CONCOURS DE L'AGRÉGATION INTERNE

ÉCONOMIE ET GESTION

SESSION 2022

SECONDE ÉPREUVE

Épreuve de cas pratique dans la spécialité correspondant à l'option choisie par le candidat

Option D

SUJET N° 1

Durée de préparation : 4 heures

Durée totale de l'épreuve : une heure (exposé : quarante minutes maximum ; entretien : vingt minutes maximum) ; coefficient 1.

AVERTISSEMENT

Si le texte du sujet, de ses questions ou de ses annexes, vous conduit à formuler une ou plusieurs hypothèses, il vous est demandé de la ou de les mentionner **explicitement** lors de votre exposé

ROBOTS[1]

SoftBank Robotics Europe (*SBRE*) est une filiale du groupe japonais *Softbank*, issue du rachat de l'entreprise *Aldebaran Robotics*, créée par Bruno Meissonnier en 2005. L'entreprise *SBRE* s'est spécialisée dans la robotique humanoïde : elle conçoit et produit des robots capables d'interagir avec des êtres humains dans divers contextes (commerciaux, éducatifs, etc.). Après le succès de *Nao*, premier robot mis sur le marché par *SBRE* en 2006, la société a produit *Pepper*, robot haut de 120 cm et doté de multiples capteurs et d'un écran tactile, en 2014. Le robot *Pepper* est capable de reconnaître des visages et des expressions émotionnelles fondamentales (tristesse, colère, etc.).

Le robot *Pepper* interagit avec des êtres humains dans des contextes variés. Il peut fournir des renseignements et orienter des personnes perdues dans des bâtiments. Il peut aider les populations vulnérables ou âgées à se repérer dans le temps et l'espace. La robotique humanoïde s'avère également prometteuse pour les équipes de recherche en psychologie et développement de l'enfant. Enfin, le robot *Pepper* est utilisé dans l'enseignement de la robotique et de l'informatique, du collège à l'enseignement supérieur. Une application permet de créer et expérimenter des programmes en langage *Python* sur le robot. L'entreprise *SBRE* s'est d'ailleurs impliquée dans la compétition *RoboCup*, tournoi international de robotique. L'annexe 1a donne un aperçu d'un robot *Pepper* et de ses caractéristiques.

La robotique humanoïde représente un champ de recherche et développement encore très exploratoire. À ce titre, l'entreprise *SBRE* veille à répondre de façon continue aux anomalies ou dysfonctionnements concernant le robot *Pepper*. Les robots étant vendus dans plusieurs régions du monde, il est difficile d'identifier un dysfonctionnement sans être présent dans la ville ou le pays d'utilisation du robot. Le maintien de la satisfaction des utilisateurs et de la qualité de la flotte, dont l'annexe 1b présente les chiffres clefs, est un enjeu majeur pour l'entreprise *SBRE*.

Pour la valorisation des données sur le fonctionnement des robots, l'entreprise *SBRE* s'appuie sur une infrastructure exploitant des services de l'informatique en nuage (*cloud*) fournis par *Amazon Web Services (AWS)*. En effet, les robots produisent continuellement des données concernant le matériel (température des moteurs, compte-tour des moteurs, codes erreurs, etc.) et le logiciel (mise à jour des programmes, journalisation des événements, etc.). Ces données, hétérogènes, sont essentielles pour assurer le suivi des robots. Le simple fait de se connecter au serveur constitue en soi une information sur le caractère actif (ou non) du robot.

Dans une démarche d'amélioration continue de la qualité, les données concernant l'état d'un robot et son temps d'activité sont importantes, comme le sont la capacité et le nombre de cycles de charge de sa batterie. L'annexe 2 présente les points de vigilance concernant l'émission et la récupération des données produites par les robots.

D'un point de vue infrastructure, les différentes données issues de la supervision en temps réel des robots et du système d'information (*SI*) de l'entreprise *SBRE* alimentent le lac de données (*datalake*) de l'entreprise dans le nuage. Le système de gestion de bases de données *NoSQL Cassandra* est alors utilisé pour consolider les données afin qu'elles puissent être exploitées pour améliorer la conception des robots. L'utilisation d'un lac de données ouvre l'analyse et l'exploration des données aux dirigeants, aux spécialistes du marketing et aux roboticiens de l'entreprise *SBRE*. L'annexe 3 présente les infrastructures participant à la collecte, à la consolidation et au stockage des données.

---

1 Ce cas est inspiré de l'entreprise Softbank Robotics et combine des éléments factuels et fictifs.

L'annexe 4 présente un extrait des tables (consolidées) de la base de données. L'annexe 5 donne un exemple de tableau de bord. À partir des tableaux de bord, l'ensemble des équipes peut se concentrer sur des dysfonctionnements particuliers.

L'entreprise *SBRE* pourrait aujourd'hui approfondir encore la démarche qualité au cœur de ses processus. Trois pistes seraient envisageables :

- Engager les équipes dans des concertations régulières. Concrètement, des équipes pourraient se réunir régulièrement pour évoquer de potentiels anomalies et dysfonctionnements, en s'appuyant sur des tableaux de bord et des outils de cercle qualité, comme des diagrammes Ishikawa. L'annexe 6 présente un diagramme annoté lors d'une réunion de remue-méninges (*brainstorming)* sur les pannes des robots.
- Ouvrir les données consolidées, notamment celles concernant le temps d'activité (*uptime*), les pannes (*has_crashed*), ou encore les chutes (*is_falling*) aux clients à l'aide d'une interface de programmation (*API*). L'annexe 7 présente un court extrait d'un article relatif aux questions de sécurité liées aux interfaces de programmation des robots, évoqué lors d'un remue-méninges (*brainstorming)* sur la démarche qualité. L'annexe 8 présente un extrait d'interface de programmation codée en langage *Python*.
- Identifier les moyens de collecte et d'analyse des témoignages de la part des personnes utilisatrices au sujet de leur satisfaction, des dysfonctionnements constatés, ou encore des attentes quant aux fonctionnalités des robots. Ces indicateurs de satisfaction pourraient être corrélés avec les données émises par les capteurs des robots.

Production attendue

À partir des éléments de contexte, des annexes et des besoins exprimés, vous devez procéder à une analyse du cas en présentant des éléments relatifs aux points suivants :

- l'articulation des enjeux techniques aux processus liés au système d'information et aux besoins de l'organisation ;
- concernant l'émission des données par les robots :
    - les infrastructures utilisées et les risques auxquels elles sont exposées,
    - une proposition d'exploitation des données consolidées,
- concernant l'ouverture des données, des extraits de code applicatif implémentant un scénario d'utilisation ;
- concernant le recueil des témoignages :
    - une proposition d'organisation des données,
    - une architecture applicative et une infrastructure de communication sécurisées.

Annexes

Annexe 1a : Description de Pepper

« Pepper est le premier robot humanoïde au monde capable d'identifier les visages et les principales émotions humaines. Pepper a été conçu pour interagir avec les humains de la façon la plus naturelle possible à travers le dialogue et son écran tactile.

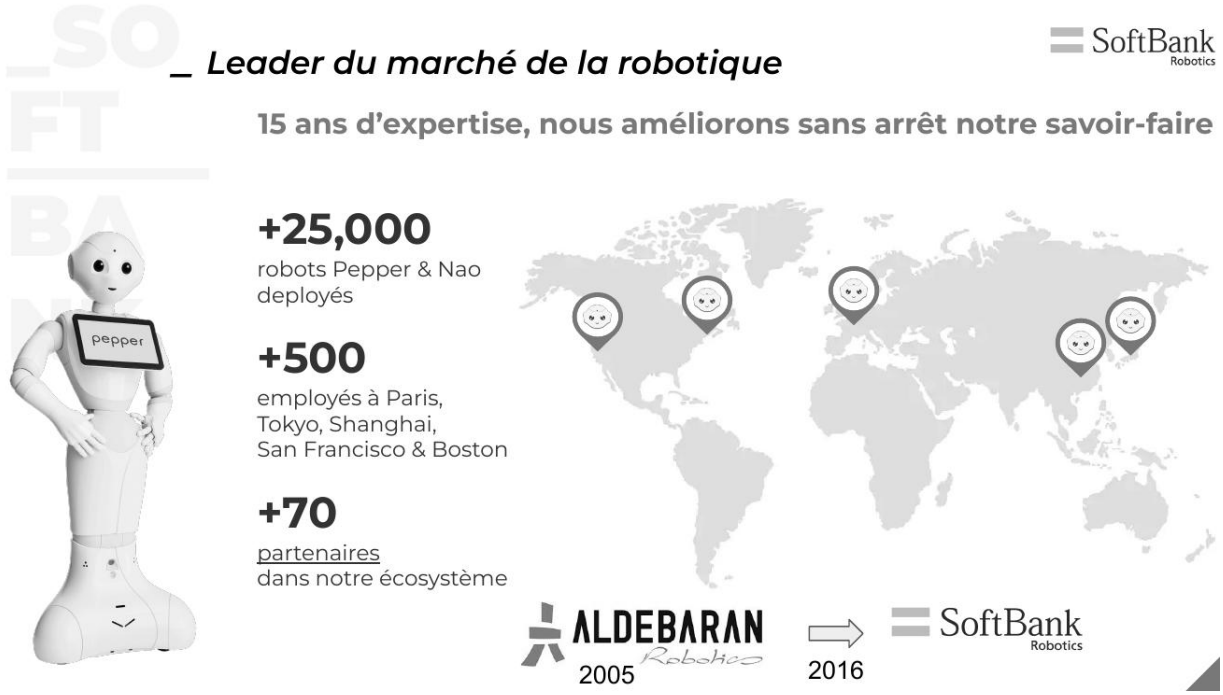
Pepper est aujourd'hui disponible pour les entreprises et le secteur de l'éducation. Déjà plus de 2000 entreprises ont adopté Pepper à travers le monde pour les assister de façon innovante dans l'accueil, l'information et l'orientation de leurs visiteurs.

- 20 degrés de liberté qui lui confèrent des mouvements naturels et expressifs.
- Reconnaissance vocale et dialogue disponibles dans 15 langues.
- Modules de perception pour reconnaître et suivre son interlocuteur du regard.
- Capteurs tactiles, LEDs et micros pour des interactions multimodales.
- Capteurs infrarouges, protections (*bumpers*), centrale inertielle, caméras 2D et 3D et sonars pour une navigation omnidirectionnelle et autonome.
- Plateforme ouverte et entièrement programmable. »

(Source : https://www.softbankrobotics.com/emea/fr/pepper)

Le degré de liberté correspond au nombre de mouvement indépendants que le robot peut effectuer.

Annexe 1b : Chiffres clefs de la flotte de Peppers (source SBRE, mars 2020)

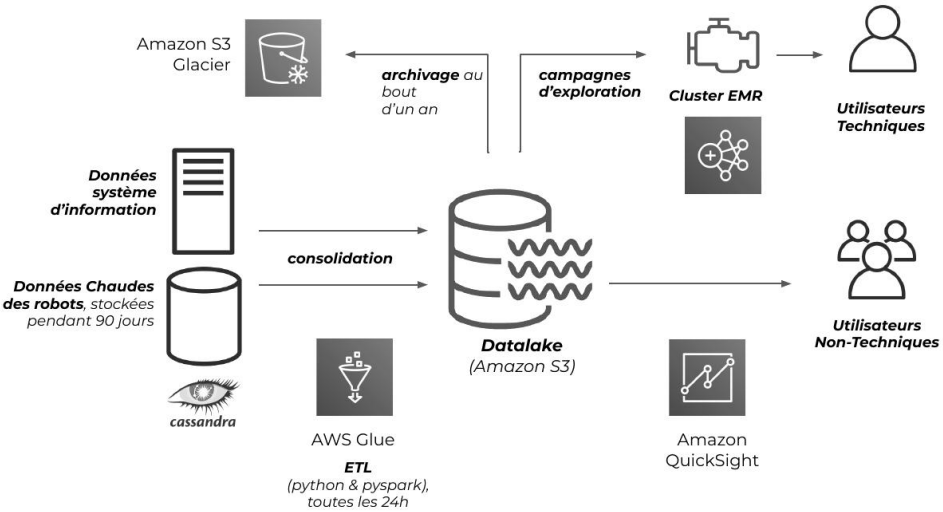


Annexe 2 : Points de vigilance concernant l'émission des données des robots vers le nuage informatique

L'entreprise SBRE mène une réflexion continue sur les critères d'émission des données :

- Le niveau de granularité des données remontées : un premier critère concernant le caractère brut ou agrégé des données. Par exemple, les données émises par le robot doivent-elles concerner chaque capteur ou plusieurs capteurs interdépendants ? Tandis que les données brutes permettent une analyse au cas par cas des capteurs, les données retravaillées sont plus faciles à manipuler.
- Le caractère personnel des données émises par les robots : les données émises par le robot ne doivent pas permettre de produire ou transmettre des informations permettant d'identifier la personne utilisatrice des services de Pepper.
- La taille des données émises : l'émission de lots de données de taille trop conséquente peut surcharger le réseau internet de la personne utilisatrice et générer à terme d'importantes quantités de données à historiciser et maintenir.
- La priorisation des données : un travail amont est nécessaire avec les équipes de Recherche & Développement pour identifier les données les plus utiles.
- L'intervalle nécessaire entre l'émission de données : une émission de données trop fréquente peut entraver le fonctionnement du robot.

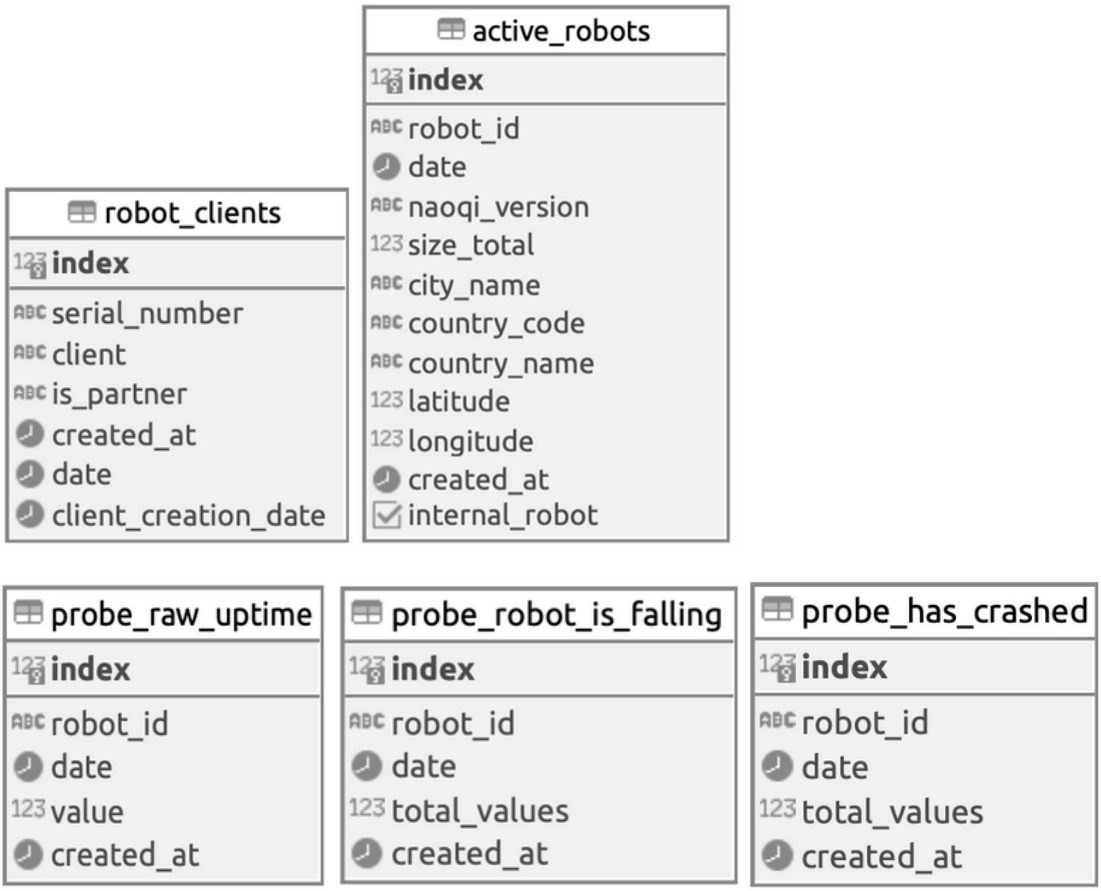
Annexe 3 : infrastructures participant à la collecte, à la consolidation et au stockage des données (schéma simplifié issue de source SBRE, mars 2020)



- Le service *AWS Glue* permet de consolider les données issues des robots et du système d'information de SBRE afin de préparer leur exploitation pour l'analyse, les rapports (*reporting*), l'apprentissage algorithmique (*machine learning*), le développement des applications, etc.
- Le service *Amazon QuickSight* permet d'alimenter les tableaux de bord fournis aux clients et aux utilisateurs internes non techniques pour enrichir la réflexion et contribuer aux prises de décisions.

- Le service *Amazon EMR* (anciennement « *Amazon Elastic MapReduce* ») est une plateforme de grappe (*cluster*), à destination essentiellement des utilisateurs techniques (roboticiens, analystes de données, etc.), permettant de traiter et d'analyser de grandes quantités de données.
- *Amazon Glacier* est un service de sauvegarde et d'archivage de données.

Annexe 4 : Tables des données extraites du lac de données en vue de la production de tableau de bord



La table robot_clients concerne les robots possédés par les clients (versus les prototypes). La table contient donc le champ serial_number qui indique le numéro de série du robot, ainsi que nom du client ou du partenaire qui l'utilise.

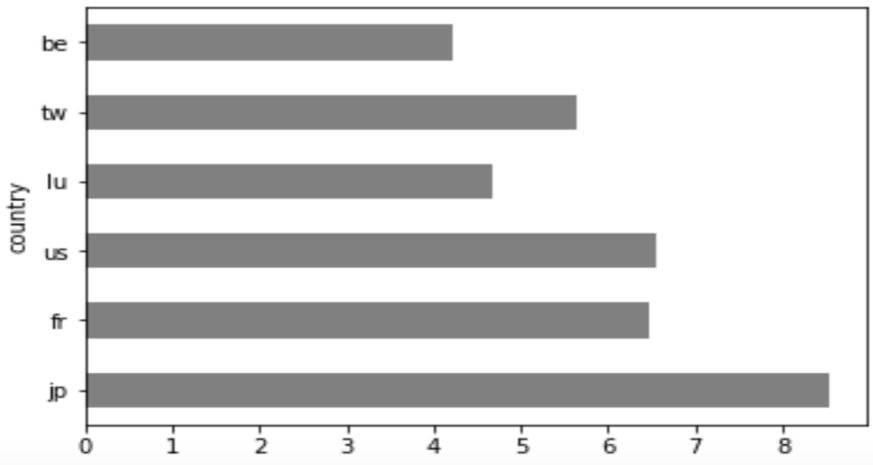
La table active_robots recense les robots en activité. La table contient notamment le champ robot_id permettant d'identifier le robot (équivalent à serial_number) et le champ correspondant à la version du système d'exploitation naoqi_version. Également dans cette table, le champ internal_robot indique si le robot est interne ou externe à la flotte.

Annexe 5 : Exemple de tableau de bord permettant le suivi de la flotte de Pepper (données fictives)

country	active_robots	active_days	total_uptime_seconds	mean_active_days_per_robot	total_uptime_hours	total_uptime_hours_per_active_day
jp	7915	1245110	3,8304E+10	157,31	10639864,5	8,55
fr	9548	99223	2313315327	10,39	642587,59	6,48
us	2101	42129	994301856	20,05	276194,96	6,56
lu	1009	42393	713431104	42,01	198175,31	4,67
tw	134	32219	654193987	240,44	181720,55	5,64
be	99	21939	333231954	221,61	92564,43	4,22

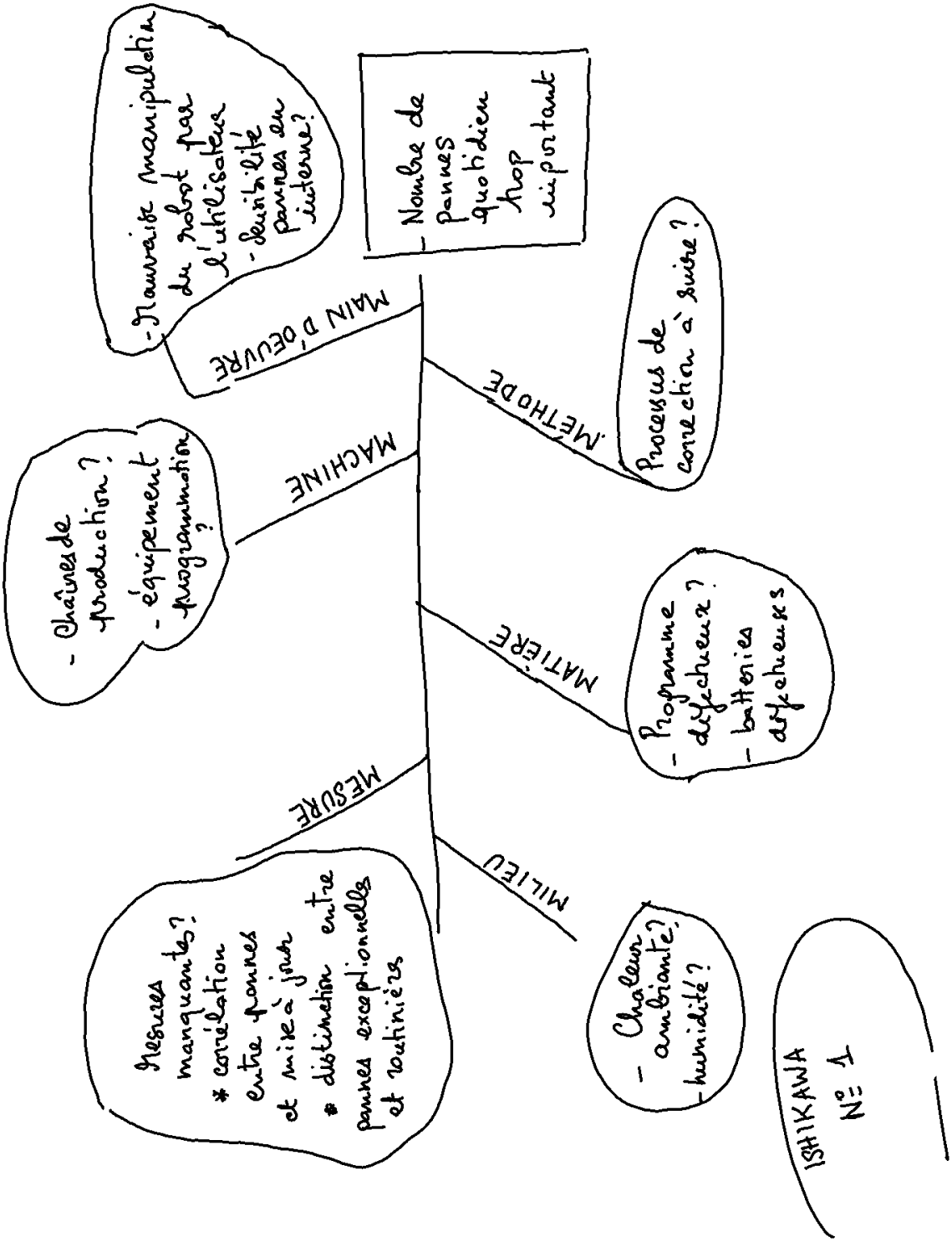
L'extrait de code suivant, écrit en Python dans un document interactif Jupyter Notebook présente sous la forme d'un diagramme en barre le nombre d'heures d'activité journalier (en abscisse) par pays (en ordonnée). Il utilise pour ce faire la bibliothèque d'analyse de données Pandas series.plot.barh. Les rectangles sur fond gris correspondent au code source dans le langage Python. Les différents mots sont colorés en fonction de leur rôle (coloration syntaxique). Apparaît ensuite l'adresse en mémoire du résultat du programme, ainsi que le résultat du programme.

```
[9]: import pandas as pd
     df = pd.read_csv("active_cooked.tsv", sep='\t', index_col=0)

[10]: df.plot.barh(y="total_uptime_hours_per_active_day", color="gray", legend=False)

[10]: <matplotlib.axes._subplots.AxesSubplot at 0x7ff3f99650b8>
```



Annexe 6 : Extrait d'un diagramme Ishikawa utilisé lors d'une réunion de remue-méninges

Afin de réfléchir de façon plus systémique aux différents obstacles à la qualité des robots, le diagramme suivant a été produit lors d'une première réunion informelle sur la qualité regroupant diverses expertises (mécaniques, production, maintenance, développement et management).



Annexe 7 : « Bien sécuriser les API, le nouveau Graal des entreprises ? » (extrait)

« Les entreprises utilisent des API (*Application Programming Interface* ou interface de programmation) pour faire communiquer des applications les unes avec les autres.

[…]

À l'heure actuelle, les entreprises multiplient les innovations de l'automatisation des chaînes de montage et de la chaîne logistique (*supply chain*) et augmentent, par conséquent, le risque de cyberattaque en raison de ces nouvelles portes d'entrée dans leurs systèmes informatiques. Dans le cas de l'objet connecté (IoT) et de l'industrie 4.0, les architectures des systèmes d'information modernes s'appuient sur des API aussi bien pour des usages internes que pour des usages externes, les nouveaux modes de communication actuels et la digitalisation des échanges forçant les usines à sortir de leur autarcie pour communiquer avec l'extérieur.

Cependant, toutes les données ne se valent pas et ne nécessitent pas le même niveau de protection. La stratégie de sécurisation des API dépend surtout du type de données transférées. À partir du moment où une application tierce se connecte à une API, il faut connaître la manière dont cette application redistribue les informations dans le nuage (*cloud*) ou ailleurs. Mais la sécurité n'est pas une simple « check-list » (utilisation de jeton – *token* –, de protocole sécurisé, etc.) mais bien de discipline et d'usages. Pour parvenir à bien sécuriser les APIs, il faut prendre en compte à la fois l'architecture et le design, mais aussi le développement du code des APIs et des applications exposants ces dernières […] »

Source : « Bien sécuriser les API, le nouveau Graal des entreprises ? » – Fabrice Hugues, Directeur innovation et solutions chez Software AG – 05/10/2019 :
https://www.usinenouvelle.com/article/avis-d-expert-bien-securiser-les-api-le-nouveau-graal-des-entreprises.N890474

Annexe 8 : exemple d'interface (API) codée dans le langage Python (extrait).

Cet exemple présente le minimum nécessaire à l'écriture d'une API en Python :

- démarrage d'un serveur HTTP (écoutant sur le port 8080) ;
- traitement d'une requête HTTP utilisant la méthode « GET » et routage en fonction du chemin de la ressource demandée (ici « /collection/ » ou « /collection/123 » par exemple) ;
- extraction de données en provenance d'une base de données SQLite ;
- envoi d'entêtes (HTTP), sérialisation de données (en XML) et envoi de la réponse au client.

```
from http.server import BaseHTTPRequestHandler, HTTPServer
import sqlite3

class ExampleAPI(BaseHTTPRequestHandler):
   def do_GET(self):                                  #self fait référence à la requête HTTP
      if (self.path.find('/collection') == 0):        #l'attribut path indique le chemin de la ...
         itemID = self.path[len('/collection/'):]     #… ressource ; exemple : /collection[/id]

         cnx = sqlite3.connect('database.db')
         cursor = cnx.cursor()                        #les requêtes s'effectuent via un curseur
         if itemID == '':
            cursor.execute('select id, label from collection')
         else:
            cursor.execute('select id, label from collection where id = :id', { 'id' : itemID })

         item = cursor.fetchone()                     #renvoie le premier enregistrement …
         if item == None:                             #… sous la forme d'une liste [id, 'label' ]
            self.send_error(404, 'Not Found')
            self.end_headers()
         else:
            self.send_response(200)
            self.send_header('Content-Type', 'application/xml')
            self.end_headers()

         xml = '<collection>'
         while item != None:
            xml += '<item><id>' + str(item[0]) + '</id><label>' + item[1] + '</label></item>'
            item = cursor.fetchone()                  #renvoie l'enregistrement suivant
         xml += '</collection>'

         self.wfile.write(bytes(xml, 'utf-8'))        #envoi du corps de la réponse
         cnx.close()

      else:
         self.send_error(400, 'Bad Request')
         self.end_headers()

if __name__ == '__main__':
   httpd = HTTPServer(('', 8080), ExampleAPI)
   httpd.serve_forever() ; httpd.server_close()
```