1. [Minor] В конфигурации hostname следовало хранить отдельно от /API

{  
 **"url"**: **"http://www.nbrb.by/API"**}

1. [Minor] Указывай доменное имя вместо localhost:

{  
 **"host"**: **"localhost"**,  
 **"port"**: **"5432"**,  
 **"database"**: **"nbrb"**,  
 **"user"**: **"postgres"**,  
 **"password"**: **"password"**}

1. В целом конфигурации разнес - это несомненно плюс к поддержке решения в будущем
2. [Minor] В resources/db/dump.sql – пусто. Не совсем понятно должно ли там чтото быть. Если нет, то нет смысла иметь эти файлы и папки.
3. [Minor] Не смысла начинать название Json схемы с json\_schema - и так очевидно что это схема, поскольку лежит в директории schema.

json\_schema\_service1\_with\_curID.json

Это касается любых наименований, например, нет смысла класс начинать со слова Class.

1. Структура класса parser некорректная – различная логика сгружена в 1 файл.

**def** parse\_db\_config(),**def** parse\_ssh\_config():

, **def** parse\_api\_config() - должен быть отдельный класс Configuration – который умеет читать данные из любых json конфигураций в твоём проекте (api, ssh, db)

**def** parse\_all\_currencies\_from\_request\_api(json\_response, number) – метод должен находится в классе с апи, от которой приходит этот ответ.

Более того, ты можешь заметить, что список полей повторяется 2 раза:

1. [Critical] Пути должны быть относительные, а не абсолютные

ssh\_config = open(  
 **'/home/ITRANSITION.CORP/e.bondarenko/self-education/rest-api-nbrb-amishchenko-and-e.bondarenko/config/ssh\_config.json'**).read()

1. [Major]Не совсем понятна логика проверки

**if** res.status\_code != HTTP\_STATUS\_OK:  
 **return** res.status\_code, []  
**else**:  
 **return** res.json()

Из документации к методу этого также не понять

Более того эта конструкция дублируется из метода в метод

1. [Critical] отсутствует общий класс для работы с API – класс который есть должен работать через этот отсутствующий класс, а не напрямую с requests library
2. [Major] api\_url следовало вынести в поле класса

Вместо регулярного вызова метода api\_url = JP.parse\_api\_config()

1. [Critical] Слишком общий метод. Не следовало выносить отдельным параметром method. Кроме этого само название method вводит в заблуждение (HTTP метод?). Документация к методу отсутствует.
2. [Critical] Нужно создать класс model и работать с ней - передавать данные через эту модель. Так как мы делали в Steampowered, Cars (game, car).
3. [Major] Параметры следовало передвавать именованными а не через kwargs

**def** generate\_query\_for\_all\_currencies(table\_name, parameters):

**{}')"""**.format(table\_name=table\_name, \*parameters)

Это касается всех запросов.

1. [Major] Следует разделить логику работу по наполнению бд (/support) и логику уже тестовую на 2 файла.
2. [Minor] Не подходящее название

self.log.info(**":::::::::: START LOG ::::::::::"**)

мне кажется мы и так поймем есть LOG или нет

1. [Major] неудачное название

get\_dynamic\_from\_db

1. [Critical] Реализацию тестов следует выполнять в тестах
2. [Critical] Захардкоженные значения

**${exists\_cur\_value}** *1*

**${start\_date}** *2016-3-1* **${end\_date}** *2017-3-30*

Всё должно получаться динамически из БД

testNbrb.py – это по сути Steps.

1. [Major] нужно вынести форматы в константы

**if** values\_from\_db[name] != datetime.strptime(values\_from\_api[name], **'%Y-%m-%dT%H:%M:%S'**):  
 **print** values\_from\_db[name], datetime.strptime(values\_from\_api[name], **'%Y-%m-%dT%H:%M:%S'**)

1. [Major] Лишняя абстракция

@staticmethod  
**def** get\_line\_with\_error():  
 *"""  
 Get line with text 'ERROR'* **:return***:  
 """* **return** Ssh.get\_line\_with\_text(**'ERROR'**)

1. Используй Capitalize 