# Skimind Database Interface

HOW TO CONNECT A DJANGO MODEL

Nathan Bangwa | IkalaWeb | 03 April 2019

# Arborescence

- Skimind
  - checkPoint
  - gFunctions
  - interfacedb
    - config.py
    - modelTables.py
  - learningModel
  - pdfManag
  - useCases

# Description

## `Skimind.interfacedb.modelTables.py`

Ce module contient des classes qui représentent les différentes tables dans la data base ainsi que leurs champs (le contenu de chaque attribut).

- `Class Matchs(object):`

  Cette table contient toutes les informations en rapport avec une liste de pari (liste des matchs).

  - Primary key

| numlist | idmatch | date |
|---|---|---|
| 1017 | 1 | 2019-10-10 |

- `Class Datasets(object):`

  Cette table contient toutes les informations des données avec lesquels le model d'apprentissage s'est déjà entrainé.

  - Primary key

| numlist | idmatch | date |
|---|---|---|
| 1017 | 1 | 2019-10-10 |

- `Class Resultat`

  Cette table contient toutes les informations en rapport avec une liste de resultat (liste des resultats)

  - Primary key

| numlist | idmatch | date |
|---|---|---|
| 1017 | 1 | 2019-10-10 |

- `Class Ticket`

  Cette table contient toutes les informations en rapport avec un ticket qui a été construit.
  - Primary key

| idticket | numlist | idmatch | date |
|---|---|---|---|
| 5 | 1017 | 1 | 2019-10-10 |
| 5 | 1017 | 2 | 1019-10-10 |
| 5 | 1017 | 5 | 2019-10-12 |

- `Class TicketStatus`

  Cette table contient les informations clés en rapport avec un ticket qui a été construit.
  - Primary key

| Idticket |
|---|
| 5 |

- `Class UserTicket`

  Cette table contient toutes les informations en rapport avec le ticket appartenant à un user.
  - Primary key

| uid | idticket |
|---|---|
| Jeanluc243 | 1 |
| msp | 2 |
| skimind | 1 |

## Skimind.interfacedb.config.py

Ce module contient une classe singleton SqlRequestFunctions dont les attributs représentent les différentes fonctions d'accès aux données se trouvant dans la data base. Et c'est cette classe qui fera office d'interface de communication entre la logique skimind et le model Django pour le CRUD.

Ce module contient aussi un objet request_object du type SqlRequestFunctions.

```
@gFunctions.singleton
Class SqlRequestFunctions(object)
     def __init__(self):
          Self.get_training_datas = lambda: list()
          Self.get_data_to_predict = lambda date: list()
          Self.get_prediction_datas = lambda date: list()
          Self.save_prediction = lambda datas: None
          Self.save_datasets = lambda datas: None
          Self.save_ticket = lambda datas: None
          Self.save_user_ticket = lambda datas: None
          Self.save_ticket_status = lambda data: None
          Self.save_matchs = lambda data: None
          Self.save_resultat = lambda data: None


request_object = SqlRequestFunctions()
```

## request_object.get_training_datas

Cet attribut stocke une fonction qui va renvoyer tous les enregistrements de la table MATCHS suivi de tous les enregistrements de la table RESULTAT figurant dans les deux tables et ne figurant pas encore dans la table DATASETS. En clair, on renvoi les données non-apprises par le model d'apprentissage.

MATCHS

| numlist | idmatch | date | MATCHS.... |
|---|---|---|---|
| 1919 | 1 | 2019-10-10 | .... |
| 1919 | 2 | 2019-10-10 | ... |
| 1920 | 1 | 2019-10-11 | ... |

RESULTAT

| numlist | idmatch | date | RESULTAT.... |
|---|---|---|---|
| 1919 | 1 | 2019-10-10 | ... |
| 1919 | 2 | 2019-10-10 | ... |
| 1920 | 1 | 2019-10-11 | ... |

DATASETS

| numlist | idmatch | Date |
|---|---|---|
| 1919 | 1 | 2019-10-10 |
| 1919 | 2 | 2019-10-10 |

RETURN

| numlist | idmatch | date | MATCHS.... | RESULTAT... |
|---|---|---|---|---|
| 1920 | 1 | 2019-10-11 | ... | ... |

## request_object.get_data_to_predict

Cet attribut stocke une fonction qui prend en paramètre une date et qui retourne tous les enregistrements de la table MATCHS suivi de tous les enregistrements de la table RESULTAT (figurant dans les deux tables), mais ne figurant pas dans la table PREDICTION et dont la date est supérieure ou égale à celle passée en paramètre.

MATCHS

| numlist | idmatch | date | MATCHS…. |
|---|---|---|---|
| 1919 | 1 | 2019-10-9 | …. |
| 1919 | 2 | 2019-10-10 | … |
| 1920 | 1 | 2019-10-12 | … |
| 1920 | 2 | 2019-10-13 | … |

RESULTAT

| numlist | idmatch | Date | RESULTAT…. |
|---|---|---|---|
| 1919 | 1 | 2019-10-9 | … |
| 1919 | 2 | 2019-10-10 | … |
| 1920 | 1 | 2019-10-12 | … |
| 1920 | 2 | 2019-10-13 | |

PREDICTION

| numlist | Idmatch | Date |
|---|---|---|
| 1920 | 2 | 2019-10-13 |

RETURN pour paramètre date = 2019-10-10

| numlist | idmatch | Date | MATCHS…. | RESULTAT… |
|---|---|---|---|---|
| 1919 | 2 | 2019-10-10 | … | … |
| 1920 | 1 | 2019-10-12 | … | … |

## request_object.get_prediction_datas

Cet attribut stocke une fonction qui prend en paramètre une date et qui retourne tous les enregistrements de la table MATCHS suivi de tous les enregistrements de la table PREDICTION. (Figurant dans les deux tables). Et dont la date est supérieure ou égale à celle passée en paramètre.

MATCHS

| numlist | idmatch | date | MATCHS…. |
|---|---|---|---|
| 1919 | 1 | 2019-10-9 | …. |
| 1919 | 2 | 2019-10-10 | … |
| 1920 | 1 | 2019-10-12 | … |
| 1920 | 2 | 2019-10-13 | … |

PREDICTION

| numlist | idmatch | Date | PREDICTION …. |
|---|---|---|---|
| 1919 | 1 | 2019-10-9 | … |
| 1919 | 2 | 2019-10-10 | … |
| 1920 | 1 | 2019-10-12 | … |
| 1920 | 2 | 2019-10-13 | |

RETURN pour paramètre date = 2019-10-11

| numlist | idmatch | Date | MATCHS…. | PREDICTION… |
|---|---|---|---|---|
| 1920 | 1 | 2019-10-12 | … | … |
| 1920 | 2 | 2019-10-13 | … | … |

## request_object.save_prediction

Cet attribut stocke une fonction qui prend en paramètre une liste des données et qui stocke cette liste comme enregistrement dans la table PREDICTION

### request_object.save_datasets

Cet attribut stocke une fonction qui prend en paramètre une liste des données et qui stocke cette liste comme enregistrement dans la table PREDICTION

### request_object.save_ticket

Cet attribut stocke une fonction qui prend en paramètre une liste des données et qui stocke cette liste comme enregistrement dans la table TICKET

### request_object.save_user_ticket

Cet attribut stocke une fonction qui prend en paramètre une liste des données et qui stocke cette liste comme enregistrement dans la table USER_TICKET

### request_object.save_ticket_status

Cet attribut stocke une fonction qui prend en paramètre une liste des données et qui stocke cette liste comme enregistrement dans la table TICKET_STATUS

### request_object.save_matchs

Cet attribut stocke une fonction qui prend en paramètre une liste des données et qui stocke cette liste comme enregistrement dans la table MATCHS.

### request_object.save_resultat

Cet attribut stocke une fonction qui prend en paramètre une liste des données et qui stocke cette liste comme enregistrement dans la table RESULTAT.

# Utilisation

Avant de commencer à demander des services à la logique skimind, il faut l'initialiser au préalable. L'initialisation se fait en passant des valeurs (fonctions) aux attributs de l'objet request_object, car ce sont ces fonctions (requêtes) qui sont la base de la logique de fonctionnement de skimind.

# EXEMPLE.

- Skimind
  - djangoQuery.py
  - test.py

# djangoQuery.py

```
import models # c'est juste un exemple

def training_datas(): -> List

def data_to_predict(date: datetime): -> list

def prediction_datas(date: datetime): -> list

def push_prediction(datas: list): -> None

def push_datasets(datas: list): -> None

def push_ticket(datas: list): -> None

def push_user_ticket(datas: list): -> None

def push_ticket_status(datas: list): -> None

def push_match(datas: list): -> None

def push_resultat(datas: list): -> None

 # Les détails de ce que fait chaque fonction ci-haut
 # se touvent juse au-dessus
```

# Test.py

```
from skimind.interfacedb.config import request_object
import djangoQuery as dq

# raccourci
Ski_ro = request_object


# Initialisation des requêtes


Ski_ro.get_training_datas = dq.training_datas
Ski_ro.get_data_to_predict = dq.data_to_predict
Ski_ro.get_prediction_datas = dq.prediction_datas
Ski_ro.save_prediction = dq.push_prediction
Ski_ro.save_datasets = dq.push_datasets
Ski_ro.save_ticket = dq.push_save_ticket
Ski_ro.save_user_ticket = dq.push_user_ticket
Ski_ro.save_ticket_status = dq.push_ticket_status
Ski_ro.save_matchs = dq.push_matchs
Ski_ro.save_resultat = dq.push_resultat
#initialisation effectuée
```