

Atelier - Adam GABRIEL, Louis LEULIETTE, Gauvain BOICHÉ

ScèneLibre

ScèneLibre revend des billets de concerts et de spectacles. Cinq salles partenaires aujourd'hui, une quarantaine prévues dans un an. Le modèle : la salle publie ses dates, les clients achètent en ligne, ScèneLibre prend une commission de 6 %.

Côté client

- rechercher un spectacle par ville, date ou artiste
- choisir une catégorie de place et une quantité
- payer par carte bancaire
- recevoir son billet par e-mail, avec un code-barres
- demander un remboursement si le spectacle est annulé

Côté salle partenaire

- publier une date, définir les catégories et les tarifs
- suivre les ventes en temps réel
- éditer la liste des entrées le soir du spectacle

Côté ScèneLibre

- valider les salles partenaires
- suivre les commissions
- éditer les factures mensuelles aux salles

Règles métier

1. Un client ne peut pas acheter plus de 6 places pour un même spectacle.
2. Le tarif réduit (-30 %) s'applique aux moins de 26 ans et aux demandeurs d'emploi, sur justificatif téléversé.
3. La vente ferme 2 heures avant le début du spectacle.
4. Un billet est nominatif : le nom du porteur est imprimé dessus.
5. En cas d'annulation, le remboursement est intégral, commission comprise, sous 14 jours.
6. Une place vendue ne peut jamais l'être deux fois.
7. La commission de 6 % est calculée sur le prix hors taxes.
8. La TVA sur les spectacles est de 5,5 %.

Architecture technique

- un site web, consulté à 70 % depuis un téléphone
- une interface d'administration pour les salles
- une base de données
- un service de paiement externe (prestataire bancaire)

- un service d'envoi d'e-mails externe
- un générateur de PDF pour les billets

Contraintes

- Le pic de vente. Quand une grosse tête d'affiche est mise en vente, tout le monde se connecte à la même minute. Le record attendu : 40 000 personnes en 10 minutes.
- L'argent. Chaque transaction est un paiement réel.
- Les données personnelles. Noms, adresses, dates de naissance, justificatifs de situation. Et les coordonnées bancaires transitent par le prestataire.
- La loi. Le site est un service au public : l'accessibilité est une obligation légale.
- L'équipe. 4 développeurs, 1 personne dédiée à la qualité — vous.
- Le budget. Modeste. Une start-up.

Ce qui existe déjà

- du code, écrit vite, sans aucun test automatisé
- une seule machine, qui sert à la fois de serveur de développement et de démo commerciale
- une base de données, remplie à la main, avec 12 spectacles factices
- les mots de passe du prestataire de paiement sont dans un fichier de configuration, envoyé dans le dépôt Git
- personne n'a jamais fait de test de charge

Mission 01 - Analyse de risque

Risque	Ce qui se passe	Gravité	Propabilité	Priorité
Clef API presta dans le dépôt GIT	Bah... voilà quoi	5	5	25
Aucun test de charge préalable	Aucun moyen de savoir si une tête d'affiche sera absorbée	1	4	4
Base de données remplie à la main	Erreurs probables si non compartimentation bdd Dev/Prod	3	3	9
Code écrit vite sans aucun test	Le code peut contenir des erreurs s'il n'est pas corrigé et testé en condition réelle	5	3	15
Serveur unique de dev et de démo	En cas d'incident, plus rien	5	2	10
Risque RGPD	Aucune vérification de la conformité RGPD sur la sécurisation des données en transit	5	4	20
Délai de mise en prod trop court	Si le projet est mal géré dans le temps, la production peut ne pas se faire à temps	3	2	6
Pas de redondance dans les prestataires	Si le prestataire rencontre un incident de son côté, les paiements sont interrompus	5	1	5

Par priorité :

1. Clefs API à purger du dépôt APRES les avoir changées
2. Risques RGPD
3. Codé écrit à la va-vite à Linter
4. Dupliquer les serveurs, ou refaire l'infra en compartimentant
5. Dupliquer les BDD : DEV, STG et PRD
6. Délai de prod à tenir (on serre les fesses)
7. Trouver un prestataire alternatif en redondance
8. Définir des tests de charge pour les affluences

*Si vous n'aviez le temps de tester qu'une seule chose, laquelle serait-ce ? => **Le test de charge***

Mission 02 - Stratégie de test

1. Clefs APIs (et tout système de connexion) à purger

Type de test	SAST (Pénétration)
Qui l'exécute	Développeur
Quand	A chaque envoi de code
Automatisé / Manuel	Automatisé, avec SonarQube
Confirmation	SonarQube permet au pipeline de continuer les étapes suivantes

2. Conformité RGPD des données en transit

Type de test	Validation & Intégration
Qui l'exécute	Réalisation : Développeur, Approbateur : Qualité
Quand	A chaque envoi de code sur des faux jeux de données
Automatisé / Manuel	Automatisé, avec un outil de journalisation verbeux
Confirmation	Aucune donnée en clair (tout chiffré) dans les artefacts du pipeline

3. Base de code incertaine

Type de test	Unitaire (Statique)
Qui l'exécute	Développeur
Quand	A chaque envoi de code
Automatisé / Manuel	Automatisé, avec ESLint ou équivalent selon langage
Confirmation	Le code est qualifié et sans erreur majeure, et le pipeline se poursuit

4. Tests généraux de bon fonctionnement (charge, accessibilité, etc)

Type de test	Charge, Accessibilité
Qui l'exécute	Développeur

Type de test	Charge, Accessibilité
Quand	A chaque envoi de code, ou Chaque nuit selon la lourdeur des tests (+ de 30 minutes)
Automatisé / Manuel	Automatique, puis Manuel
Confirmation	Artefacts résiduels validés le matin

Frise chronologique

Développeur écrit son code V Test pénétration SonarQube (10 min) V Test unitaire *linter* (3 min) V Test de validation RGPD (4 min) V Test d'accessibilité (2 min) V Test de charge (10 min, ou plus, ou moins selon critères) V Test d'intégration (3 min) V Le site est en ligne

Mission 03 - Conception des cas de test

1. Test "Scénario basique"

6% de commission sur le montant HT TVA à 5,5% Billet nominatif Pas de vente 2h avant le spectacle

1. Test d'un prix P HT fixe type 100€ => Calcul final ($P * 1,06 * 1,055$) soit attendu 111,83€
2. Billet acheté sans prénom-nom => Erreur / Refus
3. Billet acheté 10h avant le spectacle => OK
4. Billet acheté 1h avant le spectacle => Erreur / Refus
5. Billet acheté deux fois => Erreur / Refus (vérification des ID avant)

Exceptions

6. Billet acheté 1h avant le spectacle sur un autre fuseau horaire => Horaire fixée à l'heure serveur
7. Billet unique acheté au même centième de seconde => Mise en tampon des achats pour vérification, gestion à la nanoseconde, sinon un random()

2. Test "Tarif réduit"

-30% pour les moins de 26 ans

1. Commande à 12 ans => Réduction (règle pour les -18 ans ? Interdit de commander pour les mineurs ?)
2. Commande à 18 ans => Réduction
3. Commande à 25 ans => Réduction
4. Commande à 26 ans => Réduction ou Tarif normal ? (moins de 26 ans strict ou 26 ans inclus)
5. Commande à 35 ans => Tarif normal
6. Commande à 35 ans, chômeur, justifié => Réduction
7. Commande à 35 ans, chômeur, injustifié => Tarif normal

3. Test "6 places maximum"

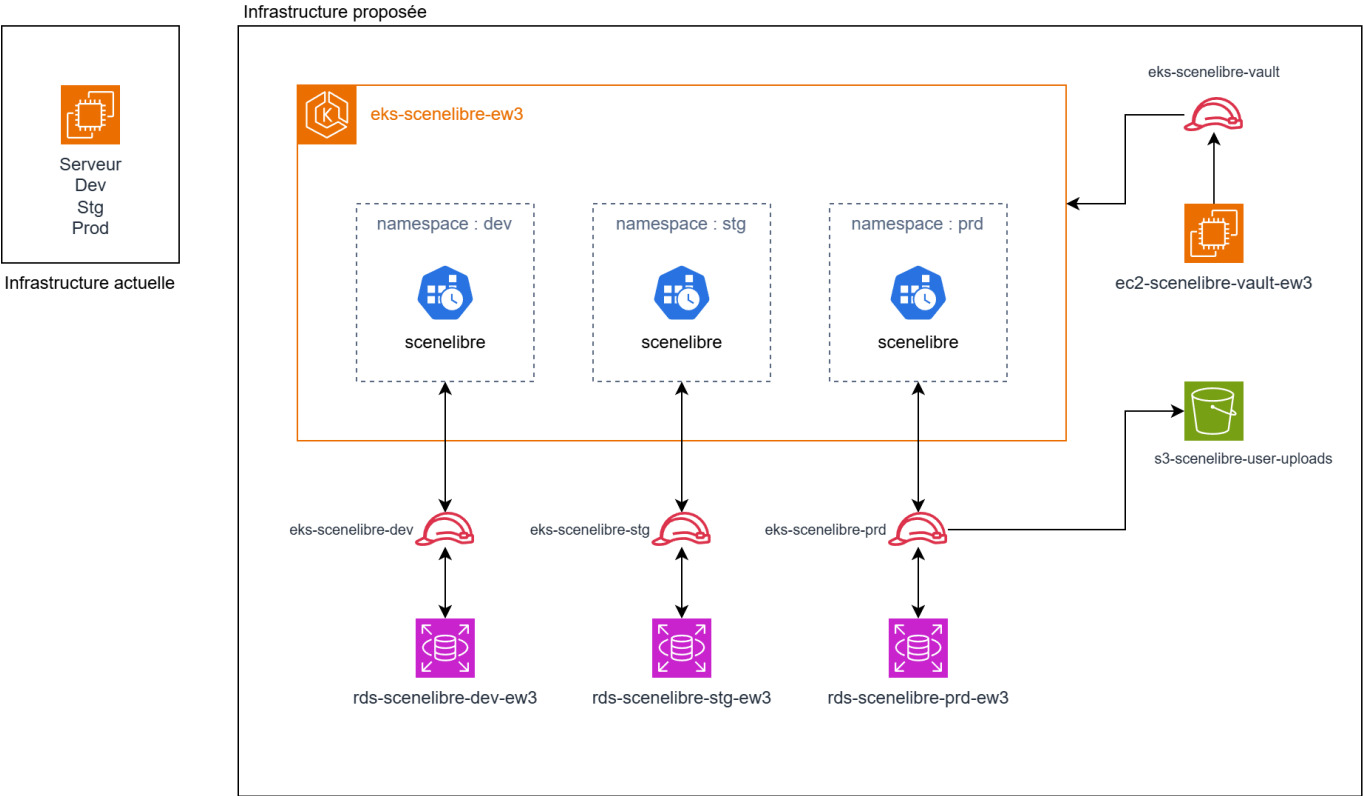
6 places maximum sur un seul spectacle Chaque place est vendue une seule fois

1. Commande de 0 place => Erreur / Refus

2. Commande de 1 à 6 places => OK
3. Commande de 7 places => Erreur / Refus

Mission 04 - Environnements

1. L'infrastructure



L'infrastructure présentée compte la Stack principale, mais ne compte pas les artefacts logiciels, les dépôts de code, les interfaces tierces comme des CloudFront ou des WAF, ni les applications déployées dans l'EKS comme celles dédiées à l'authentification des justificatifs téléversés, des tunnels de connexion, etc.

Nom	DEV	STG	PRD
Qui y accède	Les développeurs	Les développeurs	Les développeurs
Quelles données	Le code en test + faux jeu de données	Le code testé et stabilisé + faux jeu de données	Le code stabilisé + les données de production
Dimensionné comment	Avec un KEDA dans EKS, 15% des ressources	Avec un KEDA dans EKS, 15% des ressources	Avec un KEDA dans EKS, 70% des ressources
Durée de vie	Perpétuel	Perpétuel	Perpétuel
Ce qu'on y fait	On pousse du test, on casse, c'est la fête	Le test stabilisé en DEV est testé en conditions de prod	Le test stable subit les derniers tests d'intégration et de validation avec déploiement

2. Jeu de données

Mettons qu'une salle fasse avec une moyenne à 400 sur les 5 salles partenaires. Et que pour 40.000 places dispo, 50.000 tentent d'en acheter. Soit $40.000 \text{ places} / 5 \text{ salles} / 400 \text{ places} = 20 \text{ spectacles}$, pour 50.000 clients.

- 5 salles
- 20 spectacles
- 40.000 places
- 50.000 commandes

Avec l'usage d'un outil de génération de fausses identités (soit librairie type Faker, soit un outil maison qui fonctionne avec des dictionnaires faciles à trouver/peupler et facile à coder en quelques fonctions), on peut aisément peupler 50.000 clients dans une base de données de STG (pas de DEV).

Les tests vont croiser les tests de la **Mission 03** pour tester et retourner les résultats (Plus de 7 places, 0 place, chômeur injustifié, 26 ans, billets en double, etc)

- Les tests sur les + de 7 places doivent échouer
- Les billets doivent ne plus être vendus après 40.000 commandes (10.000 commandes échouées sur le volume total, sans compter les tentatives à + de 6 places)
- Un test idempotent doit renvoyer la même masse d'échecs (10.000 commandes échouées sur 50.000), les mêmes retours sur les masses éligibles aux réductions (-26 ans ou chômeurs, avec justificatifs)

3. Budget

Budget excluant masse salariale : 14.000€ / mois

- Pour EC2 Vault : t4g.small (2 vCPU, 2GB ram) = 7,5€ / mois
- Pour EKS : 1 cluster avec 2 noeuds hybrides à 8vCPUs + 1 App ArgoCD : 350€ / mois
- Pour RDS :
 - DEV et STG : db.t4g.small = 16€ x 2 (32€) / mois
 - PRD : db.t4g.xlarge = 126€ / mois
- Pour S3 : 1,1 € pour 100 Gigas de données stockées en Intelligent-Tiering avec 75% de données de + de 3 mois (Glacier IR) (à cumuler par tranches de 100 GB)

Soit 516,7€ par mois le premier mois, avec 1,1€ supplémentaire par mois chaque mois (estimation).

On peut dimensionner l'EKS et les RDS à l'aise avec un budget de 14.000€ / mois.

Mission 05 - Sécurité et conformité

1. Les trois urgences

A. Code écrit vite, sans test

Faibles potentielles, un NMAP + Metasploit pourrait ouvrir des portes d'entrée manifestes. Si le code est rédigé avec l'IA, aucune sécurité par défaut.

La mise en place des tests unitaires, la relecture du code et les patchs applicatifs + l'intégration d'une philosophie de "sécurité par défaut" (usage d'images renforcées / lancées avec des utilisateurs non **root**), d'une instance Vault dédiée aux secrets dans un VPC isolé, d'un Répartiteur de Charge avec un WAF pour

bloquer les requêtes anormales ou faire de l'obstruction sur les scans suffisent à sécuriser l'infrastructure contre 99,99% des attaques.

B. Une seule machine

Un attaquant peut lancer une attaque DDOS sur la machine et mettre l'équipe de développement au chômage technique + arrêter toute la mise en vente des billets, et faire perdre l'état des ventes actuelles.

Répartir la stack dans une grappe Kubernetes et compartimentée selon les environnements donne une résilience à l'infrastructure - pour peu que le dimensionnement en cas d'affluence soit bien géré, et que les "fausses" affluences soient bloquées par un WAF - et permet un développement serein de l'application. De plus, le point d'entrée étant protégé par défaut (avec le WAF), rentrer dans la grappe sera autrement plus compliqué.

C. Clef API du presta de paiement en clair sur le dépôt GIT

Un attaquant peut regarder le dépôt s'il est en public, et se faire passer pour ScèneLibre auprès du prestataire, et se faire payer au nom de ScèneLibre pour "n'importe quoi". Eventuellement ouvrir un site internet illégal (drogues, armes, thaïlandaises, etc) avec une gestion de paiement intitulé "ScèneLibre".

1. Changer les clefs API
2. Demander au prestataire un rapport des usages API (sauf si c'est accessible sur panneau admin client)
3. Pousser du nouveau code avec une clef API en **Secret** du dépôt GIT (en attendant Vault)

Option NUCLÉAIRE

1. Changer les clefs API
2. Demander au prestataire un rapport des usages API (sauf si c'est accessible sur panneau admin client)
3. Supprimer le dépôt et en refaire un au propre, et en invisible, sur une instance dédiée (Gitlab, ForgeJo, Gitea, etc) dans le nouveau compte nuagique sur un EC2 dédié

2. Le RGPD

Vous voulez des données réalistes pour tester. Le directeur commercial vous propose une copie de la base de démonstration, qui contient les vrais clients des cinq salles partenaires.

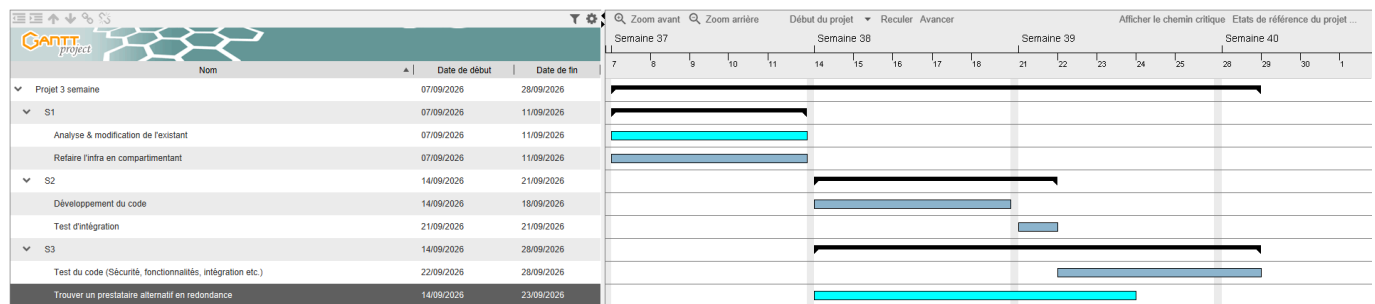
Réponse

Monsieur, Vous nous avez proposé une copie de la base de données "de démonstration" contenant les salles de spectacles partenaires actuelles. Ces données étant des données réelles, et contenant des informations sensibles et protégées par le RGPD, nous devons refuser. Les tests doivent se faire sur des données réalistes mais fictives afin d'éviter, d'une part, des fuites potentiellement néfastes pour ScèneLibre et vos partenaires, et d'autre part, une violation dans l'usage de données sensibles. Nous allons prendre le temps de générer de fausses données pour éviter des risques judiciaires et de sécurité. Cordialement,

3. Le Pentest

1. Le test se fera sur l'EKS entier (le Répartiteur de Charge) pour vérifier la résilience du WAF. Ensuite seront tentés les contournements du VPC et des connexions sur les RDS et les S3 (avec un URI absolu/scanneur d'URIs)
2. L'autorisation écrite doit mentionner le périmètre du test, les outils utilisés (même en restant vague), les méthodes (passives seulement, passives et actives, etc) et les possibilités de test (peux-tu "faire planter" l'infrastructure durant le test, ou juste remettre un rapport de faisabilité hypothétique)
3. Les conséquences d'un arrêt de la production vont dépendre du contrat. S'il est précisé et signé que c'est une possibilité, la couverture légale (contractuelle) sera suffisante. Sinon, le prestataire sera redevable pénalement de l'arrêt provoqué. Pénalités d'interruption de production, intrusion, sabotage... etc
4. Les données accessibles au public - un NSLOOKUP, les informations de registrars, les détails des pages "A propos" - sont visibles soit par obligations légales (hébergeur, siège social, directeur de la publication, SIRET, etc), mais les données de production - le code source, les identifiants, les données clients - ne doivent apparaître NULLE PART

Restitution



1. Quel est le risque numéro 1, et comment le couvrir ?

Une attaque DDOS. Un pare-feu.

2. Qu'est-ce qui bloque la mise en ligne, et qu'est-ce qui ne la bloque pas ?
3. Qu'avez-vous décidé de NE PAS tester, et pourquoi ?
4. Êtes-vous prêts à signer pour une mise en production dans trois semaines ?

Non.