Rapport Dashboard

Réalisation d'un dashboard pour une université

Date : 29/09/2021

Version : 1.0

Auteur : Camille Boucher - camille.boucher@imt-atlantique.net

Destinataires : Encadrant Fahima Djelil – Comité de pilotage

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Table des matières

1. Introduction
   - 1.1 Contexte
   - 1.2 Objectifs
   - 1.3 Problématiques
2. Corps du rapport
   - 2.1 Conception
     - 2.1.1
   - 2.2 Réalisation
     - 2.2.1 Création d'un serveur web
     - 2.2.2 Code python, chargement et traitement des données
     - 2.2.3 Code HTML
     - 2.2.4 Code JavaScript
   - 2.3 Validation
3. Conclusion
4. Glossaire
5. Bibliographie

Introduction

Contexte

Les tableaux de bord ou dashboards sont des représentations visuelles d'informations pertinentes au sein d'une entreprise ou d'une organisation. Ces dashboards se consultent souvent à partir d'un navigateur web. Les graphiques sont affichés grâce aux technologies du web. Ces dashboards peuvent être composés d'éléments textuels et visuels comme des graphiques ou des diagrammes afin de représenter des bases de données. Aujourd'hui les dashboards sont de plus en plus utilisés, car ils permettent aux entreprises ou organisations de prendre des décisions en tenant compte l'intégralité des données mises en jeu. En effet, ces derniers sont des tableaux de bords interactifs, c'est-à-dire qu'ils sont actualisés en permanence et en même temps que les données des bases de données sont mises à jour.

Objectifs

Le projet a pour but de réaliser un dashboard pour une université américaine (le client) qui nous propose des data sets contenant les données d'environ 150 000 élèves dans une cinquantaine de matières : les notes des étudiants ainsi que leurs cursus afin de les projeter dans un dashboard. Le client a besoin de traiter des données sur ces anciens élèves, les analyser et les afficher à l'aide d'un dashboard, avec certains indicateurs comme : la classification par matière, les effectifs ou bien même les résultats par genre. Ce système permettra un suivi global de la promo et pourra aider les nouveaux élèves dans leur orientation de cursus scolaire. En effet, le client souhaite traiter des données éducatives : notes de plusieurs élèves dans différentes matières, informations des cursus (associations de matières choisies par les élèves) suivis par les élèves. Il aimerait en premier lieu que ces données analysées soient accessibles à l'intégralité de la promotion. Les objectifs visés au travers de ce dashboard sont que les élèves arrivants doivent pouvoir avoir une visualisation globale de la promo précédente ainsi qu'avoir accès aux cursus des anciens élèves afin de mieux orienter leur futur choix scolaire. Ils pourront par exemple voir quelles formations ont le plus de succès, le plus d'effectif, ainsi que voir quels sont les choix de formations effectués par les anciens élèves. Les entrées du système du tableau de bord seront donc les informations anonymisées des anciens étudiants (cursus et résultats) et ses sorties seront les analyses de ces données : le dashboard pourra être créé après l'analyse et le traitement de toutes les données fournies, il permettra de mieux mettre en valeur les résultats qu'une simple base de données.

D'autre part, ce dashboard pourra également être accessible par l'équipe pédagogique et l'équipe dirigeante pour les aider à accompagner aux mieux les élèves dans leur cursus scolaire. De plus, il devra permettre un ajout ou une modification du contenu d'une manière interactive et rapide, dans l'idée que ce dashboard puisse servir au client sur une longue durée sans besoin de maintenances annuelles.

.

Problématique

Ainsi le projet répond au besoin du client en réalisant un dashboard extrait des données scolaires fournies par l'université en question afin d'accompagner au maximum les futurs élèves dans leur projet professionnel à l'aide de graphiques, chiffres et statistiques sur notamment le taux de réussite et l'effectif dans une matière, les associations classiques de matières ou bien même sur l'évolution de la moyenne des élèves dans une matière au cours des trimestres.

Annonce de plan

Pour cela, le rapport sera tout d'abord consacré à la partie de conception qui décrira le « produit » visé répondant aux fonctionnalités du cahier des charges fonctionnel (CdCF). Par la suite, le rapport présentera la phase de réalisation du produit et enfin, le corps du rapport se clôturera sur la validation du produit vis-à-vis des attentes du client et des fonctionnalités attendues. En outre, le rapport contiendra une conclusion récapitulant les phases de conception, réalisation et validation ; un résumé du rapport ; les références bibliographiques utilisées dans le rapport ; un glossaire indexant les mots /

acronymes nécessaires à une bonne compréhension du rapport et d'une annexe composée des plannings prévisionnel et réel.

Corps du rapport

Conception

Afin d'effectuer des dashboards sur d'importantes quantités de données, il est nécessaire de travailler avec des outils de programmation web comme Flask, JavaScript, HTML\ CSS et Python. Ainsi l'équipe projet a travaillé sur des plateformes de partage de code notamment GitHub.
Le but de ce projet est d'afficher un dashboard avec des visualisations personnalisées selon les données fournies par les fichiers excel qui nous ont été partagés. Ainsi les résultats que l'on souhaite présenter font directement appel aux fonctionnalités à satisfaire du cahier des charges fonctionnel :

- F1 : visualiser les effectifs des différentes matières
- F2 : visualiser le pourcentage de réussite dans chaque matière
- F3 : visualiser les associations de matières favorites (ex : matière X au semestre 1, matière Y au semestre 2)
- F4 : modifier le contenu du dashboard

Pour répondre aux besoins du client, nous avons d'abord dû réaliser un schéma conceptuel pour savoir comment nous souhaitons organiser le dashboard en fonction des fonctionnalités à satisfaire :

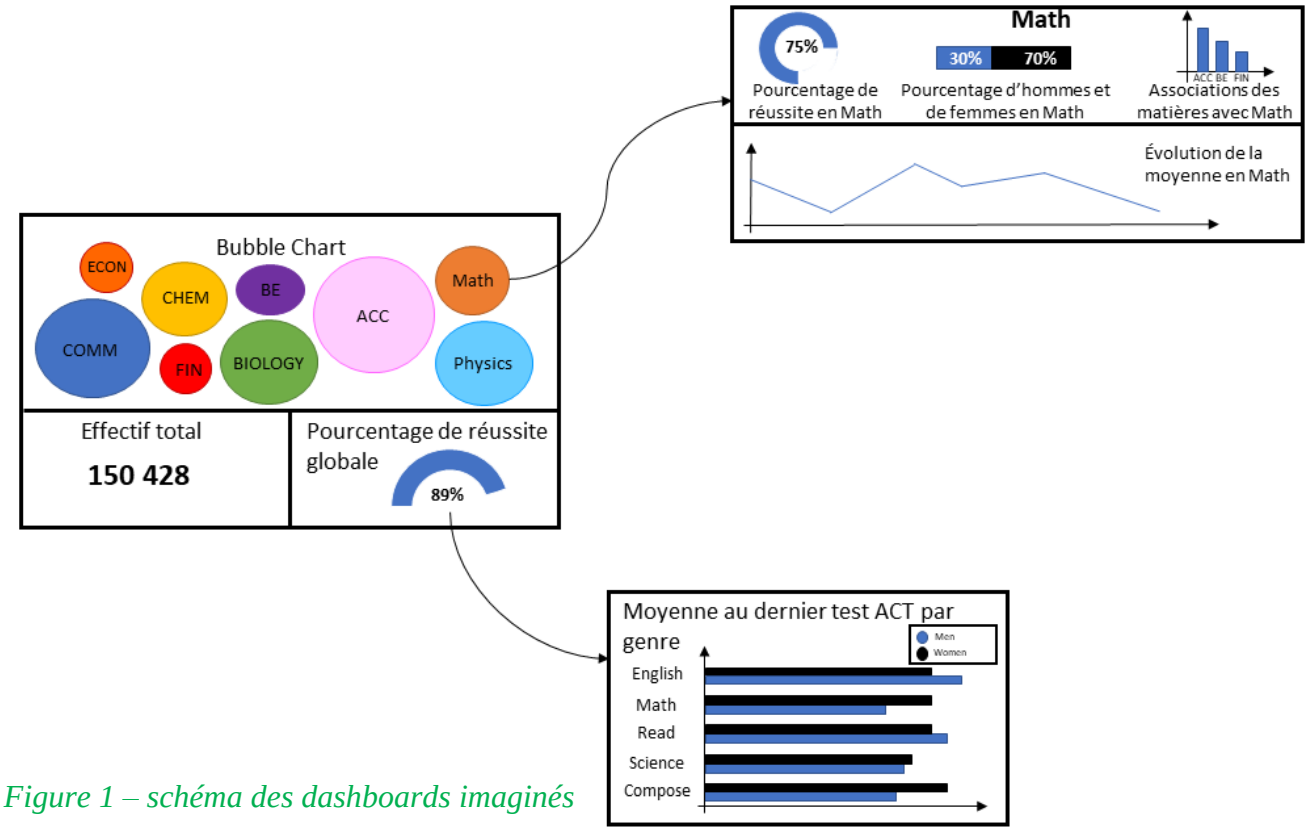

Figure 1 – schéma des dashboards imaginés

Après avoir conçu la base de le tableau de bords, il nous reste plus qu'à réaliser la partie code qui pourra afficher sur une page web les graphiques que l'on souhaite. Pour cela, nous utilisons 4 langages web différents :

Python :
Les données que l'on a utilisées sont stockées dans des fichiers cvs (student.course.csv et student.record.csv). Grâce à Python, on a pu lire ces fichiers à partir de la commande csv.DictReader,

et par suite utiliser les données qui y existent pour réaliser les fonctions nécessaires dans l'affichage des graphiques et répondre aux fonctionnalités du cahier des charges.
Pour cela, on a principalement utilisé deux listes sur python :

- list_course :cette liste contient les différentes lignes du fichier student.course.csv et chaque élément de la liste est un dictionnaire qui a pour clé "course" et pour valeur une liste de chaînes de caractères correspondant à toutes les colonnes du tableau

```
list_course: liste contenant des dictionnaires de la forme
{'course': [ANONID, SUBJECT, CATALOG_NBR ,GRD_PTS_PER_UNIT, GPAO ,DIV, ANON_INSTR_ID ,TERM]}
```

Figure 2a – liste des différents étudiants et des cours suivis

Variable	Signification
ANONID	ID Anonyme de chaque étudiant, permettant de relier chaque cours à un étudiant
SUBJECT	Libellé du cours
CATALOG_NBR	Identifiant numérique du cours
GRD_PTS_PER_UNIT	Note obtenue au cours par l'étudiant, exprimée en chiffres
GPAO	Grade Point Average - Note Moyenne globale obtenue dans d'autres cours
DIV	Discipline ou département associé au cours
ANON_INSTR_ID	Identifiant anonyme de l'enseignant
TERM	Semestre ou trimestre dans lequel le cours a été suivi par l'étudiant

Figure 2b – Description de la table student_course

- list_record :cette liste contient les lignes du fichier student.record.csv, nous n'avons pas utilisé toutes ces lignes lors du traitement de données.

Variable	Signification
MAJOR3_DESCR	libellé du troisième diplôme obtenu
MAJOR2_DESCR	libellé du second diplôme obtenu
MAJOR1_DESCR	libellé du premier diplôme obtenu
HSGPA	High School GPA (Grade Point Average)
LAST_ACT_ENGL_SCORE	résultat (note) du dernier test ACT en Anglais
LAST_ACT_MATH_SCORE	résultat (note) du dernier test ACT en Mathématiques
LAST_ACT_READ_SCORE	résultat (note) du dernier test ACT en Lecture
LAST_ACT_SCIRE_SCORE	résultat (note) du dernier test ACT en Science
LAST_ACT_COMP_SCORE	résultat (note) du dernier test ACT à l'Ecrit
SEX	genre masculin/féminin
ANONID	ID Unique de l'étudiant

Figure 3 – Description de la table student_record

Les fonctionnalités F1 (visualiser les effectifs des différentes matières) et F2 (visualiser le pourcentage de réussite dans chaque matière) ont bien été satisfaites. Cependant, on a rencontré quelques difficultés pour réaliser la fonctionnalité F3 du cahier des charges, par suite, on a décidé de réaliser d'autres fonctionnalités que nous avons trouvées intéressantes pour la représentation des données fournies l'université, afin de permettre aux nouveaux élèves arrivants d'avoir une visualisation globale sur la promo précédente.

- F5: visualiser l'effectif total de l'université
- F6: visualiser le pourcentage de réussite global dans une matière
- F7: visualiser le pourcentage d'hommes et de femmes dans une matière
- F8: visualiser la moyenne des notes dans une matière au fil du temps
- F9: visualiser la moyenne au dernier test ACT par genre

La fonctionnalité F5 prendra en entrée les identifiants des étudiants ANONID dans la table student.course et donnera en sortie leur effectif total dans l'université.

La fonctionnalité F6 prendra en entrée les valeurs de HSGPA (High School GPA (Grade Point Average) dans le fichier student.course de chaque élève et renverra en sortie la moyenne obtenue dans chacun des tests en fonction du genre.

La fonctionnalité F7 prendra en entrée le genre des étudiants d'une matière donnée et renverra en sortie le pourcentage d'hommes et de femmes

La fonctionnalité F8 prendra en entrée les notes des étudiants pour une matière donnée au cours de chaque semestre et donnera en sortie l'évolution de la moyenne globale de tous les élèves au cours du temps.

La fonctionnalité F9 prendra en entrée les résultats du dernier test ACT pour tous les élèves (en Anglais, en Mathématiques, en Lecture, en Science et à l'Écrit) et renverra en sortie les moyennes.

Pour la réalisation de F7 et F8, on a utilisé les fonctions Python suivantes :

- Nb_male_female_sub qui permet de calculer pour une matière donnée, le pourcentage de femmes et d'hommes qui la choisissent (on peut accéder aux matières à partir de la colonne SUBJECT du fichier excel student.course.csv)
- average_per_semester qui permet de calculer l'évolution de la moyenne des élèves pour une matière donnée au cours du temps.

HTML :

Le HTML est un type de langage informatique utilisé pour décrypter les pages web. Il permet de lire et d'afficher les contenus de ces pages.
Afin d'écrire en langage HTML, on utilise principalement des balises qui désignent des éléments de base fonctionnant en paire. Ce système permet d'ajouter des titres, des sous-titres, mettre un texte en gras ou en italique, introduire des éléments interactifs comme des images et des graphiques…

Par exemple, l'en-tête, **HEAD**, donne quelques informations générales sur la page web qu'on souhaite afficher comme son titre, l'encodage (pour la gestion des caractères spéciaux)
Le corps, **BODY**, est la partie principale de la page où sera affiché à l'écran tout le code que nous écrirons.

```
<!DOCTYPE html>
<html>

<head>

    <title></title>

</head>

<body>

</body>

</html>
```

Figure 4 – Structure générale d'un code HTML

Étant donné que le HTML est un langage descriptif, on utilise aussi le langage JavaScript pour que le page web réalise des actions (page web interactive). Ainsi, pour inclure du code JavaScript dans une page HTML, on utilise la balise **<script>.** On utilisera cette balise aussi pour inclure du CSS.

JavaScript :

Pour rendre les pages web interactives, on utilise le langage JavaScript. Afin d'avoir un code plus homogène nous avons décidé de séparer le code HTML du code JavaScript. Pour avoir accès au fichier JavaScript que nous venons de créer nous utilisons Flask qui permet de servir des fichiers tels quels, à condition de les placer dans un dossier appelé "static". Ainsi la commande suivante, placée dans le fichier HTML entre la balise <head> et la balise </head>, nous permet de lier nos fichiers HTML et JavaScript:

De plus, JavaScript nous est utile pour concevoir les graphiques. En effet, en utilisant la librairie HighCharts [1] nous pouvons avoir accès à des codes de JavaScript, HTML et CSS qui sont à l'origine de graphes de tout type. Sans l'utilisation de la librairie HighCharts, il nous faudrait des heures et des heures pour pouvoir coder nous-même un fichier JavaScript permettant la visualisation d'un graphe.
Concernant les fonctionnalités du CdCF, nous avons décidés d'attribuer des graphiques en accord avec les fonctionnalités demandées :

- Concernant la visualisation les effectifs des différentes matière (F1), nous avons décidé de les représenter à l'aide d'un "packed bubble chart" [2]
- Pour visualiser le pourcentage de réussite dans chaque matière (F2), un simple pourcentage avec une jauge autour nous paraissait simple et efficace [3]
- Afin de représenter les associations de matières favorites (F3), nous avons pensé à réaliser un histogramme qui représenterait les 3 matières qui vont de pair avec la matière principale choisie [4]
- L'effectif total (F5) de l'université est représenté par un simple nombre.

- Le pourcentage de réussite global (F6) (note moyenne au-dessus de 2.0 car les notes vont de 0, correspondant à un F, à 4 qui correspond à un A) est indiqué à l'aide d'un demi-cercle [5]
- Un graphique de barres négatives servira à représenter le pourcentage d'hommes et de femmes présents au sein d'une même matière (F7) [6]
- Enfin l'évolution de la moyenne des notes dans une matière au fil du temps (F8) peut être visualisée à l'aide d'une courbe progressant dans le temps [7]
- Pour finir, on peut représenter la note moyenne obtenue au dernier test ACT en fonction du genre avec un graphique à barre [8]

Réalisation

Le dashboard sera consultable à partir d'un navigateur web (Firefox, Chrome, etc.) ; les graphiques seront affichés à l'aide de code HTML, CSS et JavaScript.

Création du serveur

La première étape a été de programmer un serveur web, pour cela nous avons utilisé le langage Python et le framework (ou infrastructure de développement) Flask[9]. En effet, Python est très simple d'utilisation et agréable à utiliser.

Flask, contrairement à d'autres frameworks, n'impose pas de structure ; on peut développer une application entière sur un seul et même fichier. Nous avons tout de même suivi certaines conventions :

- Les feuilles de style, scripts, images et autres éléments qui ne seront jamais générés dynamiquement doivent être dans le dossier 'static'
- Les fichiers HTML doivent être dans le dossier 'templates'

- Le fichier ‘school.py’ contient les différentes routes de l'application
-

A l’aide du framework, nous avons créé le serveur (nous avons ici utilisé un serveur local) :

```
school = Flask(__name__)
```

Figure 5 – création d’un serveur flask

Le code suivant est propre à Flask, les lignes ‘@school.route(« / »)’ sont des décorateurs. En Python, les décorateurs sont toujours situés au-dessus d’une définition de fonction, et servent à modifier/compléter le comportement de cette fonction. Avec Flask, les décorateurs sont utilisés pour associer une URL à une fonction.

Nous avons donc créé toutes les URL dont nous avons besoin :

- La page d’accueil : **‘/dashboard/’**
- La page donnant des détails sur le GPA de l’école : **‘/dashboard/gpa/’**
- Les pages associées à chaque matière X : **‘/dashboard/X’**
  Pour accéder à ces pages, nous verrons que nous avons mis en place dans le script JavaScript un système de redirection.

Nos pages vont afficher des dashboard, nous avons placé nos codes HTML dans un fichier à part.

Pour que le serveur « lise » le code HTML nous utilisons la fonction ‘render_template’ :

```
@school.route('/dashboard/')
def effectif():
    m=render_template("dash_school.html")
    return m

@school.route('/effectif/gpa/')
def gpa():
    m=render_template("gpa.html")
    return m

@school.route('/effectif/MATH/')
def MATH():
    m=render_template("MATH.html")
    return m
```

Figure 6 – création des différents URL

Nous verrons après les scripts HTML.

Ensuite on lance le serveur avec le code

```
if __name__ == '__main__':
    school.run(host='0.0.0.0', port=8080)
```

Figure 7 – lancement du serveur

Finalement, pour lancer le programme python, nous pouvons utiliser l’invite de Commande, ensuite il faut :

- (1) Se déplacer dans le dossier contenant le fichier python
- (2) taper la commande ‘set FLASK_APP=school.py’
- (3) taper la commande ‘flask run’

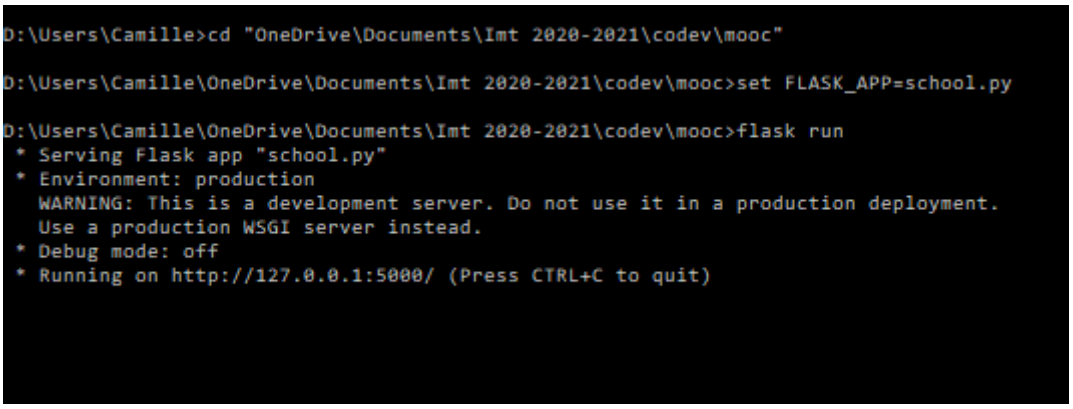

Figure 8 – invite de commande

Il suffit enfin d'ouvrir un navigateur web et d'entrer l'URL donnée par l'invite de commande + le chemin indiqué dans le fichier python.

Chargement et traitement des données

Dans le code python nous avons tout d'abord importé les librairies utiles (dont flask) puis nous avons chargé nos données qui étaient stockées dans des fichiers csv. Une fois extraites nous avons stockées ces données dans des listes afin de pouvoir les traiter ultérieurement.

```
23  file_course=open('D:/Users/Camille/OneDrive/Documents/Imt 2020-2021/codev/donnees/student.course.csv','r')
24  reader=csv.DictReader(file_course,delimiter=',')
25  list_course=[]
26  for row in reader:
27      sc=list(row.values())
28      list_course.append({'course':sc})
29  file_course.close
```

Figure 9 – chargement des données dans une liste python

Ensuite, nous avons dû traiter et analyser les données afin d'obtenir les différents graphes comme présenté lors de la conception.

Par exemple, pour le premier graphe de la page principale, le bubble chart représentant les effectifs de chaque matière, nous devons avoir la liste de toutes les matières ainsi que l'effectif associé. Voici le code python, séparé en plusieurs fonctions :

```python
# Fonction pour extraire à partir de list_course une liste de couple (Cours, Nombre d'élève)
def fonction_rajoute(element, liste_de_couple):
  for i in range(len(liste_de_couple)) :
    if liste_de_couple[i][0]==element :
      liste_de_couple[i]= (element,liste_de_couple[i][1]+1)
      break
  return

def fonction_recherche (element, liste_de_couple ):
  rep = 0
  for couple in liste_de_couple :
    if couple[0]==element :
            return True

def l_cours(l):
  cours = []
  liste_reponse = []
  for i in range(0,len(l)):
    nom_cours = l[i]['course'][1]
    if not fonction_recherche (nom_cours, cours) :
      cours.append((nom_cours,1))
    else : fonction_rajoute(nom_cours,cours)
  return cours


## Fonction pour réaliser un dico à js pour réaliser un graphique BubbleChart
def fonction_reponse_total (liste_intermediaire):
  series = []
  for element in liste_intermediaire :
    dico = {}
    d = {}
    d["name"] = element[0]
    d["value"] = element[1]
    dico["data"] = [d]
    dico["name"] = element[0]
    series.append(dico)
  return series
```

Figure 10 – code python pour obtenir les effectifs par matière

La première fonction 'fonction_rajoute' permet d'obtenir une liste de couple (cours, nombre d'étudiants). Ensuite les fonctions 'fonction_recherche' et 'l_cours' sont des fonctions intermédiaires, 'l_cours' renvoie une liste qui pourra être appelée dans la fonction finale 'fonction_reponse_totale'.

La liste de données souhaitée est donc le résultat de ces fonctions : `data= fonction_reponse_total(l_cours(list_course))` *Figure 11 – liste des effectifs par matière*

Nous avons ensuite écrit toutes les autres fonctions pour les autres graphes.

Mais comment utilisons-nous ces données puisque nos graphes sont visualisés grâce à HTML ?

Dans la fonction 'render_template' présentée précédemment nous pouvons donner en argument ces listes de données et ainsi le script HTML aura accès à ces dernières. Mais pour l'instant nous n'arrivons pas à mettre en place cette solution. Dans les codes qui suivent nous avons rentré les valeurs manuellement, donc pour seulement deux matières. Le principe étant similaire pour toutes les

matières il suffit de faire fonctionner le lien entre le script python et HTML pour pouvoir traiter toutes les matières.

Code HTML

Le script HTML sera celui lu par le serveur, c'est grâce à lui que nous allons afficher les différents éléments graphiques.

Comme expliqué précédemment lors de la conception, un script HTML doit respecter une certaine structure.

Entre les balises <head> nous avons mis des balises <script> permettant d'avoir accès à des libraires comme Highcharts.

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
<script src="https://code.highcharts.com/highcharts.js"></script>
<script src="https://code.highcharts.com/highcharts-more.js"></script>
<script src="https://code.highcharts.com/modules/solid-gauge.js"></script>
<script src="https://code.highcharts.com/modules/exporting.js"></script>
<script src="https://code.highcharts.com/modules/export-data.js"></script>
<script src="https://code.highcharts.com/modules/accessibility.js"></script>
```

Figure 12 – import de librairies et code JavaScript dans le fichier dash_school.html

Nous pouvons ensuite modifier le <title> (celui qui s'affiche dans l'onglet du navigateur).

Enfin, ce qui nous intéresse c'est le contenu de <body>, c'est ici que nous allons organiser nos différents éléments visuels et textuels.

Pour le contenu textuel nous plaçons différentes balises <h1>, <h2> (titres très importants et importants) et <p> pour les paragraphes.

On indique bien ce que nous voulons afficher dans les balises . Puis on donne entre les balises <script> le script JavaScript permettant de construire le élément graphique. Nous détaillerons ce script dans la prochaine partie.

Pour afficher plusieurs éléments graphiques, il suffit de répéter ce schéma balises   et <script> javascript </script>.

On utilise le langage CSS pour réaliser une mise en forme soignée de la page web tout en harmonisant l'ensemble des couleurs utilisé dans nos graphiques. On indique le style CSS d'un élément directement dans des balises HTML ou on peut également identifier les balises concernées à l'aide de noms afin de leur appliquer des règles distinctes. Il est possible d'écrire du code CSS dans l'entête d'un document HTML, dans la balise <head>. Il suffit pour cela de l'encadrer par une balise >style> ayant pour attribut type="text/css".. Cette méthode est ici applicable puisque nous avons très peu d'éléments ; dans le cas contraire nous créons une feuille de style externe : c'est-à-dire rédiger le code CSS dans un document externe afin qu'il puisse être appelé par différentes pages.

Rappelons qu'une règle CSS est composée de 2 éléments : le sélecteur et la déclaration. Le sélecteur indique l'élément sur lequel vont s'appliquer les déclarations CSS. La déclaration les propriétés et les valeurs à appliquer à cet élément.

```
21        <header>
22          <div class="header-background"></div>
23          <div class="title", style="margin:500px
24          box-shadow: 0 0 2px 1px rgba(175, 175, 175, 0.3);">
25            <h1>
26              DASHBOARD
27            </h1>
28          </div>
29        </header>
```

Figure 13 – création d'un titre stylisé dans le fichier dash_school.html

Ici nous créons un style pour le titre affiché en haut de la page et nous le plaçons de façon centrée avec une marge. On peut ainsi organiser tous les éléments de la page.

Code Javascript

Intéressons maintenant au code JavaScript qui se trouve entre les balises <script>.

Regardons par exemple le bubble chart représentant les effectifs par matières.

La méthode addEventListener() de EventTarget attache une fonction à appeler chaque fois que l'événement spécifié est envoyé à la cible, ici le document lui-même.

Nous y utilisons la librairie Highcharts, via 'highcharts.chart()' ligne 36. Tout d'abord, on lui donne l'identifiant de l'élément HTML dans lequel afficher le graphique (l'id que l'on a déclaré dans la balise ligne 33).

```
33          <div id="container", height=300px, width=600px></div>
34          <script>
35          document.addEventListener('DOMContentLoaded', function () {
36                const chart = Highcharts.chart('container', {
37            chart: {
38                type: 'packedbubble',
39                height: '100%'
40            },
41            title: {
42                text: 'Number of students by subject'
43            },
44            tooltip: {
45                useHTML: true,
46                pointFormat: '<b>{point.name}:</b> {point.value} students'
47            },
```

Figure 14 – création du graph bubble chart dans le fichier dash_school.html

On peut ensuite déclarer quel type de chart nous souhaitons (ici 'packedbubble'), le titre affiché ainsi que le 'tooltip' (c'est ce qui va s'afficher quand on va passer le souris sur un élément du chart).

On peut également renseigner les options d'affichages 'plotOptions'. C'est ici que nous allons créer des éléments de redirection vers une autre page web que nous avions mentionnée plus tôt.

Figure 15 – mise en place du système du redirection après avoir cliqué sur la matière

Entre les lignes 74 et 85, nous indiquons que 'series' (les données que nous donnerons après pour construire le chart) peuvent être associés à un clic du 'pointer' (souris) qui entrainera la redirection vers l'URL 'effectif/matière' sur laquelle on a cliqué. C'est grâce à ça qu'un étudiant peut se renseigner sur une matière en particulier à partir du dashboard principal. Cela permet de rendre le dashboard interactif.

Enfin nous donnons les données pour le graphe.

Les autres graphes sont affichés sur le même principe, la fonction highchart diffère selon le chart souhaité.

Validation

Ce projet a pour but d'afficher un dashboard avec des visualisations personnalisées en exploitant des données fournies sur excel. Il vise un suivi global de la promo afin d'aider les nouveaux élèves dans leur orientation de cursus scolaire. Ils pourront par exemple voir quelles formations ont le plus de succès, le plus d'effectif, ainsi que voir quels sont les choix de formations effectués par les anciens élèves.
La réalisation de ces attentes fait appel aux fonctionnalités du cahier de charge :

- F1 : visualiser les effectifs des différentes matières
- F2 : visualiser le pourcentage de réussite dans chaque matière
- F3 : visualiser les associations de matières favorites (ex : matière X au semestre 1, matière Y au semestre 2)

Après avoir conçu la base de le tableau de bord et réalisé le dashboard, il reste à vérifier l'efficacité de le réalisation et ses défauts vis-à-vis des attentes du client et des fonctionnalités du cahier de charge. Pour se faire, on peut choisir une décomposition par phase des résultats c'est à dire par rapport aux attentes du client :

- <u>Avoir une idée globale sur l'université :</u>

L'un des principaux objectifs du dashboard est de permettre aux étudiants de suivre la promotion précédente d'une manière globale. Ainsi, on a décidé de rajouter d'autres fonctionnalités déjà mentionnées dans la partie conception et qui sont intéressantes pour représenter des informations pertinentes de l'université.

- F5: visualiser l'effectif total de l'université
- F6: visualiser le pourcentage de réussite global
- F9: visualiser la moyenne aux derniers tests ACT par genre

On a ainsi essayé de répondre à ces fonctionnalités à partir de ces représentations personnalisées représentées ci-dessous :

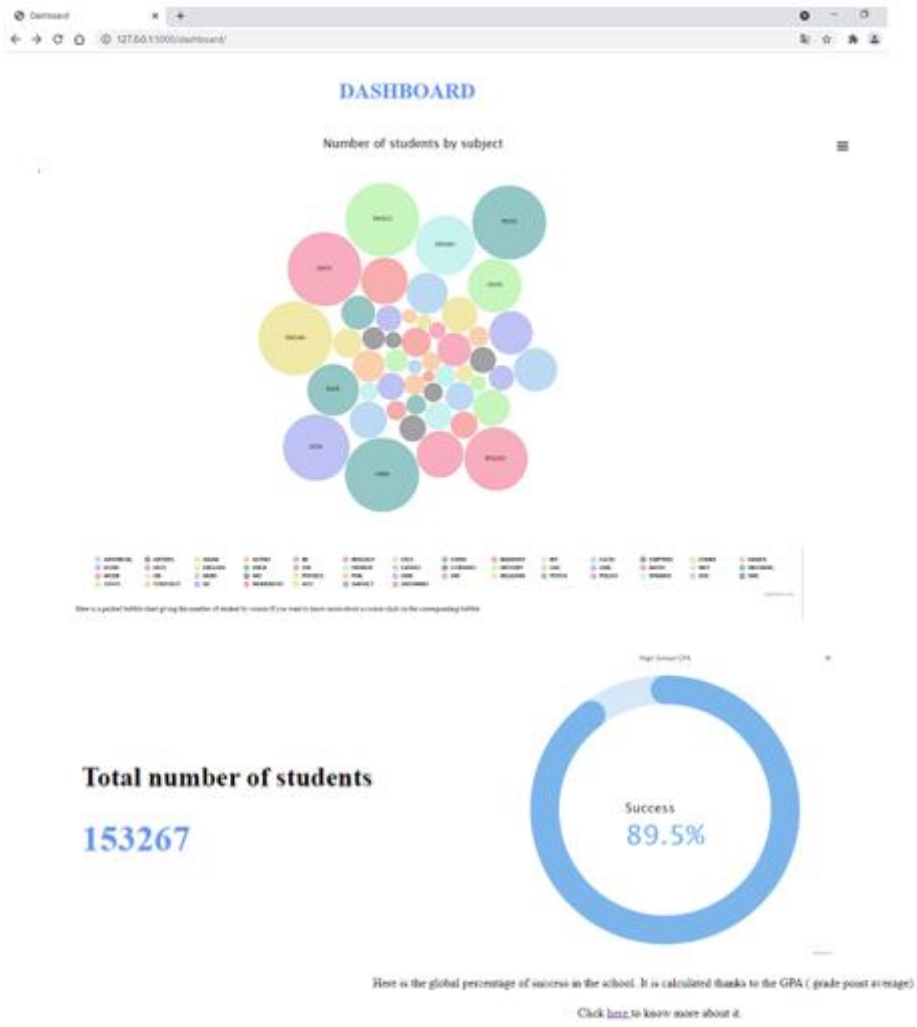

Figure 16 – Dashboard principal

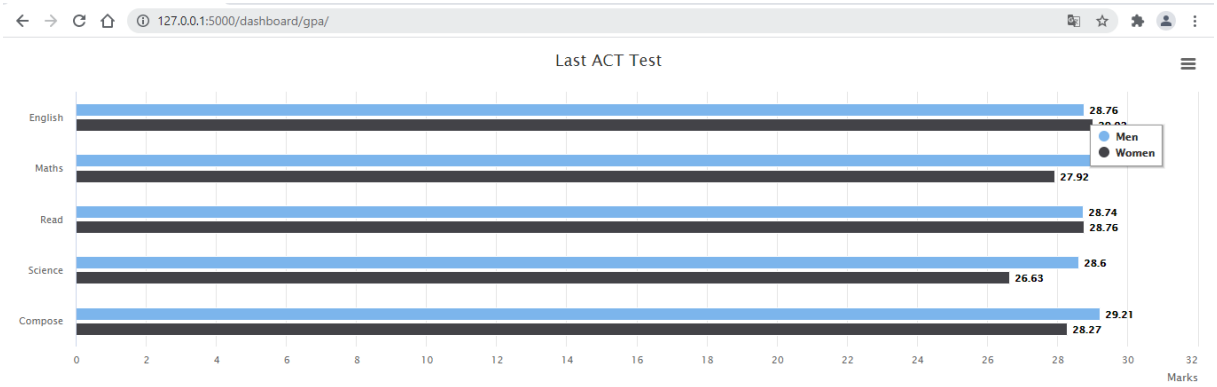

Figure 17 – Dashboard moyennes aux derniers tests ACT

Pour mieux justifier le choix de ces fonctionnalités, on peut imaginer différents scénarios et situations pertinentes vis-à-vis des besoins du client et vérifier l'efficacité de nos résultats :

- Scénario 1: une personne veut voir la répartition des effectifs de l'université pour la promo précédente (F1 réalisée par le bubble chart et F5)
- Scénario 2: Une personne veut voir le taux de réussite des autres promotions afin d'avoir une idée sur les difficultés des études dans l'université.(F6 et F9)
- Scénario 3: Une personne veut voir les résultats ainsi que le taux de succès en considérant le genre des étudiants. (F9)

Ainsi nos résultats ont été bien adéquats avec les attentes de le client à ce stade.

- Avoir une idée spécifique sur chaque matière :

Afin de mieux orienter leurs choix scolaires, les étudiants doivent pouvoir consulter les cursus suivis par les autres promotions et avoir une idée bien précise sur chaque matière. Pour se faire, on a eu recours à d'autres fonctionnalités que celles présentes dans le cahier de charge.

- F7: visualiser le pourcentage d'hommes et de femmes dans une matière
- F8: visualiser la moyenne des notes dans une matière au fil du temps

Ainsi, en cliquant sur chacune des matières, l'étudiant sera capable de visualiser le tableau de bord ci-dessous propre à chaque matière :

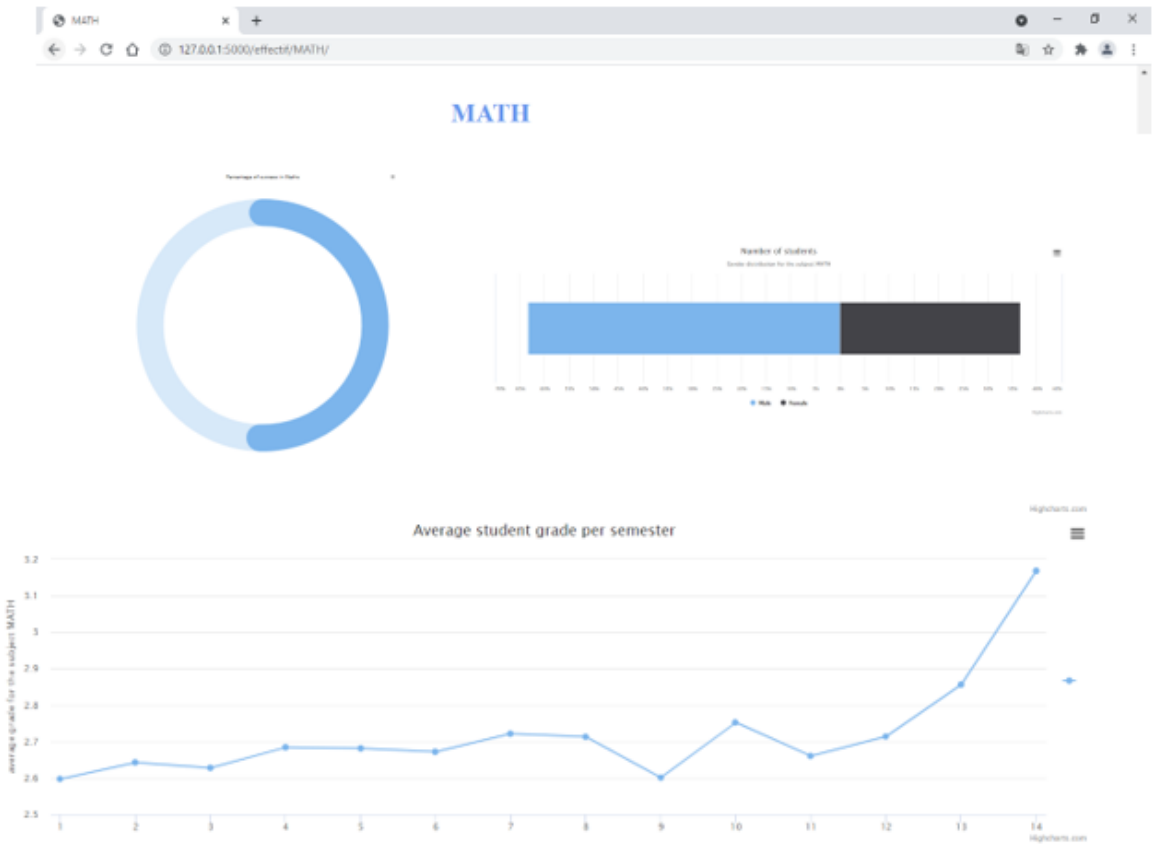

Figure 18 – Dashboard de la matière MATH

Afin d'affirmer le besoin de ces fonctionnalités, on peut penser à divers cas de figure mettant en œuvre les besoins du client et ainsi de vérifier l'efficacité des résultats obtenus :

- Scénario 1: une personne veut voir les résultats de chaque matière avant de voir les difficultés qu'il va envisager lors du choix de cette matière: On répond à ce scénario avec F2 qui est bien vérifiée par la Pie Chart.
- Scénario 2: Une personne souhaite voir comment évolue la moyenne dans une matière, montrant des signes de difficultés ou de facilités dans cette matière par diverses promo, afin de déterminer s'il souhaite prendre part à ce cours. On répond à ce scénario grâce à F8 qui est représentée par un Line Chart. Ce graph qui permettra aux étudiants de comparer les résultats précédents des autres promotions.
- Scénario 3: Une personne veut connaitre la répartition des hommes et femme afin de mieux orienter ses choix vis-à-vis à cette matière: On répond à ce scénario grâce à F7 qui est bien vérifiée par la Bar Chart.
- Scénario 4: Une personne veut connaître les associations faites par les autres étudiants entre les différentes matières afin de mieux orienter son choix: Normalement, on répond à ce scénario grâce à la F3, mais la réalisation de cette fonctionnalité semble être très compliquée lors de sa conception ainsi on a choisi de rajouter, comme indiqué précédemment pas mal de fonctionnalités afin de compenser ceci. Ainsi ce scénario ne peut être réalisé.

Conclusion

Le système dashboard a été conçu pour répondre à un besoin d'information et de renseignement. Ainsi, grâce à un travail permanent et sérieux on a réussi à réaliser un dashboard qui a permis de répondre aux attentes de le client. En effet, à la suite de la réalisation de le tableau de bord, il sera possible d'avoir une idée globale mais aussi bien précise de l'université ce qui lui permettra d'améliorer ses résultats (obtention d'une meilleure moyenne générale au fil des années) tout en minimisant les difficultés rencontrées. D'autre part, nos résultats semblent être suffisants pour connaître de manière précise les matières proposées par l'université et pour orienter le choix des étudiants vis-à-vis de leurs préférences. Mais, par manque de temps, nous n'avons pas pu répondre à l'une des fonctionnalités proposées par le cahier de charge qui consistait à consulter les associations faites par les anciens élèves entre chaque matière.

En outre, le client, l'université, souhaiterait que ce produit reste modifiable lors de l'ajout de données par des utilisateurs spécifiques. Ainsi, grâce au code python, une modification des données de l'excel va directement impacter nos scripts et sera dès lors prise en compte par le dashboard qui sera actualisé. Cette caractéristique du dashboard fait de lui un outil durable dans le temps que l'université pourra utiliser de manière efficace pendant de longues années.

Résumé:

Les tableaux de bord (ou dashboards) sont des représentations d'informations textuelles ou graphiques. Ils peuvent être utilisés par des entreprises afin d'afficher des données appropriées, pouvant être consultées à partir d'un navigateur web.
Dans ce contexte, le projet consiste à réaliser un dashboard qui nous permettra d'afficher des données éducatives d'une université américaine (qui est le client). Il consiste en une visualisation globale de la promo précédente ce qui assurera une meilleure orientation de choix des nouveaux élèves arrivants.
Le but est de parvenir à afficher différents graphiques présentants des informations pertinentes à savoir: l'effectif total de l'université, le nombre d'élèves par matière (packed bubble), le pourcentage de réussite global, la moyenne des derniers test ACT par genre, le pourcentage de réussite pour chaque matière, l'évolution de la moyenne des notes dans une matière au fil du temps et le pourcentage d'hommes et de femmes présents au sein d'une même matière .
Afin de réussir le projet et répondre aux besoins du client, on a eu recours à des outils de programmation web tels que Flask, JavaScript, HTML\ CSS et Python. On a aussi utilisé la librairie HighCharts, où on a pu avoir accès à des codes de JavaScript (pour rendre nos pages web interactives), HTML et CSS (pour réaliser une mise en forme de nos pages web avec un choix adapté de couleurs utilisées dans nos graphiques). Ces codes sont à l'origine de graphes de tout type (Packed bubble, Activity gauge, bar basic, line basic, bar negative stack …).
On a utilisé le langage Python pour lire les fichiers excel student.course.csv et student.record.csv . On a ainsi généré des listes de données qui nous ont servi pour écrire les fonctions nécessaires sur python afin de pouvoir afficher les graphiques requis. On a de même utilisé des structures de serveur Flask qui est un framework python simple permettant de réaliser des applications web évolutives.
Finalement, on a réussi à faire les affichages de graphiques décrivant les données de l'université d'une manière claire et précise. Dans la première page web, on a affiché le nombre total des étudiants, un bubble chart qui permet de visualiser les effectifs des différentes matières et un graphique présentant une jauge en arc de cercle autour du point central illustrant le pourcentage de réussite globale, qui nous ramène à une page web où on visualise la moyenne au dernier test ACT par genre. Chaque bulle du bubble chart nous ramène à une autre page web où on a affiché un graphique à barre présentant le pourcentage de femmes et hommes de la matière sélectionnée, un graphique en ligne illustrant la moyenne des notes de tous les élèves au fil du temps et une jauge en arc exposant le pourcentage de réussite dans la matière sélectionnée.

Le dashboard qu'on a conçu permet de répondre aux diverses fonctionnalités émises dans le CdFC ainsi qu'à de nouvelles fonctionnalités permettant de prévoir un plus grand nombre de scénarios. Par ailleurs, le tableau de bord permet de répondre à un besoin de renseignement et pourra constituer un élément de décisions pour les utilisateurs.

Glossaire

HTML : HyperText Markup Language, langage de balisage conçu pour représenter les pages web

CSS : Cascading Style Sheets, forment un langage informatique qui décrit la présentation des documents HTML et XML

JavaScript : langage de programmation de scripts principalement employé dans les pages web interactives

Python : langage de programmation interprété, multi-paradigme et multiplateformes

Framework : ensemble d'outils et de composants logiciels organisés conformément à un plan d'architecture et des patterns, l'ensemble formant ou promouvant un « squelette » de programme

Flask : micro framework open-source de développement web en Python

Bibliographie

[1] https://www.highcharts.com/demo

[2] https://www.highcharts.com/demo/packed-bubble

[3] https://www.highcharts.com/demo/gauge-activity

[4] https://www.highcharts.com/demo/3d-column-interactive

[5] https://www.highcharts.com/demo/gauge-activity

[6] https://www.highcharts.com/demo/bar-negative-stack

[7] https://www.highcharts.com/demo/line-basic

[8] https://www.highcharts.com/demo/bar-basic

[9] https://flask.palletsprojects.com/en/2.0.x/

[10] https://openclassrooms.com/fr/courses/4525361-realisez-un-dashboard-avec-tableau

[11] https://www.python.org/doc/