# Manuel d'installation

*Développeurs :*

Arthur BUSUTTIL

Géraud MAGNE

Louis HOLLEVILLE

Thomas LUPIN

*E-mail :*

yakabible+contact@gmail.com

26 Juin 2019

# Table des matières

# 1 Informations générales

Le service Billetterie développé par la BWC pour EPITA fournit la possibilité aux associations, à l'administration et aux étudiants épitéens de créer, gérer et participer facilement à des évènements associatifs, grâce à u système simple et intuitif.

Ce produit a pour but de permettre aux associations EPITA de d'organiser des événements grâce à un dialogue automatiser avec l'administration d'EPITA. Il permet également de gérer la vente des billets et la présence sur place. Enfin, la billetterie permet une gestion des associations en termes de membres et de permissions.

L'application a été développée principalement en Django et en Bootstrap/HTML, en cas de problème rencontré sur cet aspect, vous pouvez vous référer à la documentation technique fournie avec le produit.

Ce manuel se penchera sur le déploiement de l'application.

# 2 Fichiers nécessaires

Tout d'abord, veuillez extraire le fichier `zip` fourni, ou cloner le `git` suivant :
`https://github.com/Onolwe/yakabible`

## 2.1 Contenu de l'application

— Dossier `application_yakabible` : L'application mobile. Rendez vous à la section correspondante pour plus d'informations.
— `Procfile` : Un fichier Procfile pour une intégration facile au service de PaaS Heroku.
— `travis.yaml` et `travis.sh` : Des fichiers pour configurer l'intégration continue de l'application.
— Dossier `yakabible` : Le site web Django

# 3 Mise en place du site web Django

## 3.1 Mise en place de la base de donnée

YakaBible utilise une base de donnée SQLite. L'application nécessite deux utilisateurs de base et quelques groupes pour fonctionner correctement. Vous n'avez pas besoin de configurer cela vous-même, une base de donnée minimale est fournie.

```
$ python yakabible/manage.sh loaddata minimal.yaml
```

Cette commande va configurer deux utilisateurs :
— Manager : Le responsable des associations. Username/Password : Manager/manager
— Admin : L'administrateur Système. Username/Password : Admin/admin

Vous pouvez changer les e-mails et les mots de passe de ces comptes via l'espace d'administration en vous connectant avec les identifiants par défaut de l'administrateur système :
`http://localhost:8000/admin/`
Vous trouverez les comptes créés dans la section `Utilisateurs`

La configuration de la base de donnée est prête. Vous trouverez aussi deux autres fichiers `.yaml` pour obtenir un set d'utilisateurs et d'événements pour tester l'application.

## 3.2 Lancement du site web

Pour lancer le site web, veuillez exécuter ces commandes :

```
$ python yakabible/manage.py collectstatic
$ python yakabible/manage.py runserver
```

La première commande permet de générer un dossier `static` où se trouveront les fichiers statiques de l'application. La seconde lancera le serveur.

## 3.3 Servir les fichiers statiques

Actuellement, les fichiers statiques sont servis par le module python WhiteNoise. Pour de meilleurs performances, il est conseillé d'utiliser un serveur comme `nginx` pour servir ces fichiers. Cependant, WhiteNoise reste performant, et est très simple d'utilisation. Pour plus d'informations, voir les documentations suivantes :
`http://whitenoise.evans.io/en/stable/`
`https://docs.djangoproject.com/fr/2.1/howto/static-files/`

# 4 Mise en place de l'application mobile

Node.js est requis pour compiler l'application.
L'application mobile est écrite avec Cordova. Pour la compiler, vous devez exécuter les commandes suivantes :

```
$ cd application_yakabible
$ npm install
$ cordova platform add android (ou ios)
$ cordova run android (ou ios)
```

Pour compiler pour Android, vous devez avoir installé le SDK Android (ou Android Studio) sur votre PC. Pour iOS, vous devez posséder un ordinateur Mac.

# 5 Intégration continue

L'intégration continue au sein de YakaBible se fait grâce à Travis-CI. Un fichier `.travis.yaml` est présent et est configuré pour le projet. Ce dernier va exécuter les tests, et faire un rapport sur le `coverage`, la `coding style`, et les tests réussis à chaque commit. Ce rapport peut être envoyé sous la forme de message sur l'application `Discord`. Pour cela, il faut configurer la variable environnement `WEBHOOK_URL` vers une url de WebHook d'un salon `Discord` sur l'espace de configuration de Travis-CI.

Voici un exemple de rapport :

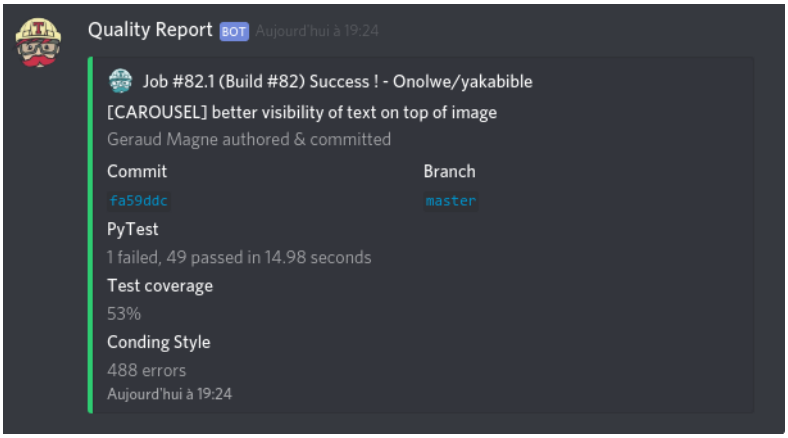


FIGURE 1 – Rapport de Travis-CI