Atelier de professionnalisation – BTS SIO2 (2025-2026)
Semestre 1

Projet EventHub Lurçat

Table des matières

1. Fiche d'identité du projet

Le projet proposé pour ce semestre, intitulé "**EventHub Lurçat**", s'inscrit dans le cadre d'une simulation professionnelle. Il vise à répondre à un besoin identifié au sein du lycée, pour lequel l'**Administration du Lycée Jean Lurçat** tiendra le rôle de client de référence.

Sa réalisation est confiée aux étudiants de deuxième année du BTS SIO, qui devront travailler en étroite collaboration. L'équipe de la spécialité **SLAM** aura la charge du développement applicatif, tandis que l'équipe **SISR** assurera la conception, la mise en place et la sécurisation de l'infrastructure réseau et système.

Le projet se déroulera sur l'ensemble du premier semestre de l'année scolaire, de **septembre 2025 à décembre 2025** inclus, sous la supervision de **M. LEGRAND Cédric** qui agira en tant que chef de projet référent, avec le soutien de **M. MARTINEZ Manuel** et **M. RIBEIRO Paul**.

L'objectif principal est de livrer un *Proof of Concept* (PoC) abouti : une plateforme web **fonctionnelle, sécurisée et entièrement documentée**, destinée à moderniser la gestion des événements internes et la réservation des ressources matérielles et physiques du lycée.

2. Cahier des charges fonctionnel (Mission de l'équipe SLAM)

La mission confiée à l'équipe de spécialité SLAM consiste en la **conception** et le développement de l'application web "**EventHub Lurçat**". L'étape de conception, qui devra être formalisée par des diagrammes UML/Merise, est tout aussi importante que le développement lui-même. Il est attendu un travail de qualité professionnelle, respectant scrupuleusement l'ensemble des exigences fonctionnelles et des impératifs de sécurité qui seront détaillés dans les sections suivantes.

Pour garantir la cohérence, la sécurité et la maintenabilité du projet, le développement devra s'appuyer sur un périmètre technologique défini et imposé. L'architecture applicative reposera sur le langage **PHP (version 8.3 ou supérieure)** et le framework **Laravel (version 12)**. La persistance des données sera assurée par un système de gestion de base de données **PostgreSQL**, dont l'infrastructure sera fournie et maintenue par l'équipe SISR.

Enfin, pour l'interface utilisateur, les technologies standards du web (**HTML5, CSS3, JavaScript**) seront mises en œuvre. L'utilisation d'un framework CSS tel que **Bootstrap 5** ou **Tailwind CSS** est autorisée et vivement recommandée afin d'accélérer le développement et d'assurer une expérience utilisateur moderne et responsive.

Un projet à deux niveaux

Nous sommes conscients que le périmètre fonctionnel de l'application "**EventHub Lurçat**" est riche et complet. Pour vous permettre de piloter votre projet de manière réaliste et de sécuriser un livrable de haute qualité, nous vous proposons de structurer votre travail en deux périmètres :

- **Le Périmètre essentiel ou MVP (Minimum Viable Product) :** C'est le cœur non négociable du projet. Il contient toutes les fonctionnalités fondamentales qui rendent l'application viable et sécurisée. La maîtrise parfaite de ce périmètre est l'objectif principal et garantit la validation des compétences clés de cet atelier. Il inclut :
    - La gestion complète des utilisateurs (3 rôles) et l'authentification sécurisée (mots de passe hachés).
    - Le CRUD complet pour les salles et les matériels.
    - La création d'événements avec la **logique de contrôle de conflit**, qui est un point critique.
    - La visualisation en calendrier et le système d'inscription/désinscription.
    - L'ensemble des exigences de sécurité applicative (anti SQLi, XSS, CSRF, RBAC).
    - La modélisation (UML/Merise) et la documentation associées.

- **Le projet complet :** Ce périmètre inclut des fonctionnalités plus complexes, souvent à la frontière avec l'infrastructure, qui démontrent une plus grande maturité technique. S'attaquer à ces points est l'opportunité de vous confronter à des défis d'intégration avancés et de viser l'excellence. Il s'agit d'ajouter au périmètre essentiel :
    - **L'intégration avec l'annuaire Active Directory** pour l'authentification des utilisateurs (nécessite une collaboration poussée avec les SISR).
    - Le système de **rappel automatique par courriel** (qui dépend du `cron` configuré par les SISR).
    - La gestion des **pièces jointes** (téléversement de fichiers).
    - L'implémentation d'un **éditeur de texte riche (WYSIWYG)** pour la description des événements.

Ce choix de périmètre vous appartient. Livrer un MVP parfaitement fonctionnel, sécurisé et documenté est déjà un gage de grande réussite. Le compléter avec les éléments du projet complet démontre une volonté et une capacité à dépasser les attentes qui seront particulièrement valorisées dans l'évaluation finale.

2.1 Gestion des utilisateurs

Le cœur de l'application "**EventHub Lurçat**" repose sur une gestion fine des utilisateurs, garantissant à la fois la sécurité des accès et une expérience adaptée à chaque profil. Le système devra implémenter un contrôle d'accès basé sur les rôles (RBAC) et un processus d'authentification sécurisé.

Rôles et permissions

Trois profils d'utilisateurs distincts coexisteront au sein de la plateforme, chacun disposant de permissions spécifiques :

- **Administrateur (ADMIN) :** Ce rôle dispose des privilèges les plus élevés, lui donnant un accès total à l'application. Cela inclut l'accès à une section privée et sécurisée de la plateforme : le **panel d'administration**. Contrairement à l'interface visible par tous (le calendrier, les fiches d'événements), ce panel est l'espace de gestion "en coulisses" de l'application, invisible pour les autres rôles. Depuis cette interface de contrôle, l'administrateur est le seul à pouvoir :

- **Gérer l'ensemble des utilisateurs :** Il visualise la liste complète, valide les nouvelles inscriptions en attente, peut modifier un rôle ou désactiver un compte.
- **Gérer l'inventaire des ressources :** Il effectue toutes les opérations de création, modification et suppression des salles et des matériels disponibles à la réservation.
- **Superviser tous les événements :** Il possède une vue d'ensemble et peut, si nécessaire, modifier ou annuler n'importe quel événement créé sur la plateforme.
- **Consulter des statistiques d'utilisation** via un tableau de bord. En somme, l'administrateur est le seul à pouvoir à la fois utiliser la plateforme comme un utilisateur standard et en piloter l'intégralité du contenu et des utilisateurs via cette interface de gestion dédiée. L'attribution de ce rôle se fera manuellement en base de données pour des raisons de sécurité.

- **Enseignant (ENSEIGNANT) :** Ce profil est destiné aux organisateurs d'événements. Un enseignant peut créer, modifier et supprimer ses propres événements, réserver les ressources nécessaires à leur tenue, et gérer les inscriptions des étudiants qui y sont associées.
- **Étudiant (ÉTUDIANT) :** Ce profil joue un rôle consultatif et participatif. Les étudiants peuvent naviguer dans le calendrier des événements, consulter les détails et s'inscrire à ceux qui sont ouverts au public. Ils ne disposent pas des droits de création d'événement ou de réservation directe de ressources.

Processus d'authentification et cycle de vie du compte

L'accès à la plateforme sera conditionné par un processus d'authentification robuste :

- **Inscription et validation :** Un formulaire d'inscription sera disponible pour les enseignants et les étudiants. À la soumission, chaque nouveau compte sera créé avec un statut "inactif". L'activation du compte n'est pas automatique ; elle requiert une action manuelle de la part d'un administrateur **depuis le panel de gestion**, garantissant ainsi un contrôle sur les utilisateurs accédant à la plateforme.

- **Connexion :** Une fois le compte activé, l'utilisateur pourra se connecter via un formulaire standard utilisant son adresse électronique et son mot de passe.
- **Sécurité des mots de passe :** La protection des identifiants est une exigence non négociable. Tous les mots de passe devront être hachés avant leur stockage en base de données. L'utilisation d'un algorithme de hachage moderne et résistant, tel qu'**Argon2id** (à privilégier) ou à défaut **Bcrypt**, est impérative.
- **Gestion du profil :** Chaque utilisateur connecté devra avoir accès à une page de profil personnelle, lui offrant la possibilité de mettre à jour des informations de base comme son nom et, surtout, de modifier son mot de passe en toute autonomie.

2.2 Gestion des ressources (CRUD via panel admin)

Le panel d'administration doit fournir à l'administrateur un contrôle total sur l'inventaire des ressources physiques et matérielles du lycée. Cette gestion est la pierre angulaire de la plateforme, car elle conditionne l'ensemble du système de réservation.

L'administrateur doit pouvoir effectuer l'ensemble des opérations de Création, Lecture, Mise à jour et Suppression (CRUD) via des interfaces dédiées, claires et efficaces pour deux entités distinctes: les salles et les matériels.

2.2.1 Module de gestion des salles

Ce module est dédié à l'inventaire et à la configuration des espaces physiques (salles de cours, amphithéâtres, salles de réunion, etc.) pouvant être réservés.

Champs de données à gérer :

- `id` : Identifiant numérique unique, généré automatiquement (clé primaire).
- `nom` : Nom explicite de la salle (ex: "Amphithéâtre A", "Salle Info 203").
- `batiment` : Champ texte pour indiquer la localisation précise dans le lycée (ex: "Bâtiment B, 2ème étage").
- `capacite` : Nombre entier représentant le nombre de places assises.
- `equipements` : Une liste à choix multiples d'équipements prédéfinis. L'interface doit permettre de cocher les options suivantes :
    - Vidéoprojecteur
    - Tableau blanc interactif
    - Système audio
    - Ordinateurs (si coché, un champ numérique doit apparaître pour préciser le nombre de postes)
    - Connexion visioconférence
    - Climatisation

- `disponible` : Un booléen (actif/inactif) permettant de retirer temporairement une salle de la réservation (ex: en cas de travaux) sans la supprimer de la base de données.
- `photo` : Champ de téléversement d'image, optionnel, pour une photo de la salle (contraintes : max 2MB, formats JPG/PNG).
- `remarques` : Champ de texte libre pour toute information complémentaire utile (ex: "Clé à récupérer à l'accueil").

2.2.2 Module de gestion des matériels

Ce module centralise la gestion de l'ensemble du matériel empruntable (informatique, audiovisuel, etc.) qui peut être associé à un événement.

Champs de données à gérer :

- `id` : Identifiant numérique unique, généré automatiquement (clé primaire).
- `nom` : Désignation claire du matériel (ex: "PC Portable Dell #01", "Micro-cravate Sennheiser").
- `categorie` : Une liste de sélection (menu déroulant) parmi les options : Informatique / Audiovisuel / Mobilier / Autre.
- `quantite_totale` : Le stock total disponible pour ce type de matériel.
- `reference` : Le numéro d'inventaire ou de série unique pour le suivi.
- `etat` : Un champ de sélection indiquant l'état actuel du matériel : Neuf / Bon / Moyen / À réparer.
- `description` : Champ de texte libre pour les spécifications techniques ou des détails pertinents.
- `localisation_storage` : Champ texte indiquant le lieu de rangement habituel (ex: "Armoire C34, salle des serveurs").
- `disponible` : Un booléen (actif/inactif) pour retirer temporairement un matériel du circuit de réservation.

2.2.3 Exigences sur les opérations de gestion (CRUD)

Au-delà de la simple définition des données, la qualité des interfaces de gestion est primordiale et doit respecter les standards suivants pour les deux modules :

- **CREATE (Créer) :** L'ajout d'une nouvelle ressource (salle ou matériel) doit se faire via un formulaire clair, intégrant une **validation robuste des données** côté serveur avant tout enregistrement en base de données (ex: la capacité doit être un nombre positif, le format de la photo doit être respecté).
- **READ (Lire) :** L'affichage de la liste des ressources doit être présenté sous forme de **liste paginée** (par exemple, 20 éléments par page) pour garantir de bonnes performances. Cette vue doit impérativement inclure des **fonctionnalités de recherche** (par nom) et de **filtrage** (par bâtiment/catégorie, par état, par disponibilité).
- **UPDATE (Mettre à jour) :** La modification d'une ressource existante doit être simple et intuitive. Une attention particulière sera portée à la traçabilité : l'implémentation d'un **historique des changements** majeurs est un plus apprécié.
- **DELETE (Supprimer) :** La suppression d'une ressource ne doit pas être une suppression physique de la base de données. Il est impératif d'implémenter une **suppression logique (soft delete)**. Concrètement, la ressource est simplement marquée comme "supprimée" (par exemple, via un champ deleted_at) mais reste présente en base. Cette approche est cruciale pour **conserver l'intégrité des données historiques** et ne pas corrompre les anciens événements qui étaient liés à cette ressource.

2.3 Gestion des évènements

La gestion des événements constitue la fonctionnalité centrale de l'application. Elle doit permettre aux administrateurs et aux enseignants de planifier, de publier et de gérer des activités, tout en offrant une consultation claire et intuitive à l'ensemble des utilisateurs.

2.3.1 Structure de l'entité "Événement"

Chaque événement enregistré dans la base de données doit être structuré selon un modèle de données précis, distinguant les informations obligatoires des champs optionnels qui enrichissent l'information.

Données obligatoires pour la création d'un événement :

- `titre` : Le nom de l'événement, limité à un maximum de **100 caractères**.
- `description` : Une description détaillée du contenu de l'événement. Ce champ doit impérativement être un champ de **texte riche**, ce qui implique la mise en place d'un éditeur de type WYSIWYG (What You See Is What You Get) côté formulaire. Une longueur minimale de **50 caractères** est requise.
- `date_debut / date_fin` : Les dates et heures précises de début et de fin de l'événement (format datetime).
- `salle_id` : La référence (clé étrangère) à la **salle unique** réservée pour l'événement.
- `organisateur_id` : La référence (clé étrangère) à l'utilisateur (Admin ou Enseignant) qui a créé l'événement. Ce champ doit être rempli automatiquement avec l'ID de l'utilisateur connecté.
- `type_evenement` : Une catégorie permettant de qualifier l'événement, à choisir dans une liste prédéfinie : Conférence / Atelier / Réunion / Formation / Autre.

- `public_cible` : Le public visé par l'événement, à choisir dans une liste : Étudiants / Enseignants / Tous / Sur invitation.

Données optionnelles pouvant être associées à un événement :

- `materiels` : La liste des matériels nécessaires. Cette fonctionnalité requiert une **relation de type 'plusieurs-à-plusieurs'** entre les événements et les matériels, avec une information complémentaire sur la **quantité** requise pour chaque matériel sélectionné.
- `nb_places_max` : Un nombre entier pour limiter le nombre d'inscriptions. Si ce champ est laissé vide (valeur NULL), le nombre de places est considéré comme illimité.
- `inscription_requise` : Un booléen (vrai/faux) indiquant si les étudiants peuvent s'inscrire à l'événement. Ce champ correspond à l'option "Autoriser les inscriptions" dans le formulaire.
- `pieces_jointes` : Une fonctionnalité de téléversement de documents. Elle doit permettre d'attacher jusqu'à **3 fichiers au format PDF**, avec une limite de poids de **5 Mo par fichier**.
- `lien_visio` : Un champ pour insérer une URL, destiné aux événements hybrides ou en ligne (ex: lien Microsoft Teams, Zoom, etc.).

2.3.2 Processus de création et contrôle de conflit

La création d'un événement, réservée aux administrateurs et aux enseignants, doit se faire via un formulaire complet reprenant l'ensemble des champs décrits précédemment. Le point le plus critique et le plus complexe de ce processus est la gestion des conflits de réservation : le système doit **impérativement et sans faille empêcher la réservation d'une salle ou d'un matériel si celui-ci est déjà réservé sur une plage horaire qui se chevauche**.

Cette vérification doit s'appliquer aussi bien lors de la **création** d'un nouvel événement que lors de la **modification** d'un événement existant. En cas de conflit, l'enregistrement doit être bloqué et un message d'erreur clair et explicite doit être affiché à l'utilisateur, lui indiquant idéalement quelle ressource est à l'origine du conflit.

La logique de contrôle de conflit est un défi technique majeur dont la réussite est non négociable. Une implémentation approximative rendrait l'application inutilisable. Pour vous guider, voici une approche structurée et professionnelle à mettre en œuvre.

- **Conseil n°1 : Localiser la logique au bon endroit.**
  La vérification doit s'exécuter côté serveur, car la base de données est la seule source de vérité. Dans un projet Laravel, la meilleure pratique consiste à encapsuler cette logique dans une **classe de Form Request personnalisée**. Cette approche permet de séparer la logique de validation de celle de votre contrôleur, ce qui rend le code plus propre, plus testable et plus facile à maintenir.

- **Conseil n°2 : Maîtriser l'algorithme de chevauchement des dates.**
  Le cœur du problème est de déterminer si deux plages horaires se chevauchent. Deux intervalles de temps (un existant `[Debut_A, Fin_A]` et un nouveau `[Debut_B, Fin_B]`) se chevauchent si et seulement si la condition suivante est vraie : `(Debut_A < Fin_B) ET (Fin_A > Debut_B)`. Cette condition unique couvre tous les cas de figure possibles et doit être la base de votre requête en base de données.

- **Conseil n°3 : Construire une requête Eloquent efficace.**
  Votre objectif est de savoir s'il existe **au moins un** événement en conflit. Une requête efficace (`->exists()`) est préférable à une requête qui récupérerait une liste de conflits (`->get()`). Votre requête devra rechercher un événement qui remplit **toutes** les conditions suivantes :

  1. Il n'est **pas l'événement que vous êtes en train de modifier** (crucial pour la fonction de mise à jour, pour éviter qu'un événement n'entre en conflit avec lui-même).
  2. Il utilise **la même salle OU au moins un des mêmes matériels**. Cela implique une clause `WHERE` complexe, probablement avec une fonction de rappel (`->where(function ($query) { ... })`) pour grouper les conditions `OR` correctement.
  3. Sa plage horaire se chevauche avec celle du nouvel événement, en appliquant la logique du Conseil n°2 dans vos clauses `WHERE`.

- **Conseil n°4 : Implémenter une Règle de Validation Personnalisée (la "Laravel Way").**
  Pour une intégration parfaite et réutilisable, la meilleure approche est de créer votre propre Règle de Validation.

  1. Générez une nouvelle règle via la commande Artisan : `php artisan make:rule NoConflictRule`.
  2. Dans la méthode `passes()` de cette nouvelle classe, vous implémenterez la logique de requête décrite au Conseil n°3. La méthode retournera `false` s'il y a un conflit, et `true` sinon.
  3. Dans la méthode `message()`, vous définirez le message d'erreur qui sera renvoyé à l'utilisateur en cas d'échec de la validation.
  4. Enfin, vous utiliserez cette nouvelle règle (`new NoConflictRule`) directement dans les règles de validation de votre `Form Request`.

En suivant cette démarche, vous construirez une fonctionnalité de validation qui est non seulement fonctionnelle et sécurisée, mais aussi élégante, maintenable et respectueuse des bonnes pratiques du framework Laravel. C'est un excellent exercice pour démontrer une compétence de développement professionnelle.

2.3.3 Visualisation des événements

L'accès à l'information sur les événements doit être centralisé et intuitif :

- **Vue Calendrier :** L'interface principale de consultation sera une page "Calendrier" affichant tous les événements à venir. Pour répondre aux besoins des utilisateurs, cette vue devra proposer, au minimum, un affichage **mensuel** et un affichage **hebdomadaire**. L'intégration d'une librairie JavaScript spécialisée comme **FullCalendar.js** est une piste technique fortement recommandée pour atteindre cet objectif.
- **Détails de l'événement :** Un clic sur un événement dans le calendrier ne doit pas recharger toute la page. Il doit déclencher l'affichage de ses informations détaillées, soit dans une **fenêtre modale**, soit sur une page dédiée. Cette vue de détail présentera l'ensemble des informations de l'événement : titre, description formatée, organisateur, pièces jointes téléchargeables, etc.

2.4 Système de reservation et de notification

Pour que "EventHub Lurçat" soit une plateforme vivante et non un simple calendrier passif, elle doit intégrer un système d'interaction robuste permettant aux utilisateurs de s'engager avec les événements. Ce système repose sur deux piliers : la gestion des inscriptions par les étudiants et un mécanisme de notifications automatisées par courriel. La mise en œuvre de cette dernière fonctionnalité constitue un point de collaboration essentiel avec l'équipe SISR.

2.4.1 Interaction utilisateur : inscription et désinscription

La plateforme doit offrir aux étudiants la possibilité de s'inscrire aux événements qui les intéressent. Cette fonctionnalité doit respecter les conditions suivantes :

- **Processus d'inscription :** Pour tout événement où les inscriptions sont autorisées (selon les options `inscription_requise` et `nb_places_max`), un étudiant connecté verra un bouton d'action clair, tel que "S'inscrire". Un clic sur ce bouton enregistrera sa participation.
- **Gestion de la capacité :** Le système doit gérer la limite de places. Si le nombre maximal d'inscrits (`nb_places_max`) est atteint, le bouton d'inscription doit être désactivé ou masqué pour empêcher toute inscription supplémentaire.
- **Processus de désinscription :** L'engagement doit être flexible. Un étudiant déjà inscrit doit avoir la possibilité d'annuler sa participation via un bouton équivalent ("Se désinscrire"), libérant ainsi sa place pour un autre utilisateur.

2.4.2 Communication automatisée : notifications par courriel

Pour assurer une communication fluide et professionnelle, l'application doit être capable d'envoyer des courriels transactionnels et planifiés. Cette fonctionnalité dépend entièrement de la bonne configuration du serveur mail qui sera mis en place par l'équipe SISR. Une communication parfaite entre les deux équipes sur les paramètres SMTP (hôte, port, identifiants) est donc **non-négociable** pour la réussite de ce module.

Deux types de notifications sont requis :

- **Notification en temps réel à l'organisateur :**
  Afin de fournir un retour immédiat, le système déclenchera automatiquement l'envoi d'un courriel à l'enseignant ou à l'administrateur organisateur **à chaque fois qu'un nouvel utilisateur s'inscrit à son événement**. Ce courriel de notification permettra à l'organisateur de suivre en temps réel l'engouement pour son événement.
- **Rappel automatisé pour les participants (J-1) :**
  Pour minimiser les absences et assurer une bonne participation, un mécanisme de rappel automatique est impératif. Ceci sera réalisé via une **tâche planifiée (Scheduled Task) de Laravel**. Cette tâche, configurée pour s'exécuter une fois par jour, aura pour mission de :
  1. Identifier tous les événements qui débuteront dans les prochaines 24 heures.
  2. Envoyer un courriel de rappel concis et informatif à tous les participants inscrits à ces événements.

Note technique importante : Le déclenchement de cette tâche planifiée Laravel repose sur la configuration d'une entrée **cron** sur le serveur web Linux. L'équipe SISR sera donc responsable de la mise en place de ce cron, qui exécutera la commande `php artisan schedule:run` chaque minute, comme le préconise la documentation officielle de Laravel.

2.5 Panel d’administration

Le panel d'administration constitue le centre de contrôle sécurisé de l'application, une zone exclusivement réservée aux utilisateurs disposant du rôle "ADMIN". Son développement doit être particulièrement soigné, car il garantit la bonne gouvernance de la plateforme. Il doit être structuré autour de trois fonctionnalités principales :

- **Tableau de bord synthétique (Dashboard) :** Dès la connexion au panel, l'administrateur doit être accueilli par une page d'accueil offrant une vue d'ensemble rapide et pertinente de l'activité de la plateforme. Ce tableau de bord doit présenter des indicateurs de performance clés (KPIs), tels que le nombre total d'utilisateurs enregistrés, le volume d'événements programmés pour le mois en cours, ainsi qu'un aperçu des ressources (salles ou matériels) les plus fréquemment réservées.
- **Interfaces de gestion (CRUD) :** C'est le cœur opérationnel du panel. Des interfaces de gestion dédiées doivent être développées pour permettre un contrôle total (CRUD : Créer, Lire, Mettre à jour, Supprimer) sur toutes les entités principales de l'application. Cela inclut des vues tabulaires claires et fonctionnelles pour gérer entièrement les utilisateurs, les salles, les matériels et l'ensemble des événements, comme détaillé dans les sections précédentes.
- **Workflow de validation des inscriptions :** Pour assurer la sécurité et le contrôle des accès, le panel doit comporter une section dédiée à la modération des nouveaux comptes. L'administrateur doit y voir une liste claire des utilisateurs en attente de validation et pouvoir activer leurs comptes en un clic, finalisant ainsi leur processus d'inscription.

2.6 Exigences de Cybersécurité (niveau applicatif)

La sécurité n'est pas une option, mais un prérequis fondamental de ce projet. Le respect scrupuleux des points suivants est **non négociable** et fera l'objet d'une évaluation spécifique et rigoureuse. L'utilisation du framework Laravel ne dispense pas d'une compréhension et d'une application correctes de ses mécanismes de protection.

- **Protection contre les injections SQL :** La base de la sécurité des données repose sur la prévention des injections SQL. L'utilisation systématique de l'ORM **Eloquent** et de ses mécanismes de *prepared statements* est obligatoire. **Aucune requête SQL brute concaténée avec des entrées utilisateur n'est autorisée dans ce projet.**
- **Protection contre le Cross-Site Scripting (XSS) :** Pour empêcher des acteurs malveillants d'injecter du code malveillant dans les pages vues par d'autres utilisateurs, toutes les données provenant de la base de données ou des utilisateurs doivent être systématiquement échappées lors de leur affichage. Le moteur de template Blade de Laravel le fait par défaut via la syntaxe `{{ $variable }}`, qui doit être utilisée en toutes circonstances pour l'affichage de données.
- **Protection contre le Cross-Site Request Forgery (CSRF) :** Pour protéger les utilisateurs authentifiés contre les attaques visant à leur faire exécuter des actions à leur insu, l'utilisation du middleware de protection CSRF natif de Laravel est impérative. Le token `@csrf` doit être inclus dans tous les formulaires qui modifient l'état de l'application (toutes les requêtes de type `POST`, `PUT`, `PATCH`, `DELETE`).
- **Validation rigoureuse des entrées :** Le principe fondamental "Ne jamais faire confiance aux données de l'utilisateur" doit être appliqué sans exception. Une validation robuste et stricte de **toutes** les données reçues de formulaires ou d'autres sources doit être effectuée **côté serveur**. L'utilisation des classes de **`Form Requests`** ou du `Validator` de Laravel est la méthode imposée pour réaliser cela.

- **Contrôle d'accès basé sur les rôles (RBAC) :** Le système de rôles (Admin, Enseignant, Étudiant) ne serait rien sans un contrôle d'accès strict qui en assure l'application. Il est impératif de protéger les routes et les fonctionnalités sensibles. Un étudiant ne doit, par exemple, jamais pouvoir accéder à une URL du panel d'administration, même en la tapant manuellement. L'implémentation de cette logique de permissions doit se faire via les outils fournis par Laravel, tels que les **Middlewares** ou les systèmes de **Gates et Policies**.

3. Cahier des charges technique (Mission de l'équipe SISR)

Alors que l'équipe SLAM se concentre sur la création de l'application, votre périmètre de responsabilité est tout aussi fondamental : construire les fondations numériques qui garantiront sa disponibilité, sa performance et sa sécurité. Vous êtes les architectes et les gardiens de l'environnement d'hébergement du projet "EventHub Lurçat".

Votre mission consiste à prendre en charge l'intégralité du cycle de vie de l'infrastructure : sa **conception** architecturale, son **déploiement** effectif, sa **sécurisation** rigoureuse et la définition des procédures de **maintenance**. L'ensemble de votre travail devra être guidé par trois principes directeurs non négociables :

- **Robustesse :** L'infrastructure doit être fiable et résiliente.
- **Cloisonnement :** La sécurité repose sur une segmentation stricte des services et des réseaux.
- **Documentation :** Chaque décision, configuration et procédure doit être documentée pour assurer la pérennité et la maintenabilité de la solution.

Une Infrastructure à deux niveaux :

Nous sommes conscients que l'architecture demandée est complète et se rapproche des standards de production. Pour vous permettre de piloter votre projet de manière réaliste et de sécuriser un livrable de haute qualité, nous vous proposons de structurer votre travail en deux périmètres :

- **Le Périmètre essentiel ou MVP (Minimum Viable Product):** C'est le socle de votre projet. Il constitue une infrastructure fonctionnelle, sécurisée et centralisée qui émule une architecture de production moderne. La maîtrise parfaite de ce périmètre est l'objectif principal et garantit la validation des compétences clés de votre formation. Il inclut :
    - Une architecture à plusieurs niveaux comprenant : 1x Firewall OPNsense, 1x Serveur Mail, et 1x Serveur de Base de Données.
    - **L'implémentation d'un Reverse Proxy** en tant que point d'entrée unique et sécurisé pour tout le trafic web (hébergeant le site Laravel et le site météo).
    - **Le déploiement du contrôleur de domaine Active Directory** pour la gestion centralisée des identités et l'authentification des utilisateurs.
    - Le cloisonnement réseau strict entre la DMZ et le LAN, incluant les règles spécifiques au Reverse Proxy et à l'Active Directory.
    - L'ensemble des exigences de durcissement système, de sauvegarde/restauration pour ces services.
    - La documentation associée (schémas, DAT/DEX).

- **Le projet complet :** Ce périmètre transforme une infrastructure robuste en une solution d'entreprise hautement résiliente et durcie. S'attaquer à ces points démontre une compréhension approfondie des enjeux de production et une plus grande maturité technique. Il s'agit d'ajouter au périmètre essentiel :
    - **La haute disponibilité du pare-feu**, via le déploiement et la configuration du second OPNsense avec le protocole CARP.
    - La mise en place des services de support avancés : un **serveur NTP interne** faisant autorité pour toute l'infrastructure.
    - Le déploiement d'un **serveur DNS sécurisé via DNS over TLS (DoT)**.
    - Notifications (mail)
    - Supervision / monitoring (suivi, tableau de bord)

Ce choix de périmètre vous appartient. Livrer un MVP parfaitement fonctionnel, sécurisé et documenté est déjà un gage de grande réussite. Le compléter avec les éléments du projet complet démontre une volonté et une capacité à dépasser les attentes qui seront particulièrement valorisées dans l'évaluation finale.

3.1 Architecture Réseau (à schématiser)

Pour ce projet, vous allez construire une infrastructure qui émule les bonnes pratiques des environnements de production modernes, en mettant l'accent sur la sécurité en profondeur, la centralisation des services et la résilience. Le premier livrable tangible de votre mission sera un schéma d'architecture réseau clair et détaillé, représentant la topologie sophistiquée et robuste attendue.

- **Périphérie (Firewall/Routeur) :**
    - Le point d'entrée de votre réseau est gardé par **deux machines virtuelles OPNsense**, configurées en cluster haute disponibilité via **CARP**. C'est votre première ligne de défense et le garant de la continuité de service.

- **Zone Démilitarisée (DMZ) :** La DMZ est bien plus qu'une simple zone d'hébergement ; elle agit comme un sas de sécurité intelligent qui filtre et oriente le trafic avant qu'il n'atteigne les applications.
    - **Le bastion avancé de votre infrastructure : 1x VM Reverse Proxy.** Ce serveur Nginx sera le **seul et unique point d'entrée** pour tout le trafic web. Son rôle est critique : il terminera les connexions HTTPS, gérera les certificats de sécurité, et agira comme un aiguilleur intelligent, distribuant les requêtes vers les serveurs applicatifs appropriés.
    - **Le cœur applicatif : 1x VM Serveur Applicatif.** Protégé derrière le reverse proxy, ce serveur Linux hébergera l'application Laravel. Pour démontrer la flexibilité de votre architecture, il servira également un **second site web beaucoup plus simple** (ex: une page météo statique), accessible via un autre nom de domaine.
    - **Le garant de la confidentialité des requêtes : 1x VM Serveur DNS.** Ce serveur fournira la résolution de noms interne, mais devra surtout être configuré pour accepter et répondre aux requêtes via le protocole sécurisé **DNS over TLS (DoT)**, chiffrant ainsi les communications DNS.
    - **Le service de communication sortante : 1x VM Serveur mail.** Ce service conteneurisé via **docker-mailserver**[1] gérera l'envoi des notifications, initié depuis le serveur applicatif.
- **Zone Locale (LAN) :** Votre réseau local intègre des services fondamentaux qui centralisent les fonctions critiques, le rendant plus cohérent et plus sécurisé.
    - **Le coffre-fort des données : 1x VM Serveur de base de données.** Ce serveur PostgreSQL est la destination finale des données de l'application.
    - **Le centre névralgique de l'identité : 1x VM Contrôleur de domaine.** Vous déploierez un serveur **Active Directory** (via Samba sur Linux). Son rôle est de centraliser et de gérer l'authentification de tous les utilisateurs de l'application, transformant une simple connexion en un processus d'authentification d'entreprise.

[1] https://github.com/docker-mailserver/docker-mailserver

- **Le métronome de l'infrastructure : 1x VM Serveur de temps.** Ce serveur **NTP** (`chrony` ou `ntpd`) sera la source de temps faisant autorité pour l'ensemble de votre parc. Une synchronisation temporelle parfaite est un prérequis non négociable pour la sécurité et le bon fonctionnement de protocoles comme Kerberos (utilisé par l'Active Directory).

3.2 Configuration des serveurs et articulation des services

Une fois l'architecture définie, la configuration de chaque machine virtuelle doit suivre des principes de rigueur et de sécurité. Votre travail consistera à appliquer des règles fondamentales sur l'ensemble du parc, puis à orchestrer finement les services clés pour qu'ils fonctionnent en harmonie.

3.2.1 Principes fondamentaux de configuration

Ces principes s'appliquent à **toutes** les machines virtuelles que vous déploierez.

- **Système d'exploitation :** La stabilité et la sécurité forment le socle de votre infrastructure. Toutes les machines virtuelles devront utiliser la distribution Debian GNU/Linux stable.
- **Provisioning et automatisation :** Au-delà de l'installation manuelle, la démarche professionnelle consiste à rendre le déploiement reproductible. Vous devrez **documenter** précisément l'installation et la configuration de tous les paquets et services (Nginx, PHP, PostgreSQL, Docker, etc.). La **création de scripts** pour automatiser une partie de ce provisionnement sera considérée comme un **bonus significatif**.

- **Principe de moindre privilège :** Ce principe de sécurité fondamental doit être appliqué à tous les niveaux. Les services web (Nginx, PHP-FPM) ne doivent **jamais s'exécuter avec l'utilisateur root**. Vous créerez des utilisateurs de service dédiés avec des permissions minimales. De même, la connexion à la base de données depuis le serveur web devra se faire avec un utilisateur PostgreSQL spécifique, dont les droits seront strictement limités à la base de données de l'application "**EventHub Lurçat**".

3.2.2 Configuration et articulation des services clés

Sur la base de ces principes, vous procéderez à la configuration spécifique des services qui animent votre architecture.

- **Pilotage du trafic par le Reverse Proxy :** Le serveur Nginx en DMZ est bien plus qu'un simple relais. Vous le configurerez pour :
    - Assurer la **terminaison SSL/TLS**, en étant le seul détenteur du certificat HTTPS.
    - **Router les requêtes en se basant sur le nom d'hôte** : `eventhub.lurcat.local` vers le serveur applicatif Laravel, et `meteo.lurcat.local` vers le second site web.
    - **Renforcer la sécurité** en injectant des en-têtes de sécurité HTTP (HSTS, etc.) dans les réponses envoyées aux clients.
- **Intégration de l'authentification centralisée :** L'Active Directory doit être pleinement opérationnel. La configuration ne s'arrête pas au serveur : vous devrez collaborer étroitement avec l'équipe SLAM pour assurer que l'application Laravel puisse déléguer son authentification à ce service via le protocole sécurisé LDAPS.
- **Synchronisation temporelle stricte :** Le serveur NTP du LAN est la référence. **Toutes les autres machines virtuelles**, sans exception, qu'elles soient dans le LAN ou la DMZ, devront être configurées pour se synchroniser exclusivement sur ce serveur interne.

3.3 Système de sauvegarde

Une infrastructure fonctionnelle est une chose, mais sa pérennité en est une autre. La mise en place d'un système de sauvegarde est donc une exigence non négociable.

- **Stratégie :** Vous implémenterez un script de sauvegarde automatisé, déclenché quotidiennement par une tâche **cron**. Ce script devra effectuer séquentiellement deux actions critiques :
    1. **Sauvegarde de la base de données :** Réalisation d'un export complet (dump) de la base de données PostgreSQL via la commande `pg_dump`.
    2. **Sauvegarde des fichiers :** Création d'une archive compressée du répertoire contenant le code source de l'application web.
- **Stockage et rétention :** Les sauvegardes ne doivent jamais être stockées sur la même machine que les données originales. Les archives générées (compressées et datées) devront être transférées vers un stockage externe, qui sera simulé par un **partage NFS sur une VM dédiée ou un volume NAS**. Une politique de rétention de **7 jours glissants** devra être mise en place.
- **Documentation et test de restauration :** Une sauvegarde n'a de valeur que si l'on est certain de pouvoir la restaurer. Vous devrez donc rédiger une **procédure de restauration claire et testée (PRD - Plan de Reprise de Données)**. Ce document expliquera, étape par étape, comment restaurer l'application et sa base de données à un état fonctionnel à partir d'une archive de sauvegarde.

3.4 Exigences de Cybersécurité (niveau infrastructure)

La sécurité de votre infrastructure ne repose pas sur un seul rempart, mais sur une succession de couches de défense intelligentes et coordonnées. Chaque service, chaque flux, chaque configuration doit être pensé sous l'angle de la sécurité. Le respect des principes décrits ci-dessous est donc la pierre angulaire de votre mission ; il est **non négociable** et fera l'objet d'une évaluation particulièrement rigoureuse.

3.4.1 Cloisonnement réseau strict (Firewalling)

Votre pare-feu OPNsense n'est pas une simple passerelle, c'est le gardien intraitable qui applique une politique de confiance zéro. Votre première mission est de sculpter des règles de pare-feu qui transforment vos zones réseau en cloisons étanches, où seuls les flux absolument essentiels et légitimes sont autorisés.

- **Protéger la façade (Internet → DMZ) :** Votre surface d'attaque exposée au monde doit être minimale. Le trafic entrant depuis Internet sera drastiquement limité : seuls les flux HTTPS (TCP/443) vers le **Reverse Proxy** et DNS over TLS (TCP/853) vers le **Serveur DNS** sont autorisés. Tout le reste est silencieusement rejeté.
- **Verrouiller le cœur du réacteur (DMZ → LAN) :** La règle la plus critique de votre architecture est la protection du LAN. La DMZ est, par définition, une zone de méfiance. Les communications initiées depuis cette zone vers votre réseau de confiance doivent être une exception rare et justifiée. Les seuls flux autorisés sont :
    - Du **serveur applicatif** vers le **serveur de base de données** (exclusivement sur le port TCP/5432).
    - Du **serveur applicatif** vers le **contrôleur de domaine** pour l'authentification (via LDAPS et Kerberos).
    - De **toutes les machines de la DMZ** vers le **serveur NTP** interne pour la synchronisation temporelle (UDP/123).

    - **Tout autre flux initié depuis la DMZ vers le LAN est formellement et impitoyablement interdit.**

- **Contrôler les mouvements internes (Intra-DMZ) :** La sécurité ne s'arrête pas aux frontières des zones. Les communications au sein même de la DMZ sont également contrôlées pour limiter la capacité d'un attaquant à se déplacer latéralement. Ainsi, le **Reverse Proxy** ne parle qu'au **serveur applicatif** (HTTP), et ce dernier ne parle qu'au **serveur mail** (SMTP).

3.4.2 Durcissement système (Hardening)

Le pare-feu protège les routes, mais chaque serveur doit être lui-même une forteresse. Le durcissement système (*hardening*) consiste à réduire la surface d'attaque de chaque machine virtuelle, en partant du principe qu'un attaquant pourrait un jour franchir le périmètre. Cette hygiène numérique doit être appliquée sur **chaque VM** :

- Maintien en condition opérationnelle par des **mises à jour de sécurité régulières et automatisées**.
- Protection de la porte d'entrée administrative (SSH) contre les attaques par force brute via la configuration de **Fail2Ban**.
- Verrouillage de l'accès SSH en **interdisant la connexion de l'utilisateur root** et en **privilégiant l'authentification par clé publique**.
- Réduction de la complexité et des risques en **désactivant tous les services et ports non essentiels**.

3.4.3 Sécurisation des flux

Les données sont les plus vulnérables lorsqu'elles voyagent. Votre travail consiste à garantir leur confidentialité et leur intégrité à chaque étape de leur parcours.

- **Le chiffrement externe :** Le **Reverse Proxy** est le seul point de contact avec le monde extérieur. C'est donc lui qui portera la responsabilité du chiffrement **HTTPS**, assurant que toute communication entre un utilisateur et votre infrastructure est protégée. La communication entre le reverse proxy et le serveur applicatif, se déroulant dans un segment réseau que vous maîtrisez, pourra rester en HTTP.
- **Le chiffrement interne critique :** La sécurité ne concerne pas que l'extérieur. Les identifiants des utilisateurs sont des données hautement sensibles. Par conséquent, la communication entre l'application et l'Active Directory **doit impérativement utiliser LDAPS** (LDAP sur TLS) pour garantir que les mots de passe ne transitent jamais en clair sur votre réseau.

3.4.4 Supervision

Une infrastructure non supervisée est une infrastructure aveugle. La supervision est l'outil qui vous permet de passer d'une posture réactive (réparer quand c'est cassé) à une posture proactive (anticiper les problèmes).

- Vous mettrez en place un **outil de supervision simple** (ex: Uptime Kuma, Zabbix) qui agira comme le tableau de bord de votre infrastructure.
- Ce système devra surveiller en permanence les "**signes vitaux**" de votre plateforme : la connectivité de base des VMs (ping), la disponibilité des services web, la réponse du service de base de données, et l'état des autres services critiques comme l'annuaire Active Directory.

4. Exigences transversales (SLAM & SISR)

La réussite de ce projet ne se mesurera pas uniquement à la qualité technique de vos livrables respectifs, mais aussi, et surtout, à votre capacité à fonctionner comme une seule et même équipe projet. Les compétences techniques sont un prérequis, mais c'est votre collaboration qui fera la différence. La réussite repose sur votre capacité à travailler ensemble.

4.1 Gestion de projet et méthodologie agile

Pour vous fournir un cadre de travail professionnel et efficace, l'adoption d'outils et de rituels agiles est imposée.

- **Dépôt de code source : la colonne vertébrale du projet**
  L'utilisation d'un **dépôt Git est obligatoire** et servira de source de vérité unique pour tout le projet. Il ne s'agit pas uniquement d'un outil pour les développeurs. Les deux équipes devront l'utiliser activement :
  - **L'équipe SLAM** y versionnera l'intégralité du code source de l'application Laravel.
  - **L'équipe SISR** y versionnera tous ses scripts (sauvegarde, supervision) et ses fichiers de configuration essentiels.
    En guise de premier exercice de collaboration, l'équipe SISR peut proposer d'héberger une instance GitLab pour le projet ; à défaut, un service externe sera utilisé.
- **Tableau Kanban : votre feuille de route visuelle**
  Pour piloter le projet avec agilité et transparence, la gestion des tâches (qu'il s'agisse de *user stories* fonctionnelles ou de tâches techniques d'infrastructure) doit se faire via un **tableau Kanban partagé** (ex: Trello, Kanboard). Ce tableau assurera que chacun ait une vision claire de l'avancement. Il devra contenir, au minimum, les colonnes suivantes pour suivre le cycle de vie d'une tâche : Backlog, À faire, En cours, À tester/Valider, Terminé.

4.2 Collaboration et communication

La technologie est un outil, mais elle ne remplace pas le dialogue. Une communication fluide et structurée est la clé pour anticiper les problèmes et gérer les dépendances inhérentes à un projet aussi interdépendant que celui-ci.

- **Rituel de synchronisation hebdomadaire**
  Une réunion de synchronisation de 30 minutes est **obligatoire** chaque semaine. Ce n'est pas une simple formalité, mais un point de rencontre essentiel pour assurer l'alignement des deux équipes. Pour garantir la traçabilité des décisions, un **compte-rendu succinct** (conseillé au format Markdown, stocké sur le dépôt Git) devra être rédigé à tour de rôle par un membre de chaque équipe. Ce document listera les points discutés, les décisions prises et les éventuels blocages.

- **Gestion des dépendances inter-équipes : un contrat de confiance**
  Vos deux missions sont profondément interdépendantes. La communication n'est donc pas une option, elle est vitale. Considérez les flux d'information suivants comme un contrat de service entre vos deux équipes :

    - **Flux SLAM → SISR (Les besoins de l'application) :** L'équipe SISR ne peut pas construire une infrastructure pertinente sans connaître les besoins précis de l'application. L'équipe SLAM a donc la responsabilité de fournir **rapidement et clairement** les prérequis techniques. Cela inclut la version exacte de PHP et la liste complète des extensions PHP requises, par exemple : `php-pgsql` pour la base de données, `php-sodium` pour le hachage des mots de passe, et surtout **`php-ldap`** pour permettre à l'application de communiquer avec le service d'annuaire Active Directory.

- **Flux SISR → SLAM (Les clés de l'infrastructure) :** Inversement, l'application ne peut pas fonctionner sans les informations de connexion à l'infrastructure. Dès que les services sont prêts, l'équipe SISR doit fournir à l'équipe SLAM un "**dossier de connexion**" formel et complet. Ce document est un livrable critique qui doit contenir toutes les informations nécessaires pour configurer le fichier .env de Laravel :
  - L'**URL d'accès** à l'application (passant par le reverse proxy, ex: `https://eventhub.lurcat.local`).
  - Les informations de connexion à la **base de données** (hôte, port, nom de la BDD, utilisateur, mot de passe).
  - Les paramètres du **serveur SMTP** pour l'envoi des courriels.
  - Les paramètres de connexion à l'**annuaire Active Directory** (adresse du serveur, Base DN, et les identifiants d'un compte de service dédié pour lier l'application à l'annuaire).

5. Livrables Finaux et soutenance

L'aboutissement de ce semestre de travail sera évalué lors d'une soutenance finale. La note attribuée reflétera la qualité globale de votre projet, en se basant sur l'exhaustivité et le professionnalisme des livrables tangibles que vous produirez, ainsi que sur la clarté et la maîtrise dont vous ferez preuve lors de votre présentation orale.

5.1 Livrables de l'équipe SLAM

L'équipe SLAM sera évaluée sur la qualité, la robustesse et la documentation de la solution applicative conçue et développée. Les attendus sont les suivants, présentés dans un ordre logique de réalisation :

1. **Dossier de conception applicative (UML/Merise) :** Tout comme l'équipe SISR doit modéliser son infrastructure via des schémas réseau, l'équipe SLAM doit formaliser la conception de son application avant de la développer. Ce dossier est un livrable critique et doit contenir au minimum :
    - **Le modèle de données :** Un **Diagramme de Classes (UML)** ou un **Modèle Conceptuel de Données (MCD - Merise)** détaillé. Ce diagramme doit représenter toutes les entités (Utilisateurs, Événements, Salles, etc.), leurs attributs, et surtout les relations qui les unissent (one-to-many, many-to-many avec cardinalités).
    - **Les fonctionnalités et les acteurs :** Un **Diagramme de Cas d'Utilisation (UML)**. Il doit identifier clairement les acteurs (Admin, Enseignant, Étudiant) et lister l'ensemble des cas d'utilisation correspondant aux fonctionnalités majeures du cahier des charges (ex: "S'authentifier", "Créer un événement", "Valider une inscription").
2. **Code source complet :** La totalité du code source de l'application, proprement commenté et structuré, doit être disponible sur la branche principale du dépôt Git commun.

3. **Documentation utilisateur :** Un guide d'utilisation clair et concis, destiné à un personnel administratif non technique, expliquant les fonctionnalités principales de la plateforme.
4. **Documentation technique :** Un fichier `README.md` exhaustif à la racine du projet. Ce document est crucial et doit détailler la procédure complète pour installer l'environnement de développement en local (prérequis, commandes d'installation, configuration du fichier .env).
5. **Une démonstration fonctionnelle complète** de la plateforme lors de la soutenance.

5.2 Livrables de l'équipe SISR

Pour l'équipe SISR, l'évaluation portera sur la formalisation, la robustesse et la maintenabilité de l'infrastructure mise en place. Les documents et artefacts suivants sont requis :

1. **Les schémas d'architecture réseau :** La représentation visuelle de votre conception, incluant un schéma logique (flux, adressage IP, VLANs) et un schéma physique/virtuel (disposition des VMs sur l'hyperviseur).
2. **Le dossier d'architecture technique (DAT) :** La carte d'identité technique de l'infrastructure. Ce document doit décrire l'ensemble des composants, justifier les choix techniques opérés, et détailler les configurations IP et les règles de pare-feu majeures.
3. **Le dossier d'exploitation (DEX) :** Le manuel de survie pour l'exploitation au quotidien de la plateforme. Il doit contenir les procédures de sauvegarde/restauration (PRD), les commandes de maintenance courante, et une synthèse de la politique de sécurisation mise en place.
4. **Les artefacts techniques versionnés :** L'ensemble des scripts d'automatisation (sauvegarde, supervision) et les fichiers de configuration majeurs (OPNsense, Nginx) doivent être versionnés sur le dépôt Git commun.

5.3 La soutenance orale

En raison des contraintes d'emploi du temps, la soutenance finale se déroulera en deux volets distincts. Il ne s'agit pas de deux présentations séparées, mais bien des **deux parties d'une même soutenance**, chacune se focalisant sur un aspect du projet commun. La cohérence entre les deux présentations sera un critère d'évaluation majeur.

Volet 1 : Soutenance applicative (Équipe SLAM - mardi après-midi)

L'équipe SLAM présentera en premier. Votre session se concentrera sur la démonstration de la solution logicielle et la justification de vos choix de développement.

Déroulé et attendus :

1. **Introduction commune :** Vous commencerez par une brève présentation du projet "**EventHub Lurçat**" dans sa globalité, en mentionnant explicitement que l'application que vous allez présenter est hébergée sur une infrastructure conçue et gérée par l'équipe SISR.
2. **Démonstration fonctionnelle complète :** Vous déroulerez les différents scénarios d'utilisation de la plateforme en naviguant avec les trois rôles (Admin, Enseignant, Étudiant). Chaque fonctionnalité clé du cahier des charges doit être démontrée en direct.
3. **Focus sur la sécurité applicative :** Vous devrez ouvrir le code sur des exemples précis pour prouver la mise en place des mesures de sécurité : montrer un `Form Request` pour la validation des entrées, un formulaire avec le token `@csrf`, un Middleware de protection de route, etc.
4. **Articulation avec l'infrastructure :** Vous devrez impérativement expliquer comment votre application interagit avec l'infrastructure des SISR. Montrez concrètement le fichier de configuration .env et expliquez comment vous avez utilisé les informations du "dossier de connexion" (base de données, serveur mail) pour faire fonctionner l'application. C'est une preuve tangible de votre collaboration.

Volet 2 : Soutenance Infrastructure (Équipe SISR - vendredi après-midi)

L'équipe SISR conclura la soutenance. Votre session se focalisera sur la présentation des fondations techniques qui rendent l'application fonctionnelle, sécurisée et résiliente.

Déroulé et attendus :

1. **Introduction et lien avec le volet 1 :** Vous commencerez en faisant directement référence à la démonstration de l'équipe SLAM. Votre point de départ sera : "*Vous avez vu l'application fonctionner mardi. Nous allons maintenant vous montrer l'architecture invisible qui le permet.*" Vous montrerez rapidement que l'application est bien en ligne sur votre infrastructure.
2. **Présentation de l'architecture :** À l'aide de vos schémas (logique et physique/virtuel), vous présenterez et justifierez l'architecture réseau mise en place (DMZ/LAN, haute disponibilité avec OPNsense).
3. **Démonstration technique des exigences clés :** Vous ne ferez pas que décrire, vous montrerez :
    - Les **règles de pare-feu** critiques sur l'interface OPNsense.
    - Le fonctionnement d'un **script de sauvegarde** et la procédure de restauration documentée.
    - La configuration du **durcissement système** (ex: la configuration SSH, Fail2Ban en action).
    - Le **tableau de bord de supervision** montrant l'état des services.
4. **Articulation avec l'application :** Vous devrez démontrer comment vous avez répondu aux besoins de l'équipe SLAM. Par exemple : "*Les développeurs avaient besoin de l'extension* `php-sodium` *pour la sécurité des mots de passe ; voici comment nous l'avons installée et configurée sur le serveur web.*" Vous expliquerez également la mise en place du cron pour le planificateur de tâches Laravel.

5.3.1 Critères d'évaluation transversaux

Bien que les présentations soient séparées, votre note finale dépendra de la **cohérence globale** et de la capacité de chaque équipe à **valoriser le travail de l'autre**. Une équipe SLAM qui ne mentionne pas l'infrastructure ou une équipe SISR qui ne fait pas le lien avec l'application qu'elle héberge sera pénalisée. Votre projet est un tout, votre soutenance doit en être le reflet.

6. Grille d'évaluation indicative – Projet EventHub Lurçat

Ce document n'est pas un barème à points, mais une grille de maturité professionnelle. L'objectif est d'évaluer votre capacité à mener un projet informatique de A à Z en respectant un cahier des charges exigeant. La note finale sera une synthèse de l'évaluation de l'ensemble des piliers ci-dessous. Un pilier jugé "Insuffisant" aura un impact significatif sur la note globale, car il met en péril l'ensemble du projet.

Pour chaque critère, l'évaluation se fera selon trois niveaux :

- ⚪ **Insuffisant :** La fonctionnalité est absente, non fonctionnelle, ou présente des défauts majeurs.
- 🟢 **Conforme aux Attentes :** Le critère est rempli de manière fonctionnelle et respecte scrupuleusement le cahier des charges. C'est le niveau attendu pour valider la compétence.
- 🌟 **Démarche Professionnelle (Bonus) :** L'étudiant n'a pas seulement répondu au besoin, il a fait preuve d'initiative, d'élégance technique, d'anticipation ou d'une rigueur qui dépasse les attentes (ex: code particulièrement propre, automatisation poussée, documentation exemplaire, etc.).

6.1 Mise en garde sur la sécurité : un prérequis non négociable

Les critères de sécurité (applicatifs et infrastructure) sont considérés comme des **piliers fondamentaux**. Une défaillance majeure et non justifiée sur l'un de ces critères (ex: une injection SQL possible, un pare-feu non configuré) entraînera une **pénalité très importante** sur la note du pilier technique concerné, quelle que soit la qualité des autres fonctionnalités. **On ne livre pas un produit non sécurisé.**

6.2 Le respect des délais et le reporting agile

Dans un projet professionnel, le client ou le chef de projet ne disparaît pas pendant des mois pour revenir le jour de la livraison. Il attend une visibilité continue sur l'avancement pour anticiper les risques et s'assurer que le projet est sur la bonne voie. C'est cette posture de **transparence et de communication proactive** que nous attendons de vous.

Concrètement, nous n'exigeons pas de rapports d'avancement formels et chronophages. En revanche, nous nous attendons à ce que vous soyez en mesure de **démontrer votre avancement à tout moment**. Vos outils de gestion de projet sont vos meilleurs alliés pour cela :

- **Votre dépôt Git** est le journal de bord de votre travail technique. Un historique de commits réguliers et bien commentés est la preuve d'une activité continue.
- **Votre tableau Kanban** doit être le reflet fidèle de l'état du projet. Il doit être mis à jour en temps réel.

Sur demande du client (votre enseignant), durant une séance d'atelier, vous devrez être capables de présenter rapidement et concrètement où vous en êtes. Cela peut prendre la forme de :

- Une revue rapide de votre tableau Kanban.
- Une démonstration de 5 minutes d'une fonctionnalité en cours de développement sur une branche Git.
- Une présentation d'un schéma d'architecture ou d'un diagramme UML sur lequel vous travaillez.
- L'explication d'un script ou d'un fichier de configuration que vous venez de finaliser.

Cette pratique n'a pas pour but de vous mettre sous pression, mais au contraire de **sécuriser votre projet**. C'est le meilleur moyen pour nous de détecter un blocage, de corriger une mauvaise interprétation du cahier des charges au plus tôt, et de vous prodiguer un conseil technique au moment où vous en avez le plus besoin. Un projet qui avance de manière transparente est un projet qui inspire confiance.

Pilier 1 : Réalisation Technique - application (SLAM)

Critère	Niveaux d'évaluation
Gestion des utilisateurs & rôles	○ Insuffisant / ● Conforme / ☀ Démarche Pro
Gestion des eessources (CRUD salles/matériels)	○ Insuffisant / ● Conforme / ☀ Démarche Pro
Gestion des événements & calendrier	○ Insuffisant / ● Conforme / ☀ Démarche Pro
Logique de contrôle de conflit (Fiabilité)	○ Insuffisant / ● Conforme / ☀ Démarche Pro
Système de réservation & notifications	○ Insuffisant / ● Conforme / ☀ Démarche Pro
Panel d'administration (Dashboard, Modération)	○ Insuffisant / ● Conforme / ☀ Démarche Pro
Sécurité applicative (Anti SQLi, XSS, CSRF, RBAC)	○ Critique / ● Requis / ☀ Démarche Pro

Pilier 2 : Réalisation Technique - infrastructure (SISR)

Critère	Niveaux d'évaluation
Architecture réseau (HA, DMZ/LAN)	○ Insuffisant / ● Conforme / ☀ Démarche Pro
Configuration des serveurs & services (Web, DB, Mail)	○ Insuffisant / ● Conforme / ☀ Démarche Pro
Système de sauvegarde (Script fonctionnel)	○ Insuffisant / ● Conforme / ☀ Démarche Pro
Procédure de restauration (documentée ET testée)	○ Insuffisant / ● Conforme / ☀ Démarche Pro
Sécurité: cloisonnement Réseau (Règles Firewall)	○ Critique / ● Requis / ☀ Démarche Pro
Sécurité: durcissement système & HTTPS	○ Critique / ● Requis / ☀ Démarche Pro
Supervision (outil en place et fonctionnel)	○ Insuffisant / ● Conforme / ☀ Démarche Pro

Pilier 3 : Qualité Professionnelle & Documentation

Critère	Niveaux d'évaluation
Qualité du code & des scripts (lisibilité, pertinence)	○ Insuffisant / ● Conforme / ☀ Démarche Pro
Documentation SLAM (user & tech `README.md`)	○ Insuffisant / ● Conforme / ☀ Démarche Pro
Documentation SISR (schémas, DAT, DEX)	○ Insuffisant / ● Conforme / ☀ Démarche Pro
Qualité générale des livrables (clarté, sans fautes)	○ Insuffisant / ● Conforme / ☀ Démarche Pro

Pilier 4 : Collaboration & Gestion de Projet

Critère	Niveaux d'évaluation
Utilisation du dépôt Git (fréquence, pertinence)	○ Insuffisant / ● Conforme / ☀ Démarche Pro
Utilisation du tableau Kanban (mise à jour, pertinence)	○ Insuffisant / ● Conforme / ☀ Démarche Pro
Qualité de la communication inter-équipes	○ Insuffisant / ● Conforme / ☀ Démarche Pro

Pilier 5 : Soutenance Finale

Critère	Niveaux d'évaluation
Clarté et maîtrise de la présentation	○ Insuffisant / ● Conforme / ☀ Démarche Pro
Qualité de la démonstration (fluidité, pertinence)	○ Insuffisant / ● Conforme / ☀ Démarche Pro
Mise en valeur de la collaboration	○ Insuffisant / ● Conforme / ☀ Démarche Pro
Cohérence entre les deux volets de la soutenance	○ Insuffisant / ● Conforme / ☀ Démarche Pro
Capacité à répondre aux questions	○ Insuffisant / ● Conforme / ☀ Démarche Pro

7. Derniers conseils

Ce cahier des charges définit le "quoi" de votre mission. Ce dernier chapitre vise à vous guider sur le "comment".

Votre réussite dans ce projet, et dans votre future carrière, dépendra autant de votre savoir-être que de votre savoir-faire. Considérez les points suivants comme des clés pour transformer ce défi académique en une véritable expérience professionnelle.

7.1 Commencez petit, commencez tôt : le pouvoir des itérations

La pire erreur que vous puissiez faire est de lire ce document, de vous sentir submergé, et de procrastiner. Un projet complexe ne se construit pas en un bloc, mais par une succession de petites victoires.

Votre premier objectif commun (Semaine 1-2) devrait être de construire un "fil de fumée" (smoke test) :

- **Mission SISR :** Monter une VM OPNsense basique, une VM web avec Nginx, et une VM BDD avec PostgreSQL. Assurez-vous simplement que le serveur web peut "pinger" le serveur de base de données.
- **Mission SLAM :** Créer un projet Laravel vierge. Votre seul objectif est de le déployer sur le serveur web des SISR et de réussir à le connecter à la base de données PostgreSQL. Affichez une simple page "Hello World" qui confirme la connexion à la BDD.

Si vous réussissez cela rapidement, vous avez prouvé que 90% de vos problèmes de communication et d'architecture de base sont résolus. Vous pourrez ensuite construire les fonctionnalités brique par brique sur cette fondation saine. **Ne visez pas la perfection, visez le progrès constant.**

7.2 Votre dépôt Git est le journal de bord de votre projet

Un `git push` n'est pas une simple sauvegarde. C'est une communication à votre équipe et à votre "futur vous". Adoptez des pratiques professionnelles dès le début :

- **Des commits atomiques et explicites :** Fuyez les messages de commit vagues comme "Modifications" ou "Mise à jour". Un bon message de commit répond à la question "Qu'est-ce que ce commit applique ?". Utilisez le mode impératif :
    - *Mauvais :* `update code`
    - *Bon :* `Feat: Add user registration form` (Feat = Feature)
    - *Bon :* `Fix: Correct SQL query for event conflict` (Fix = Correction de bug)
    - *Bon :* `Docs: Update firewall rules in DEX` (Docs = Documentation)
- **Utilisez des branches de fonctionnalités :** Ne travaillez jamais directement sur la branche principale (`main` ou `master`). Pour chaque nouvelle fonctionnalité ou correction majeure, créez une branche (`feature/user-login`, `fix/backup-script`). Cela garde votre branche principale propre et stable.

Un historique Git bien tenu est la preuve d'un travail méthodique et professionnel.

7.3 La documentation n'est pas une punition, c'est un service

Ne considérez pas la documentation comme une corvée à faire à la dernière minute. Prenez des notes au fur et à mesure que vous travaillez.

- **Pour les SISR :** Vous venez de passer trois heures à comprendre une configuration obscure d'OPNsense pour faire fonctionner le CARP ? Prenez 5 minutes pour écrire la solution dans un fichier `notes.md` dans votre dossier de documentation. Ce sera la base de votre DEX et cela vous évitera de refaire la même recherche dans un mois.
- **Pour les SLAM :** Vous avez créé une classe de service complexe pour gérer les notifications ? Prenez 2 minutes pour ajouter un commentaire en début de fichier (`PHPDoc`) expliquant son rôle et comment l'utiliser.

La documentation, c'est avant tout un service que vous vous rendez à vous-même et à vos coéquipiers.

7.4 Apprenez à être "bloqué" efficacement

Vous allez rencontrer des problèmes que vous ne saurez pas résoudre immédiatement. C'est normal. Le but n'est pas de tout savoir, mais de savoir comment trouver la solution.

Adoptez la "**Règle des 20 minutes**" :

1. Pendant 20 minutes, cherchez la solution par vous-même (recherche, tests, lecture de documentation).
2. Si au bout de 20 minutes vous êtes toujours au même point, **arrêtez**.
3. **Formulez une question claire et précise.** N'allez pas voir vos collègues ou votre enseignant en disant "Ça ne marche pas". Préparez plutôt :
    - **Ce que j'essaie de faire :** "Je tente de configurer une règle de pare-feu pour autoriser le flux PostgreSQL."
    - **Ce que j'ai déjà essayé :** "J'ai créé cette règle (montrer la règle), j'ai vérifié les logs, j'ai testé avec un telnet depuis le serveur web."
    - **Le résultat obtenu :** "Le flux est toujours bloqué et je ne comprends pas pourquoi."

C'est ainsi que l'on demande de l'aide dans le monde professionnel. Cela montre que vous avez fait votre part du travail et permet à la personne qui vous aide d'être beaucoup plus efficace.

7.5 Utiliser l'IA comme un levier, pas une béquille

Soyons clairs : vous allez utiliser des outils d'intelligence artificielle (ChatGPT, GitHub Copilot, etc.), et c'est normal. Dans le monde professionnel, ne pas savoir utiliser ces outils est déjà un handicap. Cependant, la différence entre un amateur et un professionnel réside dans la manière de les utiliser.

L'IA est un assistant exceptionnel, pas un sous-traitant.

✅ **Ce qui est une utilisation intelligente et encouragée :**

- **Pour débloquer :** "*Mon script de sauvegarde échoue avec cette erreur, voici le code. Quelles sont les pistes à explorer ?*"
- **Pour apprendre :** "*Explique-moi le principe du CARP dans OPNsense comme si j'avais 15 ans.*" ou "*Quelle est la différence entre un Gate et une Policy dans Laravel ?*"
- **Pour générer du code "boilerplate" :** "*Génère-moi un contrôleur Laravel CRUD standard pour un modèle 'Salle'. Je l'adapterai ensuite à mes besoins spécifiques.*"
- **Pour optimiser ou reformuler :** "*Voici ma fonction PHP, peux-tu me proposer une manière plus performante ou plus lisible de l'écrire ?*" ou "*Aide-moi à rédiger une section de documentation plus claire.*"

❌ **Ce qui constitue une mauvaise pratique et sera sanctionné :**

- **Déléguer la réflexion :** Copier-coller une section entière du cahier des charges en demandant "Écris le code pour ça". Vous obtiendrez un code que **vous ne comprenez pas**, que vous ne saurez ni maintenir, ni déboguer, ni justifier.
- **Copier-coller sans comprendre :** Intégrer un bloc de code ou une configuration (ex: une règle de pare-feu) sans en maîtriser chaque ligne est une faute professionnelle grave. C'est ainsi que naissent les failles de sécurité.

- **Faire effectuer le travail à votre place :** Utiliser l'IA pour réaliser une tâche dont l'objectif pédagogique est précisément que *vous* la réalisiez pour acquérir la compétence.

La règle d'or est simple : vous êtes 100% responsable du code, des configurations et des documents que vous livrez. Si un code généré par une IA est défaillant ou présente une faille de sécurité, l'erreur vous incombe entièrement. Servez-vous de l'IA pour accélérer votre travail et approfondir votre compréhension, jamais pour remplacer votre intelligence.

7.6 L'entraide : la compétence invisible la mieux valorisée

Ce projet est structuré pour une collaboration SLAM/SISR. Cependant, la collaboration ne s'arrête pas là. Nous vous encourageons vivement à pratiquer **l'entraide au sein de vos spécialités et même entre les groupes de projet.**

Un étudiant SISR bloqué sur une configuration Nginx peut tout à fait demander un conseil à un autre groupe SISR. Une équipe SLAM qui peine sur une requête Eloquent complexe a tout intérêt à en discuter avec ses camarades.

L'objectif n'est pas la compétition, mais la réussite collective. Savoir demander de l'aide et savoir en apporter est l'une des compétences les plus recherchées en entreprise. Les "silos" de connaissance sont inefficaces ; les équipes qui partagent et communiquent sont performantes.

Comme pour l'IA, il y a une limite claire à ne pas franchir :

- **Aider, c'est expliquer, guider, regarder un problème à deux.**
- **Faire à la place de l'autre, c'est lui rendre un très mauvais service** et c'est une forme de fraude.

Un bon professionnel n'est pas seulement celui qui sait faire, c'est aussi celui qui rend les autres meilleurs. Adoptez cette posture dès maintenant.

8. Ressources

8.1 Ressources pour les SLAM

8.1.1 Laravel

- Documentation officielle : https://laravel.com/docs/12.x
- A lire absolument, avec une liste de ressources intéressantes (en anglais bien sûr) : https://laravel.sillo.org/posts/cours-laravel-10-les-bases-presentation-generale
- Un tutoriel Laravel (installation et création des premiers éléments) en Français: https://cours.brosseau.ovh/tp/laravel/introduction.html

8.1.2 Composer

- Un tutoriel sur Composer à lire avant de l'utiliser avec Laravel : https://www.hostinger.com/fr/tutoriels/comment-installer-et-utiliser-composer

8.1.3 PostgreSQL

- Documentation officielle: https://docs.postgresql.fr/17/tutorial.html

8.1.4 pgadmin

- Pour gérer des bases de données PostgreSQL en mode graphique (équivalent de PHPMyAdmin) : https://www.pgadmin.org/

8.1.5 Argon2id

- Lire la doc officielle de PHP (Argon2id supporté depuis la version 7.3. de PHP): https://www.php.net/manual/fr/function.password-hash.php