TP2 Projet #2 Équipe #3

Étienne Latendresse-Tremblay LATE16039706

Juan Carlos Merida Cortes MERJ69080104

Christian Barikhan BARC84090100

Nazim Lounas LOUN74070208

Thomas Kasic KAST88050000

A-(4 pts) Charte de projet (2 pages) (niveau Stratégique)

1. (0,5 pt) Un énoncé de votre **Mission** [de votre partie du plus gros projet]. Cette mission doit être porteuse et inspirante, tout en décrivant pourquoi l'équipe existe.
   Questions suggérées pour aider à définir sa mission (tiré de Proulx 2020[1]) :
   - Pourquoi cette équipe a-t-elle été créée initialement?
   - Notre équipe a été mise en place afin de concevoir et développer certaines composantes du système essentielles à la réalisation du projet principal. Le projet a été séparé en quatre équipes afin de pouvoir limiter l'étendu des responsabilités et s'assurer que chaque équipe perfectionne leur composante attitrée.
   - Quelle est la contribution unique de cette équipe pour l'organisation ou dans le cadre du projet?
   - Notre équipe est responsable du backend, notamment du serveur HTTP, de l'API et de la base de données. Il servira aussi de médium entre le frontend et l'équipe s'occupant de l'intelligence artificielle. Dans le cadre de nos responsabilités nous devons aussi fortement documenter les points d'accès de notre API afin de

[1] Proulx, Martin (2020). Définir le « carré de sable » (ou charte d'équipe) des équipes auto-organisées, https://www.youtube.com/watch?v=Ewb9g_tfnms&t=33s .

permettre l'utilisation claire de nos ressources par les autres équipes.

- Comment l'équipe contribue au succès de l'organisation ou du projet?
- Nous nous assurons d'avoir une bonne communication avec toutes les autres équipes de développement. Cela implique de créer le produit selon les fonctionnalités établies et de l'adapter selon les changements nécessaires. Nous allons de l'avant afin d'organiser des rencontres et de faire des communications quotidiennes non seulement au niveau de l'équipe, mais aussi au niveau du projet au grand complet.
- Qu'est-ce que l'organisation risque de perdre si l'équipe cesse d'exister?
- Cela causerait une perte importante au projet. Non seulement la perte de notre équipe causerait une restructuration complète afin de combler les composantes que nous sommes responsables d'implémenter, mais l'organisation perdraient aussi notre expertise dans ces composantes qui ont fait que nous avons été administré ces composantes en premier lieu. Les décisions de l'organisation en rapport avec l'administration des tâches ont été faites en fonction des fonctionnalités que les équipes se sentaient plus à l'aise à implémenter. Chaque équipe est donc essentielle à la réalisation du projet.

- Comment l'équipe est-elle utile à ses clients/utilisateurs?
- Établir un backend bien structuré ainsi qu'une bonne gestion des données permet d'assurer aux utilisateurs et aux clients que ces données soient bien sécurisées et bien conservées. Il s'agit d'assurer que les renseignements personnels soient chiffrés et que les

accès soient limités. Il s'agit aussi de garantir qu'il n'y aura jamais de perte de données.

2. (0,2 pt) Le **client** de votre équipe (À qui l'équipe offre-t-elle ses services durant cette session?).

   Le client de notre équipe sera M. Bruel Gérançon, mais nos services seront également offerts au réseau de la santé.

3. (0,5 pt) 1 à 3 **objectifs** SMART (Spécifique, Mesurable, '*Actionable*'/Atteignable, Réaliste, Temporel [ou limité dans le Temps]). Questions suggérées pour aider à définir ses objectifs (adapté de Proulx 2020[1]) :

   - Quelles sont les 1 à 3 choses que l'équipe souhaiterait accomplir d'ici la fin de la session?

Nos objectifs sont de mettre en place un API complet et fonctionnel à 90%, une base de données sécurisée et performante puis de la documentation complète pour l'API.

   - Quels résultats l'équipe va-t-elle utiliser pour évaluer qu'elle a contribué au succès de l'organisation?

Pour l'API, on va considérer qu'une API fonctionnelle sera une API avec au moins 90% des routes qui seront implémentées et testées correctement. Pour la BD, on va considérer qu'une BD pouvant contenir les données nécessaires de manière sécuritaire ainsi qu'ayant un usage efficace sera considérée comme un succès. Un usage sécuritaire implique le chiffrement des données, des accès limités et la réplication de données contre les pertes. Pour la documentation, il est important de considérer que celle-ci doit être

complète et compréhensible. Si ces aspects sont maintenus et que le résultat est approuvé par les autres équipes, la documentation sera considérée comme un succès. Aussi, si on réussit à faire nos implémentations en respectant les délais de chaque sprint, on pourra considérer cela aussi comme un succès.

- Quels moyens l'équipe va-t-elle utiliser pour évaluer son progrès?

Afin de vérifier l'avancement du projet, il sera nécessaire de faire de faire un suivi du backlog et de s'assurer que les fonctionnalités planifiées soient mises en place dans les échéances prévues. Notre équipe fera aussi des rencontres hebdomadaires afin de répartir les tâches et obtenir plus d'informations sur l'avancement des tâches en cours. De même, des rencontres avec les autres équipes seront aussi tenues afin de voir la progression du projet et s'entendre sur les ajustements et les prochaines échéances.

Donc les 3 objectifs SMART qu'on veut accomplir avant la remise finale du projet sont:

1. Développer une API complète et fonctionnelle avec au moins 90% des routes implémentées et testées.
2. Concevoir une base de données sécurisée et performante, capable de stocker toutes les données nécessaires en garantissant une gestion efficace des accès et un chiffrement des informations sensibles.
3. Rédiger une documentation complète et approuvée par les autres équipes, de manière à ce qu'elle soit facile à comprendre et intégrable dans le projet principal.

4. (0,5 pt) La **portée** de votre projet, soit ce qui est inclus et ce qui ne l'est pas.

Inclus dans le projet:

1. Développement du backend
   1.1. Création d'un serveur HTTP pour gérer les requêtes et assurer une bonne communication entre le frontend, la base de données et l'IA.
   1.2. Implémentation d'une API REST avec authentification et gestion des permissions d'accès.
2. Base de données sécurisée et performante
   2.1. Conception des tables de la base de données relationnelle.
   2.2. Mise en place d'un chiffrement des données sensibles à l'entrée et à la sortie.
   2.3. Optimisation des performances avec des requêtes performantes.
3. Conception d'une documentation
   3.1. Rédaction d'une documentation détaillée des routes sous format OpenAPI.
   3.2. Explication de l'architecture du backend.
4. Tests
   4.1. Implémentation de tests unitaires et d'intégration

5. Sécurité et authentification
   5.1. Mise en place d'un système de gestion des utilisateurs et leurs permissions.
   5.2. Implémentation de JWT pour l'authentification.

Hors portée:

1. Développement du frontend.
2. Développement de la section IA.

5. (0,3 pt) Une **matrice des leviers** dûment remplie sur ces 2 axes :
   - Leviers : Budget – Échéancier – Portée

- Flexibilité : Souple – Moyen – Rigide

Levier	Flexibilité	Justification
Échéancier	Rigide	Les délais de chaque sprint sont fixés par le cours et le projet doit être livré avant le 24 mars. On ne peut pas obtenir du temps supplémentaire (à moins que le PO nous en donne plus)
Portée	Moyen	Il sera possible de modifier certaines fonctionnalités ou en ajouter tant que le projet reste fonctionnel et que le résultat soit livrable. Cependant le temps restant peut affecter l'ajout et le retrait de certaines fonctionnalités.
Budget	Souple	Comme le coût est basé sur l'effort (heures travaillées/personne), il peut varier en fonction de l'évolution du projet et du nombre d'heures travaillées sur chaque tâche.

6. (0,7 pt) Un résumé des **coûts anticipés** (en $) : en génie logiciel, la grosse majorité des coûts anticipés ou réels sont extrapolés à partir de l'effort estimé ou de l'effort réel (en heures-personne [h-p]). À ces coûts, on pourrait ensuite ajouter des coûts de licence ou de services informatiques pour l'infrastructure, ou tout autre coûts jugés pertinents. Toutefois, dans ce cours, seuls les coûts découlant de l'effort seront comptabilisés car nous ne devrions pas avoir aucun autre coût. Le tarif pour ce trimestre sera de 120$/h.

   Considérant que nous avons 3 sprints, le 1er ayant comme durée 2 semaines et estimant que chaque développeur peut investir en moyenne 3h par jour à chaque semaine, on a pour

notre équipe de 5 développeurs, un coût estimé pour le 1er sprint de:

3h * 10 jours = 30h/sprint; 30h/sprint * 5 devs = 150h/sprint; 150h/sprint * 120$/h =18 000 $ pour le 1er sprint.

Considérant que le 3e sprint dure le même temps et estimant que les développeurs investissent la même quantité de temps que dans le 1er sprint, on a 18 000$ pour le 3e sprint.

Finalement, considérant qu'on a environ 3 semaines pour le 2e sprint et estimant que les développeurs investissent la même quantité de temps que dans le 1er et 3e sprint, on a:

3h * 15 jours = 45h/sprint; 45h/sprint * 5 devs = 225h/sprint;
225h/sprint * 120$/h =27 000$ pour le 2e sprint.

En résumé, pour l'ensemble des sprints, en tenant compte que de l'effort, le tarif pour ce trimestre serait d'environ 63 000$. Cette estimation peut varier en fonction d'imprévus ou bien d'une implémentation rapide de la part des développeurs mais nous allons essayer de s'y conformer autant que possible.

7. (0,3 pt) Les **critères de succès** (du point de vue du client, dans un contexte d'affaires plus large, pour répondre à la question « Comment allons-nous mesurer le succès de ce projet? »).

1. Développement de l'API Backend
    a. Au moins 90% des routes de l'API doivent être implémentées et testées.
    b. L'API doit être fonctionnelle, sécurisée et documentée.
2. Base de données

   a. Les données doivent être stockées de manière sécuritaire (hashage et encryption)
   b. Les requêtes doivent retourner rapidement le contenu demandé.
3. Tests
   a. Les fonctionnalités doivent (si possible) être couvertes/validées par des tests unitaires (et d'intégration).
   b. L'API doit passer toutes les revues de code et tests de validation.
4. Documentation
   a. La documentation doit être complète et validée par les autres équipes.
5. Respect des échéances
   a. Tous les livrables doivent être terminés dans les délais imposés pour chaque sprint.
   b. Aucun retard de notre part ne doit compromettre l'intégration des fonctionnalités des autres équipes au projet principal.
6. Communication avec les autres équipes
   a. Avoir une communication efficace et fluide avec les autres équipes à travers des rencontres hebdomadaires et en communiquant par messagerie.
7. Satisfaction du client (M. Bruel Gérançon)
   a. Si le client est satisfait avec le produit final lors de la présentation de celui-ci.
8. (1,0 pt) Une liste des **risques** avec action de mitigation (M) et/ou de contingence (C) pour chacun des risques identifiés.
   Note : idéalement, vous mettez un plan de mitigation (ce qu'on fait maintenant ou bientôt pour amoindrir l'impact si jamais le risque se produit) et un plan de contingence (ce que l'on enclenchera au moment où le risque se produirait).

1. Manque de communication entre les équipes
   - M: Établir des rencontres hebdomadaires fixes

   - C: Mettre en place un canal de communication d'urgence
2. Retards dans les livrables
   - M: Suivre l'avancement quotidien du projet
   - C: Réorganiser les priorités et mobiliser plus de ressources
3. Problèmes techniques imprévus
   - M: Faire des tests réguliers et des revues de code
   - C: Avoir des solutions alternatives prêtes
4. Perte de données
   - M: Sauvegardes régulières et versionnage du code
   - C: Protocole de restauration des données
5. Conflits d'horaire
   - M: Planifier les rencontres à l'avance
   - C: Organiser des sessions de rattrapage

B- (8 pts) Charte d'équipe (2 à 5 pages) (niveau Tactique)

1. (0,5 pt) **Nom de l'équipe** et sa **liste des membres** avec une référence au nom du projet (ex. ce peut être en lien avec module et le sous-ensemble des fonctionnalités choisies).

   Medical Backend DevTeam :

   Étienne Latendresse-Tremblay

   Juan Carlos Merida Cortes

   Christian Barikhan

   Nazim Lounas

   Thomas Kasic

2. (0,5 pt) Les différentes **contraintes**, de façon à comprendre le degré d'autonomie de l'équipe à rencontrer ses objectifs. Questions suggérées pour aider à définir ses objectifs (adapté de Proulx 2020[1]) :

- Quelles sont les limites imposées à l'équipe, sur lesquelles l'équipe n'a pas de contrôle et à l'intérieur desquelles elle doit opérer?
  - Délais fixes pour les sprints
  - Budget limité aux heures-personnes
  - Dépendances avec les autres équipes
  - Technologies imposées par le projet principal
  - Respect des normes de sécurité médicales

- Quelles responsabilités sont déléguées à l'équipe et quelles sont celles qui demeurent la responsabilité du/de la gestionnaire?

  Responsabilités de l'équipe :
  - Développement du backend
  - Tests et assurance qualité
  - Documentation technique
  - Communication avec les autres équipes

  Responsabilités du gestionnaire :
  - Organiser les réunions
  - Faire un suivi du progrès de l'équipe

3. (1,0 pt) Liste des **rôles** (titre ou fonction) avec leurs **responsabilités** et affectation. Il doit y avoir un coordonnateur, au moins un développeur, au moins une personne responsable de l'assurance qualité (qui s'assure du respect de la définition de « terminé » et de l'application du processus), au moins une personne responsable de la gestion de configuration (gardien de l'intégrité des artéfacts de votre projet), au moins une personne en tant que « Point de contact technique » pour communiquer avec les autres équipes dépendantes et résoudre tout éventuel problème découlant de

cette dépendance. Vous devez aussi indiquer n'importe quel autre rôle que l'équipe juge utile et nécessaire en précisant les responsabilités, ce qui pourrait inclure le « PO » ou tout autre rôle qui doit interagir avec l'équipe ou au sein de celle-ci.

Coordonnateur: Juan Carlos Mérida Cortés

- Facilitation de la communication entre les membres de l'équipe
- Organisation des réunions
- S'assure du respect des procédures et des méthodologies

Développeurs: Christian Barikhan, Thomas Kasic, Étienne Latendresse-Tremblay, Nazim Lounas, Juan Carlos Merida Cortes

- Analyse et implémentation des fonctionnalités demandées
- Gestion des revues de code et amélioration du qualité du code
- Rédaction des tests unitaires et correction des bogues

Assurance qualité: Christian Barikhan

- Détermination et application des critères de définition de « terminé » pour chaque tâche
- Conception, exécution, et automatisation potentielle d'autres tests
- Collaboration avec les développeurs pour l'identification et la correction des anomalies

Administrateur système: Étienne Latendresse-Tremblay

- Mise en place du serveur hébergé localement
- Assurer les transmissions des voix d'accès publiquement sur le réseau
- Garantir la réplication des données et la disponibilité du système en tout temps

Responsable de la gestion de configuration: Nazim Lounas

- Gestion des outils de versionnement et de CI/CD

- Assurance de la cohérence entre les livrables et les dépendances du projet
- Assurance de l'intégrité des artéfacts

Personne point de contact technique: Thomas Kasic

- Fait le lien avec les équipes techniques externes
- Résolution des problèmes de dépendances entre équipes et systèmes
- Documentation des exigences techniques pour les autres équipes
- Anticipation des risques liés aux interactions avec d'autres composants logiciels

PO: Bruel Gérançon

Responsable sécurité: Juan Carlos Mérida Cortés

- Identification et correction des vulnérabilités dans le code et l'infrastructure
- Implémentation des pratiques de sécurité
- Assurance de la conformité aux normes de sécurité

Architecte logiciel: Étienne Latendresse-Tremblay

- Définition de l'architecture du projet en fonction des besoins techniques et métier
- Assurance de la cohérence et la scalabilité du système
- Choix des technologies et des frameworks

4. (4,0 pts) Les différentes **normes de fonctionnement de l'équipe**, soit la définition de la manière dont les membres d'équipe vont travailler ensemble :
    1. (1,5 pt) Le **processus de développement** choisi pour ce projet et la justification des activités. Ce peut être une cartographie, un diagramme d'activités. Ce processus doit être cohérent avec la définition de « terminé ».

       Nous adoptons une approche Scrum avec DevOps pour assurer un développement itératif et incrémental en garantissant des livrables de haute qualité. Ce processus

suit un cycle structuré commençant par la planification et l'analyse des exigences, où notre PO priorise les fonctionnalités en collaboration avec l'équipe. Ensuite, la conception et l'architecture sont validées par l'architecte logiciel.

Le développement sur git suit un respect des standards de code, avec des revues obligatoires avant les intégrations. Chaque fonctionnalité doit avoir une couverture d'au moins 80% de tests et respecter les critères de performance. La documentation et un README mis à jour sont obligatoires avant la validation.

2. (0,5 pt) Les différents **Outils** que l'équipe utilisera et leurs usages. Assurez-vous que tous y aient accès. L'extrant (« output ») de ces outils devra être transmissible au professeur dans un format facile à lire (fichier texte, Word, PowerPoint, Excel, PDF, JPEG, PNG, OpenOffice).
    - Google Sheets
    - Google Docs
    - Neovim
    - VS Code
    - PyCharm
    - Gitlab
    - Zrok
    - Docker Compose
    - PostgreSQL

3. (0,5 pt) La **logistique de coordination** de l'équipe. Par exemple : lieux, dates et heures de rencontres, ou mécanismes privilégiés de travail d'équipe (ex. labo, Zoom, etc.). Les moyens de communication de

l'information pertinente (p. ex. courriel, forum d'équipe, téléphone, Mattermost, etc.)

- Réunions hebdomadaires le lundi
- Communications urgentes via Discord
- Travail asynchrone avec commits réguliers

4. (0,5 pt) **Modes décisionnels principal et secondaire**. Les modes possibles sont : unanimité (nous sommes tous d'accord), consensus (une majorité est d'accord et les autres n'en sont pas dérangés), majorité (une majorité est d'accord mais une minorité sont contre), autocratie par le coordonnateur (il n'y a que lui/elle qui prend la décision et le reste de l'équipe accepte par défaut), mais vous pouvez en créer d'autres, à condition qu'ils soient expliqués clairement. Aussi, inclure comment les décisions seront prises.
   - Principal: Consensus (majorité d'accord, minorité non opposée)
   - Secondaire: Vote majoritaire en cas de blocage

5. (0,6 pt) Les **comportements attendus** au sein de l'équipe.

   Un des comportements primordial attendu chez tous les membres de l'équipe est la communication. Il est essentiel de pouvoir communiquer avec tous les membres dans des délais raisonnables afin d'obtenir un consensus pour toute décision importante. Il est aussi important que tous les membres de l'équipe soient ponctuels non seulement pour les rencontres, mais aussi pour les remises.

6. (0,4 pt) Les **comportements non tolérés**.

Les modifications de dernières minutes pouvant entraîner la défaillance du système ne seront pas tolérées. Il est important d'avoir un système fonctionnel et stable à chaque fin de sprint, donc les derniers jours devrait être consacrés à la refactorisation et à la mise en place de tests

5. (2,0 pts) **Résolution de conflits**. Si quelque chose ne se passe pas bien, comment doit réagir l'équipe? Vous pouvez vous inspirer de Wikipédia: Résolution de conflit. Vous devez prévoir au minimum le cas où un membre d'équipe ne livre pas ses engagements, car c'est l'équipe, et non la professeure, qui jugera de son sort et de sa partie de la note.

   Notre approche de résolution de conflits se base sur la communication et la transparence. Pour un membre non performant, nous commençons par une discussion d'équipe suivie d'un plan d'amélioration avec des objectifs mesurables. Pour les conflits interpersonnels, le coordonnateur agit comme médiateur. Si le conflit persiste, une réunion d'équipe est organisée. Tous les incidents et leurs résolutions sont documentés pour assurer le suivi et identifier les problèmes récurrents.

C-(3 pts) Définition de « terminé » (1/2 page) (niveau Tactique)

La définition doit comprendre la liste de toutes les conditions requises [découlant des activités à réaliser] pour assurer la qualité des fonctionnalités et autres livrables, soit « qu'est-ce qui doit être vrai pour qu'on considère notre livrable comme étant « terminé », c'est-à-dire prêt à être déployé en production. Cette définition de « terminé » doit tenir sur une demi-page (ou moins) tout en étant lisible et comprise de tous. Elle doit comprendre au moins 3 activités tactiques, dont la revue par les pairs.

Définition de "terminé" :

Une fonctionnalité est considérée comme terminée lorsqu'elle respecte nos critères rigoureux de qualité logicielle. Le code doit avoir passé une revue par au moins deux pairs, être couvert à plus de 80% par des tests unitaires et d'intégration réussis, et ne présenter aucun bogue critique ou vulnérabilité. Le respect des standards de code et une performance optimale avec des temps de réponse inférieurs à 200ms sont également requis.

La documentation fait partie intégrante de la définition du "terminé". Cela inclut une documentation API à jour au format OpenAPI, un guide d'utilisation complet, des commentaires pertinents dans le code et un README mis à jour reflétant les derniers changements. Cette documentation doit être claire, concise et validée par l'équipe.

La validation finale exige une démonstration fonctionnelle réussie, l'approbation du Point de Contact Technique et une intégration vérifiée avec les autres composants du système. L'acceptation du Product Owner est la dernière étape obligatoire avant qu'une fonctionnalité puisse être considérée comme terminée et déployable en production.

Il est attendu d'avoir 2 groupes de conditions [toujours découlant des activités d'AQ] :

1. (2 pts) AQ logicielle : soit les conditions menant à la qualité du logiciel;

- Revue de code par au moins un membre
- Tests unitaires couvrant 80%+ du code
- Tests d'intégration réussis
- Aucune vulnérabilité critique détectée
- Code respectant les standards définis

2. (1 pt) AQ de la documentation : soit les conditions menant à la qualité des différents livrables de type documentation,

- Documentation API complète (OpenAPI)
- README à jour avec instructions d'installation
- Documentation technique validée par l'équipe
- Commentaires pertinents dans le code

D-(5 pts) Liste des activités (1 à 2 pages, au besoin) (niveau Opérationnel)

Voir fichier excel : Projet2-Equipe3-Liste_Activites-Gabarit-H25