Rapport de projet

WhatSup au service de la performance

Qwirk

Table des matières

[Table des matières 1](#_Toc484737696)

[1. Introduction 2](#_Toc484737697)

[1.1. Présentation du projet 2](#_Toc484737698)

[1.2. Technologies utilisées 2](#_Toc484737699)

[2. Installation pour le développement 4](#_Toc484737700)

[2.1. Backend 4](#_Toc484737701)

[2.2. Frontend 5](#_Toc484737702)

[2.3. RTCMultiConnection 6](#_Toc484737703)

[2.4. Electron 6](#_Toc484737704)

[3. Base de données 6](#_Toc484737705)

[4. Chat bot 7](#_Toc484737706)

# Introduction

## Présentation du projet

Qwirk ++ est un projet de chat à la frontière entre Skype et Slack. Ces deux technologies ont leur avantage et leurs inconvénients.

Skype est de moins en moins utilisés en entreprises, a des problèmes de compatibilité sur les différentes plateformes et entre ces différentes versions.

Slack est un outil spécialement conçu pour les entreprises et ne permet pas d’ajouter n’importe qui se trouvant sur sa plateforme pour engager une conversation avec lui.

Qwirk ++ arrive à la croisée des deux chemins pour offrir une expérience aussi bien web que desktop ou n’importe quel utilisateur peut en ajouter un autre et commencer une conversation avec lui.

## Technologies utilisées

Le projet a été réalisé en utilisant 4 technologies en particulier :

- Django pour le backend

Django est un Framework web python très connu et très utilisés dans le monde.  
Ces nombreuses années de vie en font un des Framework les plus viable et sécure sur le marché tout en gardant la flexibilité du python pour le code

- VueJs pour le frontend

VueJs est un Framework javascript moins connu que ces concurrents maintenus par de gros groupes. Néanmoins ses performances n’ont rien à envier aux gros, il est très flexible et simple d’utilisation et est plus starré sur github que angular2.  
Vous pouvez trouver ici une comparaison plus poussée entre VueJs et ces principaux concurrents : https://fr.vuejs.org/v2/guide/comparison.html

- RTCMultiConnection pour le serveur WebRTC

RTCMultiConnection est server en nodeJS basé sur la technologie web WebRTC permettant de faire du transfert audio et vidéo via internet.

- Electron pour packager notre web app en desktop app

Electron est un des packageur web les plus stable et connus. Il permet de packager son application web dans un exécutable qui embarque un navigateur *chromium*. Il n’est pas lourd et très portable.

Le serveur backend et le frontend sont complètement séparées. Chacune de ces deux parties sont indépendantes l’une de l’autre. Ils peuvent donc être changés en fonction des évolutions technologiques sans impacté l’autre. De plus les développeurs front et back peuvent coder de nouvelles fonctionnalités de manière autonome.

Le serveur Django intègre une API REST et une API WebSocket.

L’API WebSocket s’occupe de gérer tout ce qui est en rapport avec le chat engine. C’est cette api qui va permettre aux utilisateurs d’envoyer des messages, de récupérer des messages et de recevoir des notifications.

L’API REST elle va s’occuper de la création et la gestion des utilisateurs, contact, groupes et fichiers.

Le serveur Django est aussi le seul à dialoguer avec la base de données. En développement cette base est une sqlite3 en production c’est une MySQL. Un schéma de la database est accessible dans le répertoire contenant ce fichier.

Une fois le serveur de dev lancés vous pouvez vous rendre sur l’adresse <http://localhost:8000/admin/> (après avoir créé un super user <https://docs.djangoproject.com/fr/1.11/ref/django-admin/#createsuperuser> ) pour accéder à un panel d’administration vous permettant de faire des actions sur la base de données directement en ligne.

# Installation pour le développement

Le projet récupéré contient un *.git*. Celui-ci vous permettra de switcher entre la branche master (développement) et prod (production).

Ce guide est à exécuter sur une machine linux Debian ou dérivé.

## Backend

sudo apt-get install python3-dev python-pip git redis-server nginx openssl redis-server

sudo pip install virtualenvwrapper

sudo nano ~/.bashrc

---------------- Add these lines at the end of file : ----------------

export WORKON\_HOME=$HOME/.virtualenvs

source /usr/local/bin/virtualenvwrapper.sh

source ~/.bashrc

mkvirtualenv qwirk-python -p /usr/bin/python3

workon qwirk-python

cd path/to/project/4JT/server

pip install -r requirements.txt

python manage.py runserver 8000

## Frontend

Installer node et npm de préférence via nvm :

<https://www.liquidweb.com/kb/how-to-install-nvm-node-version-manager-for-node-js-on-ubuntu-12-04-lts/>

cd path/to/project/4JT/web-app

npm install

npm run dev

## RTCMultiConnection

cd path/to/project/4JT/RtcMultiConnection

npm install

node server.js

## Electron

cd path/to/project/4JT/electron

npm install

npm start

# Base de données

Voir annexe

# Chat bot

Quand on ajout un bot à un groupe de discussion cela va activer un filtre sur les messages rentrant.

Ce filtre va chercher dans le nouveau message arrivant un pattern prédéfini. Si ce pattern est trouvé il va alors essayer de trouver l’action associé au pattern. S’il le trouve il va alors chercher à trouver les paramètres de l’action. Si les paramètres trouvés correspondent aux paramètres attendus l’action sera alors exécutée par le bot.

Exemple :

Un utilisateur rentre le message suivant :

@bot kick adrien

Le serveur va filtrer le message et trouvé le pattern @bot. Il va ensuite chercher l’action qui est kick et le paramètre adrien.  
  
Pour l’action kick le bot attend comme paramètre le nom d’un utilisateur appartenant au groupe. Si adrien appartient bien au groupe alors le slackbot va le kicker du groupe.

Bot Pattern :

@bot

Bot Action :

hello

kick [username]

ban [username]

chat [phrase attendant une réponse]