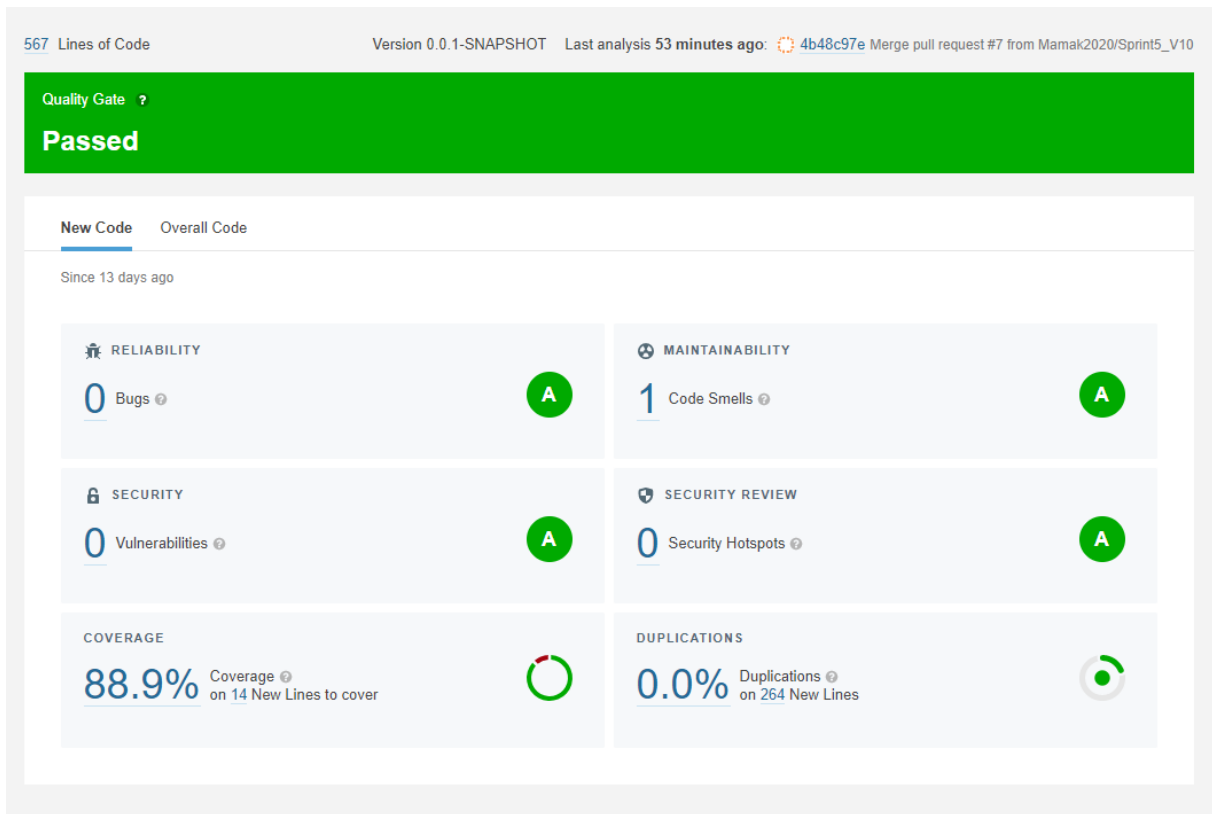
succeeded 6 minutes ago in 22s

Run Tests 17s

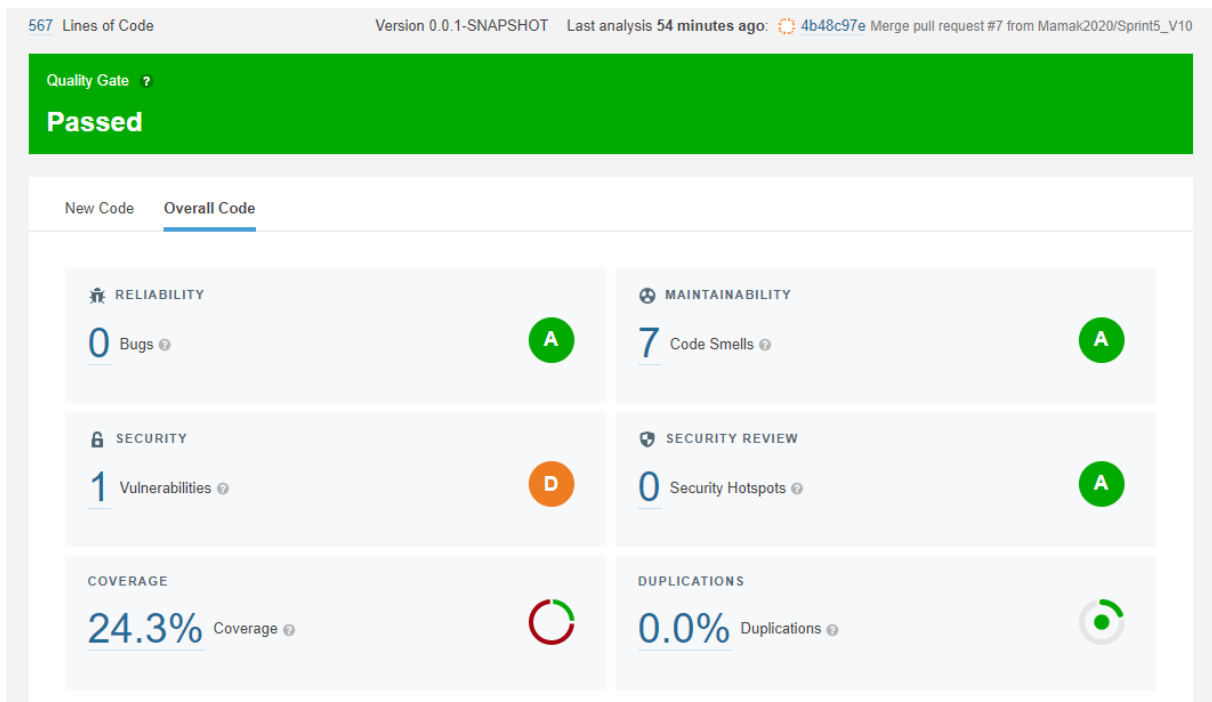
```
83 2022-02-15 09:32:04.499 INFO 1645 --- [main] o.s.b.a.h2.H2ConsoleAutoConfiguration : H2 console available at '/h2-console'. Database available at 'jdbc:h2:mem:202a7d39-fb17-4a40-97b5-c56802dc38db'
84 2022-02-15 09:32:04.719 INFO 1645 --- [main] c.m.e.c.HospitalPathologyControllerTest : Started HospitalPathologyControllerTest in 0.919 seconds (JVM running for 9.512)
85 [INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 1.369 s - in com.medhead.ers.controller.HospitalPathologyControllerTest
86 [INFO] Running com.medhead.ers.controller.EmergencyControllerTest
87 [INFO] Tests run: 5, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.329 s - in com.medhead.ers.controller.EmergencyControllerTest
88 [INFO] Running com.medhead.ers.controller.HospitalControllerTest
89 [INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.047 s - in com.medhead.ers.controller.HospitalControllerTest
90 [INFO] Running com.medhead.ers.service.HospitalPathologyServiceTest
91 [INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.043 s - in com.medhead.ers.service.HospitalPathologyServiceTest
92 [INFO] Running com.medhead.ers.service.EmergencyServiceTest
93 [INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.008 s - in com.medhead.ers.service.EmergencyServiceTest
94 [INFO] Running com.medhead.ers.PocErsApplicationTests
95 [INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.001 s - in com.medhead.ers.PocErsApplicationTests
96 [INFO] Results:
97 [INFO] Tests run: 13, Failures: 0, Errors: 0, Skipped: 0
98 [INFO]
99 [INFO] --- jacoco-maven-plugin:0.8.7:report (report) @ ers ---
100 [INFO] Loading execution data file /home/runner/work/poc-api-ers/poc-api-ers/target/jacoco.exec
101 [INFO] Analyzed bundle 'ers' with 15 classes
102 [INFO] BUILD SUCCESS
103 [INFO] Total time: 16.046 s
104 [INFO] Finished at: 2022-02-15T09:32:06Z
105 [INFO]
```

7.1.1.4 Sous SonarCloud

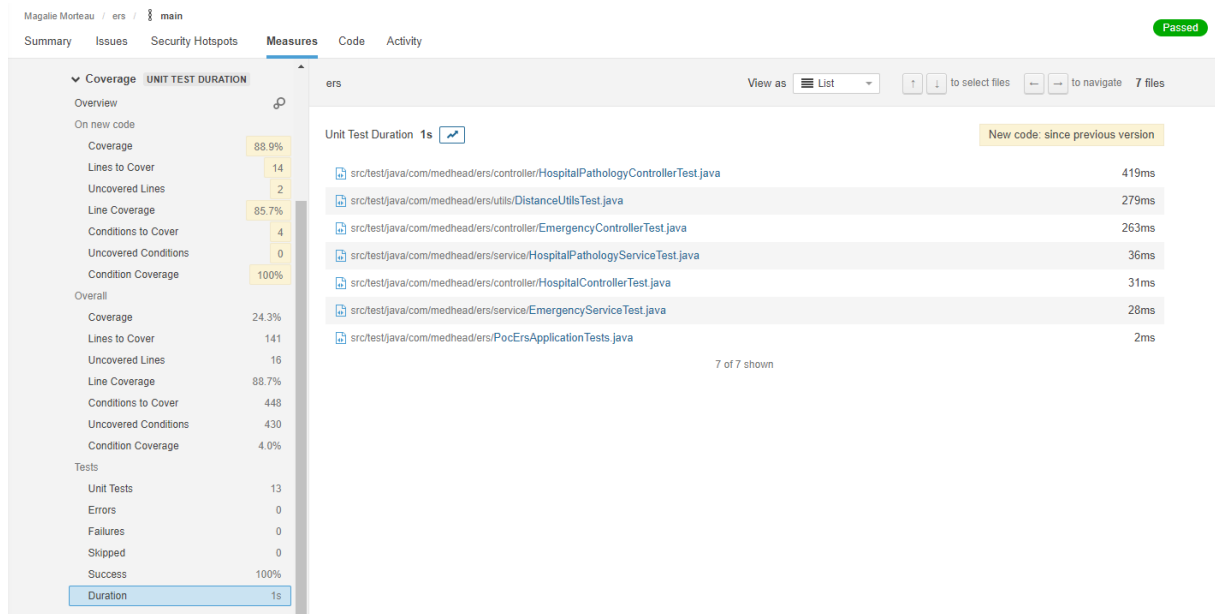
La qualité du code de la dernière mise à jour excellente.



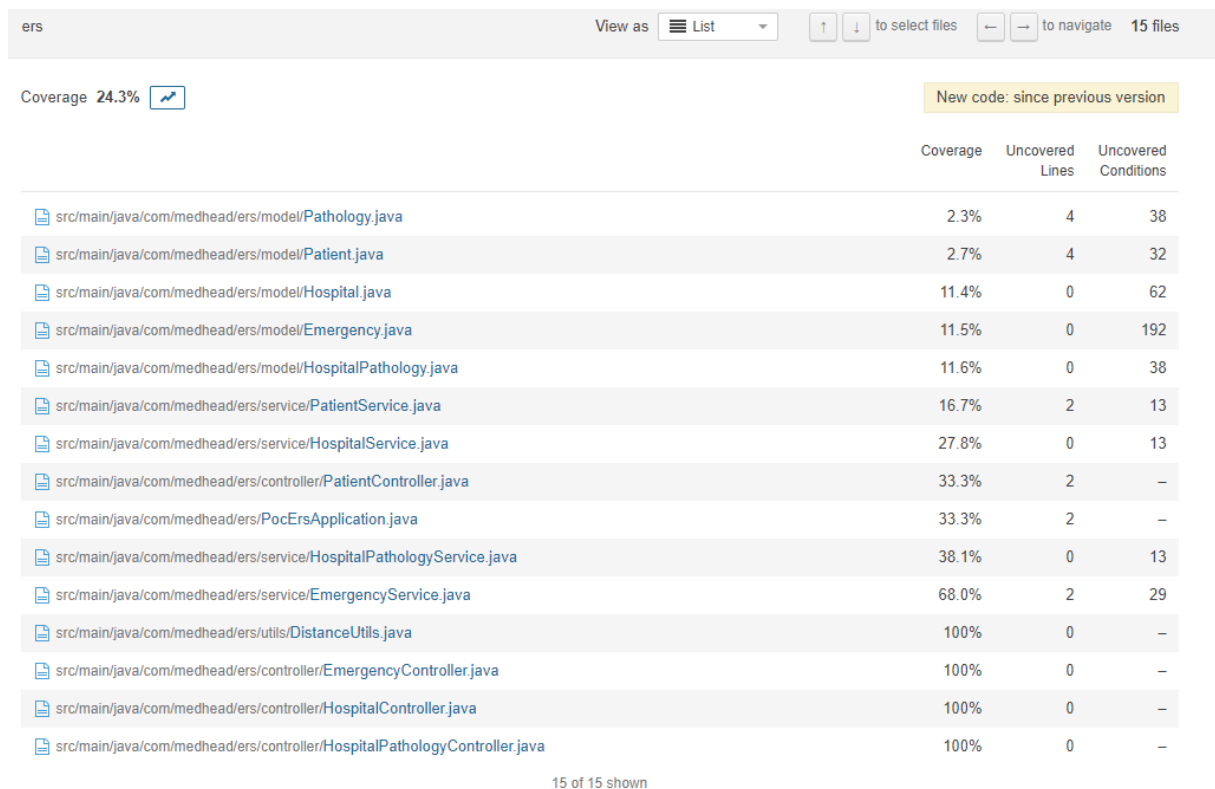
La qualité du code dans son ensemble est excellente.



Durée de l'ensemble des tests est 1 seconde : OK



La couverture des tests est de 100% sur nos principaux composants de notre API : OK



Remarque on a une vulnérabilité qu'il faudra corriger en remplaçant l'entité Emergency par un DTO, comme cela a été fait pour HospitalPathology

Project Overview

> Reliability

> Security SECURITY RATING

Overview

On new code

Vulnerabilities 0

Rating A

Remediation Effort 0

Overall

Vulnerabilities 1

Rating D

Remediation Effort 10min

> Security Review

> Maintainability

> Coverage

> Duplications

> Size

> Complexity

> Issues

ers / src / main / ... / com / medhead / ers / controller / EmergencyController.java

```
10 import org.springframework.web.bind.annotation.RestController;
11
12 import com.medhead.ers.model.Emergency;
13 import com.medhead.ers.service.EmergencyService;
14
15 @RestController
16 public class EmergencyController {
17     @Autowired
18     private EmergencyService emergencyService;
19
20     @GetMapping("/emergencies/{id}")
21     public Optional<Emergency> getEmergency(@PathVariable("id") final Long id) {
22         return emergencyService.getEmergency(id);
23     }
24
25     @GetMapping("/emergencies")
26     public Iterable<Emergency> getEmergencies() {
27         return emergencyService.getEmergencies();
28     }
29
30     @PostMapping("/emergencies")
31     public Emergency createEmergencyLog(@RequestBody Emergency emergency) {
32
33         return emergencyService.requestMedicalEmergency(emergency);
34     }
35 }
```

Replace this persistent entity with a simple POJO or DTO object. Why is this an issue? 13 days ago L31

Vulnerability

Critical

Open

Magalie Morneau

10min effort

No tags

7.1.2 Environnement de TEST

7.1.2.1 Tests manuels avec Postman

Résultat du cas de test avec le scénario S11 de la fonctionnalité F1, avec Postman pour la requête: POST /emergencies

"Given: urgence cardiologie à Lunay When: ERS Then: dispo en Cardio à Blois"

The screenshot shows the Postman interface for a POST request to `http://localhost:9001/emergencies`. The request body is a JSON object with the following fields:

```
{
  "idZone": 24,
  "idRespondez": 1,
  "idPatient": 2,
  "patientFirstName": "Maurice",
  "patientLastName": "Moss",
  "patientGender": "M",
  "patientAge": 50,
  "patientAddress": "20 rue du Progrès 41360 Lunay",
  "patientLatitude": 47.8106061,
  "patientLongitude": 0.9128109,
  "idPathology": "70",
  "dtRequest": "2022-02-07"
}
```

The response is a JSON object with the following fields:

```
{
  "id": 26712,
  "idZone": 24,
  "idRespondez": 1,
  "idPatient": 2,
  "patientFirstName": "Maurice",
  "patientLastName": "Moss",
  "patientGender": "M",
  "patientAge": 50,
  "patientAddress": "20 rue du Progrès 41360 Lunay",
  "patientLatitude": 47.8106061,
  "patientLongitude": 0.9128109,
  "idPathology": "70",
  "idHospital": 3,
  "hospitalName": "CH BLOIS SIMONE VEIL",
  "hospitalAddress": "Mail Pierre Charlot 41000 BLOIS",
  "hospitalLongitude": 1.3435976,
  "hospitalLatitude": 47.6018423,
  "hospitalServiceName": "Chirurgie cardiothoracique",
  "idHospitalService": 4,
  "distance": 39,
  "instructions": "Réservation: CH BLOIS SIMONE VEIL à Mail Pierre Charlot 41000 BLOIS - Latitude : 47.6018423 - Longitude : 1.3435976 - Distance: 39 kms - Service: Chirurgie cardiothoracique",
  "duration": 10,
  "dtRequest": "2022-02-07T00:00:00.000+00:00",
  "dtResponse": null
}
```

The status is 200 OK, Time: 25 ms, Size: 954 B.

7.1.2.2 Tests de tous les EndPoints API sous Postman

Résultat des tests avec la collection « POC API Emergency » permettant de tester les différents appels à l'API via des requêtes http (Get, Post).

| POC API Emergency No Environment, just now | | | |
|--|---|-------|---|
| RUN SUMMARY | | | |
| ▼ GET | http://localhost:9001/emergencies | 2 0 | 1 |
| | Pass Status code is 200 | | |
| | Pass Content-Type header is application/json | | |
| ▼ GET | http://localhost:9001/emergencies/1 | 3 0 | |
| | Pass Status code is 200 | | |
| | Pass Content-Type header is application/json | | |
| | Pass Found patient Magalie when Emmergency log id = 1 | | |
| ▼ POST | http://localhost:9001/emergencies | 4 0 | |
| | Pass Status code is 200 | | |
| | Pass Content-Type header is application/json | | |
| | Pass Hospital found in Blois | | |
| | Pass Response time is less than 200ms | | |
| ▼ GET | http://localhost:9001/pathologies | 2 0 | |
| | Pass Status code is 200 | | |
| | Pass Content-Type header is application/json | | |
| ▼ GET | http://localhost:9001/pathologies/70 | 2 0 | |
| | Pass Status code is 200 | | |
| | Pass Content-Type header is application/json | | |
| ▼ GET | http://localhost:9001/hospitals | 2 0 | |
| | Pass Status code is 200 | | |
| | Pass Content-Type header is application/json | | |
| ▼ GET | http://localhost:9001/hospitals/1 | 3 0 | |
| | Pass Status code is 200 | | |
| | Pass Content-Type header is application/json | | |
| | Pass Found hospital CHATEAUDUN when hospital id = 1 | | |

7.1.2.3 Tests de la fonction demande intervention via API sous Postman

Résultat des tests avec la collection « POC API Emergency Nearest Hospitals » permettant de tester les différents cas de test d'appel à l'API via la requêtes http (Post /emergencies) pour demander une intervention médicale urgente. Retourne en temps réel les coordonnées de géo localisation et les consignes de l'hôpital le plus proche.

| | | |
|---|-------------------------------|-------|
| POC API Emergency Nearest H... No Environment, just now | | |
| RUN SUMMARY | | |
| | | 1 |
| ▼ POST | ERS Lunay Cardio Blois | 4 0 |
| Pass Status code is 200 | | |
| Pass Content-Type header is application/json | | |
| Pass Hospital found in Blois | | |
| Pass Response time is less than 200ms | | |
| ▼ POST | ERS Lunay Respi Vendôme | 5 0 |
| Pass Status code is 200 | | |
| Pass Content-Type header is application/json | | |
| Pass Hospital found in Vendôme | | |
| Pass Service name is Urgence | | |
| Pass Response time is less than 200ms | | |
| ▼ POST | ERS Respi Alerte pas de dispo | 4 0 |
| Pass Status code is 200 | | |
| Pass Hospital not found | | |
| Pass Emmergency instructions Alerte found | | |
| Pass Response time is less than 200ms | | |
| ▼ POST | ERS Monnaie Neuro Tours | 4 0 |
| Pass Status code is 200 | | |
| Pass Content-Type header is application/json | | |
| Pass Hospital found in Tours | | |
| Pass Response time is less than 200ms | | |
| ▼ POST | ERS Monnaie Pédiatrie Tours | 4 0 |
| Pass Status code is 200 | | |
| Pass Content-Type header is application/json | | |
| Pass Hospital found in Tours | | |
| Pass Response time is less than 200ms | | |
| ▼ POST | ERS Pithiviers Trauma Saran | 4 0 |
| Pass Status code is 200 | | |
| Pass Content-Type header is application/json | | |
| Pass Hospital found in Saran | | |
| Pass Response time is less than 200ms | | |

7.1.2.4 Tests de montée en charge via API sous Postman

Résultat des tests avec la collection « POC API Emergency Loading Test » permettant de tester la montée en charge. On exécute 800 requêtes (Post/Emergencies) en 66 secondes. On remarquera que chaque requête s'exécute en moins de 20 ms.

| | | | | | | | |
|---|---------------------------------------|-----------------------------------|-------------------------|--------------|-----------|-------|----------------|
| POC API Emergency Stress Test No Environment, a min ago | | | | View Summary | Run Again | New | Export Results |
| All Tests | Passed (3204) | Failed (0) | | | | | |
| Pass | Response time is less than 200ms | | | | | | 1 |
| POST | ERS Lunay Pédiatrie 3 | http://localhost:9001/emergencies | / ERS Lunay Pédiatrie 3 | 200 OK | 7 ms | 951 B | 2 |
| Pass | Status code is 200 | | | | | | 3 |
| Pass | Hospital found | | | | | | 4 |
| Pass | Hospital distance is less than 100kms | | | | | | 5 |
| Pass | Response time is less than 200ms | | | | | | 6 |
| POST | ERS Lunay Trauma 3 | http://localhost:9001/emergencies | / ERS Lunay Trauma 3 | 200 OK | 10 ms | 951 B | 7 |
| Pass | Status code is 200 | | | | | | 8 |
| Pass | Hospital found | | | | | | 9 |
| Pass | Hospital distance is less than 100kms | | | | | | 10 |
| Pass | Response time is less than 200ms | | | | | | 11 |
| POST | ERS Respi Alerte 3 | http://localhost:9001/emergencies | / ERS Respi Alerte 3 | 200 OK | 6 ms | 846 B | 12 |
| Pass | Status code is 200 | | | | | | 13 |
| Pass | Hospital not found | | | | | | 14 |
| Pass | Emergency instructions Alerte found | | | | | | 15 |
| Pass | Response time is less than 200ms | | | | | | 16 |
| POST | ERS Lunay Cardio 4 | http://localhost:9001/emergencies | / ERS Lunay Cardio 4 | 200 OK | 8 ms | 951 B | 17 |
| Pass | Status code is 200 | | | | | | 18 |
| Pass | Hospital found | | | | | | 19 |
| Pass | Hospital distance is less than 100kms | | | | | | 20 |
| Pass | Response time is less than 200ms | | | | | | 21 |
| POST | ERS Lunay Neuro 4 | http://localhost:9001/emergencies | / ERS Lunay Neuro 4 | 200 OK | 7 ms | 956 B | 22 |
| Pass | Status code is 200 | | | | | | 23 |
| Pass | Hospital found | | | | | | 24 |
| Pass | Hospital distance is less than 100kms | | | | | | 25 |
| Pass | Response time is less than 200ms | | | | | | 26 |
| | | | | | | | 27 |
| | | | | | | | 28 |
| | | | | | | | 29 |
| | | | | | | | 30 |
| | | | | | | | 31 |
| | | | | | | | 32 |

7.2 Exemples d'erreur

7.2.1 Sous GitHub

Sur le Pull Request

[Pull requests](#) 1 [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insights](#) [Settings](#)

data test #7


[Open](#) Mamak2020 wants to merge 1 commit into `main` from `Sprint5_v10`

Conversation 1


Commits 1


Checks 5

Files changed 10


 Mamak2020 commented 14 hours ago Owner ...


No description provided.


 data test 52e20a1


 sonarcloud bot commented 14 hours ago ...


SonarCloud Quality Gate failed. Failed


 0 Bugs

 0 Vulnerabilities


 0 Security Hotspots


 7 Code Smells

 100.0% Coverage



 0.0% Duplication



Add more commits by pushing to the `sprint5_v10` branch on Mamak2020/poc-api-ers.







 **Some checks were not successful** Hide all checks



4 successful and 1 failing checks


  CI/CD Pipeline / Tests (pull_request) Successful in 1m Details

  CI/CD Pipeline / Build (pull_request) Successful in 23s Details

  CI/CD Pipeline / SonarCloud analysis (pull_request) Successful in 1m Details

  CI/CD Pipeline / Deploy (pull_request) Successful in 17s Details

  SonarCloud Code Analysis Failing after 21s — Quality Gate failed Details

 **This branch has no conflicts with the base branch**
Merging can be performed automatically.

Merge pull request

You can also [open this in GitHub Desktop](#) or [view command line instructions](#).

28

Sur les tests

Mamak2020/poc-api-ersPublic

PinUnwatch2Fork0Star0

<> CodeIssuesPull requestsActionsProjectsWikiSecurityInsightsSettings

Unit tests, data set, refactoring CI/CD Pipeline #44

Re-run all jobs

Summary

Jobs

TestsBuildSonarCloud analysisDeploy

Triggered via pull request 7 days ago

Mamak2020 opened #4 Sprint4_v08

StatusFailure

Total duration45s

Artifacts-

maven.yml

on: pull_request

Tests29sBuild0sSonarCloud analysis0sDeploy0s

Annotations

1 error

TestsProcess completed with exit code 1.

Testsfailed 7 days ago in 29sSearch logs

Run Tests

132 Duree: 4 ms, startedAt: 2022-02-08T18:25:53.638193844Z

133 [INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.017 s - in com.medhead.ers.service.EmergencyServiceTest

134 [INFO] Running com.medhead.ers.PocErsApplicationTests

135 [INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.009 s - in com.medhead.ers.PocErsApplicationTests

136 [INFO]

137 [INFO] Results:

138 [INFO]

139 Error: Failures:

140 Error: EmergencyControllerTest.testCreateEmergency2:87 JSON path "\$.hospitalName" expected:<CH VENDOME - MONTOIRE> but was:<null>

141 [INFO]

142 Error: Tests run: 9, Failures: 1, Errors: 0, Skipped: 0

143 [INFO]

144 [INFO] -----

145 [INFO] BUILD FAILURE

146 [INFO] -----

147 [INFO] Total time: 15.757 s

148 [INFO] Finished at: 2022-02-08T18:25:54Z

149 [INFO] -----

150 Error: Failed to execute goal org.apache.maven.plugins:maven-surefire-plugin:2.22.2:test (default-test) on project ers: There are test failures.

151 Error:

152 Error: Please refer to /home/runner/work/poc-api-ers/poc-api-ers/target/surefire-reports for the individual test results.

153 Error: Please refer to dump files (if any exist) [date].dump, [date]-jvmRun[N].dump and [date].dumpstream.

154 Error: -> [Help 1]

155 Error:

156 Error: To see the full stack trace of the errors, re-run Maven with the -e switch.

157 Error: Re-run Maven using the -X switch to enable full debug logging.

158 Error:

159 Error: For more information about the errors and possible solutions, please read the following articles:

160 Error: [Help 1] <http://cwiki.apache.org/confluence/display/MAVEN/MojoFailureException>

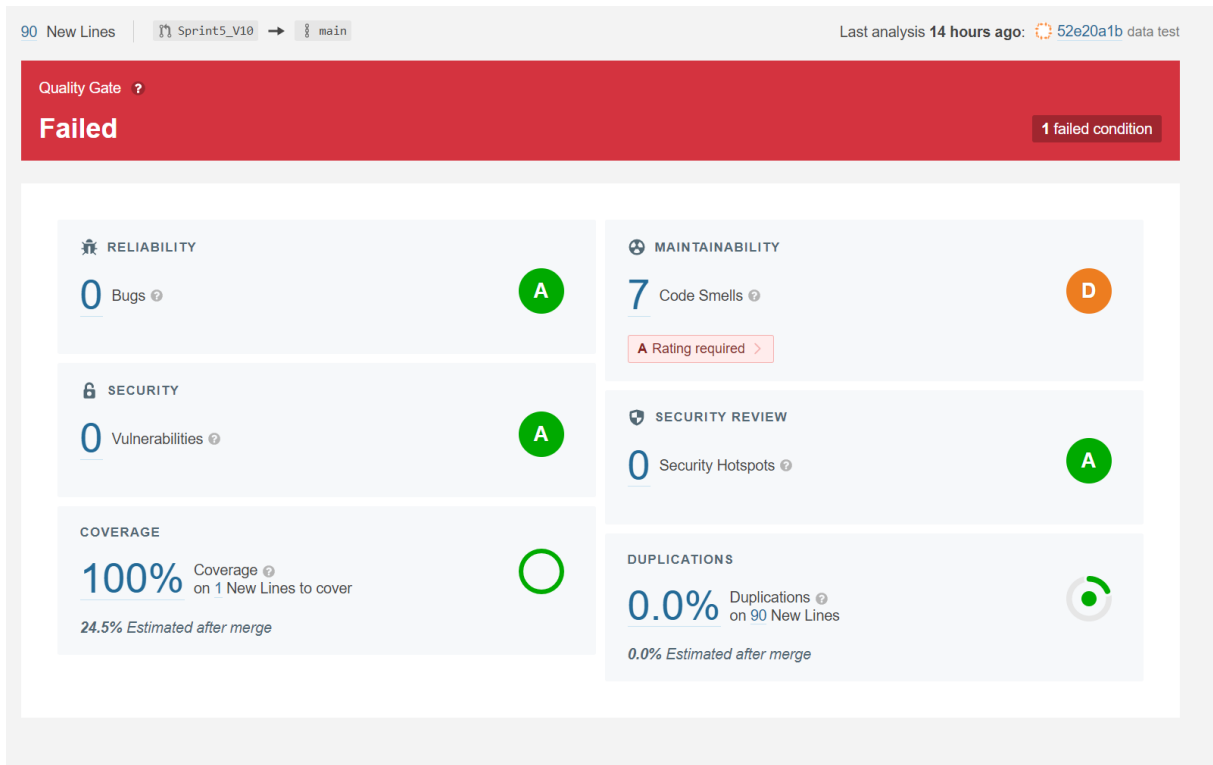
161 Error: Process completed with exit code 1.

> Post Set up JDK 17

> Post Run actions/checkout@v2

> Complete job

7.2.2 Sous SonarCloud



| Filters | | ↑ ↓ to select issues ← → to navigate ↺ 1 / 7 issues 1h effort | |
|--------------------------|---|--|--|
| Type | | src/.../java/com/medhead/ers/service/EmergencyService.java | |
| Bug | 0 | Define and throw a dedicated exception instead of using a generic one. Why is this an issue? 14 hours ago L44 No tags | |
| Vulnerability | 0 | Code Smell Major Open Magalie Morteau 20min effort | |
| Code Smell | 7 | Remove the declaration of thrown exception 'java.lang.Exception', as it cannot be thrown from method's body. Why is this an issue? 14 hours ago L44 No tags | |
| Severity | | Code Smell Minor Open Magalie Morteau 5min effort | |
| Blocker | 1 | This block of commented-out lines of code should be removed. Why is this an issue? 14 hours ago L66 No tags | |
| Critical | 0 | Code Smell Major Open Magalie Morteau 5min effort | |
| Major | 4 | Remove this useless assignment to local variable "duration". Why is this an issue? 6 days ago L132 No tags | |
| Resolution | | Code Smell Minor Open Magalie Morteau 5min effort | |
| Status | | Remove this unused "duration" local variable. Why is this an issue? 6 days ago L132 No tags | |
| Security Category | | Code Smell Minor Open Magalie Morteau 5min effort | |
| Creation Date | | This block of commented-out lines of code should be removed. Why is this an issue? 14 hours ago L134 No tags | |
| Language | | Code Smell Major Open Magalie Morteau 5min effort | |
| Rule | | src/.../java/com/medhead/ers/service/HospitalPathologyServiceTest.java | |
| Tag | | Complete the assertion. Why is this an issue? 14 hours ago L39 No tags | |
| Directory | | Code Smell Blocker Open Magalie Morteau 5min effort | |
| File | | | |
| Assignee | | | |
| | | 7 of 7 shown | |