Département Informatique

IUT de Clermont-Ferrand

IUT CLERMONT - FD
UNIVERSITÉ
Clermont Auvergne

3 Avenue Blaise Pascal,

63170 Aubière

# Rapport de projet tuteuré

**Réalisation d'un logiciel permettant d'utiliser l'appareil Leap Motion* pour lancer des scripts à partir de gestes prédéfinis reconnus par l'appareil.**


Réalisé par : Thomas BLANC, Yoann PERIQUOI

Emrick PESCE, Romain Olivier, Augustin LABORIE

Réalisé du 09/11/2020 au 29/03/2021

Professeur tuteur : Laurent Provot

Année 2020-2021

# AUTORISATION A DIFFUSER SUR L'INTRANET DE L'IUT

Le groupe composé de Romain OLIVIER, Thomas BLANC, Yoann PERIQUOI, Emrick PESCE et Augustin LABORIE autorise le DUT Informatique de Clermont-Ferrand à diffuser notre rapport sur l'intranet de L'IUT de Clermont-Ferrand.

# REMERCIEMENTS

Nous tenons à remercier M. Laurent Provot, professeur au DUT Informatique de Clermont-Ferrand, pour avoir accepté d'encadrer le projet et pour l'aide qu'il nous a apportée durant la réalisation de celui-ci.

Nous tenons également à remercier M. Cédric Bouhours, chef du département Informatique de l'IUT de Clermont-Ferrand, pour nous avoir conseillé lors de la première soutenance tenue le 18/01/2021. (Cf. III. 3. Vision générale de l'architecture)

# SOMMAIRE



Les mots suivis d'un astérisque (*) sont définis dans le lexique.

# I. Introduction

Ce rapport s'inscrit dans le cadre du projet tuteuré final obligatoire dans le cursus du DUT Informatique de Clermont-Ferrand.

Nous avons choisi de définir notre propre sujet car aucunes des propositions faites par les professeurs ne nous intéressaient réellement et nous voulions expérimenter notre propre idée. Romain Olivier et Augustin Laborie ont eu l'idée d'utiliser l'appareil Leap Motion* présent au Club informatique du DUT Informatique de Clermont-Ferrand afin de réaliser un logiciel à partir de celui-ci. Le Leap Motion* est un capteur du mouvement de la main et des doigts comme périphérique d'entrée, à l'instar d'une souris. Cet outil nous permet de renvoyer des informations sur les mains de l'utilisateur.

Le reste de l'équipe constitué de Thomas Blanc, Emrick Pesce et Yoann Périquoi ont alors apprécié l'idée proposée par leurs deux camarades et ont rejoint le groupe. En effet, une telle idée représente un sujet intéressant dans le cadre de l'approfondissement des compétences d'un développeur informatique. Partir d'une base matérielle, pour greffer un logiciel dessus, nous a paru être une expérience que nous serons amenés à retrouver dans notre vie professionnelle. Celle-ci nous permettrait, tout particulièrement, d'apprendre à utiliser un tout nouveau matériel et à acquérir des compétences dans le développement de logiciel.

Nous avons ensuite défini le logiciel que nous allions réaliser. Ce projet réside dans la réalisation d'un logiciel permettant d'utiliser l'appareil Leap Motion* pour lancer des scripts à partir de gestes prédéfinis qui seront alors reconnus par l'appareil. Nous devrons utiliser les informations fournies par le Leap Motion* afin de les traduire et de pouvoir définir des gestes. L'utilisateur pourra alors associer un geste au déclanchement d'un script. Lorsqu'un geste sera reconnu par notre logiciel à travers l'appareil, le script qui lui est associé sera exécuté.

Nous avons donc énoncé notre sujet à M. Abdelfeta HASBANI, responsable des projets tuteurés au département Informatique de l'IUT de Clermont-Ferrand. Celui-ci a validé notre sujet de projet et a proposé l'encadrement à l'ensemble du groupe enseignant.

C'est M. Laurent Provot, enseignant au DUT Informatique, qui a accepté d'encadrer notre projet et qui nous a accompagné, en prenant le rôle de client tout au long de sa réalisation.

Afin d'appréhender le projet, nous avons mis en place une certaine rigueur dans notre organisation afin de structurer notre travail selon les conseils de M. Provot. Nous nous sommes longuement interrogés sur les manières d'aborder le départ du développement du logiciel. Nous nous sommes donc demander comment utiliser l'appareil Leap Motion* afin d'exécuter des scripts.

Nous allons découvrir comment le projet a été réalisé graduellement d'abord en plaçant un contexte, en mettant en place une organisation puis on va analyser le sujet posé. Par la suite, nous allons exposer les réalisations délivrées et décrire les difficultés rencontrées. Pour finir, nous allons conclure en faisant un bilan technique et une conclusion générale en anglais et en français.

# II. Présentation du projet tuteuré

## 1. Contexte

Pour réaliser cette application nous avons choisi de diviser notre équipe en deux. Une équipe sera responsable de la réalisation du Front-End* et une autre du Back-End*. Thomas Blanc et Augustin Laborie seront responsables de la réalisation des interfaces tandis que Yoann Périquoi et Emrick Pesce réaliseront la partie métier. Romain Olivier sera responsable de naviguer entre les deux équipes pour apporter son expertise dans les deux domaines.

Ce projet n'est pas une innovation puisque de nombreux autres logiciels sont déjà en circulation sur la plateforme de téléchargement d'applications du Leap Motion* (Leap Motion SDK*). Cependant nous ne partons pas d'une base déjà existante pour réaliser notre propre logiciel, nous utilisons seulement la librairie fournie par les développeurs du Leap Motion*.

Ce projet est réalisé durant la période de la pandémie mondiale du COVID-19. Les cours au DUT Informatique de Clermont-Ferrand sont assurés seulement en distanciel. Le groupe a donc dû s'organiser pour réaliser l'intégralité du projet en distanciel.

Nous avons principalement utilisé le logiciel de communication Discord afin de transmettre nos informations ainsi que de faire des séances de programmation collaborative. Avec l'accord de M. Provot nous avons choisi d'utiliser le GitLab de l'IUT de Clermont-Ferrand qui est un service web d'hébergement et de gestion de développement de logiciels, utilisant le logiciel de gestion de versions Git. Nous avions eu la chance de déjà avoir pu expérimenter le travail à distance car nous avions tous déjà réalisé un projet en distanciel.

Dans la quête de se mettre le plus possible en situation réelle nous avons demandé à M. Laurent Provot de se considérer comme l'initiateur de ce projet pour lequel il serait alors le client faisant l'appel d'offre. Il représentera la maitrise d'ouvrage.

Réciproquement notre groupe sera considéré comme la maitrise d'œuvre avec comme chef de projet Romain Olivier qui a été élu lors d'un vote par l'équipe pour sa maitrise du sujet abordé. Nous devrions donc suivre les directives données par M. Provot en tant que client pour satisfaire ces requêtes.

## 2. Objectifs du projet et contraintes

Les fonctionnalités attendues à la fin du projet ont été mises en valeur dans un cahier des charges réalisé dans le cadre du cours de gestion de projet. Celui-ci nous a permis de savoir dès le début où nous voulions nous diriger. Une grande partie de celui-ci est retrouvé ci-dessous et explique nos attentes vis-à-vis du projet.

Nous nous sommes fixé l'objectif de délivrer un logiciel fonctionnel et déployable, avec lequel sera disponible une interaction complète via une interface graphique et une interface en ligne de commande (ou CLI*) avec le contrôleur Leap Motion*. Ces interactions nous permettront

alors de pouvoir lancer un script (suite de commandes) et ainsi de réaliser des tâches à partir d'un seul mouvement.

Celui-ci devra permettre à l'utilisateur de :

- Gérer ses scripts (créer, modifier, lire, mettre à jour et supprimer les scripts et leurs gestes déclencheurs)
- Initialiser ses scripts grâce à des mouvements
- Gérer la connexion avec le Leap Motion*
- Avoir un retour visuel de ce que perçoit le Leap Motion*
- Lancer un script à tout moment grâce à un outil qui observe en permanence le flux vidéo et repère les gestes
- Enregistrer son environnement sur une base de données distante
- Récupérer son environnement depuis n'importe où grâce au serveur distant
- S'authentifier pour accéder à son profil
- Utiliser son propre profil en local avec une gestion "hors ligne"

Ces fonctionnalités sont également décrites dans ce diagramme de cas d'utilisation :

Diagramme de cas d'utilisation de l'application HandyHand :

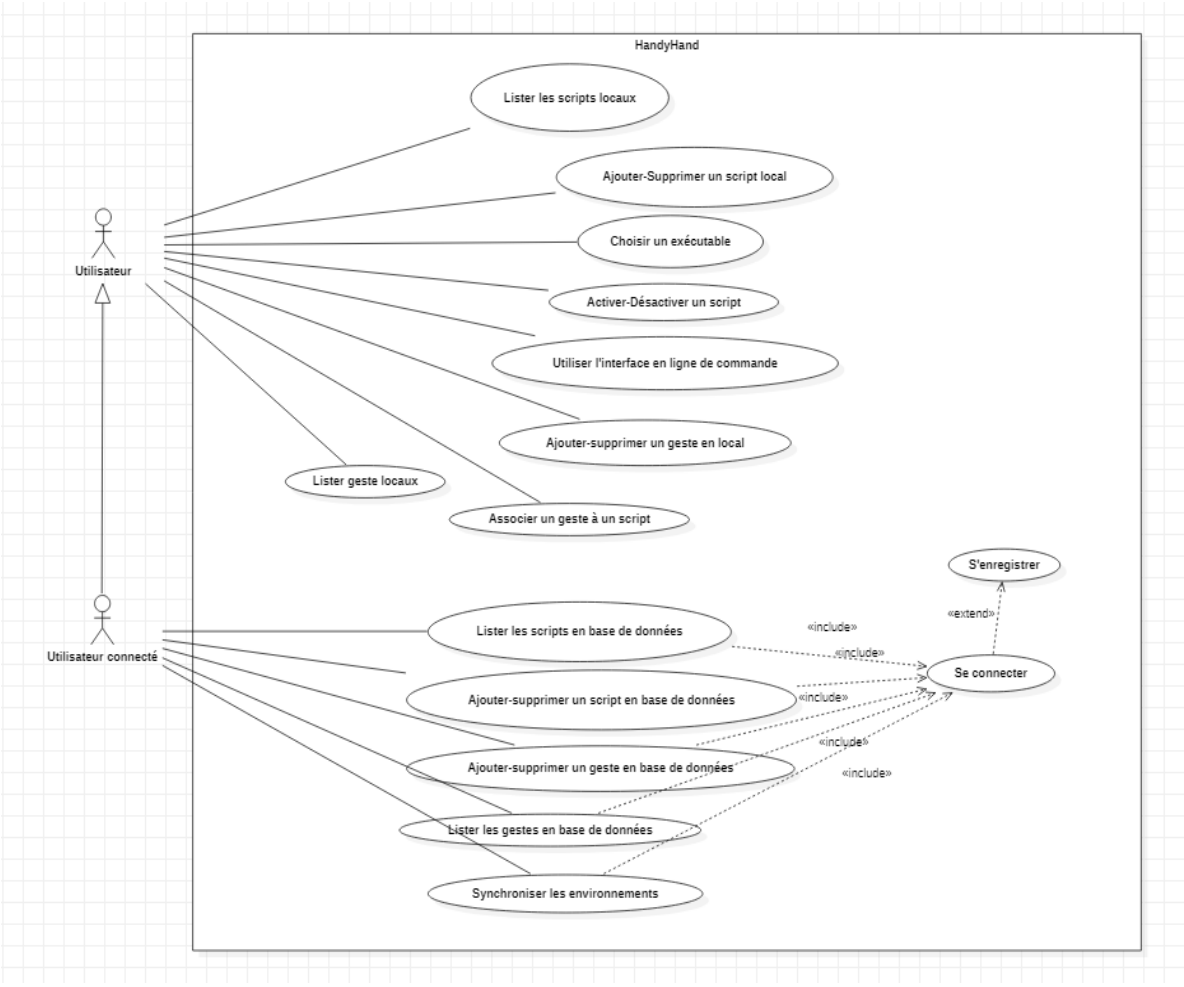



*Figure 1 : Diagramme de cas d'utilisation de l'application HandyHand*

Le projet a débuté le mercredi 9 novembre 2020 et s'achève le lundi 29 mars 2021 soit sa durée est d'un peu plus de quatre mois. Afin de terminer ce projet ambitieux à temps, il est important de correctement le gérer et de le tenir à jour grâce aux outils de gestion adéquats. Dans cette optique, nous utilisons des outils autant présents dans le domaine universitaire que dans le monde du travail : rapports individuels, diagramme de Gantt…

# III. Organisation du Projet

## 1. Mise en place méthode Agile* « light »

Lors de la mise en place du projet, M. Laurent Provot nous a demandé de suivre une gestion de projet particulière. En effet, celui-ci nous a proposé d'utiliser une méthode très populaire dans le domaine du développement de logiciel appelé la méthode Agile*. Seulement celle-ci serait allégée, d'où l'adjectif anglais « light », c'est-à-dire que tous les rouages ne seront, pas ou peu abordés. Cette méthode réside en la mise en place de « sprints ». Cette organisation commence par la définition d'histoires ou « backlogs » qui nous permettent d'énumérer les différentes tâches qui vont devoir être réalisées tout au long du projet.

Dès le début du projet toutes les tâches nécessaires à la réalisation de celui-ci ont été définies. Par la suite, lors de l'initialisation de chaque « sprint » pendant la réunion de démarrage, les « backlogs » ou tâches vont alors être sélectionnées pour être réalisées lors de la période. Nos périodes de « sprint » été de deux semaines, avec une réunion médiane au début de la deuxième semaine. A la fin de ces deux semaines avait alors lieu une « démonstration » où nous devions présenter ce qui avait été réalisés pendant les deux dernières semaines. Nous faisions alors le bilan et M. Provot nous donnait des retours sur les travaux qui avaient été réalisés comme le ferait un client.

Nous avions étudié cette méthode de travail grâce au livre « Scrum depuis les tranchées » écrit par Henrik Kniberg mis à disposition sur le site web de M. Provot. Il nous a permis d'apprendre les bases de cette méthode et de la rendre efficace même si c'était la première fois que tous les membres du groupe l'utilisaient.

Afin de suivre notre avancée, M. Provot nous a aussi demandé de réaliser individuellement un fichier qui serait tenu à jour tout au long du projet et auquel il aurait accès à tout moment. Celui-ci contient le rapport des sessions de travail avec la date, le nombre d'heures travaillées et ce qui a été réalisé.

## 2. Gestion de projet

Lors de cette période une matière consistant en la gestion de projet nous a aussi aidé à apporter de l'organisation au projet. En effet, il nous a été demandé de réaliser de nombreux diagrammes de Gantt ainsi que des rapports individuels hebdomadaires à l'image de ceux demandé par M. Provot. Nous avons aussi produit un cahier des charges encore une fois un peu allégé pour décrire toutes les attentes du projet. Une grande partie de celui-ci est retrouvé dans la partie II.2 Objectif du projet et dans l'annexe. On y voit toutes les fonctionnalités attendues. Celui-ci nous a permis, dans un premier temps, de définir le contexte, les enjeux, les objectifs techniques ainsi que les livrables et les axes de développement envisagés. Nous avons donc pu vérifier avec notre professeur de gestion de projet Mme. Goi et M. Provot la concordance et la faisabilité de notre projet.

Dans le cadre de la mise en place de l'organisation du projet nous avons aussi tracé un grand nombre de diagramme de GANTT : graphique qui permet de visualiser l'ensemble des tâches accomplies et à accomplir avant la fin d'un projet.

On peut par exemple retrouver ici un extrait du diagramme de GANTT du prévisionnel commun sur la période du 7 décembre 2020 au 9 janvier 2021 retrouvable au complet en annexe.

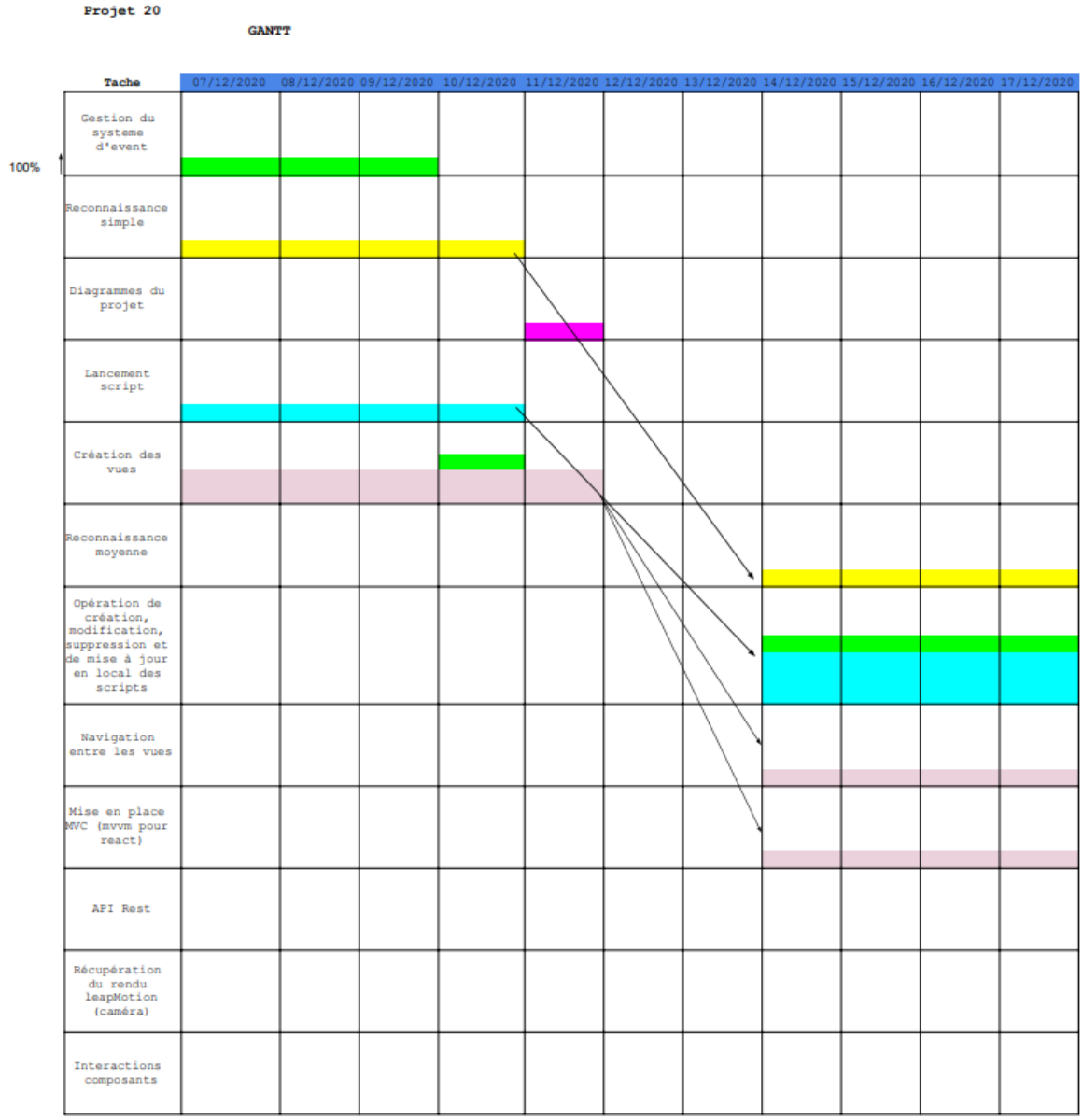



*Figure 2 : Extrait diagramme de GANTT du prévisionnel commun pour la période du 7 décembre 2020 au 9 janvier 2021.*

La réalisation a été facilitée étant donné la mise en place de la méthode Agile* que nous avait demandé de faire M. Provot précédemment. En effet nous n'avions qu'à reporter les tâches prévues pendant les sprints.

Cette organisation nous a permis de suivre nos objectifs tout au long du projet et d'expérimenter des méthodes que nous retrouverons sûrement dans notre vie professionnelle.

# IV. Analyse

## 1. Analyse UML

Avant de démarrer l'écriture de code, nous avons essayé de décrire le plus possible notre application à l'aide de diagrammes UML. Le langage UML est un langage de modélisation graphique à base de pictogrammes conçu comme une méthode normalisée de visualisation dans les domaines du développement logiciel et en conception orientée objet. Celui-ci nous a permis de décrire la partie métier de notre application. Nous avons aussi essayé de décrire la partie Front-End* de notre application grâce à des maquettes ou esquisses qui sont décrites dans le développement de la partie Front-End*. C'est dans cet élan que nous avons voulu décrire l'architecture de notre application via différents diagrammes.

Nous avons déjà vu plus tôt un diagramme de cas d'utilisation nous permettant de voir rapidement quelles fonctionnalités devra contenir l'application.

Nous pensons que le diagramme permettant le plus d'analyser notre projet est un diagramme de classe, c'est pourquoi nous avons tenu à tracer quelques esquisses de celui-ci lors du développement de chaque partie et de le compléter tout au long du projet.

Vous pouvez voir ici un extrait du diagramme de classe de notre application représentant le paquetage de persistance locale de l'application :

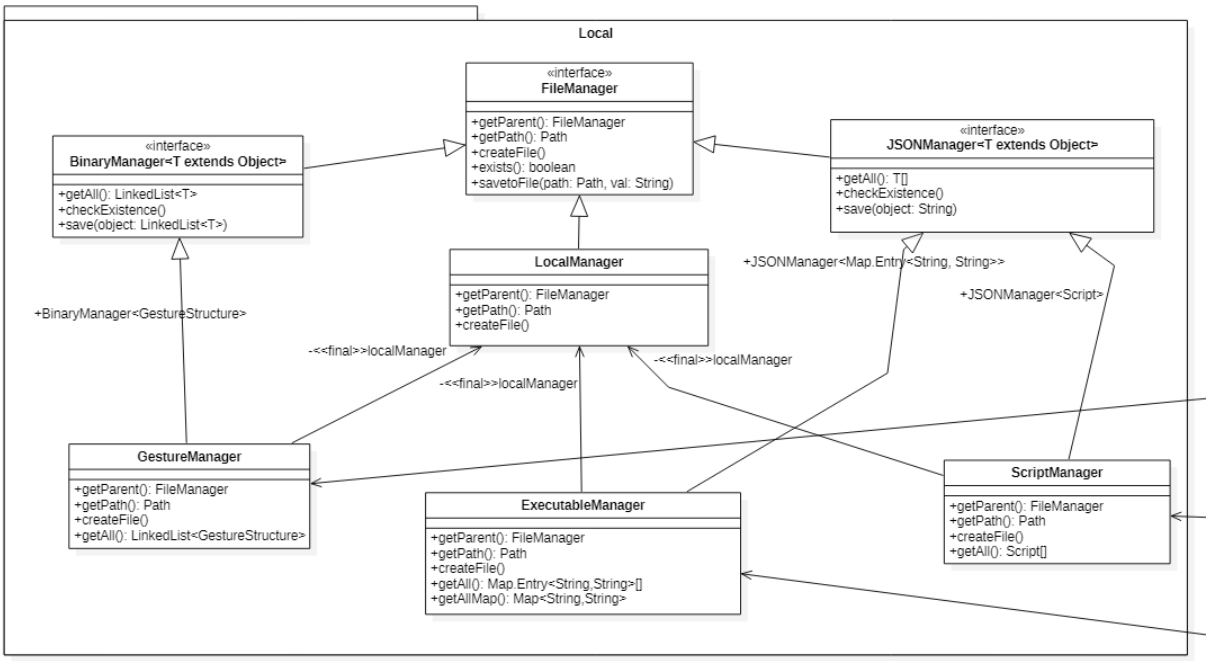



*Figure 3 : Extrait du diagramme de classe, paquetage de la persistance locale.*

Même si la définition du diagramme de classe nous a posé des problèmes au début, cela nous a permis de nous interroger sur la façon dont nous voulions organiser notre architecture. De voir les facettes que nous voulions mettre en place pour permettre l'extensibilité du code. On remarque par exemple, dans ce diagramme de classe, la présence d'une classe *FileManager* qui

gère les fichiers pour la persistance locale. Celle-ci peut être héritée et permet aux autres classes de spécialiser plus facilement les classes filles. En effet, dans le cas où nous voudrions sauvegarder nos données d'une manière différente, nous n'aurions qu'à créer une classe qui héritera alors de la classe mère dans la laquelle nous n'aurions qu'à réécrire les méthodes afin qu'elles correspondent à notre nouvelle façon de sérialiser.

On remarque que la réalisation de diagrammes UML nous a permis d'avoir une vision plus claire sur ce que nous voulions faire et de démarrer rapidement le processus de réflexion sur la manière d'implémenter nos classes afin de garantir un développement plus propre et l'extensibilité que nous recherchions. Nous retrouverons également plus tard un diagramme de séquence nous permettant de décrire le déroulement d'une fonction complexe.

## 2. Choix des technologies

Nous avons donc dû faire de nombreux choix quant aux langages et quant aux « Framework* » ou ensemble cohérent de composants logiciels structurels. Ceux-ci servant à créer les fondations ainsi que les grandes lignes de tout ou partie d'un logiciel. Notre application s'ouvrant à de multiples types de programmation, nous avons donc eu de nombreuses réflexions.

L'application bureau fut développée grâce à un « stack » ou pile de technologies de JavaScript nommé TypeScript (apportant en l'occurrence un typage et une plus grande simplicité pour des projets avec de multiples développeurs). Afin de porter ce langage vers une application bureau, nous avons utilisé le Framework* nommé Electron. Celui-ci permettant de porter un stack web vers un environnement bureau. Pour le Framework* permettant de réaliser l'interface utilisateur, nous avons décidé d'utiliser le très connu ReactJS développé par Facebook afin d'offrir une simplification et surtout l'apprentissage d'une nouvelle technologie inconnue pour la majorité d'entre nous.

La CLI, ou interface en ligne de commande, a été développée en Python afin de profiter de sa simplicité et sa flexibilité lors de l'implémentation. En effet, Python offre une syntaxe très simple et une écriture rapide. De plus, à l'aide du module « Rich », nous réussissons à obtenir un résultat stylisé des données.

Le Back-End* fut lui aussi sujet à une réflexion importante quant aux choix des technologies. En prenant compte que nous voulions que notre application puisse interagir avec le SDK* Leap Motion* 2.3 (afin de permettre l'utilisation du SDK* sur plateformes Linux et Windows), nous avions le choix parmi de nombreux langages. Ainsi, nous avons conclu que l'utilisation de Java serait une bonne décision. En effet, la masse de travail étant tout de même importante, l'utilisation d'un langage que nous connaissions nous semblait être appropriée. De plus, Java offre des Framework* intéressants pour les API REST* avec le très connu Spring ou aussi EJML nous simplifiant les calculs matriciels. Les connaissances supplémentaires découleront des parties spécialisées comme le développement d'une API REST* ou d'une gestion de données avec une base données NoSQL.

Pour finir, les scripts que l'on peut associer à notre application peuvent être de tout type. Il suffit d'ajouter un exécutable et de l'associer à son type MIME (qui est un identifiant de format de données) pour pouvoir interpréter le fichier. Par défaut, plusieurs langages et exécutables sont présents telle que le Python, Javascript (par Node), Lua, Ruby et PHP qui sont des langages interprétés et qui peuvent donc servir de base aux scripts.

Nous avons choisi de baser la sélection de nos langages par rapport aux compétences que nous voulions acquérir lors de la réalisation de ce projet. Celle-ci s'est calée sur notre envie de rendre cette application portable et extensible, d'où, par exemple, la sélection d'un langage permettant de faire interface bureau ou bien web.

## 3. Vision générale de l'architecture

Notre application se découpe en plusieurs éléments bien distincts, et nous avons tenté de scinder les différentes instances afin d'obtenir une application la plus modulable possible. Malgré tout, l'ensemble n'est pas encore parfaitement modulable comme nous le souhaitions. En effet, lors du processus de développement, nous avons mis un point d'honneur à insister sur une démarcation forte entre le Front-End*, le Back-End* et la base de données. Revenons sur les utilités de chaque sous-ensemble.

### a. Front-End*

Nous retrouvons dans cette partie, toutes les méthodes d'accès et d'interaction avec le Back-End*, parmi lesquelles nous pouvons noter la CLI*, l'application bureau ou encore les requêtes HTTP*. Chacun de ces outils proposent la modification de l'adresse de l'API REST* afin de permettre d'utiliser celle qui vous convient, sur la machine de votre choix, etc... La modularité du Front-End* est donc complète et semblable à nos attentes.

### b. Base de données

Pour notre base de données, nous utilisons MongoDB qui est accessible tout simplement en précisant l'adresse d'accès. Afin d'également scinder cette partie et d'offrir la plus grande modularité possible.

### c. Back-End*

Notre Back-End* va venir faire le lien entre ces outils qui sont le Front (API REST*) et la base de données. Mais aussi venir faire la communication avec le Leap Motion* et la logique (calcul matriciel, etc…). Comme on peut le noter, c'est beaucoup d'actions pour notre métier et nous avions dans l'optique de scinder ce cœur logique en deux parties distinctes :

- L'interaction avec le Leap Motion* afin de simplifier au maximum les demandes faites à cet élément.

- La logique, la gestion des fichiers locaux et la gestion de la base de données.

Les apports de cette architecture sont importants. En effet, le Leap Motion* pouvait être mis sur un ordinateur peu puissant et donc avoir un processus peu coûteux. Car il se déchargerait de tous ces calculs et de sa gestion locale sur un autre ordinateur plus puissant.

Nous aurions donc concentré, d'un côté, la gestion des informations du Leap Motion* pour en tirer ce qui était vraiment utile et de l'autre côté, avoir un cœur capable de comprendre et de rendre ces données vivantes et utiles.

Voici une représentation de cette architecture décrite auparavant :

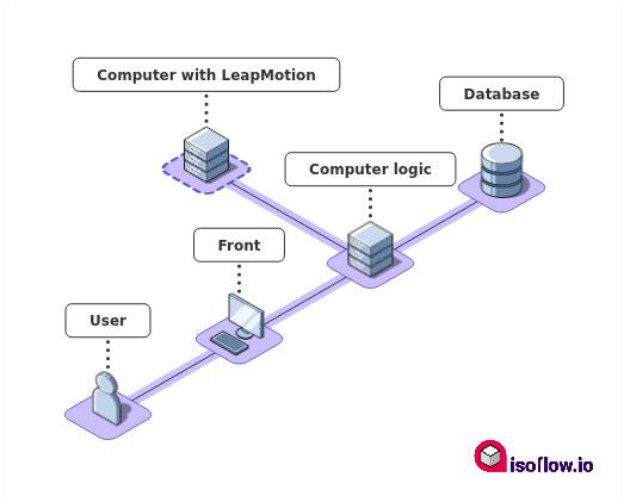



*Figure 4 : Représentation architecture théorique*

Pour des raisons de temps et de masse de travail, nous n'avons malheureusement pas abouti à cette scission pourtant fortement bénéfique. Le "Cœur" du projet est donc toujours en un seul gros bloc.

Voici une représentation de l'architecture actuelle du projet :

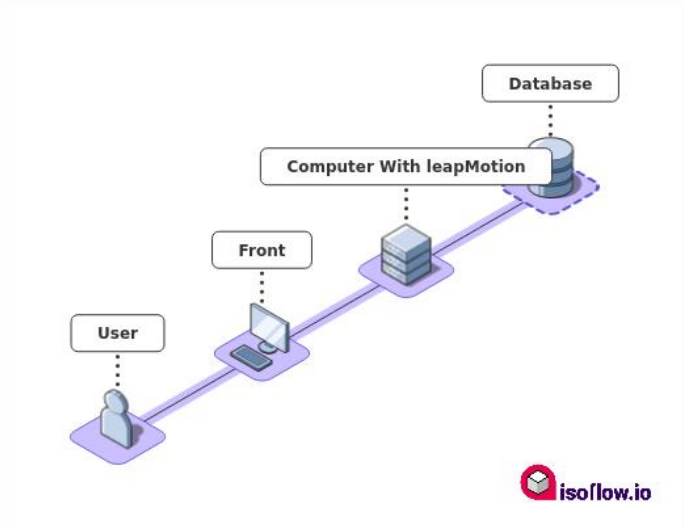



*Figure 5 : Représentation architecture réelle*

# V. Réalisation de notre application

## 1. Front-End*

### a. Utilité

Notre projet est avant tout centré sur la reconnaissance de mouvement et sur l'attachement d'un script à un mouvement. Cependant, nous souhaitions apporter une approche simple aux potentiels utilisateurs. Pour cela, nous avons décidé de concevoir deux environnements orientées Front-End* : une application bureau et un CLI. Ces deux options apportent l'une et l'autre des avantages notables et des contraintes.

### b. Application bureau

- Introduction

L'application bureau a commencé son développement dans le même temps que l'application Back-End* pour la réalisation du design, des différents Wireframe* ou encore de l'établissement des personas et autre contexte.

Par la suite, l'équipe dédiée au Front-End* a commencé le développement de l'application bureau en utilisant des technologies telle que Electron afin d'utiliser un langage et des Framework* Web pour créer l'application (Typescript, React).

Cette décision fut prise avant tout, dans une optique d'apprendre et de découvrir des technologies non encore étudiées, récentes, et de plus en plus courantes dans le milieu du développement Web. Nous pouvons aussi noter leur portabilité et le peu de « refactor* » ou réusinage de code à faire en cas de développement d'une application mobile ou encore d'une interface web pur.

La gestion de projet a été pensée de telle sorte que lorsque l'API REST* serait utilisable, alors le développement des vues serait terminé. Ainsi, l'étape finale serait leurs liaisons.

Depuis cette application bureau, il est possible d'enregistrer une adresse sur laquelle est hébergée l'API REST* et ainsi interagir avec elle.

- Conception des vues

Après avoir fait une étude des fonctionnalités de notre application, nous avons pu construire visuellement l'application grâce à des sketchs, maquettes ou encore Wireframes*. A chaque élaboration d'une vue, le client représenté par M. Provot, nous communiquait des voies d'améliorations à faire. Tel que des problèmes d'ergonomies ou de visuels. Tout au long du processus de création des vues, nous avons été en lien avec le client pour créer une interface correspondante à sa demande et à l'ergonomie voulue.

La réalisation des vues a commencé très tôt dans le projet pour pouvoir finaliser l'interaction entre elles et le Back-End* le plus tôt possible. Pour pouvoir réduire le temps passé au

maximum, nous nous sommes organisés de la sorte. Tout d'abord, une réalisation minimaliste des vues avec uniquement les objets graphiques positionnées au bon endroit sans faire attention à la couleur ou à la taille de ceux-ci. Les vues ont été réalisées dans l'ordre suivant : page scripts, page mes scripts, page de connexion et d'enregistrement, page d'accueil, page de réglages, page gestures et page exécutable. Un aperçu de chacune de ces vues est disponible en annexe.

- Navigation

Dans un deuxième temps, nous nous sommes penchés sur la réalisation de la navigation dans l'application. Celle-ci se fait par le biais d'un menu déroulable grâce à l'icône la plus en haut à gauche de l'écran. Ce menu contient autant de partie qu'il y a de page navigable. Seuls les pages de connexion et d'enregistrement ne se trouvent pas dans le menu mais directement en haut à gauche dans l'entête de toutes les pages. Ce menu est disponible sur toutes les vues de l'application pour une navigation rapide, sauf sur la page de connexion et d'enregistrement.

- Chartes graphiques, couleurs, tailles

Ensuite, après le travail de définition fait précédemment, nous avions besoin de définir un esthétique à notre application. Nous avons donc mis en place la charte graphique de l'application, ainsi que la taille des éléments, la forme, les couleurs pour se rapprocher de l'aspect final de l'interface. Globalement, nous avons conçu une application épurée, qui n'est pas surchargée d'éléments, car le but de celle-ci, est d'être la plus pratique et simple d'utilisation. Par exemple, en très peu de clic, l'utilisateur peut retrouver ces scripts, les désactiver ou les activer. La charte graphique suivie dans ce projet est le thème Polar Night se trouvant sur le site suivant : https://www.nordtheme.com/.

- Détails des vues

➢ Page d'accueil

La page d'accueil est la première vue que l'utilisateur aperçoit après le lancement de l'application. Celle-ci n'est composée que de deux éléments graphiques, un retour visuel de la caméra du Leap Motion* et un indicateur permettant de savoir les statuts du Leap Motion*. S'il est connecté ou non à l'API. Lorsque que celui-ci n'est pas connecté, l'indicateur nous informe en passant du bleu au rouge, et une image d'erreur apparaît.

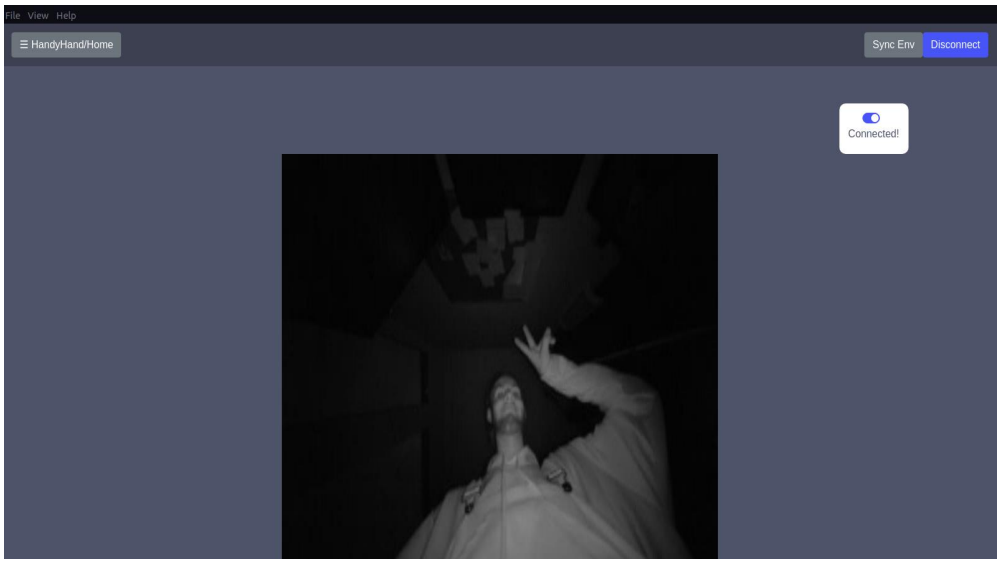



*Figure 6 : Exemple affichage accueil avec Leap Motion* connecté*

- Page scripts

La page Scripts sert de bibliothèque à l'utilisateur, celle-ci recense les scripts en base de données. Ceux qui ont été conçu par la communauté afin d'être partagés. Dans cette vue, deux modes sont disponibles. Le premier affiche les scripts en listes et le deuxième en grilles. L'utilisateur modifie le visuel selon sa préférence car aucune donnée n'est perdue en le changeant. Concernant les scripts, nous retrouvons visuellement plusieurs informations. Le titre du script, sa description pour détailler les fonctionnalités, son état (s'il est activé ou non) et pour finir le geste auquel il est relié.

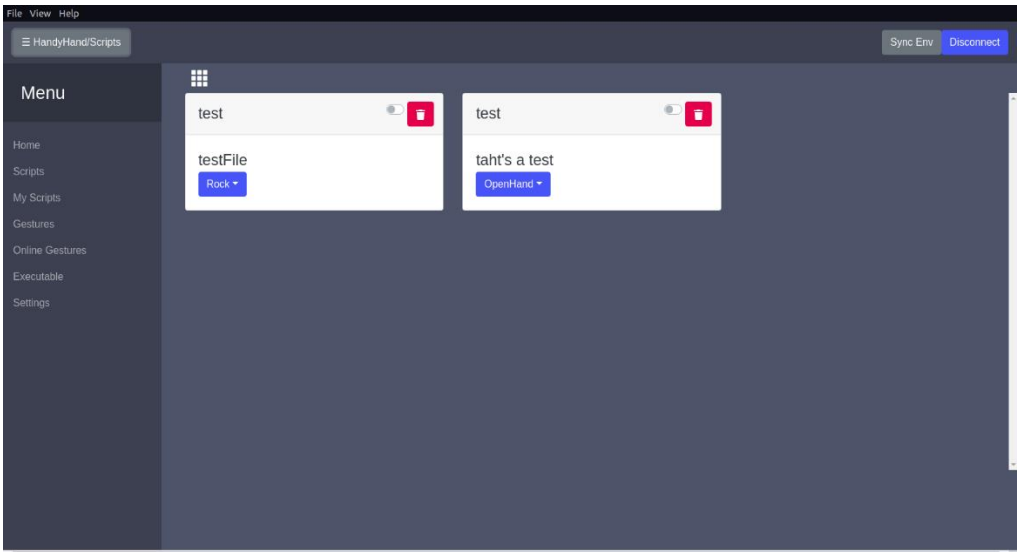



*Figure 7 : Exemple affichage des scripts en icône*

- Page mes scripts

La page mes scripts est semblable en tout point à la page scripts. Les scripts affichés sont les scripts que l'utilisateur possède en local. Les scripts possèdent les mêmes informations que dans la page script, un titre, une description… Un bouton permettant de rajouter un script à l'application se trouve en dessous de l'entête en haut à gauche. Celui-ci permet de se diriger vers une nouvelle fenêtre où l'on peut retrouver les informations qui suivent. Un formulaire à remplir où est demandé une description ainsi qu'un un chemin d'accès au script fait par l'utilisateur.

On retrouve un bouton permettant de valider le formulaire. En cliquant sur celui-ci, l'utilisateur se retrouve sur la page mes scripts avec son nouveau script ajouté.

- Page gestures

Les gestes sont, comme nous avons pu le voir auparavant, une partie centrale de notre application. Deux pages leurs sont dédiés, gestures et my gestures, ils profitent de la même découpe que les scripts (version en ligne et version locale). Au niveau du design, il est identique aux autres pages dans un souci d'unicité.

Vous pouvez enregistrer un nouveau geste en lui donnant des informations tel qu'un nom, une description, s'il contient une ou deux mains, si la distance entre les mains et doigts importe et pour finir vous pouvez valider. Alors, un compte à rebours apparaitra, et lorsque celui-ci sera fini, la position de vos mains au-dessus du Leap Motion* sera sauvegardée. Vous pouvez à présent relier ce nouveau geste à un script dans les pages dédiées au script.

- Page réglages

Une page de réglages est aussi présente et vous permet de gérer les paramètres généraux de l'application. Actuellement, cette page permet simplement de modifier l'adresse sur laquelle l'API doit être appelée (étant donné que c'est le seul paramètre général que nous ayons à traiter).

- Page exécutables

Afin de contrôler les différents exécutables qui prendront en charge l'exécution des scripts, nous avons mis en place une page dédiée aux exécutables. Celle-ci vous fait apparaître l'ensemble des exécutables disponibles avec une association d'un type MIME et d'un chemin d'accès. Vous pouvez modifier le chemin pour permettre à l'application de lancer vos scripts avec exécutable souhaité.

- Page connexion et enregistrement

À l'aide de la barre supérieure, on a accès aux pages de connexion et d'enregistrement. Pour créer un nouveau compte, vous devrez indiquer un courriel et un mot de passe ainsi qu'une validation de courriel.

Par la suite, vous pouvez vous connecter à l'aide de vos informations et vous serez automatiquement rediriger lorsque vous serez connecté.

### c. Command Line Interface ou interface en ligne de commande

Le but du CLI* est multiple. En effet, il est plus simple pour des ordinateurs embarqués d'interagir avec un CLI* plutôt qu'une interface graphique (la plupart n'ont pas d'écran ou de moteur graphique). De plus, ce genre de gestion est beaucoup moins coûteux pour l'outil qui l'utilise.

Il est aussi un outil plus rapide et efficace pour une action donnée. C'est donc un outil très intéressant pour un développeur qui veut tester des fonctions ou encore un utilisateur n'ayant pas peur du terminal.

Pour implémenter la CLI*, nous avons choisi l'utilisation d'un langage interprété très populaire : Python. Le langage Python nous a permis d'utiliser des paquets comme requests pour faire les requêtes HTTP*, Rich pour rendre un rendu visuel plaisant et simplifier la lisibilité, ou encore Magic pour la détection des types MIME.

Les options disponibles sont simples et exactement les mêmes que pour le client bureau. De plus, on peut noter la présence d'une « documentation » intégrée et détaillée pour chacune des fonctions ce qui permet une compréhension simple et rapide de toutes les options disponibles.

### d. Requête HTTP*

Évidemment tout outil du Front-End* doit être alimenté par une source de données. Nous avons, comme cité plusieurs fois auparavant, fait usage d'une API REST* développée en Java.

Il est évidemment possible de requêter l'API REST* directement à l'aide d'outils tel que Curl. Il est assez simple pour quiconque lisant les références de l'API REST* de formuler ces appels. Sachant que nous avons détaillé pour chaque requête, la méthode mais aussi les paramètres ou encore les entêtes demandés.

L'API REST* n'a pas été pensée dans cette optique, mais le fait est, que c'est totalement possible. Nous n'y émettons aucun blocage, nous avons nous même beaucoup utilisé ces méthodes durant le processus de développement de l'application.

# 2. Back-End*

## a. Reconnaissance

Notre projet se base sur le Leap Motion* qui nous renvoie certaines informations sur les mains qu'il repère. Cependant, il ne nous permet pas de faire une reconnaissance de gestes. En effet, les principales informations qu'il renvoie sont sur les positions et orientations des mains dans l'espace. Ainsi, pour mener à bien notre projet, il a fallu créer nous même une reconnaissance de gestes.

Pour nous, c'est cette partie qui a été la plus importante. En effet, la reconnaissance d'un geste via le Leap Motion* figure parmi la problématique de notre projet. C'est pourquoi nous avons choisi de la détailler plus que le reste de notre application.

La première tentative fut de calculer différentes informations sur la main et ses doigts à partir des différentes positions que l'on recevait du Leap Motion*. Cependant, celle-ci était beaucoup trop contraignante, et nous avons alors décidé de changer de méthode. Cette deuxième tentative se base sur la comparaison des positions que renvoie l'outil sur les mains.

Nous allons en premier lieu, vous présenter les informations utiles sur le Leap Motion*. Nous enchainerons ensuite sur la première méthode que nous avons utilisé pour notre projet. Enfin, nous terminerons avec la méthode de comparaison finale.

Le Leap Motion* nous permet de récupérer différentes coordonnées des mains, doigts et même plus. Pour que ceci soit possible, il utilise un système de coordonnées cartésiennes, avec les distances en millimètre. C'est un repère orthonormé en trois dimensions, dont son origine se trouve centrée sur le dessus du Leap Motion*, comme le montre cette illustration :

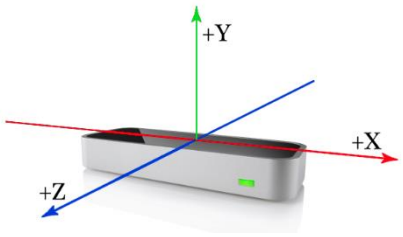



*Figure 8 : Système de coordonnées du Leap Motion**

Une main est constituée de doigts, qui sont eux-mêmes constitués d'os. En soit, une main est composée de 27 os, cependant, les carpes, c'est-à-dire ceux situés au début de la main, proche du poignet, ne sont pas pris en compte par le Leap Motion*. En effet, celui-ci nous renvoit des informations sur quatre types d'os présents dans les doigts :

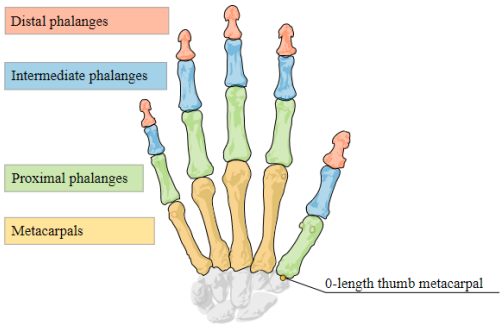



*Figure 9 : Schéma des os d'une main*

Pour être plus précis, le pouce possède uniquement trois os, cependant, pour une histoire de simplicité du code, il est considéré avec quatre os, dont un, le métacarpe, avec une longueur de 0. Ainsi, pour résumer, le Leap Motion* considère qu'une main possède cinq doigts, et que chacun d'eux contient quatre os, étant la phalange distale, la phalange intermédiaire, la phalange proximale et le métacarpe. De plus, pour chacun des os, il nous renvoie la position de son centre ainsi que celle de ses extrémités.

Au-delà de ces os, il renvoie également la position du centre de la main, qui est définie comme étant le milieu de la paume de la main. De plus, à partir de ce même point, il va nous donner la direction et l'orientation de la main. Pour ceci, il utilise un système de vecteurs : celui de la direction, ainsi que le vecteur normal de la paume définissant ainsi l'orientation.

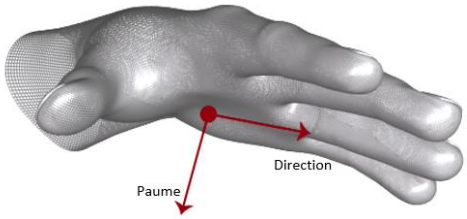



*Figure 10 : Vecteurs de la paume et de la direction de la main*

On remarque ainsi que le Leap Motion* renvoie des informations utiles pour différentes utilisations.

Le Leap Motion* est assez complet, cependant, il ne donne pas non plus toutes les informations dont nous avons besoin. En effet, dans notre quête de la reconnaissance de gestes, nous devons ajouter des apports personnels. Ainsi, au départ, pour permettre de reconnaître un geste codé, nous avons fait en sorte de créer des informations complémentaires sur la main. Tout d'abord, nous avions trouvé le pourcentage de courbure pour chaque doigt en utilisant des distances entre certaines positions, alliées à un produit en croix se servant de données que nous avions trouvé lors de tests. Ensuite, on a utilisé des calculs similaires pour trouver d'autres informations, tel que lorsque des doigts se touchent, ou que deux mains sont collées. Par la

suite, nous avons utilisé une nouvelle manière pour calculer le pourcentage de courbure des doigts. Nous avons utilisé les différentes coordonnées des os afin de trouver des vecteurs et calculer l'angle de ceux-ci. Bien qu'elle nous ait été utile pour faire différents tests, la reconnaissance précédente n'est pas optimale. En effet, celle-ci est contraignante pour différentes raisons. Tout d'abord, pour que le Leap Motion* reconnaisse un geste dit complexe (avec par exemple deux mains retournées côte à côte, et dont certains doigts sont pliés et d'autres non), il va falloir s'y reprendre à plusieurs fois, voir même faire une certaine séquence de mouvements pour bien que le Leap Motion* remarque le geste correctement (par exemple montrer ses deux mains côté face, plier ses doigts, tourner ses mains et enfin les coller). Ceci est alors complexe et non intuitif pour tout utilisateur. Ensuite, si nous voulons ajouter des méthodes de reconnaissances, telles que la détection de l'espace entre deux doigts, il faudrait réfléchir à comment faire pour coder, et enfin partager le code dans une mise à jour. Cependant ceci, bien que réalisable, n'est pas la meilleure, ni la plus simple des solutions pour l'utilisateur. Enfin, il faut coder chaque geste « à la main ». Effectivement, nous pourrions proposer une interface utilisateur, où chacun pourrait créer son propre geste (en demandant que tel ou tel doigt soit plié ou non), mais ceci ne serait pas intuitif à tout le monde, et d'ailleurs très limitant.

Tout ceci nous amène à devoir réfléchir à une meilleure manière de reconnaissance. Celle-ci devra pouvoir laisser l'utilisateur créer son geste personnalisé, qui sera également conforme à la taille de sa main et facile à utiliser. Étant donné que le Leap Motion* nous renvoie des coordonnées de différentes parties de la main, on pourrait se dire qu'il est possible de toutes les comparer avec un autre geste. Les coordonnées sont des positions dans l'espace, ce qui veut dire qu'il faudrait que le geste soit exactement au même endroit, ce qui est très contraignant. Nous devons alors trouver un moyen qui puisse faire en sorte que la position ne rentre pas en compte. Ensuite, ceci inclurait uniquement une certaine taille de main. Un enfant ne pourrait pas faire le geste d'un adulte, et vice-versa. Enfin, la rotation de la main est prise en compte dans les coordonnées, tandis que nous voulons que le geste soit reconnu quel que soit la direction de la paume de la main (à condition que le Leap Motion* puisse détecter correctement la main complète). Ces trois points peuvent se résoudre respectueusement avec la translation, la mise à l'échelle, et la rotation.

La translation permet, pour un objet géométrique, de déplacer tous ses points de la même distance, avec la même direction et le même sens. Dans notre cas, ceci signifie changer la position initiale de la main vers un autre endroit. Nous voulons centrer la main vers un certain point quel que soit la position de base que l'utilisateur ait donnée. Pour ceci, nous allons définir le centre de la paume de la main comme étant le centre du repère. Pour ce faire, il faut tout simplement soustraire à chaque coordonnée de la main les valeurs sur chaque axe du centre de la paume de la main :

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} - \begin{pmatrix} xp \\ yp \\ zp \end{pmatrix} = \begin{pmatrix} x - xp \\ y - yp \\ z - zp \end{pmatrix}$$

Avec $\begin{pmatrix} x \\ y \\ z \end{pmatrix}$ les coordonnées d'un vecteur quelconque, étant ici l'un de ceux de la main, et $\begin{pmatrix} xp \\ yp \\ zp \end{pmatrix}$ les coordonnées du centre de la paume de la main.

En faisant cela, nous recentrons chaque main, quelle que soit sa position de base, vers le centre du repère. Comme on vient de le voir, on peut le faire en utilisant un vecteur que l'on soustrait à chaque coordonnée de la main. Cependant, dans notre cas, nous n'allons pas utiliser cette technique, mais plutôt des matrices. Pour ce faire, nous allons rendre le vecteur homogène. Ceci signifie qu'il possède une ligne en plus. Faire ceci permet en général d'avoir de biens meilleures performances dans les logiciels informatiques. Le principe est que pour avoir le vrai vecteur, il faudra diviser chacun de ses éléments, par le nouvel élément présent à la fin. Ensuite, nous utilisons une matrice carrée que l'on multipliera avec le vecteur pour trouver de nouvelles coordonnées. Ceci nous sera utile pour les futurs calculs, lorsqu'on voudra multiplier plusieurs matrices entre elles, si elles sont toutes carrées et de même taille, nous n'aurons pas de problème de sens de produit matriciel. Par conséquent, nous utilisons la base d'une matrice identité de taille quatre (étant donné que notre vecteur est homogène, et donc avec quatre lignes). En modifiant la matrice, on peut trouver cette équation :

$$\begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} 1*x+0*y+0*z+t_x*1 \\ 0*x+1*y+0*z+t_y*1 \\ 0*x+0*y+1*z+t_z*1 \\ 0*x+0*y+0*z+1*1 \end{pmatrix} = \begin{pmatrix} x+t_x \\ y+t_y \\ z+t_z \\ 1 \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix}$$

Ainsi, on peut facilement faire notre translation en changeant les valeurs de $t_x$, $t_y$ et $t_z$, avec par exemple des valeurs négatives. Par conséquent, dans notre cas nous mettrons les valeurs du vecteur position du centre de la paume de la main, multipliées par $-1$ pour bien qu'elles soient soustraites et non additionnées.

Par la suite, nous voulons faire une mise à l'échelle. Ceci revient à multiplier chaque valeur de chaque vecteur par un même nombre. Pour faire cela, on reprend tout simplement notre matrice identité que l'on multiplie par notre valeur (ou on remplace les 1 de la diagonale par notre valeur définie). Et ensuite on multiplie nos vecteurs par la matrice que nous avons obtenue :

$$\begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} 2*x+0*y+0*z+0*1 \\ 0*x+2*y+0*z+0*1 \\ 0*x+0*y+2*z+0*1 \\ 0*x+0*y+0*z+2*1 \end{pmatrix} = \begin{pmatrix} 2x \\ 2y \\ 2z \\ 2 \end{pmatrix}$$

Cela revient pour notre cas à :

$$\begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} 2x' \\ 2y' \\ 2z' \\ 2 \end{pmatrix}$$

L'ordre est important, il faut bien multiplier la matrice de mise à l'échelle par la gauche. Pour optimiser nos calculs, nous pouvons calculer le produit matriciel des deux matrices afin de ne pas le faire à chaque fois :

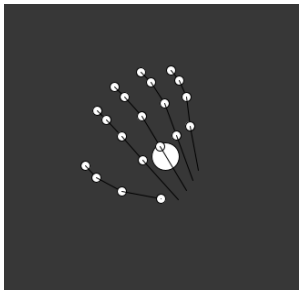
$$\begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 0 & 0 & 2t_x \\ 0 & 2 & 0 & 2t_y \\ 0 & 0 & 2 & 2t_z \\ 0 & 0 & 0 & 2 \end{pmatrix}$$

Nous voulons que chaque main, quel que soit sa taille de base, ait la même proportion. Pour ce faire, nous allons définir une taille souhaitée à la fin de la mise à l'échelle. Nous prendrons la taille d'un os comme outil de comparaison. L'os métacarpien du majeur est le plus grand de la main, ce qui permettra d'avoir plus de marge qu'avec la phalange distale par exemple. Ainsi, notre but est qu'à chaque fois que la mise à l'échelle est faite, l'os choisi aura une taille que l'on aura décidé au préalable. Dans notre cas, nous allons dire 75 (il n'y a pas de réelle raison, mais c'est une taille qui ne diffère pas beaucoup de nos mains). Nous trouvons grâce à un produit en croix l'équation suivante :

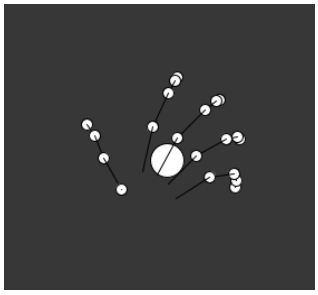
$$x = \frac{75}{te}$$

Avec $te$ la taille de l'os en entrée et $x$ la valeur recherchée pour obtenir 75 à partir de $te$.

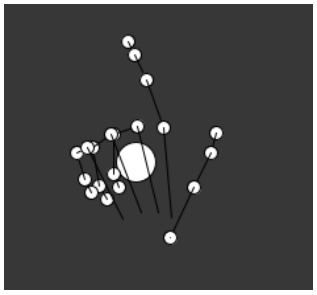
Ainsi, ceci nous donne, avec la translation, notre main centrée et mise à l'échelle. Cependant, si nous l'affichons, elle tourne toujours :



La paume de la main est dirigée vers le bas gauche



La main est penchée en avant et légèrement inclinée vers la droite



La paume de la main est dirigée vers le ciel gauche

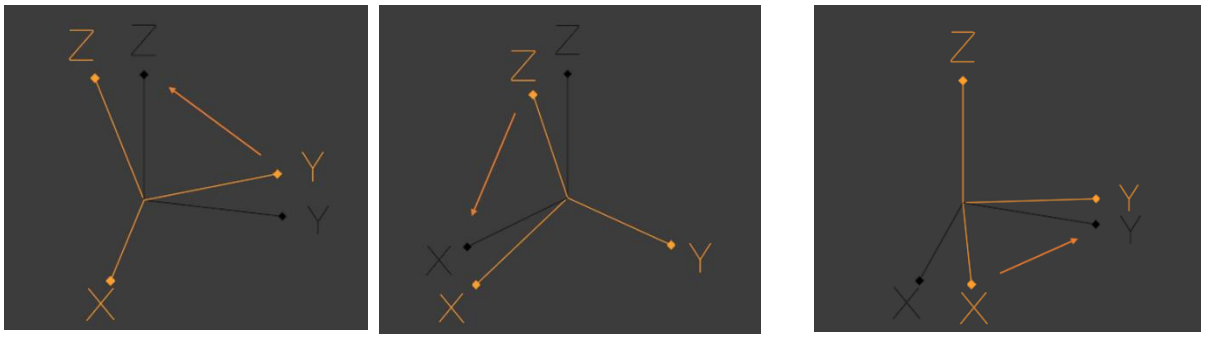
*Figure 11 : Représentations squelettiques de mains ayant toujours les rotations*

Il existe des matrices dites de rotation, qui permettent, comme leur nom l'indique, de faire une rotation d'un vecteur. Cette rotation se fait suivant un axe, qui peut être un vecteur choisi suivant la matrice. Voici les matrices de rotations respectivement autour des axes x, y et z :

$$R_x(\theta) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos\theta & -sin\theta \\ 0 & sin\theta & cos\theta \end{pmatrix}, \quad R_y(\theta) = \begin{pmatrix} cos\theta & 0 & sin\theta \\ 0 & 1 & 0 \\ -sin\theta & 0 & cos\theta \end{pmatrix}, \quad R_z(\theta) = \begin{pmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Avec $\theta$ l'angle de rotation en radiant.

Les rotations opèrent ainsi :




*$R_x$ tourne l'axe $y$ vers l'axe $z$* *$R_y$ tourne l'axe $z$ vers l'axe $x$* *$R_z$ tourne l'axe $x$ vers l'axe $y$*

*Figure 12 : Représentations de rotations par rapport aux axes d'un repère canonique*

On peut remarquer que ces trois matrices font tourner les vecteurs suivant un certain axe, qui ne bouge pas. Nous pouvons remplacer cet axe par un vecteur unitaire $\vec{u}\begin{pmatrix} u_x \\ u_y \\ u_z \end{pmatrix}$. Pour cela, nous utilisons cette formule :

$$R = \begin{pmatrix} u_x^2(1-c)+c & u_x u_y(1-c)-u_z s & u_x u_z(1-c)+u_y s \\ u_x u_y(1-c)+u_z s & u_y^2(1-c)+c & u_y u_z(1-c)-u_x s \\ u_x u_z(1-c)-u_y s & u_y u_z(1-c)+u_x s & u_z^2(1-c)+c \end{pmatrix}$$

Avec $c = cos\theta$ et $s = sin\theta$.

Le Leap Motion*, de son côté, nous renvoie deux vecteurs unitaires et trois angles (en radians) de rotation pour chaque main. Les vecteurs sont ceux de la direction où pointe la main (donc vers les doigts), et celle où pointe la paume de la main.

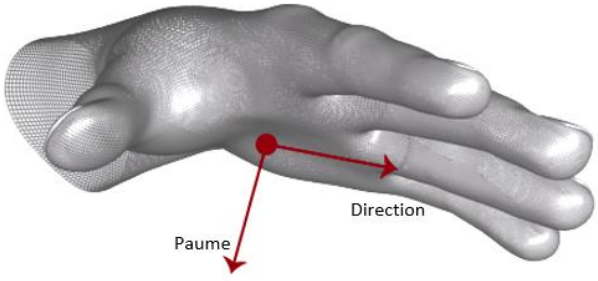



*Figure 10 : Vecteurs de la paume et de la direction de la main*

Les rotations correspondent au Yaw, Pitch et Roll (en anglais), soit respectivement la direction, l'aileron et la profondeur. Voici une image pour mieux comprendre :

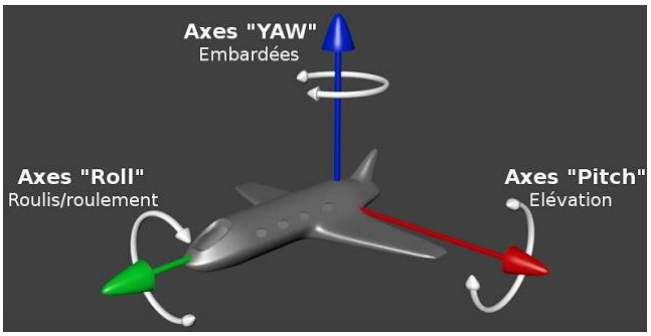



*Figure 13 : Illustration des axes Yaw, Pitch, Roll*

Dans notre cas de la main, ceci donnerait plutôt :

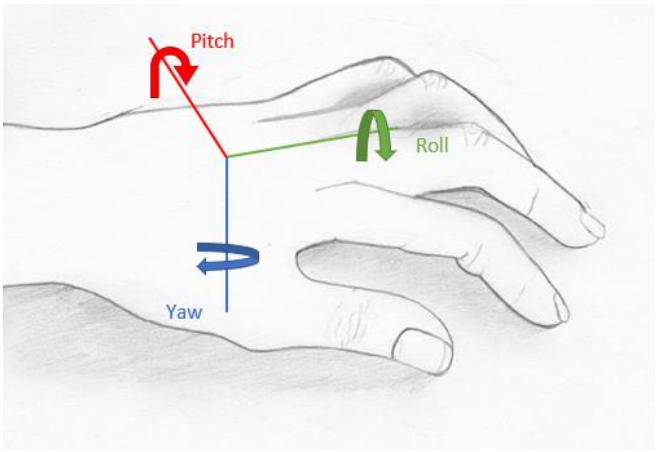



*Figure 14 : Illustration des axes Yaw, Pitch, Roll sur une main*

C'est-à-dire que la rotation Yaw (se dirige vers le bas de la main) fait tourner la main tout en ayant le bras qui se déplace. Cette rotation se fait autour de l'axe du vecteur de la paume. La rotation Roll (se dirige vers l'avant de la main) fait lever ou baisser le pouce suivant l'angle. Elle se fait autour du vecteur Direction. Enfin, la dernière est Pitch (se dirige vers la gauche de la main sur l'image), étant quand on penche la main vers l'avant ou l'arrière. Elle correspond à un axe qui ne nous est pas donné par le Leap Motion*. Par la suite nous utiliserons des axes nommés $x$, $y$ et $z$ qui correspondent respectivement aux axes Pitch, Roll et Yaw, où notre perspective est passée en dessous de la main. Ainsi, on pourrait se dire que l'on a simplement à tourner la main avec les valeurs négatives des rotations renvoyées. Cependant, ceci ne va pas fonctionner. En effet, la rotation que l'on applique ne se fera pas par rapport au repère global du Leap Motion*, mais plutôt à celui de la main. Ainsi nous n'arriverons pas à une rotation nulle par rapport au repère du Leap Motion*. Ceci est assez complexe à se représenter lorsque l'on n'est pas habitué à ce genre de calcul. Par conséquent, on va vous l'expliquer en image :

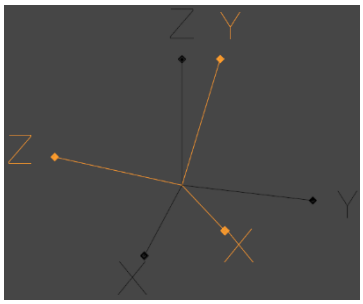



*Figure 15 : Représentation d'un repère tourné de* $70°$ *en* $x$, $-10°$ *en* $y$ *et* $30°$ *en* $z$

On peut voir ici deux repères. Le noir correspond à celui du Leap Motion*, et l'orange à celui de la main. On a tourné l'orange de $70°$ en $x$, $-10°$ en $y$ et $30°$ en $z$, soit comme si le Leap Motion* nous renvoyait ces valeurs pour une main. Maintenant appliquons les rotations négatives sur ce repère, soit $-70°$ en $x$, $10°$ en $y$ et $-30°$ en $z$ :

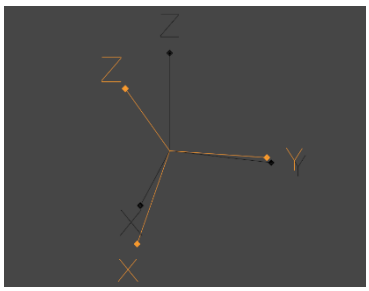



*Figure 16 : Représentation d'un repère tourné de* $70°$ *en* $x$, $-10°$ *en* $y$ *et* $30°$ *en* $z$ *puis de* $-70°$ *en* $x$, $10°$ *en* $y$ *et* $-30°$ *en* $z$

On remarque alors directement que les axes ne sont pas aux mêmes positions, donc cela n'est pas ce que nous voulons.

Lors de notre projet, nous n'avons pas utiliser la bonne manière pour faire en sorte que la rotation soit totalement enlevée. En effet, après différents tests et recherches nous sommes arrivé à un point où la rotation avait presque totalement disparue. Cependant, lors de certains gestes avec un angle de la main assez fort, tel que quand les doigts pointent vers le bas ou le haut, on peut remarquer qu'il reste tout de même des erreurs. Nous pensions que cela venait du fait que le Leap Motion* ne peut pas avoir des données ultra précises et donc que cela venait de lui. Ainsi, partant de cela, nous n'avons pas vu d'inconvénient dans la mesure où, à partir du moment où l'utilisateur se sert correctement du Leap Motion*, il ne devrait pas tomber sur ce genre de cas de figure. Cependant, en rédigeant le rapport, nous avons remarqué que finalement c'est notre méthode utilisée qui ne convient pas. Nous ne pouvons régler ceci par manque de temps. Par conséquent, nous vous la présentons quand même dans la mesure où nous nous en servons, mais celle-ci ne serait pas adapté à d'autres projets. Il est tout de même nécessaire de rajouter que ceci peut être facilement modifié, sans toucher au reste du projet. Pour cela il suffit de changer la manière dont la rotation est remise à « zéro », étant donné que le reste du code est détaché de cette partie.

Tout d'abord, nous tournons la main suivant un vecteur. Nous utilisons celui de la paume, soit l'axe Yaw ($z$ sur les images). Quant à la valeur utilisée pour la rotation, celle-ci sera l'angle de Yaw, étant multipliée par $-1$ si la main est à l'envers. Ceci est déterminé par l'angle Roll. Si sa

valeur absolue est inférieure à $90°$ ($\frac{\pi}{2}$ en radian), soit quand la main ne dépasse pas la position verticale pour rejoindre l'horizontale, alors on multiplie l'angle de Yaw par $-1$, sinon on ne fait rien. Sur notre image, ce sera l'axe $z$, et un angle de $-30°$ :

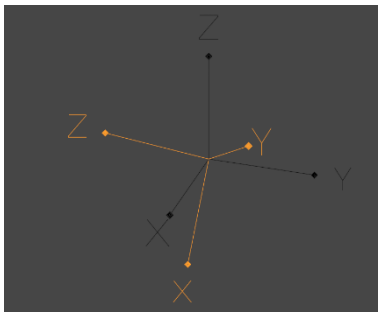



*Figure 17 : Représentation de la première rotation pour revenir dans un même repère*

Le fait d'avoir fait cette rotation, vous vous en douterez, a également fait bouger les rotations précédentes de $x$ et $y$. On se retrouve avec, à peu près $69.2°$ en $x$, $18.2°$ en $y$ et $19.6°$ en $z$. Ensuite, nous utilisons la matrice de rotation dans l'espace global, soit l'espace du Leap Motion*. Ainsi, nous prenons la matrice $R_x$ vue précédemment, que nous appliquons avec une rotation de l'angle de l'axe $x$, soit Pitch, multipliée par $-1$. Cependant, il faut que ce soit le nouvel angle trouvé précédemment, et que la valeur soit en radian. Donc nous prenons ici $69.2° = 1{,}204 radians$. Ceci nous donne alors :

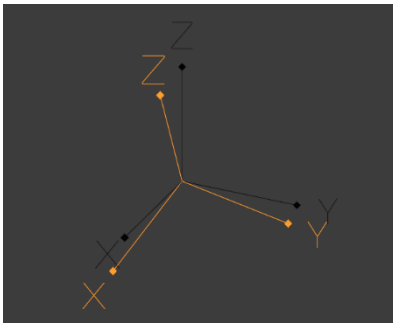



*Figure 18 : Représentation de la seconde rotation pour revenir dans un même repère*

Avec $-5.56°$ en $x$, $24.2°$ en $y$ et $-11.3°$ en $z$.

Enfin, nous faisons la même chose avec l'axe $z$ et l'angle de Roll, soit celui de $y$.

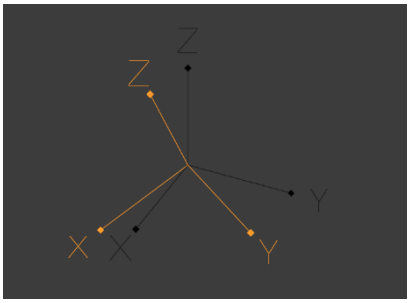


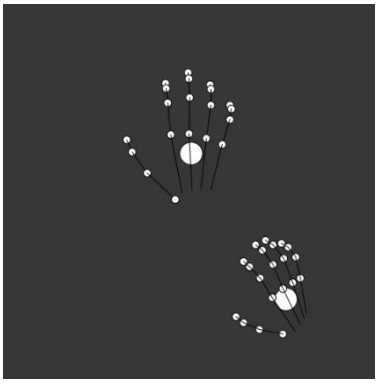

*Figure 19 : Représentation de la troisième rotation pour revenir dans un même repère*

Comme dit précédemment, ce que nous avons ne convient pas forcément partout, mais ceci nous suffit dans notre projet. Maintenant que nous possédons ces trois matrices de rotation, nous pouvons les multiplier entre elles. Pour ce faire, la dernière que l'on a trouvée, sera la première dans la multiplication. En effet, si par exemple nous avons les matrices $A$, $B$ et $C$ avec $A$ la première rotation, $B$ la seconde, et $C$ la troisième, alors nous devons faire $C * B * A$. L'ordre des matrices étant important dans les multiplications matricielles, et notre vecteur se trouvant à droite de la multiplication, nous aurions d'abord $A * \vec{v}$ puis $B * \overrightarrow{v'}$ et enfin $C * \overrightarrow{v''}$, soit $C * B * A * \vec{v}$. D'où l'utilité de faire ceci dans cet ordre. Sur ce même principe, nous réunissons les matrices de mise à l'échelle et de translation, dans l'ordre de la translation, la mise à l'échelle et enfin la rotation. Ce qui nous donne une nouvelle matrice $N$ :

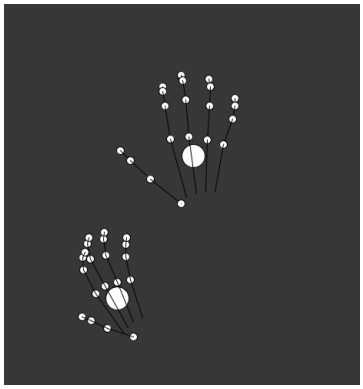
$$N = C * B * A * M * T$$

Avec $M$ la matrice de mise à l'échelle, et $T$ celle de la translation.

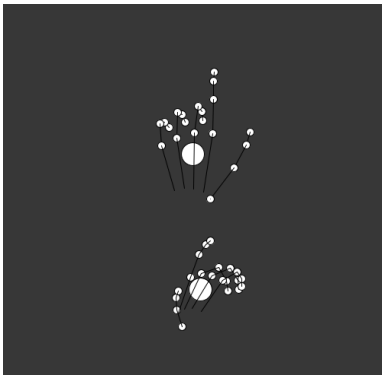
Regardons ce que cela donne lorsqu'on l'utilise sur les mains. On peut voir, sur les images suivantes, la main, qui a était translatée, mise à l'échelle et tournée, en haut. En dessous est ce que nous renvoie le Leap Motion*, c'est-à-dire sans qu'on ne modifie les valeurs des positions de la main.



La main se trouve à droite et est légèrement tournée



La main se trouve à gauche et est légèrement tournée



La main est orientée partiellement avec sa face vers le ciel

*Figure 20 : Représentations squelettiques de mains avant et après l'application de la matrice permettant de revenir à un même repère*

On peut voir sur ces images que la main est bien centrée, et tournée. La mise à l'échelle ne se distingue pas dans la mesure où elle est légère en comparaison de la taille de la main de base, mais elle est bien présente.

La main est orientée avec les doigts vers le haut, et la paume vers l'avant

La main est penchée sur la droite, avec la paume vers l'avant et les doigts dirigés vers la droite

*Figure 21 : Représentations squelettiques de mains avant et après l'application de la matrice permettant de revenir à un même repère et possédant un certain décalage*

Ici on peut remarquer les problèmes expliqués précédemment, qui arrivent quand l'utilisateur essaie quelque chose de complexe, qui ne rentre pas vraiment dans une utilisation réelle du Leap Motion*.

Nous avons alors réussi à faire en sorte que les mains soient remises dans un même repère, ce qui va nous permettre de pouvoir les comparer entre elles. Pour ce faire nous comparons tout simplement chaque coordonnée des deux mains entre elles. Évidemment, on ne peut pas refaire exactement le même geste à chaque fois, il nous faut alors accepter une certaine divergence. Pour ceci, lors de la comparaison, nous faisons en sorte que si l'une des valeurs (donc $x$, $y$ ou $z$) de l'une de coordonnées des mains a un écart trop grand avec la valeur comparée, alors on considère que les deux mains ne font pas le même geste. Nous avons mis cette valeur à $20$, soit $20$mm de divergence possible, dans la mesure où elle permet de reconnaître un geste même quand l'utilisateur ne fait pas exactement le même, tout en évitant qu'il en reconnaisse un autre, s'ils sont assez proche, avec par exemple les gestes de cœur et de rond avec les deux mains.

Pour permettre de stocker les gestes, et donc les coordonnées, nous avons créé différentes classes. Nous possédons *BoneStructure* qui contient les coordonnées des bouts d'un os, et son type. Une méthode *compare*, renvoyant un booléen et prenant en paramètres une autre instance de cette même classe et la divergence que l'on accepte, permet de savoir si les coordonnées des deux os sont similaires. Le principe est simple, si pour l'un des éléments des coordonnées, la valeur absolue de la différence entre les deux os est supérieure à la divergence, alors la méthode renvoie faux, sinon elle renvoie vraie. Évidemment, si les os ne sont pas du même type, la comparaison renvoie faux. Ensuite, nous avons la classe *FingerStructure* pour les doigts. Elle contient son type et également, pour chaque os qu'elle possède, une instance de la classe *BoneStructure* où se trouve les coordonnées de ses os. Elle possède une méthode *compare* qui appellera celles de ses os. Elle renvoie un booléen qui sera faux si l'un des os renvoie faux, et vrai s'ils renvoient tous vrai. La classe *HandStructure* fonctionne sur le même principe que cette dernière, en contenant les doigts et non les os, avec évidemment, son type, c'est-à-dire si c'est la main droite ou gauche. De plus, elle possède également les vecteurs de la paume et celui de la direction de la main, qui sont également pris en compte lors de la

comparaison, sur le même principe pour les coordonnées des os. Celle-ci garde également la matrice $N$, créée précédemment. Pour comparer un geste fait avec deux mains, on utilise la classe *DoubleHandStructure* qui possède deux instances de *HandStructure* et quatre longueurs. Celles-ci sont des distances entre les deux mains. Il y a alors celle entre le bout de l'os du métacarpe de l'index de chaque main, le début du même os, et ces deux mêmes positions pour l'auriculaire. On revient sur le même principe de comparaison, avec la même divergence, mais ici nous avons la possibilité de choisir d'inclure ou non les distances. Enfin nous avons la classe *GestureStructure* qui va permettre de définir un geste. Celle-ci va alors contenir une instance de l'interface IDefineStructure, dont *HandStructure* et *DoubleHandStructure* héritent. Elle possède également un id, un nom et une description, mais aussi un booléen pour savoir si, dans le cas d'un geste à deux mains, la distance entre ces deux dernières est utile ou non. En effet, dans certains gestes à deux mains, elles doivent être collées, tandis que dans d'autres, la distance entre les deux n'est pas prise en compte. Les classes pour les os et les doigts possèdent une méthode pour qu'elles puissent se cloner, tout en utilisant une matrice pour revenir à un même repère. La classe pour la main (et non la double) possède alors une méthode similaire, mais qui va utiliser sa propre matrice qu'elle possède, et non une passée en paramètres. Ainsi, cette même classe définit une méthode de comparaison prenant en compte la matrice, afin que les mains soient comparées dans le même repère. Une classe nommée *StructureManager* est également présente afin de permettre de récupérer une instance de *IDefineStructure* à partir d'une *Frame* du Leap Motion* (la classe Frame permet de récupérer les données que le Leap Motion* renvoie à un certain moment). Elle possède aussi des méthodes pour comparer deux *IDefineStructure* avec ou sans la distance pour les *DoubleHandStructure*, et en prenant en compte ou non la matrice qu'elles possèdent.

Ainsi, nous avons réussi à faire en sorte de comparer des gestes que l'utilisateur pourra définir de lui-même. Si nous prenons le diagramme de séquence de *compareGestures* se trouvant dans l'annexe, nous pouvons reprendre les différentes étapes. Tout d'abord, la comparaison va dépendre si la distance entre les mains est utile ou non, même si nous avons qu'une seule main. Si elle l'est, la classe *GestureStructure* utilisera la méthode *compareWithNormalization* de *StructureManager*.

Cependant, si elle ne l'est pas, ce sera *compareWithNormalizationWithoutDistance*. Celle-ci appellera la bonne méthode suivant le type de classe, c'est-à-dire si on a deux mains elle utilisera celle au même nom dans la classe *DoubleHandStructure*, sinon elle prendra *compareWithNormalization* dans la classe *HandStructure*. Celle des deux mains, appellera la méthode précédente pour une main, sur les mains qu'elle possède. Cette dernière utilisera sa méthode pour récupérer son instance normalisée et celle de l'instance avec laquelle elle est comparée, puis appellera sa méthode pour comparer les deux nouvelles *HandStructure*. Cette comparaison descendra jusqu'à la classe BoneStructure, et, suivant le résultat de chaque os, renverra un booléen. C'est ce dernier qui nous permet de savoir si les gestes effectués sont les mêmes ou non.

C'est ainsi que ce conclu notre méthode de reconnaissance. Une reproduction de la logique de la fonction décrite peut être retrouvée dans le diagramme de séquence en annexe.

### b. API REST*

Une des étapes principales dans notre application est le transit de données. Comme nous avons pu le voir dans la partie III.3 Vision générale de l'architecture, nous avons scindé notre application en plusieurs blocs distincts et faisons transiter les données par le biais de notre API REST*.

Celle-ci est à la fois capable de communiquer avec le Leap Motion*, avec la base de données ou encore de faire de la gestion locale des données. Elle est capable d'appeler les méthodes de notre métier afin d'être le manager de nos actions. Pour cela nous avons utilisé le Framework* Spring Boot afin de développer notre API REST*. Cet outil est connu pour être très puissant et très utilisé dans le monde professionnel.

Une API REST* fonctionne sur la base du protocole HTTP*, nous adressons des actions définies pour une certaine adresse et ainsi pouvons communiquer avec un utilisateur lorsque celui-ci les requêtes. Pour illustrer cela, nous allons utiliser un exemple très simple, un utilisateur peut vouloir savoir si un Leap Motion* est actuellement connecté, pour cela nous avons défini que lorsque qu'une requête HTTP* GET est faite à l'adresse /leap/state, nous lui retournons un booléen indiquant l'état au moment T.

Si celui-ci veut nous envoyer des données alors il peut le faire à travers le corps d'une requête POST. On retrouve de nombreux types de requête telle que DELETE, MODIFY, POST, GET, à l'aide de ceci nous communiquons de manière naturelle avec l'API REST*. Une fois ces requêtes réceptionnées, nous pouvons attribuer des actions métiers selon le contexte voulu comme requêter la base de données ou le Leap Motion*.

Celle-ci nous a notamment permis de faire les interactions avec la base de données à l'aide du module MongoDB implémenté dans l'outil Spring Boot.

### c. Base de données

Afin de permettre à l'utilisateur de stocker ses données en ligne et de pouvoir les récupérer depuis n'importe où, nous avons mis en place une base de données. Nous voulions profiter de ce projet pour expérimenter l'interaction avec une base de données NoSQL qui est une famille de systèmes de gestion de base de données qui s'écarte du paradigme classique des bases relationnelles. En effet, comme le nom le suggère, celle-ci n'utilise pas le langage bien connu SQL ou « Structured Query Language » mais le langage JSON*.

Exemple d'un script stocké dans la base de données écrite en JSON* :

{

"_id": "ZWxweGV4YW50bA==",

"execType": "Python",

"args": ["-h"],

"file":
"cGFyZW50cywgYmFiaWVzID0gKDEsIDEpCndoaWxlIGJhYmllcyA8IDEwMDoKICAgIHByaW5";

"description": "An Amazing script !!",

"idGesture": "b3BlbiBoYW5kYW4gb3BlbmVkIGhhbmQ=",

"_class": "Core.Script.Script"

}

C'est pourquoi, nous avons choisi d'utiliser MongoDB. MongoDB est un système de gestion de base de données orienté documents, répartissable sur un nombre quelconque d'ordinateurs et ne nécessitant pas de schéma prédéfini des données. Ce choix a aussi été fait par rapport au Framework* d'API REST*, Spring BOOT qui contient des méthodes déjà implémentées permettant d'utiliser MongoDB. De plus, cet outil populaire est vraiment apprécié des développeurs et retrouvé dans un grand nombre de projets.

Nous avons choisi de simplifier le plus possible la base de données. Pour cela nous avons réalisé un MLD ou modèle logique de donnée nous permettant de décrire comment celle-ci sera constitué.

Voici le MLD de la base de données de HandyHand :

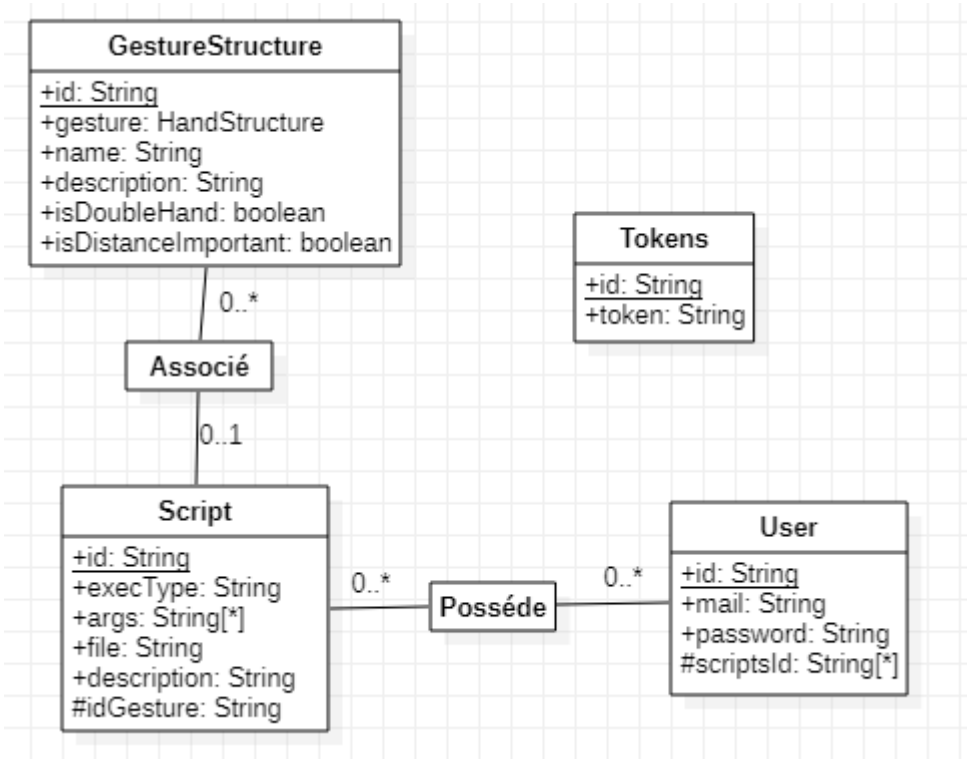



*Figure 22 : MLD de la base de données*

Chaque cadre représente un document dans la base données. Toutes les données représentées sur ce diagramme sont écrites en tout lettre et peuvent être lu en clair sur la base données. Cela nous a permis de débugger la base de données à chaud facilement.

La modification de cette base de données est possible grâce aux fonctions qui sont appelées dans l'API REST*. Le format NoSQL de la base de données nous permet de créer un code plus

robuste dans lequel aucune requête est écrite en tout lettres. Il facilite aussi l'implémentation de ces fonctions.

# VI. Difficultés rencontrées

## 1. Difficultés techniques

Nous avons décidé dès le départ de se servir de ce projet pour apprendre le plus possible aussi bien au niveau des technologies que des méthodes de travail. Il en va donc de soi que nous ayons fait face à plusieurs problèmes lors du processus d'apprentissage des technologies. On considérera ces problèmes comme des adversités que nous avions pris en compte pour la réalisation de notre projet. Ces adversités "maîtrisées" sont majoritairement liées aux nouvelles technologies telle que Spring BOOT, le NoSQL (avec MongoDB), LeapSDK*, TypeScript, ReactJS, Electron, Python, etc…

Évidemment nous avons fait face à de multiples autres problèmes que nous n'avions pas prévu telle que l'utilisation de la SDK* adapté pour le Leap Motion*, la non-sérialisation des matrices du Framework* EJML avec Jackson (Moteur de sérialisation de Spring) ou encore l'utilisation du moteur Processing dans sa version Raw (et non son langage indépendant). On peut aussi noter la transformation d'image renvoyé par le Leap Motion* vers des données dans un format d'image valide le tout en limitant le temps pour la génération de cette image (gestion de la déformation, utilisation de « shaders » ou nuanceurs, simplification des images et autres techniques).

## 2. Difficultés organisationnelles

Cependant les problèmes techniques ne furent pas ceux qui nous apportèrent le plus de fils à retordre. En effet, l'organisation d'un projet sur le long terme et impliquant autant de collaborateurs était nouveau pour nous. Ainsi, nous avons face à plusieurs problèmes durant tout le processus.

Nous avons eu à gérer quelques retards sur certains sprints ou encore des rushs lors de la réalisation dû à des minimisations de l'ampleur des tâches. A présent, nous avons appris à mieux appréhender un problème ainsi que les conséquences qui peuvent venir avec.

Un des événements qui a le plus bouleversé notre organisation fut la démission d'un de nos collaborateurs. En effet, ayant trop minimisé la situation, la balance des tâches et l'ampleur du travail qui s'accumulait laissa un impact visible sur l'équilibre et la cohésion de l'équipe. Nous en avons fait le constat trop tard. Heureusement, après une requalification des tâches, un remaniement des équipes et une redéfinition des objectifs, l'équipe a su se remettre sur les rails afin d'avancer dans ce projet de manière unie. Nous avons aussi eu à gérer une charge de travail extérieure assez importante due aux nombreux autres projets. Durant certaine période assez chargée, il était compliqué pour chacun d'investir autant de temps que nous l'aurions voulu.

# VII. Bilan technique

## 1. Réalisations

Lors de la conclusion de ce projet, nous avons plusieurs réalisations terminées. Entre autres, nous notons deux interfaces capables de communiquer avec l'API REST* que sont l'application bureau et l'interface en ligne de commande. Nous avons aussi réalisé, comme cité auparavant, une API REST* permettant de faire communiquer notre métier avec le reste de l'application. Celle-ci, communique particulièrement avec la base de données, le Leap Motion* et les fichiers locaux. L'une de nos réalisations majeures a aussi été la production d'une reconnaissance permettant à l'utilisateur de configurer lui-même les gestes qui devront être reconnus. Nous avons donc amélioré et rendu extensible une fonction déjà contenue dans la librairie du Leap Motion*. Cependant, nous n'avons pas réalisé tout ce que nous voulions.

## 2. Protocoles expérimentaux

Un protocole expérimental a été réalisé lors de ce projet, celui-ci a pour but d'évaluer la performance de notre algorithme de reconnaissance des gestes. Dans ce rapport se trouve le condensé du protocole expérimental réalisé. Celui-ci est récupérable en entier sur le Git du projet dans le dossier conception.

Ci-dessous se trouve les résultats obtenus lors des expériences. Une expérience avec un geste simple tel que la pierre et une expérience avec un geste dit « compliqué » tel que le cœur avec les deux mains ont été choisies. Toutes deux ont pour but d'évaluer la reconnaissance d'un geste par rapport à un geste prédéfini. Chaque expérience se déroule en cinq étapes. La première consiste à effectuer le geste enregistré le plus précisément possible pour savoir si l'algorithme reconnait avec ou sans difficulté le geste de base. Ensuite les quatre dernières étapes consistent à reproduire le geste enregistré avec différentes inclinaisons comme le montre les photos présentes ci-dessous. Une batterie de cinquante gestes a été effectuée pour chaque étape. Après avoir pris notes des résultats obtenus par l'algorithme ceux-ci ont été transformés en pourcentage que vous pouvez retrouver ci-dessous.

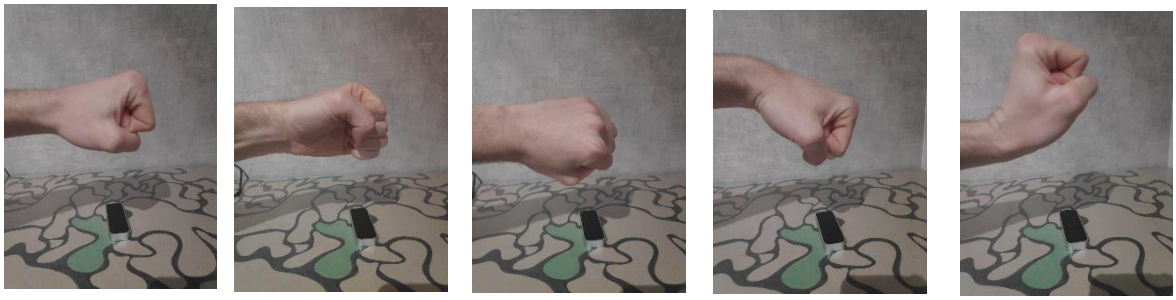


*Figure 23 : Illustration orientation de la main*

Tableau de résultats d'expérience :

| Expérience | Gestes | Inclinaison à gauche | Inclinaison à droite | Inclinaison en avant | Inclinaison en arrière | Geste effectué avec précision |
|---|---|---|---|---|---|---|
| N°1 | Pierre | 90% | 72% | 78% | 82% | 100% |
| N°2 | Cœur | 0% | 0% | 0% | 0% | 82% |

*%=Taux de reconnaissance*

Ces expériences nous révèlent que notre algorithme est efficace si l'utilisateur reproduit avec le plus de précision possible son geste, 100% de reconnaissance pour la pierre et 82% pour le cœur. Seulement dès que le geste est reproduit avec différentes inclinaison le taux de reconnaissance baisse pour un geste simple et devient nul pour un geste compliqué comme le cœur.

## 3. Evolutions du projet dans le futur

Le projet étant clôturé dans le cadre du DUT nous avons tout de même des idées pour le compléter. Nous pensons que la plus grosse amélioration serait de supprimer toutes les dépendances entre le Back-End* et le Front-End*. Cela nous permettrait alors de totalement découpler les deux parties et ainsi de proposer à des utilisateurs une prise en charge des services de reconnaissance à distance permettant alors à la machine de l'utilisateur, potentiellement un mini-contrôleur, de seulement avoir à envoyer les images prise par le Leap Motion*. Cette idée, a été proposée par M. Bouhours et c'est dans cette ambition que nous avons choisi de faire communiquer le Back-End* et le Front-End* avec une API REST*. Cependant c'est un objectif que nous n'avons pas pu atteindre faute de temps. Nous pensons que c'est la principale voie d'amélioration du projet et nous avons déjà mis des points d'extension permettant d'implémenter cela plutôt facilement.

Par ailleurs, nous avons utilisé une pile de technologies web pour le développement de l'interface bureau dans l'objectif de rendre notre application la plus portable possible. C'est pourquoi, l'une des voies d'amélioration serait d'implémenter une interface web (en éliminant la surcouche Electron). De plus, Electron offre la possibilité d'un portage mobile qui peut s'avérer être un angle de développement intéressant.

La création de scripts publics pour les utilisateurs est aussi une piste pour permettre à ceux-ci n'ayant pas les compétences de créer leur propre script pour pouvoir utiliser notre application.

# VIII. Conclusion

Pour la réalisation de ce projet, nous avons pris les devants en proposant notre propre sujet avec des outils encore inconnus avant le début de celui-ci. Pour la première fois, nous avons suivi une organisation Agile* sur un projet que nous devions définir nous-même. Ce fut une occasion pour nous de découvrir le travail en équipe et l'organisation sur le long terme. Nous avons appris à communiquer de manière efficace entre les membres de l'équipe afin d'avancer dans ce projet commun malgré les conditions particulières du distanciel. Nous avons dû faire preuve d'adaptation pour fournir un travail en temps et en heure même si nous étions freinés par d'autres projets.

Nous avons vu ce projet, avant tout comme une manière d'apprendre de nouvelles technologies, langages et techniques. Ce fut pour nous, un travail d'apprentissage et d'adaptation supplémentaire afin de rendre le produit que nous souhaitions. Cette curiosité d'en apprendre beaucoup sur des technologies autres que celle travaillées au cours du DUT fut un choix qui a stimulé notre curiosité et nous a énormément apporté.

Nous pensons que cette adaptation dont nous avons dû faire preuve se retrouvera être une vraie force dans le monde du travail. Nous avons porté une attention particulière à choisir des technologies populaires comme le Framework* Spring BOOT ou bien la pile de technologies ReactJS qui nous permettrons de nous démarquer sur le marché du travail grâce au savoir supplémentaire que nous avons cherché à acquérir.

Nous avons réussi à réaliser notre souhait initial qui été d'utiliser un outil matériel (le Leap Motion*) pour ajouter une couche logiciel grâce à nos connaissances. C'est encore une fois un aspect de l'informatique que l'on regrette de ne pas avoir pu explorer dans notre cursus. Ainsi nous avons profité de cette occasion pour approfondir ces sujets à travers notre projet.

Pour finir, les délais imposés pour la réalisation d'un projet aussi complexe ont été plutôt courts surtout à la suite de l'abandon d'un de nos collaborateurs. Nous avons fait preuve de flexibilité et de persévérance, parfois pour respecter les délais mais aussi changer notre vision du projet. L'adaptation du point de vue de l'organisation fut assez déroutante pour l'ensemble des membres de l'équipe mais celle-ci a été une expérience sociale nouvelle et intéressante pour nous.

Nous avons appris tant sur l'aspect technique dans lequel nous avons tout fait pour apprendre de nouvelles choses. Mais aussi dans l'aspect social et organisationnel qui nous a appris à communiquer, travailler en équipe, préparer, organiser, le tout en continuant notre cursus normal en parallèle.

Ce projet nous a tous beaucoup appris et nous sommes très heureux de l'impact qu'il a pu avoir sur nous en tant qu'étudiants et futurs professionnels.

# IX. Résumé en Anglais

This project is part of our two-year university diploma in Computer Science at the University Institute of Technology, UCA in Clermont-Ferrand. We did a four-month project in a group of 5.

The goal of the project is the development of software allowing the use of the Leap Motion device to launch scripts from predefined gestures which will then be recognized by the device. The Leap Motion is a computer hardware sensor device that supports hand and finger motions as input, analogous to a mouse, but requires no hand contact or touching. We used this camera and the library made available by the developers of the Leap Motion to develop this application.

Our project has been through multiple stages starting with the brainstorming stage where we all worked on how we wanted the project to look like. After that, both teams worked on the conception, each of them in their development axis. For example, the Back-End team spent most of the time defining the general architecture or even the package diagram of the Core. While the Front-End team were working on the graphical chart, the personas, Wireframes and other necessary UI/UX documents. After this period of conception, when the ideas were clear and everybody was on the same wavelength, we all fixed which languages and Frameworks we were about to work with and then we started the development process.

Once the development process started, we all focused on our tasks that we previously defined. The first weeks has mostly been dedicated to learning the new technologies and environments. Through all the process, Mr. Provot was always here to advise us and give his opinion about our decisions with an exterior point of view, which helped us a lot keeping the project on the track. He brought us advises from his professor status but also with the status of client we attributed him. We faced multiples problems during the realization of our project, as much on the development side than the social and organizational one. Of course, we had to deal with multiples bugs and unknown technical aspect that we had to learn and deal with during the process. Although, we also had to face the loss of a member during the realization, this has been quite disturbing for us and the organization we previously settled. At this point of the project, we also had to deal with all the side work from other project but in the end, we managed to achieve our goals. After re-planning and reorganizing the different tasks and the teams we managed to get back on the pure realization of our application, with a clear and defined organization.

In conclusion, the project is up to what we wanted to do at least but it remains a lot of ideas that came up during the realization. If we had more time, we would have definitely disassociated the Back-End and the Front-End in order that this application would be totally optimized. But after all we are totally satisfied with what we managed to do in 14 weeks in a group of 5.

# BIBLIOGRAPHIE ET WEBOGRAPHIE

https://docs.spring.io/spring-framework/docs/5.1.0.RELEASE/spring-framework-reference/
https://developer-archive.leapmotion.com/documentation/java/index.html
https://fr.wikipedia.org/wiki/Vecteur
https://ent.uca.fr/moodle/pluginfile.php/1390368/mod_resource/content/0/1a_calcul_matriciel_slides.pdf
https://www.docteurclic.com/symptome/douleur-des-doigts.aspx
https://fr.wikipedia.org/wiki/Cosinus
https://fr.wikipedia.org/wiki/Bijection_r%C3%A9ciproque
https://fr.wikipedia.org/wiki/Arc_cosinus
https://fr.wikipedia.org/wiki/Radian
https://fr.wikipedia.org/wiki/Matrice_(math%C3%A9matiques)#D%C3%A9finitions
https://fr.wikipedia.org/wiki/Produit_matriciel
https://fr.wikipedia.org/wiki/Coordonn%C3%A9es_homog%C3%A8nes
https://fr.wikipedia.org/wiki/Matrice_de_rotation#Dimensions_embo%C3%AEt%C3%A9es
https://arduino103.blogspot.com/2015/08/detecter-les-mouvements-sur-le-hat-sense.html
https://www.researchgate.net/figure/The-users-hand-with-the-three-axes-and-the-names-of-the-orientation-around-these-axes_fig17_46720588
https://docs.spring.io/spring-Framework/docs/5.1.0.RELEASE/spring-Framework-reference/overview.html#overview
https://www.baeldung.com/
https://github.com/FasterXML/jackson
https://www.electronjs.org/docs
https://reactjs.org/docs/getting-started.html
https://react-bootstrap.github.io/
https://github.com/lessthanoptimal/ejml
https://www.typescriptlang.org/
https://babeljs.io/
https://webpack.js.org/
https://eslint.org/
https://yarnpkg.com/
https://processing.org/
https://gradle.org/
https://github.com/google/gson
https://www.mongodb.com/
https://requests.readthedocs.io/en/master/
https://github.com/willmcgugan/rich
https://stackoverflow.com/
https://www.blender.org/

# LEXIQUE

Leap Motion : capteur permettant de virtualiser nos mains. Cela nous permet ainsi de lancer un traitement prédéfini lorsqu'un certain mouvement est reconnu.

Back-End : terme désignant un étage de sortie d'un logiciel devant produire un résultat. C'est la partie invisible de l'iceberg qui permet de faire les traitements nécessaires afin d'échanger les données avec le Front-End*.

Front-End : terme désignant toutes les parties d'un logiciel avec lequel un utilisateur va interagir. On retrouve ici la partie visible de l'iceberg (interface graphique, interface en invite de commande…)

API : ou « interface de programmation d'application » ou « interface de programmation applicative » est un ensemble normalisé de classes, de méthodes, de fonctions et de constantes qui sert de façade par laquelle un logiciel offre des services à d'autres logiciels.

REST API : signifie « Representational state transfer » API, c'est un style architectural d'application se basant sur le protocole HTTP. Il sert à la communication de données à un instant T par le biais de requêtes/réponses HTTP.

Requêtes HTTP : Requêtes exprimées par le client envoyé vers un serveur en respectant le protocole HTTP. Celles-ci permettent une communication de représentation de données à un instant T.

Framework : Ensemble cohérent de composants logiciels qui servent à l'abstraction d'un domaine spécifique de l'application pour celui qui l'utilise.

Méthode AGILE : Méthode de travail basé sur la planification adaptative, le développement évolutif et l'amélioration continue. Elle repose sur une grande communication et une capacité d'adaptation constante.

Sprint AGILE : Point central de la méthodologie Agile, le sprint est une période définie afin de réaliser une tâche X qui subira par la suite une vérification, adaptation et/ou validation.

UI : Signifie User Interface et représente l'ensemble des outils avec lesquelles le client va interagir.

UX : Signifie User eXperience et représente la manière par laquelle un utilisateur va prendre en main, appréhender et interagir avec l'UI.

Wireframe : Est une représentation squelettique, aidant à l'organisation et la disposition des éléments d'un UI pour améliorer l'UX.

Base de données non relationnelle : Modélisation d'une base de données sans relations de tables. Elles peuvent être basées sur différentes architectures (documents, colonnes, graph…).

Refactor : Processus de restructuration d'un code existant vers une version plus convenable (utilisation de patrons de conceptions, simplification, types d'implémentations, etc ...)

ID : Abréviation d'identifiant

CLI : Signifie Command Line Interface qui représente un programme capable de recevoir des commandes sous la forme de lignes de textes afin d'exécuter les actions métiers adaptées.

JSON : Signifie JavaScript Object Notation. C'est un format standardisé souvent utilisé pour l'échange et le stockage de données basé sur le principe des Clés-Valeurs. Il a la particularité d'être lisible par l'homme.

SDK : ou kit de développement est un ensemble d'outils logiciels destinés aux développeurs, facilitant le développement d'un logiciel sur une plateforme donnée.

# ANNEXES

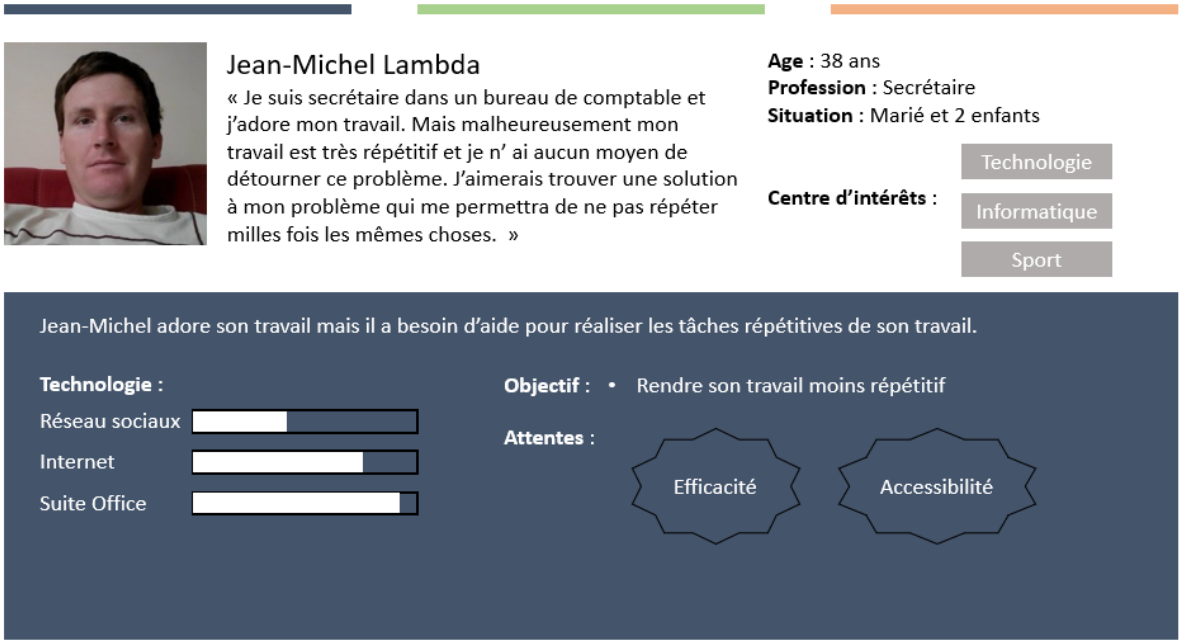
## I.Personas :

**Jean-Michel Lambda**

« Je suis secrétaire dans un bureau de comptable et j'adore mon travail. Mais malheureusement mon travail est très répétitif et je n'ai aucun moyen de détourner ce problème. J'aimerais trouver une solution à mon problème qui me permettra de ne pas répéter milles fois les mêmes choses »

Jean-Michel adore son travail mais il a besoin d'aide pour réaliser les tâches répétitives de son travail.

**Age** : 38 ans
**Profession** : Secrétaire
**Situation** : Marié et 2 enfants
**Profil Technique** : Se tient au courant des nouvelles technologies, maîtrise les bases de l'informatique.
**Logiciels préférés** : Word, Excel, Photoshop, …




**Lucas Chaux**

« J'ai subi un grave accident quand j'étais jeune, lors de cet accident j'ai perdu l'usage de mes jambes. J'aimerais malgré mon handicap prendre mon indépendance et m'acheter un appartement pour y vivre au quotidien. »

Lucas a peur d'habiter seul dans un appartement, car il ne sait pas s'il pourra y arriver. Plusieurs tâches sont très difficiles à réaliser avec son handicap.

**Age** : 26 ans
**Profession** : Informaticien
**Situation** : Célibataire

**Profil Technique** : Accro à la programmation, se renseigne constamment sur les nouveautés informatiques et technologiques.
**Logiciels préférés** : IntelliJ, Eclipse, Visual Studio, ….

Lucas Chaux

« J'ai subi un grave accident quand j'étais jeune, lors de cet accident j'ai perdu l'usage de mes jambes. J'aimerais malgré mon handicap prendre mon indépendance et m'acheter un appartement pour y vivre au quotidien. »

**Age** : 26 ans
**Profession** : Informaticien
**Situation** : Célibataire

**Centre d'intérêts** : Jeux vidéos, Informatique, Lecture

Lucas a peur d'habiter seul dans un appartement, car il ne sait pas s'il pourra y arriver. Plusieurs tâches sont très difficiles à réaliser avec son handicap.

**Technologie** :
Réseau sociaux
Internet
Console de jeux
Programmation

**Objectif** : • Faciliter l'interaction avec son environnement

**Attentes** : Efficacité, Accessibilité, Simplicité

## II.USER STORIES :

Jean Michel veut utiliser le logiciel HandyHand

Scenario 1 :

- Etant donné que Jean Michel n'a pas de compte.
- Lorsque celui-ci va cliquer sur "Créer un compte sur la page d'accueil"
- Alors il va rentrer un username et un mot de passe
- Et pourra par la suite se connecter

Scenario 2 :

- Étant donné que Jean Michel a un compte.
- lorsqu'il entrera ses information (username/mot de passe) et cliquera sur "Connexion"
- Alors il sera dirigé vers la page d'accueil

Lucas veut éteindre la lumière de sa chambre

Scenario 1 :

- Étant donné que le script existe déjà et est inclus dans la Base de données du logiciel
- Lorsque Lucas va vérifier dans l'onglet "Scripts" si un script existe déjà pour éteindre sa lumière.
- Alors il trouvera le script
- Et Lucas s'informera donc sur le mouvement de main à réaliser pour le lancer.
- Et Lucas éteins sa lumière.

Scenario 2 :

- Étant donné que le script n'existe pas
- Lorsque Lucas va vérifier dans l'onglet "Scripts" si un script existe déjà pour éteindre sa lumière.
- Alors celui-ci verra que le script n'est pas présent.
- Alors Lucas naviguera donc vers l'onglet mes scripts afin de le créer.
- Lorsque Lucas clique sur "Ajouter une interaction", entre un nom, une description et visualise ce que le capteur détecte.
- Alors celui-ci enregistre le mouvement désiré en cliquant sur "Enregistrer le mouvement".
- Suite à cela il parcourt ses fichiers et sélectionne le script lié au contrôle de sa lumière.
- Et enregistre son interaction
- Et il peut maintenant contrôler sa lumière.

# III.CAHIER DES CHARGES

## Index



## Présentation générale du projet

### Présentation des acteurs

Maître d'ouvrage : Laurent Provot

Chef de Projet : Romain Olivier

Maîtrise d'œuvre :

Emrick Pesce,

Thomas Blanc,

Yoann Periquoi,

Augustin Laborie,

Romain Olivier

Pour donner suite à l'appel d'offre de M. Laurent Provot, les deux collaborateurs Romain Olivier et Augustin Laborie ont décidé de proposer une offre acceptée par l'initiateur. Pour la bonne réalisation du projet, l'équipe a dû s'agrandir et rallier Emrick Pesce, Thomas Blanc ainsi que Yoann Periquoi.

Il est donc logique que M. Laurent Provot soit le maître d'ouvrage car il est à l'initiative de l'idée du projet sans pour autant faire partie de la production. Pour donner suite à la formation de l'équipe et de réunions préalable, nous avons décidé d'élire le chef de projet par vote. Le résultat fut l'élection de Romain Olivier. Tous les membres de l'équipe sont cependant hiérarchiquement équivalents. C'est pourquoi chacun d'eux font partie de la maîtrise d'oeuvre.

* Pour le bien de la réalisation du projet scolaire, cette situation est factice. La description de chacun des membres est disponible dans leurs pages personnelles.

### Nature de la prestation demandée

Ce projet réside en la réalisation d'un logiciel permettant d'utiliser l'appareil Leap Motion* pour lancer des scripts à partir de gestes prédéfinis qui seront alors reconnus par l'appareil.
Le Leap Motion* est un capteur permettant de virtualiser nos mains. Cela nous permet ainsi de lancer un traitement prédéfini lorsqu'un certain mouvement est reconnu. Ce projet n'est pas une innovation puisque de nombreux autres logiciels sont déjà en circulation sur la plateforme de téléchargement d'application du Leap Motion* (Leap Motion* SDK*). Cependant nous ne partons pas d'une base déjà existante pour réaliser notre propre logiciel, nous utilisons seulement la librairie fournie par les développeurs du Leap Motion*.

Nous devrons livrer un logiciel fonctionnel et déployable avec lequel sera disponible une interaction complète via une interface graphique et une interface en ligne de commande (ou CLI) avec le contrôleur Leap Motion*. Ces interactions nous permettrons alors de pouvoir lancer un script (suite de commandes) et ainsi de réaliser des tâches à partir d'un seul mouvement.

- Celui-ci devra permettre à l'utilisateur de :
- Gérer ses scripts (créer, modifier, lire, mettre à jour et supprimer les scripts et leurs gestes déclencheurs)
- Initialiser ses scripts grâce à des mouvements
- Gérer la connexion avec le Leap Motion*
- Avoir un retour visuel de ce que perçoit le Leap Motion*
- Lancer un script à tout moment grâce à un outil qui observe en permanence le flux vidéo et repère les gestes
- Enregistrer son environnement sur une base de données distante
- Récupérer son environnement depuis n'importe où grâce au serveur distant
- S'authentifier pour accéder à son profil
- Utiliser son propre profil en local avec une gestion "hors ligne"

Ces fonctionnalités sont observables dans le diagramme de cas d'utilisation accessible sur le wiki du projet.

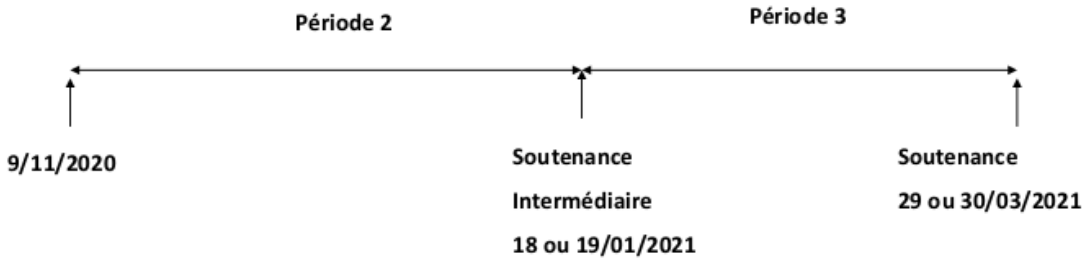
Lors de ces interactions, on sera capable de rattacher des processus afin de se servir du Leap Motion* comme un hub (voir Google Home / Alexa). Ce logiciel a plusieurs objectifs :

Une utilisation quotidienne pour quiconque possédant un Leap Motion*

Une aide supplémentaire pour les personnes en situation d'handicap

Une solution innovante pour des interactions sans contact (respect des règles sanitaires)

Estimation des grandes étapes et dates butoirs




## Organisation de l'équipe et du travail

Ce projet se réalisera majoritairement en distanciel du fait de la crise sanitaire actuelle.
Aucune réunion ou meeting n'est donc possible en présentiel sur la première période de réalisation.

Pour la réalisation du projet, l'équipe de développement a opté pour une méthodologie inspirée de SCRUM (en version allégée).

Les outils de conceptions de l'équipe seront :

Communication : Discord / BBB / Teams

Version Control : Git/GitLab/GitKraken/CircleCI

Éditeur : IntelliJ IDEA, VSCode, Sublime Text, DBeaver

Matériel : Ordinateurs personnels (Windows/Linux), Leap Motion*

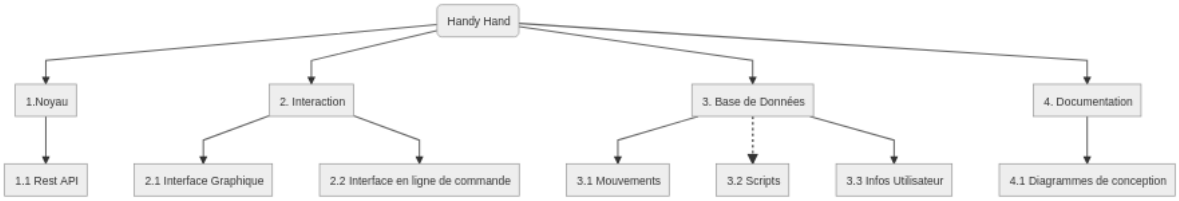
Langage : Java, JavaFx, Spring (API), JavaScript (React),MarkDown

Design : Adobe XD, Balsamiq, Photoshop

## Cahier des charges fonctionnel

Fonctionnalités attendues

Lors de ce projet, nous allons fournir différentes fonctionnalités :




| Niveaux | Description |
|---|---|
| Niveau initial | **HandyHand**: Application reconnaissant des mouvements de mains et exécutant des scripts à partir de ceux-là |
| Niveau 1 | **1. Noyau:** Responsable de toutes les interactions métiers avec le LeapMotion<br>**2. Interaction:** Responsable de l'Interaction Homme-Machine<br>**3. Base de Données** Responsable du stockage des données<br>**4. Documentation** Documentation détaillée de la structure de l'application et de ses composants |
| Niveau final | **1.1 Rest API** Permet la transition de données entre toutes les interfaces et le noyau sans réécriture unique<br>**2.1 Interface Graphique** Interface ergonomique permettant la configuration des interactions avec le LeapMotion<br>**2.2 CLI** Interface simplifiée permettant la configuration sur des outils sans interface graphique<br>**3.1 Mouvements** Collection de données représentants les mouvements compris par le noyau<br>**3.2 Scripts** Collection de scripts reliables à des interactions avec le LeapMotion<br>**3.3 Infos Utilisateurs** Collection d'informations données par l'utilisateur pour les interactions (Clé API, ...)<br>**4.1 Diagrammes de Conception** Description détaillée de l'architecture de l'application suivant les normes UML |

Une description plus poussée de ces tâches peut être retrouvée dans la partie BackLogs de notre wiki.

### Contraintes sur la réalisation du projet

Lors de la réalisation de ce projet, nous faisons face à différentes sortes de contraintes.

Nous pouvons relever les difficultés du travail en distanciel durant l'intégralité de la première période. Entre autres, on note l'utilisation de nos ordinateurs personnels comme outils principaux. La récupération des Leap Motion* en situation de confinement. Les connexions internet instables et limitées de nos résidences. L'impossibilité de contact humain entre les collaborateurs mais aussi avec le maître d'ouvrage qui complexifie la communication et ralentit le processus de développement.

Nous faisons aussi face à des contraintes de temps, le projet ayant des dates butoirs à respecter. Nous devons également prendre en compte le temps de réalisation de projets scolaires parallèles et des cours universitaires.

### Contraintes sur l'utilisation du produit

L'utilisation du produit requiert l'obtention d'un Leap Motion* ainsi qu'un ordinateur quelconque (Windows/Linux/Raspberry) avec Java installé.

Avoir au minimum une main fonctionnelle avec 5 doigts.

L'appareil Leap Motion* nécessite une petite "période d'adaptation" afin de comprendre la distance à laquelle il faut placer sa main, les gestes reconnus...

L'application nécessite la création de scripts et donc d'une connaissance poussée de l'informatique et d'un langage de programmation pour mettre au point un traitement efficace.

Critères d'appréciation de la qualité du produit

L'appréciation du produit se fera à travers différents vecteurs. Notamment les retours utilisateurs, l'ergonomie, la précision de détection, la rapidité d'exécution.

Les retours utilisateurs se feront tout d'abord via le maître d'ouvrage qui donnera alors son retour lors de démonstrations effectuées dans le cadre de fin de "sprint". Tous les retours seront alors pris en compte pour développer la suite.

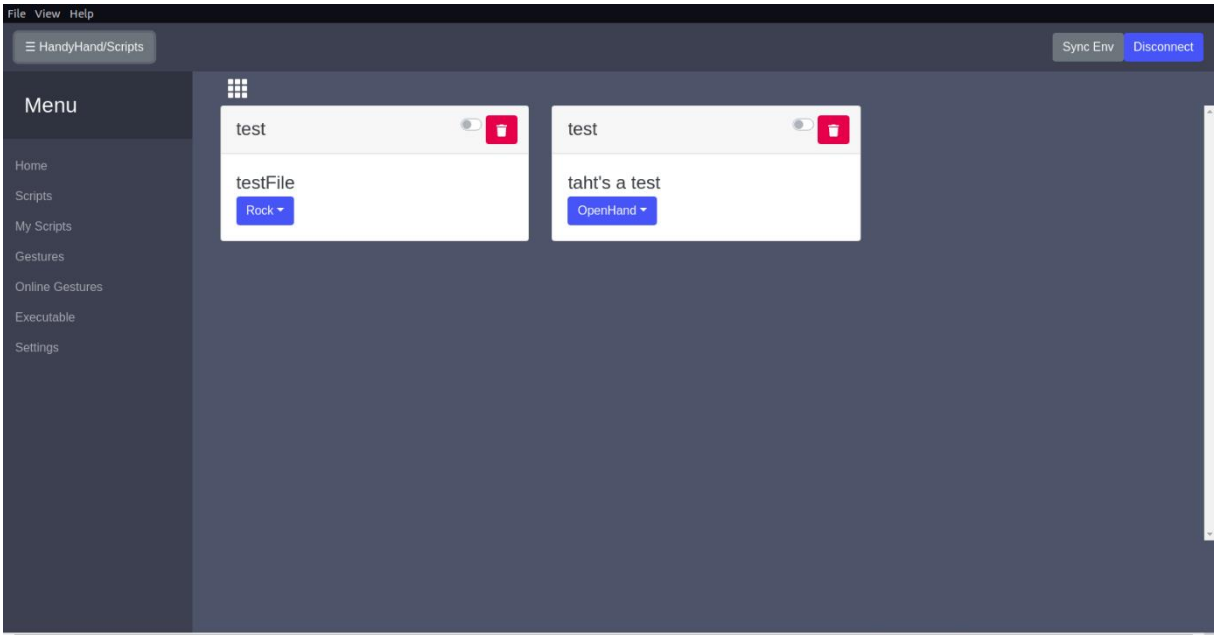
L'ergonomie est pour nous un point très important car nous voulons rendre cette application accessible à tous et plus particulièrement aux personnes en situation d'handicap. Celle-ci pourra alors contenir des fonctionnalités comme un mode de contraste élevé ou bien la lecture des textes survolés. Elle sera aussi mise en valeur de par l'organisation de l'interface Homme-Machine mais également par une mobilité omniprésente permettant d'accéder à l'application facilement depuis n'importe où.

La précision de détection sera testée grâce à des batteries de tests permettant définir le nombre de fois qu'un mouvement est repéré en fonction du nombre d'essais. De plus on dénombre plusieurs difficultés de reconnaissance pour certains mouvements, c'est pourquoi l'acceptation de la précision pour un mouvement simple comme un doigt levé sera plus stricte que celle pour un cœur formé avec les mains.

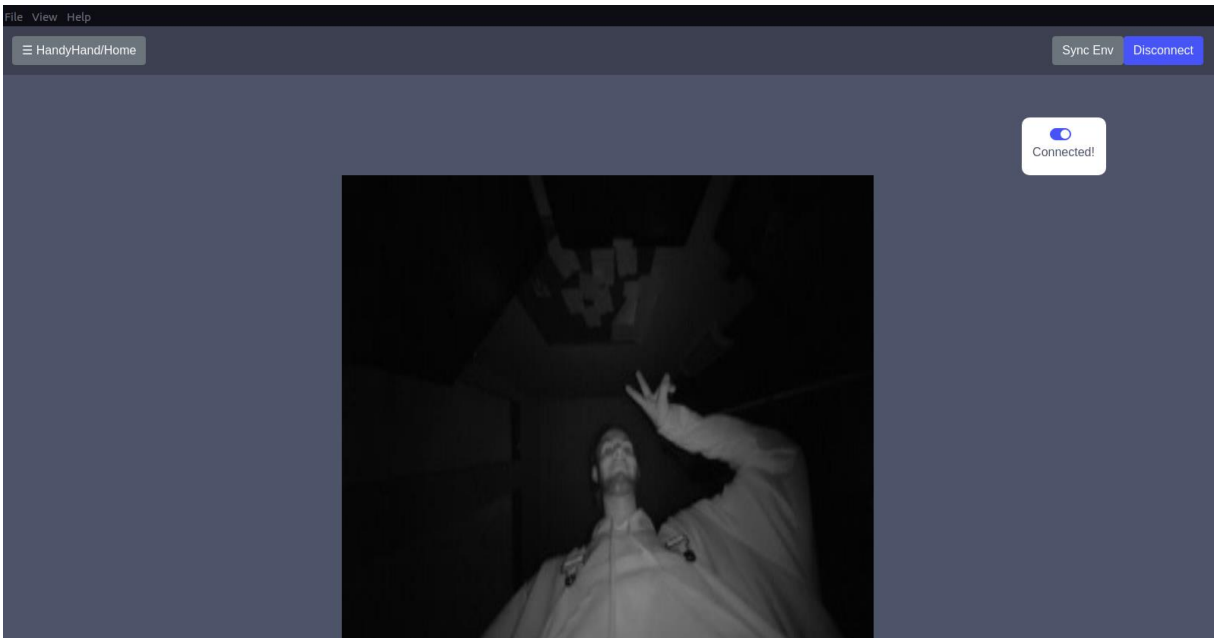
La rapidité d'exécution est aussi un point important dans l'exécution de l'application. Elle devra être capable de se démarrer en moins de 10 secondes. La récupération de l'environnement utilisateur devra elle aussi être rapide via un enchaînement de requêtes efficaces c'est-à-dire moins de 3 secondes et idem pour la sauvegarde de l'environnement. Pour finir, l'exécution de scripts lors de la reconnaissance d'un mouvement connu devra être faite en moins de 3 secondes elle aussi.

Pour résumer, nous cherchons à rendre cette application aussi accessible qu’efficace.

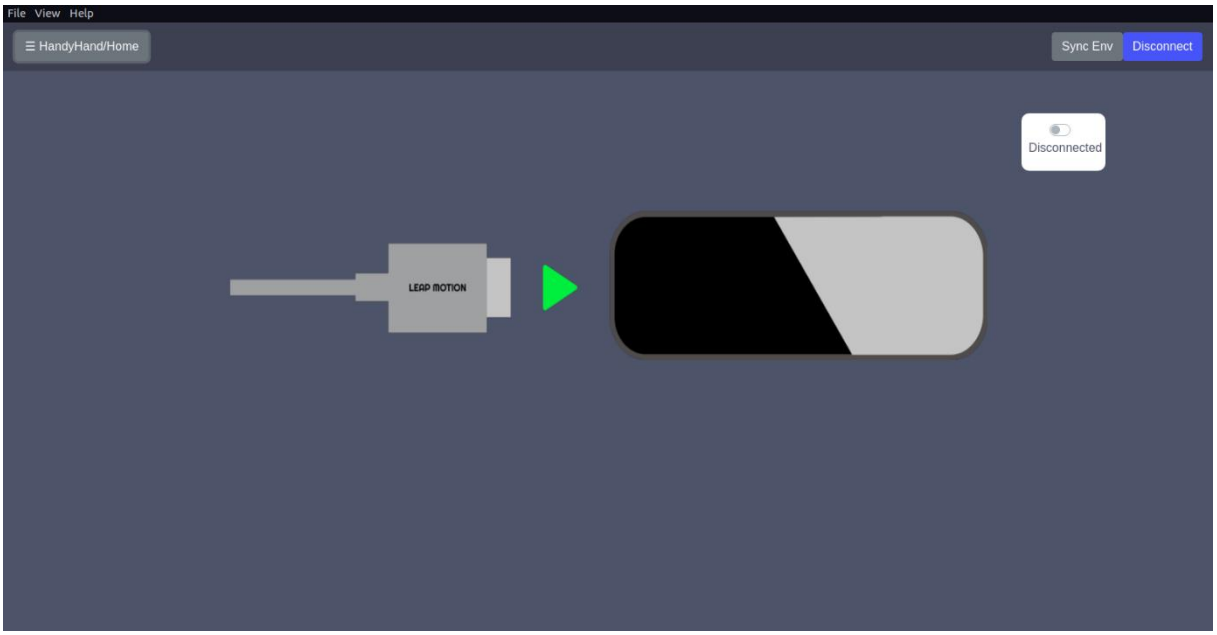
## IX.Visuels des vues



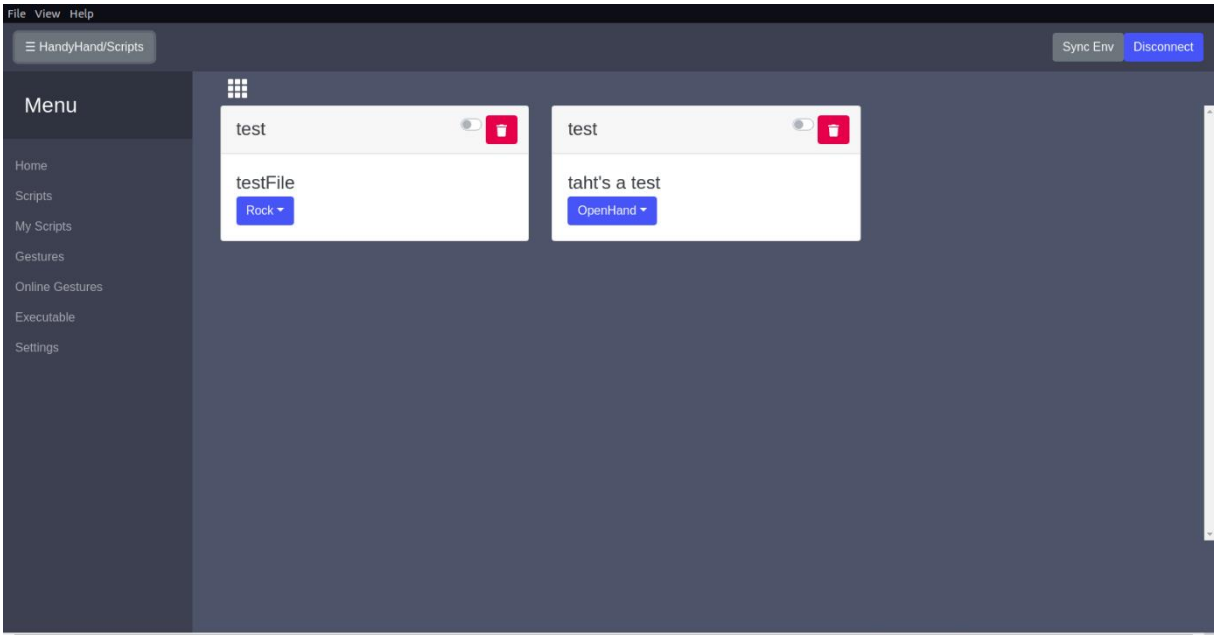

*Navigation*



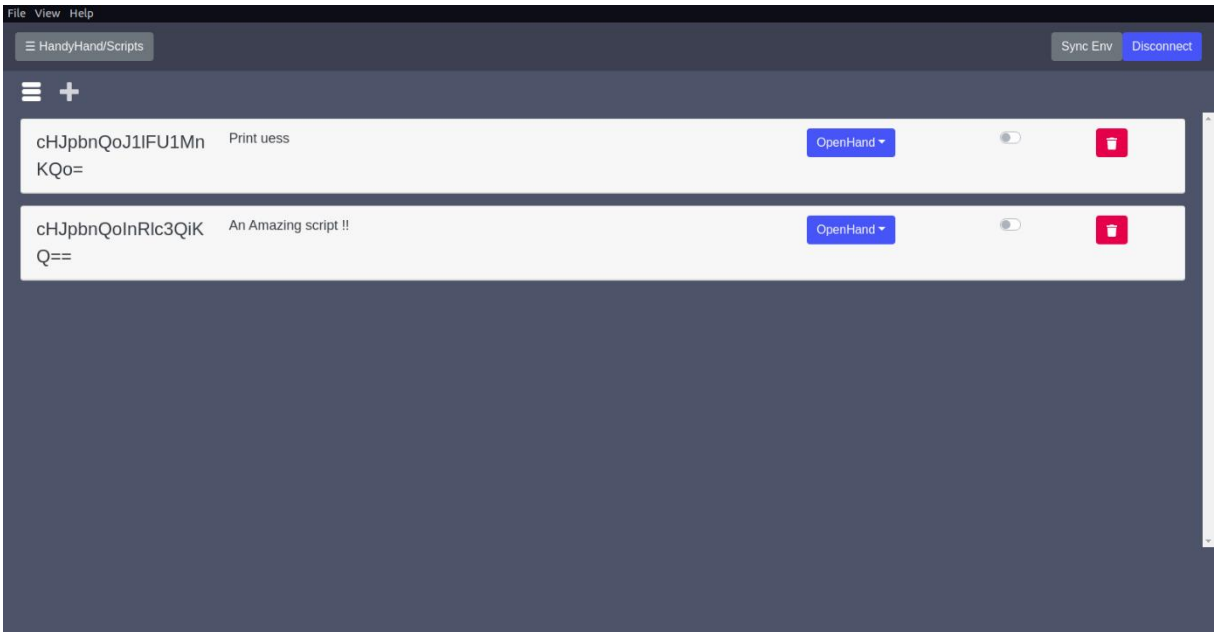

*Accueil avec Leap Motion* connecté*



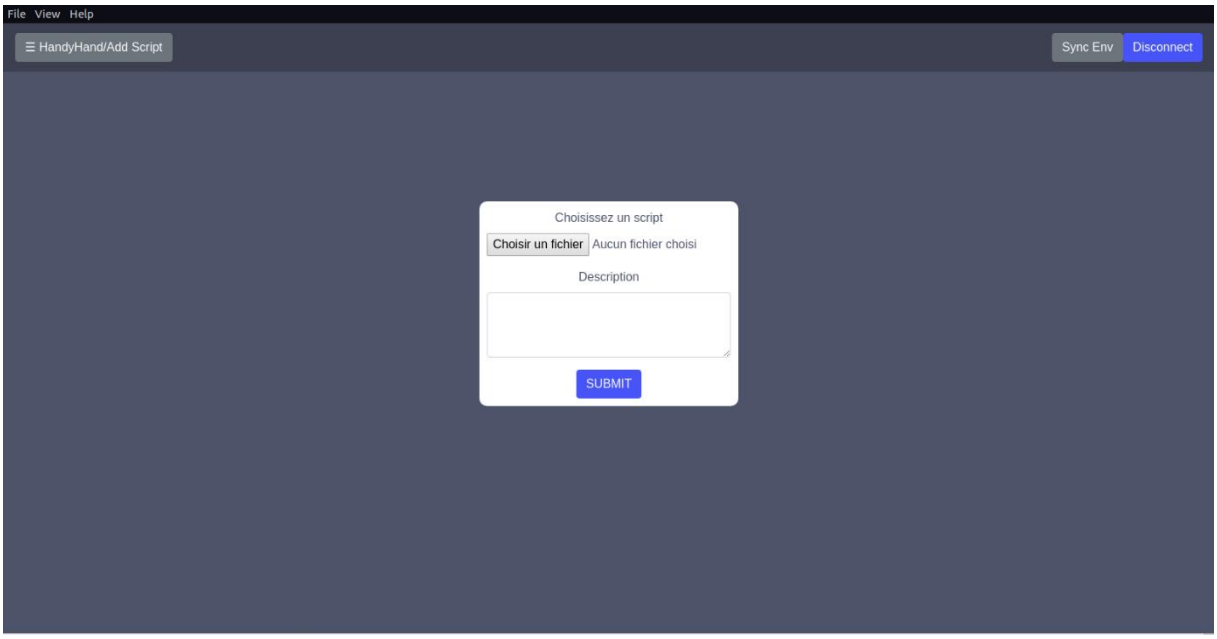

*Accueil avec Leap Motion* non connecté*



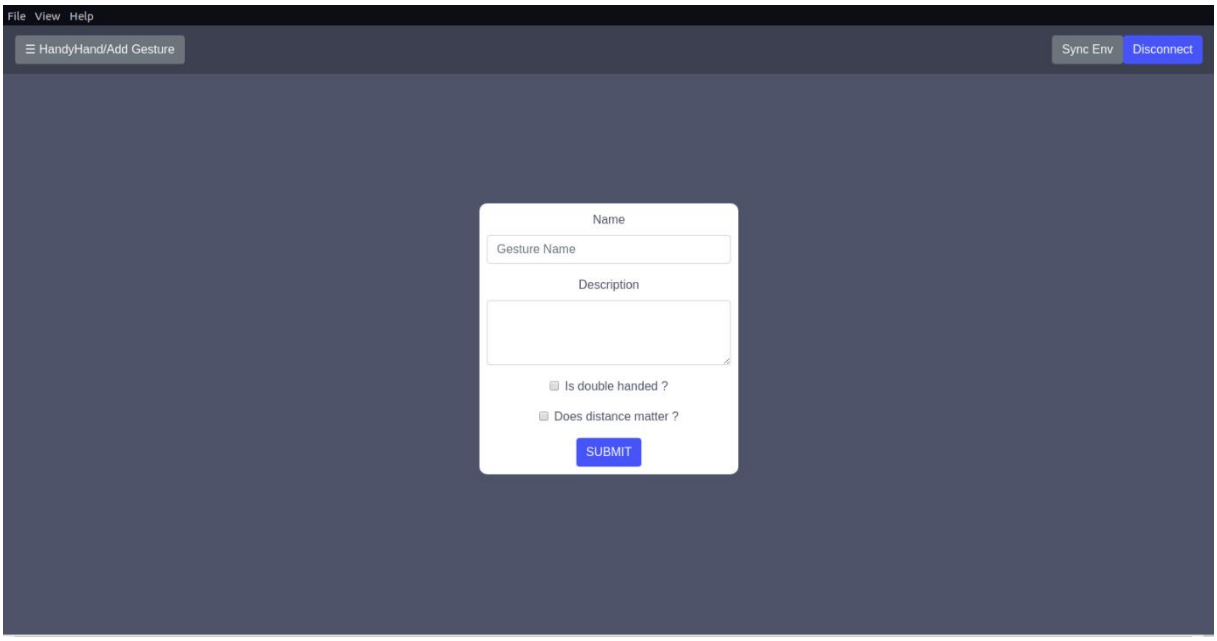

*Page Script*



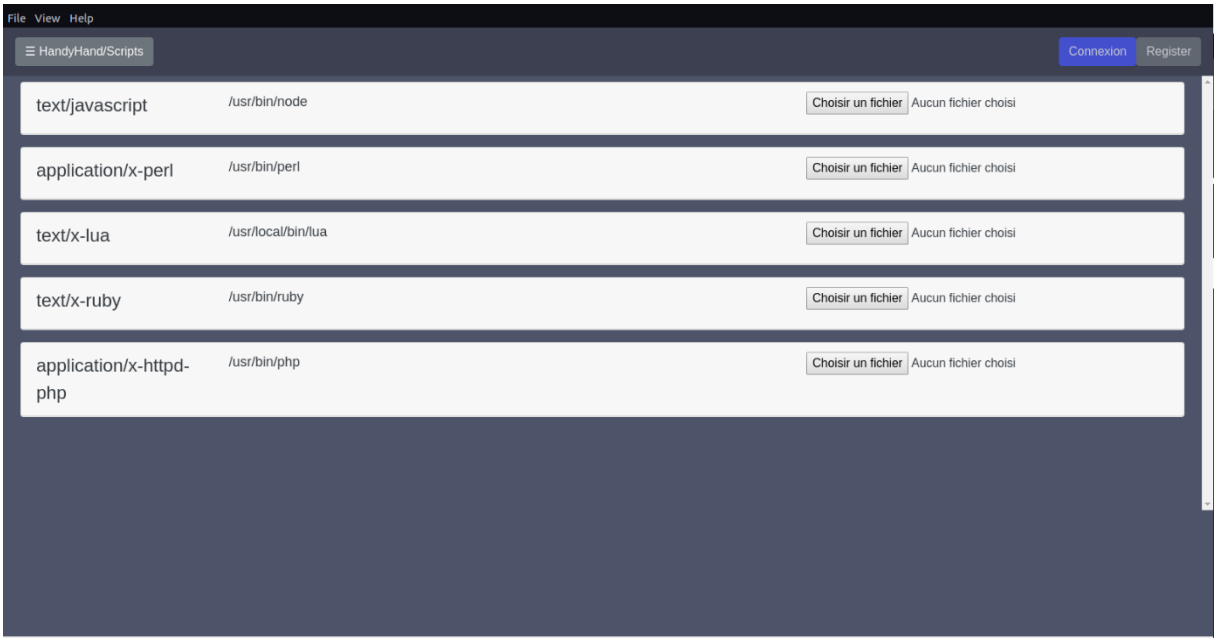

*Page Mes Script*



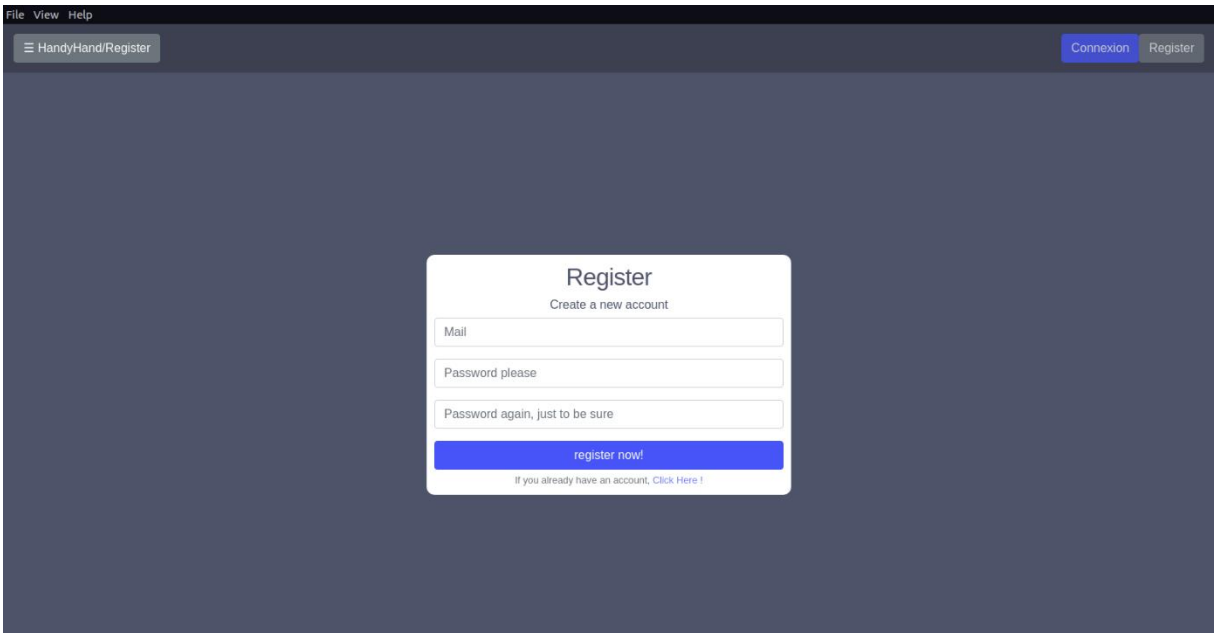

*Page ajouter un script*



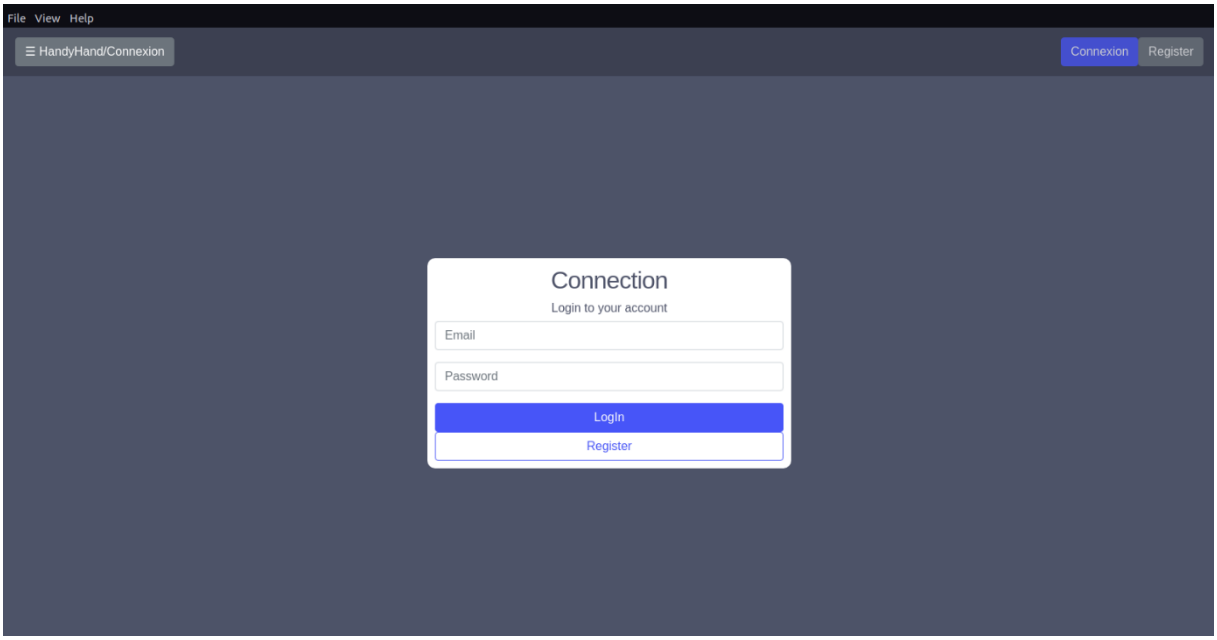

*Page ajouter une gesture*




*Page éxecutable*

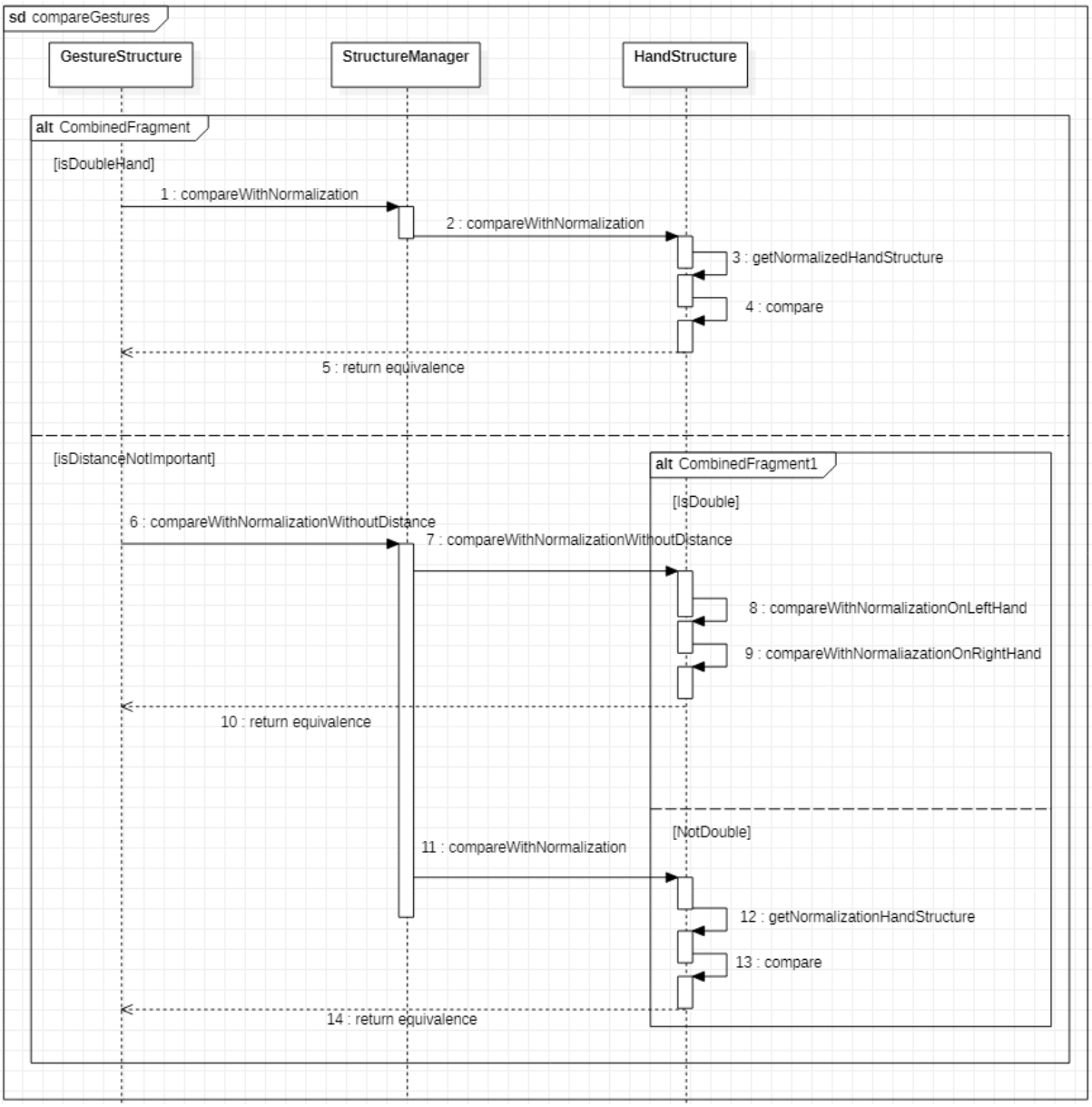



*Page enregistrement*




*Page connexion*

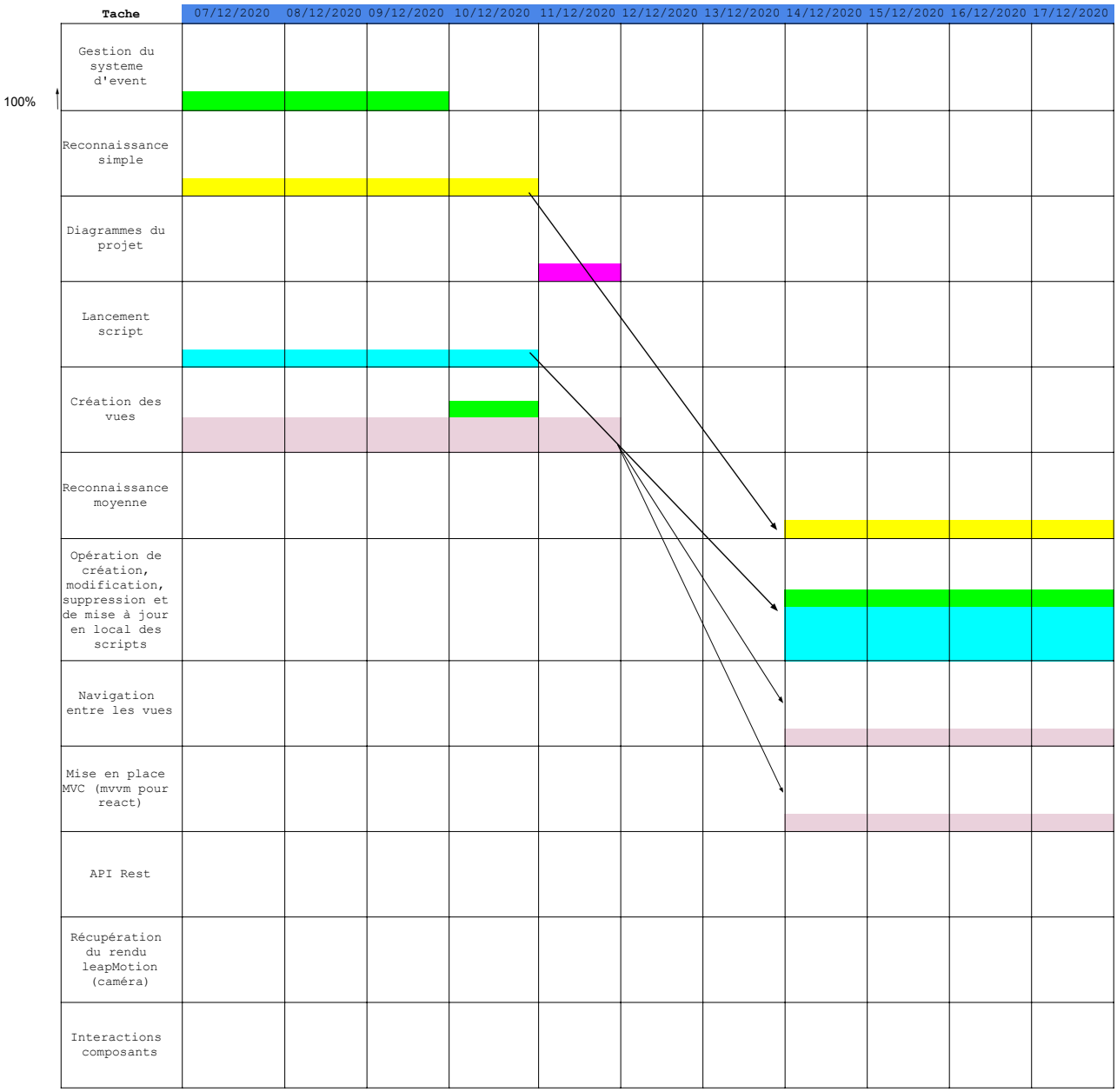
## X. Diagramme de séquence simplifiée correspondant à la méthode de reconnaissance :



## XI. Diagramme de GANTT pour la période du 07/12/2020 au 09/01/2021

**Projet 20**

**GANTT**




| **Légende :** | |
|---|---|
| Membres : | |
| Thomas | |
| Augustin | |
| Emrick | |
| Romain | |
| Yoann | |
| Equipes : | |
| FrontEnd | |
| BackEnd | |
| Dépendence : | → |

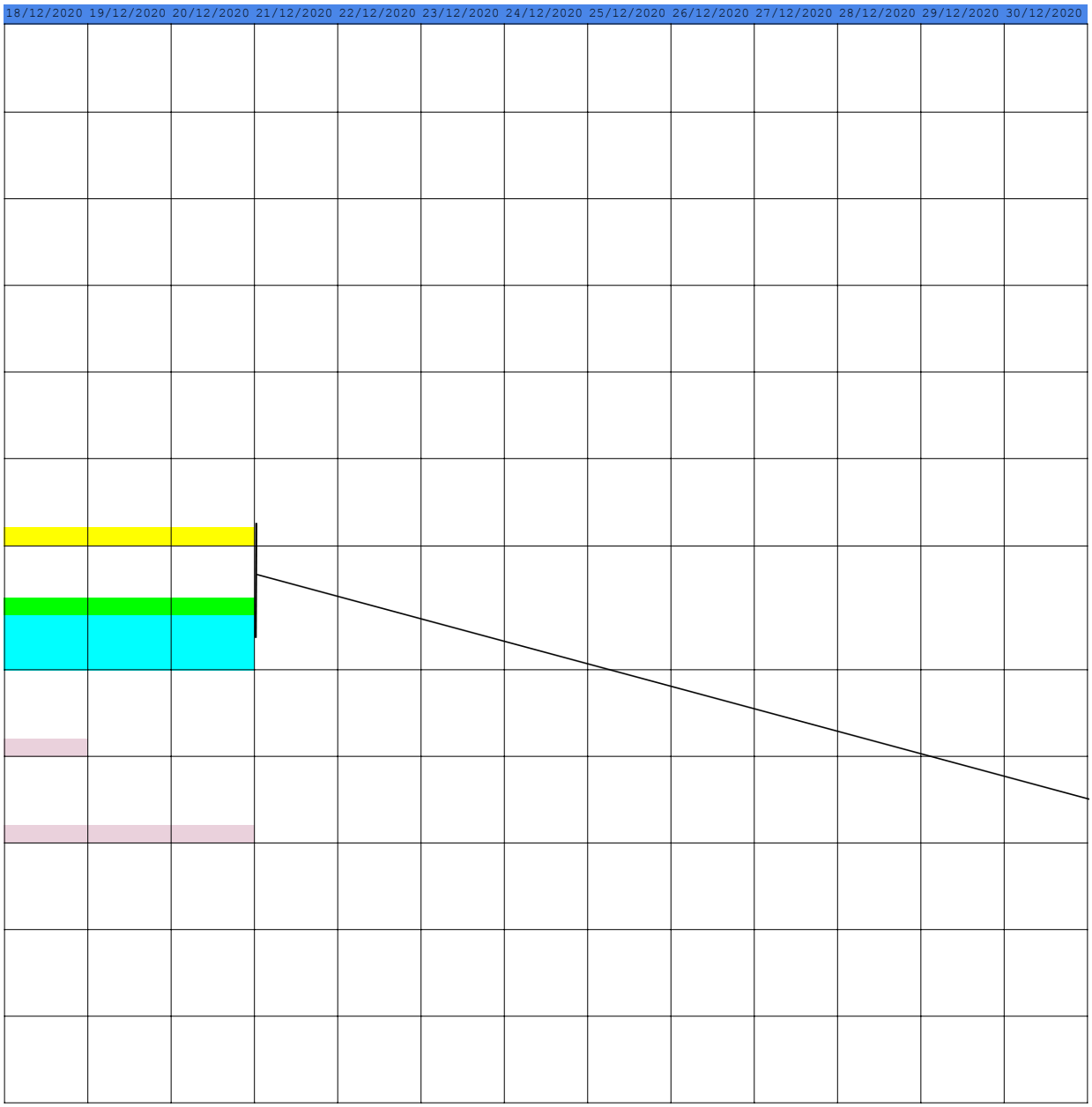
**Chemins critique :**

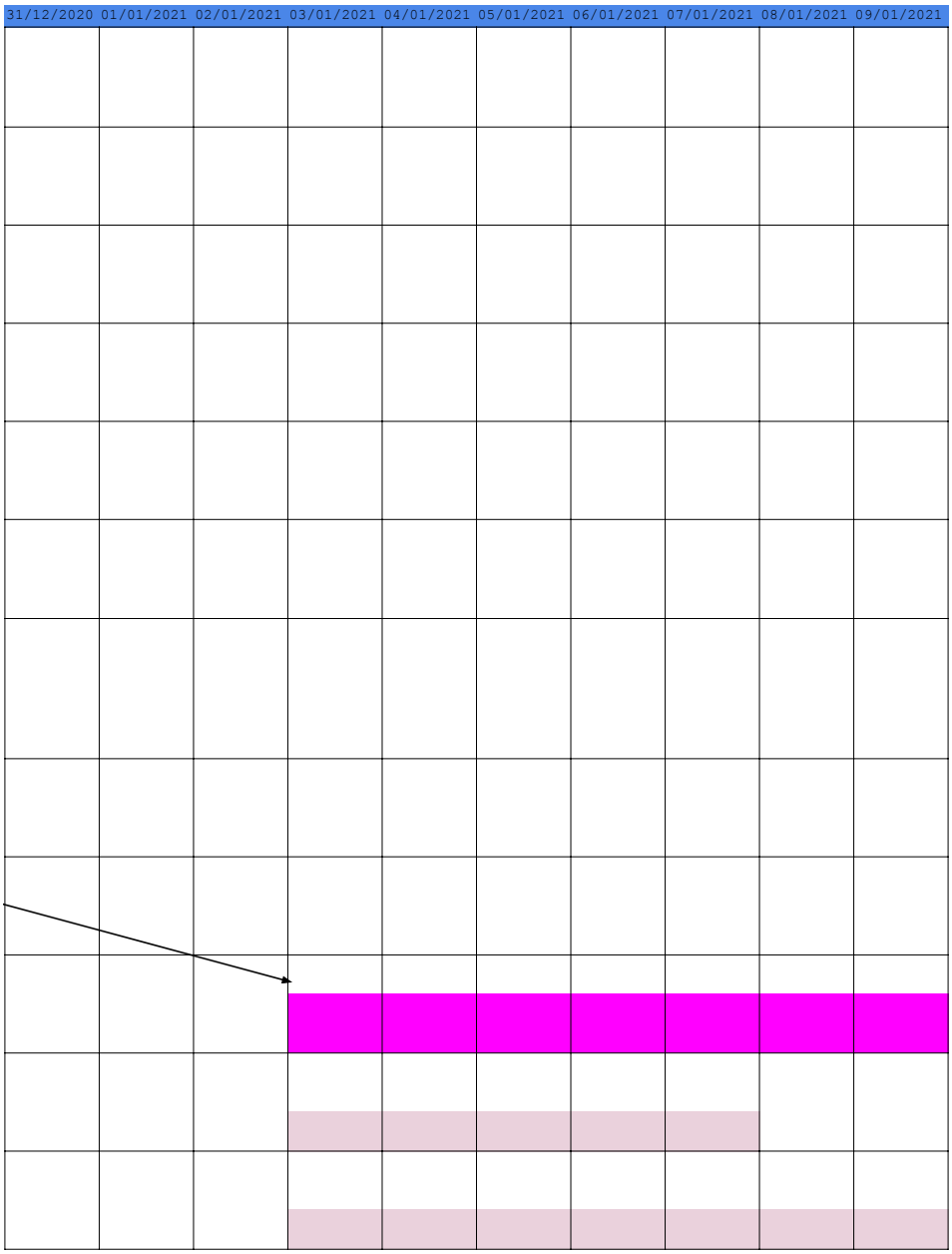
Reconnaissance simple -> reconnaissance moyen -> API Rest

Lancement script -> opération de création, modification, suppression et de mise à jour en local des scripts

Création des vues -> Navigation entre les vues/ Mise en place MVC (mvvm pour React)

La plupart des chemins sont critiques car les tâches ont une forte dépendence entres elles. Le développement de l'application découle d'un dévelopemment progressif nécessitant la fin de la tâche précédente avant de passer à la suivante.


18/12/2020
19/12/2020
20/12/2020
21/12/2020
22/12/2020
23/12/2020
24/12/2020
25/12/2020
26/12/2020
27/12/2020
28/12/2020
29/12/2020
30/12/2020


31/12/2020
01/01/2021
02/01/2021
03/01/2021
04/01/2021
05/01/2021
06/01/2021
07/01/2021
08/01/2021
09/01/2021