

Fiche de rendu — Opération mise en production

ScèneLibre
Billetterie en ligne

Équipe : Groupe 7

Membres : Clément, Lucas G, Floren

Mission 1 — Analyse de risque

Le risque	Ce qui se passe	G	P	G×P
Serveur qui tombe sous la charge	Une tête d'affiche est mise en vente : 40 000 personnes se connectent en quelques minutes sur une infrastructure jamais testée en charge → site lent ou indisponible.	5	5	25
Survente d'une place	Deux clients achètent la même place au même moment (forte contention sur un pic) : le système ne bloque pas assez vite et vend deux fois le même siège.	5	5	25
Mot de passe du prestataire de paiement dans le dépôt Git	Les identifiants bancaires sont versionnés en clair dans le code source, accessibles à quiconque a accès au dépôt (ou à son historique).	5	4	20
Pas de séparation prod / dev / test	Une seule machine sert le développement, la démo commerciale et la production : un test ou un déploiement peut casser la démo ou la vraie vente en cours.	4	4	16
Aucun test qui mène à des bugs critiques	Le code est écrit vite, sans aucun test automatisé : un bug sur le calcul du prix, de la commission ou d'un remboursement part en production sans être détecté.	4	4	16
Fuite de données personnelles (RGPD)	Noms, adresses, dates de naissance, justificatifs de situation (demandeur d'emploi) mal protégés ou mal stockés → fuite ou traitement non conforme.	4	3	12
Fraude sur les différents tarifs	Un client détourne le tarif réduit (faux justificatif) ou abuse du remboursement en cas d'annulation → préjudice financier répété.	4	3	12
Non-conformité accessibilité	Le site, qui est un service au public, n'est pas utilisable par des personnes en situation de handicap → non-respect d'une obligation légale.	3	3	9

Si vous n'aviez le temps de tester qu'une seule chose :

Le comportement du site et de la réservation de places pendant un pic de vente massif (charge + survente réunies) : c'est le seul scénario où deux risques à 25 se déclenchent en même temps, sur le même événement, devant le plus de monde — celui qui peut à la fois coûter de l'argent réel et couler la réputation de ScèneLibre en une seule soirée.

Mission 2 — Stratégie de test

Risque n° 1 : Serveur qui tombe sous la charge

Type de test	Test de charge
Qui l'exécute	Un développeur (en rotation dans l'équipe)
Quand	Avant chaque livraison, et systématiquement avant toute mise en vente d'une grosse tête d'affiche
Automatisé ou manuel, et pourquoi	Automatisé : il doit être rejoué régulièrement (à chaque évolution du code) sans mobiliser quelqu'un à chaque fois
Critère de réussite (chiffré)	Temps de réponse < 2-3 s et 0 erreur 500 jusqu'au nombre d'utilisateurs simulés visé

Risque n° 2 : Survente d'une place

Type de test	Test unitaire (sur la fonction de réservation / décrémentation du stock)
Qui l'exécute	Le développeur qui écrit la fonctionnalité
Quand	À chaque envoi de code (en intégration continue)
Automatisé ou manuel, et pourquoi	Automatisé : un bug de concurrence est quasi impossible à détecter à la main, il faut le rejouer à chaque commit
Critère de réussite (chiffré)	Tous les cas limites passent : dernière place, stock à 0, une seule requête aboutit sur N requêtes simultanées

Risque n° 3 : Mot de passe du prestataire de paiement dans Git

Type de test	Scan de secrets (sur tout l'historique Git)
Qui l'exécute	Un développeur
Quand	À faire en urgence, immédiatement, puis à chaque commit/push ensuite
Automatisé ou manuel, et pourquoi	Automatisé : un pré-commit hook bloque tout secret avant qu'il ne parte, sans dépendre de la vigilance humaine
Critère de réussite (chiffré)	0 secret détecté dans tout l'historique du dépôt

Risque n° 4 : Pas de séparation prod / dev / test

Type de test	Test end-to-end (E2E), en environnement de recette isolé
Qui l'exécute	La QA
Quand	Avant chaque mise en production
Automatisé ou manuel, et pourquoi	Automatisé : le parcours complet doit être rejoué à l'identique avant chaque déploiement, sans erreur humaine possible
Critère de réussite (chiffré)	Paiement, remboursement intégral sous 14 jours, calcul des tarifs et de la commission : résultats exacts, sur un environnement isolé (jamais de vraie clé de paiement)

La frise — du code écrit à la mise en ligne

le dev écrit son code

- Tests unitaires (dev, à chaque commit). BLOQUANT
- Scan de secrets (automatique, à chaque push). BLOQUANT
- Revue de code par un autre développeur. BLOQUANT
- Tests d'intégration / E2E sur l'environnement de recette (QA). BLOQUANT
- Test de charge / pic (avant une grosse mise en vente uniquement). BLOQUANT pour ce type de mise en ligne, pas pour un correctif mineur
- Test manuel exploratoire de la QA sur les nouveautés — non bloquant, en parallèle

Ce que nous ne ferons PAS, et pourquoi

- Un test de charge à 40 000 utilisateurs réels sur une infrastructure à l'échelle de la prod, à chaque déploiement : trop coûteux à l'année pour une start-up — on le réserve à quelques campagnes ciblées avant les plus grosses mises en vente (voir Mission 4c).
- Une couverture de test à 100 % du code existant avant la mise en ligne : avec 4 développeurs et zéro test au départ, ce serait irréaliste dans les délais — on cible en priorité les zones qui touchent à l'argent réel (paiement, remboursement, commission) et à l'attribution des places.
- Des tests manuels exhaustifs sur toutes les valeurs d'âge ou de quantité de places : inutile, les partitions d'équivalence (Mission 3a) montrent qu'un petit nombre de cas suffit à couvrir tous les comportements possibles.

Mission 3 — Conception des cas de test

3a. Le tarif réduit — moins de 26 ans

Famille	Exemple	Résultat attendu
Nettement en dessous	15 ans	Éligible
Juste en dessous (limite – 1 jour)	25 ans et 364 jours	Éligible
Sur la limite exacte	26 ans, jour J	Non éligible — la règle dit « moins de »
Juste au-dessus	26 ans et 1 jour	Non éligible
Nettement au-dessus	40 ans	Non éligible
Valeur invalide	Âge négatif (date de naissance dans le futur)	Rejeté / erreur de saisie

3a bis. La limite de 6 places

Famille	Exemple	Résultat attendu
Valeur nulle	0	Refusé
Nettement en dessous	4	Accepté
Juste en dessous (limite – 1)	5	Accepté

Sur la limite exacte	6	Accepté — la limite est incluse
Juste au-dessus (limite + 1)	7	Refusé
Très grand nombre	50	Refusé
Valeur non valide	1,5 (flottant)	Refusé, message d'erreur
Cumul sur plusieurs commandes	4 achetées, puis nouvelle commande de 3 pour le même spectacle	À clarifier avec le PO : la règle porte-t-elle sur une commande ou sur le client cumulé ?

3b. Table de décision

Moins de 26 ans ?	Justificatif fourni ?	Justificatif valide ?	Tarif appliqué	Statut
Oui	Oui	Oui	Réduit	À tester
Oui	Oui	Non	Plein	À tester
Oui	Non	—	Plein	À tester
Non	Oui	Oui	Plein *	À tester
Non	Oui	Non	Plein	À tester
Non	Non	—	Plein	À tester
Oui	Non	Oui	Cas qui n'existe pas	Test de robustesse
Non	Non	Oui	Cas qui n'existe pas (idem)	Test de robustesse

3c. Les 10 cas que personne ne prévoit

- Deux onglets ouverts par le même client, chacun avec un panier de 5 places.
La limite de 6 places est-elle vérifiée par panier ou par client, en temps réel côté serveur ?
- Le justificatif est flou, dans une langue étrangère, ou un fichier corrompu.
Qui le vérifie, qui le valide, et sous quel délai avant la fermeture des ventes ?
- Le spectacle est reporté (nouvelle date), pas annulé.
La règle ne parle que de l'annulation : quel recours pour un client indisponible à la nouvelle date ?
- Coupure réseau : le paiement est accepté par la banque mais le serveur ne reçoit jamais la confirmation.
Le client est débité sans billet. La place reste-t-elle bloquée, ou est-elle relâchée et revendue à quelqu'un d'autre ?
- La salle modifie le prix ou la catégorie d'une place pendant qu'un client finalise son paiement.
Quel prix est appliqué ? Ou la transaction est-elle annulée automatiquement si le prix change en cours de route ?
- Le nom du client contient des caractères spéciaux (accents, apostrophe, nom composé).
Impression du billet, code-barres et base de données gèrent-ils correctement un cas pourtant très courant ?
- Demande de remboursement d'une seule place sur un panier de 6.
La gestion d'une annulation partielle est-elle prévue, ou seule la commande entière peut être remboursée ?

8. Le spectacle est annulé après avoir déjà commencé.

Ce cas rentre-t-il dans la règle de remboursement, alors que le spectacle a bel et bien eu lieu un peu ?

9. L'horaire du spectacle est avancé à la dernière minute.

La fermeture des ventes « 2h avant » est-elle recalculée automatiquement, ou reste-t-elle calée sur l'ancien horaire ?

10. Le client clique sur « Payer », revient en arrière avec le navigateur, puis reclique sur « Payer ».

Le client est-il débité deux fois pour la même commande ?

Mission 4 — Les environnements

4a. Combien d'environnements nous faut-il ?

	Env. 1 — Production	Env. 2 — Préprod / Démo	Env. 3 — Développement	Env. 4 — Test / Recette
Nom et rôle	Le vrai site utilisé par les clients et les salles	Montrer un produit quasi final aux prospects	Écrire et tester du code au fil de l'eau	Valider une fonctionnalité une fois assemblée, avant la prod
Qui y accède	Clients finaux, salles partenaires ; devs en accès contrôlé et journalisé	Équipe commerciale + prospect	Le développeur, seul	Les 4 développeurs + la QA
Quelles données	Réelles	Inventées (catalogue vitrine, figé)	Inventées (jeu de données factice minimal)	Anonymisées ou inventées, jamais réelles
Dimensionné comment	Pour le trafic réel + marge de sécurité pour les pics	Très léger, juste besoin d'être stable et rapide	Ressources minimales (poste local)	Petite VM, proportionnée à l'usage réel de l'équipe
Durée de vie	Permanent	Permanent, contenu figé	Permanent, réinitialisable à volonté	Permanent, remis à zéro régulièrement
Ce qu'on y fait	Uniquement des déploiements déjà validés en recette	Démonstrations commerciales	Écriture de code, tests unitaires, débogage	Tests d'intégration, E2E, non-régression, recette
Ce qu'on n'y fait PAS	Aucun test, aucune expérimentation	Aucun développement, aucun test, jamais de vrai paiement	Aucun test de charge, jamais de vraie clé de paiement, jamais de démo	Démo commerciale, test de charge à pleine échelle, vrais paiements

Le jeu de données

Combien de spectacles, clients, commandes ?

- Spectacles : quelques centaines (≈ 200-300), avec une distribution réaliste — beaucoup de spectacles à faible affluence et une poignée de « grosses têtes d'affiche » à forte contention, pour reproduire le vrai scénario de pic.

- Catégories de places par spectacle : 3 à 5, pour exercer aussi la logique de tarification et de quotas, pas seulement le volume.
- Clients : au moins 50 000 à 100 000 comptes simulés, pour dépasser largement le pic ciblé de 40 000 et garder de la marge.
- Commandes / historique : plusieurs centaines de milliers, pour tester la performance des recherches, du suivi des ventes et de la facturation sur un volume comparable à une année réelle d'activité.

D'où viennent ces données ? Pourquoi pas la production ?

Génération synthétique par script, jamais copie de la production, pour trois raisons :

- RGPD : les données de prod contiennent noms, adresses, dates de naissance, justificatifs de situation. Les réutiliser hors de leur finalité d'origine, même « pour tester », est un traitement non autorisé, sanctionnable par la CNIL.
- Paiement : toute donnée liée à un paiement réel ne doit jamais réapparaître dans un environnement moins sécurisé que la production.
- Risque d'action réelle involontaire : un test rejouant des commandes réelles pourrait déclencher de vrais e-mails, voire de vraies actions chez le prestataire de paiement si l'isolation n'est pas parfaite.

Le script de génération peut s'inspirer de statistiques publiques du secteur (répartition d'âge des acheteurs, par exemple) pour rester réaliste, sans jamais copier un enregistrement individuel réel.

Quels cas tordus y placer volontairement ?

- Des clients ayant exactement 26 ans le jour du test.
- Des commandes à exactement 6 places, et des tentatives à 7.
- Des places en tout dernier exemplaire disponible, pour déclencher la contention pendant le test de charge.
- Des justificatifs invalides, corrompus, ou dans un format inattendu.
- Des noms avec caractères spéciaux, accents, noms très longs.
- Des spectacles dont la fermeture des ventes (2h avant) tombe pile pendant la fenêtre du test.
- Des doublons volontaires (même client, même e-mail, casse différente) pour tester la déduplication.

Comment garantir qu'un test qui échoue soit rejouable à l'identique ?

- Jeu de données déterministe : le script de génération utilise une graine aléatoire (seed) fixe. Relancer le produit, exactement les mêmes spectacles, clients, commandes.
- Snapshot de la base avant chaque campagne de test, restauré avant de rejouer, plutôt que de repartir d'un état « à peu près pareil ».
- Scripts de charge eux-mêmes déterministes et versionnés dans le dépôt (sans jamais y inclure de secret).
- Horodatage et logs exhaustifs pendant le test, pour analyser un échec sans devoir le rejouer une deuxième fois pour comprendre.

Le budget

Comment tester la charge sans payer un deuxième site à l'année ?

- Infrastructure cloud à la demande : louer la puissance nécessaire uniquement pendant la fenêtre du test, puis l'éteindre — quelques heures par mois, pas 365 jours d'un serveur surdimensionné.

- Tests dégressifs + extrapolation : tester à échelle réduite (ex. 10 % du trafic cible), trouver le point de rupture, puis extrapoler mathématiquement plutôt que payer systématiquement l'infrastructure à pleine échelle.
- Outils facturés à l'usage (k6 Cloud, Gatling Enterprise...) plutôt qu'un abonnement fixe à l'année.
- Mutualiser l'environnement de recette : le surdimensionner temporairement juste avant un test de charge planifié, puis le redescendre — au lieu de maintenir un 5^e environnement dédié en permanence.
- Comparer le coût du test à celui d'un échec réel : un budget de test de charge se justifie facilement face au coût d'un plantage le jour d'une grosse tête d'affiche (ventes perdues, remboursements, image, partenaires qui doutent).

Mission 5 — Sécurité et conformité

5a. Les trois urgences

La situation	Le scénario d'attaque	La correction	Délai
1. Mot de passe du prestataire de paiement dans le dépôt Git	Un attaquant récupère un accès en lecture au dépôt, consultation de l'historique des commits et peut donc retrouver des traces de secrets ou même de fichiers actuels	Révoquer et régénérer les identifiants auprès du prestataire et purger l'historique de Git	Immédiat (24h)
2. Une seule machine sert de serveur développement de et démo commerciale	Si un attaquant accède à l'environnement de dev alors il accède à tous les environnements, donc les données réelles sont compromises et les clés utilisées si elles sont stockées sur cette machine	Séparer physiquement et logiquement les différents environnements, isoler les accès par clés et fermer les ports exposés sur internet	Court terme (1 semaine)
3. Code écrit vite, sans aucun test automatisé sur les règles métiers sensibles	Sans passer par le web l'attaquant peut directement envoyer des requêtes construites à la main vers le serveur	Vérifier et tester que chaque règle métier sensible soit bien contrôlé côté serveur.	Avant la prochaine mise en ligne

5b. Votre réponse au directeur commercial

On ne peut pas utiliser les données réelles des clients pour les tests car ces données sont protégées par la loi et les réutiliser en dehors de l'usage prévu est interdit sans l'accord explicite des personnes concernées.

Si un test tourne mal c'est directement la confiance des clients, des salles partenaires qui est en jeu en plus du risque financier.

Cependant, nous pouvons créer des jeux de données factices tout aussi réaliste qui pourront nous permettre d'effectuer nos tests sans aucun risque juridique.

5c. Le cadrage du pentest

Sur quel environnement, et pourquoi pas un autre : Sur l'environnement de préprod, copie la plus fidèle de ce qui est en prod mais avec des données factices.

Ce que doit contenir l'autorisation écrite : Le périmètre exact, les dates et horaires d'intervention, les types de tests, un contact d'urgence de chaque côté (client/presta), clause de confidentialité et une mention explicite qui autorise légalement l'intrusion.

Que faire si le pentester casse quelque chose : Le fait que le pentest se déroule sur un environnement de préprod permet de ne rien casser de sensible, un simple snapshot restaurable est à anticiper.

Quelles données doivent s'y trouver / lesquelles surtout pas : Les données générées doivent s'y trouver et les données réelles elles ne doivent pas être présentes.