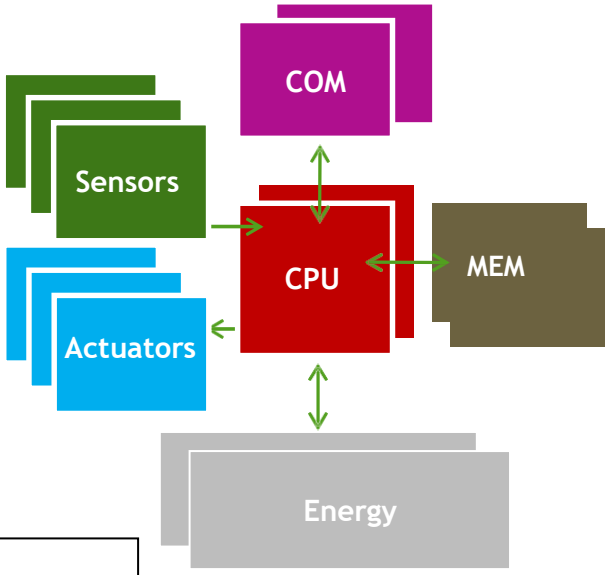
# FICHE TECHNIQUE DE l'EQUIPEMENT CIBLE MOBILE

Noms du trinôme :

- Quentin BEAUCHET
- Yann FORNER
- Michael MARYNOWICZ

Constructeur et intitulé de la cible :

Wear Os Small Round API 30




## Alimentation :

La montre fonctionne sur batterie et on la charge par induction.

Notre application n'est pas prévue pour une utilisation prolongée sans recharger la montre car celle-ci envoi des données vers notre API et récupère sa position de manière constante ce qui réduit considérablement la durée de la batterie.

Nous avons fait le choix de ne pas passer par le téléphone pour que la montre puisse être utilisée indépendamment de celui-ci.

## Mémoires :

Nous stockons seulement les layouts et quelques variables ce qui fait que l'application nécessite peu de ressources.

## CPUs :

Notre application nécessite de rester active en fond pour envoyer des informations à l'api à intervalles régulières mais cela n'impacte pas négativement l'utilisation de la montre.

## Modules de Communications :

Nous n'utilisons pas de communications Bluetooth entre la montre et le téléphone car nous voulions que celle-ci reste indépendante et donc elle effectue elle-même les requêtes http GET et POST vers notre API.

## Capteurs :

Le seul capteur de la montre que nous utilisons et la géolocalisation, celui-ci étant suffisant pour le type d'application que nous voulions créer.

## Actionneurs :

L'application permet à l'utilisateur de démarrer la géolocalisation et de la stopper, il peut aussi régler la fréquence de celle-ci. De plus il est aussi possible de d'envoyer des évènements manuellement ce qui permet à l'utilisateur de signaler des éléments intéressant lors de son trajet en envoyant leurs types, leurs positions et leurs dates.

# ENVIRONNEMENT LOGICIEL

## Environnement de Développement

Nous avons utilisé Android Studio et nous avons testé notre application depuis l'émulateur ***Wear Os Small Round API 30*** sous Windows/linux et mac.

## Limitations Logiciels

Evidement notre application nécessite l'accès au réseau et une montre pouvant utiliser la géolocalisation.

# PROJET

## Objectifs :

Le but de notre application est de permettre à un utilisateur de notifier les autres utilisateurs d'évènements qu'il considère importants lors de son trajet. Pour cela notre application envoi a Intervalle régulière sa position vers une API distance qui se charge d'agréger les données de tous les utilisateurs dans une base de donnée et les affiche dans un Dashboard. Certains des types d'événements sont des bouchons, des travaux sur le bord de la route, des déchets …
L'utilisateur peut choisir la fréquence a laquelle sa position est envoyé automatiquement vers l'API et de stopper cet envoi. L'application fonctionne en arrière-plan grâce a une notification permanente.

### Données collectées :

- La géolocalisation de la montre.
- L'id unique de la montre.
- L'heure et la data de chaque évènement.

*Commentaires :*

*L'utilisation d'un id unique peut poser un problème car si celui est récupéré par un pirate il permettrais à celui-ci d'identifier la montre en permanence car ce numéro est inchangeable. Nous avons opté pour cette approche qui nous évitais de devoir réaliser un système d'identifiants de connexion avec pseudo et mot de passe.*

### Traitement des données collectées sur la cible :

- On affiche les évènements dans une carte interactive.
- On affiche les types évènements dans un histogramme.
- On affiche les types évènements dans un digramme circulaire.

*Commentaires :*

*Toutes ces données sont visibles depuis notre Dashboard.*

### Transmission des données collectées à un système distant :

- Ces données sont transmises en utilisant le protocole http vers l'API qui se charge de les stocker dans une base de données SQL.

### Stockage des données collectées sur un système distant :

- Elle sont stockées dans une base de données SQL dans 3 tables différentes, une pour les utilisateurs, une pour les types d'évènements et une pour les évènements eux même.
Un utilisateur et un type peuvent avoir plusieurs événements.

## Contraintes et Solutions

Du fait que la fonction principale de notre application est la géolocalisation, on se doit d'envoyer des informations vers l'API à intervalles réguliers. Notre application étant une démonstration nous n'avons pas pris en compte les contraintes de bande passante et de consommation d'Energie.

Pour améliorer ses deux points il serait possible de stocker les données en local dans la montre et de les envoyer de façon journalière.