Guide pour la création des badges membres, staffeurs et entreprises

Kilian Pouderoux

2 juillet 2023

1 Introduction

Certains d'entre vous ont peut-être déjà eu la chance inouïe de participer à la fameuse perm badges pendant laquelle, de longues heures durant, nous nous acharnions à créer badge sur badge à la main, sans pause et tout ça pour ne pas être pris à la fin. Cette époque est désormais révolue (on espère) car nous avons maintenant à notre disposition de jolis outils qui vont nous permettre d'automatiser tout ça. Ce guide a pour but de vous aider, future génération, à vous y retrouver avec les outils qu'on a développé pour que vous puissiez les utiliser, les améliorer et développer les vôtres.
Le tuto qui a servi de base est celui-ci.
Le but est donc de pouvoir créer rapidement et en grand nombre des badges comme ceux-ci :

2 Présentation des fichiers

```
Badges
├── Badges STAFF
│   ├── base.html
│   ├── generateur_badges.py
│   ├── logo_BP.png
│   └── database.csv
└── Badges ENT
    ├── base_ent.html
    ├── generateur_badges_ent.py
    ├── logo_forum.png
    └── database_ent.csv
```

Il y a donc deux dossiers qui groupent les fichiers qui permettent de créer les badges staff et ent. Les fichiers sont similaires et fonctionnent de la même manière mais attention à ne pas les mélanger. Dans les grandes lignes, le fichier python va lire la base de données CSV, créer un fichier HTML de contenu à partir d'un template qui lui sera fourni et qui sera inséré dans le fichier base pour être finalement lu et converti en PDF. Vous pouvez voir le fichier base comme un squelette dans lequel on va insérer le contenu pour faire un fichier complet.
Vous serez amenés à modifier tous les fichiers présents à part les logos. Les fichiers python et html sont bien documentés, n'hésitez pas à les analyser pour vous en imprégner avant de modifier quoi que ce soit.

Les bases de données ont un format à respecter : La base de données STAFF doit contenir 3 colonnes : (orthographiées de cette manière)

— nom
— prenom
— role (responsable, staffeur,...)

La base de données ENT doit contenir 4 colonnes : (orthographiées de cette manière)

— nom
— prenom
— role (RH, data scientist,...)
— logo (chemin absolu vers le logo de l'entreprise, au format png ou jpg et pas en PDF)

Donc pour chaque personne il faut spécifier ces 3 ou 4 paramètres. Le logo est à spécifier pour CHAQUE personne de la database ENT.
Les fichiers HTML n'ont pas beaucoup besoin d'être modifiés, seulement pour changer le style visuel du badge (la couleur, police...) en fonction de vos envies.
Les exemples dans ce tuto ne concernent pas spécialement les badges ent ou staff. Tout ce qui va être dit est donc valable pour les deux dossiers, surtout la partie concernant le fonctionnement des fichiers python.

3 Installation des outils nécessaires

L'ordi sur lequel toutes ces manips ont été faites est un Macbook Pro M1 OS : 13.3.1 donc les commandes d'installation peuvent différer en fonction de votre ordi.

Tout est basé sur l'utilisation de deux modules python : *pdfkit et jinja2*, que vous pouvez installer en lançant la commande suivante dans un terminal :

```
pip install pdfkit jinja2
```

L'installation de pdfkit nécessite l'installation de *wkhtmltopdf*, que vous pouvez installer en utilisant Hombrew (Homebrew est installable depuis https://brew.sh/) et la commande suivante :

```
brew install homebrew/cask/wkhtmltopdf
```

Dans le code python vous allez voir cette ligne :

```
config = pdfkit.configuration(wkhtmltopdf='/usr/local/bin/wkhtmltopdf')
```

Cette ligne est à modifier par vos soins quand vous faites votre installation locale. Pour la modifier il faut d'abord lancer dans un terminal

```
where wkhtmltopdf
```

puis copier le résultat et l'insérer à la place du chemin spécifié dans la ligne config.
C'est bon ! Tous les outils sont installés et vous pouvez passer à la suite !

4 Fonctionnement et modifications

Étudions ensemble le fonctionnement un peu plus précisément, histoire que vous vous y retrouviez. Dans un premier temps, la base de données CSV est lue et utilisée pour créer les données qu'on va utiliser. Les données sont séparées en groupe de 10 (pour afficher 10 badges par page) et on complète le dernier groupe avec des données arbitraires s'il n'est pas de taille 10. On va ensuite remplir ce qu'on appelle le *dictionnaire de contexte* qui va être utile pour remplacer les variables dans le code HTML par les valeurs qu'on veut lui passer. Il est de la forme suivante :

```
{
'nom0': 'Kilian POUDEROUX',
'role0': 'Responsable',
'nom1': 'Antoine GUIGUI',
'role1': 'Responsable',
.....
}
```

avec 2 ou 3 paramètres par personne en fonction si il y a le logo ou pas.
Dans la suite du code, vous pouvez voir une énorme string avec du code HTML dedans. Ce code correspond à une ligne de 2 badges sur la feuille A4. L'idée est de créer autant de ligne que nécessaire et ensuite de les afficher 10 par 10 sur une feuille. On enchaîne ensuite avec la création de toutes ces lignes, en remplaçant les accolades que vous avez pu voir dans la string précédente par des trucs un peu bizarre du genre :

```
"{{{{nom{name}}}}}".format(name=10*i+j)
```

Ce qui correspond en fait à

```
{{nom0}}, {{role0}}, {{nom1}}, {{role1}}, ...
```

L'utilisation de ces trucs bizarres a quand même une utilité : créer des *placeholder* dans le code HTML pour ensuite y mettre les données souhaitées. Le module *jinja2* fonctionne comme cela, il faut que le code HTML contienne des {{*nom_placeholder*}}, qui correspondent à des emplacements de variable, pour qu'il puisse procéder ensuite au remplacement.
Une fois le contenu créé, on va créer le fichier HTML (qui va s'appeler contenu.html) correspondant. Ce fichier sera ensuite loadé et ses placeholder remplacés par ce qui est dans le dictionnaire de contexte. Si vous avez tout bien suivi vous aurez remarqué que les noms des clés dans le dictionnaire sont les mêmes que les noms des placeholder, c'est bien pour une raison. L'avant dernière étape est d'insérer ce contenu fraîchement créé à l'intérieur du template base.html. Pour ce faire, on crée un nouveau dictionnaire de contexte avec la clé *'contenu'* (Vous pouvez ouvrir le fichier base.html et voir qu'il y a un placeholder du nom de *contenu*). On load le fichier base.html et on insère le nouveau dictionnaire de contexte (qui contient donc tout le fichier contenu.html). On obtient donc une énorme string qui correspond à notre fichier HTML complet. Les dernières lignes du code sont juste là pour créer les paramètres du fichier pdf et la conversion. On retrouve la ligne concernant *wkhtmltopdf* 3 que vous ne devez pas oublier de modifier.

Et voilà ! Le PDF est créé avec tous les badges que vous voulez !

Petit récap sur tout ce qu'il est nécessaire de modifier, sachant que les fichiers sont cherchés seulement par leur nom dans le dossier, et que les images sont chechées par leur chemin absolu.

- la ligne config avec le chemin absolu vers *wkhtmltopdf*
- la ligne où on lit la database si vous changez son nom
- les lignes du genre :

  ```
  <img src="/Users/kilianpouderoux/Desktop/FORUM/Badges/BADGES_ENT\
  logo_forum.png" class="image" style="width: 100%">
  ```

  qui correspondent au chemin absolu vers les images, notamment les logos du bp et du forum
- dans les fichiers HTML les lignes qui sont marquées, pour changer la couleur du badge en fonction du rôle staffeur ou responsable
- dans les fichiers de database pour les noms, prénoms et rôles bien sûr mais aussi le chemin absolu vers les logo pour les badges ent

5 Recommandations et améliorations

En générant vos badges, vous verrez que quelques petits soucis peuvent apparaître :

- Nom ou rôle à rallonge
- Logo trop grand ou petit
- Logo mal rogné

(Il y en a sûrement d'autres mais ils ne sont pas encore apparus)
Pour certains noms ou rôles à rallonge, la génération se passe bien mais en général ça foire et il faut les faire à la main. Si tout se passe bien, la taille des logos s'adapte automatiquement pour être bien lisible et ne pas dépasser sur les autres éléments du badge mais ça peut arriver que ça ne soit pas suffisant. La solution est, là aussi, de faire les badges en question à la main. Concernant les logos mal rognés, il faut au maximum les rogner de sorte à éviter tout espace superflu autour du logo pour qu'il s'affiche de la meilleure manière possible. Certains logos très longs (genre Capgemini Invent) sont un peu compressés

horizontalement mais la différence avec le logo original se voit difficilement. C'est encore une chose à faire à la main si ça ne va pas.
La technique de gérer à la main les exceptions est excellente puisque c'est du cas par cas mais ça peut être très quali d'arriver directement à gérer ces exceptions directement dans le code. Dernier détail, le fait que la génération se fasse par paquet de 10, même si le dernier groupe ne comporte qu'une seule personne, veut dire qu'on aura au pire une page avec un badge utile dessus et 9 autres inutiles. Vous pouvez améliorer ça pour que ça tombe pile poil sur le nombre nécessaire.

6 Conclusion et contact

Voilà, vous savez tout sur la génération des badges, il ne reste plus qu'à vous y mettre ! Je pense que c'est une bonne idée de désigner quelqu'un (parmi les pôles qui gèrent les badges, donc le BP ou la log (ou d'autres ?)) comme grand manitou des badges qui s'occupera de la génération de tous les badges. Comme ça tout est installé sur un seul ordi et cette personne maîtrise bien les outils, histoire de pas perdre de temps dans l'organisation et la vérification. Mais après libre à vous de changer la manière de fonctionnement.
Et ça pourrait être pas mal de rééditer cette fiche tuto chaque année si il y a des modifications, ou juste indiquer qui était le respo badges dans votre édition, pour qu'il y ait une transmission d'année en année.

N'hésitez pas à me contacter (kilian.pouderoux@student-cs.fr ou 0644239902) si vous avez des questions sur le fonctionnement.

Amusez vous bien