



Rapport

NIS CUBE

Abdelwaheb SEBA
Mohammed Ryad DERMOUCHE
Chatodit LOKONGA KONGOLO
Sedik SI MEHAND

Groupe L3L1
2024 - 2025

Informations sur le document

Les informations d'identification du document

Référence du document : NIS-Rapport-V1.00

Version du document : 1.00

Date du document : 05/05/2025

Auteur(s) : Groupe L3L1

Les éléments de vérification du document

Validé par : Nicolas DENIS

Validé le : 05/05/2025

Soumis le : 20/05/2025

Type de diffusion : Document électronique (.pdf)

Confidentialité : Réservé aux membres du groupe L3L1

L'encadrant monsieur Nicolas DENIS

Le responsable de l'UE monsieur David JANISZEK

Les membres du jury lors de la soutenance du projet

Table des matières

| | | |
|----------|---|-----------|
| 1 | Introduction | 4 |
| 1.1 | Contexte | 4 |
| 1.2 | Historique | 4 |
| 1.3 | Objectifs du document | 4 |
| 2 | Maîtrise d'œuvre et d'ouvrage | 5 |
| 2.1 | Maîtrise d'œuvre | 5 |
| 2.2 | Maîtrise d'ouvrage | 5 |
| 3 | Spécifications fonctionnelles et techniques | 6 |
| 3.1 | Exigences fonctionnelles principales | 6 |
| 3.2 | Exigences non fonctionnelles | 6 |
| 4 | Produit | 8 |
| 4.1 | Inscription & Connexion | 8 |
| 4.2 | Utilisateur | 10 |
| 5 | Technologies utilisées | 13 |
| 5.1 | Langages | 13 |
| 5.2 | Outils de développement | 13 |
| 5.3 | Framework | 14 |
| 5.4 | Base de donnée | 15 |
| 6 | Architecture du projet | 15 |
| 6.1 | schéma relationnel de la base de données | 15 |
| 6.2 | diagramme de classes | 17 |
| 6.3 | architecture du backend | 17 |
| 6.4 | Architecture du frontend | 18 |
| 7 | Challenges affrontés | 19 |
| 7.1 | Gestion du temps : | 19 |
| 7.2 | Maîtrise de nouvelles technologies : | 19 |
| 7.3 | Appropriation des notions de cybersécurité : | 19 |
| 7.4 | Analyse de la directive NIS2 : | 19 |
| 7.5 | Développement des compétences transversales : | 20 |
| 8 | Planification | 20 |
| 8.1 | Répartition des tâches | 20 |
| 8.1.1 | Documentation | 20 |
| 8.1.2 | Développement | 20 |
| 8.2 | Diagramme de Gantt | 21 |
| 8.3 | Planning | 21 |

| | |
|---|-----------|
| 9 Organisation par l'équipe | 22 |
| 10 Compétences acquises | 23 |
| 10.1 Compétences techniques | 23 |
| 10.2 Compétences méthodologiques | 24 |
| 10.3 Compétences transversales et humaines | 24 |
| 11 Perspectives d'amélioration | 25 |
| 12 Usage de l'intelligence artificielle dans le projet | 26 |
| 13 Vérification de la documentation | 27 |
| 14 Synthèse du projet | 28 |
| 15 Conclusion | 29 |

1 Introduction

1.1 Contexte

Avec l'évolution constante des cybermenaces et l'entrée en vigueur de la directive **NIS2** (Network and Information Security), les entreprises et administrations doivent renforcer leur niveau de sécurité et assurer leur conformité aux nouvelles exigences européennes. Afin de répondre à ce besoin, le projet **NIS CUBE** propose une solution innovante sous la forme d'une **interface web** permettant aux entreprises d'estimer leur niveau de mise en conformité et, par conséquent, d'améliorer leur sécurité.

Le principe du projet repose sur un processus d'**évaluation interactive** : les utilisateurs répondent à un ensemble de questions visant à mesurer la conformité des entités qu'ils représentent par rapport aux exigences de la directive NIS2. En fonction des réponses fournies, des **scores** sont attribués, et les résultats sont affichés sous forme de **représentations graphiques** afin d'identifier clairement les aspects à améliorer.

1.2 Historique

Historiquement, la première directive **NIS** (2016) avait pour objectif d'établir un niveau élevé commun de sécurité des réseaux et des systèmes d'information à travers l'Union européenne. Face à l'augmentation des menaces informatiques et à la nécessité d'impliquer un plus grand nombre d'acteurs, la directive **NIS2** est venue renforcer et étendre les obligations de sécurité dans plusieurs secteurs (énergie, transports, santé, infrastructures numériques, etc.).

Ces évolutions ont généré de nouveaux besoins : les organisations doivent désormais rendre compte de leur conformité plus précisément et faire preuve de davantage de réactivité en matière de cybersécurité. C'est dans ce contexte que s'inscrit **NIS CUBE**, afin de fournir aux acteurs concernés un outil simple et adapté pour piloter leur mise en conformité.

1.3 Objectifs du document

Ce document a pour objectif de présenter de manière structurée le projet **NIS CUBE**, développé dans le cadre de notre formation. Il s'adresse à toute personne souhaitant comprendre :

- les enjeux auxquels répond le projet ;
- les choix techniques et méthodologiques réalisés ;
- les différentes phases de développement de l'application ;
- l'architecture mise en place et les technologies utilisées ;
- les résultats obtenus et les perspectives d'évolution.

Il permet également de conserver une trace complète du travail effectué, tant sur le plan technique que documentaire, et constitue un support de référence pour toute personne amenée à utiliser, maintenir ou faire évoluer la solution.

2 Maîtrise d'œuvre et d'ouvrage

2.1 Maîtrise d'œuvre

La maîtrise d'œuvre regroupe ici l'ensemble des responsabilités techniques et organisationnelles liées à la réalisation du projet. De la conception à la livraison, chaque étape a été portée par les membres du groupe avec sérieux et engagement.

L'équipe de développement est composée de quatre étudiants en troisième année de licence Informatique et Applications à l'Université Paris Cité :

- Abdelwaheb SEBA
- Mohammed Ryad DERMOUCHE
- Chatodit LOKONGA KONGOLO
- Sedik SI MEHAND

Chacun a pris part activement à la conception de la plateforme **NIS CUBE**, en mettant en œuvre ses compétences pour analyser les besoins, développer les fonctionnalités et s'assurer du bon fonctionnement de l'application. La coordination, la répartition équitable des tâches et les échanges réguliers ont permis de faire avancer le projet de manière fluide.

Tout au long du semestre, nous avons bénéficié du regard bienveillant et des conseils éclairés de notre encadrant, **M. Nicolas Denis**. Sa disponibilité et ses retours constructifs ont grandement facilité notre progression, même dans les moments de blocage.

Cette collaboration étroite entre étudiants et encadrant a été l'une des forces de notre projet, rendant cette expérience à la fois enrichissante et formatrice.

2.2 Maîtrise d'ouvrage

Le rôle de la maîtrise d'ouvrage revient à la personne qui exprime les besoins et les attentes autour du produit final, joue le rôle de « client » et valide les jalons clés du projet.

Dans le cas de **NIS CUBE**, cette mission a été assurée par **M. Nicolas Denis**, qui a défini les objectifs initiaux, orienté les choix fonctionnels et validé l'ensemble des documents livrés tout au long du projet.

Grâce à ses retours fréquents et clairs, nous avons pu ajuster notre approche et améliorer continuellement notre travail pour nous rapprocher d'une solution utile, pertinente et conforme à la directive **NIS2**.

Il est également intervenu à plusieurs reprises pour nous aider à dépasser certaines difficultés techniques ou méthodologiques. Ces échanges ont largement contribué à la qualité finale du projet, en garantissant une bonne adéquation entre les attentes et le produit réalisé.

3 Spécifications fonctionnelles et techniques

3.1 Exigences fonctionnelles principales

| Titre de l'exigence | Description |
|---|--|
| Authentification sécurisée avec choix du type d'entité (EE ou EI) | Lors de l'inscription, l'utilisateur est invité à choisir le type d'entité qu'il représente. Ce choix conditionne l'affichage des questionnaires adaptés. La connexion est protégée par un mot de passe sécurisé, avec une option d'authentification par code à durée éphémère envoyée à l'adresse électronique communiquée. |
| Export des résultats | Offre la possibilité d'exporter l'ensemble des résultats et recommandations aux formats PDF et CSV. |
| Questionnaires dynamiques avec sauvegarde de progression | L'application propose des questionnaires modulaires selon le profil utilisateur. Elle enregistre automatiquement l'avancement de l'utilisateur afin qu'il puisse reprendre son évaluation à tout moment. |
| Scoring automatique avec transparence des calculs | Un moteur de règles calcule les scores par domaine et global, selon des pondérations définies, garantissant une évaluation traçable et cohérente. |
| Statistiques globales et par utilisateur | L'application propose une visualisation des tendances : taux de conformité moyen, répartition des scores par domaine, nombre total d'évaluations effectuées, etc. |
| Historique des évaluations | L'application conserve un historique des évaluations réalisées pour chaque entité. L'utilisateur peut consulter, comparer ou rééditer une évaluation antérieure à des fins de suivi et d'amélioration continue. |

3.2 Exigences non fonctionnelles

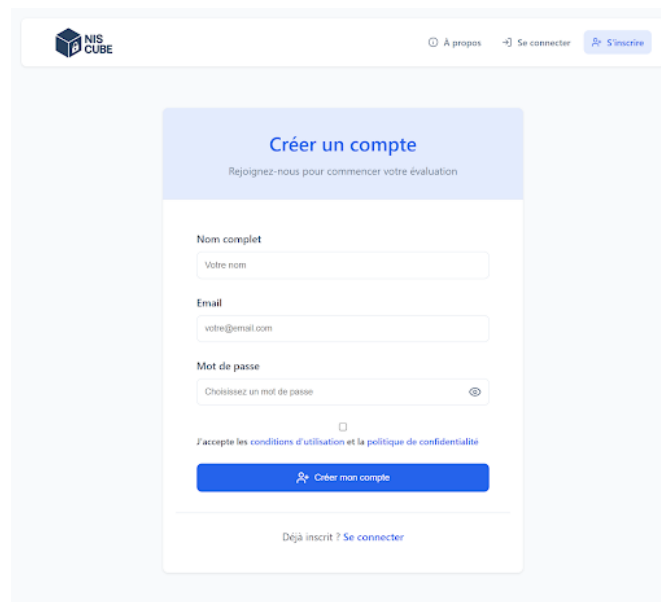
| Titre de l'exigence | Description |
|-----------------------------------|---|
| Une interface simple et intuitive | Les utilisateurs finaux ne sont pas toujours des techniciens ou experts en informatique. Ils souhaitent une application facile à utiliser, rapide à prendre en main, avec des fonctionnalités accessibles en quelques clics, même en situation d'urgence. |

| | |
|------------------------|--|
| Sécurité des données | Les données utilisateurs sont protégées via un chiffrement en transit et au repos. Le respect du RGPD est assuré, avec notamment la possibilité de suppression intégrale des données à la demande. |
| Maintenabilité du code | L'architecture logicielle suit une logique modulaire, facilitant la lisibilité, les mises à jour et les évolutions futures. Une documentation technique complète accompagne le projet pour garantir sa pérennité. |
| Compatibilité | L'application doit être compatible avec les principaux navigateurs web (Chrome, Firefox, Edge). |
| Tolérance aux pannes | L'application doit pouvoir gérer les erreurs sans interruption brutale du service. En cas d'incident (perte de connexion, erreur serveur...), des messages clairs doivent être affichés à l'utilisateur, et un mécanisme de reprise doit être prévu lorsque cela est possible. |
| Scalabilité | L'application doit pouvoir gérer une augmentation progressive du nombre d'utilisateurs ou de données sans dégradation significative des performances. L'architecture doit permettre l'ajout de ressources ou de modules sans nécessiter une refonte complète. |
| Traçabilité | Toutes les actions sensibles (connexions, modifications de données, suppressions) doivent être journalisées. Ces logs doivent être accessibles de manière sécurisée pour permettre une analyse en cas d'incident. |

4 Produit

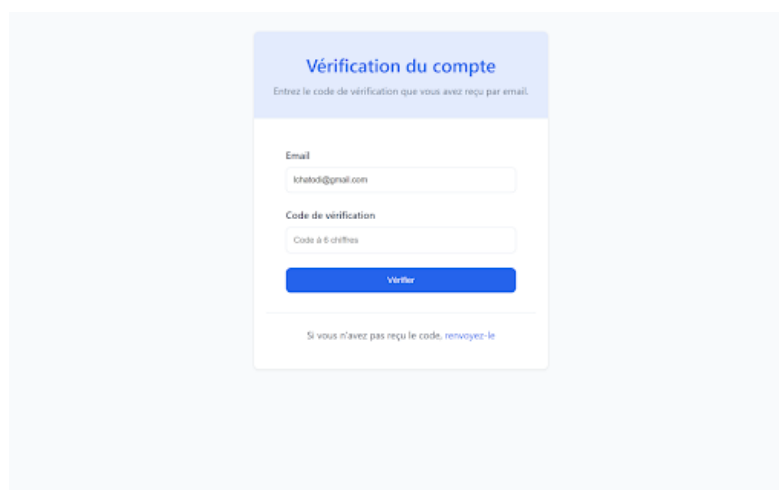
4.1 Inscription & Connexion

Pour accéder à l'ensemble des fonctionnalités de l'application, l'utilisateur doit obligatoirement s'authentifier. Deux options s'offrent à lui : se connecter avec un compte existant ou en créer un nouveau via l'interface prévue à cet effet.



The screenshot shows the 'Créer un compte' (Create account) form. At the top, there is a header with the NIS CUBE logo and links for 'À propos', 'Se connecter', and 'S'inscrire'. The form itself has a blue header with the title 'Créer un compte' and a subtitle 'Rejoignez-nous pour commencer votre évaluation'. The form contains three input fields: 'Nom complet' (Full name) with placeholder text 'Votre nom', 'Email' with placeholder text 'votre@email.com', and 'Mot de passe' (Password) with placeholder text 'Choisissez un mot de passe' and a toggle icon. Below the password field is a checkbox for 'J'accepte les conditions d'utilisation et la politique de confidentialité'. At the bottom of the form is a blue button labeled 'Créer mon compte' and a link 'Déjà inscrit ? Se connecter'.

Inscription Lors de son inscription, l'utilisateur n'a qu'à fournir une adresse e-mail et un mot de passe. Afin de limiter la collecte de données personnelles, seules ces informations sont requises pour permettre à l'utilisateur de s'authentifier au sein de l'application. Aucune autre donnée personnelle n'est demandée.



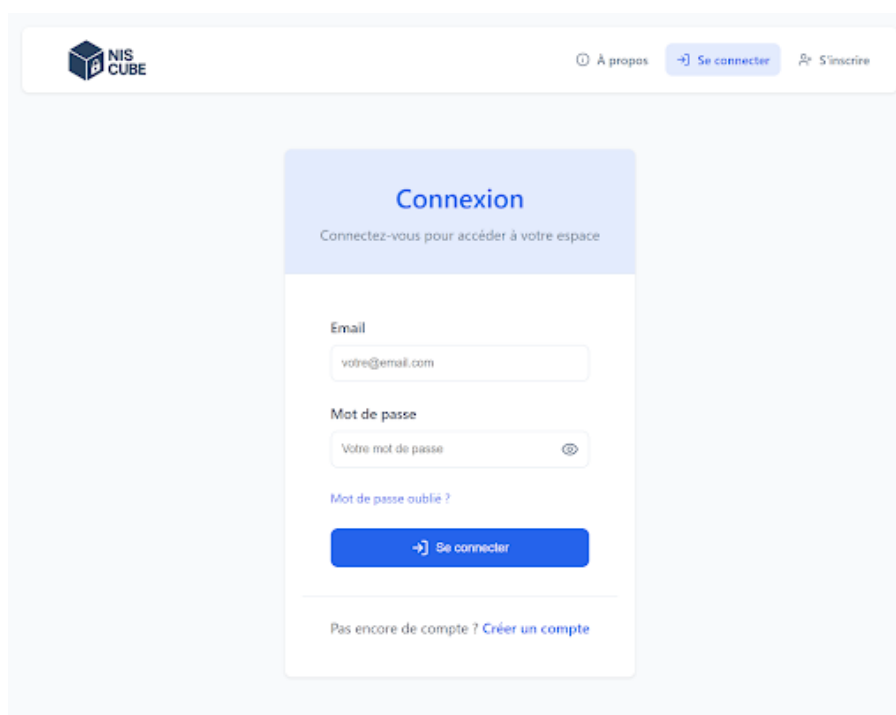
The screenshot shows the 'Vérification du compte' (Account verification) form. It has a blue header with the title 'Vérification du compte' and a subtitle 'Entrez le code de vérification que vous avez reçu par email'. The form contains two input fields: 'Email' with placeholder text 'khatrd@gmail.com' and 'Code de vérification' with placeholder text 'Code à 6 chiffres'. Below the code field is a blue button labeled 'Vérifier'. At the bottom of the form is a link 'Si vous n'avez pas reçu le code, renvoyez-le'.

Vérification compte Après son inscription, l'utilisateur reçoit par e-mail un code de vérification. Pour finaliser la création de son compte, l'utilisateur doit saisir

ce code sur cette page. Cette étape permet de sécuriser l'accès et de valider l'adresse e-mail fournie.

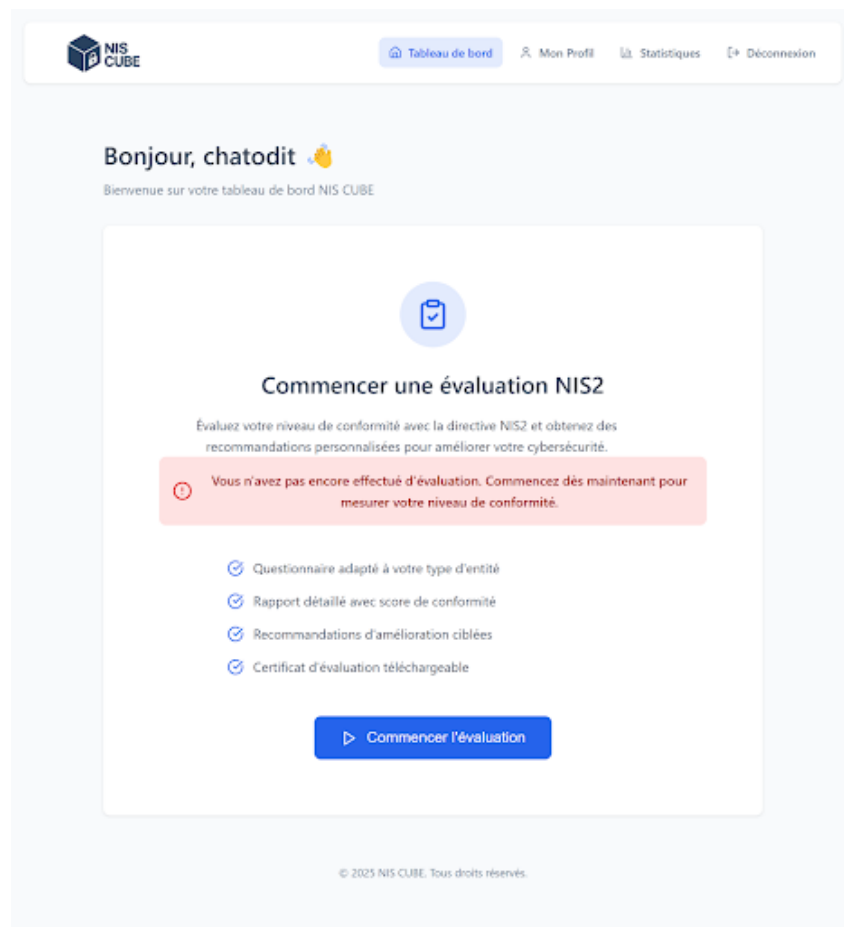


L'utilisateur doit saisir le code ci-dessous sur la page de vérification afin de finaliser la création de son compte.

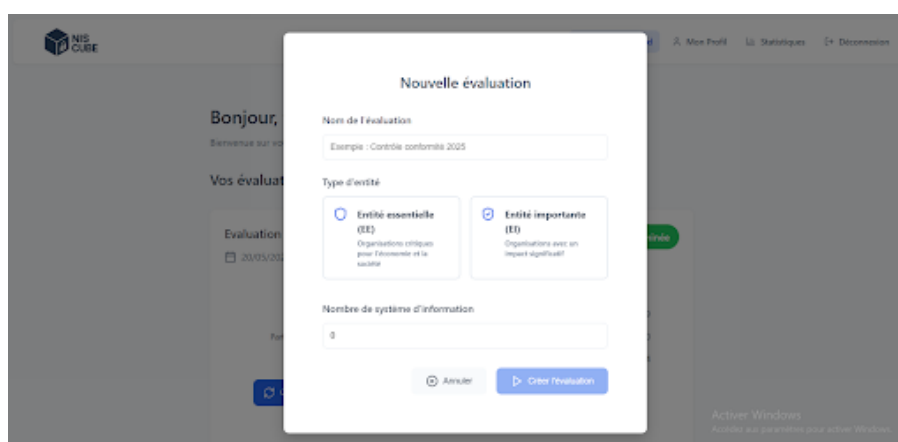
A screenshot of the NIS CUBE website's login page. The header includes the NIS CUBE logo and navigation links: 'À propos', 'Se connecter', and 'S'inscrire'. The main content area is titled 'Connexion' with the subtitle 'Connectez-vous pour accéder à votre espace'. It features two input fields: 'Email' (containing 'votre@email.com') and 'Mot de passe' (containing 'Votre mot de passe' and a toggle icon). Below the password field is a link 'Mot de passe oublié ?'. A blue 'Se connecter' button is positioned below the inputs. At the bottom, there is a link 'Pas encore de compte ? Créer un compte'.

Connexion Pour accéder à son compte, l'utilisateur doit saisir son adresse e-mail et son mot de passe. Ces deux informations suffisent à authentifier l'utilisateur, conformément à notre volonté de minimiser la collecte de données personnelles.

4.2 Utilisateur



Sur cette page, l'utilisateur retrouve l'ensemble des évaluations qu'il a déjà effectuées, si c'est le cas. Il peut y consulter les détails ou simplement en garder une trace. Un bouton dédié permet également à l'utilisateur de lancer une nouvelle évaluation à tout moment.



Lorsque l'utilisateur clique sur le bouton pour démarrer une nouvelle évaluation, une fenêtre pop-up s'affiche. Celle-ci permet à l'utilisateur de renseigner les

informations nécessaires : le nom de l'évaluation, le type d'entité de l'organisation (essentielle ou importante), ainsi que le nombre de systèmes d'information présents au sein de l'organisation.



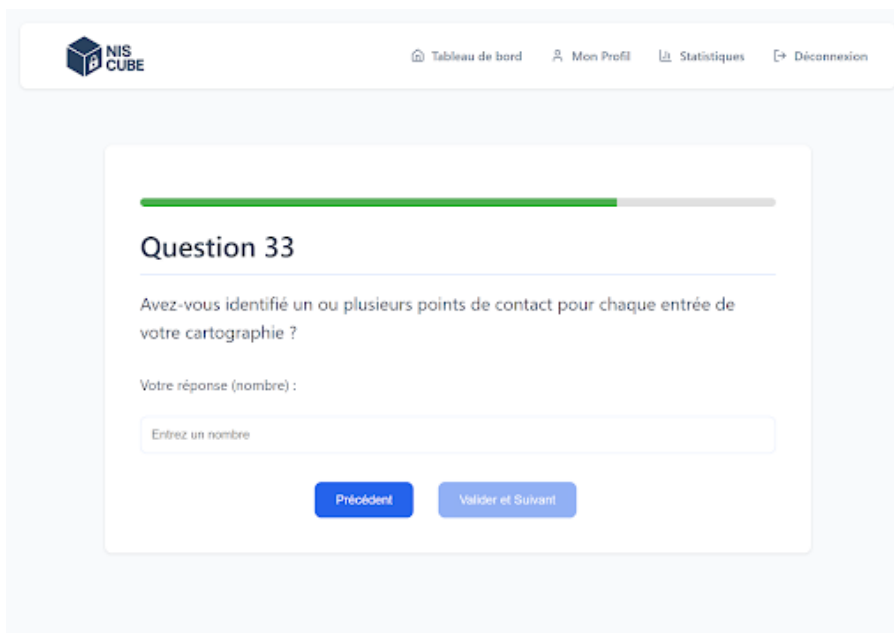
Votre Axe Actuel : Structurer

Objectif 5

- Description de l'objectif de sécurité
R1_5 - Organiser l'audit : L'entité planifie des audits de cybersécurité réguliers sur ses SI réglementés.
- Justifications et Risques Associés
Sans audit, des failles peuvent passer inaperçues et compromettre la sécurité globale de l'entité.

Suivant

Avant de répondre aux questions, l'utilisateur est informé de l'objectif de sécurité concerné ainsi que de l'axe (ou domaine) de la politique NIS2 auquel ces questions se réfèrent, afin de bien comprendre le contexte et les enjeux de l'évaluation.



Question 33

Avez-vous identifié un ou plusieurs points de contact pour chaque entrée de votre cartographie ?

Votre réponse (nombre) :

Entrez un nombre

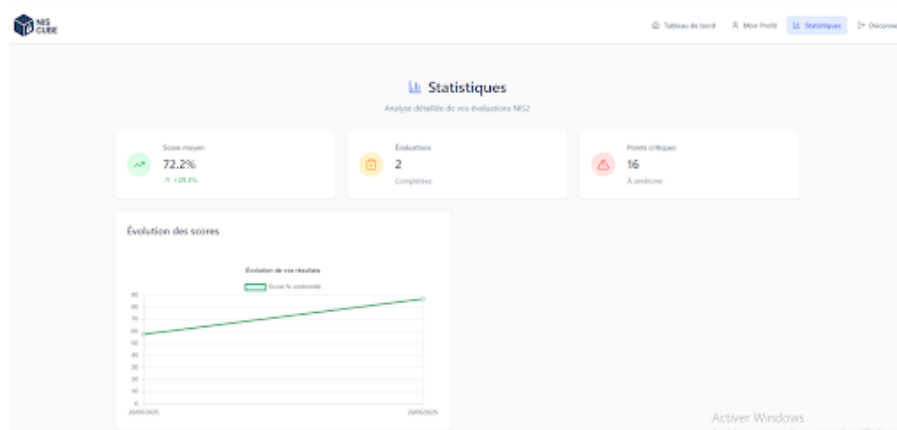
Précédent **Valider et Suivant**

Pour chaque question, l'utilisateur pourra répondre selon le format adapté : en saisissant un texte, en sélectionnant une date ou en choisissant entre « Oui » et « Non ». Tout au long de son évaluation, l'utilisateur bénéficie d'une barre

de progression lui permettant de visualiser son avancement et de mieux se situer dans le processus.



Sur la page des résultats, l'utilisateur peut visualiser l'état de conformité de son évaluation. Chaque objectif de la politique NIS2 est représenté sous forme de cube coloré : la couleur indique le niveau de conformité atteint pour cet objectif. En sélectionnant un cube, l'utilisateur accède, à droite de l'écran, au détail des questions associées, avec pour chacune l'indication de sa conformité ou non, ainsi que des recommandations adaptées.



Sur la page Statistiques, l'utilisateur peut consulter le score moyen obtenu sur l'ensemble de ses évaluations, ainsi qu'un graphique illustrant l'évolution de ce score dans le temps, lui permettant de suivre ses progrès en matière de conformité.

5 Technologies utilisées

5.1 Langages

JavaScript

JavaScript a été le langage principal utilisé pour le développement de notre projet, à la fois côté serveur et côté client. Sa polyvalence permet de manipuler aisément les interactions dynamiques dans l'interface utilisateur ainsi que de gérer les requêtes serveur. Ce choix est motivé par sa large adoption, la richesse de son écosystème, ainsi que la facilité d'intégration avec les frameworks modernes comme Vue.js et Express.

SQL (Structured Query Language)

Le langage SQL a été utilisé pour la définition, la manipulation et la gestion des données dans la base de données PostgreSQL. Il permet de réaliser des opérations complexes de sélection, d'insertion, de mise à jour et de suppression des données de manière efficace et sécurisée.

5.2 Outils de développement

| Outil | Description |
|--------------------|---|
| Visual Studio Code | Visual Studio Code est un éditeur de code source puissant et extensible. Il offre un support avancé pour JavaScript, SQL, ainsi qu'une intégration avec les systèmes de contrôle de version (Git). Son interface intuitive et ses nombreuses extensions ont permis d'améliorer la productivité lors du développement, du débogage, et des tests de l'application. |
| pgAdmin | pgAdmin est une interface graphique dédiée à l'administration de bases de données PostgreSQL. Il a été utilisé pour concevoir, gérer et monitorer la base de données du projet, facilitant l'exploration des tables, la rédaction de requêtes SQL, ainsi que la gestion des connexions. |
| Postman | Postman est un outil de test et développement d'API. Il nous a permis de simuler des requêtes HTTP (GET, POST, PUT, DELETE, etc.) vers notre serveur Express, de tester les différents endpoints, d'analyser les réponses et de valider le comportement attendu de notre API sous différents scénarios. |

| Outil | Description |
|-----------|---|
| MobaXterm | Nous avons utilisé le logiciel MobaXterm pour effectuer l'ensemble des opérations nécessaires à la configuration du VPS, telles que le déploiement des fichiers du projet, la mise en place du serveur Nginx, ainsi que l'édition et la gestion des fichiers de configuration. Cet outil nous a offert une interface pratique combinant terminal SSH et explorateur de fichiers, facilitant ainsi l'administration à distance du serveur. |

5.3 Framework

| Framework | Description |
|------------|--|
| Vue.js | Vue.js est un framework JavaScript progressif conçu pour construire des interfaces utilisateur interactives et réactives. Sa facilité d'intégration et son architecture component-based ont permis de créer une expérience utilisateur fluide et modulable. Vue.js s'adapte aussi bien pour des projets simples que complexes grâce à ses fonctionnalités modernes comme la gestion de l'état et le routing. |
| Vite.js | Vite est un outil de build moderne et un serveur de développement rapide conçu pour les applications front-end. Il supporte nativement ES Modules, permettant un hot-reload instantané lors du développement Vue.js, ce qui améliore considérablement la vitesse de développement et le temps de feedback. Vite optimise aussi la compilation pour la production, réduisant les temps de chargement. |
| Express.js | Express est un framework léger et flexible pour Node.js, utilisé pour développer l'API backend. Il permet de structurer rapidement les routes HTTP, de gérer les middlewares pour les fonctionnalités telles que l'authentification et la gestion des erreurs, et de connecter l'application à la base de données. Ce framework est idéal pour développer des applications web performantes et évolutives. |

| Framework | Description |
|-----------|--|
| Jest | Jest est un framework de test JavaScript développé par Facebook, conçu pour tester des applications JavaScript modernes. Il offre une configuration simple, des tests unitaires et d'intégration, ainsi qu'un support pour les tests asynchrones. Dans notre projet, Jest a été utilisé pour garantir la qualité du code en vérifiant le bon fonctionnement des composants Vue.js et des routes Express, permettant ainsi de détecter rapidement les régressions et d'assurer une couverture de test adéquate. |

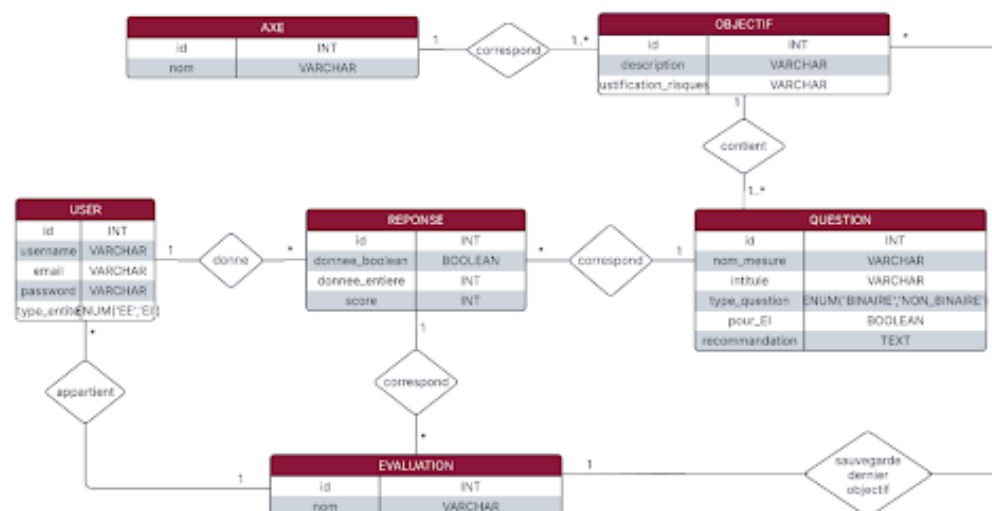
5.4 Base de donnée

PostgreSQL (psql)

PostgreSQL est un système de gestion de base de données relationnelle avancé, libre et open source, reconnu pour sa robustesse, sa conformité aux standards ANSI SQL, et ses fonctionnalités avancées (transactions, contraintes, index, types de données complexes). Elle a été choisie pour stocker et gérer efficacement les données relationnelles du projet tout en assurant la cohérence et la sécurité.

6 Architecture du projet

6.1 schéma relationnel de la base de données



À partir du MCD, nous avons établi le Modèle Logique de Données (MLD) en tenant compte des exigences du projet (types de données, clés primaires, contraintes, etc.). Ce MLD se concrétise sous la forme de cinq tables : **AXE**,

OBJECTIF, QUESTION, USER, REPONSE, EVALUATION. Cette section présente la structure relationnelle de la base de données pour le projet **NIS CUBE**. Chaque table est décrite ci-dessous avec quelques descriptions et contraintes :

- **Table AXE**

Un *AXE* représente un ensemble d'objectifs NIS2. Chaque axe regroupe plusieurs objectifs.

- **Table OBJECTIF**

La table *OBJECTIF* appartient à un *AXE*. La clé primaire est composite et se compose de (*objectif_id*, *axe_id*).

- **Table USER**

Chaque utilisateur représente une entité. Le champ *email* est utilisé pour l'authentification et l'envoi de codes MFA.

- **Table EVALUATION**

Chaque évaluation est initiée par un utilisateur (*user_id*) et peut être nommée pour permettre une meilleure organisation des sessions d'évaluation. Le champ *id_dernier_objectif* permet à l'utilisateur de reprendre son évaluation là où il s'était arrêté.

- **Table QUESTION**

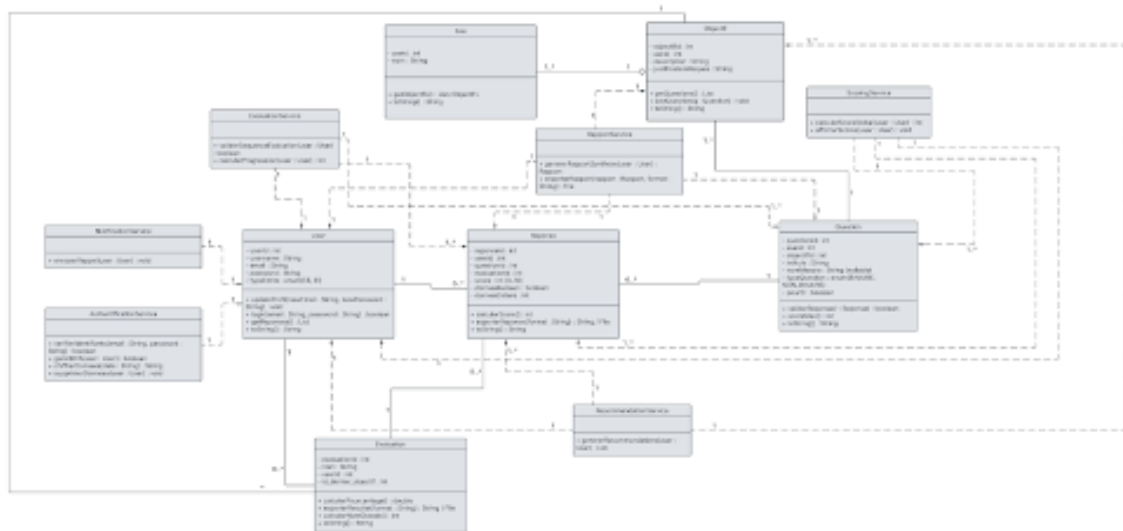
Chaque question dépend d'un *OBJECTIF*.

- **Table REPONSE**

La table *REPONSE* enregistre les réponses (ou scores) des utilisateurs pour chaque question dans une évaluation donnée. La clé primaire est composite et se compose de (*reponse_id*, *evaluation_id*, *user_id*, *question_id*).

Ainsi, la base de données permet de lier les utilisateurs, leurs réponses, les questions, les évaluations, les objectifs et les axes. Ce modèle relationnel offre la flexibilité nécessaire pour gérer les différentes configurations de questions et pour distinguer les entités essentielles (EE) des entités importantes (EI).

6.2 diagramme de classes



6.3 architecture du backend

Le backend de l'application est conçu selon une architecture modulaire **MVC** (Modèle – Vue – Contrôleur). Cette structure permet une séparation claire des responsabilités, facilitant la lecture, la maintenance et les évolutions futures du code. Le serveur est construit avec **Express.js**, interfacé avec une base de données PostgreSQL.

| Élément de la structure front-end | Description |
|-----------------------------------|---|
| /controllers/ | Les contrôleurs traitent les demandes reçues depuis les routes. Ils orchestrent l'exécution des actions, comme les lectures ou écritures en base, les vérifications, ou les appels à des services. |
| /routes/ | Les routes définissent les endpoints de l'API REST . Chaque entité de l'application a son propre fichier de route, facilitant la lisibilité et la modularité. |
| /models/ | Un modèle est responsable de l'accès aux données. C'est l'interface entre ton application et la base de données. Il encapsule les structures de données et les requêtes SQL. |

| | |
|--------------|---|
| /middleware/ | Les middlewares sont utilisés pour intercepter et traiter les requêtes avant qu'elles n'atteignent les routes ou les contrôleurs. |
|--------------|---|

6.4 Architecture du frontend

L'organisation du code front-end repose sur une architecture claire et modulaire, facilitant la maintenance, l'évolutivité et la lisibilité du projet. Chaque dossier remplit un rôle précis, qu'il s'agisse de la gestion des vues, de la logique métier, des composants réutilisables ou des ressources statiques. La structure ci-dessous décrit les principaux dossiers présents dans le projet ainsi que leur fonction respective.

| Élément de la structure front-end | Description |
|-----------------------------------|--|
| /views/ | Regroupe les vues principales de l'application correspondant à chaque page routée (connexion, inscription, tableau de bord, évaluation, résultats, statistiques. . .). Chaque fichier représente une page fonctionnelle. |
| /assets/ | Contient les fichiers statiques comme les logos, icônes, illustrations ou images utilisées dans l'UI. |
| /components/ | Vues réutilisables. Organisation possible en sous-dossiers par fonction (navigation, formulaire, modales, etc.). |
| /components/evaluation/ | Contient les composants spécifiques à l'évaluation : affichage des scores, cubes colorés, objectifs, détails de conformité, etc. |
| /composables/ | Contient des fonctions TypeScript personnalisées réutilisables dans plusieurs composants (ex : gestion des dates, validation de formulaire, appels API, logique de calcul de score. . .). |
| /store/ | Dossier contenant la gestion d'état globale de l'application (avec Pinia ou Vuex), notamment le fichier <code>evaluationStore.ts</code> pour stocker et gérer les données liées aux évaluations. |
| /test/ | Contient les tests unitaires ciblant les composants critiques, les vues dynamiques ou la logique métier. |

| Élément de la structure front-end | Description |
|-----------------------------------|---|
| .env.production | Fichier de configuration des variables d'environnement pour la production : URL de l'API, clé publique, etc. Protège les paramètres sensibles et facilite les déploiements. |
| style/ | Regroupe les fichiers CSS/SCSS globaux (variables, reset, typographie, couleurs...) appliqués à l'ensemble de l'application pour assurer une cohérence visuelle. |

7 Challenges affrontés

7.1 Gestion du temps :

- Il a fallu apprendre à **hiérarchiser les priorités**, planifier les tâches avec rigueur et respecter des échéances parfois serrées.
- L'utilisation d'un **diagramme de Gantt** a été déterminante pour structurer notre avancement.
- La **rotation du rôle de chef de projet** nous a permis de maintenir une dynamique constante et une bonne répartition des responsabilités.

7.2 Maîtrise de nouvelles technologies :

- Nous avons dû apprendre à utiliser des outils encore peu maîtrisés tels que **Vue.js** pour le front-end, **PostgreSQL** pour la base de données, et **Nginx** pour la configuration du serveur.
- Le **déploiement sur un VPS** a nécessité une compréhension de l'environnement Linux et des configurations réseaux.
- Cet apprentissage progressif a été facilité par une **répartition équilibrée des tâches techniques**, favorisant l'entraide et la montée en compétence collective.

7.3 Appropriation des notions de cybersécurité :

- Le projet nous a amenés à explorer des notions complexes mais essentielles comme :
 - **Le chiffrement des données utilisateurs**
 - **Le respect des normes RGPD**

7.4 Analyse de la directive NIS2 :

- La directive étant **récente, technique et parfois ambiguë**, il a été nécessaire de faire un **travail d'interprétation réglementaire**.

- Nous avons traduit les exigences en **critères d'évaluation mesurables**, en lien direct avec des fonctionnalités précises du logiciel.

7.5 Développement des compétences transversales :

- Le projet nous a permis de renforcer des aptitudes essentielles dans un cadre professionnel, telles que :
 - **Le travail en équipe**, basé sur l'écoute et la complémentarité
 - **La communication claire**, aussi bien orale qu'écrite
 - **La gestion de projet agile**, avec des ajustements fréquents face aux imprévus
 - **L'autonomie et l'implication** de chaque membre pour garantir la réussite collective

8 Planification

8.1 Répartition des tâches

8.1.1 Documentation

L'ensemble de l'équipe a contribué activement à l'élaboration des livrables du projet. Cette étape de documentation a été réalisée de façon collaborative, dans le but de garantir une compréhension partagée des enjeux fonctionnels et techniques.

8.1.2 Développement

Pour la phase de développement, nous avons réparti les tâches de manière équilibrée : deux membres de l'équipe se sont concentrés sur le back-end, tandis que les deux autres ont pris en charge le front-end. Cette division du travail nous a permis d'avancer efficacement en parallèle. À la fin de chaque étape majeure, nous **réunissions** afin de mettre en commun nos avancées, assurer l'intégration harmonieuse entre les deux parties, et valider le bon fonctionnement de l'ensemble.

8.2 Diagramme de Gantt

8.3 Planning

Le diagramme de Gantt se divise en plusieurs phases essentielles :



- **Phase 1 : Conception et mise en place de la base de données (03/03/2025 - 11/03/2025)**
 - Création de la base de données.
 - Intégration des questions dans la base de données.
 - Mise en place des routes et endpoints pour l'API.
- **Phase 2 : Développement des interfaces utilisateur (17/03/2025 - 10/04/2025)**
 - Développement des pages principales : accueil, connexion, inscription, tableau de bord, profil, évaluation, résultats, historique.
 - Intégration des pages front-end avec le backend.
- **Phase 3 : Tests et finalisation du développement (14/04/2025 - 20/04/2025)**
 - Tests locaux des fonctionnalités.
 - Préparation du serveur et configuration des dépendances.
 - Déploiement et sécurisation du serveur.
- **Phase 4 : Documentation et soutenance (21/04/2025 - 25/04/2025)**
 - Rédaction des manuels d'utilisation et du rapport final.
 - Préparation à la soutenance.

9 Organisation par l'équipe

Dès les premières semaines du projet, nous avons mis en place une dynamique de travail collaborative et structurée. Chaque semaine, nous tenions une réunion avec notre encadrant, **M. Nicolas Denis**, afin de faire le point sur notre avancement, discuter des difficultés rencontrées, et obtenir des retours sur les livrables en cours. Ces échanges réguliers ont été importants pour maintenir une bonne cohésion d'équipe.

Nous avons choisi de suivre une **méthodologie en cycle en V**, adaptée à la nature du projet. Cette approche nous a permis de progresser étape par étape : de l'analyse des besoins à la conception, puis au développement, avant de terminer par les tests et la validation. Elle nous a offert une vision claire du déroulement global, tout en structurant nos phases de travail.

Pour favoriser l'implication de chacun, nous avons instauré le **rôle tournant de chef de projet**. Chaque semaine, l'un d'entre nous prenait la responsabilité d'organiser les tâches, d'assurer la communication interne, de rédiger le compte-rendu de réunion, et de faire le lien avec notre encadrant. Cela nous a permis de développer nos compétences en gestion de projet, tout en maintenant une charge de travail équitable.

Nous avons aussi veillé à rester **flexibles** et à nous adapter aux aléas du projet. Chacun a participé selon ses points forts, que ce soit en développement, en conception graphique, en tests ou en documentation. Cette complémentarité a largement contribué à la réussite du projet **NIS CUBE**.

10 Compétences acquises

10.1 Compétences techniques

| Compétence | Description |
|--|---|
| Développement Full-Stack | Au cours de ce projet, nous avons acquis une compréhension approfondie du développement full-stack, ce qui signifie que nous avons appris à construire une application complète, allant de l'interface utilisateur (frontend) à la gestion du serveur et de la base de données (backend). Cette expérience nous a permis de maîtriser les technologies frontales, pour créer des interfaces utilisateur attrayantes et intuitives. Par ailleurs, nous avons également travaillé sur le backend, en utilisant des frameworks comme Express.js pour gérer les requêtes, les connexions à la base de données, tout en assurant la communication avec le modèle de données. |
| Gestion de base de données relationnelle | La conception et la manipulation d'une base de données PostgreSQL ont été des éléments clés de notre apprentissage. Nous avons appris à modéliser des données de manière efficace, en appliquant des principes de normalisation pour éviter la redondance et garantir l'intégrité des données. Cette compétence est essentielle pour concevoir des applications robustes et évolutives, tout en optimisant les performances de l'application et en garantissant une bonne expérience utilisateur. |
| Cybersécurité | Le projet NIS CUBE nous a également sensibilisés aux enjeux cruciaux de la cybersécurité. Nous avons mis en pratique les concepts fondamentaux tels que le chiffrement des données et les accès non autorisés. Nous avons sécurisé les points d'échange entre le frontend et le backend grâce à des mécanismes d'authentification sécurisés. En intégrant ces notions dans notre architecture, nous avons pu développer des solutions robustes, protégeant les informations sensibles et assurant la sécurité des systèmes d'information. |

| Compétence | Description |
|--|---|
| Serveur, déploiement et mise en production | Enfin, nous avons appris à déployer et à mettre en production une application dans un environnement réel. Grâce à l'utilisation de serveurs privés virtuels (VPS), de Nginx pour le proxy des requêtes HTTP, et de PM2 pour le déploiement et la gestion des processus Node.js, nous avons acquis des compétences pratiques en matière de déploiement d'applications sécurisées et stables. L'utilisation de certificats SSL a également été essentielle pour garantir la sécurité des communications entre le serveur et les utilisateurs. |

10.2 Compétences méthodologiques

| Compétence | Description |
|--------------------------|--|
| Organisation de projet | La mise en place d'une gestion en cycle en V nous a appris à planifier nos étapes, suivre les livrables et assurer une progression structurée. |
| Documentation rigoureuse | Rédiger un cahier des charges, un plan de tests, un manuel d'utilisation ou une documentation technique nous a permis de mieux formaliser nos travaux. |
| Tests et validation | Nous avons compris l'importance des tests unitaires pour garantir la fiabilité du logiciel. |

10.3 Compétences transversales et humaines

| Compétence | Description |
|-------------------|--|
| Travail en équipe | La répartition des rôles, la communication constante et la complémentarité des compétences ont été essentielles à la réussite du projet. |
| Gestion du temps | Concilier travail académique, contraintes techniques et échéances nous a appris à prioriser nos actions. |

| Compétence | Description |
|---------------------------------|---|
| Autonomie et auto-apprentissage | Face aux défis techniques et réglementaires, nous avons développé notre capacité à apprendre par nous-mêmes et à résoudre les problèmes de manière proactive. |

11 Perspectives d'amélioration

| Amélioration | Description |
|-------------------------------|--|
| Planification d'évaluations | Donner la possibilité de planifier des évaluations récurrentes (ex : une fois par trimestre), avec alertes ou notifications automatiques pour respecter un calendrier de conformité. |
| Benchmarking anonyme | Proposer aux utilisateurs (avec leur accord) de comparer anonymement leurs résultats avec ceux d'autres organisations similaires (même secteur, taille, pays...), pour se situer par rapport au marché. |
| Traduction multilingue | Si tu cibles plusieurs pays européens, prévoir l'internationalisation (i18n) pour proposer l'interface en plusieurs langues (français, anglais, allemand...). |
| Système de suivi ou de rappel | Mettre en place un module de suivi de mise en conformité avec la possibilité pour l'utilisateur de marquer des objectifs comme "en cours", "corrigé", ou "à revoir", avec des rappels ou des notifications pour les prochaines actions à réaliser. |

12 Usage de l'intelligence artificielle dans le projet

L'intelligence artificielle a été un soutien précieux dans la conduite du projet **NIS CUBE**. Nous avons mobilisé différents agents conversationnels pour accélérer certaines tâches, approfondir notre compréhension et améliorer la qualité de notre production. Chaque outil a été utilisé selon ses points forts :

- **ChatGPT (OpenAI)** : principalement utilisé pour la **rédaction des documents** liés au projet (cahier des charges, manuel utilisateur, etc.). ChatGPT nous a aidés à structurer nos idées, améliorer la clarté de nos formulations, et homogénéiser le ton du document. Son apport a été particulièrement utile pour la mise en forme de textes complexes et la synthèse de contenus techniques.
- **Qwen (Alibaba)** : utilisé dans les phases de **développement backend et logique métier**, notamment pour obtenir des suggestions de structuration d'API REST, gérer les flux de données dans le modèle MVC, et optimiser certains scripts.
- **Claude (Anthropic)** : sollicité principalement pour le **développement frontend et la revue de code**, Claude s'est avéré pertinent pour améliorer l'expérience utilisateur, proposer des ajustements ergonomiques et aider à la conception de composants Vue.js. Il a aussi apporté un regard critique sur la sécurité des interfaces et la cohérence générale de notre application.

Ces intelligences artificielles ont été utilisées comme **assistants techniques et rédactionnels**, jamais comme substituts à notre travail. Toutes les suggestions ont été relues, adaptées et intégrées avec discernement.

Cette collaboration avec l'IA a enrichi notre approche du projet, tout en développant notre esprit critique, notre autonomie et notre capacité à intégrer des outils modernes dans une démarche professionnelle et responsable.

13 Vérification de la documentation

La vérification de la documentation a joué un rôle central dans le bon déroulement du projet **NIS CUBE**. Elle nous a permis d'assurer la cohérence entre les différentes phases du développement — de la conception initiale jusqu'au produit final.

Tout au long du projet, nous nous sommes appuyés sur plusieurs documents clés, qui ont constitué des repères fiables pour orienter nos choix et valider chaque étape importante.

Cahier des charges

Ce document a posé les fondations du projet. Il a permis de définir clairement les objectifs, les besoins des utilisateurs, les fonctionnalités attendues ainsi que les contraintes à respecter. Il a guidé nos décisions dès les premières phases de conception.

Cahier de recette

Ce document a défini les scénarios de tests fonctionnels à suivre pour valider chaque fonctionnalité développée. Il a permis de s'assurer que toutes les exigences définies dans le cahier des charges étaient bien respectées.

Conception détaillée

Il s'agit du document technique de référence qui traduit les besoins fonctionnels en éléments concrets. On y retrouve :

- l'architecture générale de l'application ;
- les choix technologiques retenus ;
- le modèle relationnel de la base de données ;
- les diagrammes de classes.

Ce support nous a permis de structurer notre développement autour d'une logique claire et modulaire.

Plan de tests

Il a encadré les différentes étapes de validation technique du projet :

- tests unitaires,
- tests d'intégration,
- tests fonctionnels.

Il nous a aidés à détecter et corriger les anomalies au fil du développement, pour livrer un produit robuste et fiable.

Manuel d'installation

Ce guide à destination des techniciens détaille les étapes pour déployer l'application sur un serveur. Il décrit les prérequis, la configuration des dépendances, et les commandes à exécuter. Il permet une installation rapide, sans blocage.

Manuel d'utilisation

Ce document s'adresse aux utilisateurs finaux. Il explique comment créer un compte, répondre aux questionnaires, consulter les résultats, exporter les rapports, etc. Il a été conçu pour garantir une prise en main facile, même pour un public non technique.

En nous appuyant régulièrement sur ces documents, nous avons pu maintenir une bonne coordination d'équipe, assurer une couverture complète des besoins utilisateurs et garantir la qualité finale de la solution livrée, en adéquation avec la directive **NIS2**.

14 Synthèse du projet

Le projet **NIS CUBE** avait pour objectif de concevoir et développer une application web d'auto-évaluation de la conformité à la directive européenne **NIS2**. Notre démarche a été structurée autour d'un processus rigoureux : analyse des besoins, conception technique, développement, tests et déploiement.

Sur le plan technique, nous avons mis en œuvre une architecture complète :

- un **backend** en Node.js avec Express.js,
- un **frontend** en Vue.js avec Vite,
- une base de données relationnelle PostgreSQL,
- un **serveur VPS** sécurisé via Nginx.

Plusieurs aspects de **cybersécurité** ont été intégrés :

- authentification multi-facteurs,
- chiffrement des données sensibles,

La directive **NIS2**, bien qu'issue d'un cadre juridique et organisationnel, a été traduite en fonctionnalités concrètes :

- questionnaires dynamiques adaptés aux entités,
- système de scoring automatisé,
- export de rapports personnalisés.

Côté méthodologie, nous avons suivi un **cycle en V**, avec une organisation en sous-équipes et des points hebdomadaires avec notre encadrant. Cela nous a permis de piloter efficacement notre avancement et de documenter chaque phase (cahier des charges, conception, plan de tests, manuels...).

Ce projet a été l'occasion de mobiliser de nombreuses compétences en **développement full-stack**, en **modélisation des données**, en **cybersécurité** et

en **gestion de projet**. Il constitue une expérience complète, professionnalisante, que nous pourrions valoriser en entreprise ou en alternance.

15 Conclusion

Le projet **NIS CUBE** nous a permis de vivre une expérience concrète et professionnalisante, ancrée dans les enjeux actuels de cybersécurité et de conformité réglementaire. En répondant aux exigences de la directive **NIS2**, nous avons conçu une plateforme fonctionnelle, complète et intuitive, à destination des organisations souhaitant évaluer leur niveau de sécurité.

Ce travail nous a permis de mobiliser et de développer de nombreuses compétences :

- sur le plan technique : architecture web full-stack, développement sécurisé, déploiement sur serveur, gestion de base de données, etc. ;
- sur le plan méthodologique : travail en cycle en V, coordination d'équipe, gestion documentaire et planification ;
- sur le plan humain : communication, collaboration, adaptabilité et autonomie.

Nous sommes fiers du chemin parcouru, des défis surmontés, et du résultat livré. Ce projet représente une étape importante dans notre parcours académique, et un véritable tremplin pour notre avenir professionnel dans les domaines du développement, de la cybersécurité ou de la gestion de projet.

Nous remercions chaleureusement **M. Nicolas Denis** pour son accompagnement constant, ses conseils précieux et sa disponibilité tout au long du semestre.

Glossaire

- **API (Application Programming Interface)** : Interface qui permet à différentes applications de communiquer entre elles en définissant des méthodes et des conventions d'échange de données.
- **Authentification** : Processus de vérification de l'identité d'un utilisateur, souvent par le biais d'un nom d'utilisateur et d'un mot de passe.
- **Back-end** : Partie d'une application qui gère la logique métier, les bases de données et les serveurs, généralement invisible pour l'utilisateur final.
- **Base de données** : Système organisé pour stocker, gérer et récupérer des données. Dans ce projet, PostgreSQL est utilisé comme système de gestion de base de données.
- **Chiffrement** : Technique de sécurité qui transforme des données en un format illisible pour protéger leur confidentialité.
- **Directive NIS2** : Règlement européen visant à renforcer la sécurité des réseaux et des systèmes d'information au sein de l'Union européenne.
- **Frontend** : Partie d'une application avec laquelle l'utilisateur interagit directement, incluant l'interface utilisateur et l'expérience utilisateur.
- **Framework** : Ensemble d'outils et de bibliothèques qui facilitent le développement d'applications en fournissant une structure de base.
- **Gestion de projet** : Discipline qui consiste à planifier, exécuter et contrôler un projet pour atteindre des objectifs spécifiques dans un délai imparti.
- **MFA (Multi-Factor Authentication)** : Méthode d'authentification qui nécessite plusieurs preuves d'identité avant d'accorder l'accès à un utilisateur.
- **Modèle MVC (Modèle - Vue - Contrôleur)** : Architecture logicielle qui sépare une application en trois composants principaux : le modèle (données), la vue (interface utilisateur) et le contrôleur (logique métier).
- **Normes RGPD (Règlement Général sur la Protection des Données)** : Règlement de l'Union européenne qui protège la vie privée des citoyens et régle le traitement des données personnelles.
- **Scalabilité** : Capacité d'une application à gérer une augmentation du nombre d'utilisateurs ou de données sans dégradation des performances.
- **Sécurité des données** : Ensemble de pratiques et de technologies visant à protéger les données contre les accès non autorisés, la corruption ou la perte.
- **Tests** : Méthodologie de tests qui consiste à vérifier le bon fonctionnement du code et à identifier les erreurs.
- **VPS (Virtual Private Server)** : Serveur virtuel qui simule un serveur physique dédié, offrant un environnement isolé pour exécuter des applications.
- **Serveur mutualisé** : Serveur partagé entre plusieurs utilisateurs.