


# Guidage d'une personne malvoyante par application Android

Berthelot Florian  
Dantchiawa Mohamed-Ouaraqua  
Guitard Alan  
Marcy Nicolas



# Plan

1. Description du domaine
    - a. Contexte
    - b. Solution proposée
    - c. Fonctionnement d'une application
  2. Fonctionnement du logiciel
  3. Architecture de l'application
    - a. Module de saisie d'adresse
    - b. Module de gestion de carte et d'itinéraires
    - c. Module synchronisation
  4. Tests
  5. Bilan
- 

# Description du domaine d' intervention

## Contexte existant

Applications existantes adaptées aux aveugles :

- TalkBack



- Saisie vocale



- Projet Navi'Rand



## Contexte existant

Les GPS classiques proposent diverses fonctions :

- Guidage par voix
- Carte Google Maps
- Géolocalisation du téléphone ou d'une adresse entrée
- Tracé du chemin sur la carte
- Guidage par sigle (fig. 1)

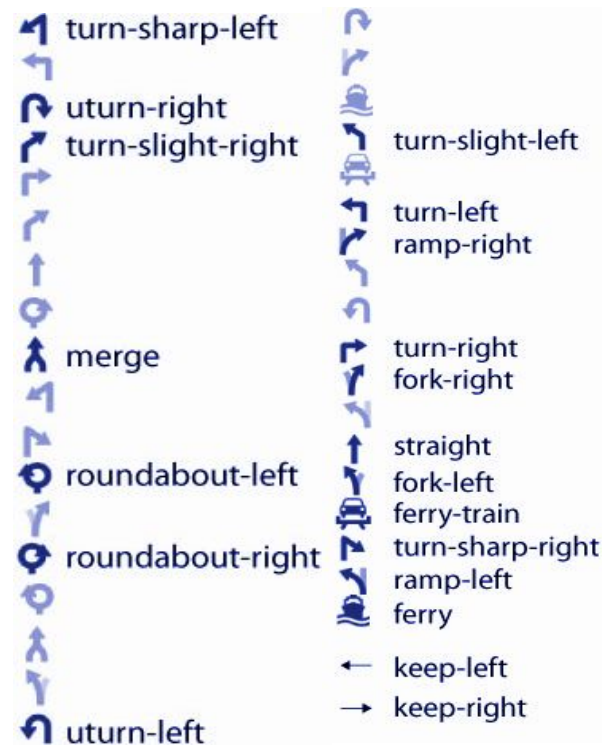


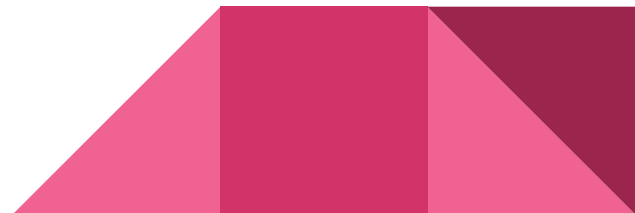
Figure 1: Sigles de directions

## Solution proposée

Utiliser un autre moyen comme vecteur de communication d'informations : **Le sens du toucher, par la vibration**

Deux modes :

- Un seul téléphone : vibre une ou deux fois pour une direction.
- Deux téléphones : l'un dédié pour la gauche, l'autre pour la droite.



# Fonctionnement d'une application Android

Composantes d'une application :

- Activités
- Services (Google Maps, ...)
- Broadcast et Intent Receivers
- Content Providers

Fonctionnement d'une activité :

- UI thread, tâches de fond
- Callbacks

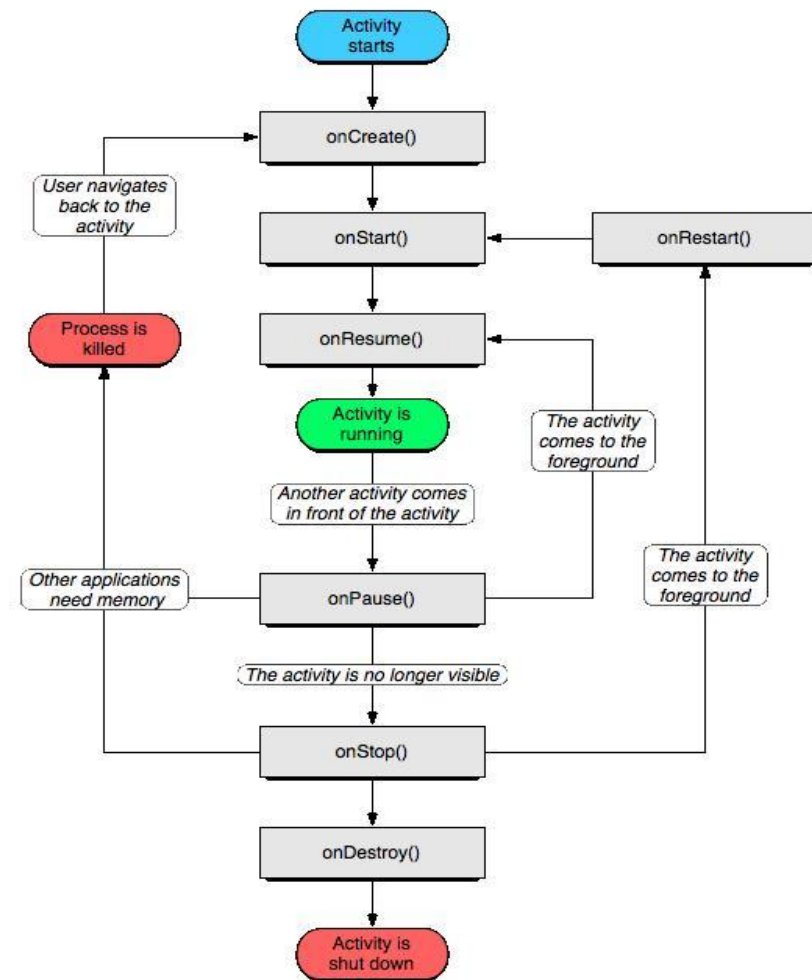



Figure 2: Cycle de vie d'une activité

# Environnement et services utilisés

## Services utiles:

- Google Maps API
  - Maps
  - Directions
  - Geocoder
- Capteurs de l'appareil
  - Wifi, réseau mobile
  - GPS
  - Accéléromètre

## Environnement de développement et de tests :

- Android Studio 
- Android SDK version 24
- JUnit4
- Mockito

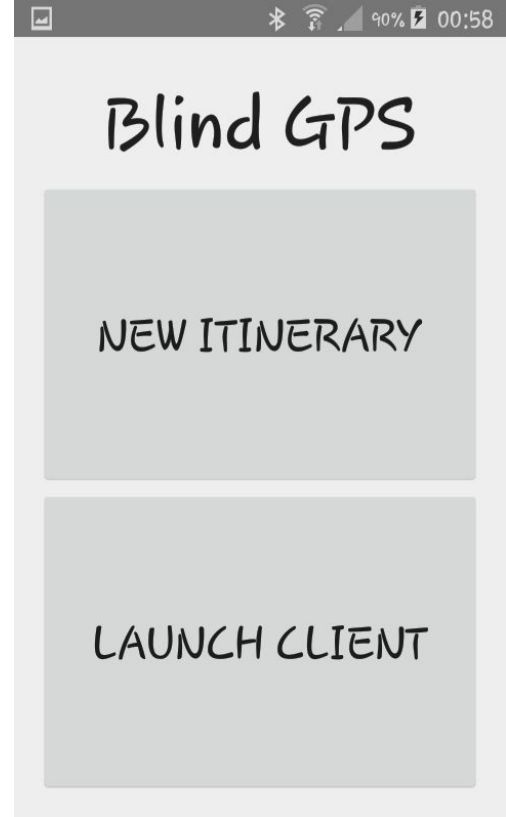


# Fonctionnement du logiciel

# Ouverture de l'application

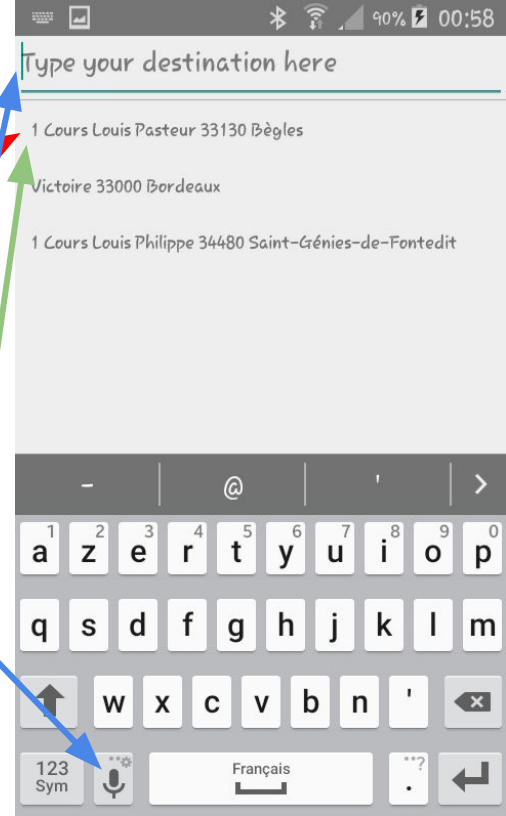
Deux boutons:

- *Nouvel itinéraire* → démarre l'activité de saisie d'adresse (en ayant au préalable vérifier la connexion réseau)
- *Lancer client* → affiche une liste d'appareils Bluetooth à proximité afin de sélectionner le téléphone serveur avec lequel se coupler



# Interface de saisie d'adresse

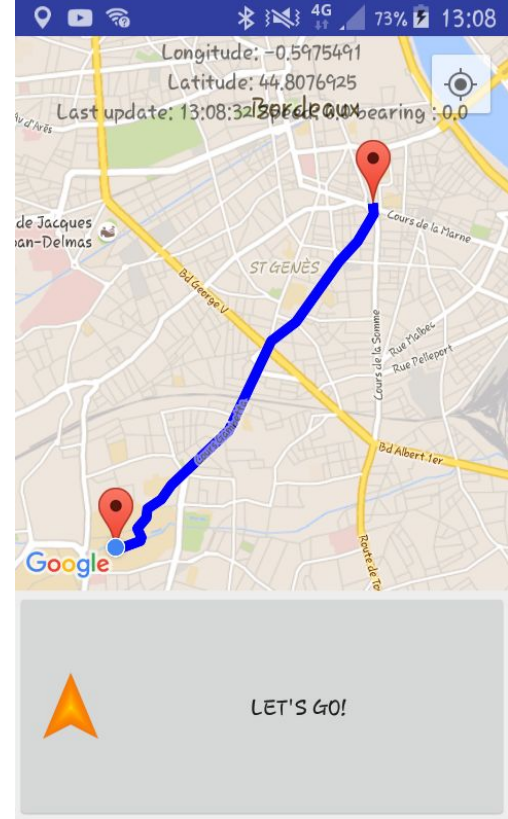
- Sélection d'une adresse récente
- Tape au clavier d'une adresse ou saisie vocale
- Sélection d'une suggestion d'adresse



# Carte

L'utilisateur peut appuyer sur le bouton dès qu'il est prêt à partir.

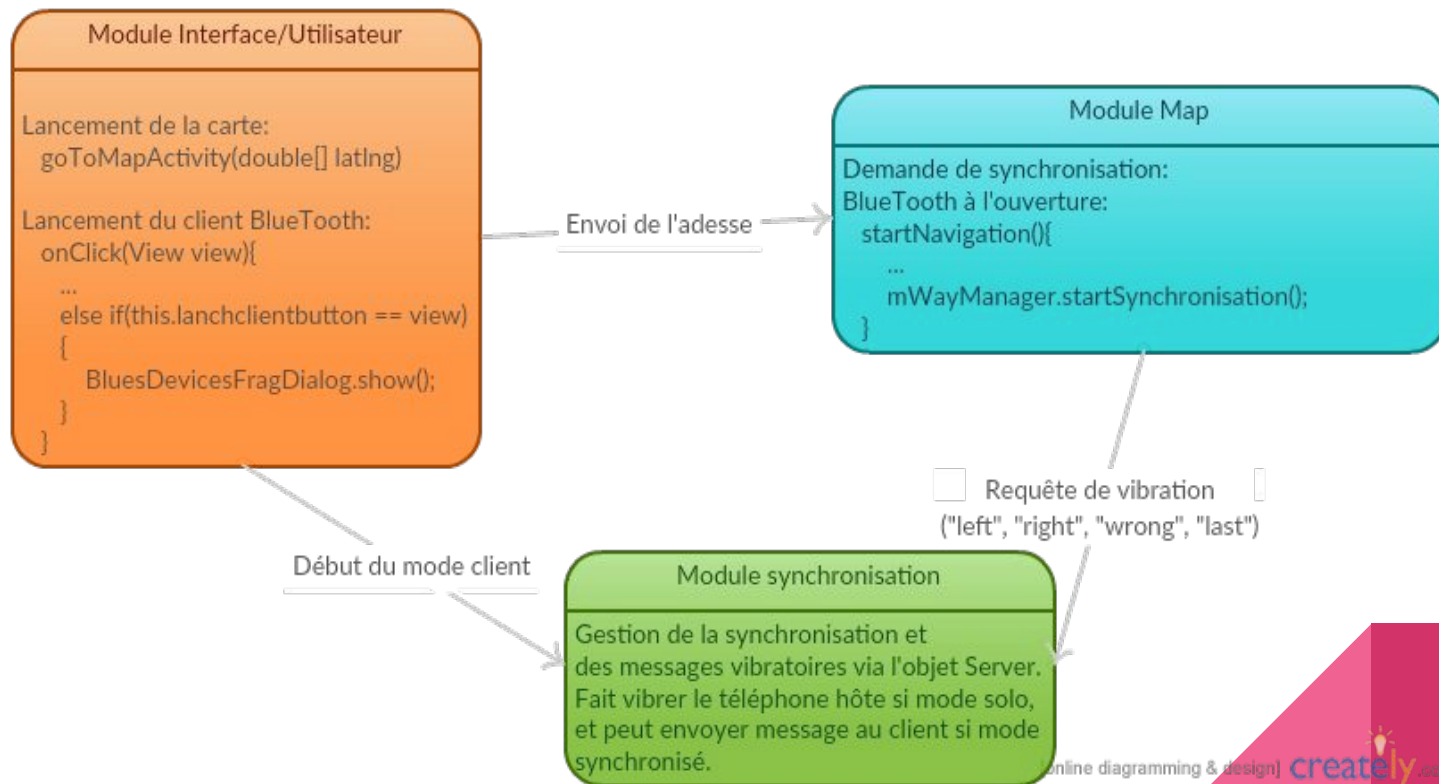
Activité qui contient les objets de gestion des vibrations et lance le serveur Bluetooth.



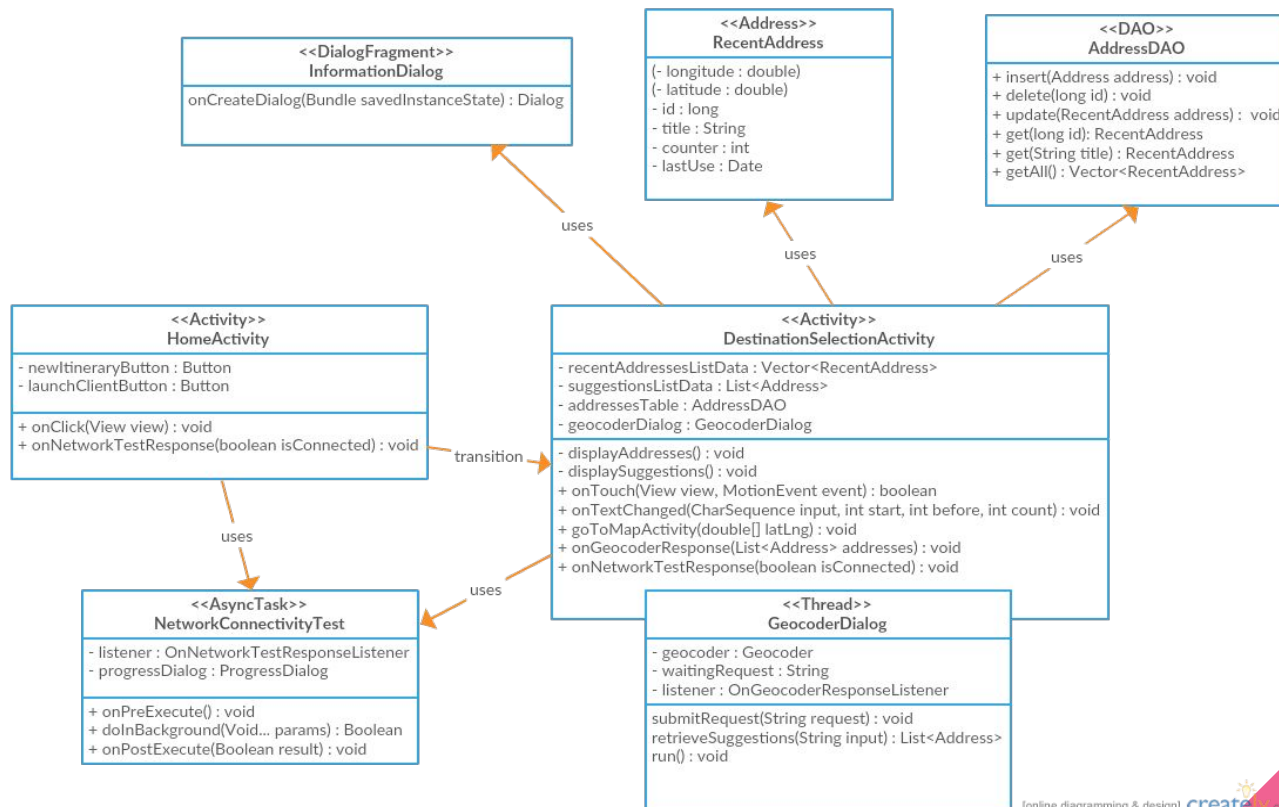


# Architecture et conception de l' application

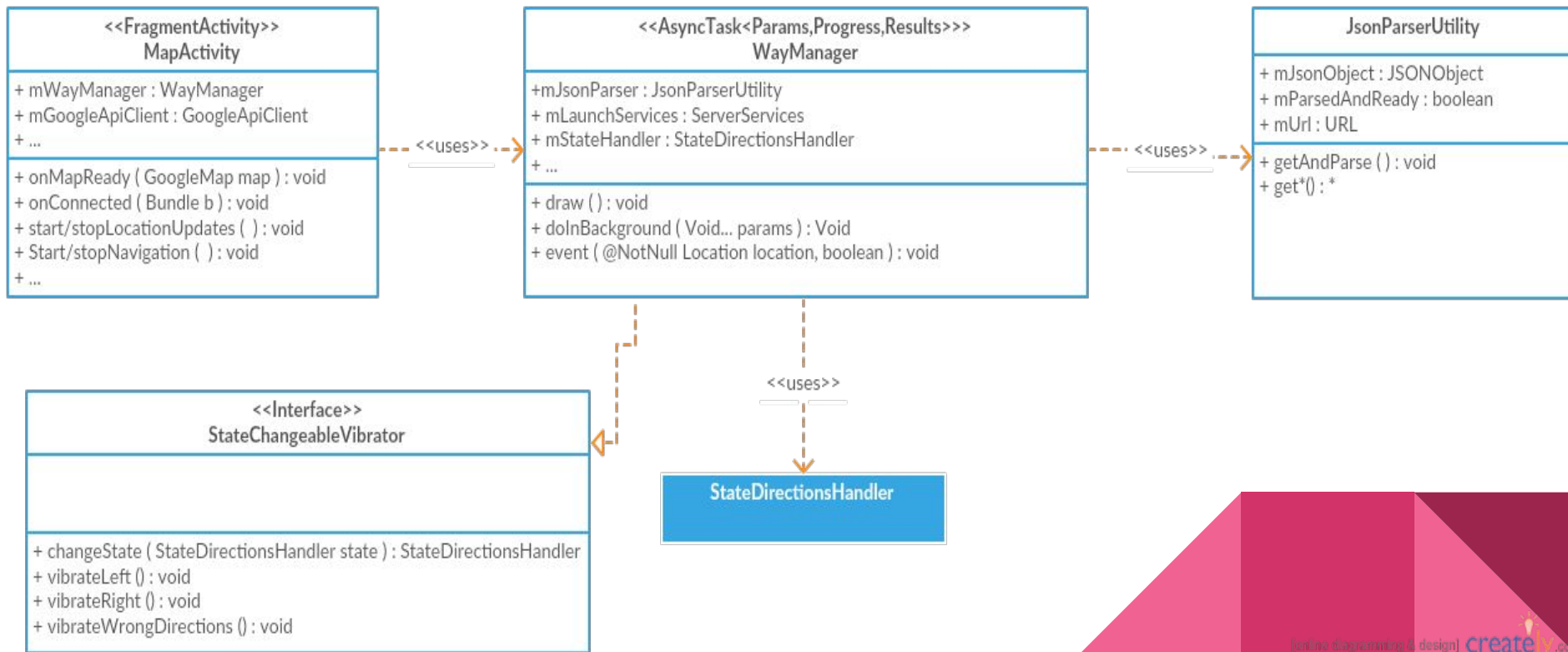
# Architecture des modules



## Module interface utilisateur

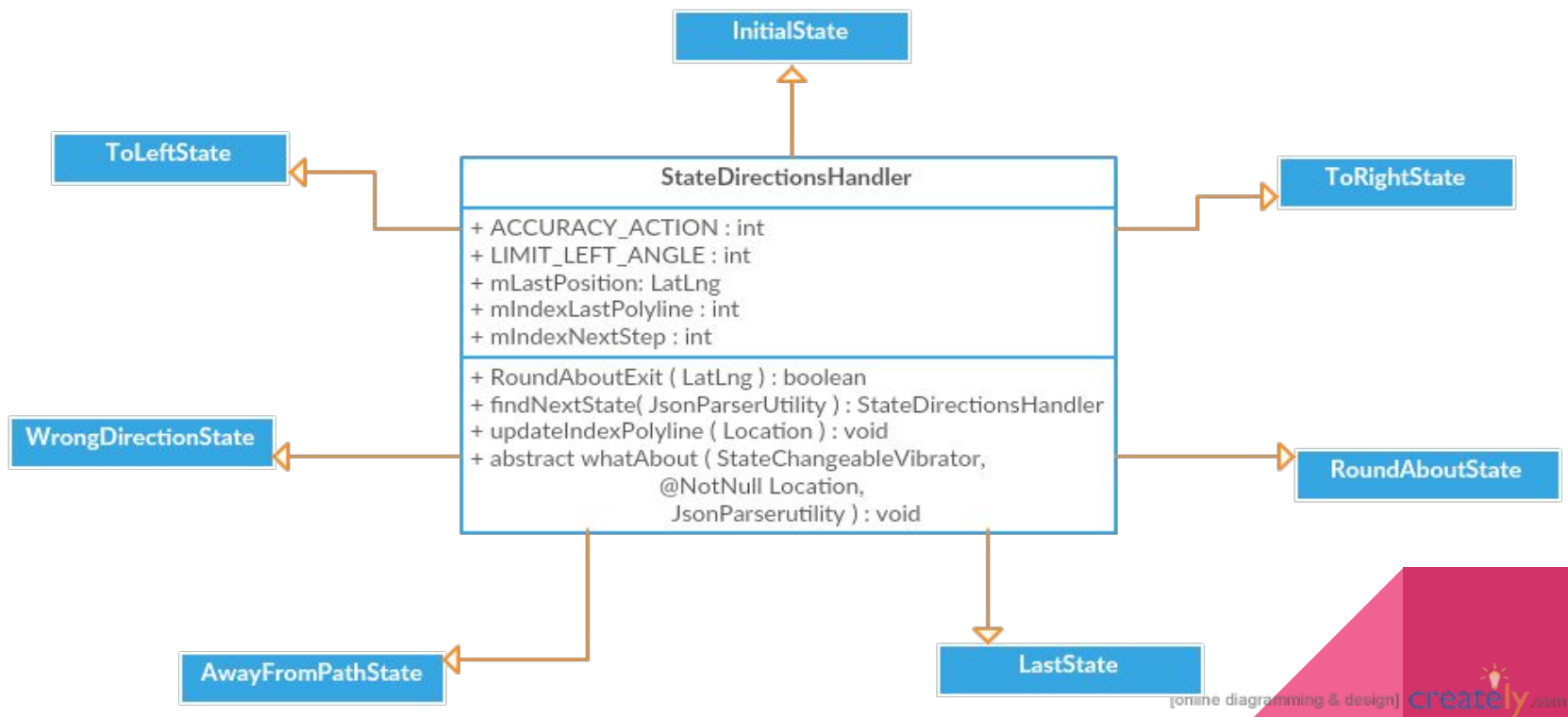


# Module Gestion de carte et Itinéraire





# Gestion des états

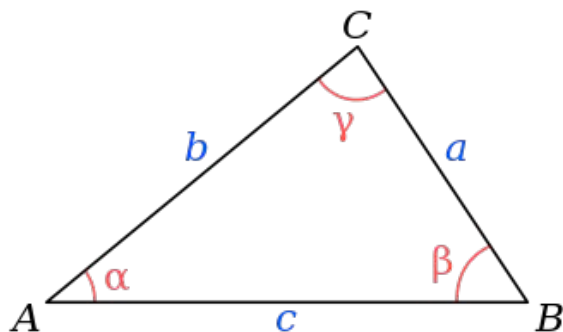


# Quelques détails d'implémentation (1)

## Problème de passage de rond-point:

Deux étapes à gérer contrairement aux autres états.

Loi des cosinus:  $c^2 = a^2 + b^2 - 2ab \cos \gamma$ .  $\Rightarrow \gamma = \frac{\arccos(A^2 + B^2 - C^2)}{2AB}$

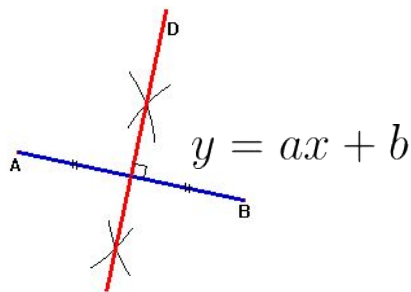


## Quelques détails d'implémentation (2)

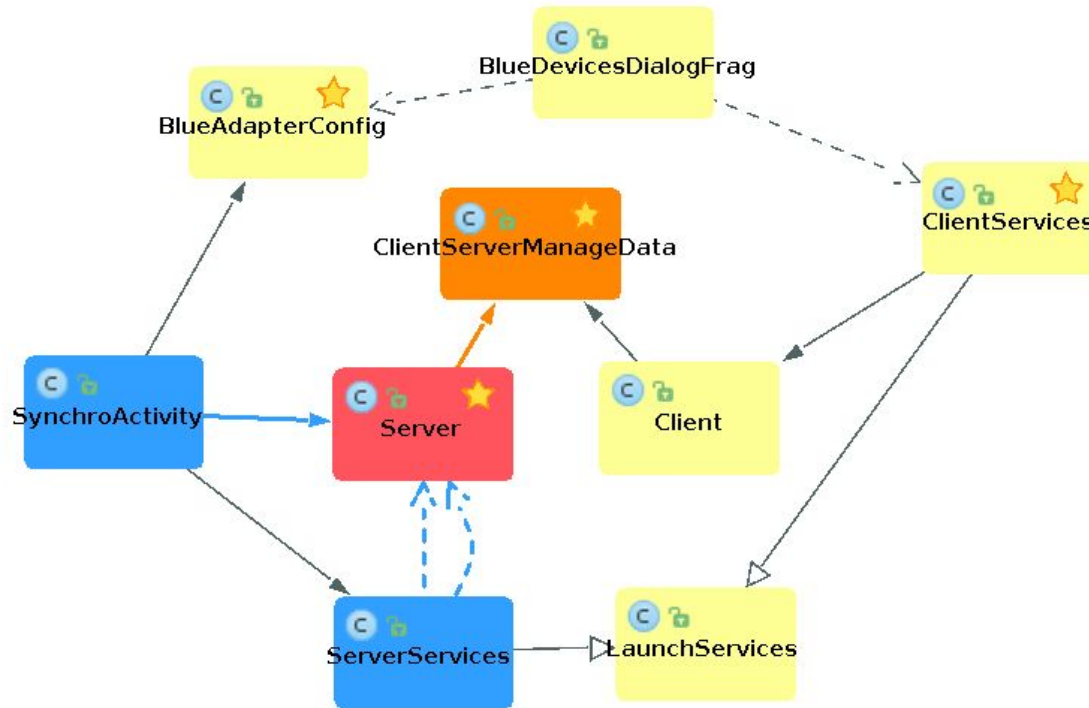
### Approximation de la position sur le chemin

Deux droites perpendiculaires sont liées par :  $a1 * a2 = -1$

Trouver le b de la droite cherchée.



# Module Synchronisation



————> Dépendance

————> Étends

- - - - -> Référence

# Architecture Client/Serveur

- Le serveur est démarré lorsqu'on démarre l'itinéraire.
- Lorsque le client se connecte, une boîte de dialogue affiche que la connexion s'est bien passée.
- Le serveur choisit le côté pour lequel son téléphone vibre.
- Le serveur fait vibrer le client ou se fait vibrer soi-même.



## Description du protocole

- la fonction writeBlue(String) dans le serveur
- les pings du client vers le serveur et les acquittements
- Changement de mode en cas de déconnexion (duo ou solo)



# Tests

# Tests Unitaires

- Utilisation de JUnit4 :
  - Annotations
  - Assertions
- Utilisation de Mockito :
  - Mocker
  - Modifier le comportement
  - Vérifier



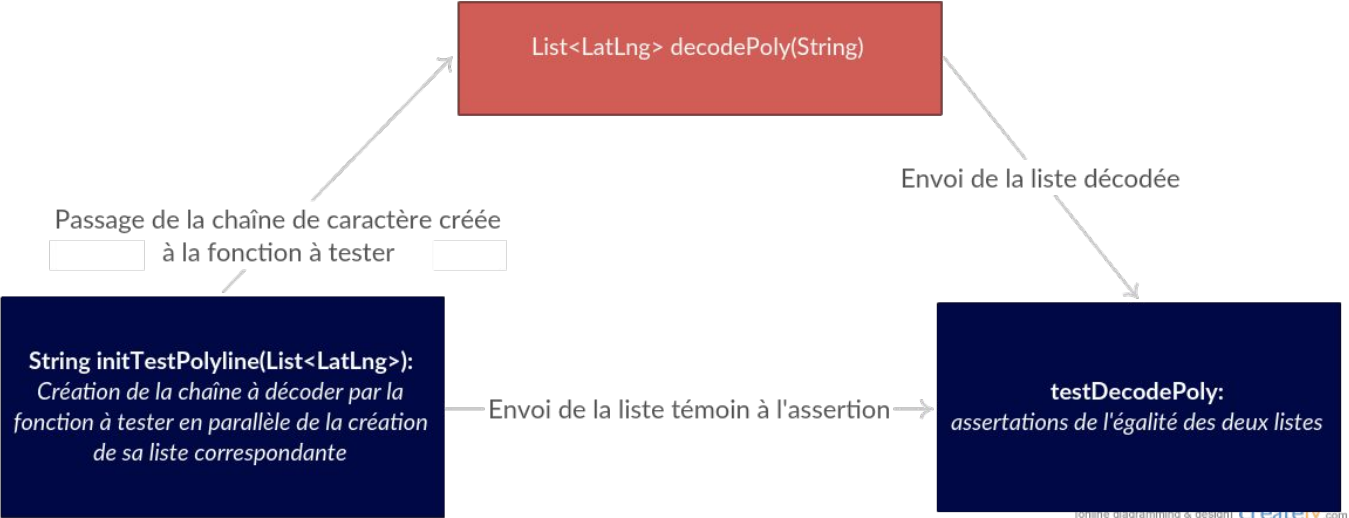


# Méthodes de tests

- Tests sur méthodes de calculs géométriques
  - Test des cas critiques
  - Test par oracles
- Tests sur méthodes d'envoi de message par BluetoothSocket



# Exemple de test sur la méthode decodePoly (String)





# BILAN

# Améliorations possibles

- Envoi de coordonnées GPS du client vers le serveur
  - Augmentation de la précision de la localisation
- Synchronisation avec d'autres terminaux Bluetooth
  - Bracelet, casque ou écouteurs, hardware créé pour l'application,...
- Ajout d'une fonctionnalité style TalkBack sur toute l'application
  - Ergonomie plus adaptée aux aveugles
- Amélioration du protocole Bluetooth
  - Augmentation de la sécurité, interruptions de discussions mieux gérées
- Amélioration du code vibratoire
  - Compréhension de l'utilisateur



**Merci de votre attention!**

