14/06/2021

Louis-auguste Dumas

DOlisquad

Master API conception

Table des matières

[Description des fonctionnalités : 2](#_Toc74585354)

[Définition des besoins : 3](#_Toc74585355)

[1) API 3](#_Toc74585356)

[2) Contrôle 3](#_Toc74585357)

[3) Sécurité 3](#_Toc74585358)

[Solutions : 4](#_Toc74585359)

[1) Les librairies 4](#_Toc74585360)

[2) Les urls 4](#_Toc74585361)

[3) Solution technique 5](#_Toc74585362)

[Annexe 6](#_Toc74585363)

# Description des fonctionnalités :

L’API master joue un rôle central au sein du projet. Elle me permet à l’ensemble des clients peut importe leur technologie de communiquer avec le back logique.

Elle doit être capable de servir d’interface pour les clients pour le contrôle manuelle.

Elle doit aussi être capable d’activé des fonctionnalités. Comme le « mode toutou » et le « mode patrouille ».

Pour rappel, le « mode toutou » permet de suivre l’utilisateur automatiquement à partir de la reconnaissance faciale.

Le « mode patrouille » permet de faire patrouiller le drone dans une pièce définit. Et qui cherche des utilisateurs et prévoit un bruit en fonction de ce dernier.

Elle doit aussi être sécurisé pour éviter que qu’un utilisateur non autorisé puisse contrôler le drone.

# Définition des besoins :

## API

L’api master comme son nom l’indique est une API. Elle doit être capable d’accepter plusieurs requêtes par secondes. Et d’effectuer les besoins dans un délaie impartie.

## Contrôle

L’api doit offrir toutes les fonctionnalités pour répondre aux exigences de performances mais aussi répondre aux besoins. Qui sont l’activation des fonctionnalités et le contrôle manuel.

Pour rappel, le « mode toutou » permet de suivre l’utilisateur automatiquement à partir de la reconnaissance faciale.

Le « mode patrouille » permet de faire patrouiller le drone dans une pièce définit. Et qui cherche des utilisateurs et prévoit un bruit en fonction de ce dernier.

L’api master doit aussi être capable de communiquer avec l’api de gestion des users.

## Sécurité

Bien que le contrôle du drone par inconnus ne soit pas « grave » et le que le drone n’offre aucune protection contre une utilisation malveillantes. Il doit être impossible pour une personne non autorisée d’utiliser le drone.

Pour ce faire l’api master étant le point d’entrée des commandes elle doit offrir une résistance contre les attaques.

Et bien entendu limiter le transfert de login au maximum en utilisant une solution alternative.

# Solutions :

## Les librairies

Pour répondre aux besoins des différentes exigences il nous faudra les librairies suivantes.

* Jwt
* Flask
* Flask cors
* Requests

## Les urls

Pour répondre aux besoins des différentes exigences il nous faudra les url suivantes.

* Il faut une url pour tester la mise en production (ping)
* Une suite d’url pour la gestion des user
  + Création de compte
  + Modification de compte
  + Surpression
  + Récupération
* Il faut une url pour activer et désactiver le mode toutou
* Il faut une url pour activer et désactiver le mode patrouille
* Il faut une suite d’url pour le mode manuelle.
  + Faire avancer le drone de x cm
  + Faire reculer le drone de x cm
  + Faire monter le drone de x cm
  + Faire descendre le drone de x cm
  + Faire tourner le drone de x cm
  + Faire avancer le drone pendant x secondes
  + Faire reculer le drone pendant x secondes
  + Faire monter le drone pendant x secondes
  + Faire descendre le drone de pendant x secondes
  + Faire tourner le drone de pendant x secondes
  + Prendre une photo
  + Bonus divers
  + …

Chacune des urls ayant un tiret correspondent à un fichier python permettant d’avoir un prefix dans l’url.

Par exemple, dans le cas des urls gestions de l’users, elles vont se présenter comme suit :

CRUD de l’utilisateur : <http://localhost:xxxx/users/>. Connexion de l’utilisation : <http://localhost:xxxx/users/connect/>

Autre exemple pour le cas des commandes manuelles :

L’url de base est : <http://localhost:xxxx/manual/>. Cependant, cette dernière ne permet aucune action.

Pour utiliser les fonctionnalités de ce fichier et de ce package il faudra ajouter le type d’action souhaité. Pour faire tourner le drone par exemple : <http://localhost:xxxx/manual/rotate>

## Solution technique

Flask offre la possibilité d’utiliser des « blueprint » (qui s’apparentent à un package), qui permettent de diviser les urls en fonction de leur objectif. En divisant un maximum les url en fichier cela permet d’augmenter la lisibilité du projet en évitant des fichiers de plusieurs milliers de lignes.

Les tirets du chapitre 2 vont correspondent aux « blueprint » des fonctionnalités. Voir les screens en annexe.

En ce qui concerne la sécurité, JWT nous semble être un bon compromis nous permettant d’offrir une sécurité suffisante vue le contexte. De plus un système de refresh token va être mis en place.

Le token aura une durée variable en fonction du contexte (production/dev). Dans un contexte de dev un token ayant une durée de vie illimité est une attrayant d’un point de vue efficacité. Dans le cadre d’une mise en production la durée de vie d’un token doit rester contrôlé. Avant chaque action le token doit être vérifié. Le refresh token va nous permettre d’obtenir un nouveau token sans devoir utiliser à nouveau les logins de l’utilisateur.

Le « mode toutou » et le « mode patrouille » ne vont pas pouvoir s’exécuter sans que l’api timeout. Il va falloir utiliser des thread/fork pour permettre à la fonctionnalité de continuer de fonctionner. Voir annexe.

# Annexe

Une image contenant texte, capture d’écran, moniteur, écran

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

Process principal de l’api

Url du « mode » fork pour la fonctionnalité.

Retour de l’api