



VERA

# Guide du développeur

Pour prendre en main et maîtriser l'application

Design by Héloïse Jedaï BOUE YA

---

# AVANT-PROPOS

« *Un bon schéma vaut mieux qu'un long discours* »

Le **guide du développeur** est le témoin fidèle du développement et de la transformation sophistiquée du monde de la communication technique.

Ce guide est le fruit de constants efforts, des recherches, des mises au point qui en font un ouvrage de référence précis et rigoureux. Toujours actualisé, le **guide du développeur** du projet **Véra** s'enrichit à chaque nouvelle édition.

Nous nous sommes efforcés de réaliser un ouvrage digne de ceux auxquels il est destiné. La présentation générale, la mise en page et l'ordonnancement des textes, des figures, des tableaux et des couleurs sont une conception originale et la propriété intellectuelle de l'auteur.

Que soit ici remercié le Docteur Maxime DEVANNE pour sa contribution à l'enrichissement de cet ouvrage.

---

# PRÉFACE

Le présent guide explore les différentes fonctionnalités fournies par l'interface graphique **Véra** conçue dans le but de favoriser la visualisation des données issues de plusieurs capteurs. Il a pour objectif de vous aider à exploiter l'ensemble de ces fonctionnalités.

Ce guide s'adresse aux utilisateurs finaux de l'interface graphique **Véra** ainsi qu'à ses développeurs.

Cette documentation peut contenir des liens vers des sites Web de sociétés ou d'organisations indépendantes du projet **Véra** et hors de son contrôle. Nous effectuons aucune évaluation ou déclaration à propos de l'accessibilité de ces sites Web.

Les conventions de texte suivantes sont utilisées dans le présent document :

Convention	Signification
<b>Caractères gras</b>	Indiquent les sous-répertoires qui composent le projet <b>Véra</b> dans son ensemble.
<i>caractères italiques</i>	Indiquent les titres, les mises en évidences ou les variables pour lesquelles vous fournissez des valeurs particulières.

---

# TABLE DES MATIÈRES

<b>Avant-propos</b>	<b>i</b>
<b>Préface</b>	<b>ii</b>
<b>1 Introduction</b>	<b>2</b>
<b>2 Assistant d'installation de Véra</b>	<b>4</b>
2.1 Systèmes d'exploitation pris en charge . . . . .	4
2.2 Espace minimum requis . . . . .	4
2.3 Exécution de l'assistant d'installation . . . . .	4
<b>3 Mise en route de Véra</b>	<b>7</b>
3.1 Génération des binaires sous Unix . . . . .	8
3.2 Lancement de l'interface graphique sous Unix . . . . .	8
3.2.1 Le menu File . . . . .	9
3.2.2 Le menu Monitoring . . . . .	10
3.2.3 Le menu Tools . . . . .	16
3.2.4 Le menu Help . . . . .	17
3.3 La spécificité de Windows . . . . .	17

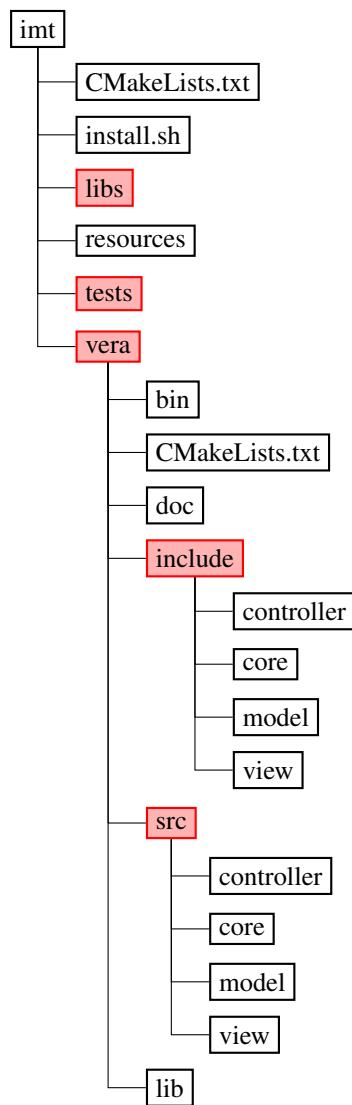
---

# CHAPITRE 1

---

## INTRODUCTION

**Véra** est une interface graphique qui a été conçue pour favoriser la visualisation des mesures de données issues de multiples capteurs, tout en étant indépendante des plateformes qui l'hébergent : elle est cross-plateforme. Par ailleurs, le projet **Véra** possède une structure hiérarchique favorisant son développement, son déploiement, son installation, son lancement ainsi que des tests unitaires.



L'application a été développée en s'inspirant du design pattern MVC (*Model View Controller*), avec le *modèle* qui serait chargé de gérer les données, la *vue* qui s'occuperait de l'affichage et le *contrôleur* qui aurait la charge de la logique du code et qui prendrait les décisions. Par ailleurs, une partie *core* a été adjointe pour assurer le traitement des données en lecture et en écriture.

Aussi, l'application **Véra** est orientée composants car reposant exclusivement sur le framework **Qt**, ce qui favorise son indépendance du point de vue système : elle a donc pour vocation d'être cross-plateforme, c'est-à-dire de fonctionner aussi bien sur un hôte de type Windows, MacOS ou une distribution GNU/Linux, si tant est que les dépendances logicielles soient satisfaites. Enfin, la gestion de l'ensemble du code source a été assurée par le service **GitHub** sur un dépôt **privé**.

---

# CHAPITRE 2

---

## ASSISTANT D'INSTALLATION DE VÉRA

Cette section présente le concept d'assistant d'installation de **Véra** qui se constitue suivant les plateformes hôtes soit en un script d'installation (*install.sh*) ou soit à un gestionnaire d'installation propriétaire.

### 2.1 Systèmes d'exploitation pris en charge

L'assistant d'installation prend en charge les systèmes d'exploitation **Kali Linux 2020.1**, **CentOS 8.1**, **Windows 10**, **macOS**, systèmes sur lesquels l'interface graphique **Véra** a été testée et dont nous assurons le bon fonctionnement.

### 2.2 Espace minimum requis

Fichiers	Espace minimum requis
/var	2 GiB ( <i>espace libre</i> )
/tmp	2 GiB ( <i>espace libre</i> )
/home	5 GiB ( <i>espace libre</i> )

Pour déterminer l'éligibilité de votre système GNU/Linux, nous vous recommandons d'utiliser la commande suivante :

```
1 [user@hostname ~]# df -h
```

### 2.3 Exécution de l'assistant d'installation

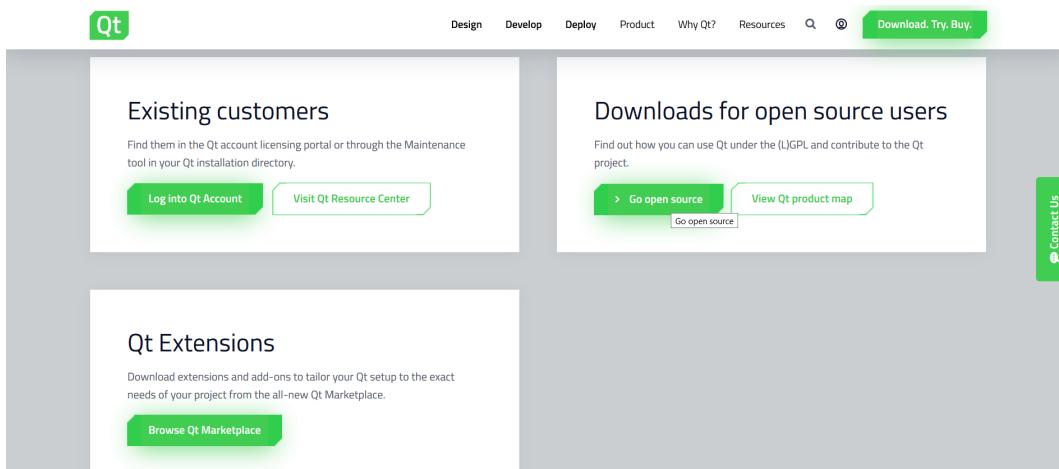
Pour exécuter l'assistant d'installation sur une distribution GNU/Linux, nous vous recommandons premièrement de vérifier que vous avez accès à internet en exécutant la commande suivante :

```
1 [user@hostname ~]# ping 8.8.8.8
```

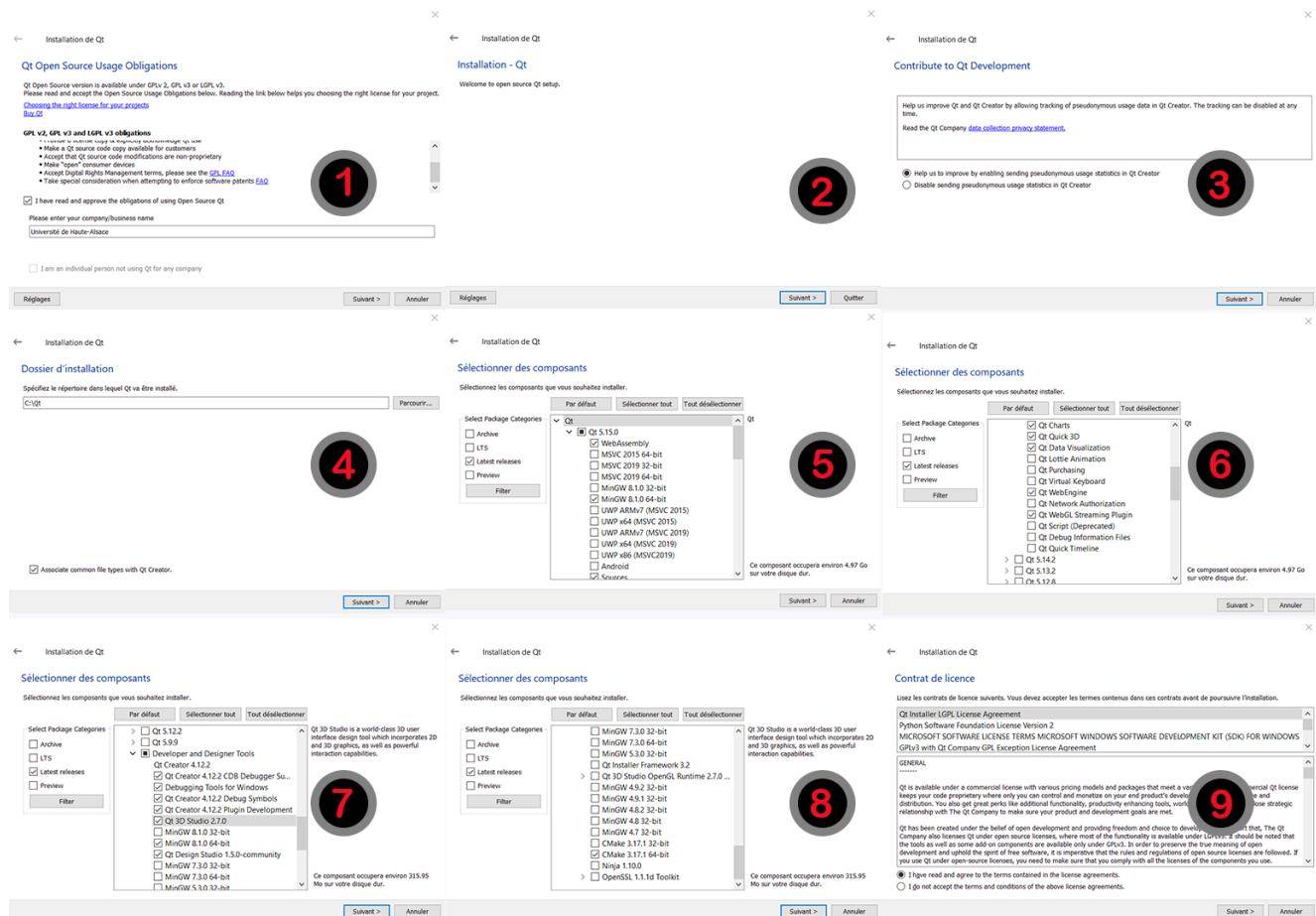
Une fois que le système hôte est connecté, exédez les commandes suivantes :

```
1 [user@hostname ~]# git clone https://github.com/Hethsron/imt.git
2 [user@hostname ~]# cd imt/
3 [user@hostname ~/imt]# chmod +x install.sh && ./install.sh
```

Pour ce qui est de Windows et de macOS, l'assistant d'installation consiste en un **installateur** téléchargeable fourni par les développeurs du framework **Qt**.



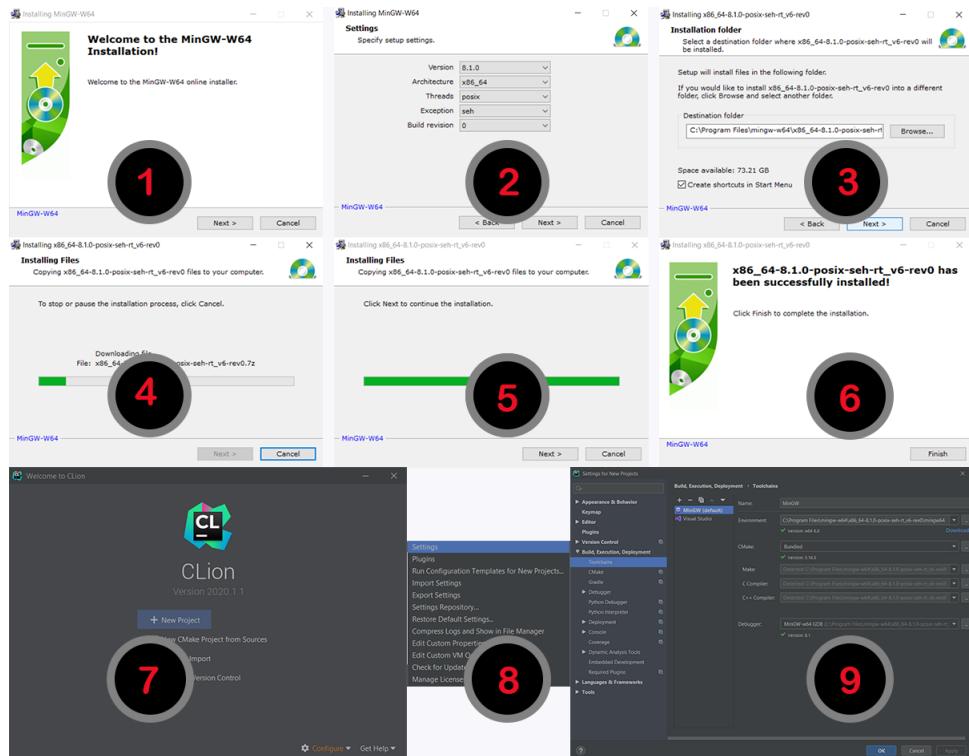
L'utilisation de cet installateur étant simpliste, les étapes à suivre qui concourent à l'installation des dépendances nécessaires au bon fonctionnement de l'interface graphique **Véra** sont décrites ci-dessous :



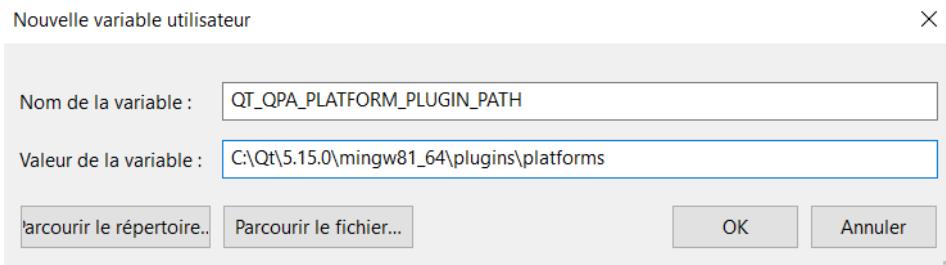
En outre, une fois les dépendances satisfaites, nous vous recommandons de vérifier uniquement dans le cas d'un hôte Windows, que la configuration suivante, définie dans les fichiers « **CMakeLists.txt** » des répertoires **vera** et **tests**, est équivalente à celle de votre plateforme.

```
1 set(CMAKE_PREFIX_PATH "C:\\Qt\\5.15.0\\mingw81_64")
```

Aussi, dans le cadre de Windows, nous vous recommandons vivement d'installer le compilateur **MinGW** ainsi que l'IDE **CLion** tout en respectant scrupuleusement les étapes suivantes :



Par ailleurs, la définition de la variable d'environnement nommée **QT\_QPA\_PLATFORM\_PLUGIN\_PATH** relative au répertoires des plugins ainsi que la génération du binaire avec CLion et la copie de certains fichiers DLL dans le répertoire **vera/bin** revêtent une importance capitale pour l'exécution de l'interface graphique sous Windows.



Les fichiers DLL à copier dans le dossier **vera/bin** sont ceux présent le cas échéant dans le répertoire :

1 C:\Qt\5.15.0\mingw81\_64\bin

Ces fichiers DLL sont :

1 Qt5Core.dll  
2 Qt5Gui.dll  
3 Qt5Help.dll  
4 Qt5Multimedia.dll  
5 Qt5MultimediaWidgets.dll  
6 Qt5Network.dll  
7 Qt5PrintSupport.dll  
8 Qt5Sql.dll  
9 Qt5Widgets.dll

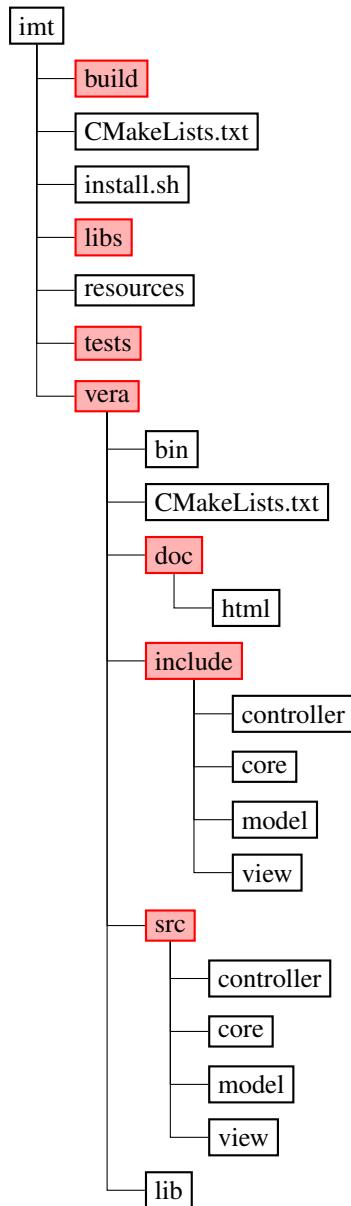
---

# CHAPITRE 3

---

## MISE EN ROUTE DE VÉRA

Cette section concerne l'utilisation de l'interface graphique **Véra**, lorsque l'assistant d'installation a été exécutée. Cette utilisation est rendue possible par la génération de certains binaires.



### 3.1 Génération des binaires sous Unix

Tout commence par la création d'un sous dossier nommé **build** par le moyen de la commande suivante :

```
1 [user@hostname ~/imt]# mkdir build
```

Cette étape est primordiale avant toute utilisation de l'interface graphique **Véra** dans un hôte. Une fois le dossier **build** créé, s'en suit la génération des binaires qui serviront à son utilisation. Aussi, par ce biais, une documentation en HTML (*HyperText Markup Language*) du code source, se basant sur la framework « **doxygen** », sera automatiquement générée dans le sous-dossier **html** dont le parent est le dossier **doc**. En outre, pour générer lesdits binaires, nous vous recommandons d'exécuter les commandes suivantes :

```
1 [user@hostname ~/imt]# cd build/
2 [user@hostname ~/imt/build]# cmake ..
3 [user@hostname ~/imt/build]# make
4 [user@hostname ~/imt/build]# cd ..
```

Par ailleurs, l'accès à la documentation du code source est rendu possible par le canal d'un navigateur web et/ou en exécutant la commande suivante :

```
1 [user@hostname ~/imt]# firefox vera/doc/html/index.html
```

### 3.2 Lancement de l'interface graphique sous Unix

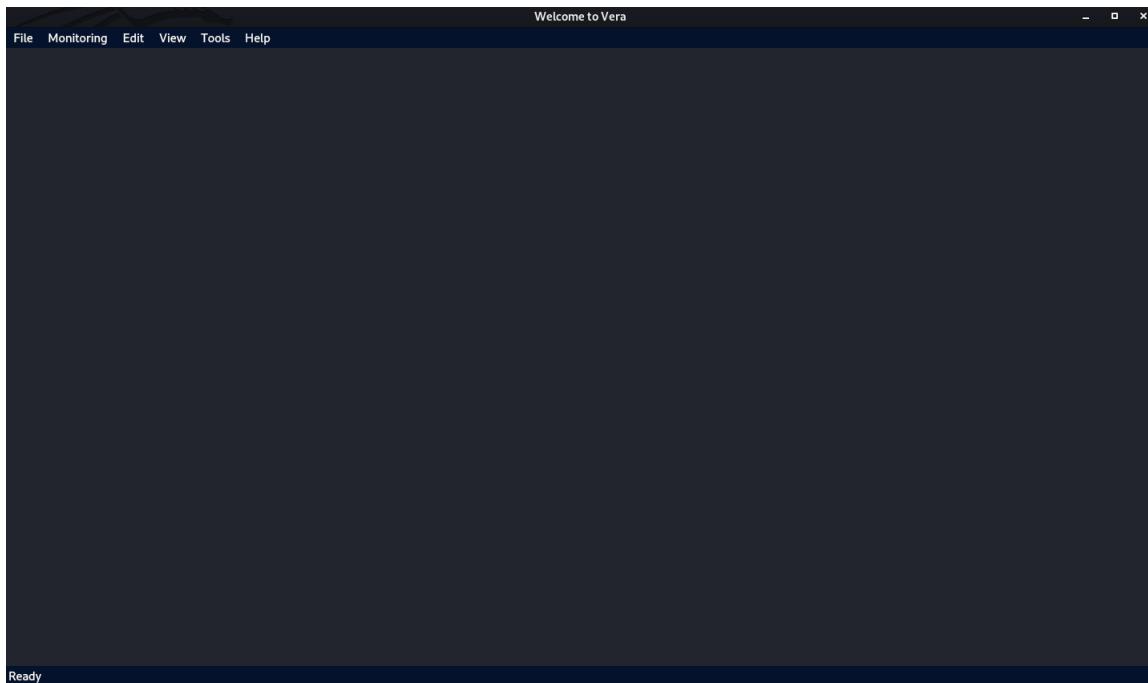
Le lancement de l'interface graphique susmentionnée a été largement simplifié. Il est rendue possible à l'unique condition que les binaires aient été générées et que l'ensemble des configurations susmentionnées ait été effectué. Ainsi, pour mettre en route l'application graphique **Véra**, nous vous recommandons d'exécuter les commandes suivantes :

```
1 [user@hostname ~/imt]# cd vera/bin/
2 [user@hostname ~/imt/vera/bin]# ./vera
```

Cette manipulation aura pour effet immédiat de lancer une fenêtre de démarrage qui se présentera comme suit :



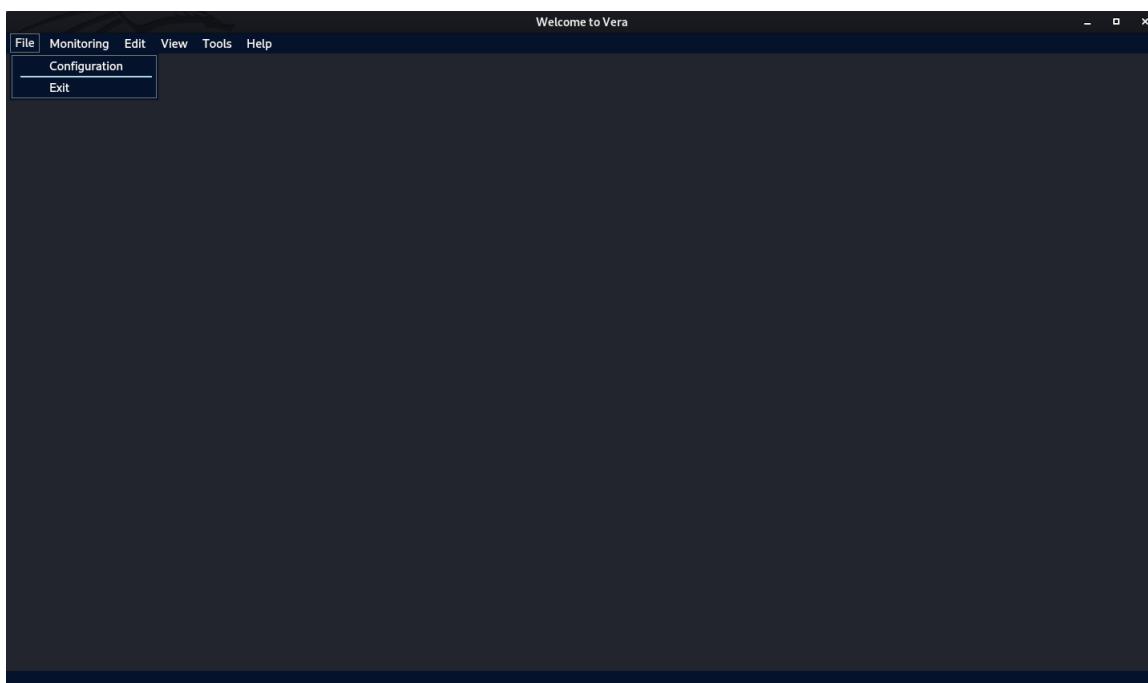
Cette dernière a pour but de favoriser le chargement des ressources nécessaires au bon fonctionnement de l'application, en toute sécurité. Une fois la barre de progression arrivée à 100%, la fenêtre principale de l'application sera visible. Celle ci se présentera comme suit :



L'interface graphique telle que présenté se compose d'une barre de menu, d'une barre de statut, d'une zone centrale pour la visualisation. La barre de menu comprend les menus **File**, **Monitoring**, **Edit**, **View**, **Tools** et **Help**, chacun d'eux ayant un rôle bien précis, que nous nous efforcerons à détailler par la suite.

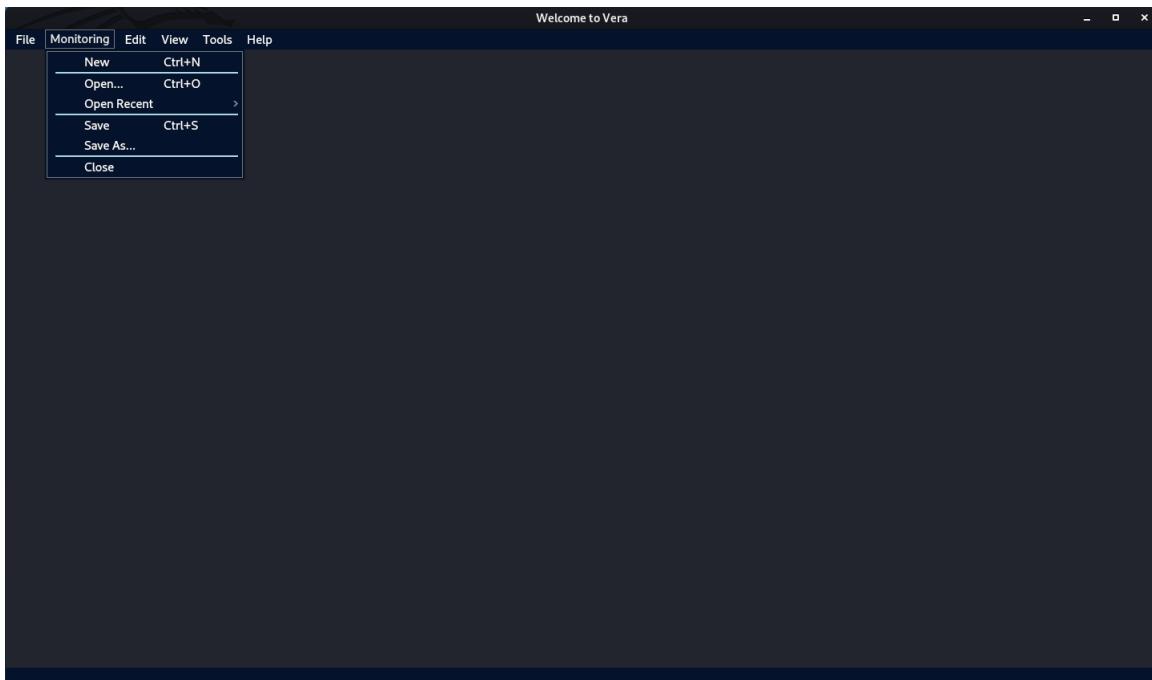
### 3.2.1 Le menu File

Le menu **File** comprend deux actions à savoir **Configuration** pour la configuration de l'interface graphique et **Exit** pour fermer l'application **Véra**. Ce dernier se présente comme suit :

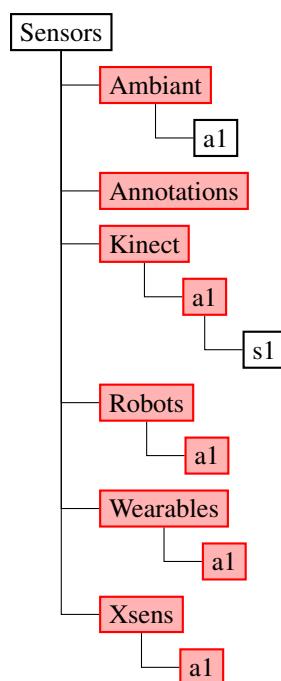


### 3.2.2 Le menu Monitoring

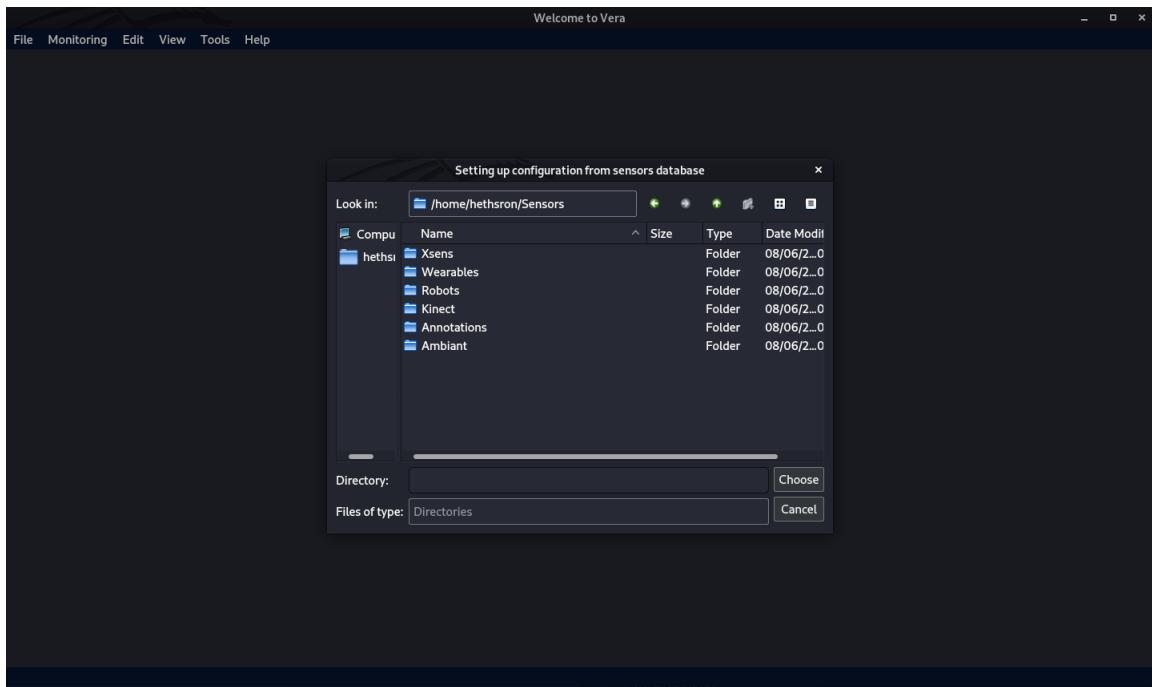
Le menu **Monitoring** est le menu prépondérant de l'application car il permet de satisfaire les objectifs de l'application en l'occurrence, ce pour quoi l'application a été conçue. Il se compose de six actions dont **New** pour choisir la base de données de capteurs permettant ainsi de créer un fichier de configuration relatif à cette dernière, **Open** pour ouvrir un fichier de configuration préalablement sauvegardé sur un disque, **Open Recent** pour ouvrir un fichier de configuration préalable sauvegardé dans la session courante, **Save** pour sauvegarder ledit fichier sur le système dans un répertoire par défaut choisi par l'application et sous un nom par défaut, **Save As...** pour sauvegarder ledit fichier sur le système dans un répertoire différent de celui choisi par défaut par l'application et sous un autre nom et, enfin **Close** pour fermer la visualisation.



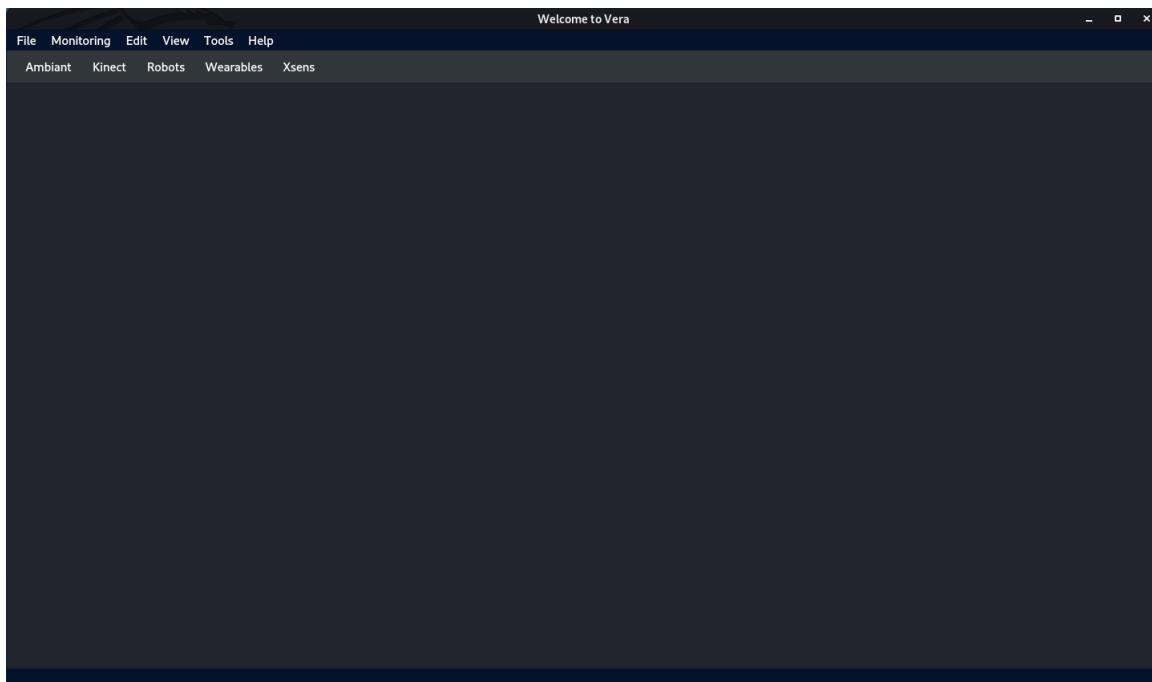
Par ailleurs, nous recommandons pour le bon fonctionnement de l'application que la base de données de capteurs soit organisée suivant l'architecture suivante :



Ainsi, l'action sur **New** ouvrira un explorateur qui permettra de choisir la base de données qui le cas échéant se nomme **Sensors** mais, qui dans d'autres circonstances pourrait avoir un autre nom.

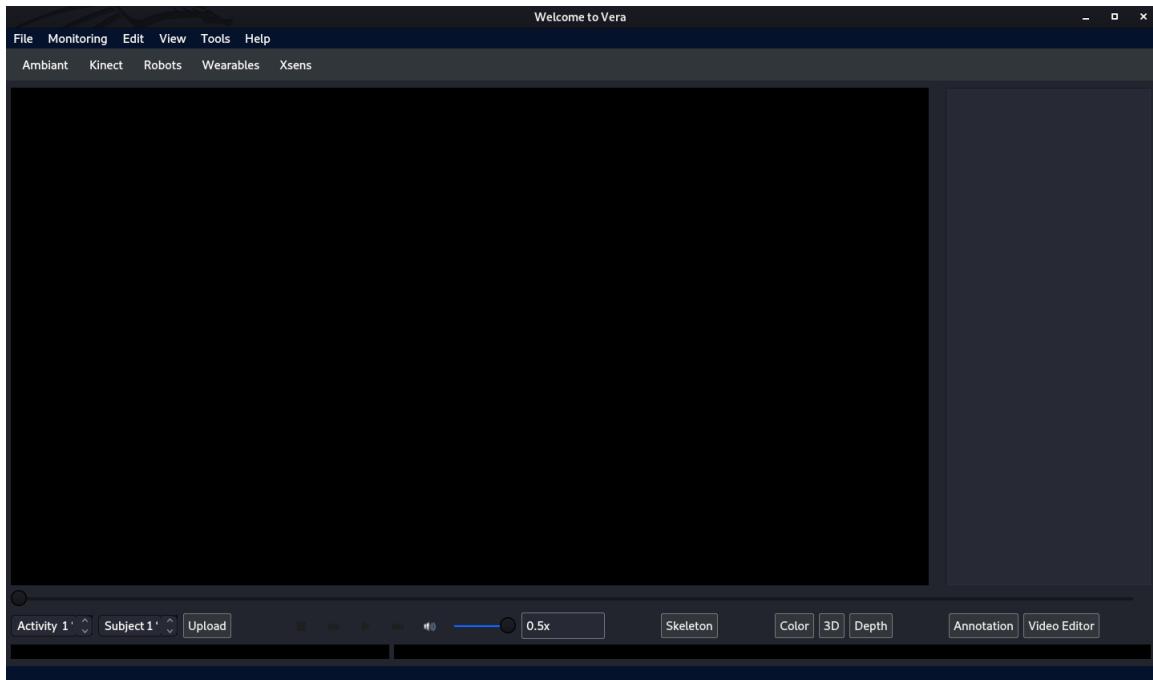


Vous l'avez compris, le plus important étant le nom des différents capteurs qui, nous tenons à vous rassurer, peuvent être en minuscule ou en majuscule : cela ne pose aucun problème. La création du fichier de configuration est faite sur la base des lettres qui composent les différents noms des dossiers. L'action sur **Choose** permet quant à elle ajouter une barre d'outils à la fenêtre principale qui se présentera comme suit :

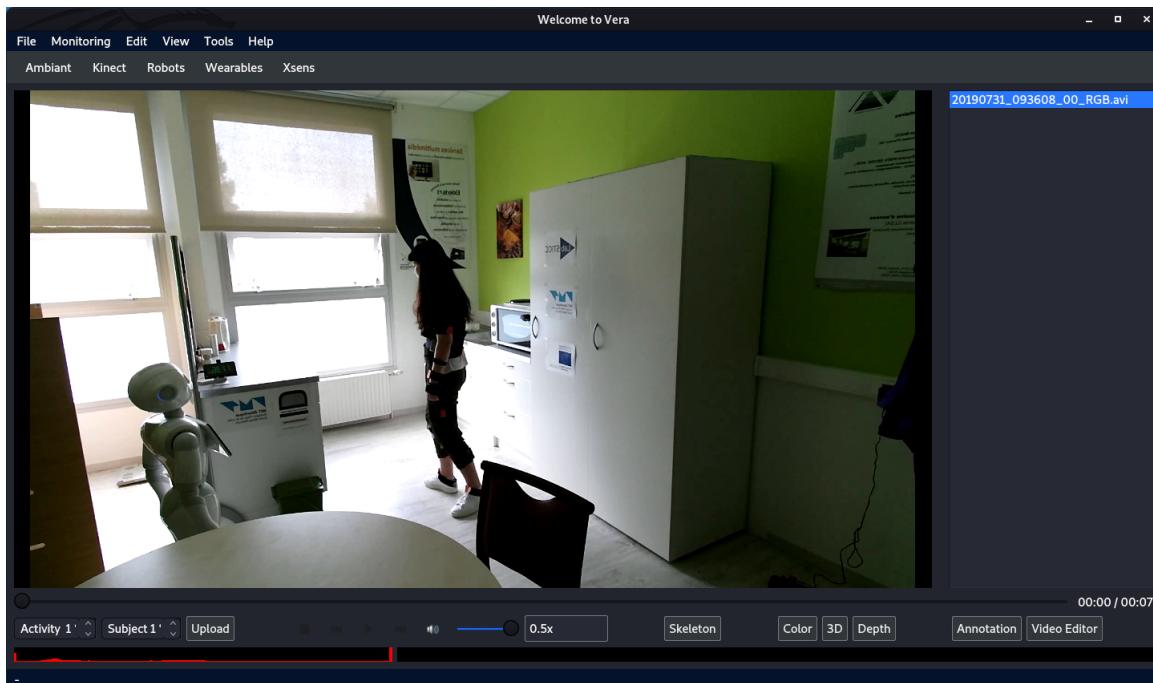


Cette barre d'outils se compose de cinq outils qui sont relatifs aux cinq capteurs dont traite l'interface graphique **Véra**, en l'occurrence **Ambiant**, **Kinect**, **Robots**, **Wearables** et **Xsens**. Chacun de ces différents outils a pour but de permettre à l'utilisateur de visualiser au moyen de l'application, sur la zone centrale, les données qui se réfèrent au capteur choisi.

A titre d'exemple, une action dans la barre d'outils sur le capteur **Kinect** aura pour effet de générer la vue suivante :

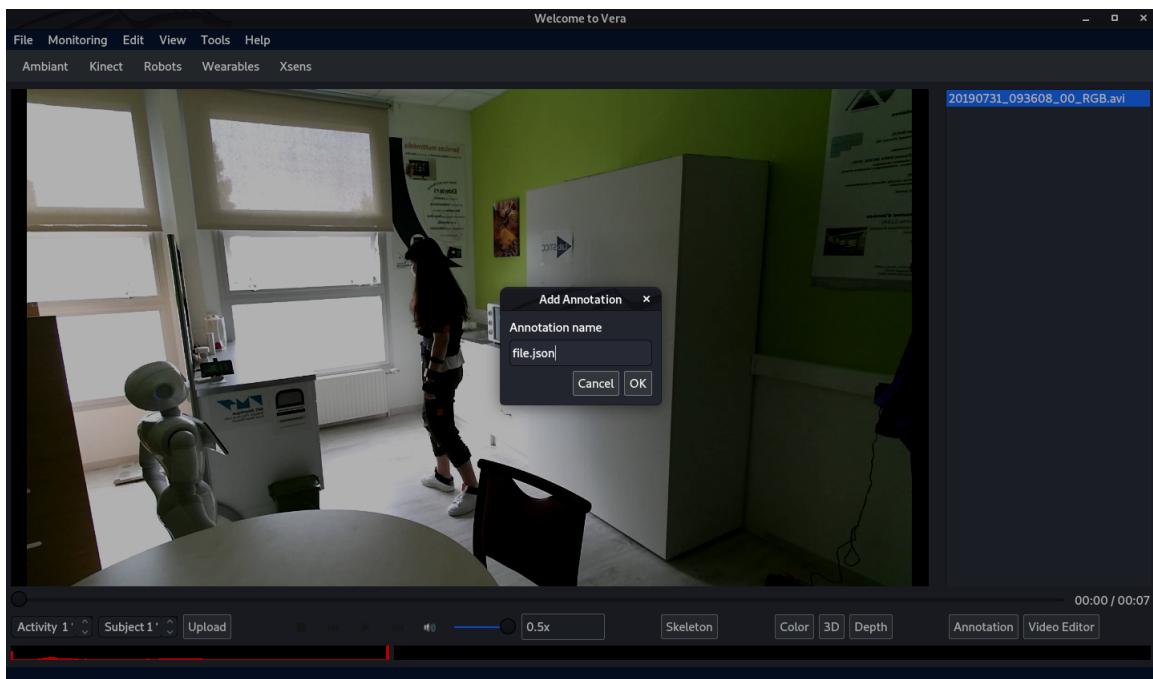


Dans cette configuration, l'utilisateur pourra choisir le numéro de l'activité dans **Activity** ainsi que celui à priori associé à l'expérience dans **Subject** qu'il souhaiterait visualiser ; le bouton **Upload** permettant de charger les données de l'expérience qui satisfont le choix de l'utilisateur. Le cas échéant, l'action sur le bouton **Upload** permettra de visualiser l'expérience suivante :

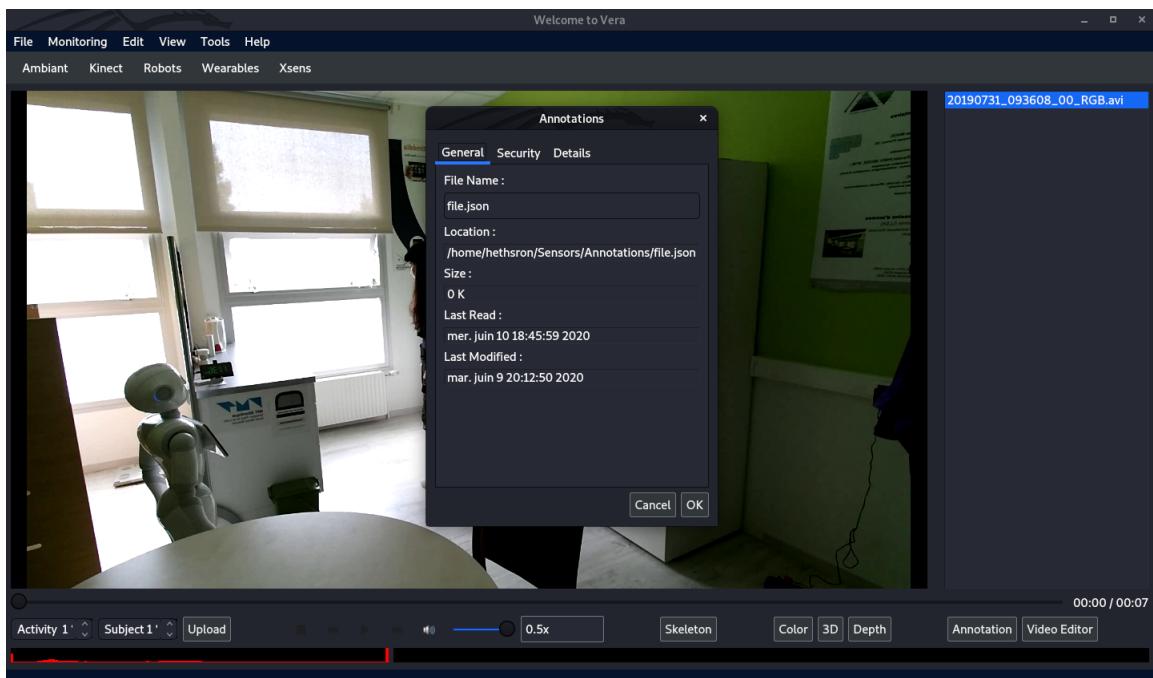


Les boutons **Skeleton**, **Color**, **3D** et **Depth** ont été prévus dans l'unique but d'offrir à l'utilisateur l'opportunité de visualiser respectivement le squelette de la personne traquée en deux dimensions, la salle observée en nuance de gris, en 3D ainsi que la profondeur de la salle. Les boutons **Annotation** et **Video Editor** ont été quant à eux prévus afin de respectivement annoter l'expérience et modifier l'expérience.

L'annotation est saisie en cliquant sur le bouton **Annotation**. Cette action a comme incidence le résultat qui se présente comme suit :

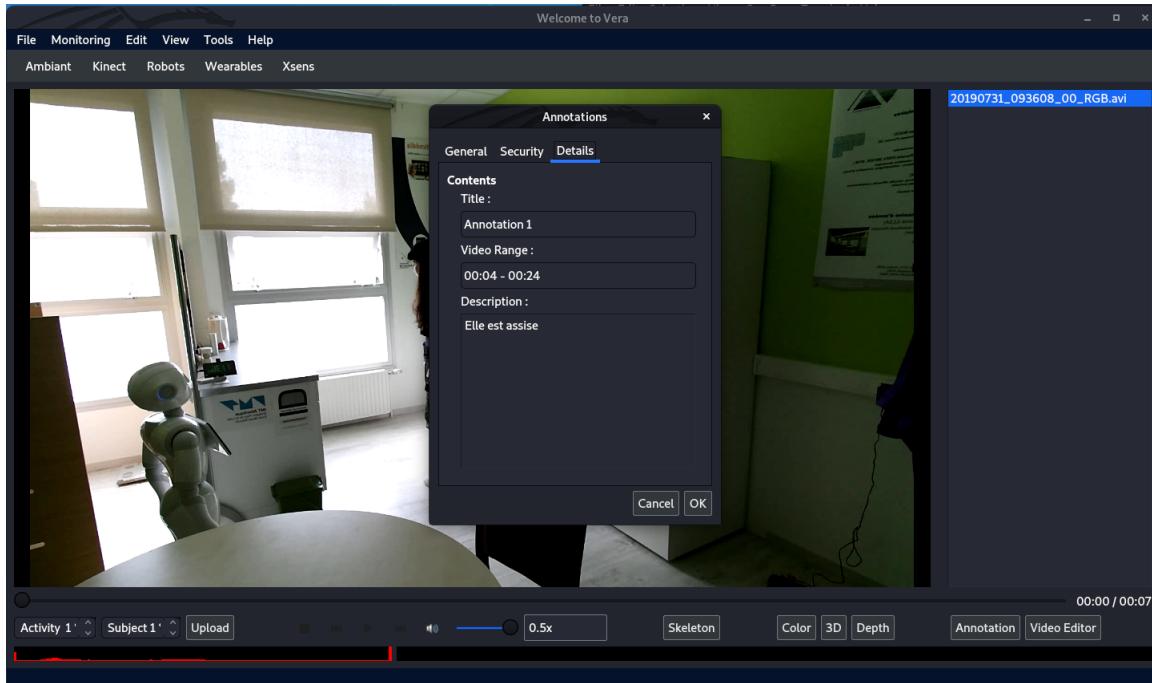


En effet, une annotation est un fichier qui se définit par son nom et son extension. Dans le cadre du projet **Véra**, les annotations sont des fichiers JSON. Ces derniers sont sauvegardés par l'application dans le dossier **Annotations** de la base de données de capteurs. Lorsque le fichier spécifié existe ses données sont chargées et peuvent ainsi être altérées. Dans le cas contraire, le fichier est créé avec les données qui le composent. Le cas échéant, le fichier «*file.json*» existant, l'action sur le bouton **OK** aura pour incidence de charger ses données et de fournir le résultat qui se présentera comme suit :

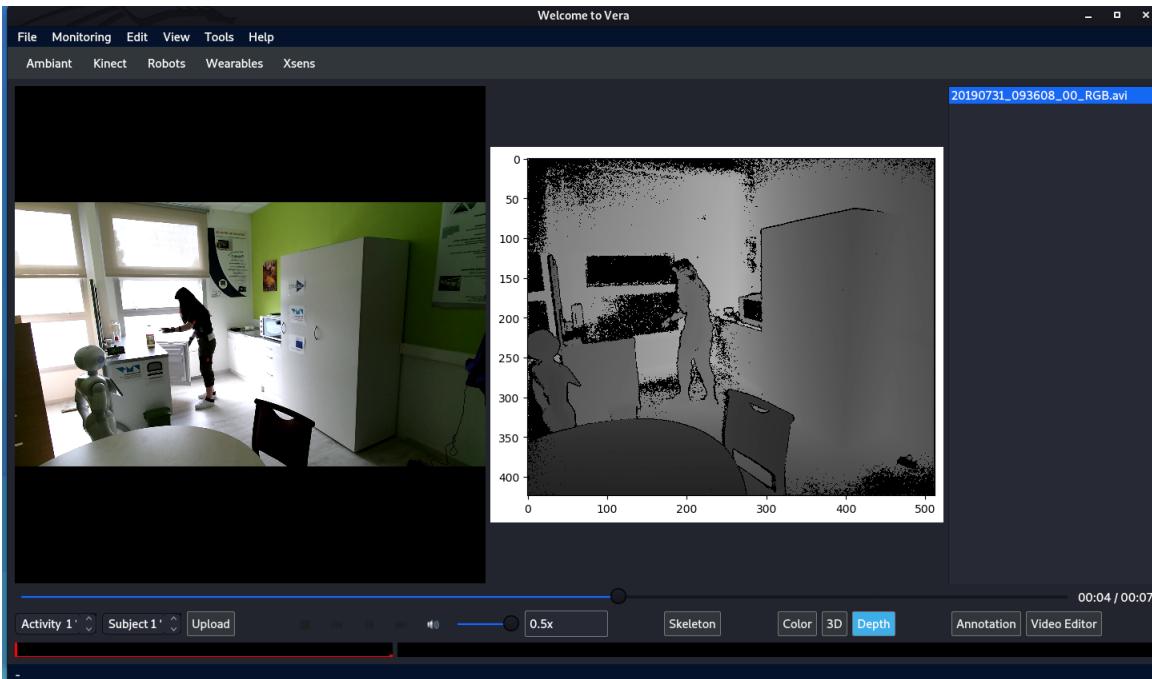


Les onglets **General**, **Security** et **Details** ont été prévues pour permettre à l'utilisateur de visualiser les caractéristiques du fichier représentant une annotation et d'inférer les informations constitutives à cette dernière.

Le cas échéant l'onglet **Details** comprend des informations qui bien entendu peuvent être altérées par l'utilisateur.



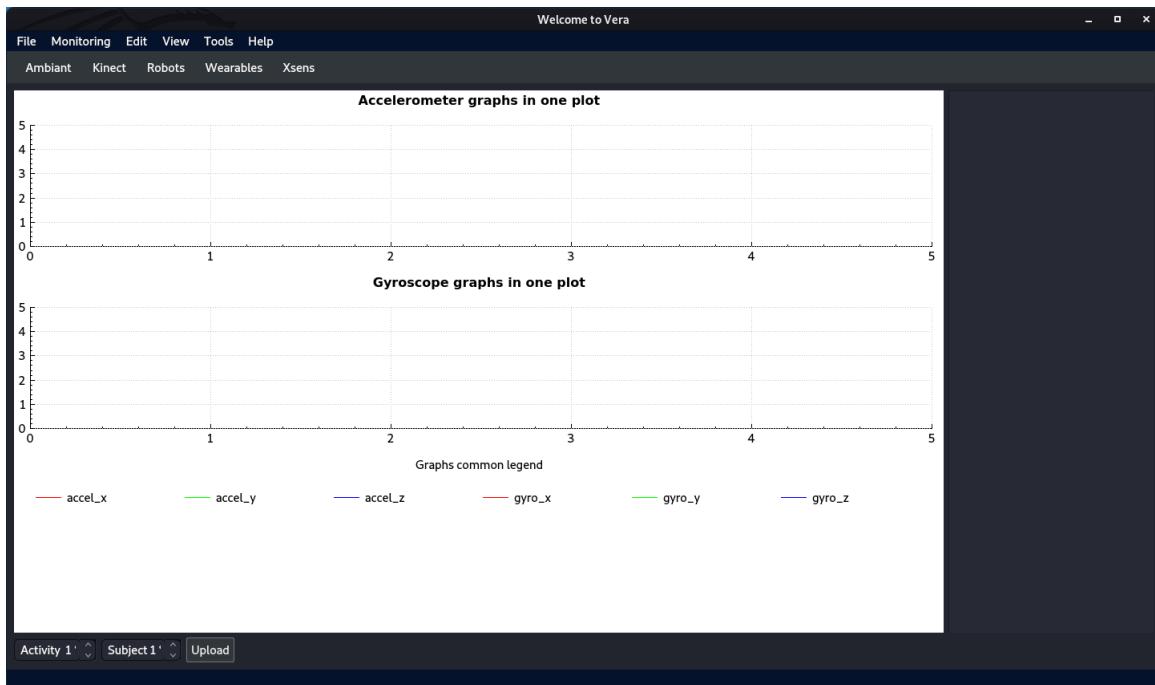
Par ailleurs, la profondeur est visualisée au moyen d'un clic sur le bouton **Depth**. Elle est synchronisée sur la lecture du flux vidéo, ce qui a pour incidence de produire le résultat suivant :



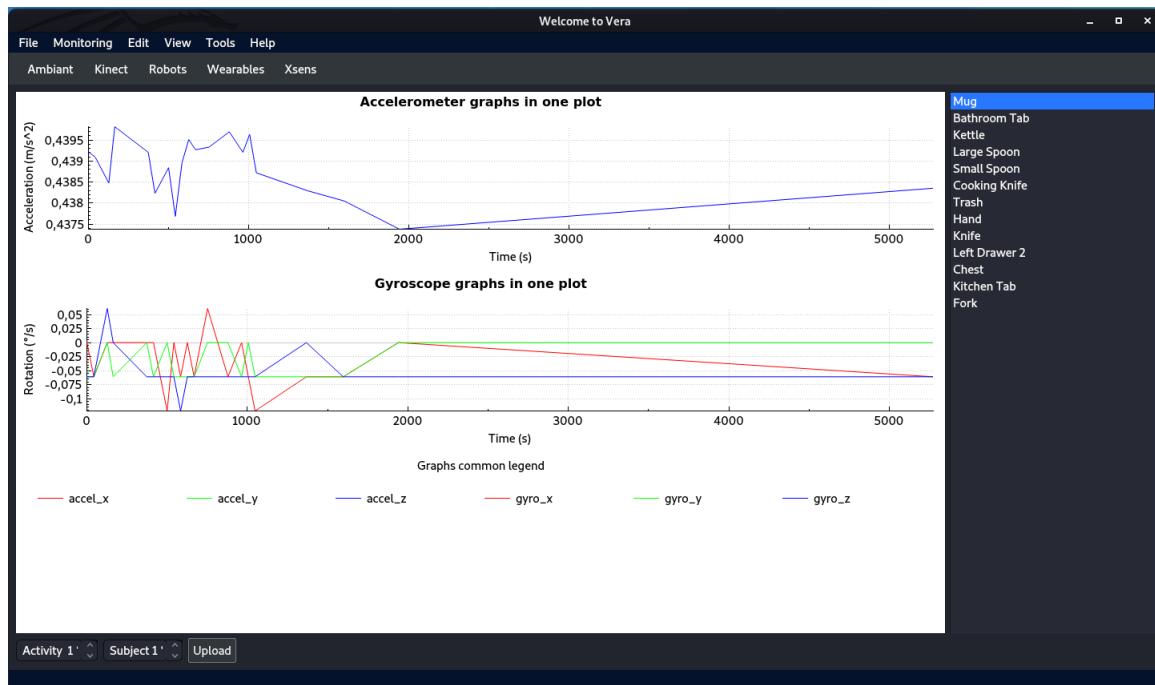
Par ailleurs, cette vue peut également être cachée, en cliquant pour une seconde fois sur le bouton **Depth**. Par cette astuce, l'utilisateur peut ou ne pas choisir de superposer les vues. Cette fonctionnalité supplémentaire permet de rendre l'application ergonomique.

Enfin, nous devons préciser que dans la mesure où il y'aurait plusieurs flux vidéos à visualiser qui sont relatifs à une seule expérience, il faudrait de nouveau cliquer sur **Depth** pour charger les bonnes données de profondeurs relatifs au flux vidéo en cours de lecture.

A titre également d'exemple, une action dans la barre d'outils sur le capteur **Wearables** aura pour effet de générer la page suivante :



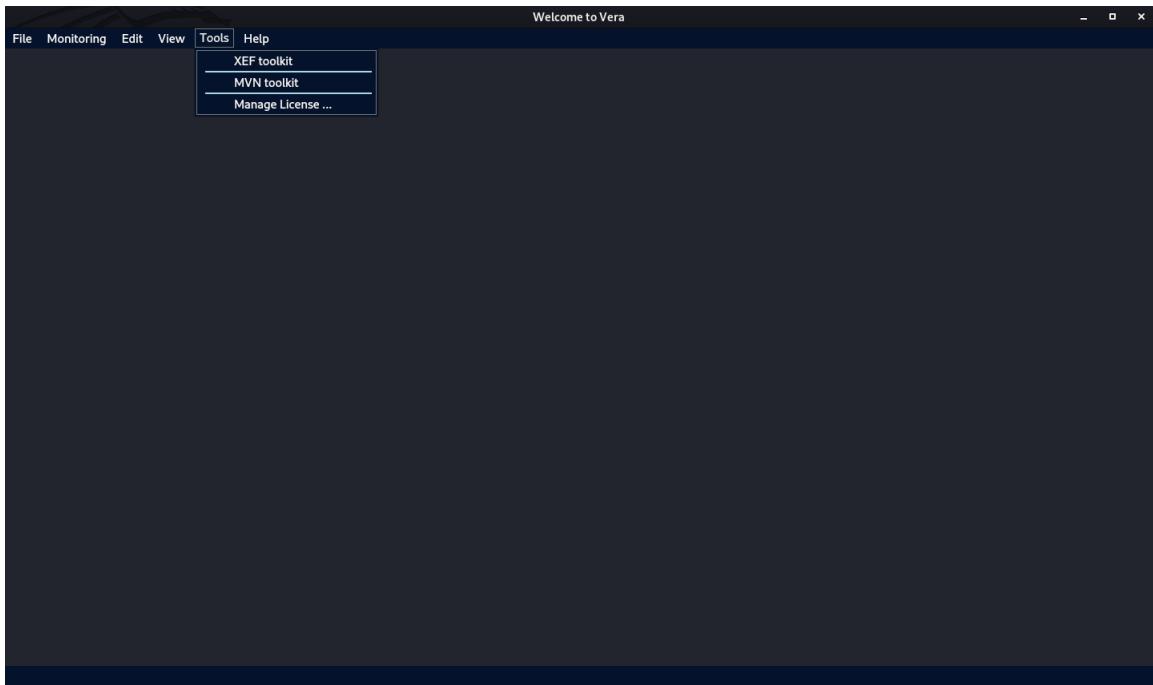
Dans cette configuration, l'utilisateur pourra choisir le numéro de l'activité dans **Activity** ainsi que celui à priori associé à l'expérience dans **Subject** qu'il souhaiterait visualiser ; le bouton **Upload** permettant de charger les données de l'expérience qui satisfont le choix de l'utilisateur. Le cas échéant, l'action sur le bouton **Upload** permettra de visualiser l'expérience suivante :



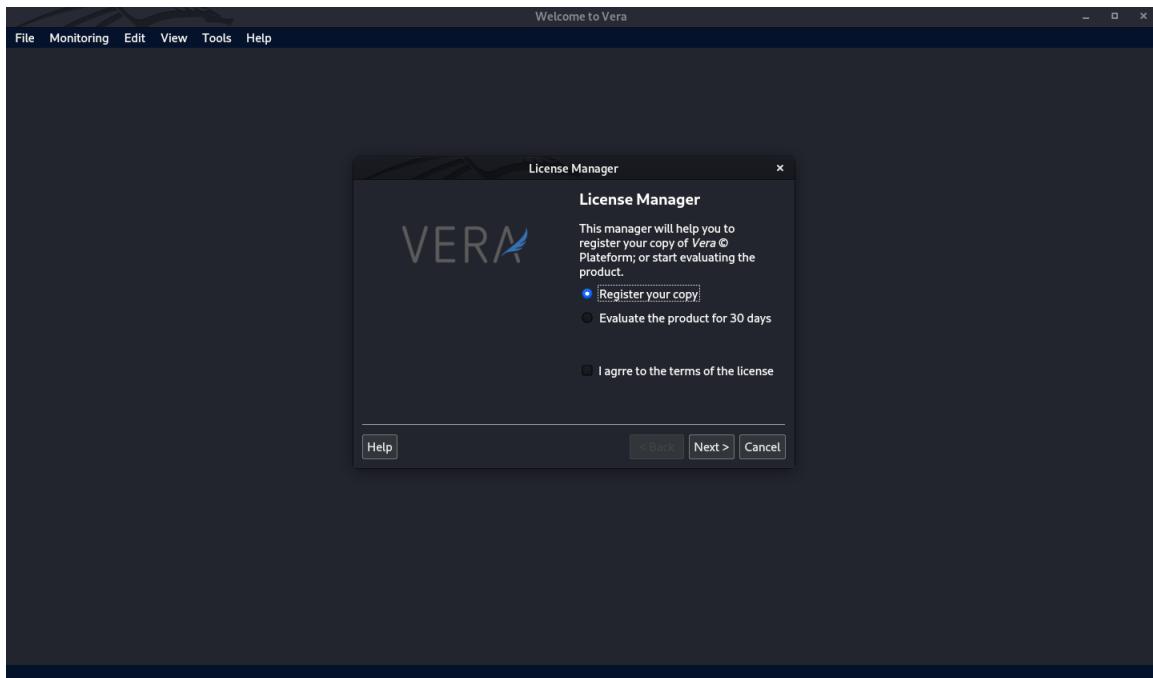
La liberté est ainsi donnée à l'utilisateur de cliquer sur les pièces que compose l'expérience dans l'unique but de visualiser les différentes mesures de données. Aussi, les graphes ont été conçus de manière à être interactifs afin de permettre à l'utilisateur d'agrandir ou de réduire la vue.

### 3.2.3 Le menu Tools

Le menu **Tools** est le menu qui a été prévu d'une part, pour intégrer voire annexer les outils propriétaires à l'application **Véra** et d'autre part, pour assurer la gestion de la licence de ladite application. Ainsi, il se compose de trois actions dont **XEF toolkit**, **MVN toolkit** et **Manage License**.



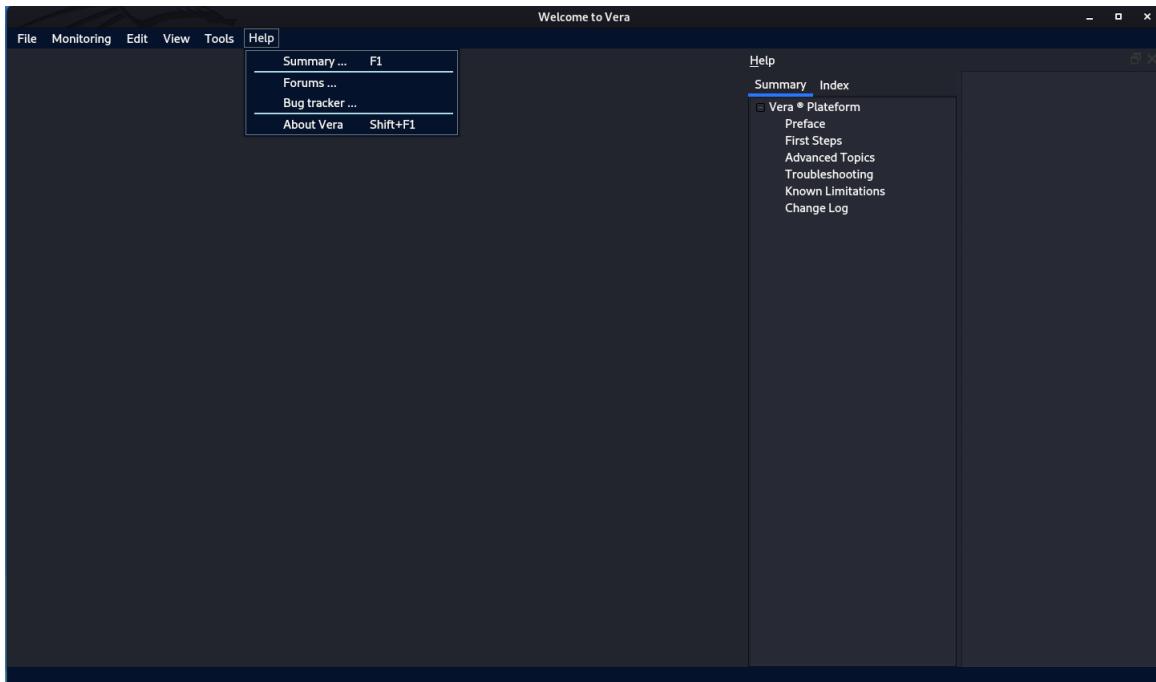
Ainsi, l'action sur **Manage License** permet d'ouvrir un gestionnaire de licence qui permet le cas échéant, d'une part d'enregistrer l'application de manière définitive avec une clé du produit et, d'autre part de l'évaluer pendant une durée de 30 jours.



Par ailleurs, les actions **XEF toolkit** et **MVN toolkit** ont été prévues pour assurer la gestion des formats propriétaires de fichiers que sont les formats XEF et MVN, en favorisant leur conversion en des formats universels.

### 3.2.4 Le menu Help

Le menu **Help** est le menu qui a été prévu pour être un support didactique pour l'utilisateur. Il se compose de quatre actions dont **Summary** qui donne accès à un didacticiel, **Forums** qui est prévu pour donner accès à l'utilisateur à un espace communautaire lui permettant de trouver des réponses aux problématiques rencontrées lors de l'utilisation de **Véra**, **Bug tracker** qui est prévu pour reporter des anomalies ou bogues et **About Vera** pour des informations constructives.



## 3.3 La spécificité de Windows

Sur un système d'exploitation Windows, la génération des binaires ainsi que le lancement de l'interface graphique sont combinés grâce au concours d'un environnement de développement intégré, le cas échéant **CLion**, pour donner ceci :

