Groupe 6 : Jérémy MASSON, Nathan DALAINE, Maxence BRES, Antoine RIVOIRE, Baptiste BETEND, Tom SOLVERY

Fonctionnalités Système Partie 1

Fonctionnalités	**Réponses**	**Partie réservée Enseignant**
Authentification (JWT, OAuth2, …)	JWToken	
Bases de données utilisées	H2 en dev,PostgreSQL en prod	
Canevas Web utilisés	Angular,Typescript	
Nombre de ressources	9	
Nombre de Entity	6	
Nombre de Schedule	2	
Transaction XA	Spring Framework	
Echanges Client-Serveur (Websocket, REST, Stream (Kafka, RabbitMQ) ...)	Http, REST	
Gestion des médias (BD, File Systems, Caches, CDN, ...)	Aucune	
Caching	EhCache	
Ouitls collaboratifs (scm git, bug tracker, code review,…)	Git, gitlab for code review, gitkraken, messenger	
Méthodologies de test (unitaire, intégration, e2e, performance,…)	Performance, intégration,unitaire,	
Frameworks de test client (swagger, …)	swagger	
Performances : injection et supervision (JMeter, Gatling, …)	Gatling	
Gestion des dépendances (coté serveur, coté client,…)	Aucun rajout par rapport a jhipster	
Intégration en continue/ Déploiement en continue (CD-CI)	CI CD avec gitlab	

Groupe & Thème du service :

Livraison en continue	Non	
Cloud utilisé (Azure, AWS, Heroku, GCP, OVH, on premise ...)	Azure	

Fonctionnalités Système Partie 2

Fonctionnalités	**Réponses (Non/Oui + commentaires si oui)**	**Partie réservée Enseignant**
Haute disponibilité	Non	
Mécanismes pour le déploiement automatisé (Docker, Docker Compose, Kubernetes, Swarm, Rancher, ...)	Oui, Docker Compose pour déployer les containers Docker (postgreSQL, application, grafana)	
Déploiement automatisé sur une plateforme cloud	Oui publication automatique d'une image Docker de l'application à chaque push sur master. Mais besoin de relancer manuellement docker-compose sur la VM	
Interface CLI ou Shell pour l'administration et le bulk loading (initialisation du catalogue du service, l'ajout de nouveaux produits). Vous pouvez utiliser l'interface EJB facade directement ou bien une interface RESTful.	Interface cli généré par Jhipster en mode admin	
Gestion de l'internationalisation (i18n) des applications web et mobiles. Remarque : vous pouvez utiliser les principes et outils appris dans l'UE Communication Langagière.	Oui, traduction en fr et en	
Gestion de la confidentialité avec SSL/TLS lors des phases de login, signin, et de paiement, RGPD	Oui, https avec un reverse proxy nginx et un certificat SSL délivré par Let's Encrypt. Ainsi que rgpd	
Gestion de la concurrence et de la reprise sur panne avec des transactions ACID	non	

Groupe & Thème du service :

Gestion asynchrone et transactionnelle de l'envoi des courriels via AMPQ	Non	
Suivi du click stream avec des Filters en vue d'une analyse Big Data avec un ESP (Click Analytics, Recommender System).	Non	
Framework d'automatisation des tests (Junit, ...)	Oui, JUnits et Jest	
Intégration en continue (par exemple avec Jenkins, Travis-CI sur GitHub)	Oui, avec Gitlab CICD	
Livraison en continue (Rolling update)	Non	
Injection de Pannes (Netflix Simian Army, ...)	Non	
Reprise sur panne	Non	
Performances (résultat du injection de charge avec Apache JMeter ou Gatling)	Oui Gatling	
Infrastructure de supervision du système (Telegraf, Prometheus ...)	Oui Prometheus, Grafana	
Validation des services REST (Swagger, ...)	Oui Swagger	
Renseignement de la notice relative à la protection de la vie privée. (RGPD …)	Oui notice RGPD et CGU	
Utilisation de OAuth ou OpenID pour le login	OAuth	
Utilisation d'API tiers (Stripe, PubNub, Firebase, Google analytics, GeoIP, Criteo…)	Stripe	
Autres (listez les autres fonctionnalités intégrées) :		

Groupe & Thème du service :