

Faites adhérer

les parties prenantes

Avec une POC

Sommaire :

I. Le Contexte

II. Les Contraintes

III. Structuration de la POC

IV. Éléments à conserver

V. Conclusions

Sommaire :

I. Le Contexte

II. Les Contraintes

III. Structuration de la POC

IV. Éléments à conserver

V. Conclusions

I. Le Contexte

Institutions médicales Britanniques

Multi-plateforme

Multi-plateforme

Dette technique

Délai d'intervention trop long

I. Le Contexte

Institutions médicales Britanniques

Multi-plateformes

Affectation de lits

Affectation de lits

Selon les cas

Pathologie du patient

Distance à parcourir

Disponibilité & urgence

I. Le Contexte

Institutions médicales Britanniques

Multi-plateformes

Affectation de lits

Recherche l'uniformisation

Sommaire :

I. Le Contexte

II. Les Contraintes

III. Structuration de la POC

IV. Éléments à conserver

V. Conclusions

II. Les Contraintes

Un outil

- JAVA

- Haute disponibilité

- Haute sécurité

Sommaire :

I. Le Contexte

II. Les Contraintes

III. Structuration de la POC

IV. Éléments à conserver

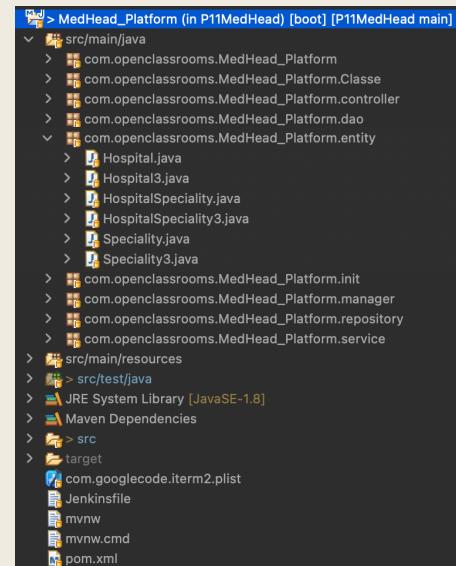
V. Conclusions

III. Structuration de la POC

État d'avancement du projet

État d'avancement du projet

Les bases de données



```
MedHead_Platform (in P11MedHead) [boot] [P11MedHead main]
src/main/java
> com.openclassrooms.MedHead_Platform
> com.openclassrooms.MedHead_Platform.Classe
> com.openclassrooms.MedHead_Platform.controller
> com.openclassrooms.MedHead_Platform.dao
> com.openclassrooms.MedHead_Platform.entity
> Hospital.java
> Hospital3.java
> HospitalSpeciality.java
> HospitalSpeciality3.java
> Speciality.java
> Speciality3.java
> com.openclassrooms.MedHead_Platform.init
> com.openclassrooms.MedHead_Platform.manager
> com.openclassrooms.MedHead_Platform.repository
> com.openclassrooms.MedHead_Platform.service
src/main/resources
src/test/java
JRE System Library [JavaSE-1.8]
Maven Dependencies
src
target
com.googlecode.iterm2.plist
Jenkinsfile
mvnw
mvnw.cmd
pom.xml
```

État d'avancement du projet

Les bases de données

Les points d'entrée

```

@ResiController
public class MainController {
    @Autowired
    private HospitalService hospitalService;
    //*****Zone Tests*****
    //GetMapping("hospital/speciality")
    public List<String> getSpecialityCity(@RequestParam String city) {
        List<String> speciality = hospitalService.findByIdSpecialitiesByCity(city);
        return speciality;
    }
    //GetMapping("hospital/cityspeciality")
    public List<String> getCitySpeciality(@RequestParam String speciality) {
        List<String> city = hospitalService.findByCityBySpecialities(speciality);
        return city;
    }
    //GetMapping("hospital/groupcity")
    public List<String> getGroupCity(@RequestParam String city) {
        List<String> group = hospitalService.findByGroupsByCity(city);
        return group;
    }
    //GetMapping("hospital/namcity")
    public List<String> getNameCity(@RequestParam String city) {
        List<String> name = hospitalService.findByIdNameByCity(city);
        return name;
    }
    //GetMapping("hospital/bedsity")
    public List<Integer> getBedsByCity(@RequestParam String city) {
        List<Integer> beds = hospitalService.findByBedsByCity(city);
        return beds;
    }
    //GetMapping("hospital/bedcity")
    public List<Integer> getBedByCity(@RequestParam String city) {
        Integer bed = hospitalService.findByBedByCity(city);
        return bed;
    }
    //GetMapping("hospital/loncity")
    public List<Double> lon = hospitalService.findByLonByCity(city);
    return lon;
    //GetMapping("hospital/latcity")
    public List<Double> lat = hospitalService.findByLatByCity(city);
    return lat;
    //*****DQL*****
    //GetMapping("hospital/id")
    public List<Hospital> getHospitalById(@RequestParam Long id) {
        List<Hospital> hospitalId = hospitalService.findByIdHospitalById(id);
        return hospitalId;
    }
    //GetMapping("hospital")
    public ResponseEntity<List<Hospital>> getAllHospital() {
        try {
            List<Hospital> list = hospitalService.findAll();
            if(list.isEmpty() || list.size() == 0) {
                return new ResponseEntity<List<Hospital>>((HttpStatus.NO_CONTENT));
            }
            return new ResponseEntity<List<Hospital>>(list, HttpStatus.OK);
        } catch (Exception e) {
            return new ResponseEntity<List<Hospital>>(null, HttpStatus.INTERNAL_SERVER_ERROR);
        }
    }
    //PostMapping("hospital")
    public ResponseEntity<Hospital> save(@RequestBody Hospital hospital) {
        try {
            return new ResponseEntity<Hospital>((hospitalService.save(hospital), HttpStatus.CREATED));
        } catch (Exception e) {
            return new ResponseEntity<Hospital>((null, HttpStatus.INTERNAL_SERVER_ERROR));
        }
    }
    //*****DQL*****
    //GetMapping("hospital?")
    public ResponseEntity<List<Hospital>> getAllHospitalBySpeciality(@RequestParam String speciality) {
        try {
            List<Hospital> list = hospitalService.findByIdBySpeciality(speciality);
            if(list.isEmpty() || list.size() == 0) {
                return new ResponseEntity<List<Hospital>>((HttpStatus.NO_CONTENT));
            }
            return new ResponseEntity<List<Hospital>>(list, HttpStatus.OK);
        } catch (Exception e) {
            return new ResponseEntity<List<Hospital>>(null, HttpStatus.INTERNAL_SERVER_ERROR);
        }
    }
    //GetMapping("hospital/withbeds")
    public List<Hospital> getHospitalNumberedWithBeds(@RequestParam String speciality) {
        List<Hospital> hospitalWithBeds = new ArrayList<Hospital>();
        Hospital nearestHospital = new Hospital();

```

État d'avancement du projet

Les bases de données

Les points d'entrée

Les retours Postman

Les retours Postman

Nom de l'établissement

Groupes spécialités

Spécialités

Capacité d'accueil

Lits vacants

Apte pour pathologie et géolocalisation

http://localhost:9010/hospital/speciality

GET http://localhost:9010/hospital/speciality

Params Authorization Headers (8) **Body** Pre-request Script Tests Set

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

```

1   "specialityRequest": "CARDIOLOGIE",
2   "latPatient": 4.5,
3   "lonPatient": 1.5

```

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```

1 [
2   {
3     "id": 1,
4     "city": "Bordeaux",
5     "name": "Hôpital Saint-André",
6     "beds": 150,
7     "bedsa": 75,
8     "lat": 44.83,
9     "lon": -0.6,
10    "distance": 448.3100212019907,
11    "hospitalCenter": "Bordeaux",
12    "numberOfBeds": 150,
13    "geographicalPositionLat": 44.83,
14    "geographicalPositionLon": -0.6
15  },
16  {
17    "id": 3,
18    "city": "Brest",
19    "name": "Hôpital de la Cavale Blanche",
20    "beds": 10,
21    "bedsa": 5,
22    "lat": 48.39,
23    "lon": -4.48,
24    "distance": 490.67254619786274,
25    "hospitalCenter": "Brest",
26    "numberOfBeds": 10,
27    "geographicalPositionLat": 48.39,
28    "geographicalPositionLon": -4.48

```

État d'avancement du projet

Les bases de données

Les points d'entrée

Les retours Postman

Les tests

Les tests

CRUD

Unitaire JUnit

Intégration Mockmvc

```

@Test
public void testSpecialitycity() throws Exception {
    mockMvc.perform(get("/hospital/specialitycity?city=Brest")).andDo(print())
        .andExpect(jsonPath("$.0", is("CARDIOLOGIE"))).andExpect(status().isOk());
}

@Test
public void testCityBySpeciality() throws Exception {
    mockMvc.perform(get("/hospital/cityspeciality?speciality=CARDIOLOGIE")).andDo(print())
        .andExpect(jsonPath("$.0", is("Bordeaux"))).andExpect(status().isOk());
}

@Test
public void testGroupsByCity() throws Exception {
    mockMvc.perform(get("/hospital/groupscity?city=Nantes")).andDo(print())
        .andExpect(jsonPath("$.0", is("MEDECINE GENERALE"))).andExpect(status().isOk());
}

@Test
public void testNameByCity() throws Exception {
    mockMvc.perform(get("/hospital/namecity?city=Mulhouse")).andDo(print())
        .andExpect(jsonPath("$.0", is("Centre Hospitalier"))).andExpect(status().isOk());
}

@Test
public void testBedsabyCity() throws Exception {
    mockMvc.perform(get("/hospital/bedsacity?city=Orléans")).andDo(print()).andExpect(jsonPath("$.0", is("100")))
        .andExpect(status().isOk());
}

@Test
public void testBedsByCity() throws Exception {
    mockMvc.perform(get("/hospital/bedscity?city=Toulouse")).andDo(print()).andExpect(jsonPath("$.0", is("100")))
        .andExpect(status().isOk());
}

@Test
public void testLonByCity() throws Exception {
    mockMvc.perform(get("/hospital/loncity?city=Lille")).andDo(print()).andExpect(jsonPath("$.0", is("100")))
        .andExpect(status().isOk());
}

@Test
public void testLatByCity() throws Exception {
    mockMvc.perform(get("/hospital/latcity?city=Mulhouse")).andDo(print()).andExpect(jsonPath("$.0", is("100")))
        .andExpect(status().isOk());
}

```

État d'avancement du projet

Les bases de données

Les points d'entrée

Les retours Postman

Les tests

Le contrôle continu CI/CD

Le contrôle continu CI/CD

Hébergement GitLab

Framework Surefire

Framework JaCoCo

Édition de rapports

  Surefire Report

Last Published: 2021-12-28 | Version: 0.0.1-SNAPSHOT
Built by Maven

Surefire Report

[\[Summary\]](#) [\[Package List\]](#) [\[Test Cases\]](#)

Tests	Errors	Failures	Skipped	Success Rate	Time
18	0	7	0	61.111%	20.701

Note: failures are anticipated and checked for with assertions while errors are unanticipated.

Package List

[\[Summary\]](#) [\[Package List\]](#) [\[Test Cases\]](#)

Package	Tests	Errors	Failures	Skipped	Success Rate	Time
com.openclassrooms.MedHead_Platform	18	0	7	0	61.111%	20.701

Note: package statistics are not computed recursively, they only sum up all of its institutes numbers.

com.openclassrooms.MedHead_Platform

	Class	Tests	Errors	Failures	Skipped	Success Rate	Time
MedHeadPlatformApplicationTests	MedHeadPlatformApplicationTests	6	0	1	0	83.333%	20.194
	HospitalControllerTest	12	0	6	0	50%	0.507

Test Cases

[\[Summary\]](#) [\[Package List\]](#) [\[Test Cases\]](#)

MedHeadPlatformApplicationTests

Sommaire :

I. Le Contexte

II. Les Contraintes

III. Méthodologie

IV. Éléments à conserver

V. Conclusions

IV. Éléments à conserver

Les bases de données

Les tests

Les ends points

Le repository

Le contrôle continu

Sommaire :

I. Le Contexte

II. Les Contraintes

III. Structuration de la POC

IV. Éléments à conserver

V. Conclusions

V. Conclusions

Une POC qui permet de se projeter

Une POC surveillée

Une POC prête à évoluer

Merci de votre attention

J'espère que cette présentation vous a plu

Je me tiens à votre disposition pour toutes informations complémentaires