

Rapport : Conseils rédaction Rapport et Exemple de plan

Voici un plan général assez standard pour un rapport de développement logiciel, vous pouvez (devez) le personnaliser en fonction des spécificités de votre projet, le simplifier ou le complexifier selon votre projet.

Un seul compte-rendu est à rendre par groupe.

I. Conseils Généraux pour la Rédaction du Rapport

1. **Clarté et Précision :**

- ✓ Utilisez un langage clair et évitez le jargon non expliqué.
- ✓ Définissez les termes techniques lors de leur première apparition.

2. **Structure et Cohérence :**

- ✓ Suivez la structure proposée pour assurer une présentation logique.
- ✓ Assurez-vous que chaque section se connecte harmonieusement avec les autres.

3. **Illustrations et Schémas :**



- ✓ Intégrez des images, des tableaux et des graphiques pour soutenir vos explications.
- ✓ Légendez et numérotez toutes les figures, et référencez-les dans le texte.

4. **Références :**

- ✓ Citez toutes les sources externes utilisées
- ✓ Incluez une bibliographie ou une sitographie à la fin de votre rapport

5. **Orthographe et Grammaire :**

- ✓ Relisez attentivement votre rapport pour corriger les fautes.
- ✓ Utilisez des outils de correction grammaticale si nécessaire.

	Projet	
	RAPPORT DE PROJET V2	

6. **Formatage :**

- ✓ Respectez les consignes de mise en page fournies
 - Police : Calibri ou Times New Roman ou Calibri
 - Taille police 11 ou 12
 - Marges 1,27 cm maxi 2cm
- ✓ Utilisez des styles cohérents pour les titres, sous-titres et paragraphes.

7. **Originalité :**

- ✓ Évitez le plagiat en rédigeant avec vos propres mots et en citant correctement les sources.
- ✓ Mettez en avant vos réflexions personnelles et analyses critiques.

II. Les premières pages :

1. **Page de garde :**

- ✓ Titre du projet.
- ✓ Nom des étudiants / membres de l'équipe.
- ✓ Au milieu une image d'illustration
- ✓ Date de soumission.
- ✓ Nom du cours et de l'enseignant.

2. **Résumé :**

- ✓ Brève description du projet et des principaux résultats.

3. **Table des matières**

- ✓ Avec les numéros de pages

III. Première partie : partie commune

1. **Présentation :**

- ✓ Présentation du contexte.
- ✓ Description générale du projet.
- ✓ Objectifs et finalités du projet.

- ✓ Importance et justification du projet.

- ✓ Définition précise de ce qui est inclus et exclu du projet.
- ✓ Limites et contraintes du projet.

2. Analyse des besoins exigences fonctionnelles:

Analyse - Objectifs : Elle consiste à comprendre en profondeur les **besoins** et les **exigences** du client ou des utilisateurs finaux.

Durant cette phase, l'équipe de projet **recueille et documente les spécifications fonctionnelles** (ce que le système doit faire) et les spécifications **non fonctionnelles** (les contraintes de performance, de sécurité, d'ergonomie, etc.).

L'analyse permet d'identifier les **cas d'utilisation**, de développer des **scénarios** utilisateurs et de créer des modèles conceptuels de données (**MCD**).

En synthétisant ces informations, l'équipe établit une vision claire et partagée des objectifs du projet, ce qui sert de fondation pour la conception générale ultérieure

- ✓ Description détaillée des besoins du logiciel.
- ✓ Rédaction du cahier des charges fonctionnelles : liste détaillée des fonctionnalités du projet
- ✓ Spécifications fonctionnelles :
 - Description des interactions utilisateur.
 - Scénarios d'utilisation ou cas d'utilisation.
- ✓ Spécifications non fonctionnelles :
 - Exigences liées à la performance, la sécurité, l'ergonomie, etc.
 - Normes et réglementations à respecter.
- ✓ Contraintes techniques :
 - Plateformes et technologies à utiliser ou à éviter.
 - Compatibilité avec d'autres systèmes ou logiciels.

3. Cahier des charges : Sécurité – Cybersécurité



- ✓ Analyse des Risques de Sécurité
 - évaluer les menaces spécifiques au type de projet (logiciel ou site web).
 - Identifier les données sensibles qui seront manipulées et les conséquences potentielles de leur compromission.
- ✓ Conformité et Réglementations :
 - Examiner les lois et normes de sécurité applicables
 - RGPD pour les données personnelles,
 - la norme OWASP pour les applications web
- ✓ Définition des Exigences de Sécurité :
 - Par exemple, exiger l'authentification forte, le chiffrement des données sensibles, ou des contrôles d'accès spécifiques.
- ✓ Identification des Interactions Externes :
 - Analyser les interactions du projet avec des services tiers ou des API externes, qui peuvent être des points d'entrée pour les attaques.
 - Identifier les contrôles à mettre en place pour sécuriser ces échanges

- ✓ Solutions envisagées

4. Phases, planification et répartition des tâches

- ✓ Découpage du projet en différentes phases ou étapes.
- ✓ Estimation de la durée de chaque phase.
- ✓ Répartition équitable suivant les compétences de chacun(e)
- ✓ Jalons clés et dates importantes.

IV. Partie personnelle : une partie par étudiant(e)

1. Introduction

- ✓ Rappel des tâches à réaliser
- ✓ Positionnement

2. Fin de l'Analyse

- ✓ Identification des Scénarios et Cas d'Utilisation :
- ✓ Développez des scénarios d'utilisation typiques ou critiques.
- ✓ Créez des diagrammes de séquence pour illustrer ces scénarios.
- ✓ Recherche de solutions :
 - Recherche de solutions : parce qu'il n'y a pas qu'une seule solution
 - Explorer différentes options technologiques (langages, frameworks, outils) ou matériel et justifier vos choix

3. Conception :

a) Conception générale

Conception générale - Objectifs : Elle vise à élaborer une **vision globale de l'application** avant d'entrer dans les détails techniques de l'implémentation.

Durant cette phase, il s'agit de définir l'**architecture du système** en identifiant les principaux **composants, modules** ou **services**, ainsi que les interactions entre eux.

Cette représentation macro du projet permet **de visualiser la structure** et le flux de l'application, généralement à l'aide de diagrammes tels que les **diagrammes UML** (classes, séquences, activités) et les modèles conceptuels de données (**MCD**).

La conception générale facilite la communication entre les membres de l'équipe, assure une compréhension commune des objectifs et pose les bases pour la conception détaillée.

- ✓ Visualiser la structure et le flux de l'application.
 - Diagrammes UML : diagramme de classes, de séquences, d'activités, etc.

- Modèle Conceptuel de Données, MCD/MLD si base de données
- ...
- ✓ **Utilisation de diagrammes d'architecture logicielle pour décrire le haut niveau de l'application :**
 - Diagramme d'architecture logicielle et/ou matérielle
 - Le diagramme UML de déploiement peut-être utilisé, mais aussi visio, lucidchart ou draw.io
- ✓ Architecture générale du logiciel.
- ✓ Les difficultés rencontrées et les solutions apportées
- ✓ **Interface utilisateur :**
 - Présentation de l'interface.
 - Justification des choix de design d'interface et de l'expérience utilisateur.
 - Expérience utilisateur : l'accessibilité et l'ergonomie

b) Conception détaillée

Conception détaillée - Objectifs : Alors que la **conception générale** offre une vue d'ensemble de l'architecture du système, la **conception détaillée se concentre sur la spécification précise** de chaque composant individuel.

Durant cette phase, les développeurs élaborent des **descriptions approfondies** des modules, des classes, des interfaces, des bases de données et des algorithmes qui seront implémentés.

Ils utilisent des **diagrammes UML** détaillés tels que les diagrammes de classes, de séquences et d'états pour représenter les interactions internes et le comportement du système.

La conception détaillée sert de plan directeur pour la **phase d'implémentation**, assurant que tous les aspects techniques sont clairement définis.

Elle permet d'anticiper et de résoudre les problèmes potentiels avant le codage, ce qui réduit les risques d'erreurs et améliore l'efficacité du développement.

Différences entre conception générale et conception détaillée :

Conception Générale : Pose les **fondations** du projet en définissant **quoi** sera construit et **comment** les grandes parties du système interagiront entre elles.

Conception Détaillée : Fournit le **plan précis** pour la construction, détaillant **comment** chaque partie sera réalisée techniquement.

La conception générale assure que l'équipe est alignée sur la vision globale, tandis que la conception détaillée garantit que chaque développeur dispose des informations nécessaires pour coder efficacement sa partie du système.

Cette partie doit fournir une compréhension approfondie des éléments spécifiques et techniques de votre travail

- ✓ Architecture Logicielle Détaillée :
 - Description détaillée de l'architecture logicielle (par exemple, modèle MVC, microservices, etc.).
 - Illustration par des diagrammes d'architecture détaillés.
- ✓ Conception des Composants :
 - Description détaillée de chaque composant du système.
 - Utilisation de diagrammes UML tels que les diagrammes de classes, de séquences, d'états, etc.
 - Détails sur les méthodes, classes, et leur interaction.
- ✓ Conclusion :
 - Résumé des points clés de la conception détaillée.
 - Discussion sur l'impact de ces détails sur le projet global.

4. Implémentation :

Implémentation - Objectifs : les spécifications élaborées lors des phases de conception générale et détaillée sont concrètement réalisées.

C'est à ce stade que les développeurs traduisent les designs en code source, en utilisant les langages de programmation, frameworks et outils technologiques choisis précédemment.

L'implémentation consiste à créer les modules, classes, méthodes et interfaces définis,

- ✓ Détails techniques de la réalisation.
- ✓ Problèmes rencontrés et solutions adoptées.
- ✓ mais aussi les bonnes pratiques de codage et les patterns de conception utilisés si utilisés

5. Tests unitaires et validation :

Tests - Objectifs : Son objectif principal est de **vérifier et valider** que le logiciel fonctionne conformément aux spécifications définies lors des phases d'analyse et de conception.

Durant cette étape, différentes méthodes de test sont employées pour **détecter et corriger les bugs**, assurer la fiabilité, la performance et la qualité globale du produit.

Les **tests unitaires** sont effectués pour valider le bon fonctionnement des plus petites unités de code, comme les fonctions ou les méthodes individuelles.

Les **tests d'intégration** s'assurent que les différents modules ou composants du système interagissent correctement entre eux.

- ✓ Stratégie de tests.
- ✓ Cas de tests et scénarios.
- ✓ Résultats et interprétation.
- ✓ Bugs identifiés et leur résolution.
- ✓ Automatisation des tests

6. Tests d'intégration (collectif)

- ✓ Dans un contexte collectif, cela montre l'importance de la collaboration et de l'intégration continue.

7. Discussion :

- ✓ Retour sur les choix effectués.
- ✓ Limitations du logiciel actuel.
- ✓ Comparaison avec d'autres solutions existantes

V. Suite et fin de partie générale

1. Conclusion et perspectives :

- ✓ Bilan du projet.
- ✓ Axes d'amélioration et fonctionnalités futures.

2. Annexes:

- ✓ Lien dépôt Git
- ✓ Capture d'écrans.
- ✓ Résultats détaillés des tests.
- ✓ Tout autre élément jugé pertinent.

3. Bibliographie, sitographie et références :

- ✓ Sources, articles, livres, outils et technologies utilisés.