MedHead

Faites adhérer

les parties prenantes

Avec une POC

Sommaire :

Sommaire :

I. Le Contexte

MedHead

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 :

II. Les Contraintes

Un outil

- JAVA

- Haute disponibilité

- Haute sécurité

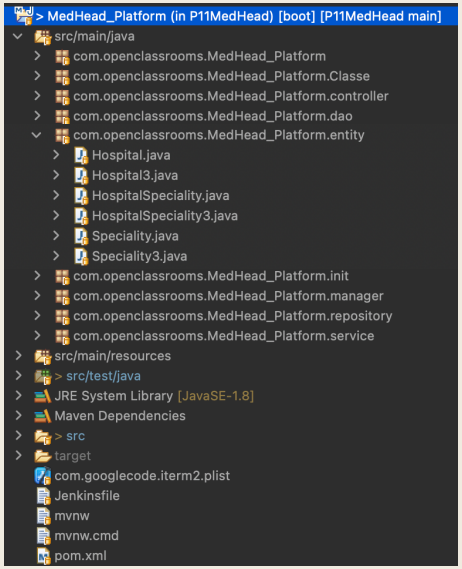
Sommaire :

III. Structuration de la POC

État d'avancement du projet

État d'avancement du projet

Les bases de données



État d’avancement du projet

MedHead

Les bases de données

Les points d’entrée

```java
@RestController
public class MainController {

    @Autowired
    private HospitalService hospitalService;

  //**********************Zone_Tests*****************************************************************************************

    @GetMapping("/hospital/specialitycity")
    public List<String> getSpecialityByCity(@RequestParam String city) {
        List<String> speciality = hospitalService.findBySpecialitiesByCity(city);
        return speciality;
    }

    @GetMapping("/hospital/cityspeciality")
    public List<String> getCityBySpecialty(@RequestParam String speciality) {
        List<String> city = hospitalService.findByCityBySpecialities(speciality);
        return city;
    }

    @GetMapping("/hospital/groupscity")
    public List<String> getGroupsByCity(@RequestParam String city) {
        List<String> groups = hospitalService.findByGroupsByCity(city);
        return groups;
    }

    @GetMapping("/hospital/namecity")
    public List<String> getNameByCity(@RequestParam String city) {
        List<String> name = hospitalService.findByNameByCity(city);
        return name;
    }

    @GetMapping("/hospital/bedsacity")
    public List<Integer> getBedsaByCity(@RequestParam String city) {
        List<Integer> bedsa = hospitalService.findByBedsaByCity(city);
        return bedsa;
    }

    @GetMapping("/hospital/bedscity")
    public List<Integer> getBedsByCity(@RequestParam String city) {
        List<Integer> bed = hospitalService.findByBedsByCity(city);
        return bed;
    }

    @GetMapping("/hospital/loncity")
    public List<Double> getLonByCity(@RequestParam String city) {
        List<Double> lon = hospitalService.findByLonByCity(city);
        return lon;
    }

    @GetMapping("/hospital/latcity")
    public List<Double> getLatByCity(@RequestParam String city) {
        List<Double> lat = hospitalService.findByLatByCity(city);
        return lat;
    }

    // ********Nouveau*****************

    @GetMapping("/hospital/idid")
    public List<Hospital> getHospitalById(@RequestParam Long id) {
        List<Hospital> hospitalById = hospitalService.findByHospitalById(id);
        return hospitalById;
    }

    @GetMapping("/hospital")
    public ResponseEntity<List<Hospital>> getAllHospitals() {
        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<>(null, HttpStatus.INTERNAL_SERVER_ERROR);
        }

    }

    @PostMapping("/hospital")
    public ResponseEntity<Hospital> save(@RequestBody Hospital hospital) {
        try {
            return new ResponseEntity<>(hospitalService.save(hospital), HttpStatus.CREATED);
        } catch (Exception e) {
            return new ResponseEntity<>(HttpStatus.INTERNAL_SERVER_ERROR);
        }
    }

    // ********Nouveau*****************

    @GetMapping("/hospital2")
    public ResponseEntity<List<Hospital>> getAllHospitalsBySpeciality(@RequestParam String speciality) {
        try {
            List<Hospital> list = hospitalService.findByBySpeciality(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<>(null, HttpStatus.INTERNAL_SERVER_ERROR);
        }

    }

    @GetMapping("/hospital/speciality")
    public List<Hospital> getAllHospitalsNumberOfBedsSpeciality(@RequestBody HospitalWithDisponibilityRequest request) {
        List<Hospital> bedsDispo = hospitalService.findByBySpeciality(request.getSpecialityRequest());
        List<Hospital> hospitalWithBeds = new ArrayList<Hospital>();
        Hospital nearestHospital = new Hospital();
```

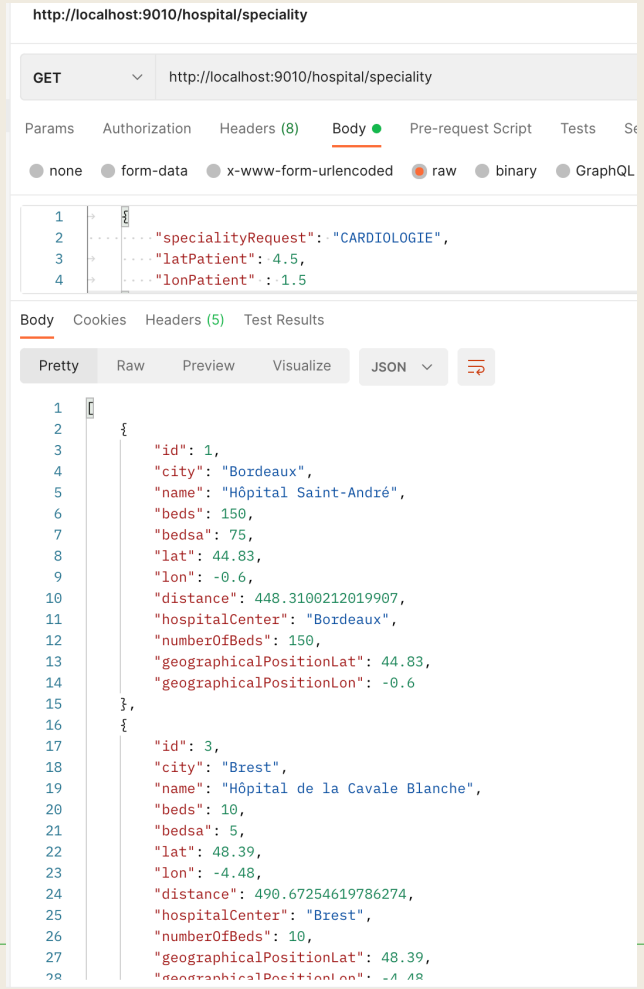
État d’avancement du projet

MedHead

- 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



État d'avancement du projet

MedHead

- Les bases de données
- Les points d'entrée
- Les retours Postman
- Les tests

Les tests

MedHead

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("
            .andExpect(status().isOk());
}

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

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

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

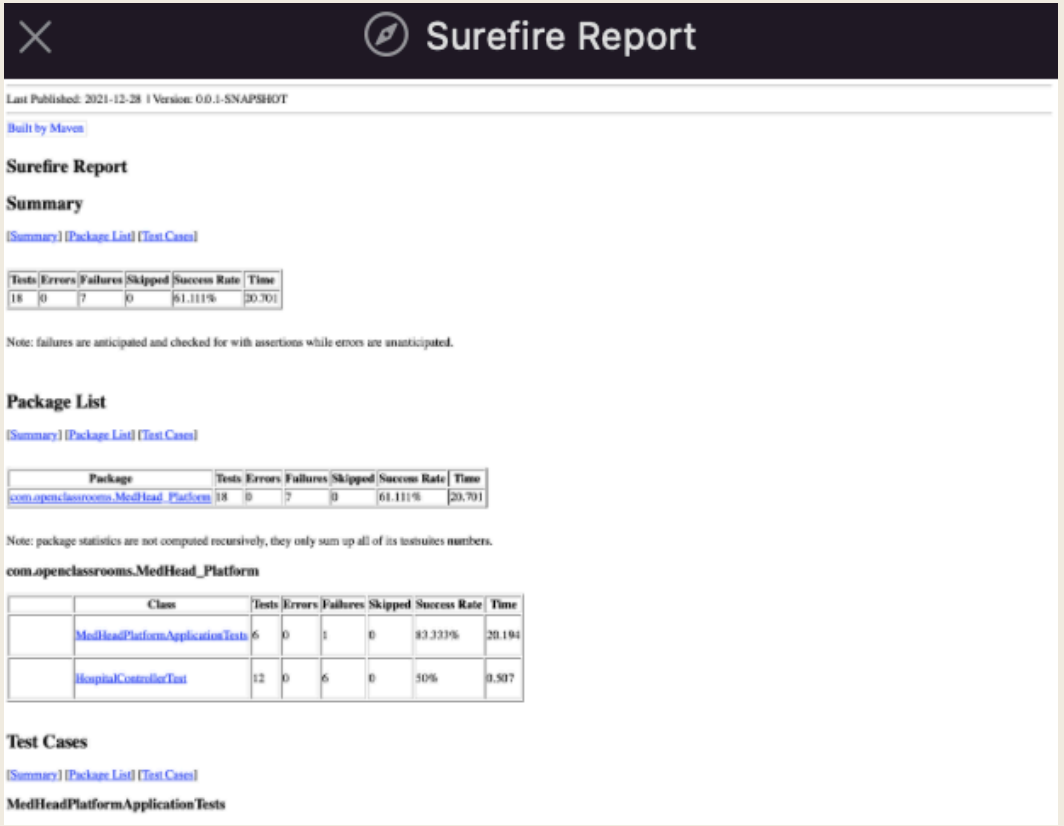
État d'avancement du projet

MedHead

- Les bases de données
- Les points d'entrée
- Les retours Postman
- Les tests
- Le contrôle continue CI/CD

Le contrôle continue CI/CD

- Hébergement GitLab
- Framework Surfire
- Framework JaCoCo
- Édition de rapports



Surefire Report

Last Published: 2021-12-28 | Version: 0.0.1-SNAPSHOT

Built by Maven

Surefire Report

Summary

[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 testsuites numbers.

com.openclassrooms.MedHead_Platform

	Class	Tests	Errors	Failures	Skipped	Success Rate	Time
	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 :

IV. Éléments à conserver

MedHead

- Les bases de données
- Les tests
- Les ends points
- Le repository
- Le contrôle continu

Sommaire :

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