MEDHEAD

Reporting

SOMMAIRE

1.Technologie utilisées

1.1.Backend

La technologie utilisée pour réaliser la partie backend de cette POC est JAVA, celle-ayant été imposée au début du projet.

1.2.Frontend

La technologie utilisée pour réaliser la partie frontend de cette POC est REACT, basée sur JAVASCRIPT.

La raison du choix de cette technologie est que je la connaissais déjà, et je l'ai donc donc choisi afin de gagner du temps.

2.Respect des normes et des principes

2.1. Respect des normes

Les normes qui devront être respectés sont :

- Prendre en compte l'accessibilité, avec un niveau minimal de AA second les recommandations du W3C, l'idéal étant la norme AAA,
- La partie backend devra utiliser le protocole HTTP2, et en HTTPS,
- L'utilisation des cookies devra être fait de façon minimale et aucune données personnelle ne devra être récupérée sans l'accord au préalable des utilisateurs, dès la première page,
- Le bas de page devra inclure un lien vers une page concernant :
    - les mentions légales, qui comprendra les informations sur MEDHEAD avec :
        - Son adresse,
        - La raison sociale,
        - Le nom du ou responsables de la société,
        - Le numéro de SIRET et de SIREN,
    - Les conditions générale d'utilisations
    - La politique de confidentialité, qui devra inclure les informations collectées, comment elles sont utilisées et les buts des ces collectes d'information, le temps qu'elles sont gardées,
    - les cookies, un lien devra être disponible en bas de chaque page du site, vers une page qui expliquera leurs utilités,
    - Une page permettant de fournir des informations sur la méthode de contacter la société, et éventuellement un formulaire de contact

2.2.Protection et sauvegarde des données

La protection des données selon le RGPD devra être respectées.

Les informations qui devront être collectées sont :

- Le nom de la personne,
- Son prénom,
- Son adresse,
- Son numéro de téléphone,
- Son Email,

Aucune autre information personnelle et n'étant pas utile pour cette application ne devra être demandée.

Une déclaration auprès de la CNIL devra également être fait concernant les informations que le site récupère.

Les données devront êtres archivés et rendues anonymes au bout d'une période bien définie, elles ne devront pas être disponible plus d'1 an maximum, et les données archivés ne devront pas être gardées plus de 2 ans.

2.3.Respect des principes

L'application sera basé sur une architecture microservice.

Les informations que devront fournir les utilisateurs lors de la réservation d'un lit devra être dans un autre microservice.

Si l'application comprend des informations sur l'utilisation d'un compte, ces dernières devront être dans un autre microservice.

La partie Backend sera accessible que par API depuis la partie Frontend, celle-ci pouvant ensuite être réalisé soit en REACT, VUE.JS ou ANGULAR.

3.Résultats et enseignement de la POC

3.1.Résultats

Partie backend

Les test unitaire ont été réalise avec Junit, ainsi que les tests d'intégrations.

Ces test peuvent être lancés en ligne de commande avec :

```
mvn tets.
```

Ils peuvent aussi être lancé au lancement du build avec :

```
mvn package
```

Les test de charges et de performances ont été réalisé avec JMeter.
Les résultats disponibles sur le repository GitHub https://github.com/stual37/oc-p11-backend/tree/master/rapport JMeter.
Ce résultat est également disponible en annexe.

Partie frontend

Les test unitaire ont été réalise avec JEst, ainsi que les tests d'intégrations.

Ces test peuvent être lancé en ligne de commande avec :

```
npm test.
```

Dans les deux cas, les test sur la CI-CD ont été réalisé avec Jenkins.
Un fichier Jenkinsfile a été fournis pour chaque partie, et une explication de leur mis en place se trouve dans chaque readme.
Un exemple est disponible sur le repository GitHub https://github.com/stual37/oc-p11-backend//Rapport%20Jenkins

3.2.Enseignement de la POC

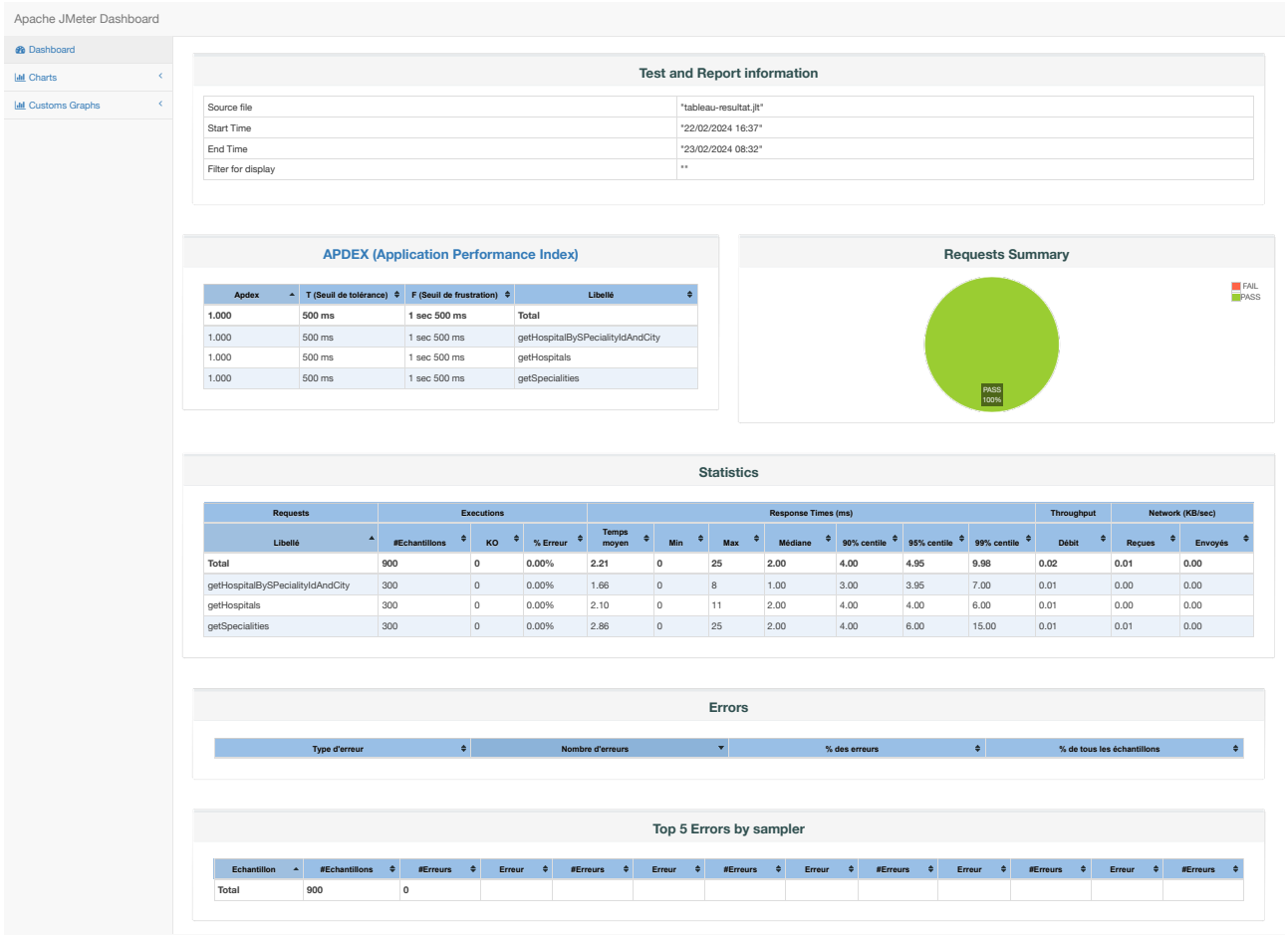
La réalisation de la POC comprend deux parties pour le développement, afin de démontrer qu'il est possible ainsi de réaliser des versionning de la partie backend et de la partie frontend de façon indépendante.

Chaque partie contient ses propres tests que ce soit unitaire, intégration ou CI-CD.

Annexe

Rapport Jmeter

Ci-dessous un exemple de rapport de tests sous JMeter :



Apache JMeter Dashboard

Dashboard
Charts
Customs Graphs

Test and Report information

Source file	"tableau-resultat.jlt"
Start Time	"22/02/2024 16:37"
End Time	"23/02/2024 08:32"
Filter for display	""

APDEX (Application Performance Index)

Apdex	T (Seuil de tolérance)	F (Seuil de frustration)	Libellé
1.000	**500 ms**	**1 sec 500 ms**	**Total**
1.000	500 ms	1 sec 500 ms	getHospitalBySPecialityIdAndCity
1.000	500 ms	1 sec 500 ms	getHospitals
1.000	500 ms	1 sec 500 ms	getSpecialities

Requests Summary

FAIL
PASS
PASS 100%

Statistics

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Libellé	#Echantillons	KO	% Erreur	Temps moyen	Min	Max	Médiane	90% centile	95% centile	99% centile	Débit	Reçues	Envoyés
Total	**900**	**0**	**0.00%**	**2.21**	**0**	**25**	**2.00**	**4.00**	**4.95**	**9.98**	**0.02**	**0.01**	**0.00**
getHospitalBySPecialityIdAndCity	300	0	0.00%	1.66	0	8	1.00	3.00	3.95	7.00	0.01	0.00	0.00
getHospitals	300	0	0.00%	2.10	0	11	2.00	4.00	4.00	6.00	0.01	0.00	0.00
getSpecialities	300	0	0.00%	2.86	0	25	2.00	4.00	6.00	15.00	0.01	0.01	0.00

Errors

Type d'erreur	Nombre d'erreurs	% des erreurs	% de tous les échantillons

Top 5 Errors by sampler

Echantillon	#Echantillons	#Erreurs	Erreur	#Erreurs	Erreur	#Erreurs	Erreur	#Erreurs	Erreur	#Erreurs	Erreur	#Erreurs
Total	**900**	**0**										

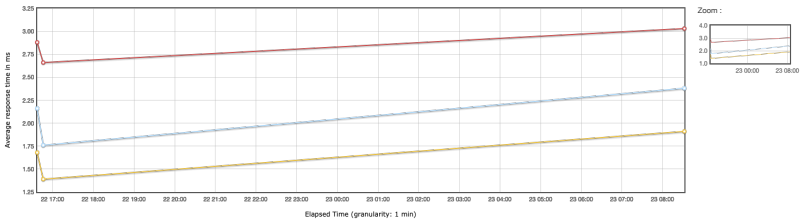
Apache JMeter Dashboard

- Dashboard
- Charts
  - Over Time
    - Response Times Over Time
    - Response Time Percentiles Over Time (successful responses)
    - Active Threads Over Time
    - Bytes Throughput Over Time
    - Latencies Over Time
    - Connect Time Over Time
  - Throughput
  - Response Times
- Customs Graphs

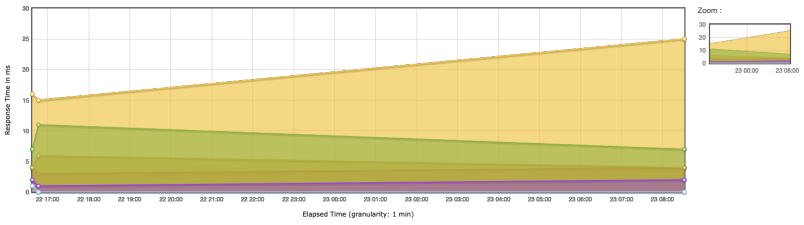
Test and Report information

File:	"tableau-resultat.jtl"
Start Time:	"22/02/2024 16:37"
End Time:	"23/02/2024 08:32"
Filter for display:	""

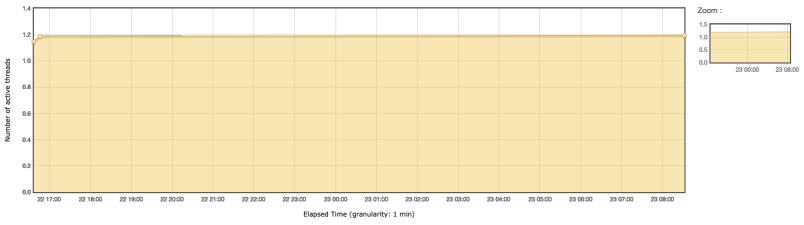
Response Times Over Time



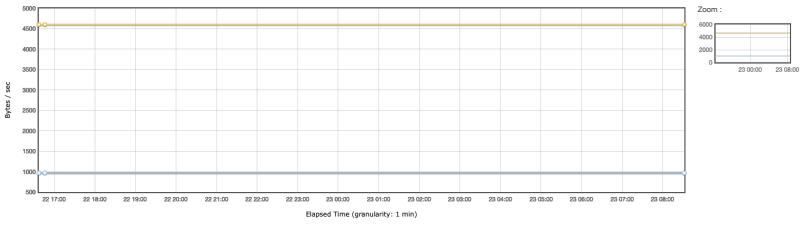
Response Time Percentiles Over Time (successful responses)



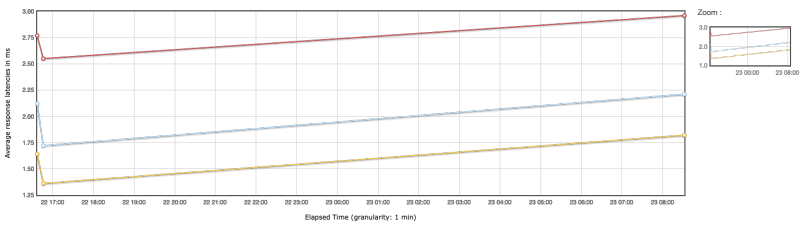
Active Threads Over Time



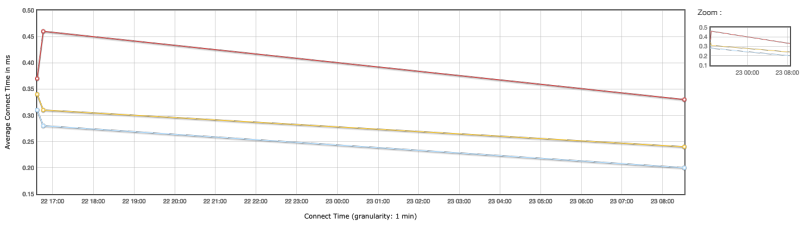
Bytes Throughput Over Time



Latencies Over Time



Connect Time Over Time



Exemple de rapport Jenkins avec erreurs

CI-dessous une partie d'un exemple de rapport d’une erreur de CI/CD sous Jenkins, qu’il est possible de voir au complet sue le lien suivant https://github.com/stual37/oc-p11-backend//Rapport%20Jenkins/

[INFO] --- spring-boot:3.2.2:repackage (default) @ backend ---

[INFO] Replacing artifact with classifier exec /var/jenkins_home/workspace/oc-p11-contenaire/pipeline-backend/target/backend-0.0.2-exec.jar with repackaged archive, adding nested dependencies in BOOT-INF/.

[INFO] The original artifact has been renamed to /var/jenkins_home/workspace/oc-p11-contenaire/pipeline-backend/target/backend-0.0.2-exec.jar.original

[INFO] --

[INFO] BUILD SUCCESS

[INFO] --

[INFO] Total time: 5.568 s

[INFO] Finished at: 2024-02-26T10:24:53Z

[INFO] --

Post stage

[Pipeline] junit

Recording test results

No test report files were found. Configuration error?

None of the test reports contained any result

[Checks API] No suitable checks publisher found.

Error when executing success post condition:

java.lang.NoSuchMethodError: No such DSL method '$archiveArtifacts' found among steps [archive, bat, build, catchError, checkout, deleteDir, dir, dockerFingerprintFrom, dockerFingerprintRun, dockerNode, echo, emailext, emailextrecipients, envVarsForTool, error, fileExists, findBuildScans, getContext, git, input, isUnix, junit, library, libraryResource, load, mail, milestone, node, parallel, powershell, properties, publishChecks, publishHTML, pwd, pwsh, readFile, readTrusted, resolveScm, retry, script, sh, sleep, stage, stash, step, timeout, timestamps, tm, tool, unarchive, unstable, unstash, validateDeclarativePipeline, waitForBuild, waitUntil, warnError, withChecks, withContext, withCredentials, withDockerContainer, withDockerRegistry, withDockerServer, withEnv, withGradle, wrap, writeFile, ws] or symbols [GitUsernamePassword, agent, all, allBranchesSame, allOf, always, ant, antFromApache, antOutcome, antTarget, any, anyOf, apiToken, apiTokenProperty, architecture, archiveArtifacts, artifactManager, assembla, attach, authorInChangelog, authorizationMatrix, batchFile, bitbucket, bitbucketBranchDiscovery, bitbucketBuildStatusNotifications, bitbucketForkDiscovery, bitbucketPublicRepoPullRequestFilter, bitbucketPullRequestDiscovery, bitbucketServer, bitbucketSshCheckout, bitbucketTagDiscovery, bitbucketTrustEveryone, bitbucketTrustNobody, bitbucketTrustProject, bitbucketTrustTeam, …