Rapport de stage

Auteur : Houda Hannouni

Maître de stage : Sébastien Combéfis

Superviseur de stage : Rémy Taymans

Année académique : 2019 - 2020

Table des matières

1 Introduction

Les étudiants de dernière année à l'Ecam doivent participer à un stage d'immersion de minimum 6 semaines. Ceci, afin de plonger dans la vie active et d'y découvrir les réalités du métier en appliquant les compétences apprises tout au long de leurs 4 années d'étude.

J'étais particulièrement intéressée par les entreprises qui proposent aux étudiants de participer à un ou plusieurs projets dans lequel/lesquels plusieurs aspects sont abordés et qui n'exploitent pas que les compétences de développeur que détiennent les étudiants en ingénierie industrielle en informatique.

Monsieur Combéfis m'a alors donné l'opportunité d'effectuer un stage qui correspond à mes attentes chez EDITx. Opportunité que j'ai saisie rapidement !

Mon TFE se déroulera également chez eux.

Le travail effectué pendant le stage constituera donc de base pour définir les perspectives du travail de fin d'étude.

2 Présentation de l'entreprise

2.1 Présentation générale

EDITx est une entreprise fondée en 2016 en tant que SPRL. Les fondateurs sont Serge Goffin et Alexandre Dembour.

Son activité principale est l'organisation de quiz et de challenges sous forme de concours dans le domaine de l'informatique à l'aide d'une plateforme en ligne[1].

EDITx vise 2 types de public différents pour des causes différentes :

- Les étudiants et professionnels pour entraîner leurs compétences, gagner des prix lors des concours et, pourquoi pas, se faire engager par l'entreprise sponsor.
- Les professeurs et spécialistes en IT pour enrichir la base de données de questions. En échange, ils ont accès à celle-ci.

2.2 Répartition des rôles

7 principaux acteurs internes ont été identifiés dans l'entreprise :

- Serge Goffin : co-fondateur et directeur financier. Il s'occupe donc de toute transaction financière mais aussi du recrutement.
- Alexandre Dembour : co-fondateur et directeur commercial. Il se charge de trouver des entreprises clientes qui seraient intéressées par l'organisation d'un challenge par EDITx. Il gère également la promotion des challenges ainsi que leurs processus de création.
- Julien Carlier : assiste Alexandre Dembour dans la recherche d'entreprises clientes et dans la gestion et la promotion des challenges.
- Amaury Hausman et Lennert Frank : commerciaux. Ils font principalement de la prospection.
- Olivier Mourisco : informaticien. Il s'occupe de la gestion de la plateforme en ligne.
- Sébastien Combéfis : « Head Advisory Board ». Il fait le lien entre EDITx et le monde académique. Il donne aussi de bons conseils techniques.

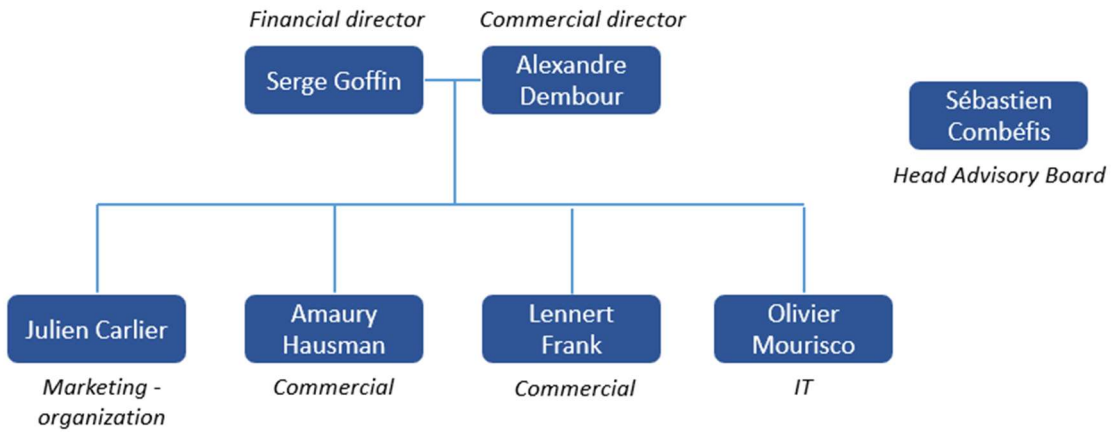


Figure 1 Organigramme

[1] https://editx.eu/

2.3 Fonctionnement global

EDITx contacte un potentiel client (ou une entreprise voulant par elle-même devenir cliente la contacte) pour l'organisation d'un challenge IT autour d'un thème bien spécifique. Ce dernier est déterminé par l'entreprise cliente qui finance le concours. En échange, celle-ci obtient les informations des participants ainsi que leurs résultats. Le but des entreprises est généralement l'embauche de profils spécifiques. Elles augmentent par la même occasion leur visibilité dans le domaine de l'informatique.

EDITx sollicite alors des professeurs et/ou spécialistes en IT pour fournir des questions en lien avec la thématique demandée. Comme expliqué au point ***2.1***, en échange, ces spécialistes et professeurs ont accès à la base de données de questions.

Une fois les questions rassemblées, le challenge peut être créé, mis en ligne, et une foule de participants peuvent s'y inscrire.

4 acteurs sont identifiés :

- Les entreprises
- Les professeurs/spécialises
- Les participants
- EDITx

Les professeurs et spécialistes répondent indirectement au besoin des entreprises, ceci par l'intermédiaire d'EDITx. Souvent, la personne de contact de ces entreprises clientes est la personne se chargeant de la gestion des ressources humaines. Cette dernière n'est pas spécialisée dans le domaine de l'informatique contrairement aux professeurs et spécialistes. Lorsque les RH font la demande d'un challenge, ils emploient généralement des mots flous, trop larges, « à la mode », qui englobent plusieurs notions. Nous les avons nommés « buzz words ». Cette appellation sera utilisée pour toute la suite de ce rapport.

EDITx permet donc :

- Aux entreprises de trouver les profils qu'ils recherchent ainsi que d'augmenter leur visibilité dans le domaine de l'IT.
- Aux étudiants et professionnels de s'exercer et d'apprendre de manière ludique.
- Aux professeurs contributeurs d'utiliser la base de données de questions afin de, par exemple, s'en inspirer pour leurs cours.

3 Objectifs du stage

3.1 Contexte

Par le caractère trop flou des « buzz words » employés par les RHs (représentants des entreprises clientes) il se peut que les professeurs ne délivrent pas les questions adéquates au challenge demandé.
Une problématique chez EDITx est donc de trouver comment faire l'intermédiaire entre des acteurs de profils très différents.

Une deuxième problématique concerne la base de données qui est un élément très important. En effet, celle-ci ne bénéficie pas d'une organisation solide et pratique.

3.2 Objectifs

Les objectifs de ce stage sont donc de répondre aux deux problématiques décrites au point précédent.

- Pour une meilleure organisation de la base de données, nous avons décidé d'y intégrer une classification scientifique des domaines de l'informatique.

- Pour faciliter le lien entre le vocabulaire des RHs et celui des informaticiens, nous avons opté pour la création d'un outil capable d'effectuer la traduction entre les « buzz words » et la nouvelle classification.

Ces deux objectifs sont liés. Le but final est de créer une application représentant la classification et l'outil de traduction décrit ci-dessus.

Voici les valeurs ajoutées de cette solution :

- Simplification du travail des professeurs/spécialistes : grâce à la classification, ils auront plus de facilité à trier ou à retrouver des questions.
- Meilleure compréhension de la demande : lorsqu'un RH proposera d'effectuer un challenge en utilisant un mot buzz comme thème, EDITx pourra utiliser l'outil traducteur pour être sûr d'être bien compris mais aussi pour transférer la demande aux professeurs/spécialistes de manière plus précise, liée directement à la classification qu'ils auront pour habitude d'utiliser.
- Augmentation de la crédibilité d'EDITx : notamment lorsque l'entreprise se déplace chez les clients, une démonstration peut être faite en leur présentant des exemples de questions.

4 Etapes et suivi du projet

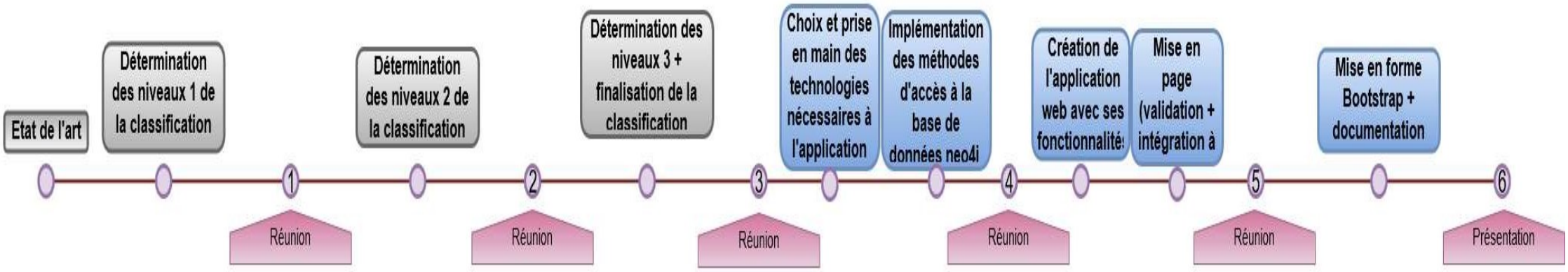
La durée totale du stage est de 6 semaines.

Les 3 premières semaines ont été consacrées à la création de la classification et donc, les 3 dernières semaines à la conception de l'application.

Un suivi hebdomadaire a été assuré par Sébastien Combéfis et Serge Goffin.

Une timeline résumant les grandes étapes est présentée à la page suivante.

Le numéro au-dessus des labels nommés « réunion » représente la semaine écoulée. J'effectuais 2 réunions par semaine : une avec Sébastien Combéfis pour les questions techniques et une avec Serge Goffin pour la supervision du projet en général. Pour une question de clarté, ces deux réunions sont représentées par un seul label « réunion ».


Etat de l'art
Détermination des niveaux 1 de la classification
Détermination des niveaux 2 de la classification
Détermination des niveaux 3 + finalisation de la classification
Choix et prise en main des technologies nécessaires à l'application
Implémentation des méthodes d'accès à la base de données neo4j
Création de l'application web avec ses fonctionnalité
Mise en page (validation + intégration à
Mise en forme Bootstrap + documentation
1
2
3
4
5
6
Réunion
Réunion
Réunion
Réunion
Réunion
Présentation

5 Création d'une nouvelle classification IT

5.1 Spécifications

Avant d'effectuer un état de l'art, il faut d'abord définir ce que l'on recherche. Donc, les caractéristiques que doit avoir la classification souhaitée par EDITx. Celles-ci sont au nombre de trois :

- Minimum 3 niveaux : il s'agit d'une limite fixée par les membres d'EDITx pour ne pas noyer les utilisateurs dans un trop-plein d'informations.
- Scientifique : la classification va, entre autres, être utilisée par des professeurs et des spécialistes en IT. Il est donc nécessaire de leur présenter une classification qui leur convient le mieux. Cela permet aussi d'avoir un niveau de crédibilité et de classer les questions plus facilement tout en restant correct.
- Compréhensible un minimum par tout autre profil non-IT : la classification sera aussi utilisée par les membres d'EDITx dont la plupart ne présentent pas de profil IT. Celle-ci doit rester intuitive sans perdre son caractère scientifique pour qu'au moins les premiers niveaux soient compréhensibles par tous.

5.2 Etat de l'art

Après de nombreuses heures de recherches, j'ai pu constater que les classifications scientifiques des domaines de l'informatique n'étaient pas choses très courante. Ce qui peut sembler très surprenant puisqu'il s'agit d'un domaine fort important dans la société actuelle.

J'ai néanmoins trouvé trois classifications scientifiques intéressantes :

- CSO[2] (the Computer Science Ontology) : il s'agit d'un outil qui a généré automatiquement une ontologie à grande échelle des sujets de recherche de l'informatique à l'aide d'un algorithme[3] et qui a ensuite été affiné manuellement par des experts du domaine.
- La bibliothèque numérique[4] de l'ACM[5] : cette bibliothèque stocke principalement des articles scientifiques et des revues dans le domaine de l'informatique.
- Une classification orientée concepts[6] : cet article présente une structuration des domaines de l'informatique sous forme de « concepts ». Un point de vue bien différent des deux précédentes classifications.

[2] https://cso.kmi.open.ac.uk/home
[3] Algorithme Klink-2
[4] https://dl.acm.org/ccs/ccs.cfm
[5] Société internationale vouée à l'informatique qui soutient et développe l'innovation et la recherche scientifique liés à ce domaine.
[6] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.474.1231&rep=rep1&type=pdf&fbclid=IwAR1Lb8FF9onnLo2pE8lt7BoyM7YO3sU2uYra94_qG-cJED3Kfnw5J-O7t2U

Les deux premières ne correspondent pas aux spécifications recherchées par EDITx. En effet, bien qu'elles soient scientifiques, elles présentent plus de trois niveaux de profondeur. Celles-ci ont donc un vocabulaire trop précis, ce qui rend la classification trop complexe et moins intuitive. Seuls de vrais spécialistes du domaine peuvent la comprendre.

La dernière présente un point de vue très intéressant. L'avantage par rapport aux deux classifications précédentes est qu'elle ne présente que deux niveaux, ce qui rend le vocabulaire plus accessible. La classification reste cependant pas très intuitive pour les personnes non spécialisées dans le domaine. Cette vue sous forme de « concepts » correspondrait particulièrement bien à un profil plus analytique.

Comme les humains sont tous différents et que chacun présente une perception et une réflexion spécifique à lui, nous avons d'abord pensé à présenter aux utilisateurs de l'application deux points de vue de la classification ; un point de vue orienté « concepts » dans lequel on aurait présenté cette classification rendue plus adaptées, et un point de vue plutôt intuitif. L'utilisateur utilisera alors la classification qui correspond au mieux à son fonctionnement cognitif.

Après réflexion, nous avons préféré mettre cette classification de côté (pour l'instant) pour d'abord se concentrer sur la classification principale souhaitée.

Comme les classifications scientifiques actuelles ont été jugées trop complexes pour ce projet, je me suis penchée vers des classifications moins scientifiques mais ouvertes à tous comme par exemples les sites vendeurs de produits IT qui présentent leur classification pour une meilleure recherche de leurs produits (comme les livres qui traitent de sujets informatiques). Les articles/blogs à caractère moins scientifique (c'est-à-dire qui ne sont pas sous forme d'article scientifique) ont également fait partie des ressources consultées.

Ce genre de classification est souvent limité à un seul niveau. S'il en possède plusieurs, les informations présentées sont soit floues soit pas très convaincantes et peuvent être même quelquefois inexactes.

C'est pourquoi j'ai décidé de créer cette classification tant essentielle dans le monde de l'informatique.

5.3 Méthodologie et réflexion

5.3.1 Etape 1 : niveaux 1

Le premier objectif était de définir l'intitulé des champs de niveaux 1 de la classification. Pour cela, j'ai commencé par stocker un maximum de liens représentant une catégorisation des domaines de l'informatique (classifications scientifiques ou non, articles, schémas, …). Il s'agissait principalement des ressources découvertes pendant l'état de l'art.

J'ai alors examiné chacun de ces liens dans le but de comprendre la logique derrière chaque classification. En même temps, je repérais les mots qui revenaient le plus ainsi que les grands domaines abordés le plus souvent.

Les principales caractéristiques souhaitées pour un domaine de niveau 1 sont :

- La clarté : ce domaine exprime-t-il clairement l'idée représentée ? Les utilisateurs de la future application comprendront-ils facilement ce que comporte ce domaine ?
- L'utilité : ce domaine est-il vraiment nécessaire ? Est-il si important dans l'informatique que pour former à lui seul un champ de niveau 1 ?

Bien que le premier objectif fût de définir les premiers niveaux, la réflexion ne devait pas s'arrêter qu'à ceux-ci. En effet, à ce stade, j'avais constaté à quel point les domaines de l'informatique sont liés. Par exemple, un des domaines qui devait absolument se retrouver quelque part dans la classification est le « cloud computing ». Celui-ci fait partie de plusieurs thématiques de l'informatique comme les services réseaux et les architectures distribuées. Il devra donc être présenté deux fois puisque les deux thèmes de l'informatique ne feront surement pas partie d'un même niveau 1.

Le but était donc aussi de définir les premiers niveaux de façon à minimiser les redondances dans le niveau 2. En plus de rechercher l'intitulé des niveaux 1, il fallait également réfléchir à la façon d'organiser les deuxièmes niveaux.

5.3.2 Etape 2 : niveaux 2

Pendant l'étape 1, une idée globale des niveaux 2 avait déjà été acquise.

Pour chacun des niveaux 1 validés, une méthodologie similaire à celle de l'étape 1 fut employée avec d'autres ressources. Je me plongeais dans chacun de ces domaines de niveau 1 en brassant un maximum d'informations sur internet. Je notais les mots les plus intéressants en précisant leurs fréquences d'apparition et, éventuellement, les sous-domaines auxquels ils sont liés (afin d'anticiper les niveaux 3).

5.3.3 Etape 3 : niveaux 3

La même méthodologie a également été employée pour les champs de niveau 3. A noter que si un champ de niveaux 2 était jugé assez clair, il ne présentait pas de sous-domaines de niveau 3. En effet, plus un niveau supplémentaire est ajouté, plus il est difficile de prendre en compte l'entièreté des sous-domaines en gardant la classification intuitive et simple de compréhension.

Après chaque étape, le travail était présenté à Sébastien Combéfis et à Serge Goffin pour validation et remarques.

5.4 Présentation de la classification

La classification se trouve dans les annexes, au point ***8.1***.

5.5 Liens avec la classification

2 types de relations avec la classification ont été créées :

- Liens entre les buzz words et les domaines
- Liens entre les domaines de la classification

5.5.1 Liens entre les buzz words et les domaines

Pour rappel, l'un des objectifs du stage (point ***3.2***) est de créer un outil de traduction entre les « buzz words » et la classification.

La première étape était de retrouver les « buzz words » les plus utilisés dans le monde de l'informatique. Pour se faire, j'ai principalement consulté les sites d'actualité. Les rédacteurs de ces articles ne sont pas des spécialistes dans le domaine de l'informatique et ont tendance à utiliser beaucoup de « buzz words ».

La seconde étape consistait à effectuer les liens. Pour cela, je me suis penchée sur les définitions les plus précises de ces « buzz words » ainsi que sur des articles plus sérieux et scientifiques.

Ces liens sont présentés dans les annexes, au point ***8.2***.

5.5.2 Liens entre les domaines de la classification

Comme expliqué au point *4.3.1* (voir exemple avec « cloud computing »), les domaines de l'informatique sont liés entre eux.

J'ai donc jugé utile de présenter ces liens (annexes, point ***8.3***) et d'ajouter à l'application cette troisième fonctionnalité qui, au départ, ne faisait pas partie du cahier des charges.

6 Conception d'une application informatique

6.1 Fonctionnalités de l'application

L'application doit être capable de :

- Présenter la classification de la manière la plus claire possible.
- Présenter l'outil permettant la traduction des « buzz words » par les champs adéquats de la classification.
- Présenter les liens entre les champs de cette classification.

6.2 Modèle de données

Nous pouvons constater que cette application présentera surtout des liens :

- La classification présente des liens entre les champs et les sous-champs.
- L'outil traducteur créera des liens entre la classification et les « buzz words ».
- Il y a des liens intéressants entre les champs de la classification.

Le modèle de données le plus approprié quant à l'exploitation de toutes ces relations est le modèle de données orienté graphes.

6.2.1 Modèle de données orienté graphe : principe

Le principe est illustré dans la figure 2 ci-dessous.

Ce modèle de données ne stocke que des nœuds et des relations. Ces deux entités possèdent chacune un type spécifique et des propriétés. Concernant ces dernières, elles ne sont pas obligatoires et il n'y a pas de schémas fixes à suivre, les nœuds et relations peuvent avoir un nombre différent de propriétés.

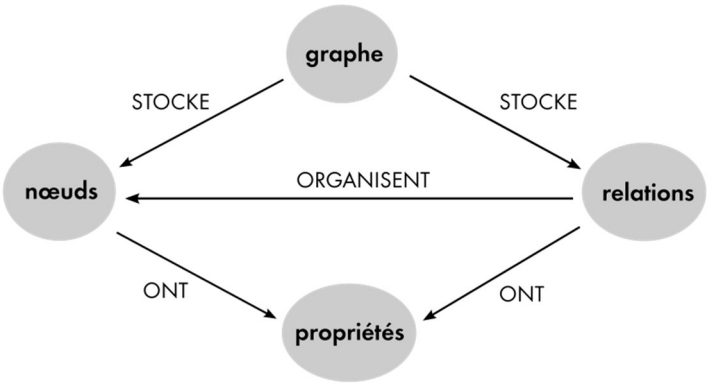


Figure 2 principe du modèle de données orienté graphe

Les relations organisent les nœuds. Il s'agit d'un modèle intuitif et pourtant très puissant.

Remarque : la figure 2 est elle-même un modèle orienté graphe comportant 4 nœuds et 5 relations (sans propriétés).

L'intérêt qui réside dans ce modèle est la recherche facile et non-coûteuse des relations entre les nœuds tandis qu'une base de données relationnelle sera contrainte à utiliser des opérations de jointures entre les tables qui sont coûteuses au système et plus complexes.

6.2.2 Modèle de données spécifique au projet

Le modèle de données utilisé dans l'application est illustré par la figure 3 ci-jointe.

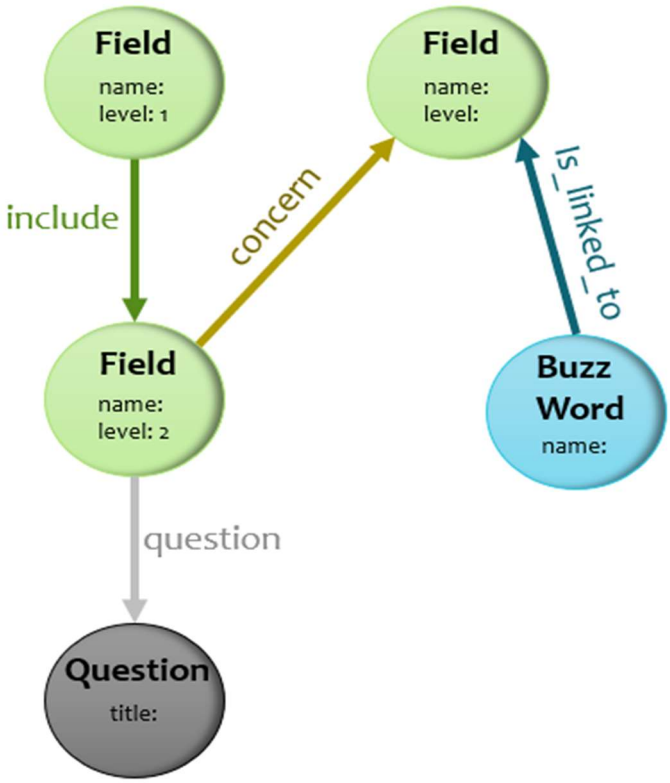


Figure 3 Modèle de données relatif au projet

3 types de nœuds y sont représentés :

- Nœud de type « **Field** » : représente les champs et sous-champs. La propriété « level » indique le niveau du champ dans la classification. Si le level est égal à 1, cela signifie que le champ correspondant est un domaine « racine », c'est-à-dire qu'il n'est pas le sous-champ d'un autre. Si cette même propriété est égale à 2, il s'agit alors d'un sous-champ d'un autre domaine de niveau 1, etc. La propriété « name » indique tout simplement le nom du champ.
- Nœud de type « **BuzzWord** » : représente les « buzz words » principaux utilisés par les RHs. Il n'a qu'une seule propriété « name » qui indique le nom du « buzz word ».
- Nœud de type « **Question** » : illustre les questions des quiz et des challenges. L'intitulé de la question est représenté par la propriété « title ».

Les relations sont au nombre de 4 :

- Relation de type « **include** » : illustre la relation entre un champ de la classification et son sous-champ.
- Relation de type « **question** » : représente la relation entre un sous-champ et une question correspondante.
- Relation de type « **concern** » : représente le lien entre les champs.
- Relation de type « **Is_linked_to** » : représente le lien entre un « buzz word » et un ou plusieurs champs de la classification. C'est cette relation qui sera utilisée pour implémenter l'outil de traduction (voir point ***3.2***, objectif 2).

6.3 Technologies et outils utilisés

6.3.1 Neo4j

Le système de gestion de base de données orienté graphe utilisé pour ce projet est Neo4j. Son code source est libre et il est développé en java par la société Suédo-Américaine Neo technology. Il s'agit d'un SGBD natif ayant son propre langage de requête nommé Cypher.

6.3.2 Python

Le langage de programmation choisi est le Python. Celui-ci possède plusieurs avantages pour ce projet :

- Il fait partie des langages les plus couramment utilisés en science de données.
- Il s'agit d'un des langages de référence dans le domaine de l'IA.
- 2 drivers neo4j ont été conçus pour ce langage : neo4j-driver qui est un driver officiel et Py2neo créé par la communauté.
- C'est un langage multifonctions, il est beaucoup utilisé pour la science des données mais aussi pour le développement web. Or, la mise en place de ce projet nécessite ces deux fonctionnalités.
- Il possède d'autres avantages comme sa grande communauté active, son code open source et sa simplicité d'utilisation.

L'environnement de développement adopté est PyCharm.

6.3.3 Py2neo

Py2neo a été choisi comme driver de la base de données neo4j de cette application. Il possède certaines fonctions qui simplifient l'implémentation sans passer par le langage de requête Cypher. Néanmoins, d'autres types de fonctions sont mises à disposition pour permettre aux utilisateurs d'utiliser le langage Cypher.

Py2neo possède aussi un ORM mais celui-ci n'a pas été utilisé puisqu'après analyse de la documentation et teste d'un exemple, j'ai constaté que la manière de programmer présentée perdait le côté intuitif de Neo4j. De plus, la documentation était moins claire et la communauté utilisant ce module peu nombreuse.

6.3.4 Flask

Parmi les frameworks les plus répandus dans le développement web avec Python, il y a Django et Flask. Django est un gros framework présentant d'innombrables fonctionnalités et est donc lourd.

Flask est un micro-framework. Il est léger et répond parfaitement aux besoins de l'application souhaitée par EDITx. C'est donc ce dernier qui a été retenu.

6.3.5 Bootstrap

Bootstrap est un framework CSS très connu et possède plusieurs avantages :

- Simple et rapide pour le design et la mise en page d'un site ou d'une application web.
- Possède un système de grilles permettant le positionnement aisé des éléments HTML.
- Responsive, c'est-à-dire qu'il permet une adaptation des pages en fonction de la taille de la fenêtre ou du dispositif utilisé (pc, smartphone, etc.).
- Garanti la compatibilité du CSS quel que soit le navigateur utilisé.

Bien que l'application ne servira que de Backend, une belle et claire représentation visuelle est nécessaire pour que celle-ci soit bien comprise par les membres d'EDITx. De plus, il s'agit d'une opportunité pour moi de découvrir cet outil populaire dans le monde du développement web.

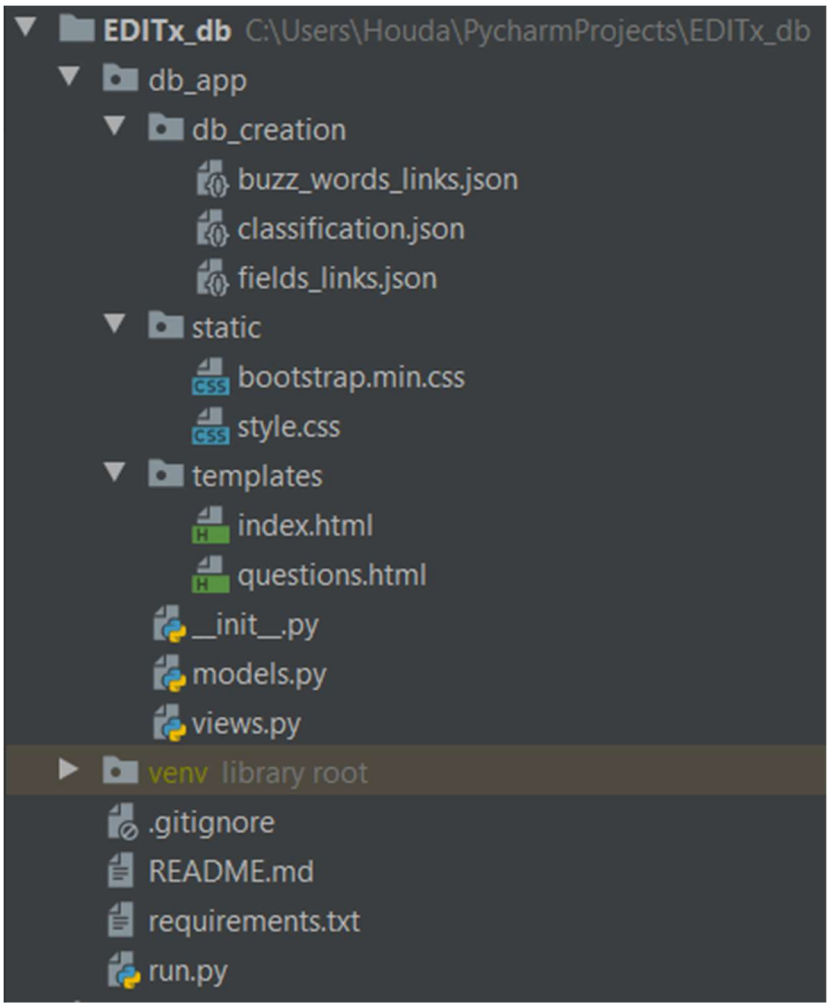
6.3.6 Github

Afin d'assurer une bonne gestion du projet informatique, Github a été un bon allié. Voici le lien :

https://github.com/Houdaaaa/EDITx_db

6.4 Architecture générale

L'organisation des fichiers correspond aux conventions basiques du micro-framework Flask :



Le dossier « static » comporte les feuilles de style CSS.

Le dossier « templates » comprend tous les fichiers HTML. Ces derniers suivent tous le format Jinja 2 qui est le moteur de template associé à Flask.

Le dossier « db_creation » contient 3 documents json qui servent à la création de la base de données d'EDITx. Pour plus de détails, voir point ***8.4***.

Le fichier « __init__.py » est le premier fichier à être chargé. Il sert principalement ici à importer les modules nécessaires au projet.

Le fichier « models.py » comporte l'implémentation des classes et fonctions relatives au projet.

Le fichier « views.py » contient les différentes routes de l'application. Les fonctions définies dans le fichier « models.py » sont appelées dans celui-ci.

Le fichier « run.py » permet de lancer l'application.

« venv » indique que le projet est sur un environnement virtuel.

Le fichier texte « requirements.txt » comprend toutes les dépendances dont le projet a besoin pour fonctionner correctement.

Le principe est simple : le fichier « views.py » traite l'information pour chaque route en appelant les fonctions adéquates du fichier « models.py ». Une fois l'information traitée et remodelée, la page HTML associée à la route est appelée. Des paramètres peuvent être passés à la page HTML qui se chargera de les afficher correctement. Cette dernière est liée à une ou plusieurs feuilles de style CSS pour une meilleure mise en forme.

6.4.1 Connection avec la base de données

La connexion à la base de données Neo4j se fait dans le fichier models.py qui comporte également toutes les méthodes d'accès à cette base de données.

6.5 Quelques détails d'implémentation

6.5.1 Models.py

C'est dans ce fichier que la connexion à la base de données est effectuée. Celui-ci compte pour l'instant une seule classe nommée « Database » qui ne contient que des méthodes statiques.

Ce fichier comporte les fonctions d'ajout, de suppression et de modification de :

- Champs
- Questions
- Buzz words
- Relations

Il contient également des fonctions capables de :

- Trouver les sous-domaines (à 1 niveau près) à partir du nom d'un domaine.
- Trouver tous les domaines ayant le même niveau de profondeur dans la classification (à l'aide de la propriété « level »).
- Trouver tous les champs liés à un buzz word ainsi que leur chemin dans la classification (donc si un champ lié à un buzz word est de niveau 3, la fonction retrouve également le ou les champs qui le contiennent de niveau 1 et 2).
- Trouver le sous-graphe d'un champ.
- Trouver tous les champs liés à un champ donné (donc ayant la relation « concerns » avec ce champ).
- Mettre en place la classification dans la base de données.
- Vider toute la base de données.

6.5.2 Dossier « templates »

Ce dossier contient deux fichiers HTML :

- « Index.html » qui représente la classification et l'outil permettant de faire le lien entre un buzz word et la classification.
- « Question.hmtl » qui présente les questions liées à un champ donné, les sous-champs de ce dernier (s'il en a) ainsi que les champs liés à celui-ci.

6.5.3 Views.py

Comme il y a deux fichiers HTML, il y a donc 2 routes.

La première permet de récupérer toute la classification pour l'afficher dans la page « index.html ». Elle permet aussi de récupérer les domaines liés à un buzz word définit par l'utilisateur. Si aucun buzz word n'a été demandé, les champs liés au buzz word par défaut « cloud computing » seront récupérés et transférés à la page HTML associée.

La deuxième route permet de récupérer toutes les questions liées au domaine souhaité pour les afficher dans la page « questions.html ». Elle récupère également tous les sous-champs du domaine en question ainsi que les champs qui y sont liés (relation « concerns »).

6.5.4 Dossier « static »

Il y a dans les fichiers HTML des liens permettant d'inclure le framework bootstrap ainsi que la feuille de style propre au projet « style.css ».

Le code CSS et responsif (ainsi que l'HTML) du menu déroulant présenté dans la page index.html a été inspiré d'un exemple figurant dans l'éditeur de code en ligne « Codepen »[7].

Le système de grilles Bootstrap a été utilisé pour une mise en page responsive et claire.

Remarque : Il est important de renseigner dans les pages HTML en premier le lien vers Bootstrap et ensuite celui de la feuille de style « style.css » pour que cette dernière ait la priorité sur Bootstrap. Il est ainsi possible d'écraser certaines propriétés d'un composant Bootstrap pour l'adapter à notre souhait.

6.5.5 Dossier « db_creation »

Ce dossier contient 3 fichiers de type json :

- « Classification.json » : représente la classification créée. Le fichier json a une clé principale nommée « items », sa valeur est une liste de dictionnaires dont chacun correspond à un champ de niveau 1. Pour plus d'informations, voir annexes point ***8.4.1***.
- « buzz_words_links.json » : représente le lien entre chaque buzz word et un ou plusieurs champs de la classification. Chaque clé de ce fichier indique un buzz word dont la valeur est une liste de champs de la classification. Il se trouve également dans les annexes, point ***8.4.2***.
- « fields_links.json » : représente les liens entre les champs de la classification. Il s'agit d'un json dont chaque clé est un champ et dont la valeur correspond aux autres champs liés à celui-ci. Voir annexes point ***8.4.3***.

Le fichier « models.py » contient des fonctions qui exploitent ces fichiers json afin de créer la base de données Neo4j.

[7] https://codepen.io/ArCodee/pen/LjvNQO?fbclid=IwAR0yH9eKXsLbfRflZhpflNc3qGIBydE65ufS4b5x1b9LSWGKxttGIVNXV8M

6.6 Manuel d'utilisation

6.6.1 Installation des dépendances nécessaires

Le fichier « requirements.txt » contient toutes les dépendances nécessaires ainsi que leurs versions.

Il est conseillé de créer un environnement virtuel (pycharm le fait automatiquement) dans lequel il faut installer toutes ces dépendances à l'aide de la commande « *pip* ».

6.6.2 Création d'une base de données Neo4j

Il faut tout d'abord installer Neo4j (Neo4j Desktop est conseillé) et y créer un nouveau projet nommé EDITx.

Le mot de passe définit doit être « *editx* ».

Pour démarrer le serveur, cliquer sur « *start* ».

6.6.3 Mise en place de la classification

La fonction database_creation() présente dans le fichier « models.py » permet de créer toute la classification ainsi que tous les liens nécessaires entre les champs et avec les buzz words (grâce aux fichiers jsons).

Il suffit donc de lancer cette fonction une seule fois pour remplir la base de données.

6.6.4 Lancer le projet

Il est possible de lancer l'application grâce à la commande suivante :

Python run.py

À condition d'être dans le répertoire où le fichier « run.py » se trouve.

Si vous utilisez un IDE comme Pycharm, lancer le projet grâce au bouton « run ».

Un lien local vous sera alors présenté pour découvrir l'application.

6.6.5 Modification de la classification

Pour l'instant, la modification de la classification se fait via les fichiers json.

Une fois la modification terminée, lancer la fonction « *delete_all* » pour vider toute la base de données.

Lancer ensuite la fonction « *database_creation* » pour reconstruire la base de données.

Il s'agit bien évidemment d'une solution provisoire, les améliorations se feront lors du TFE.

7 Conclusion

7.1 Bilan des objectifs

Les objectifs fixés pour ce stage sont repris ci-dessous :

- Création d'une classification des domaines de l'informatique adaptée aux besoins d'EDITx.

- Implémentation d'un outil capable d'effectuer la traduction entre les « buzz words » et la nouvelle classification.

L'application informatique présente bien la classification et la traduction des « buzz words » par des champs de cette classification. Une fonctionnalité en dehors du cahier des charges a même été ajoutée (lien entre les champs de la classification).

Nous pouvons donc conclure que tous les objectifs ont été atteints.

7.2 Perspectives TFE

Concernant les perspectives du travail de fin d'étude, nous pensons à exploiter l'application autour de multiples projets et à l'améliorer en fonction des constatations que nous en retirerons.

Plusieurs types d'amélioration peuvent être envisagées comme :

- La gestion d'erreurs dans le code.
- La sécurité de l'application web.
- L'ajout de fonctionnalités concernant la gestion de la base de données et donc de la classification (exemple : ajout, suppression, modification d'un champ).
- La mise en place d'une API.

Concernant la classification, celle-ci n'est pas figée et sera améliorée au fil du temps en fonction du retour des utilisateurs.

8 Annexes

8.1 Computer science classification

Computer science classification

- Network & telecommunication
  - Networks architectures
    - Topologies
    - Design principles
  - Networks equipments
    - Fibers
    - Bridges & switches
    - Routers
    - Adapters & repeaters
    - Physical firewalls
  - Wireless & mobile telecommunication
  - Network administration
    - Network cabling
    - Routing management
    - Security management
    - Access right management
  - Services
    - API
    - Cloud computing
  - Network protocols

- Cyber-security
  - Systems security
  - Architecture & design security
  - Networks security
    - Firewalls
    - Routers/switchs security
    - Intrusion detection & prevention systems
    - Email filtering
  - Identify & access management
    - Authentification & identification
    - Access management
  - Business & compliance
  - Cryptography

- Software[8]
  - CMS
  - ERP & CRM
  - CAD
  - DBMS
  - Computer graphics & animation
  - Software development
  - Operating systems
  - Mobile development
  - Gaming development
  - Libraries

- Databases
  - ORM & ODM
  - NoSQL DB models
    - Column
    - Key-value
    - Document
    - Graph
  - Relational DB models
  - DBMS
    - 1

- Programming
  - Languages
    - 1
  - Paradigms
    - Imperative & derivatives
    - OO & derivatives
    - Declarative & derivatives
  - Frameworks
    - 1
  - Compilers & interpreters
  - Good practices

[8] *Several software have been identified but for a better visibility they are not displayed, for more details see internship report*

- Modeling, design & conception
  - Software architectures
  - UML
    - Diagrams
    - Design patterns
  - Test & software quality
  - Version control & maintenance
  - Good design practices
  - Software development methodologies & project management
- Visual computing
  - Image analysis & processing
    - 2D
    - 3D & more
  - Virtual & augmented reality
  - User ergonomy
    - UX
    - UI
- Artificial intelligence
  - Natural language processing
    - Translation & speech
    - Information retrieval
    - Language analysis
  - Machine learning
    - Neural network & deep learning
    - Predictive analytics
  - Computer vision
    - Machine vision
    - Image analysis & processing
  - Decision support systems
  - Data mining
- Computer systems
  - Mobiles
  - Embedded & cyber-physical systems
    - Sensors networks
    - Robotic
    - Sensors & actuators
    - Embedded systems
    - Systems on chip
  - Real-time systems
  - Distributed systems
    - Cloud computing
    - Computer clusters
    - Grid computing
  - Operating systems

    - Quantum computer systems

- Operating systems
    - Protection & security
    - I/O systems
    - Architecture
        - Monolythic
        - Layered
        - VM based
        - Micro-kernel
        - Embedded
        - Real-time
        - distributed
    - Process management
        - Process synchronization
        - Deadlocks
        - Threads
        - Cpu scheduling
    - Storage management
        - Files systems
        - Memory management
    - OS types
        - Linux
        - Windows
        - MacOS
        - IOS
        - Android
    - Virtualization
        - VM
        - Emulation

- IT Governance
    - Digital transformation
    - Ethics
    - Green IT
    - Compliance
        - Certificates & standards
        - Laws & regulations
        - Security & business policies
    - Digital literacy

8.2 Buzz words links

Buzz words links

- Compliance
  - Compliance
  - Business & compliance
  - Certificates & standards
  - Laws & regulations
  - Security & business policies
- GDPR
  - Certificates & standards
  - Laws & regulations
  - Security & business policies
- Cloud computing
  - Cloud computing
- Cryptography
  - Cryptography
  - Security management
- Digitalisation
  - Digital transformation
  - Ethics
  - API
  - Cloud computing
- Blockchain
  - Networks security
  - Databases
  - Ethics
  - Compliance
- Big data
  - Digital transformation
  - Green IT
  - Ethics
  - Compliance
  - Databases
  - Artificial intelligence
  - Cloud computing
- Virtualization
  - Networks architectures
  - VM
  - Emulation

- Containers
  - Distributed systems
  - API
  - Emulation
- Machine learning
  - Neural networks & deep learning
  - Predictive analytics
- Open data
  - Compliance
  - Digital transformation
  - Green IT
  - Digital literacy
  - Databases
- Quantum computing
  - Artificial intelligence
  - Quantum computer systems
- Virtual reality
  - Virtual & augmented reality
  - CAD
  - User ergonomy
  - Image analyses & processing
  - Neural language processing
  - Real-time systems
  - Embedded systems
- Data science
  - Data mining
  - Machine learning
  - Databases
  - Distributed systems
- IOT
  - Embedded systems
  - Robotic
  - Sensors & actuators
  - Sensors networks
  - Systems on chip
  - Systems security
  - Cloud computing
  - Databases
  - API
  - Wireless & mobile telecommunication
  - Network equipment

- Business intelligence
  - Decision support systems
  - Cloud computing
  - ERP & CRM
- Edge computing
  - Sensors networks
  - Distributed systems
  - API
  - Cloud computing
- 4.0 industry
  - Sensors & actuators
  - Robotic
  - Sensors networks
  - Embedded systems
  - Wireless & mobile telecommunication
  - Networks architectures
  - Digital transformation
  - Ethics
  - Systems security

8.3 Fields links

Fields links

- Firewalls
  - Physical firewalls
- Wireless & mobile telecommunications
  - Mobiles
  - Embedded systems
  - Sensors networks
- Network cabling
  - Network equipments
- Routing management
  - routers
- Security management
  - Networks security
- Access rights management
  - Identify & access management
- Cloud computing
  - Distributed systems
  - databases
- Computer systems
  - Systems security
- Business & compliance
  - compliance
- Software development (IDE)
  - Modeling, design & conception
- Good practices
  - Good design practices
- Software architectures
  - Networks architectures
  - Architecture & design security
- Software development methodologies & project management
  - ERP & CRM
- Image analysis & processing
  - Computer vision

- User ergonomy
  - Computer graphics & animations
  - Gaming development
  - Mobile development
- Mobiles
  - Mobile development
- Real-time systems
  - Robotic
  - Embedded systems
- Protection & security
  - Identify & access management

8.4 Json files

8.4.1 Classification

Voici le début du fichier json. Pour le consulter entièrement, veuillez vous rendre sur le lien Github suivant : https://github.com/Houdaaaa/EDITx_db/tree/master/db_app/db_creation

8.4.2 Buzz words links

```
{
  "Compliance": [ "compliance", "business & compliance","certificates & standards", "laws & regulations", "security & business policies"],
  "GDPR" : ["certificates & standards", "laws & regulations", "security & business policies"],
  "Cloud computing" : ["cloud computing"],
  "Cryptography" : ["cryptography", "security management"],
  "Digitalisation" : ["digital transformation", "ethics", "API", "cloud computing"],
  "Blockchain" : ["networks security", "cryptography", "Databases", "ethics", "compliance"],
  "Big data" : ["digital transformation", "green IT", "ethics", "compliance", "Databases", "Artificial intelligence", "cloud computing"],
  "Virtualization" : ["networks architectures", "VM", "emulation"],
  "Containers" : ["distributed systems", "API", "virtualization"],
  "Machine learning" : ["neural networks and deep learning", "predictive analytics"],
  "Open data" : ["compliance", "digital transformation", "green IT", "digital literacy", "Databases"],
  "Quantum computing": ["Artificial intelligence", "quantum computer systems"],
  "Virtual reality" : ["virtual & augmented reality", "CAD", "user ergonomy", "image analysis & processing", "natural language processing", "real-time systems", "embedded systems"],
  "Data science" : ["data mining", "machine learning", "Databases", "distributed systems"],
  "IOT" : ["embedded systems", "robotic", "sensors & actuators", "sensors networks", "system on chip", "systems security", "cloud computing", "Databases", "API",
          "wireless & mobile telecommunications", "network equipments"],
  "Business intelligence" : ["decision support systems", "cloud computing", "ERP & CRM"],
  "Edge computing": ["sensors networks", "distributed systems", "API", "cloud computing"],
  "4.0 Industry": ["sensors & actuators", "robotic", "sensors networks", "embedded systems", "wireless & mobile telecommunications", "networks architectures",
                "digital transformation", "ethics", "systems security"]
}
```

8.4.3 Fields links

```
{
  "firewalls": ["physical firewalls"],
  "wireless & mobile telecommunications": ["mobiles", "embedded systems", "sensors networks"],
  "network cabling" : ["network equipments"],
  "routing management": ["routers"],
  "security management": ["networks security"],
  "access rights management": ["identify & access management"],
  "cloud computing": ["distributed systems", "Databases"],
  "Computer systems": ["systems security"],
  "business & compliance" : ["compliance"],
  "software development (IDE)": ["Modeling, Design & Conception"],
  "good practices" : ["good design practices"],
  "software architectures" : ["networks architectures", "architecture & design security"],
  "software development methodologies & project management": ["ERP & CRM"],
  "image analysis & processing" : ["computer vision"],
  "user ergonomy" : ["computer graphics & animations","gaming development", "mobile development"],
  "mobiles": ["mobile development"],
  "real-time systems": ["robotic", "embedded systems"],
  "protection & security" : ["identify & access management"]
}
```

8.5 Concept-oriented classification

	Topic Categories	CS	SE	IS		Topic Categories	CS	SE	IS
1.0	**Problem-Solving Concepts**	**14.7%**	**5.9%**	**5.9%**	**6.0**	**Systems/software management concepts**		**11.5%**	**6.8%**
1.1	Algorithms	5.8%	0.5%	0.2%	6.1	Project/product management (incl. risk management)	0.2%	3.3%	3.1%
1.2	Mathematics/Computational Science	6.7%	-	-					
1.3	Methodologies (object, function/process, information/data, event, business rules, ...)	-	4.9%	0.8%	6.2	Process management	-	2.2%	0.6%
					6.3	Measurement/metrics (development and use)	-	6.2%	0.8%
1.4	Artificial Intelligence	2.4%	0.5%	4.9%	6.4	Personnel issues	-	0.3%	-
					6.5	Acquisition of (Packaged/Custom) Software	0.2%	0.5%	2.3%
2.0	**Computer Concepts**	**28.7%**	**10.9%**	**0.0%**	**7.0**	**Organizational concepts**	**0.3%**	**1.9%**	**65.6%**
2.1	Computer/hardware principles/architecture	10.2%	-	-	7.1	Organizational Structure	-	0.5%	5.0%
2.2	Intercomputer communication (networks, distributed systems)	17.7%	9.5%	-	7.2	Strategy	-	-	6.6%
					7.3	Alignment (incl. business process reengineering)	-	0.5%	6.9%
2.3	Operating systems (as an augmentation of hardware)	0.80%	1.4%	-	7.4	Organizational learning/knowledge management	-	-	4.4%
2.4	Machine/assembler-level data/instructions	-	-	-	7.5	Technology transfer (incl. innovation, acceptance, adoption, diffusion)	0.1%	0.3%	19.4%
3.0	**Systems/software concepts**	**19.1%**	**54.8%**	**6.4%**					
					7.6	Change management	-	-	1.6%
3.1	System architecture/engineering	0.48%	1.9%	2.9%	7.7	Information technology implementation	-	-	1.6%
3.2	Software life cycle/engineering (incl. requirements, design, coding, testing, maintenance)	-	8.7%	1.4%	7.8	Information technology usage/operation	-	-	24.4%
					7.9	Management of "computing" function	0.2%	-	11.6%
3.3	Programming languages	3.8%	3.8%	1.4%	7.10	IT Impact	-	0.3%	15.3%
3.4	Methods/techniques (incl. reuse, patterns, parallel processing, process models, data models...)	3.8%	18.2%	0.2%	7.11	Computing/information as a business	-	-	-
					7.12	Legal/ethical/cultural/political (organizational) implications	-	0.3%	3.4%
3.5	Tools (incl. compilers, debuggers)	5.3%	12.2%	0.2%					
3.6	Product quality (incl. performance, fault tolerance)	1.8%	8.4%	1.4%					
3.7	Human-computer interaction	3.2%	1.1%	1.4%					
3.8	System security	0.80%	0.5%	0.2%					
4.0	**Data/information concepts**	**15.4%**	**7.6%**	**3.0%**	**8.0**	**Societal concepts**	**-**	**0.3%**	**1.4%**
4.1	Data/file structures	1.9%	0.8%	-	8.1	Cultural implications	-	-	0.2%
4.2	Data base/warehouse/mart organization	8.4%	4.6%	1.6%	8.2	Legal implications	-	-	0.2%
4.3	Information retrieval	4.0%	1.4%	0.4%	8.3	Ethical implications	-	-	-
4.4	Data analysis	0.64%	0.5%	0.6%	8.4	Political implications	-	0.3%	1.0%
4.5	Data security	0.48%	0.3%	0.4%					
5.0	**Problem domain-specific concepts**	**21.5%**	**2.7%**	**6.4%**	**9.0**	**Disciplinary issues**	**-**	**3.5%**	**4.3%**
5.1	Scientific/engineering (incl. bioinformatics)	0.48%	0.3%	-	9.1	"Computing" research	-	1.1%	3.3%
5.2	Information systems (incl. decision support, group support systems, expert systems)	0.64%	1.6%	6.4%	9.2	"Computing" curriculum/teaching	-	2.4%	1.0%
5.3	Systems programming	-	-	-					
5.4	Real-time (incl. robotics)	0.16%	0.5%	-					
5.5	Edutainment (incl. graphics)	20.2%	0.3%	-					

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.474.1231&rep=rep1&type=pdf&fbclid=IwAR1Lb8FF9onnLo2pE8lt7BoyM7YO3sU2uYra94_qG-cJED3Kfnw5J-O7t2U

9 Bibliographie

https://algocool.fr/bootstrap/

https://openclassrooms.com/fr/courses/4425066-concevez-un-site-avec-flask/4525776-installez-flask

https://skm.kmi.open.ac.uk/skm-data/uploads/2018/05/ISWC2018_CSO_21_CR_4_FO_EM_AS.pdf

https://www.google.com/url?sa=i&url=https%3A%2F%2Funiverthabitat.com%2Fbase-des-donn%25C3%25A9es-orient%25C3%25A9es-graphe&psig=AOvVaw2qU9cbpCDFZkwYk00Nd6J4&ust=1572282851867000&source=images&cd=vfe&ved=0CAIQjRxqFwoTCOCLhZj4vOUCFQAAAAAdAAAAABAx

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.474.1231&rep=rep1&type=pdf&fbclid=IwAR1Lb8FF9onnLo2pE8lt7BoyM7YO3sU2uYra94_qG-cJED3Kfnw5J-O7t2U