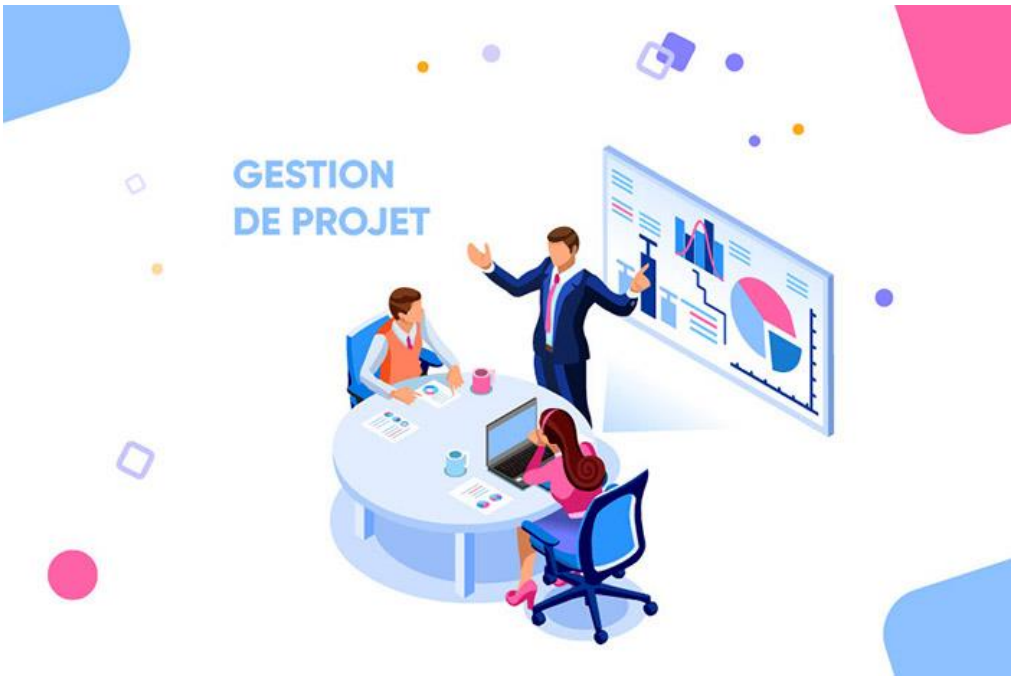
GABRIEL Adam
BOICHÉ Gauvain
LEULIETTE Louis

Contexte 3
I. Cadrer le besoin 3
1. Identifier les utilisateurs et parties prenantes. 3
2. Formuler le problème à résoudre et les bénéfices attendus 3
3. Lister les besoins fonctionnels et non fonctionnels 4
a. Besoins fonctionnels 4
b. Besoins non fonctionnels 4
4. Identifier les informations manquantes et préparer au moins 8 questions à poser au client 4
II. Définir les objectifs et le mérimètre 5
1. Objectifs SMART(E) 5
2. Périmètre du projet 6
a. Inclus dans le projet 6
b. Exclu du projet 6
3. Proposer un MVP réaliste pour la première version. 6
4. Priorisation MoSCoW 7
a. Must have 7
b. Should have 7
c. Could have 8
d. Won't have 8
II. Définir les objectifs et le périmètre 8
1. Rédiger 3 à 5 objectifs SMARTE. 9
III. Formaliser les besoins 9
1. Rédiger au minimum 8 user stories. 9
2. Ajouter des critères d'acceptation à au moins 4 user stories. 9
3. Identifier les principales contraintes techniques, de sécurité et d'utilisation. 10
IV. Organiser et estimer le travail 11
1. Découper le projet en grandes activités puis en tâches. 11
2. Estimer la charge ou la complexité des principales tâches. 12
3. Identifier les dépendances et les tâches parallélisables. 12
4. Proposer une répartition des responsabilités dans l'équipe. 13

V. Choisir la méthode de gestion 13
1. Comparer Cycle en V et AGILE pour ce projet..... 13
2. Choisir l'approche que vous estimez la plus adaptée et justifier votre choix. 14
3. Décrire l'organisation pratique retenue : rythme, réunions, validations et suivi. ... 15
VI- Construire un planning 16

GABRIEL Adam
BOICHÉ Gauvain
LEULIETTE Louis



Contrôle continu

Filière : Informatique

Option : Administrateur des systèmes et réseaux

Livecampus : École d'informatique en ligne

Bachelors et Mastères en IA, cybersécurité, data et développement

Thème : Gestion de projet

Encadré par :
AUMAJY Yannick

Présenté par :
GABRIEL Adam
BOICHÉ Gauvain
LEULIETTE Louis

Octobre 2026

Contexte

La société NovaServices compte 180 salariés répartis sur trois sites. Les demandes de congés sont encore gérées par formulaires PDF envoyés par e-mail.

Le service RH souhaite mettre en place une application web interne permettant de consulter les soldes, déposer une demande, faire valider les congés par le manager et disposer d'un historique.

La direction souhaite une première version utilisable dans quatre mois. Elle demande une solution simple, sécurisée et accessible depuis un ordinateur ou un smartphone.

I. Cadrer le besoin

1. Identifier les utilisateurs et parties prenantes.

Utilisateurs :

- Employés
- Managers
- RH / Gestionnaire de paie

2. Formuler le problème à résoudre et les bénéfices attendus

Problématique :

- 180 salariés
- Application web interne demandé par la RH
- Utilisable dans quatre mois
- Solution simple, sécurisée et accessible depuis un ordinateur ou un smartphone.

3. Lister les besoins fonctionnels et non fonctionnels

a. Besoins fonctionnels

- Se connecter / Se déconnecter

- Consulter son solde de congé

- Poser un congé / Agir dynamiquement sur ces demandes

- Vérifier les demandes de congés (en cours et historique)

- Pour le manager -> Approuver ou refuser

- Délais (Ne peut pas poser un congé du jour au lendemain)

b. Besoins non fonctionnels

- Échange chiffré et compartimentation des comptes

- Compatible ordinateur et téléphone

- Capacité -> 200 Utilisateurs simultanés (Peut évoluer)

- Pour la perte de données -> Sauvegarde quotidienne

- Conformité RGPD (sécurisation des comptes et obfuscation de la base de données)

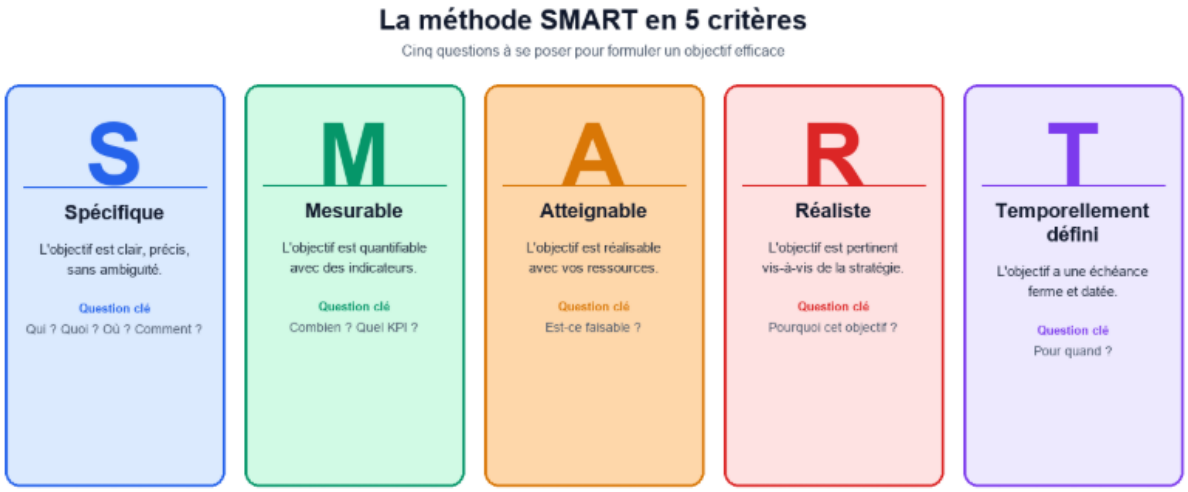
4. Identifier les informations manquantes et préparer au moins 8 questions à poser au client

1) Comment sont gérés les comptes utilisateurs ?
2) Peut-on escalader les demandes en cas de d'absence d'un manager ?
3) Est-ce qu'il y a une visibilité sur les congés déjà poser par les collègues du même service ?
4) Quels sont les délais de validation après demande de l'employé ?
5) Quel est votre budget ?
6) Qui assure la maintenance ?
7) Comment les congés sont-ils calculés
8) L'application web est-elle accessible publiquement ou via VPN ?
9) Y a-t-il besoin d'un service de notification ?

II. Définir les objectifs et le mérimètre

1. Objectifs SMART(E)

Rappel de la méthode SMART :



Digitaliser les demandes de congés D'ici 4 mois, permettre aux 180 salariés de consulter leur solde et de déposer une demande de congé en ligne depuis un ordinateur ou un smartphone, afin de supprimer la gestion par formulaires PDF et e-mails.

Réduire le délai de traitement Dans les 3 mois suivant le déploiement, réduire d'au moins 50 % le temps moyen de traitement d'une demande de congé grâce à un workflow de validation numérique.

Centraliser le suivi des congés À la mise en production, permettre aux salariés, managers et RH de consulter un historique fiable et centralisé des demandes avec leur statut : en attente, acceptée, refusée ou annulée.

Sécuriser l'accès aux données Dès la première version, garantir que chaque utilisateur n'accède qu'aux informations correspondant à son rôle : salarié, manager ou RH, avec une authentification sécurisée.

Déployer une première version rapidement Mettre en production une V1 fonctionnelle dans un délai maximal de 4 mois, couvrant les besoins essentiels des trois sites et utilisable sur ordinateur et smartphone via navigateur web.

2. Périmètre du projet

a. Inclus dans le projet

- On parle que de la première version (sous 4 mois) :
- Authentification des utilisateurs.
- Gestion des rôles : salarié, manager, RH/administrateur.
- Consultation du solde de congés.
- Demande de congé par l'employer.
- Consultation des demandes en cours et passées.
- Validation des manager.
- Mise à jour dynamique du statut de la demande.
- Historique des congés.
- Notifications par e-mail.
- Interface web utilisable sur ordinateur et smartphone.
- Gestion des 3 sites de NovaServices.
- Administration basique des utilisateurs et des soldes par les RH.
- Sécurisation des données et gestion des droits d'accès.
- Gestion du télétravail

b. Exclu du projet

- Une application mobile native iOS/Android.
- La gestion complète de la paie.
- La gestion des absences (médicales et des arrêts maladie).
- La gestion des notes de frais.
- L'intégration avec de nombreux logiciels externes.
- Des fonctionnalités avancées d'intelligence artificielle.
- Des statistiques et tableaux de bord RH très poussés.
- La gestion de workflows extrêmement complexes avec plusieurs niveaux de validation.
- La gestion internationale ou multilingue si NovaServices n'en a pas besoin.

3. Proposer un MVP réaliste pour la première version.

Salarié → demande → Manager → validation → historique → RH

- Fonctionnalités du MVP
- Connexion sécurisée
- Gestion des identités
- Consultation du solde
- Dépôt d'une demande
- Validation/refus par le manager
- Notification par e-mail
- Historique des demandes
- Mise à jour du solde (dynamique)
- Interface ordinateur + smartphone
- Administration RH basique

L'idée est de ne pas chercher à reproduire tout le système RH dans la première version. Le MVP doit simplement remplacer efficacement le processus actuel basé sur PDF + e-mail.

4. Priorisation MoSCoW

a. Must have

Indispensable Fonctionnalités nécessaires au fonctionnement de la V1 :

- Authentification sécurisée
- Gestion des rôles et des droits
- Consultation du solde de congés
- Création d'une demande
- Contrôle de la disponibilité du solde
- Validation/refus par le manager
- Notification des utilisateurs
- Historique des demandes
- Mise à jour des soldes
- Interface responsive ordinateur/smartphone
- Administration RH de base

b. Should have

Important Fonctionnalités utiles mais pouvant être ajustées si le délai devient contraignant :

- Commentaire du manager lors d'un refus.
- Annulation d'une demande par le salarié.
- Rappels aux managers pour les demandes en attente.
- Recherche et filtrage de l'historique.
- Export des données par les RH.
- Calendrier permettant au manager de visualiser les absences de son équipe.

c. Could have

Souhaitable À envisager après le MVP :

- Tableau de bord RH avec statistiques.
- Graphiques sur les congés.
- Notifications push.
- Personnalisation avancée de l'interface.
- Import/export automatisé avec le logiciel RH existant.
- Gestion de délégation de validation.
- Calendrier partagé plus avancé.

d. Won't have

Pas dans la V1 Application mobile native :

- Gestion de la paie.
- Gestion complète des arrêts maladie.
- Gestion des notes de frais.
- Intelligence artificielle.
- Planification avancée des effectifs.
- Intégrations multiples avec des systèmes externes.
- Fonctionnalités internationales ou multilingues non nécessaires.

II. Définir les objectifs et le périmètre

1. Rédiger 3 à 5 objectifs SMARTE.

Spécifique	
Mesurable	
Atteignable	
Réaliste	
Temporellement défini	4 mois suffiront pour sortir une version MVP
Ethique	

III. Formaliser les besoins

1. Rédiger au minimum 8 user stories.

1. En tant qu'employé, je veux poser un congé dans un mois, avec le bon nombre de congés disponibles
2. En tant qu'employé, je veux poser un congé pour demain, avec le bon nombre de congés disponibles
3. En tant qu'employé, je veux poser un congé avec un nombre de congés insuffisant
4. En tant que chef, je veux accepter un congé d'un employé
5. En tant que chef, je veux refuser un congé d'un employé
6. En tant que chef, je veux imposer un congé à un employé
7. En tant que chef, je veux poser des congés pour un employé que je ne gère pas, en remplacement d'un autre chef lui-même absent
8. En tant que RH, je veux consulter les congés restants et les reporter sur le gestionnaire de paie

2. Ajouter des critères d'acceptation à au moins 4 user stories.

1. En tant qu'employé, je veux poser un congé avec un nombre de congés insuffisant
    2. Suis-je connecté avec le bon profil ?
    3. Ai-je accès au solde de congés cohérent (affiché sur le dernier bulletin de paie) ?
    4. Puis-je poser un nombre de jours libres ?
    5. Puis-je prendre des congés anticipés et avoir un solde négatif ?
    6. Les jours sont-ils bien calculés (tenant compte des vacancelles, des jours fériés) ?
7. En tant que chef, je veux accepter un congé d'un employé
    8. Est-ce bien mon employé ?
    9. Ai-je bien accès à ses demandes ?

10. Puis-je bien accepter ?
11. En tant que chef, je veux imposer un congé à un employé
    12. Ai-je un rappel des obligations légales d'un congé imposé ?
    13. Dois-je avoir une approbation d'une RH (admin) pour le faire ?
    14. Puis-je imposer des congés anticipés ?
    15. Puis-je imposer une durée libre ?
16. En tant que chef, je veux poser des congés pour un employé que je ne gère pas, en remplacement d'un autre chef lui-même absent
    17. Ai-je besoin d'un lien avec ce chef absent ?
    18. Disposés-je de droits de chef temporaires ?
    19. Qui les donne ?
    20. Puis-je accepter des congés comme si j'étais son chef ?
21. En tant que RH, je veux consulter les congés restants et les reporter sur le gestionnaire de paie
    22. Puis-je les consulter pour n'importe qui ?
    23. Puis-je faire une extraction ?
    24. Puis-je faire une recherche avec filtres ?

3. Identifier les principales contraintes techniques, de sécurité et d'utilisation.

Les contraintes relevées sont :

- Techniques :
  - Doit être résilient pour toute l'entreprise (180 employés). Tous les employés se connectent rarement au même moment, mais prévoir 200 connexions simultanées suffira, y compris en cas de croissance
- Sécuritaires :
  - Exigences RGDP
  - Les échanges doivent être chiffrés dans le navigateur pour éviter les interceptions et les falsifications
  - Les connexions doivent être réalisées avec des standards de sécurité empêchant les connexions pirates (jetons temporaires, énigmes serveur-client, pages dynamiques, etc)
- Utilitaires :
  - Doit être utilisable sur bureau, ainsi que sur téléphone (iOs et Android). Le point renvoie à une contrainte technique : faire une application au moteur partageable sur 3 systèmes d'exploitation et 2 architectures différentes : une application à moteur résilient et embarqué type Java, Electron ou Tauri est à prévoir.

IV. Organiser et estimer le travail

1. Découper le projet en grandes activités puis en tâches.

2. Définir la pile technique
    3. Quel moteur backend pour la logique métier / les opérations
    4. Quel stockage des données (DB, fichiers froids type JSON ?)
    5. Déploiement dans le nuage ou sur site ?
    6. Frontend sur serveur (application qui est une simple URL) ou dans le client (application complète qui récupère les données backend) ?

7. Développer un backend fonctionnel et testable
    8. Utiliser un moteur résilient, léger, puissant et maintenable (Node.js, Django).
    9. Privilégier un déploiement scalable (par image Docker, k8s compatible)
    10. Gérer le dépôt sur une forge mondiale (Gitlab ou Github) soit auto-hébergée sur une instance répliquée et sauvegardée (sur un VPS privé multi-régions)

11. Développer un frontend fonctionnel et responsive

    a) Application dédiée (.exe, .dmg, .apk, etc) [Avantages : serveur déchargé]

    11. Utiliser un front simple (JavaScript, Python, Go) compilable par Electron (hyper compatibilité, mais logiciel lourd car embarquant tout Chromium) ou Tauri (demande un paramétrage en Rust, un compilateur dédié par OS, mais application hyper légère en sortie)
    12. Gestion de pages dynamiques, affichant les résultats du backend dans des champs prédéfinis
    13. Gérer les tunnels de connexion backend <-> frontend de façon sécurisée. Possiblement uniquement par VPN pour simuler un serveur local avec authentification par jetons locaux

    b) Frontend sur le serveur [Avantages : aucun besoin d'application, juste une URL]

    A. Utiliser un front simple (JavaScript, Python, Go) compatible avec les moteurs de rendu WebKit, Chromium et Gecko
    B. Connexion directe dans réseau Docker unifié

C. Connecter frontend et backend
    D. Déployer un faux jeu de données d'employés sur une instance de préproduction
    E. Connecter le frontend à la base de données préprod
    F. Vérifier que les champs sont bien présents, qu'aucune erreur ne s'affiche
    G. Sécuriser les données en transit avec un système de jetons sévères et infalsifiables

H. Faire tourner les instances depuis des images Docker à utilisateurs non-root
I. Moindre privilège par défaut

J. Tester l'infrastructure
K. Test d'intégrité du code (Linter)
L. Test SAST et DAST
M. Test de charge
N. Test de connexions basiques frontend-backend
O. Test unitaires sur des fonctions clefs
P. Test de Bout-en-Bout pour chaque Histoire d'Utilisateur définie

2. Estimer la charge ou la complexité des principales tâches.

A. Définir la pile technique

- Discuter d'une architecture technique (facile)
- Décider d'un déploiement sur site (matériel à acheter, paramétrer, et connecter au réseau) ou d'un déploiement nuagique (coûts réguliers, demande d'un expert, prestation externe) (complexe)

B. Développer un backend fonctionnel et testable
- Prévision d'une maintenance à 6 mois, 2 ans, 10 ans (moyenne)
- Composer avec des outils/librairies résilientes, fiables, possiblement éditées par de grands acteurs avec un minimum de dépendances (complexe)
- Code hébergé dans un endroit clair, accessible, transmissible et bien commenté (facile)

C. Développer un frontend fonctionnel et responsive
- Trancher avec le client sur une application ou une URL sur serveur (complexe)
- Définir la tech du frontend (facile)

D. Connecter frontend et backend
- Définir la sécurité frontend <-> backend (moyenne)
- Tester les connexions (facile)
- Evaluer les risques de piratage avec un audit Pentest (complexe et coûteux)

E. Tester l'infrastructure
- Définir les test (moyenne)

3. Identifier les dépendances et les tâches parallélisables.

Les dépendances sur un projet technique vont surtout reposer côté client, avec les ambiguïtés :

- Déploiement sur place ou dans le nuage ?
- Application à installer ou page internet ?
- Quelle maintenance à prévoir, et par qui ?
- Quel budget ?

Des questions simples dont les réponses peuvent conduire jusqu'à un refus du contrat selon la proposition.

Concernant les tâches parallélisables, encore une fois, tout dépend d'un facteur non calculable qui est le rapport entre équipiers devs.

Techniquement, Frontend et Backend peuvent se faire ensemble, par les mêmes personnes (plutôt qu'un back et un front, on peut avoir deux profils fullstack) qui se séparent bien les tâches définies dans les Epiques du projet.

4. Proposer une répartition des responsabilités dans l'équipe.

- Lead Dev :
  - Définir la pile technique, défendre les choix applicatifs et de déploiement
  - Liaison PM <-> Client sur la partie technique pure en cas de besoin
  - Gérer la partie déploiement (InfraOps)
  - Vérifier les MR, faire de la revue de code
- Dev 1 (plus senior) :
  - Coder l'applicatif avec portée backend
  - Rôle DevOps pour aider au déploiement (gérer les tests backend, image Docker)
  - Faire les MR et participer aux revues de code
- Dev 2 (plus junior) :
  - Coder l'applicatif avec portée frontend
  - Faire les MR et participer aux revues de code

V. Choisir la méthode de gestion

1. Comparer Cycle en V et AGILE pour ce projet.

Critère	Cycle V	Agile (Scrum / Kanban)
Principe	Tout est séquencer en phase : spécification du besoin, conception, développement puis test. Chaque conception à sa phase de test	Développement par itération de courte durée, avec la livraison d'une fonctionnalité utilisable régulière

Besoins	Doit être stables avec un cadrage clair et complet dès le départ	Peux évoluer en cours et/ou se préciser durant le projet
Retour Utilisateur	Durant la recette a la fin du projet	A chaque livraison d'une fonctionnalité
Gestion du changement	A chaque changement il faut revoir tout le cycle du projet	Le besoin est réévalué en permanence, donc les changements sont intégrés à la méthode
Risque principal	Découvrir un écart entre ce qui à été produit et le besoin en fin de projet	Etre hors périmètre en cas de cadrage faible
Visibilité	Bonne sur la documentation du projet mais faible sur le produit car retour utilisateur en fin de processus	Livraison régulière donc le client a une visibilité claire sur l'avancement du projet
Documentation	Très complète	Plus légère
Adapté si	Périmètre non évolutif	Besoin à éclaircir, livraison rapide

2. Choisir l'approche que vous estimez la plus adaptée et justifier votre choix.

La méthode retenue est la méthode **SCRUM** avec jalons de vérification fixes pour respecter le délai de 4 mois.

Le choix a été fait pour plusieurs raisons :

- Besoin partiellement connu : la demande laisse de nombreuses zones de flou notamment les règles de congés, les types d'absences, le circuit de validation et autre. Un cycle en V aurait obligé à cadrer toutes ces inconnues dès le départ.
- Le livrable est orienté utilisateur : Les RH, employés et manager doivent valider l'ergonomie de l'outil (facilité d'utilisation, accessible ordinateur + téléphone). Des démos régulières permettent des corrections rapides.
- V1 livrée en 4 mois : le découpage en sprint et le MVP (consulter son solde, déposer une demande,…) permet de prioriser les plus « utiles » et si le temps venait à manquer, « sacrifier » les MoSCoW « could » et non la qualité globale de l'outil.

- Risque métier maîtrisé : l'application est interne, la sécurité est traitée via des critères d'acceptation ainsi que des test dédiés plutôt qu'un cycle lourd.
- Pourquoi pas Kanban ? : il convient pour des « petites » demandes régulières, là nous avons une échéance et un MVP à livrer, le rythme des sprints est donc plus adapté.

3. Décrire l'organisation pratique retenue : rythme, réunions, validations et suivi.

Échéance de 4 mois soit environ 8 sprints de 2 semaines chacun

Période	Contenu
Sprint 0 (S1 – S2)	Cadrage, architecture…
Sprint 1 à 5 (S3 – S12)	DEV des fonctionnalités par ordre de priorité calqué sur MoSCrow
Sprint 6 (S13 – S14)	Finalisation, tests sécurité et performance
Sprint 7 (S15 – S16)	Livraison en recette utilisateur, corrections et préparation de la mise en PROD

Rôles :

- Product Owner : Responsable RH, amène le besoin, priorise les idées, valide les livrables
- Scrum Master : chef de projet, il permet de faciliter les réunions et de lever les points de blocages
- Équipe de développement : développeurs, testeur, référent sécurité/infra

Organisation des réunions :

Réunion	Fréquence	Durée	Objectif
Sprint Planning	Début de chaque sprint	2h	Définir les user stories du sprint
Daily	C'est dans le nom	15min	Avancement et blocage de chacun
Sprint Review / Demo	Fin de sprint	1h	Présenter le livrable au RH et au manager pilote
Rétrospective	Fin de sprint	45min	Débrief sur le fonctionnement SCRUM et définir axe d'amélioration

Comité de pilotage	mensuel	1h	Point avec la direction pour discuter des risques, budget et jalons

Validations

- Critère de validation : code review effectuées, tests passés, critère d'acceptation validé, sécurité tester
- Validation par sprint : Product owner accepte ou refuse chaque user story de la review
- Jalons à valider côté direction : fin du cadrage, mvp fonctionnel, fin de recette, accord de mise en PROD

Suivi

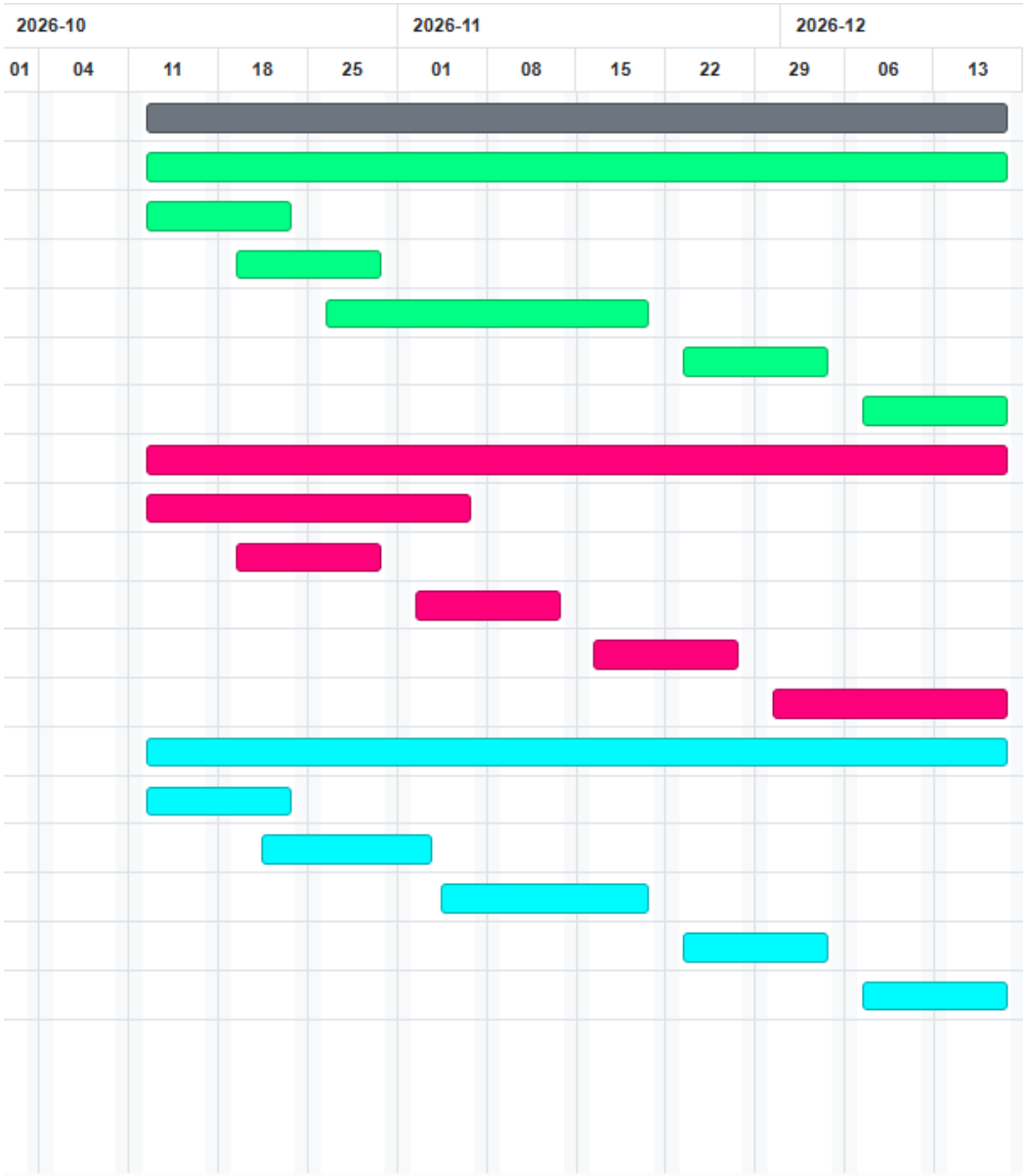
- Un **tableau de bord** de suivi : backlog, burndown du sprint, vélocité.
- Un **registre des risques** mis à jour à chaque sprint, avec les mesures d'atténuation (lien avec le plan de gestion des risques du livrable).
- Un **compte rendu**
- Après chaque review et chaque comité de pilotage.
- Des outils possibles : Jira, Azure DevOps ou Trello pour le backlog, et Teams pour la communication.

VI- Construire un planning

Légende :

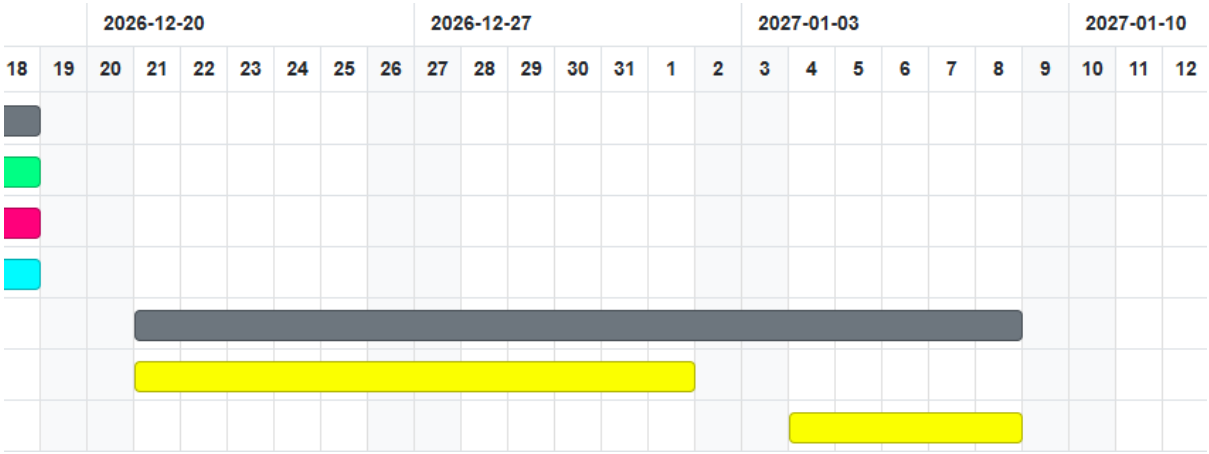
Lead Dev - Developpeur 1 - Developpeur 2 - RH

ID	Task Name	Start	End	Duration
1	API web fonctionnel	2026-10-12	2026-12-18	50 days
5	Lead Dev	2026-10-12	2026-12-18	50 days
6	Architecture globale de l'application	2026-10-12	2026-10-23	10 days
9	Authentication, sécurité et gestion des rôles	2026-10-19	2026-10-30	10 days
10	Backend	2026-10-26	2026-11-20	20 days
11	BDD et règle métier	2026-11-23	2026-12-04	10 days
12	Tests d'intégration, revue de code et mise en production	2026-12-07	2026-12-18	10 days
7	Dev 1	2026-10-12	2026-12-18	50 days
13	Interface utilisateur et responsive mobile	2026-10-12	2026-11-06	20 days
14	Page de connexion et tableau de bord salarié	2026-10-19	2026-10-30	10 days
15	Formulaire de demande de congés	2026-11-02	2026-11-13	10 days
16	Calendrier, soldes et historique	2026-11-16	2026-11-27	10 days
17	Tests d'interface et corrections	2026-11-30	2026-12-18	15 days
8	Dev 2	2026-10-12	2026-12-18	50 days
18	API et gestion des utilisateurs	2026-10-12	2026-10-23	10 days
19	Gestion des congés et calcul des soldes	2026-10-21	2026-11-03	10 days
20	Workflow de validation par le manager	2026-11-04	2026-11-20	13 days
21	Espace RH et historique des demandes	2026-11-23	2026-12-04	10 days
22	Tests backend et gestion des données	2026-12-07	2026-12-18	10 days


2026-10
2026-11
2026-12
01
04
11
18
25
01
08
15
22
29
06
13

GABRIEL Adam
BOICHÉ Gauvain
LEULIETTE Louis

1	API web fonctionnel	2026-10-12	2026-12-18	50 days
5	Lead Dev	2026-10-12	2026-12-18	50 days
7	Dev 1	2026-10-12	2026-12-18	50 days
8	Dev 2	2026-10-12	2026-12-18	50 days
23	RH	2026-12-21	2027-01-08	15 days
24	Test de l'application et amélioration	2026-12-21	2027-01-01	10 days
25	Resultat final V1	2027-01-04	2027-01-08	5 days



Projet : API web fonctionnel	Lead Dev	Dev 1	Dev 2	RH
Architecture globale de l'application	R	C	C	I
Authentication, sécurité et gestion des rôles	R	A	A	I
Backend	R	C	A	I
BDD et règle métier	R	A	A	I
Tests d'intégration, revue de code et mise en production	R	A	A	I
Interface utilisateur et responsive mobile	A	R	C	I
Page de connexion et tableau de bord salarié	A	R	C	I
Formulaire de demande de congés	A	R	C	I
Calendrier, soldes et historique	A	R	C	I
Tests d'interface et corrections	A	R	C	I
API et gestion des utilisateurs	A	C	R	I
Gestion des congés et calcul des soldes	A	C	R	I
Workflow de validation par le manager	A	C	R	I
Espace RH et historique des demandes	A	C	R	I
Tests backend et gestion des données	A	C	R	I
Test de l'application et amélioration	A	I	I	R
Résultat final V1	A	I	I	R

R - Reponsable | A - Approbateur | C - Consultant | I - Informer