TIPE 2018

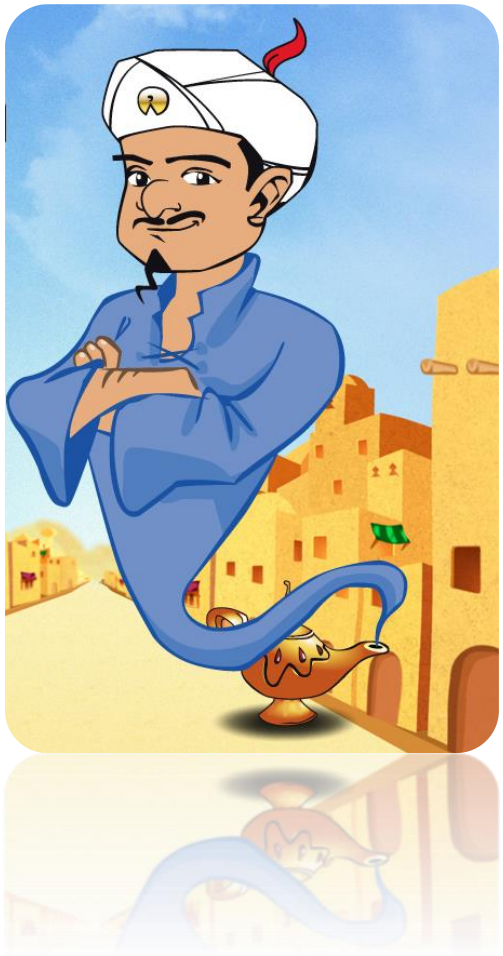
TAUPINATOR

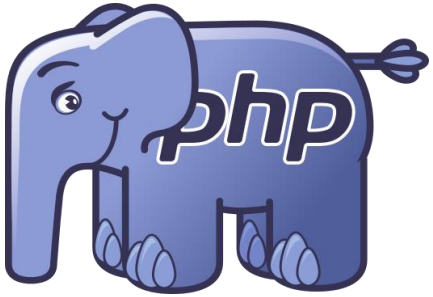
L'algorithme de la taupe

- Ian RODRIGUEZ -

Origine du projet

- Akinator, le génie qui trouve tout
- Projet en informatique (acquérir des connaissances)
- Pas un simple « Qui est-ce » !





TAUPINATOR

I. But et objectifs

II. Différentes manières d'arriver au résultat

III. 1ers problèmes et solutions techniques

IV. Évolution du modèle (V1 à V4)

V. Perspectives d'amélioration

- Problématique retenue -

- Ecrire un algorithme permettant de déterminer un personnage dans une liste donnée avec :
  - Un minimum de questions (dichotomiques) posées
  - Une prise en compte des erreurs éventuelles de l'utilisateur

Personnage	Nombre de questions posées
Edith Piaf	9
Mark Zuckerberg	14
Steve Jobs	21
Barack Obama	16
Donald Trump	16
Bill Gates	12
Hillary Clinton	14
François Hollande	15
Nicolas Sarkozy	13
Adolf Hitler	10

Akinator

Moyenne de 14 questions/personnage
(sur ces 10 personnages)

Base de données de plus de 200.000 personnages

- Objectifs -

1 - Conception d'un algorithme naïf à partir de nos connaissances sur Akinator

- L'erreur de l'utilisateur
- Les performances (mesure/optimisation)
- La méthode de recherche/structure du programme

Gestion des données : MySQL

- Gestion simplifiée des données : l'outil PhpMyAdmin

- Une base de données sur un unique serveur

- Conçu pour le traitement d'une grande quantité de données

- Une documentation importante et de nombreux modules et outils déjà intégrés

- Utilisation sur un serveur distant (Apache+serveur web+serveur sql)

Organisation de la BDD

	Table	Action	Lignes	Type	Interclassement	Taille	Perte
☐	dichotomie	Afficher Structure Rechercher Insérer Vider Supprimer	7	InnoDB	latin1_swedish_ci	16 Kio	-
☐	personnages	Afficher Structure Rechercher Insérer Vider Supprimer	84	InnoDB	latin1_swedish_ci	16 Kio	-
☐	questions	Afficher Structure Rechercher Insérer Vider Supprimer	242	InnoDB	latin1_swedish_ci	160 Kio	-

Table dichotomie (V1)

#	Nom	Type	Interclassement	Attributs	Null	Défaut	Extra
1	ID	int(11)			Non	*Aucune*	AUTO_INCREMENT
2	question	varchar(200)	latin1_swedish_ci		Non	*Aucune*	
3	mot_cle	varchar(200)	latin1_swedish_ci		Non	*Aucune*	
4	valeur	varchar(200)	latin1_swedish_ci		Non	*Aucune*	

Table personnage

#	Nom	Type	Interclassement	Attributs	Null	Défaut	Extra
1	ID	int(11)			Non	*Aucune*	AUTO_INCREMENT
2	nom	varchar(50)	latin1_swedish_ci		Non	*Aucune*	
3	mots_cle	text	latin1_swedish_ci		Non	*Aucune*	
4	statistiques	int(3)			Non	*Aucune*	

Table question (intelligentes)

#	Nom	Type	Interclassement	Attributs	Null	Défaut	Extra
1	ID	int(11)			Non	*Aucune*	AUTO_INCREMENT
2	question	varchar(200)	latin1_swedish_ci		Non	*Aucune*	
3	mots_cle	varchar(200)	latin1_swedish_ci		Non	*Aucune*	
4	valeur	varchar(200)	latin1_swedish_ci		Non	*Aucune*	
5	personnages	text	latin1_swedish_ci		Oui	*NULL*	
6	statistiques_posee	int(5)			Oui	0	
7	statistiques_persos	varchar(2000)	latin1_swedish_ci		Oui	0	

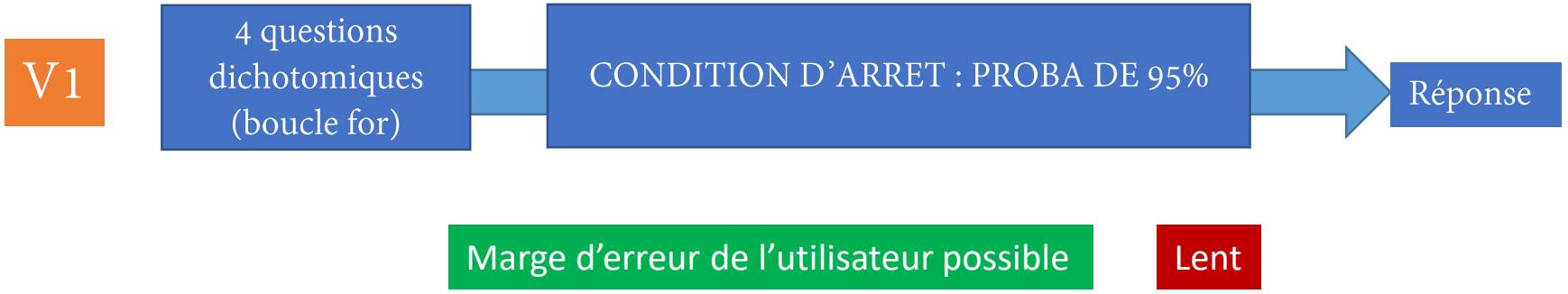
Exemple

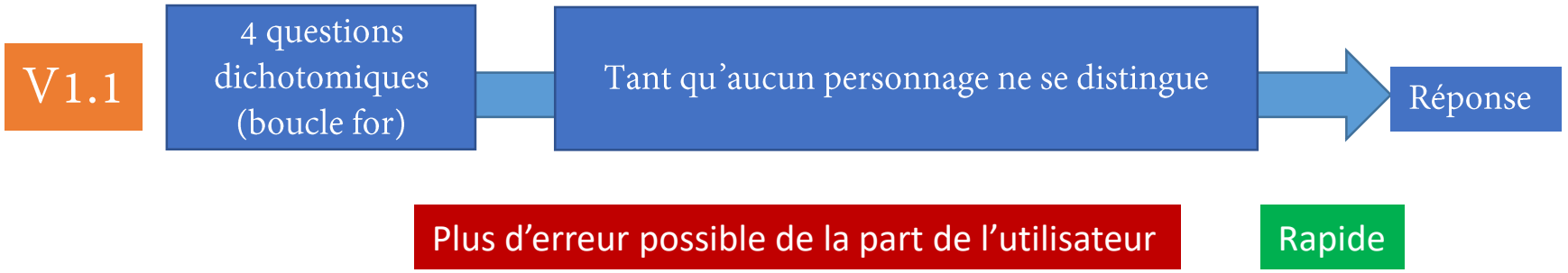
Données sur un personnage

Colonne	Type	Fonction	Null	Valeur
ID	int(11)			2
nom	varchar(50)			Mark Zuckerberg
mots_cle	text			informatique|silicon_valley|facebook|pdg|vivant|usa|riche|harvard|homme
statistiques	int(3)			11

1ère version et base de l'algorithme (V1 et V1.1)

- Fonctionnement : Questions basées sur le personnage à la plus haute probabilité





2ème version : Dichotomie améliorée

`DICHOTOMIX()`

- Couper en deux l'intervalle des solutions possibles : trouver la meilleure question !

Comment ?

On a : la liste contenant les personnages associés à leurs mots clé respectifs et leurs probabilités

On veut :

Personnage	Poids
Adolf Hitler	.65
Barack Obama	.65
Guy Georges	.65
Marilyn Monroe	.65
Edith Piaf	.25
Marie Curie	.12

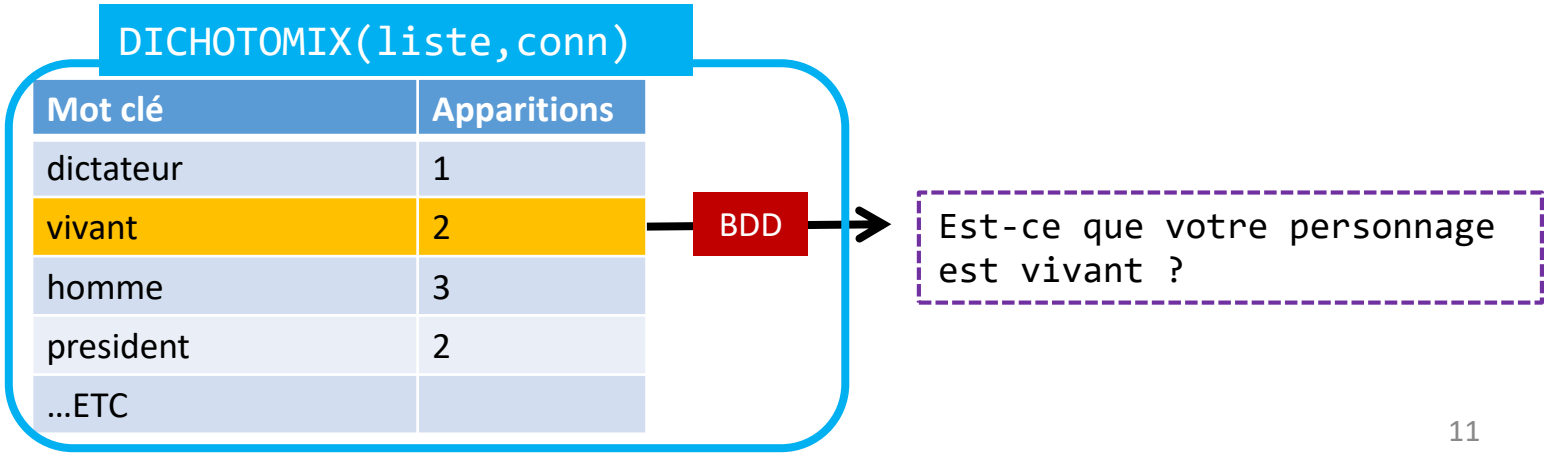
Question validée par un groupe

Est-ce que votre personnage a été chef d'état ?

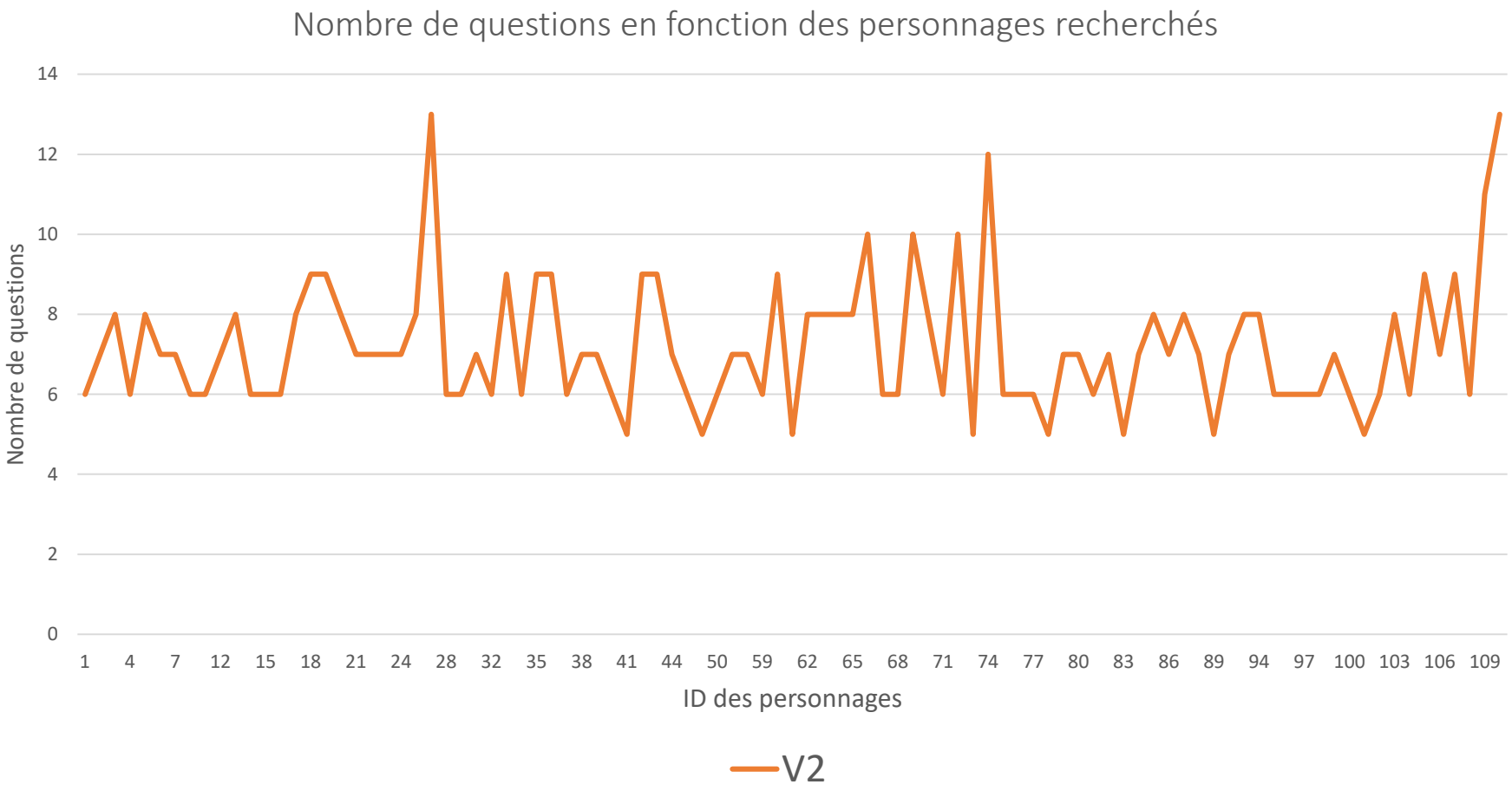
Illustration de la dichotomie améliorée

Adolf Hitler : dictateur|president|mechant|allemagne|homme|guerre|mort|politique|moustache|reich|mort_1945
Barack Obama : president|president_usa|democrate|politique|vivant|noir|usa|harvard|homme
Guy Georges : homme|vivant|serialkiller|prison|tueur|violeur|violeur_1990
Marilyn Monroe : femme|mort|usa|acteur|chanteur|beaute|kennedy|suicide

- Création d'une liste avec tous les mots clé en commun associés à leur nombre « d'apparitions » (boucle 1)
- Récupération dans la BDD de la question associée au mot clé se rapprochant le plus de la valeur souhaitée (boucle 2)



Résultats de la version 2



Moyenne de 7.2 questions/personnage

3ème version : Système de vérification et question de sécurité

- Question de sécurité
- Enregistrement dans la BDD de nouveaux personnages

Personnage	Poids
Marilyn Monroe	.80
Guy Georges	.65

Question unique à chaque perso

Votre personnage a-t-il chanté un « happy birthday, M. President » assez spécial ? (OUI/NON)

```
Si la réponse est NON :
```

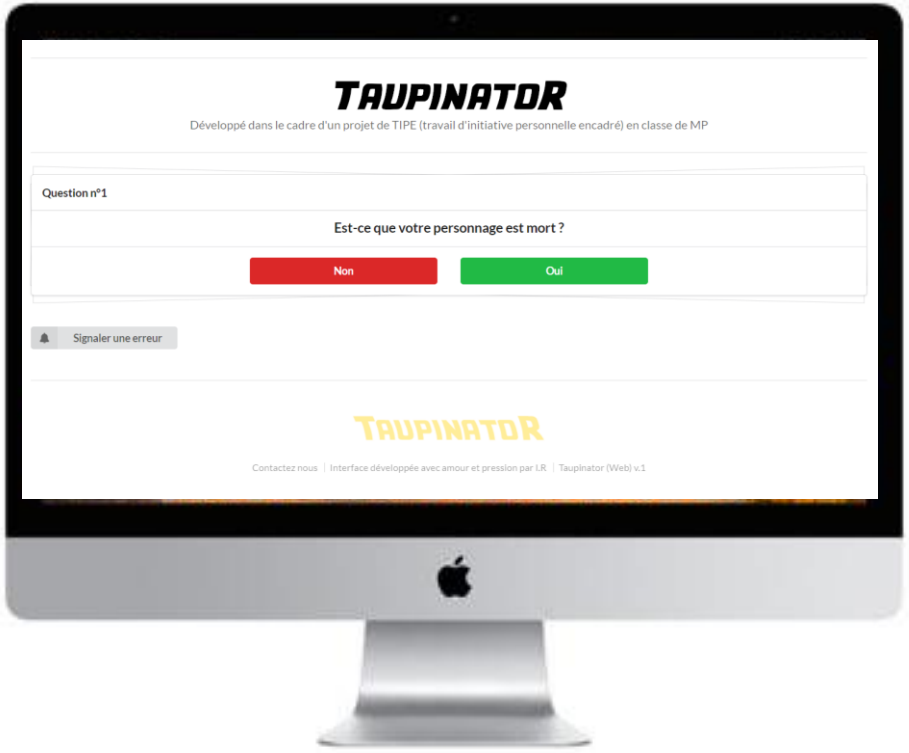
De qui s'agit-il ?

Ajout dans la table de vérification du nouveau personnage et des mots clé

4ème version de l'algorithme : interagir avec l'utilisateur

- Obtenir des statistiques sur les parties (erreurs de l'utilisateur)
- Compléter la base de données
- Améliorer les performances de l'algorithme
- Interface de contrôle / d'administration

- Interface graphique (module Tkinter)
- Marge d'erreur possible et adaptable facilement



4ème version de l'algorithme : taupinator.fr

- Implémentation de l'algorithme en PHP 7
- Site en HTML5/PHP7/Semantic UI
- Marge d'erreur possible et adaptable facilement

Exemples de l'adaptation de l'algorithme pour le web

Conservation des données entre pages → $_SESSION

Sécurisation des données reçues →

ID	token	id_question	IP	timestamp
469	15s1dukuxtxdxd9	111	90.62.159.106	1526661386
470	inei86f8ag0n32p	112	90.62.159.106	1526661430

Accessibilité du site « H24 » → OVH.com

Mesure de l'audience du site et analyse des parties jouées → Google Analytics

Interface d'administration de taupinator.fr

TAUPINATOR

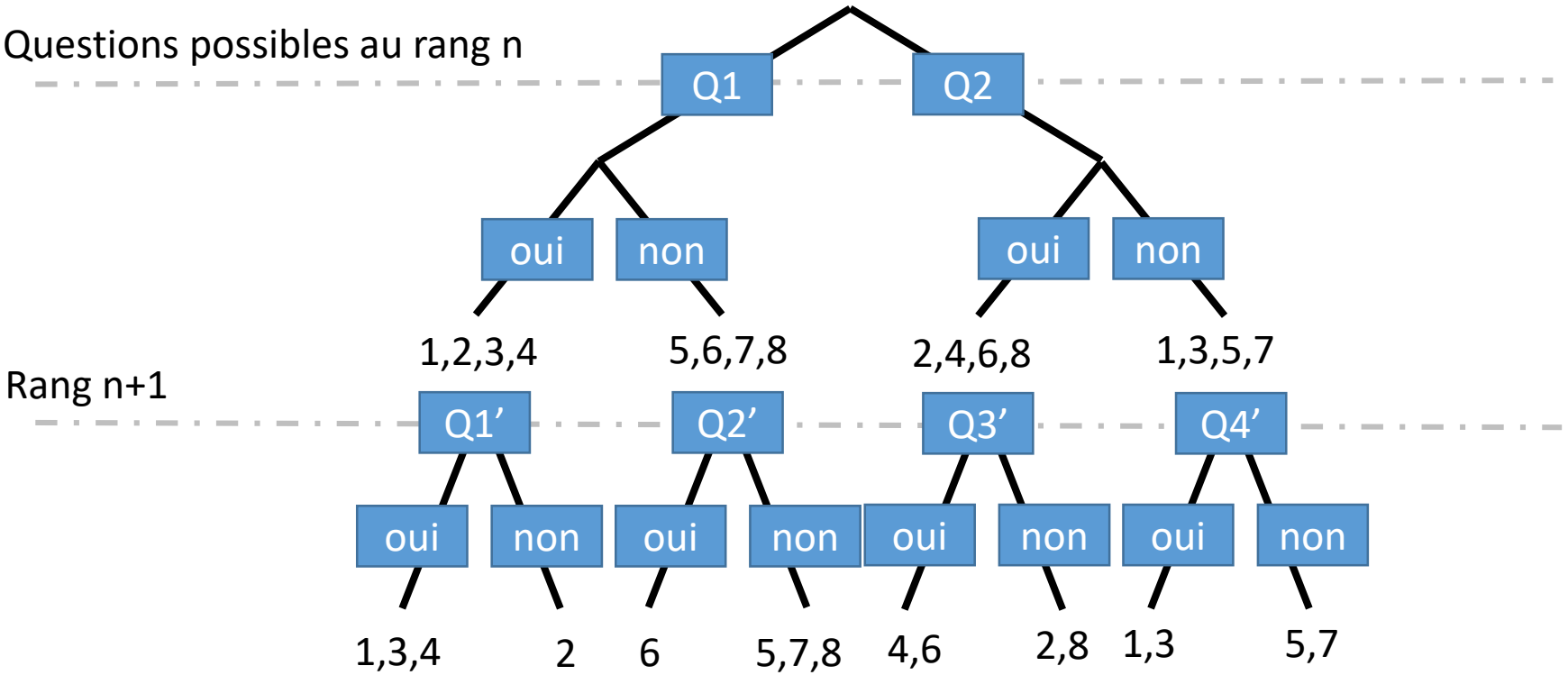
Statistiques de Taupinator

81	21	25.9
PARTIES	PERSONNAGES NON TROUVÉS	% NON TROUVÉS

Personnage trouvé	Personnage pensé	Nombre de questions fausses	Nombre de questions posées	Erreurs	IP	Meta	Date
Mahatma Gandhi	Socrate	3	7	politique: OUI, tue: OUI, liberte: OUI	90.62.159.106	id=85&nb_questions=8utilisateur=Mozilla/5.0 (Linux; Android 7.0; SM-A520F Build/NRD90M) AppleWebKit/	Fri, 18 May 2018 18:41:46 +0200
Steeve Jobs	Mark Zuckerberg	1	9	mort: OUI	91.161.222.226	id=2&nb_questions=10utilisateur=Mozilla/5.0 (Linux; Android 8.0.0; SM-G950F Build/R16NW) AppleWebKit	Fri, 18 May 2018 20:47:41 +0200
Nelson Mandela	Rosa Parks	1	6	politique: OUI	92.184.105.241	id=33&nb_questions=7utilisateur=Mozilla/5.0 (iPhone; CPU iPhone OS 10_3_3 like Mac OS X) AppleWebKit	Sat, 19 May 2018 11:03:46 +0200
Cristiano Ronaldo	Zinedine Zidane	1	8	club_juventus: NON	90.62.159.106	id=31&nb_questions=9utilisateur=Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,	Sat, 19 May 2018 23:12:17 +0200
Winston Churchill	Adolf Hitler	1	5	ecrivain: OUI	90.100.180.77	id=1&nb_questions=6utilisateur=Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,	Sun, 20 May 2018 21:41:13 +0200
Michael Jackson	Pablo Escobar	3	7	politique: NON, usa: OUI, mort_tragique: OUI	90.36.89.232	id=16&nb_questions=8utilisateur=Mozilla/5.0 (Linux; Android 7.0; SAMSUNG SM-A320FL Build/NRD90M) App	Mon, 21 May 2018 04:21:40 +0200
Charles de Gaulle	Winston Churchill	1	8	general: OUI	80.12.37.12	id=67&nb_questions=9utilisateur=Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML	Tue, 22 May 2018 15:02:07 +0200

Evolution du modèle : le raisonnement rétrograde

Personnage	1	2	3	4	5	6	7	8
Attribut 1								
Attribut 2								
Attribut 3								



-Annexes-

retrotomix()

```
def retrotomix(l,q,cursor,v=1):

    # Mise à jour de l1 sur la base des personnages à la + haute proba
    l1 = probabilites_triage(l,1)
    l1 = l1[:]
    # Liste des mots clé communs aux personnages ayant la probabilité MAX
(déjà testés)
    liste_mots_cle = []

    # Nombre de personnages qui doivent satisfaire un mot clé parfait
    nb_ref = len(l1)/2
    nb_ref_retro = len(l1)/4

    # Nombre actuel de personnages qui satisfont le "meilleur mot clé"
    nb_actuel = 10000

    # Liste contenant les mots clés possibles associés à leur nombre
d'apparition
    liste_mots_cle_app  =[]

    # Liste contenant le 1er mot clé, son ID, le second mot clé, puis le NB
ref du second
    retrotomix = []

    mc = 0
    while mc != 1:
        # On récupère tous les mots clé en commun et n'étant pas déjà dans
les questions posées :

        for i in range(len(l1)):
            for mot_cle in l1[i][2] :
                if (mot_cle not in liste_mots_cle and mot_cle not in q) :
                    liste_mots_cle+=[mot_cle]

        # On compte le nombre de fois qu'un mot clé apparaît
        for mot_cle in liste_mots_cle:
            nb=0
            for i in range(len(l1)):
                if(mot_cle in l1[i][2] and mot_cle != ''): #REPARATION !!
OUFF !
                    nb+=1

            nb = abs(nb_ref-nb) #### C H A N G E M E N T ###
            liste_mots_cle_app.append([mot_cle,nb])
            liste_mots_cle_app= sorted(liste_mots_cle_app, key=lambda
x:x[1], reverse=False) # A changer ?

        # On parcours les 4 premiers résultats de la liste de mots clé :
les 4 plus adéquats
        for i in range(min(len(liste_mots_cle_app),4)) : # Sécurité : si
moins de 4 mots clés !

            # On crée une nouvelle base de questions, pour les mêmes motifs
            base_questions_retrograde = q[:]

            # Création base questions
            base_personnages_retrograde = l1[:]

            # On lui ajoute le mot clé
            base_questions_retrograde+=[liste_mots_cle_app[i][0]]
            donnees = (liste_mots_cle_app[i][0],)
            cursor.execute("SELECT ID FROM questions WHERE valeur =
%s",donnees)

            # On prends l'ID de la question à la plus haute probabilité (un
peu inutile dans notre cas!)
            id_question = None
            for (ID) in cursor :
                if v == 1 :
                    print(ID[0])
                id_question = ID[0]

            # SECURITE
            if id_question != None:
                mc=1

            # On dispose de id_question - "future question"
            # Sélection des données liées à cette question
            cursor.execute("""SELECT
ID,question,mots_cle,valeur,personnages FROM questions WHERE ID = %s""",
(id_question, ))

            # Récupération des données
            for (ID, question, mot_cle, valeur, proba) in cursor :

                # On sort de la boucle en confirmant qu'une nouvelle
question a été trouvée
                deja_posee = False

                # Liste des mots clé communs aux personnages ayant la
probabilité MAX (déjà testés)
                liste_mots_cle_2 = []

                # On récupère tous les mots clé en commun et n'étant pas
déjà dans les questions posées :
                for j in range(len(base_personnages_retrograde)):
                    for mot_cle in base_personnages_retrograde[j][2] :
                        if (mot_cle not in liste_mots_cle_2 and mot_cle not
in base_questions_retrograde) :
                            liste_mots_cle_2+=[mot_cle]

                # On compte le nombre de fois qu'un mot clé apparaît
                for mot_cle in liste_mots_cle_2:
                    nb=0
                    for k in range(len(l1)):
                        if(mot_cle in l1[k][2] and mot_cle != ''): #
RÉPARATION !! OUF !
                            nb+=1

                    nb = abs(nb_ref_retro-nb) #### C H A N G E M E N T ###
                    # Mise à jour de retrotomix

retrotomix.append([liste_mots_cle_app[i][0],id_question, mot_cle,nb])

    # Triage par ordre croissant sur la base du nombre de référence du
second mot clé
    retrotomix= sorted(retrotomix, key=lambda x:x[3], reverse=False)
    # print(retrotomix)

    # On renvoi l'id de la question dont la réponse de l'utilisateur mènera
à la question donnant le plus petit nombre de personnages restants
    return(retrotomix[0][1])
```

Déroulé opérationnel du TIPE

- Choix de la structuration des données, d'une méthode de stockage et des modules nécessaires au développement en Python
- Développement de la 1ère version de l'algorithme de manière empirique
- Passage de l'algorithme vers un modèle de dichotomie, restructuration de la base de données et comparaison de la performance des modèles
- Entretien téléphonique avec Monsieur Arnaud Megret, PDG de la société Elokence et créateur d'Akinator, qui nous conforte dans l'idée de travailler sur la gestion des erreurs de l'utilisateur et se montre réticent sur l'utilisation des réseaux de neurones
- Mise en place de divers algorithmes visant à prendre en compte l'erreur de l'utilisateur : algorithme de vérification, question de sécurité
- Développement et mise en ligne du site taupinator.fr
- Analyse et exploitation des parties effectuées sur taupinator.fr afin de réduire les possibilités de mauvais personnages trouvés en modifiant la base de données

taupinator.py

```
#!/usr/bin/en python3
# -*- coding : utf-8 -*-

import mysql.connector
import random
import sys
# On modifie le chemin
sys.path.append('C:/Users/Ian/Desktop/TIPE/CODES')
from moteur import *
from interface import *


### A L G O R I T H M E ####### P R I N C I P A L #########
# Connexion à la base de données
conn =
mysql.connector.connect(host="localhost",user="root",password=
"",database="taupinator_v2")
cursor= conn.cursor(buffered=True)
cursor1= conn.cursor()
cursor2= conn.cursor()
cursor3= conn.cursor()

#Présentation
print("- TAUPINATOR - V 2.1")
print("Pensez à un personnage...")

base_personnages = []                  #LISTE QUI CONTIENDRA TOUS
LES PERSONNAGES
base_questions = []                    #LISTE QUI CONTIENDRA TOUS
LES MOTS CLÉ DÉJÀ POSES
nombre_question = 0                    #NOMBRE DE QUESTIONS
POSÉES AU TOTAL
lro = []                               # LISTE DES ID DES
RÉPONSES 'OUI'
lrn=[]                                 # LISTE DES ID DES
RÉPONSES 'NON'
precision=1                            # AJOUT V2.0  - PRÉCISION
DU SYSTÈME : NBRE DE QUESTIONS D'ÉCART


# RECUPERATION DES PERSONNAGES
cursor.execute("""SELECT * FROM personnages""")
for (ID, nom, mot_cle, valeur) in cursor :
    liste_mots_cle = mot_cle.split('|') # Créer liste de mots
clés à partir de mot cle ds BDD
    base_personnages+=[[ID, nom, liste_mots_cle, 0, valeur]] #
0 : probabilité + (V2 : ajout du 5ème élément : probabilité :
valeur)

# Triage des personnages de la liste par ordre de proba
base_personnages = probabilites_triage(base_personnages)

# 1ère partie - IA CONDITION PROBABILITÉ DES DEUX PERSOS
PRINCIPAUX MOINS ÉLEVÉE
while   base_personnages[0][3]-((precision-1) /
(nombre_question+1)) <= base_personnages[1][3] :

    # On initialise avec une valeur comme si la question avait
déjà été posée
    deja_posee = 1

    # Tant que l'on tombe sur une question déjà posée
    while deja_posee :

        #ID de la prochaine question
        id_question = dichotomix(base_personnages,
base_questions,cursor1,0)

        # Sélection des données
        cursor.execute("""SELECT
ID,question,mots_cle,valeur,personnages FROM questions WHERE
ID = %s""", (id_question, ))

        # Récupération des données
        for (ID, question, mot_cle, valeur, proba) in cursor :

            # On incrémente le nombre de questions avant le
calcul de probabilité
            nombre_question+=1

            # On pose la question
            reponse = input(question) # MODIF V4

            # Mise à jour des statistiques
            (lro,lrn) =
probabilites_statistiques_utilisateur(reponse,lro,lrn,id_quest
ion,cursor2,conn)

            #On modifie les probabilités

probabilites_maj_ia(base_personnages,valeur,int(reponse),nombr
e_question, proba,0)

            # On sort de la boucle en confirmant qu'une
nouvelle question a été trouvée
            deja_posee = False
            # On ajoute le nouveau mot clé à base_question
            base_questions+=[mot_cle]
```

```
            # Triages des probabilités
            base_personnages =
probabilites_triage(base_personnages)



# Affichage des probabilités
base_personnages = probabilites_triage(base_personnages)
print('----------------------')
print('JE PENSE À.....', base_personnages[0][1])
print(str(nombre_question)+' posées')

# Ajout système de probabilité dans la base de données
probabilites_statistiques(lro,lrn,base_personnages[0][0],curso
r3,conn,0)

cursor.close()
cursor1.close()
cursor2.close()
cursor3.close()

#################### F I N ### D E ##### L'A L G O R I T H M E
### P R I N C I P A L ########
```

moteur.py

```
#### - COTÉ UTILISATEUR - ##################
# FONCTION DE BASE : Augmenter ou diminuer la probabilité d'un
personnage
# arguments - l : liste du personnage - n : rang de la question
posée - p : proba de la question
def probabilites_calcul(l,n,p):
      l[3]=((l[3]*(n-1)+p)/n)


# FONCTION : MISE A JOUR DE LA PROBABILITÉ DE LA LISTE DE
PERSONNAGES
# arguments - l : liste des personnages - v : mot clé à chercher
dans les listes - c : choix de l'utilisateur (1:OUI ou 0:NON)
# - n : rang de la question
def probabilites_maj(l,v,c,n):

      if(c == 1):      #Si l'utilisateur à répondu OUI - les
personnages qui vérifient la condition voient leur proba augmenter
            p1 = 1
            p2 = 0
      else:            #Sinon, il a répondu NON - les personnages qui
vérifient la condition voient leur proba diminuer
            p1 = 0
            p2 = 1

      for personnage in l :
            #print(personnage[2])
            if (v in personnage[2]) == True:
                  print (bcolors.OKBLUE + 'Le
personnage',personnage[1], 'vérifie cette condition !' +
bcolors.ENDC)
                  probabilites_calcul(personnage, n,p1)
            else :
                  print (bcolors.WARNING + 'Le
personnage',personnage[1], 'ne vérifie pas cette condition !' +
bcolors.ENDC)
                  probabilites_calcul(personnage, n,p2)


# FONCTION : MISE A JOUR DE LA PROBABILITÉ DE LA LISTE DE
PERSONNAGES
# arguments - l : liste des personnages - v : mot clé à chercher
dans les listes - c : choix de l'utilisateur (1:OUI ou 0:NON) - q :
mode verbose
# - n : rang de la question
def probabilites_maj_ia(l,v,c,n,p,q=1):

      #Transformation de la chaîne de caractères p (probabilité) en
dictionnaire
      #Fonction trouvée sur fir3net.com
      proba = dict(x.split('=') for x in p.split(','))

      for personnage in l :
            #print(personnage[2])
            if(c == 1): #Si l'utilisateur à répondu OUI
                  p1 = float(proba.get(personnage[0],1)) # Correspond à l'ID du
personnage - si il existe, on le prend en compte !
                  p2 = 1 - float(proba.get(personnage[0],1))
            else:       #Sinon
                  p1 = 1 - float(proba.get(personnage[0],1))
                  p2 = float(proba.get(personnage[0],1))

            if (v in personnage[2]) == True :
                  if q == 1:
                        print (bcolors.OKBLUE + 'Le
personnage',personnage[1], 'vérifie cette condition !' +
bcolors.ENDC)
                  probabilites_calcul(personnage, n,p1)
            else :
                  if q == 1:
                        print (bcolors.WARNING + 'Le
personnage',personnage[1], 'ne vérifie pas cette condition !' +
bcolors.ENDC)
                  probabilites_calcul(personnage, n,p2)

# FONCTION IA : CHOIX D'UNE QUESTION ADÉQUATE SUR LA BASE DU
PERSONNAGE AYANT LA PLUS GRANDE PROBA
# arguments - l : liste des personnages - q : liste des mots clé des
questions - cursor : curseur BDD
def probabilites_question_ia(l, q, cursor):
      # Si plusieurs personnages ont la proba la plus élevée, ont
prendra celui qui a été joué le + de fois
      l=probabilites_triage(l)
      id_perso = l[0][0]
      id_question =0

      query = "SELECT ID,personnages FROM questions WHERE
(personnages REGEXP '" + str(id_perso) + "=') AND (mots_cle NOT
REGEXP '"+ '|'.join(q) + "')"
      cursor.execute(query)

      # On prends l'ID de la question à la plus haute probabilité
      proba_question = 0
      for (ID, proba) in cursor :
            proba = dict(x.split('=') for x in proba.split(','))
            #print(proba)
            # Conversion str nécessaire et int par dessus !
            if float(proba.get(str(id_perso))) >= proba_question :
                  id_question = ID
                  proba_question = float(proba.get(str(id_perso)))

      # Dès lors, la variable id_question contient l'ID de la
question la + adéquate
      return id_question


# OU : FONCTION IA POUR DÉPARTAGER CEUX QUI ONT LA PLUS GRANDE PROBA
(OU DIVISER L'INTERVALE EN 2)
```

```
# FONCTION : TRIAGE DES PROBABILITÉS PAR ORDRE DÉCROISSANT
(OBSOLÈTE)
# argument - l : liste complète des personnages
def probabilites_triage1(l):
      l=sorted(l, key=lambda x:x[3], reverse=True)
      return l

# FONCTION TRIAGE DES PROBABILITÉS PAR ORDRE DÉCROISSANT AMÉLIORÉ
# Trie des personnages ayant la proba la + élevée en fonctions des
fois où le personnage a été joué
# argument - l : liste complète des personnages - O : option renvoi
uniquement la liste l1 si égal à 1
def probabilites_triage(l, O=0):
      l=sorted(l, key=lambda x:x[3], reverse=True)
      # On récupère la probabilité maximale : qui correspond à celle
du premier perso (trié ainsi !)
      proba_max = l[0][3]

      l1 = [] # liste qui contiendra les personnages avec la plus
haute proba
      nb = 0 # Nombres de personnages à la même proba
      for personnage in l :
            if personnage[3] == proba_max :
                  l1+=[personnage]
                  nb+=1
      # On trie les personnages à la plus grande probabilité
      l1= sorted(l1, key=lambda x:x[4], reverse=True)

      # Si l'option est activée
      if O==1:
            return l1
      else:
            l = l1 + l[nb:]
            return l

# FONCTION D I C H O M I X - l1 : liste des personnages, l2 - liste
des mots clé de questions déjà posées
def dichotomix(l1,q,cursor,v=1):
      # Mise à jour de l1 avec uniquement les personnages à la +
haute proba
      l1 = probabilites_triage(l1,1)
      # Liste des mots clé communs aux personnages ayant la
probabilité MAX (déjà testés)
      liste_mots_cle = []
      # Nombre de personnages qui doivent satisfaire un mot clé
parfait
      nb_ref = len(l1)/2
      # Nombre actuel de personnages qui satisfont le "meilleur mot
clé"
      nb_actuel = 10000

      if v == 1 :
            print('NB REF :',nb_ref)

      # SECURITE
      mc=0
      while mc != 1:

            # On récupère tous les mots clé en commun et n'étant
pas déjà dans les questions posées :
            for i in range(len(l1)):
                  for mot_cle in l1[i][2] :
                        if (mot_cle not in liste_mots_cle and
mot_cle not in q) :
                              liste_mots_cle+=[mot_cle]

            # On compte le nombre de fois qu'un mot clé apparaît
            for mot_cle in liste_mots_cle:
                  nb=0
                  for i in range(len(l1)):
                        if(mot_cle in l1[i][2] and mot_cle !=
''): #REPARATION !! OUFF !
                              nb+=1

                  if nb == nb_ref:
                        choix_mot_cle = mot_cle
                        break
                  elif abs(nb_ref-nb) <= abs(nb_ref-nb_actuel):
#### C H A N G E M E N T ###
                        nb_actuel = nb
                        choix_mot_cle = mot_cle

                  if v == 1 :
                        print('NB : ',nb,' mot clé : ',mot_cle,
'val1', abs(nb_ref-nb), 'val2', abs(nb_ref-nb_actuel))

            if v == 1 :
                  print('MOT CHOISI :', choix_mot_cle)

            #On choisi la question comportant le mot clé :
choix_mot_clé
            donnees = (choix_mot_cle,)
            cursor.execute("SELECT ID FROM questions WHERE
valeur = %s",donnees)

            # On prends l'ID de la question à la plus haute
probabilité (un peu inutile dans notre cas!)
            proba_question = 0
            id_question = None
            for (ID) in cursor :
                  if v == 1 :
                        print(ID[0])
                  if int(ID[0]) >= proba_question :
                        id_question = ID[0]

            # SECURITE
```

```
                if id_question != None:
                        mc=1
                else :
                        # Question n'existe pas!
                        liste_mots_cle = []
                        q+=[choix_mot_cle]
                        nb_actuel = 10000 # REPARATION -
REINITIALISATION NB ACTUEL

                if(v == 1):
                        print(liste_mots_cle)
                        print('Q:',q)

        # Dès lors, la variable id_question contient l'ID de la
question la + adéquate
        return id_question


# FONCTION : MISE A JOUR DES STATISTIQUES DE L'UTILISATEUR
def
probabilites_statistiques_utilisateur(reponse,lro,lrn,id_question,cu
rsor,conn):
        if int(reponse) == 1:
                lro+=[id_question]
        else :
                lrn+=[id_question]

        # Incrémentation du compteur de question posée
        donnees = (id_question,)
        cursor.execute("UPDATE questions SET statistiques_posee =
statistiques_posee+1 WHERE ID = %s",donnees)
        conn.commit()

        return(lro,lrn)



### - COTÉ DEVELOPPEUR - ##################

# FONCTION : AFFICHAGE DES PROBABILITES DE CHAQUE PERSONNAGE PAR
ORDRE CROISSANT
def probabilites_affichage(l,n):
        l=probabilites_triage(l)
        print('-PROBABILITE DES PERSONNAGES')
        print('Nombre de questions :',n)
        for personnage in l :
                print(personnage[1],':',personnage[3])

# FONCTION : PROBABILITÉ - MISE A JOUR DES STATISTIQUES
# arguments - lro : liste des ID des questions où l'utilisateur a
répondu 'OUI' -lrn : liste des ID des questions où l'utilisateur a
répondu 'NON' - id_perso : ID du personnage trouvé
def probabilites_statistiques(lro,lrn,id_perso,cursor,conn,v=1):
        # OUI
        for i in lro:
                # Récupération des données probabilité 'OUI'
                donnees = (i,)
                cursor.execute("SELECT personnages,statistiques_posee
FROM questions WHERE ID = %s",donnees)
                for proba,stats in cursor :
                        proba1 = dict(x.split('=') for x in
proba.split(',')) # Conversion
                        if v:
                                print(proba1)
                                print('float :',
float(proba1.get(str(id_perso),0)))

                        if    float(proba1.get(str(id_perso),0)) != float(0)
:
                                remplacer =
str(id_perso)+'='+str(float(proba1.get(str(id_perso))))

                                # Calcul de la nouvelle probabilité
                                nouvelle_proba =
(float(proba1.get(str(id_perso)))*(stats-1)+1)/stats


                                remplacement =
str(id_perso)+'='+str(nouvelle_proba)
                                proba2=str.replace(proba, remplacer,
remplacement)
                                donnees = (proba2,i)
                                if v:
                                        print(proba2)
                                cursor.execute("UPDATE questions SET
personnages = %s WHERE ID = %s",donnees)
                                conn.commit()

        # NON
        for i in lrn:
                # Récupération des données
                donnees=(i,)
                cursor.execute("SELECT personnages,statistiques_posee
FROM questions WHERE ID = %s",donnees)
                for proba,stats in cursor :
                        proba1 = dict(x.split('=') for x in
proba.split(',')) # Conversion
                        if    float(proba1.get(str(id_perso),0)) != float(0)
:
                                remplacer =
str(id_perso)+'='+str(float(proba1.get(str(id_perso))))

                                # Calcul de la nouvelle probabilité
                                nouvelle_proba =
(float(proba1.get(str(id_perso)))*(stats-1))/stats
```

```
                    remplacement =
str(id_perso)+'='+str(nouvelle_proba)
                    proba2=str.replace(proba, remplacer,
remplacement)
                    donnees = (proba2,str(i))
                    cursor.execute("UPDATE questions SET personnages
= %s WHERE ID = %s",donnees)
                    conn.commit()

     donnees = (id_perso,)
     cursor.execute("UPDATE personnages SET statistiques =
statistiques+1 WHERE ID = %s",donnees)
     conn.commit()


#### Algorithme de vérification
def recherche_mc(mot,lr6bis):      #Renvoie True si le mot est dans lr1
et False sinon.
    a=lr6bis.find(mot)
    if a>=0:
        return(True)
    else:
        return(False)

def compare_mc(lr1,lr6bis):
     #lr1: liste utilisateur
     #Renvoi lr1-(lr1 inter lr6bis)  cad renvoi les mots clé présents
dans lr1 et pas présents dans lr6bis
    lr1bis=''
    n=len(lr1)
    mot=''
    j=0
    i=0
    while i<n:
        if lr1[i]=='|':
            mot=lr1[j:i]   #tant qu'on a pas rencontrer un | le mot
n'est pas terminé
            if recherche_mc(mot,lr6bis) == False:
                lr1bis+=mot + '|'
            j=i+1
        i+=1
    return (lr1bis)
```

taupinator.php

```php
<?php

// Afficher les erreurs à l'écran
ini_set('display_errors', 1);
// Enregistrer les erreurs dans un fichier de log
ini_set('log_errors', 1);
// Nom du fichier qui enregistre les logs (attention aux droits à
l'écriture)
ini_set('error_log', dirname(__file__) . '/log_error_php.txt');
// Afficher les erreurs et les avertissements
error_reporting(e_all);

session_start();

// CONNEXION A LA BDD
$bdd = new
PDO('mysql:host=teufiecopjtaupe.mysql.db;dbname=teufiecopjtaupe;charset=utf
8', 'teufiecopjtaupe', 'Trollman1289');

/* ======================================================== */
function nouvellePartie()
{
      $partie = [];
      array_push($partie, nouvelleBase());
      array_push($partie,[]);
      array_push($partie,[]);
      return($partie);

}

function token($var){
        $string = "";
        $chaine = "a0b1c2d3e4f5g6h7i8j9klmnpqrstuvwxy123456789";
        srand((double)microtime()*1000000);
        for($i=0; $i<$var; $i++){
            $string .= $chaine[rand()%strlen($chaine)];
        }
        return $string;
}


function nouvelleQuestion($base_personnages, $base_questions)
{
      global $bdd;
      global $num_question;
      global $partie;

      $precision=1;                                          # AJOUT V4  -
PRÉCISION DU SYSTÈME : NBRE DE QUESTIONS D'ÉCART
                                                             # AJOUT V4
- PRECISION DU SYSTEME : NBRE DE QUESTIONS POSÉES (VAUT 1 POUR 0 Q)

      if($base_personnages[0][3]-($precision /$num_question) <=
$base_personnages[1][3]) // Si les deux premiers personnages ont quasiment
la même proba : 1 question d'écart
      {
                  $id_question = dichotomix($base_personnages,
$base_questions);

                  // Sélection des données
                  $req = $bdd->prepare("SELECT
ID,question,mots_cle,valeur,personnages FROM questions WHERE ID = ?");
                  $req->execute(array($id_question));

                  // Récupération des données
                  $donnees = $req->fetch();

                  // On incrémente le nombre de questions avant le calcul
de probabilité
                  //$num_question++;

                  // ON POSE LA QUESTION
                  $question = $donnees['question']; # MODIF V4
                  $token = token(15);

                  $req = $bdd->prepare("INSERT INTO
utilisateurs_tokens(token,id_question,IP,timestamp) VALUES(:token,
:id,:ip,:timestamp)");
                  $req->execute(array(
                        'token' => $token,
                        'id' => $donnees['ID'],
                        'ip' => $_SERVER['REMOTE_ADDR'],
                        'timestamp'=> time()
                        ));



                  // Mise à jour des statistiques (?)
                  // (lro,lrn) =
probabilites_statistiques_utilisateur(reponse,lro,lrn,id_question,cursor2,c
onn)

                  // ON AFFICHE LA QUESTION
                  $affichage = '';
                  if($num_question <=1 && isset($_GET['new']))
                  {
                        $liste_personnages = '';
                        $req2 = $bdd->query("SELECT * FROM personnages
ORDER BY nom");
                        while ($personnage = $req2->fetch()) //
Simplification possible (?)
                        {
                              $liste_mots_cle = explode("|",
$personnage['mots_cle']);
                              $liste_personnages.='
                                          <div class="item">
                                              <!-- <img class="ui
avatar image" src="img/helen.jpg"> -->
                                              <div class="content">
                                                <div
class="header">'.$personnage['nom'].'</div>
                                              </div>
                                            </div>';
                              //array_push($base_personnages,
[$personnage['ID'], $personnage['nom'], $liste_mots_cle, 0]); // 0 : poids
par défaut
```

```
                        }
                              $affichage .='
                              <div class="ui modal">
                                <i class="close icon"></i>
                                <div class="header">
                                  Liste des personnages
                                </div>
                                <div class="content">
                                  <div class="description"> <!-- AEE462A7FA0
-->
                                    <div class="ui message"><div class="ui
horizontal divided list">'.$liste_personnages.'</div></div>
                                  </div>
                                </div>
                                <div class="actions">
                                  <div class="ui black deny button">
                                    Fermer
                                  </div>
                                </div>
                              </div>
                              <script>$(\'.ui.modal\')
.modal(\'show\');</script>';


                        }

                        $affichage .='    <!-- Question -->
                              <div class="ui piled segments">
                                <div class="ui segment">
                                  <h4>Question n°'.$num_question.'</h4>
                                </div>
                                <div class="ui segment">
                                  <h3 class="ui center aligned
header">'.$question.'</h3>
                                </div>
                                <div class="ui segment">
                                    <form
action="?id_page='.strval($num_question+1).'" method="POST">
                                    <div class="ui two column centered grid">
                                      <div class="four column centered row">
                                        <div class="column"><button type="submit"
name="non" class="ui button fluid red">Non</button></div>
                                        <div class="column"><button type="submit"
name="oui" class="ui button fluid green">Oui</button></div>
                                        <input type="hidden" name="token"
value="'.$token.'"/>
                                      </div>
                                    </div>
                                    </form>
                                </div>
                              </div>
                              <!-- Fin Question -->';

      }
      else { // Sinon : on a le bon personnage : pas de nouvelle question
            // AFFICHAGE PERSONNAGE TROUVÉ
            if(!isset($_POST['personnage']) AND !isset($_POST['valide']))
            {
                  $base_personnages = triage($base_personnages);

                  $pre_affichage='';
                  foreach(array_slice($base_personnages,1) as $personnage)
                  {
                        $pre_affichage.="<option
value=\"$personnage[0]\">$personnage[1]</option>";
                  }

                  $affichage = '                        <div class="ui
segment">
                                    <div class="ui vertical masthead center
aligned segment">
                                      <div class="ui text container">
                                      <!-- Image -->
                                        <img class="ui small circular
left floated image" src="img/mn_logo_w.png">

                                        <h2>Je pense à...
'.$base_personnages[0][1].' ! </h2>
                                        <div class="ui section
divider"></div>
                                        <form action="#" method="POST">
                                        <input type="submit" name="valide"
class="ui huge primary button" value="OUI, c\'est la bonne réponse"/>
                                        <div class="ui section
divider"></div>
                                          <div class="ui form">
                                            <div class="inline field">
                                              <label>Non, il s\'agit
de :</label>
                                              <select
name="personnage">'.$pre_affichage.'</select>
                                              <input type="submit"
name="erreur" style="margin-left:0.2em" class="ui submit button"
value="Envoyer"/>
                                            </div>
                                        </div>
                                        </form>
                                        <div class="ui section
divider"></div>
                                        <h4 class="red">Veuillez
répondre pour valider la partie svp</h4>
                                      </div>
                                    </div>
                                  </div>';

                  $contenu = 'personnage : '.$base_personnages[0][1].' - IP
: '.$_SERVER['REMOTE_ADDR'].' - DATE : '.strval(time())."\n";
                  file_put_contents('adm/log.txt', $contenu, FILE_APPEND);
            }
            else // Le joueur vient de valider sa partie
            {
                  // L'utilisateur a fait une erreur
                  if(isset($_POST['personnage']) AND
isset($_POST['erreur']))
                  {
                        $bonne_partie = 'false';
```

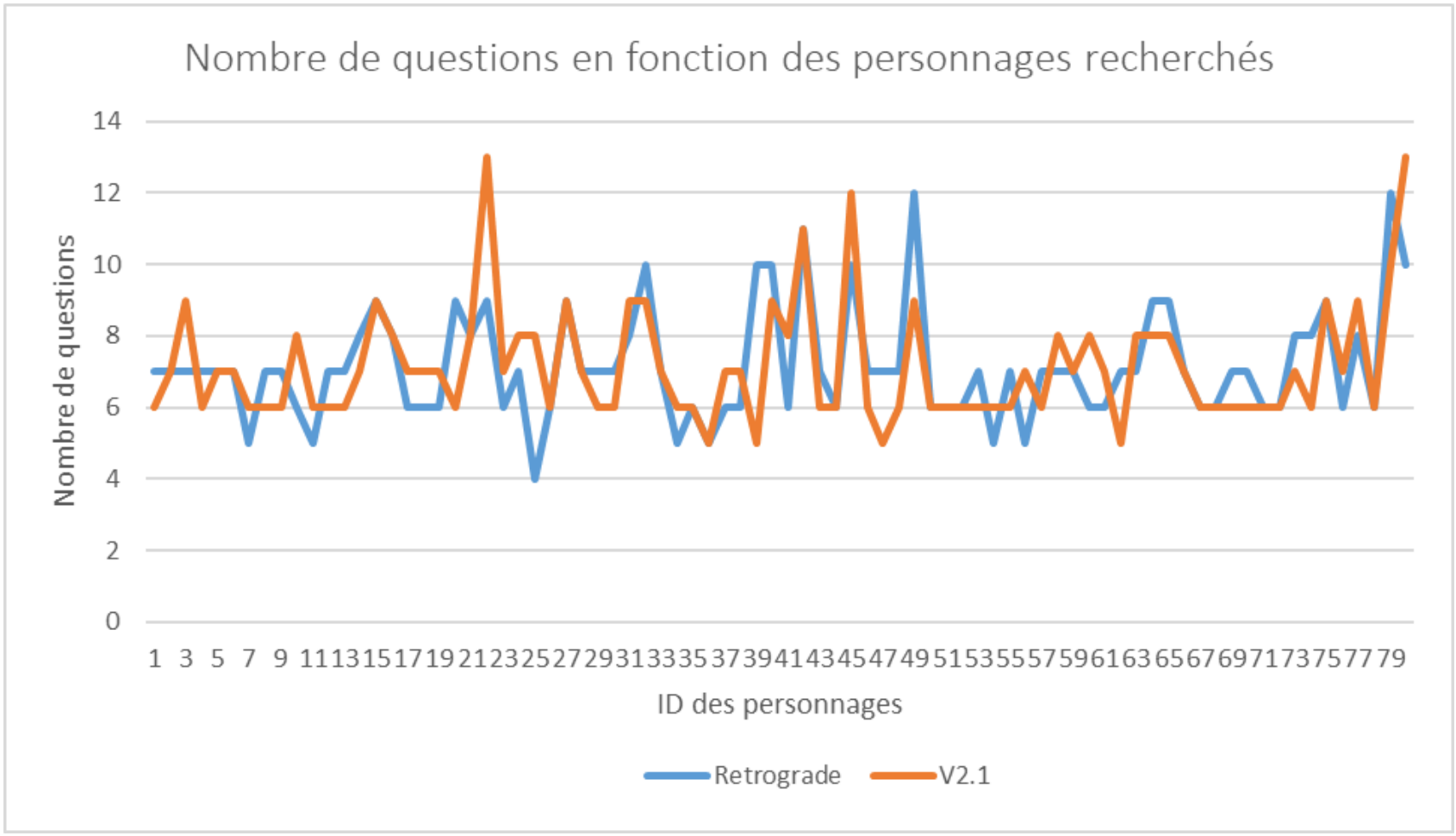
```
                        $meta =
'id='.htmlspecialchars($_POST['personnage']).'&nb_questions='.strval($num_q
uestion);
                  }
                  else
                  {
                        $bonne_partie = 'true';
                        $meta = 'nb_questions='.strval($num_question);
                  }

                  // Formatage de $base_personnages pour la BDD -> format
txt
                  $base_personnages_texte = '';
                  foreach($base_personnages as $personnage)
                  {
                        $base_personnages_texte.=implode(",",
$personnage)."|";
                  }

                  // Ajout des données dans la BDD
                  $req = $bdd->prepare("INSERT INTO
utilisateurs_parties(base_personnages,
base_questions,base_reponses,etat_partie,meta,IP,timestamp)
VALUES(:base_personnages, :base_questions,:base_reponses,
:etat_partie,:meta,:ip,:timestamp)");
                  $req->execute(array(
                        'base_personnages' => $base_personnages_texte,
                        'base_questions' => implode(",", $base_questions),
                        'base_reponses' => implode(",", $partie[2]),
                        'etat_partie' => $bonne_partie,
                        'meta' =>
$meta.'utilisateur='.$_SERVER["HTTP_USER_AGENT"],
                        'ip' => $_SERVER['REMOTE_ADDR'],
                        'timestamp'=> time()
                        ));

                  // On détruit la partie créée
                  session_destroy();

                  $affichage = '                        <div class="ui
segment">
                                    <div class="ui vertical masthead center
aligned segment">
                                        <div class="ui text container">
                                        <!-- Image -->
                                          <img class="ui small circular
left floated image" src="img/mn_logo_w.png">

                                          <h2>Merci d\'avoir joué :D !</h2>
                                          <div class="ui section
divider"></div>
                                          <a href="taupinator.php?new=1"
class="ui huge primary button"><i class="redo icon"></i> Nouvelle
partie</a>
                                          <a href="index.php" class="ui huge
button"><i class="home icon"></i></a>
                                          <div class="ui section
divider"></div>
                                            <div class="ui form">
                                                                  <h3>N\'hésitez pas à
partager Taupinator sur les réseaux sociaux</h3>
                                                                  <div class="fb-like" data-
href="http://taupinator.fr/" data-layout="standard" data-action="like"
data-size="large" data-show-faces="true" data-share="true"></div>
                                                            </div>
                                                            <div class="ui section
divider"></div>
                                                            <!-- <h4 class="red">Veuillez
répondre pour valider la partie svp</h4> -->
                                                      </div>
                                                </div>
                                          </div>';

                  }
            /* Ajout système de probabilité dans la base de données
            probabilites_statistiques(lro,lrn,base_personnages[0][0],cursor3,conn
,0) */

            }

            return($affichage);
}

function nouvelleBase()
{
      global $bdd;
      $base_personnages = [];
      $req = $bdd->query("SELECT * FROM personnages");
      while ($personnage = $req->fetch()) // Simplification possible (?)
      {
            $liste_mots_cle = explode("|", $personnage['mots_cle']);
            array_push($base_personnages, [$personnage['ID'],
$personnage['nom'], $liste_mots_cle, 0]); // 0 : poids par défaut
      }
      return($base_personnages);
}

function triage($l,$c=0)
{

      foreach($l as $k => $v) {
         $poids[$k] = $v[3];
      }

      array_multisort($poids, SORT_DESC, $l);

      /*
      function comparaison($a, $b)
      {
          if ($a[3] == $b[3]) {
              return 0;
          }
          elseif ($a[3] < $b[3])
          {
            return(-1);
          }
```

```
            else {
              return(1);
            }
      }

      usort($l, "comparaison");
      */

      $proba_max = $l[0][3];

      $l1 = [];
      $nb = 0;
      foreach ($l as $personnage)
      {
            if($personnage[3] == $proba_max)
            {
                  array_push($l1, $personnage);
                  $nb++;
            }
      }

      // Si l'option est activée
      if($c==1)
      {
            return($l1);
      }
      else {
            return($l);
      }

}

function dichotomix($base_personnages, $base_questions)
{
      global $bdd;
      // Liste des personnages triés selon la plus haute proba [4]
      $l1 = triage($base_personnages,1);
      # Nombre de personnages qui doivent satisfaire un mot clé parfait
      $nb_ref = count($l1)/2;
      # Nombre actuel de personnages qui satisfont le "meilleur mot clé"

      $condition = 1;
      //array_push($base_questions,'vivant');
      while($condition)
      {
                  # Liste des mots clé communs aux personnages ayant la
probabilité MAX (déjà testés)
                  $liste_mots_cle = [];
                  # Nombre actuel de personnages qui satisfont le "meilleur
mot clé"
                  $nb_actuel = 10000;
                  # On récupère tous les mots clé en commun et n'étant pas
déjà dans les questions posées :
                  for($i =0; $i < count($l1)-1;$i++)
                  {

                        foreach ($l1[$i][2] as $mot_cle) {
                              if(!in_array($mot_cle,$liste_mots_cle) &&
!in_array($mot_cle,$base_questions))
                              {
                                    array_push($liste_mots_cle,$mot_cle);
                              }
                        }
                  }
                  # On compte le nombre de fois qu'un mot clé apparaît
                  foreach($liste_mots_cle as $mot_cle)
                  {
                        $nb=0;
                        for ($i=0;$i < count($l1)-1;$i++) {
                              if(in_array($mot_cle,$l1[$i][2]) && $mot_cle
!= ''){
                                    $nb++;
                              }
                        }

                        if($nb == $nb_ref AND $mot_cle != '' AND  $mot_cle
!= '|') {
                              $choix_mot_cle = $mot_cle;
                        }
                        elseif(abs($nb_ref-$nb) <= abs($nb_ref-$nb_actuel)
&& $mot_cle != '' &&  $mot_cle != '|') { #### C H A N G E M E N T ###
                              $nb_actuel = $nb;
                              $choix_mot_cle = $mot_cle;
                        }
                  }

                  // On choisit la question comportant le mot clé :
choix_mot_clé
                  $req = $bdd->query("SELECT ID FROM questions WHERE valeur
= '$choix_mot_cle'");
                  // Sécurité : si la question n'est pas dans la BDD
                  if($req->rowCount() >= 1)
                  {
                        $condition = 0;

                  }
                  else
                  {
                        array_push($base_questions,$choix_mot_cle);
                  }

      }
      # Dès lors, la variable id_question contient l'ID de la question la +
adéquate
      $donnees = $req->fetch();
      return($donnees[0]);

}

function majProbas($l,$v,$c,$n,$p)
{
      $proba1 = explode(',', $p);
      $proba = [];
      foreach($proba1 as $proba2)
```

```
	{
		$proba[$proba1[0]] = $proba1[1];
	}

	foreach($l as $cle => $personnage){
		if(!empty($proba[$personnage['ID']]))
			{
				$proba_perso = $proba[$personnage['ID']];
			}
			else
			{
				$proba_perso = 1;
			}
		if($c == 1) { // Si l'utilisateur à répondu OUI
			$p1 = $proba_perso; // Correspond à l'ID du personnage - si il existe, on le prend en compte !
			$p2 = 1 - $proba_perso;
		}
		else {			// Sinon
			$p1 = 1 - $proba perso;
			$p2 = $proba_perso;
		}

		if (in_array($v,$personnage[2]))
		{
			$l[$cle][3]=(($personnage[3]*($n-1)+$p1)/$n);
		}
		else
		{
			$l[$cle][3]=(($personnage[3]*($n-1)+$p2)/$n);
		}
	}
	return($l);
}
/* ================================================================================
*/


if(isset($_GET['id_page']) && !empty($_GET['id_page']))
{
	$num_question = abs((int) $_GET['id_page']); // Correspond théoriquement à l'ID de la page et au numéro de la question qui va être posée
}
else {
	$num_question = 1;
}

// Si l'utilisateur n'a pas entré de partie
if(empty($_SESSION['partie']) OR !isset($_SESSION['partie']))
{
	$partie = nouvellePartie();
}
else {
	$partie = $_SESSION['partie'];
}
```

```
$base_personnages = $partie[0]; // Le premier élément de $partie correspond à la base des personnages
$base_questions = $partie[1]; // Le second, à la base des questions posées


if(!isset($num_question) or empty($num_question) or $num_question == 0) // Si aucun paramètre n'a été transmis dans l'URL : nouvelle partie
{
	$partie = nouvellePartie();
}
else { // Un paramètre a été transmis
	$num_question = abs((int) $num_question); // On s'assure qu'il s'agit d'un nombre
	/* Sécurité supplémentaire : token : être sûr qu'il ne change pas d'onglet, qu'il ne change pas de navigateur pour tomber sur l'ID exacte
	On doit normalement avoir $num_question = count($partie)+1 */

	// Si l'utilisateur vient de répondre à une question
	if(isset($_POST['token']))
	{
		if(!isset($_POST['non']) && !isset($_POST['oui']))
		{
				// On lui fait peur
		}
		else
		{
			if(isset($_POST['non']))
			{
				$reponse = 0;
			}
			else
			{
				$reponse = 1;
			}

			$req = $bdd->prepare("SELECT id_question FROM utilisateurs_tokens WHERE token = ?");
			$req->execute(array($_POST['token']));

			if($req->rowCount() < 1)
			{
			 // L'utilisateur a tenté de frauder
			}
			else {
				$id_question = $req->fetch();
				$req = $bdd->query("SELECT valeur,personnages FROM questions WHERE ID = '$id_question[0]'");
				$donnees = $req->fetch();
				// On modifie les probabilités
				$partie[0] = majProbas($partie[0],$donnees['valeur'],$reponse,$num_question-1, $donnees['personnages']);
				// On ajoute le nouveau mot clé à base_question
				array_push($partie[1],$donnees['valeur']);
				// On ajoute le mot clé et la réponse à base_reponse
				array_push($partie[2], $reponse);
				// Triages des probabilités
```

```
                        $partie[0] = triage($partie[0]);
                    }
                }
        }
}
// Si l'utilisateur a réinitialisé la partie ou a fait un retour vers une
question précédente
if($num_question != count($partie[1])+1)
{
        if($num_question > count($partie[1])+1)
        {

                $num_question = 1;

                $partie = nouvellePartie(); // Nouvelle partie : l'utilisateur
a entré un faux ID
                $affichage = nouvelleQuestion($partie[0], $partie[1]);
        }
        else // l'ID est inférieur : correspond à un retour à une question
précédente
        {
                // On supprime toutes les questions supérieures
                // $base_questions = array_slice($base_questions, 0,
count($base_questions)-1); : PRENDRE EN COMPTE LES PROBAS QUI EVOLUENT !
                $partie = nouvellePartie(); // Nouvelle partie : l'utilisateur
a entré un faux ID
                $affichage = nouvelleQuestion($partie[0], $partie[1]);
        }
}
else // On a le bon ID de question
{
        $affichage = nouvelleQuestion($partie[0], $partie[1]);
}

// MISE A JOUR DES VARIABLES
$_SESSION['partie'] = $partie;

/* -- P A G E ---------- P A G E ------ A F F I C H A G E ------------ */

include('header.php'); ?>

                <!-- Partie centrale du site -->
                <div class="ui-text container">

                <?php echo $affichage; ?>

                                <a class="ui compact labeled icon button"
href="contact.php"><i class="bell icon"></i>Signaler une erreur</a>
                                <a class="ui compact labeled icon button"
href="index.php"><i class="home icon"></i>Quitter</a>
                </div>
        </div>


<?php include('footer.php'); ?>
```



V2.1 : 7.15 questions par personnage

RETRO : 7.1625 questions par personnage