Rapport technique

Une image contenant dessin, clipart, dessin humoristique, art

Description générée automatiquement

Tang’air – Site Web

Table des matières

[Introduction 3](#_Toc138618649)

[Définition des objectifs 4](#_Toc138618650)

[Structure du projet 5](#_Toc138618651)

[Le client 5](#_Toc138618652)

[Base.html 5](#_Toc138618653)

[LandingPage.html 5](#_Toc138618654)

[ViewFlights.html 5](#_Toc138618655)

[Login.html 5](#_Toc138618656)

[Signup.html 5](#_Toc138618657)

[ViewProfilePage.html 5](#_Toc138618658)

[AddFlightPage.html 6](#_Toc138618659)

[TeamPage.html 6](#_Toc138618660)

[ForgotPassword.html 6](#_Toc138618661)

[Clafete.html 6](#_Toc138618662)

[Le contrôleur 7](#_Toc138618663)

[def get\_db(): 7](#_Toc138618664)

[def close\_db(e=None): 7](#_Toc138618665)

[def archive\_flights(): 7](#_Toc138618666)

[def check\_user\_exists(eMail): 7](#_Toc138618667)

[def check\_credentials(username, password): 7](#_Toc138618668)

[def fill\_db\_signup(request): 7](#_Toc138618669)

[def open\_session(eMail): 7](#_Toc138618670)

[def calcul\_note(idUser,statut): 7](#_Toc138618671)

[def get\_promo(): 7](#_Toc138618672)

[def get\_flights(request): 7](#_Toc138618673)

[def get\_flights\_frommap(airport): 7](#_Toc138618674)

[def get\_user\_flights(idUser,pilot): 7](#_Toc138618675)

[def get\_user\_fellowpassenger(flights): 7](#_Toc138618676)

[def get\_airports(): 8](#_Toc138618677)

[def get\_aircrafts(): 8](#_Toc138618678)

[def fill\_db\_newflight(request) : 8](#_Toc138618679)

[def fill\_db\_reserveflight(request): 8](#_Toc138618680)

[def refresh\_user\_flights() : 8](#_Toc138618681)

[def user\_infos\_changes(request): 8](#_Toc138618682)

[def user\_editflight(request): 8](#_Toc138618683)

[def user\_cancelflight(idUser,idVol) : 8](#_Toc138618684)

[def user\_not\_listed(idUser,idVol) : 8](#_Toc138618685)

[def user\_notflying(idUser,idVol) : 8](#_Toc138618686)

[def note\_alreadyexists(request) 8](#_Toc138618687)

[def confirm\_passenger(idVol,idPassager) : 8](#_Toc138618688)

[def cancel\_passenger(idVol,idPassager) : 8](#_Toc138618689)

[def fill\_db\_newnote(request) : 8](#_Toc138618690)

[La base de données 9](#_Toc138618691)

[Conclusion et axes d’amélioration 11](#_Toc138618692)

# Introduction

Dans le cadre de la fin du semestre 6 à l’ENAC, il nous a été demandé de réunir nos connaissances en système de gestion de base de données et application web pour coder en groupe de 4, un projet de site web.

Ce document constitue le rapport technique de ce projet. Il se décompose en 3 parties distinctes :

* Définitions des objectifs
* Structure du projet
* Conclusion et axes d’amélioration

# Définition des objectifs

Le projet Tang’air est un site web permettant de mettre en relation les élèves pilotes (détenteurs d’un PPL) et les étudiants de l’ENAC via une plateforme de co-avionnage.

Il doit permettre en autre aux utilisateurs de :

* S’enregistrer en se liant à un campus
* Rechercher des vols proposés par un pilote
* Réserver un vol
* Chatter avec les pilotes dont le vol a été réservé
* Modifier son profil et ses informations

Pour un pilote, le site doit reprendre toutes ces features et en ajouter d’autres :

* Proposer un vol en y indiquant toutes les informations nécessaires
* Modifier les détails liés à un vol donné
* Gérer les réservations et les demandes
* Lier son compte à son numéro de license

# Structure du projet

Le projet s’articule, comme tout site web, autour de 3 entités :

* Le client
* Le contrôleur (serveur codé python avec le framework flask)
* La base de données SQlite

## Le client

Le client est composé en tout de 10 pages HTML assemblées avec Jinja. Il a été pensé pour être le plus léger possible et le plus facile d’utilisation.

### Base.html

Page permettant de construire toutes les autres grâce au concept de l’héritage de template de Jinja. On y trouve :

* La navbar
* Le corps de la page html
* Le footer

### LandingPage.html

Page d’accueil du site. Elle permet à tous les visiteurs d’appréhender le concept du site, de chercher des vols via la barre de recherche et de, naviguer dans la map interactive.

### ViewFlights.html

Page permettant aux visiteurs de visualiser le résultat d’une recherche via la barre de recherche ou la carte interactive.

Elle construit autant de cartes que de vols trouvés en y présentant toutes les informations nécessaires à la réservation.

### Login.html

Page permettant aux visiteurs de s’authentifier sur le site.

### Signup.html

Page permettant aux visiteurs de s’enregistrer sur le site.

### ViewProfilePage.html

C’est la page maîtresse du site. La majorité des fonctionnalités sont accessibles grâce à elle. Seuls les utilisateurs authentifiés peuvent y accéder via la navbar.

Elle permet en autre de :

* Visualiser les informations du profil de l’utilisateur
* Modifier les informations de l’utilisateur via un modal
* Visualiser les réservations de l’utilisateur (informations du vol, Statut de la réservation, annuler la réservation)
* Visualiser, si l’utilisateur est pilote, les vols qu’il a publié sur le site. Il a la possibilité de modifier[[1]](#footnote-1) les informations du vol ou de l’annuler
* Visualiser les vols passés dans la rubrique nommée ‘Vols archivés’ et de noter le pilote et les passagers.

### AddFlightPage.html

Page d’ajout des vols. Elle est accessible seulement pour les utilisateurs pilotes depuis un bouton situé dans la navbar.

### TeamPage.html

Page pour visualiser les développeurs du site tang’air

### ForgotPassword.html

Page permettant de reset son mot de passe

### Clafete.html

Page avec la mascotte tanguy qui tourne.

## Le contrôleur

Le serveur est un script python qui s’axe sur deux grandes parties à savoir la définitions des fonctions comprenant les requêtes SQL pour accéder à la base de données et la définition des routes appelant ces mêmes fonctions.

### def get\_db():

Fonction permettant d’initier la connexion à la base de données.

### def close\_db(e=None):

Fonction permettant de fermer la connexion à la base de données.

### def archive\_flights():

Fonction permettant d’archiver les vols dont la date est inférieure à la date du jour de son exécution.

### def check\_user\_exists(eMail):

Fonction qui vérifie si l’adresse mail de l’utilisateur existe dans la base de données.

### def check\_credentials(username, password):

Fonction qui vérifie que l’adresse mail et le mot de passe correspondent bien dans la table User.

### def fill\_db\_signup(request):

Fonction qui rempli la base de donnée lors de l’enregistrement d’un nouvel utilisateur

### def open\_session(eMail):

Fonction qui construit la variable session de l’utilisateur. L’email est utilisé pour chercher l’utilisateur dans la base de données. On récupère les informations nécessaires.

### def calcul\_note(idUser,statut):

Fonction qui calcule la moyenne des notes des utilisateurs en allant chercher les notes à l’aide de l’identifiant User dans la table ‘Notation’ et du statut du noté (passager ou pilote)

### def get\_promo():

Fonction permettant de récupérer toutes les promotions saisies dans la table ‘Promotion’.

### def get\_flights(request):

Fonction permettant de récupérer les vols associés à la recherche d’un utilisateur dans la barre de recherche de la landing page.

### def get\_flights\_frommap(airport):

Fonction homologue de get\_flights(). On utilise l’id aéroport pour chercher directement les vols dans la vue User\_Pilote\_Vol.

### def get\_user\_flights(idUser,pilot):

Fonction permettant de récupérer les réservations d’un utilisateur. La variable pilot est un booléen qui oriente la recherche dans la base de données.

### def get\_user\_fellowpassenger(flights):

Fonction qui créer une liste pour chaque flights passés en paramètre. Cette liste contient toutes les informations des passagers qui sont liés à ce vol via la vue ‘Passagers’.

### def get\_airports():

Fonction permettant de récupérer tous les aéroports contenus dans la table ‘Aerodrome’

### def get\_aircrafts():

Fonction permettant de récupérer tous les avions contenus dans la table ‘Avion’.

### def fill\_db\_newflight(request) :

Fonction qui remplit la base de données lors de l’ajout d’un nouveau vol.

### def fill\_db\_reserveflight(request):

Fonction qui permet de créer une réservation dans la table

### def refresh\_user\_flights() :

Fonction permettant de reconstruire les variables session[‘flights’] et session[‘flightsPilot’] ainsi que les variables session[‘passengers’] et session['passengersPilot']. Ces variables permettent de construire les composants cartes représentant les vols lors de la visualisation du profil de l’utilisateur.

### def user\_infos\_changes(request):

Fonction qui met à jour les informations de la table User pour un utilisateur donné.

### def user\_editflight(request):

Fonction qui met à jour les données de la table Vol lors de la modification de ces dernières par un pilote.

### def user\_cancelflight(idUser,idVol) :

Fonction qui supprime la réservation sur un vol d’un passager dans la table ÊtrePassager pour un vol donné.

### def user\_not\_listed(idUser,idVol) :

Fonction qui renvoi un booléen. Le test est vrai si l’utilisateur n’est pas déjà passager du vol qu’il tente de réserver, il renvoi faux si non.

### def user\_notflying(idUser,idVol) :

Fonction homologue de la précédente qui vérifie que l’utilisateur n’est pas le pilote du vol qu’il tente de réserver.

### def note\_alreadyexists(request)

Fonction qui vérifie que l’utilisateur ne note pas plus de deux fois le même utilisateur sur un même vol.

### def confirm\_passenger(idVol,idPassager) :

Fonction qui met à jour la table ÊtrePassager et qui valide la réservation d’un utilisateur sur un vol.

### def cancel\_passenger(idVol,idPassager) :

Fonction qui supprime la réservation d’un passager sur un vol donné.

### def fill\_db\_newnote(request) :

Fonction qui remplit la table Notation lorsqu’un utilisateur en note un autre sur un vol archivé.

## La base de données

Voici la base de données utilisée pour le projet :

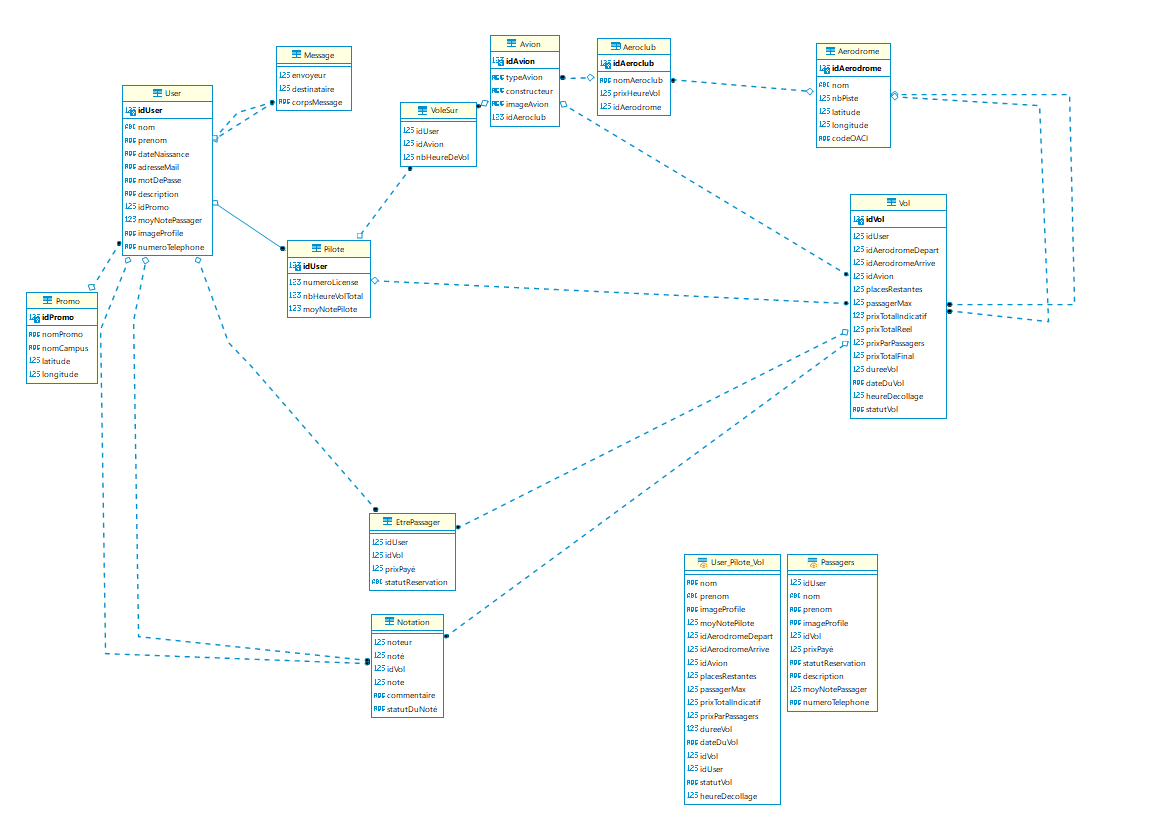


Figure 1 – UML du projet tang’air

La base de données a été générée avec un script python et le module SQlite.

La Table aérodrome est remplie grâce au script aerodrome\_filler.py. Ce script cherche les correspondances entre le nom du terrain et son identifiant OACI (ex : Nice côte d’Azur / LFMN). Il renseigne automatiquement les coordonnées du terrain, ce qui permet de générer la carte interactive avec les markers sur les aéroports.

# Conclusion et axes d’amélioration

En conclusion de ce projet, nous avons réussi à atteindre la majorité des objectifs initiaux. Nous avons pu mettre en place une architecture simple mais efficace et qui laisse une marge de progression possible.

Le temps nous as manqué pour coder la fonctionnalité de chat entre passagers et pilote. Ce qui, pour une application sociale, est préférable.

Il serait également intéressant de donner la possibilité à chaque utilisateur de pouvoir cliquer sur les photos de profils de tous les utilisateurs qu’ils rencontrent sur l’application pour leur permettre d’arriver sur un profil détaillant qui est la personne.

La possibilité de trier les vols qui sont affichés sur la page ViewProfilePage.html est également plus que nécessaire pour éviter une surcharge visuelle de cette dernière.

La possibilité d’ajouter un profil d’utilisateur qui serait gestionnaire d’un aéroclub et de leur permettre d’administrer leur flotte, prix d’heures de vols, infos utiles etc...

Enfin, des fonctionnalités de statistiques sur les vols effectués par les utilisateurs quel que soit leur profil peut s’avérer intéressante et satisfaisante.

1. Le pilote ne pourra pas abaisser le nombre de passagers max si ce nombre est inférieur au nombre de réservation actuelles sur son vol. [↑](#footnote-ref-1)