

Rapport : 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.

Les premières pages :

1. Page de titre :

- ✓ Titre du projet.
- ✓ Nom des étudiants / membres de l'équipe.
- ✓ 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

I. 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:

- ✓ 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. 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.

4. Budget et ressources :

- ✓ Estimation du coût total du projet.
- ✓ Détail des ressources nécessaires (humaines, matérielles, financières).

5. Critères d'acceptation :

- ✓ Conditions à remplir pour que le projet soit considéré comme terminé.
- ✓ Méthodes de validation et de vérification.

6. Risques et plan de gestion des risques :

- ✓ Identification des risques potentiels.
- ✓ Stratégies pour atténuer ou gérer ces risques.

II. 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

- ✓ **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

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.

	Projet	
	RAPPORT	

- ✓ Conclusion :
 - Résumé des points clés de la conception détaillée.
 - Discussion sur l'impact de ces détails sur le projet global.

4. Implémentation :

- ✓ 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 :

- ✓ 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

III. Suite et fin de partie générale

1. Conclusion et perspectives :

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

2. Annexes:

- ✓ Code source (ou un lien vers un 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.