Rapport

Projet Android - info805 & info806

Samy Amarouche & Raphaëlle Huynh

0. Introduction

Ce projet fait partie d'un module universitaire où l'idée était de mélanger développement mobile et intelligence artificielle pour répondre à un besoin concret. Le but principal, c'était de créer une application Android capable de détecter automatiquement les chutes à vélo et d'envoyer une notification.

Pour atteindre cet objectif, le projet a été divisé en trois volets principaux :

1. **Développement de l'application Android** : Une application mobile permettant de collecter les données des capteurs (accéléromètre, gyroscope, GPS) et de les transmettre via le protocole MQTT.
2. **Détection des chutes** : Un système basé sur l'intelligence artificielle capable d'analyser les séries temporelles des données collectées pour identifier les anomalies correspondant à des chutes.
3. **Notification lors d'une chute** : Une notification doit être envoyer sur l 'appareil lorsque une chute est détecté.

Le projet a été réalisé par une équipe de deux personnes, avec une répartition claire des responsabilités entre le développement mobile et la détection d'anomalies. Ce rapport détaille les différentes étapes du projet, en commençant par la gestion de projet, suivie du développement de l'application, le captage des données, la structuration des données et l'analyse de celle-ci.

1. Gestion du Projet

1.1 Planification Initiale

Dès le début du projet, nous avons mis en place différentes méthodes de gestion afin de structurer notre travail. Nous avons repris les étapes vues en cours : étude du projet, définition de personas, création de maquettes, puis élaboration d'une liste de tâches à réaliser pour construire le diagramme de PERT, suivi du Gantt.

Initialement, le projet avait pour objectif principal de développer une application capable de détecter une chute à vélo et d'envoyer une notification. Par la suite, nous avons choisi d'ajouter d'autres fonctionnalités afin de rendre l'application plus complète et réellement utile pour des cyclistes. L'idée était d'en faire un véritable outil, tout en respectant la partie détection et notification demandée dans le sujet.

1.2 Personas et Analyse des Utilisateurs

Pour concevoir une application adaptée à un large éventail d'utilisateurs, nous avons identifié trois profils types :

1. **Personnes âgées** : Ces utilisateurs pourraient avoir l'application installée et configurée par leurs proches pour des raisons de sécurité. L'interface doit donc être simple et intuitive pour éviter toute confusion.

2. **Parents d'enfants cyclistes** : Les parents pourraient utiliser l'application pour surveiller la sécurité de leurs enfants. Ici encore, la simplicité et la clarté de l'interface sont essentielles.

3. **Utilisateurs avertis** : Ces utilisateurs, à l'aise avec la technologie, souhaitent utiliser l'application pour une utilisation classique, comme le suivi de leurs trajets ou la détection de chutes.

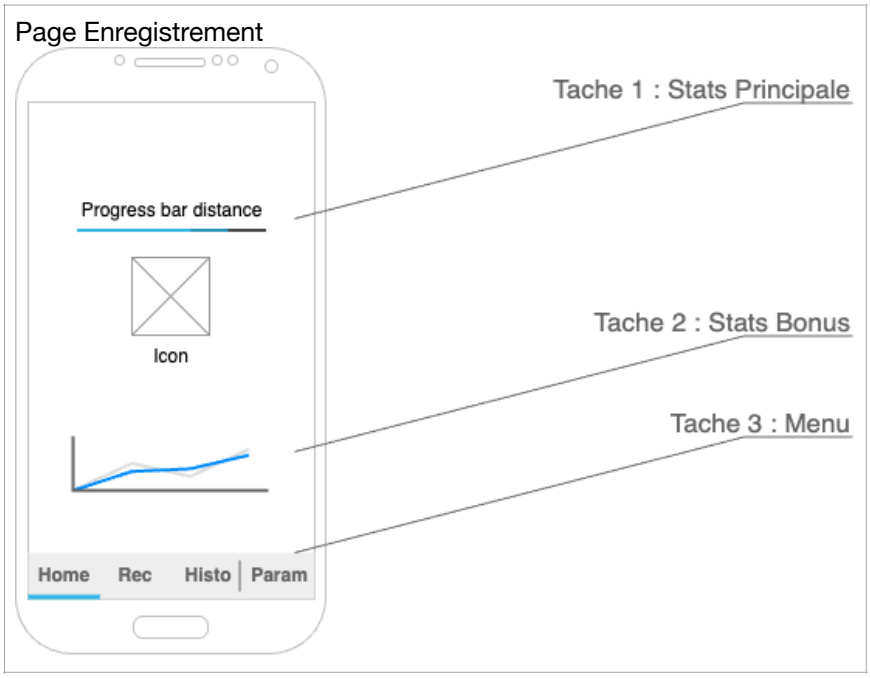
Pour répondre à ces besoins variés, nous avons conçu une interface simple et épurée. Bien que cela rende l'application visuellement « vide », mais à la fois pas trop pour que l'application puisse être utiliser naturellement par des

cycliste. L'application doit aussi pouvoir être claire et avoir un système de navigation très simple.

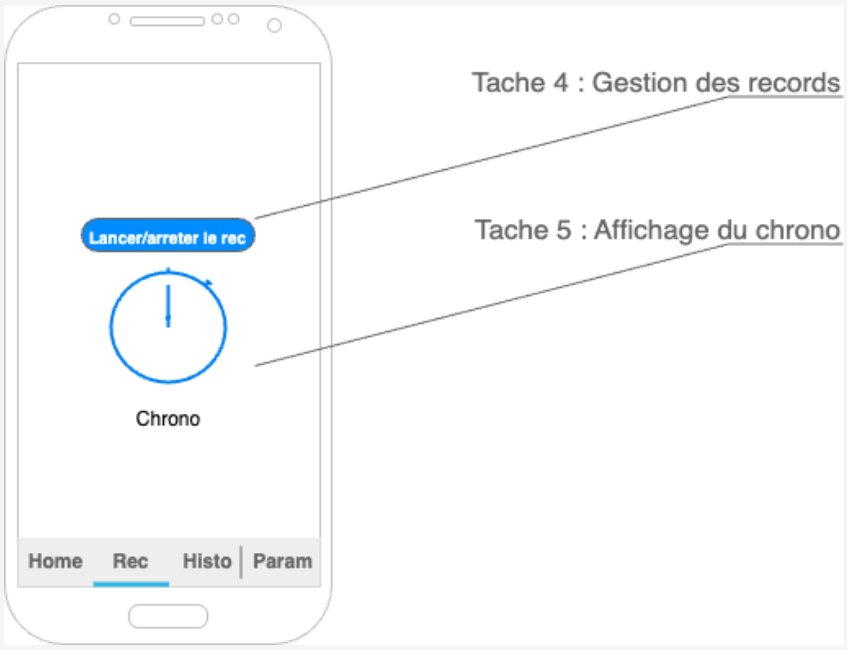
1.3 Réalisation de maquette pour l'application

À partir des conclusions tirées de l'analyse des personas, nous avons pu élaborer des maquettes, qui nous ont ensuite permis de définir les tâches à réaliser pour le projet, et ainsi de construire les diagrammes de PERT et de Gantt.

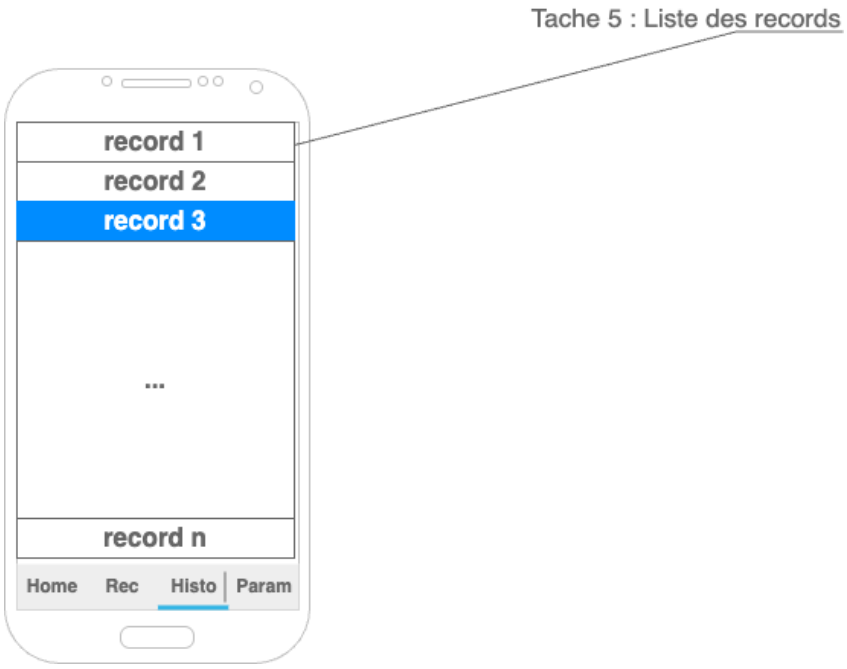
Nous avons identifier les pages suivante :

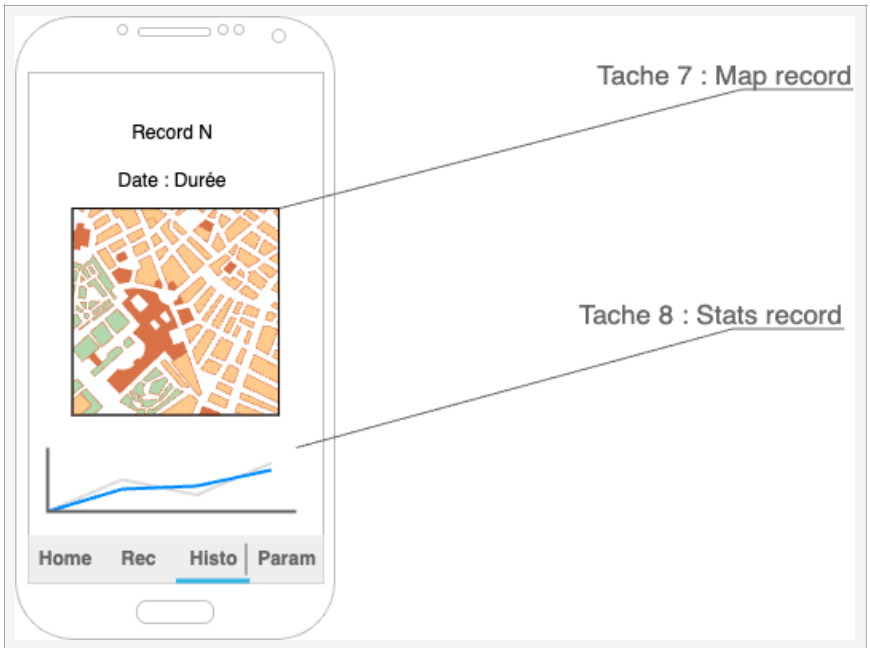


Page Enregistrement

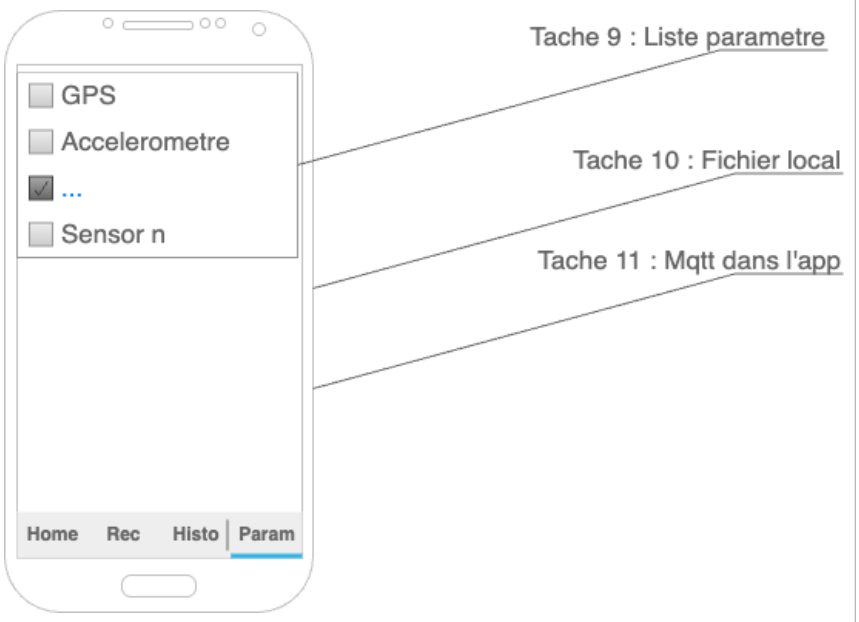


Page Historique





Page Details d'une session



Page Parametre

Pour chacune de ces pages, nous avons identifié des tâches liées au développement du projet. Il restait ensuite à en ajouter de nouvelles, notamment pour la partie détection d'anomalies.

Tache	Description	Necessite	Durée Estimé	Priorité
Tache A	Créer une barre de progression pour afficher une statistique principale, comme la distance, sur la page d'accueil.	C	2J	Normal
Tache B	Créer d'autres statistiques pour la page d'accueil.	C	2J	Bonus

Tache	Description	Necessite	Durée Estimé	Priorité
Tache C	Créer un menu en bas pour naviguer dans l'application.		3J	Absolu
Tache D	Gestion des enregistrements et envoi des données avec chrono. Enregistre une session de vélo.	J, K	5J	Absolu
Tache E	Affichage d'un chrono pendant l'enregistrement.	D	1J	Bonus
Tache F	Liste des enregistrements dans l'historique de l'application.	L	1J	Absolu
Tache G	Affichez une carte pour retracer le chemin parcouru lors d'un trajet enregistré dans l'historique.	F	2J	Bonus+
Tache H	Statistique supplémentaire dans l'historique.	F	2J	Bonus
Tache I	Paramètres de l'application	C	4J	Important
Tache J	Gestion des paramètres de fichiers locaux	I	2J	Important
Tache K	Gestion de l'envoi de données par MQTT	I	1J	Absolu
Tache L	Gestion des fichiers locaux contenant les données de l'historique.	D	2J	Important
Tache M	Récolte de données pour la détection de chute		1J	Absolu
Tache N	Développement d'un algorithme d'aide à la détection de chute à vélo.	M	7J	Absolu
Tache O	Développement d'un serveur pour la détection des chutes.	N, I	2J	Absolu
Tache P	Développement d'une deuxième application pour notifier en cas de détection de chute.	O	2J	Important

1.4 Pert et Gantt

Nous avons créé le diagramme de PERT pour ce projet en nous basant sur les tâches définies précédemment. Le chemin critique, mis en évidence en rouge, met en avant les tâches essentielles. Les tâches bonus, affichées en vert, sont rattachées à la fin du projet car elles peuvent être ajoutées jusqu'à la dernière minute avant le rendu final.

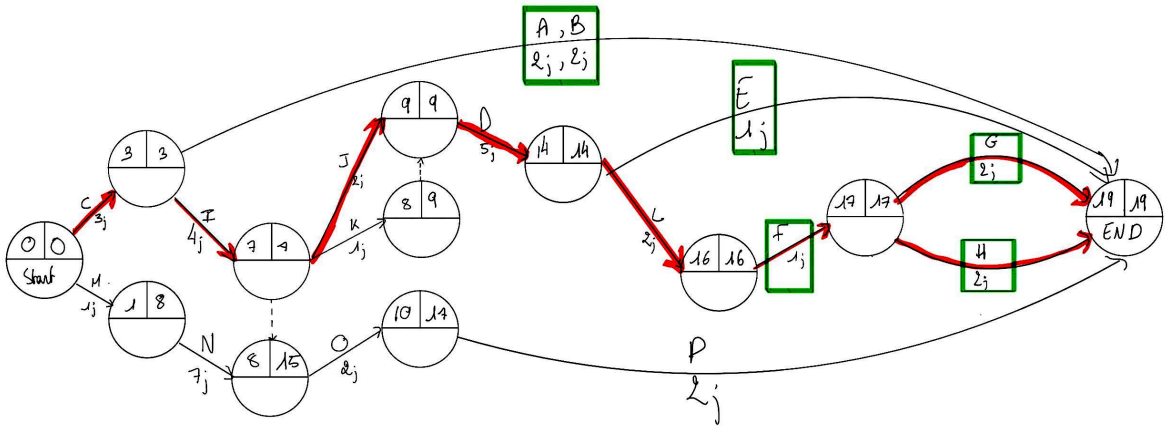


Figure 1 : Pert

Après avoir finalisé le diagramme de PERT, nous avons immédiatement élaboré le diagramme de Gantt en nous basant sur les informations précédemment définies. Nous avons fixé la date de fin du projet à la semaine des devoirs surveillés, soit celle du 21 avril, en prévoyant une petite marge pour les ajustements éventuels.

	Task Name	Duration	Start	ETA
1	Tache C	3	27/03	30/03
2	Tache I	4	30/03	03/04
3	Tache J	2	03/04	05/04
4	Tache K	1	03/04	04/04
5	Tache D	5	05/04	10/04
6	Tache L	2	10/04	12/04
7	Tache F	1	12/04	13/04
8	Tache G	2	13/04	15/04
9	Tache H	2	13/04	15/04
10	Tache E	1	10/04	11/04
11	Tache A	2	30/03	01/04
12	Tache B	2	30/03	01/04
13	Tache P	2	06/04	08/04
14	Tache O	2	04/04	06/04
15	Tache N	7	28/03	04/04
16	Tache M	2	27/03	28/03

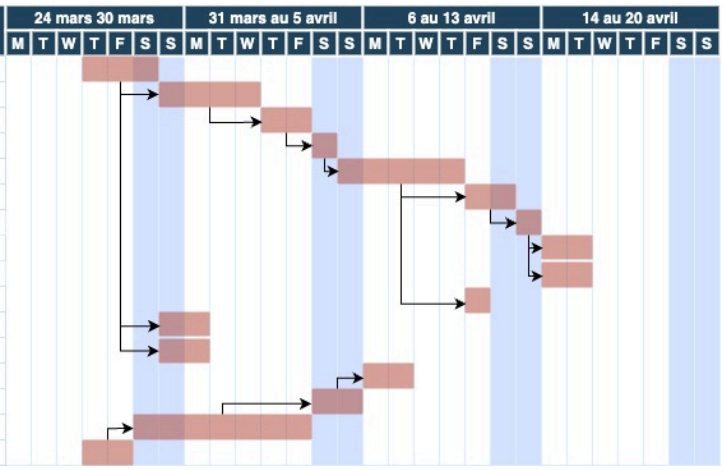


Figure 2 : Gantt

Nous avons choisi de nous baser sur les dates de fin de projet pour plusieurs raisons. Tout d'abord, nous avions plusieurs projets en cours simultanément. De plus, nous avons mal compris le sujet et, au lieu de nous engager dans la mauvaise direction, nous avons préféré attendre de pouvoir poser nos questions et obtenir des consignes claires afin de nous concentrer pleinement sur le projet.

1.5 Répartition des taches et Analyse des risques

Nous n'avions pas pensé a faire une matrice des risques au début du projet, voici une analyse rétrospective des principaux risques identifiés et des stratégies de mitigation mises en place :

Risque	Probabilité	Impact	Mitigation
Retard dans la collecte des données de chutes	Moyenne	Moyen	En attendant les données réelles, nous utilisons des données en ligne pour tester les algorithmes.
Complexité de l'intégration MQTT	Basse	Haut	Tests réguliers de la communication entre l'application et le serveur.
Interface utilisateur trop simpliste	Moyenne	Faible	Priorisation des fonctionnalités essentielles pour garantir une expérience utilisateur fonctionnelle
Retard global du projet	Moyenne	Moyen	Travail en parallèle sur l'application et la détection d'anomalies pour minimiser les dépendances

1.6 Workflow Git

Étant donné que nous étions seulement deux à travailler sur le projet et que nos tâches étaient bien distinctes, nous avons choisi un workflow Git simplifié. Nous avons principalement travaillé sur une seule branche, avec des commits explicites pour documenter les modifications. Bien que cette approche soit moins rigoureuse qu'un workflow basé sur des branches multiples (feature/*, develop, etc.), elle s'est avérée suffisante pour notre contexte.

1.7 Documentation et Communication

La documentation du projet, bien qu'encore en cours d'amélioration, comprend un fichier README présentant le projet et ses objectifs, ainsi que des commentaires dans le code pour expliquer les parties complexes. Pour la

communication, nous avons utilisé Discord pour les discussions, ce qui s'est avéré particulièrement utile pour synchroniser nos efforts, notamment lors des phases critiques du projet.

1.8 Suivi de l'avancement

Même avec les méthodologies définies au début, nous avons opté pour une méthode plus flexible au cours du projet. Notamment car il s'agit d'un projet à deux personnes sur un temps court. Cela implique plusieurs choses. Dès le départ, nous avons écarté la méthodologie Scrum pour ces raisons, car elle est trop exigeante en termes de temps par rapport au gain de productivité d'équipe qu'elle peut apporter. En revanche, le Gantt et le PERT ont été globalement suivis, avec des retards sur certaines tâches sous-évaluées. Après coup, nous avons réalisé qu'il manquait peut-être une étape : déterminer en équipe la durée d'une tâche avec des cartes. C'est une méthode que nous n'avons pas utilisée, car oubliée, mais qui n'aurait peut-être pas changé grand-chose pour une équipe de deux personnes.

Hormis cela, nous n'avions pas plus défini la gestion de notre projet au-delà du Gantt et du PERT. Nous avons appliqué une méthode proche d'une méthode en V, avec peu d'interaction avec la MOA (Monsieur Fouchal pour notre projet).

Lors de la dernière session avant le rendu, nous avons profité de l'occasion pour clarifier et modifier après coup des parties qui n'étaient pas claires dès le départ pour nous. Mais ces changements ont pu être faits à temps.

Le retard sur le projet ne concerne pas des tâches que nous avions priorisées « d'absolu ». Pour nous, la consigne est respectée, mais certaines tâches « importantes » qui nous tenaient à cœur ne sont pas (pour le moment) implémentés. C'est un peu dommage, mais c'est aussi lié aux modifications que nous avons dû faire après la dernière session et aux retards pris sur certaines tâches.

1.9 Gestion des Imprévus

Plusieurs aspects du projet ont pris du retard, comme mentionné. Nous avons tenu nos priorités, en particulier celles spécifiées dans le sujet. Par conséquent, l'affichage des cartes souhaitées et l'intégration des graphiques n'ont pas pu être mis en place.

Les cartes

Ce n'est pas par manque de temps que cette fonctionnalité n'a pas pu être implémentée. L'intégration d'une carte Google Maps dans une application nécessite une API payante de Google. Nous avons passé plusieurs jours à explorer des alternatives, mais aucune ne fonctionnait correctement, ce qui a retardé la tâche. Nous avons envisagé d'utiliser l'API Google, pensant que pour notre utilisation, cela ne nous coûterait que quelques centimes. Cependant, lors de l'abonnement à l'API Google, nous avons découvert que la carte utilisée ne fonctionnait pas avec le service, sans aucune information supplémentaire. Après plusieurs jours de travail sur ce problème, nous avons dû abandonner cette partie, ce qui est un peu dommage. Pour les statistiques, une version presque fonctionnelle existait, mais par manque de temps, nous avons dû la retirer de la version actuelle. Cependant, il est possible qu'elle soit incluse dans la version finale.

1.10 Conclusion sur la Gestion du Projet

En résumé, nous avons dévié des méthodologies enseignées en classe. Le temps nous a manqué, notamment pour certaines tâches du projet. Cependant, l'application des méthodes vues en cours a été plutôt réussie, à quelques oublis et exceptions près, notamment pour l'estimation de la durée des tâches.

2. Pipeline Application

2.1 Objectifs et Fonctionnalités Principales

L'application Android développée dans le cadre de ce projet a pour objectif principal de détecter les chutes à vélo et d'envoyer des alertes en cas d'accident. Pour cela, elle repose sur les fonctionnalités suivantes :

- **Envoi de données via MQTT** : Les données collectées par les capteurs du téléphone sont transmises au serveur toutes les secondes.
- **Détection de chutes** : Les données envoyées sont analysées par un serveur Python pour identifier les anomalies correspondant à des chutes.
- **Affichage et envoi d'alertes** : En cas de détection d'une chute, une notification est envoyée à l'utilisateur.

En complément, l'application vise à offrir des fonctionnalités supplémentaires pour une utilisation classique par les cyclistes :

- **Historique des trajets** : Accès aux trajets enregistrés, avec leur durée, leur tracé et les données collectées.
- **Personnalisation des données envoyées** : L'utilisateur peut choisir dynamiquement les capteurs dont les données seront transmises via MQTT.

2.2 Architecture de l'Application

L'application repose sur une architecture classique pour les applications Android, utilisant Java et XML pour le développement. Elle est principalement constituée de deux types de composants :

- **Activités** : Utilisées pour gérer l'affichage des différentes pages de l'application.
- **Services** : Fonctionnant en arrière-plan pour collecter les données des capteurs et les envoyer via MQTT.

Cette structure permet de séparer les responsabilités entre l'interface utilisateur et les processus de collecte et d'envoi des données, garantissant ainsi une meilleure modularité et une gestion efficace des ressources.

2.3 Développement Technique

Intégration des Capteurs

L'application intègre plusieurs capteurs pour collecter des données variées :

- **GPS** : Latitude, longitude, altitude.
- **Accéléromètre** : Mesure des accélérations sur les axes X, Y, Z.
- **Gyroscope** : Mesure des vitesses angulaires.
- **Magnétomètre** : Mesure des champs magnétiques.
- **Capteurs environnementaux** : Humidité, température, pression.

Ces données sont collectées en temps réel et transmises au serveur via MQTT à une fréquence d'une fois par seconde, lorsque l'enregistrement est lancé. De plus ces données, lorsqu'elles sont activées, sont également stockées sur l'appareil de l'utilisateur pour la gestion de l'historique.

Envoi Dynamique des Données

L'une des fonctionnalités les plus complexes à implémenter a été la personnalisation dynamique des données envoyées. Grâce à une interface utilisateur intuitive, l'utilisateur peut activer ou désactiver l'envoi des données de certains capteurs en temps réel à l'aide de sliders.

2.4 Interface Utilisateur

L'interface utilisateur de l'application est organisée autour de quatre menus principaux :

4. **Home** : Page d'accueil de l'application.
5. **Enregistrement** : Permet d'enregistrer les données collectées sur le téléphone et de les envoyer via MQTT.

6. **Historique** : Donne accès aux trajets enregistrés, avec des informations comme la durée, le tracé et les données collectées. Cette page, bien qu'importante, n'est pas encore complètement finalisée.

7. **Capteurs** : Affiche une liste des capteurs disponibles. En sélectionnant un capteur, l'utilisateur accède à une page dédiée affichant les données collectées par ce capteur (par exemple, les axes X, Y, Z pour l'accéléromètre ou la latitude et la longitude pour le GPS).

L'interface a été conçue pour être simple et épurée, afin de s'adapter à différents profils d'utilisateurs, comme décrit dans la section précédente.

2.5 Limites et Améliorations Futures

Bien que l'application atteigne ses objectifs principaux, certaines limitations subsistent :

- **Pages incomplètes :** La page de l'historique, qui devrait afficher des graphiques et des courbes des données collectées, n'est pas encore finalisée en raison de la complexité de son implémentation.

- **Personnalisation limitée :** Si la personnalisation des données envoyées est fonctionnelle, l'ajout d'autres options pourrait enrichir l'expérience utilisateur.

Pour l'avenir, plusieurs améliorations sont envisagées :

- **Amélioration des pages existantes :** Intégration de widgets et de graphiques sur les pages comme l'historique ou la page d'accueil.

- **Nouvelles fonctionnalités :** Intégration d'options supplémentaires pour les cyclistes, telles que des statistiques avancées et une carte permettant d'afficher les trajets déjà effectués.

3. Captage

3.1 Objectif

L'objectif principal de cette partie est de développer le matériel nécessaire à la collecte de données pour la détection des chutes à vélo. La collecte et l'envoi de données via MQTT ont été implémentés au début du projet, puis améliorés ultérieurement pour intégrer des fonctionnalités supplémentaires. Cela nous a permis de recueillir un nombre important de données, ce qui nous a permis d'utiliser ces données pour la détection des chutes tout en continuant à travailler sur le projet.

3.2 Presentation des données

Les données utilisées pour notre projet proviennent principalement de l'accéléromètre, mais aussi du GPS et du gyroscope. Nous utilisons des données appelées séries temporelles, qui varient au fil du temps. Notre objectif est d'analyser ces données à l'aide de graphiques, avec le temps en abscisse et la valeur du capteur en ordonnée.

Capteur	**Valeurs récupérer**
GPS	Longitude, Latitude
Accelerometre	X, Y, Z
Gyroscope	X, Y, Z
Magnétomètre (bonus)	X, Y, Z
Temperature (bonus)	°C
Humidité (bonus)	% d'humidité
Pression (bonus)	Pression

3.3 Envoie des données

Au début, les données envoyées par MQTT comprenaient toutes les données disponibles sur le téléphone. Cependant, lorsque nous avons décidé de réorganiser le code du projet, nous avons supprimé tous les capteurs et les

envois de données. Plus tard, après avoir repensé la structure de l'application, nous avons réintégré tous les capteurs. Désormais, les utilisateurs peuvent choisir quelles données envoyer, et l'envoi des données ne se produit que lorsqu'une session de vélo est en cours dans l'application.

Grâce à ces modifications, nous avons pu intégrer un système de collecte de données qui enregistre simultanément sur le téléphone et envoie les données via MQTT. Cette fonctionnalité est cruciale pour l'historique de l'application, car elle permet de collecter des données même dans les zones sans réseau ou lorsque le réseau est désactivé. Cependant, la détection de chute ne s'effectue pas localement, ce qui empêche l'envoi de notifications. Nous y reviendrons plus tard.

4. Structure des données

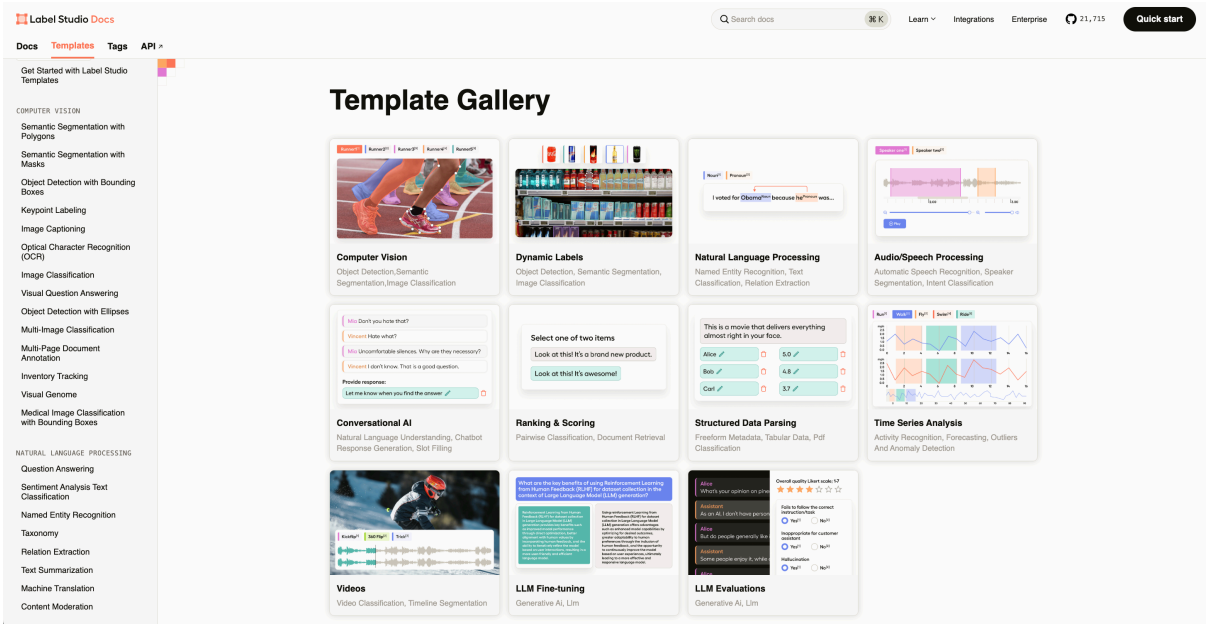
4.1 Structuration des données

Pour la structuration des données, nous souhaitons adopter la même approche pour les données récupérées via MQTT et celles récupérées sur le téléphone. Étant donné que les données MQTT sont plus contraignantes et sont fournies sous forme de dictionnaires, nous allons également utiliser une liste de dictionnaires pour les fichiers locaux. Enfin, nous enregistrerons le tout dans un format JSON.

4.2 Collecte des données et labélisation

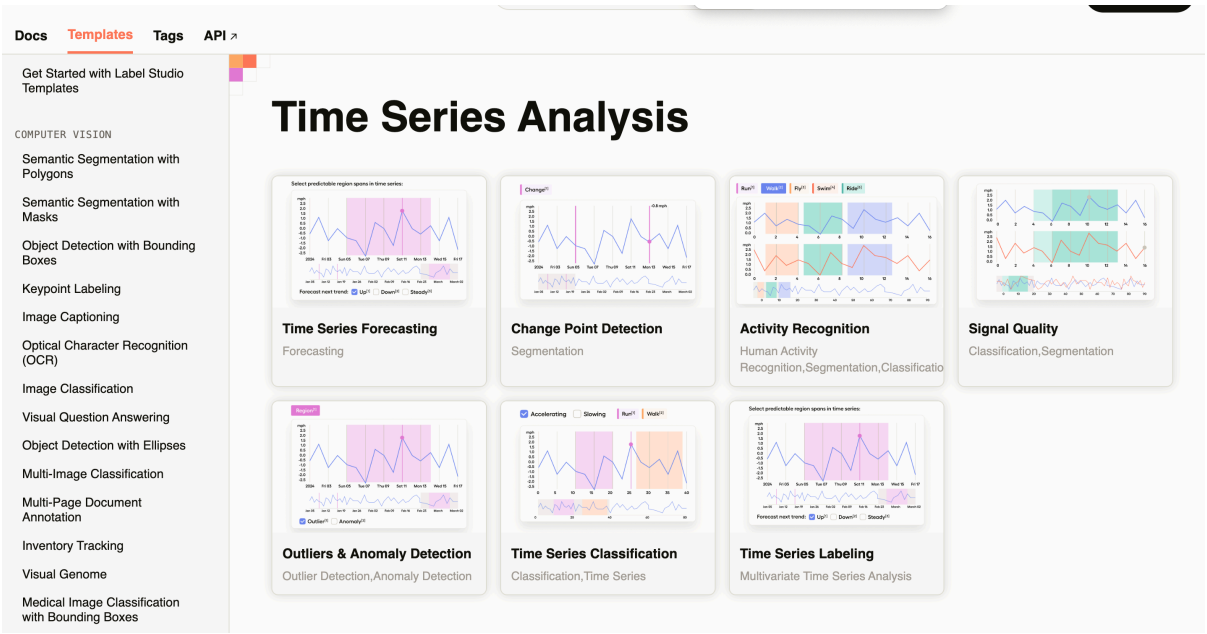
Pour les données, nous avons collecté quelques cas de chute à vélo nous-mêmes, comme demandé dans le sujet. Nous avons également utilisé des données ouvertes accessibles en ligne pour obtenir un nombre plus important de capteurs sur des vélos. Dans les deux cas, l'étape suivante pour la détection consiste à préparer nos données pour les labellisés. Pour cette partie, nous avons choisi d'utiliser l'outil LabelStudio, disponible via Python.

LabelStudio est une application web Django installable via un `pip install` qui offre un outil de labellisation complet pour divers types de données, notamment les images, les vidéos, l'audio, le texte et les séries temporelle.



Capture d'écran 1 : Template de labélisation via label studio
Source : https://labelstud.io/templates

Plusieurs templates sont disponibles pour les séries temporelles, ici on va utiliser plutôt le template Activity Recognition et venir l'adapter à notre projet et à nos données.



Capture d'écran 2 : Liste des template pour les TimeSeries
Source : https://labelstud.io/templates/gallery_timeseries

En combinant les données en ligne et nos propres données, nous avons obtenu près de 350 fichiers de données, soit plus de 800 000 données pour chaque variable. Toutes les données ont été labellisées afin d'identifier les chutes dans la base de données. Nous avons utilisé deux labels : « normal » et « chute ». Après labellisation, nous avons constaté que nous avions beaucoup plus de données classées comme « normal » par rapport au nombre de « chute ». Cependant, en raison de la complexité de la collecte de données, nous n'avons pas pu en obtenir beaucoup plus.

5. Analyse

5.1 Les algorithmes à étudier

N'ayant jamais développé d'outils de classification sur des séries temporelles, nous explorons toutes les solutions possibles. Nous avons effectué une étude rapide des outils potentiels et avons établi une liste de ceux-ci.

- L'outil proposé, déterminer les k voisins d'une donnée
- LSTM (Long-Short Term Memory)
- CNN
- GRU
- TCN
- TFT

L'objectif est d'explorer systématiquement ces algorithmes de classification afin de déterminer s'il est possible de détecter efficacement les chutes à l'aide de l'un d'entre eux.

5.2 k voisins

L'objectif de ce projet est de mettre en œuvre un système de détection de chute en utilisant l'algorithme fourni. Nous avons codé et testé cette méthode, et les résultats de l'entraînement n'ont pas été prometteurs. La matrice de confusion ne donne pas des résultat satisfaisant du tout. Nous avons testé plusieurs valeurs de K (5, 10, 20, 50) et avons choisi d'utiliser 50 car elle a donné les meilleurs résultats. Cependant, l'utilisation de valeurs de K supérieures à 50 entraîne un temps d'exécution significativement plus long.

Lors de la mise en œuvre du modèle, nous avons remarqué que de nombreuses chutes n'étaient pas détectées. Pour la mise en œuvre initiale, nous avons tenté de prédire la classe toutes les 50 données, car c'était la première fois que nous travaillions avec des données temporelles.

Les résultats étaient catastrophiques lors de la visualisation sur des fichiers d'exemple. Nous avons donc modifié le code pour qu'il ne fonctionne pas toutes les 50 secondes, mais d'une manière différente. Le signal est découpé en fenêtres glissantes de taille fixe, avec un chevauchement pour éviter toute perte d'information. Chaque segment est transformé en un vecteur de caractéristiques, puis normalisé et passé au modèle de classification pour prédire la classe (Accident ou Normal). Si la confiance du modèle est supérieure à 0,85, la prédiction est acceptée ; sinon, elle est ignorée (« Non-classifié »). En cas de changement de classe soudain, une confiance encore plus élevée (supérieure à 0,90) est exigée pour éviter les faux positifs. Cette méthode permet d'obtenir des résultats plus cohérents dans l'ensemble. Cependant, la méthode K-Voisin décrite précédemment donne des résultats vraiment inutilisables pour une mise en fonctionnement.

5.3 LSTM

Ce modèle offre de meilleurs résultats que le précédent. Cependant, un défi majeur réside dans la quantité de données, notamment les chutes. Pour trouver un équilibre entre les classes, nous devons nous contenter de 400 découpages. Il est important de noter que cela ne représente pas seulement 400 données, mais 400 fichiers de séries temporelles labellisés (200 de chaque classe). Malgré la quantité de données collectées, ce nombre est relativement faible.

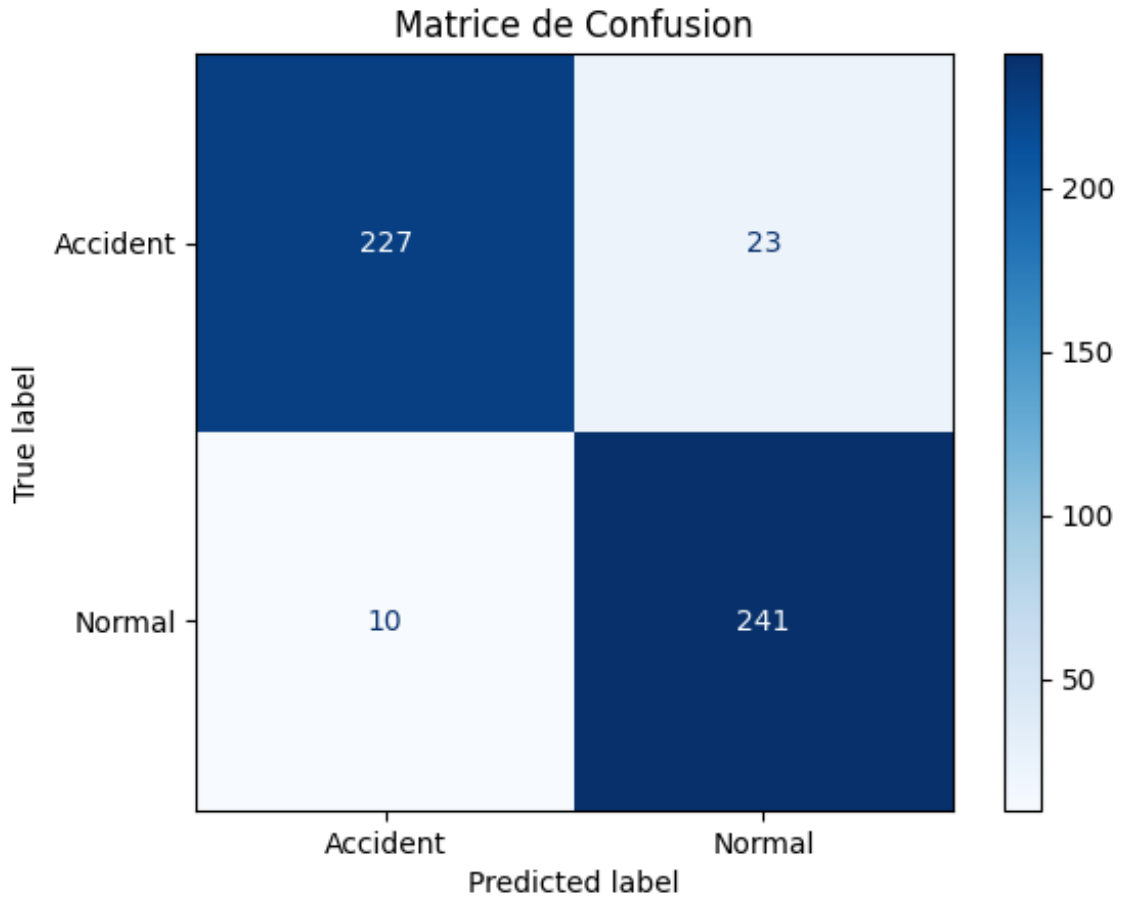


Figure 3 : Matrice de convolution LSTM

La matrice de convolution montre des résultats très prometteurs. Cependant, lors des tests après l'intégration, les résultats sur des cas d'usage sont meilleurs que ceux du modèle précédent, mais présentent encore des zones de défaut.

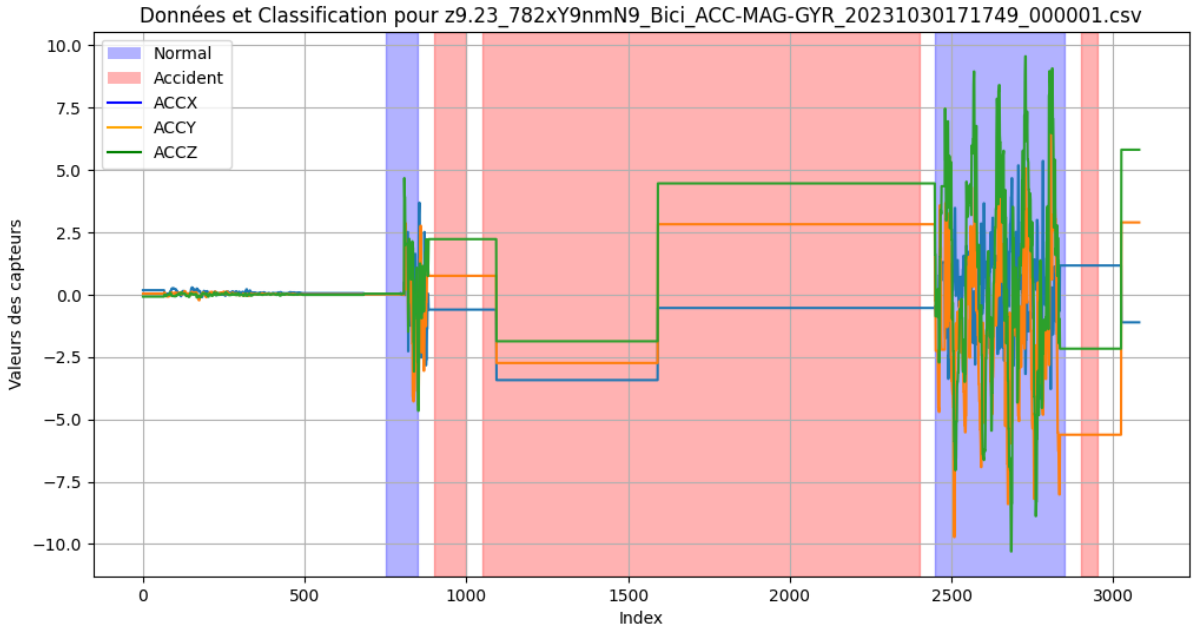


Figure 4 : Detection de chute

Dans cet exemple, nous avons un graphe qui détecte les zones normales avant une chute ou un événement similaire. C'est très proche de ce que l'on souhaite avoir en temps réel pour des données reçu via MQTT.

5.5 Detection d'anomalie

Après avoir exploré des approches supervisées telles que les k-voisins et LSTM, nous avons opté pour une approche différente basée sur la détection d'anomalies. Cette méthode s'est avérée particulièrement prometteuse car elle ne nécessitait pas de données étiquetées de chutes, qui, comme nous l'avons vu, sont difficiles à obtenir en grande quantité.

Notre approche consistait à analyser les données de conduite normale pour établir un profil de référence. Nous avons utilisé des fenêtres glissantes pour analyser les données temporelles et attribuer des scores d'anomalie à chaque fenêtre, ce qui nous a permis d'identifier les déviations significatives par rapport au comportement normal.

Le plus gros problème était la génération de nombreux faux positifs : des événements normaux mais brusques, comme des freinages ou des virages

serrés, étaient fréquemment détectés comme des anomalies. De plus, la méthode ne prenait pas suffisamment en compte la séquentialité des événements, ce qui conduisait à des détections isolées peu pertinentes.

Bien que cette expérience n'ait pas abouti à une solution viable pour notre application finale, elle nous a permis de mieux comprendre les limites des techniques de détection d'anomalies dans le contexte de la détection de chutes à vélo. Cela justifie également notre décision de nous concentrer sur les approches supervisées présentées précédemment, malgré leurs limitations en termes de données d'entraînement nécessaires.

5.4 Conclusion

Les résultats obtenus avec LSTM se sont avérés les plus prometteurs dans notre cas. Nous avons également testé d'autres modèles, mais leurs résultats n'ont pas été aussi satisfaisants que ceux de LSTM. Les problèmes rencontrés étaient souvent similaires : de bons résultats en phase d'entraînement, mais des performances décevantes en phase d'inférence. Seul CNN a pu s'approcher des résultats de LSTM. Certaines méthodes développées n'ont pas pu être implémentés en raison de problèmes de dépendances non résolus.

6. Conclusion

Le projet de développement d'une application Android pour la détection de chutes à vélo a été une expérience enrichissante, mêlant développement mobile et intelligence artificielle. Malgré les défis rencontrés, nous avons réussi à atteindre les objectifs principaux du projet :

- La collecte et l'envoi des données capteurs via MQTT.
- La mise en place d'un système de détection de chutes basé sur l'analyse de séries temporelles.
- La création d'une interface utilisateur simple et accessible, adaptée à différents profils d'utilisateurs.

Notre approche, mêlant flexibilité dans la gestion de projet et travail en parallèle sur des volets distincts, nous a permis de surmonter les contraintes de temps et de ressources. Si certaines fonctionnalités secondaires, comme l'affichage graphique des données historiques, n'ont pas pu être finalisées, elles représentent des pistes d'amélioration intéressantes pour l'avenir.

Ce projet a également mis en lumière l'importance de l'adaptabilité dans la gestion de projet, notamment en ce qui concerne la collecte des données et l'intégration des différentes composantes techniques. Les choix méthodologiques, bien que parfois éloignés des standards classiques, se sont avérés efficaces dans notre contexte.

En conclusion, ce projet nous a offert une expérience complète et très enrichissante du développement via une gestion de projet. Bien que le résultat final ne nous ait pas entièrement satisfaits, nous considérons la partie gestion de projet comme une réussite. Certaines améliorations pourraient être apportées d'ici la fin complète du projet.