Одно из крупнейших НИИ занимается разработкой функции, которая производит вычисления для массива пар простых чисел a и b типа Int. У них возникла потребность в периодическом тестировании этой функции на различных наборах данных. Постепенно они накапливают массивы входных данных, на которых хотят тестировать свою функцию. Цель тестирования - ручная проверка обработанных данных в интерфейсе и проверка отсутствия эксепшенов на всех наборах данных.

Функция принимает и возращает JSON.

Формат входного набора данных - масив из словарей чисел а и b

[

{

"a": 1,

"b": 2

},

]

Формат выходных данных - словарь с результатом работы функции:

{ ‘result’: 0 }

Таким образом, необходимо:

- принимать наборы данных через интерфейс

- сохранять наборы данных в БД

- запускать обработку всех наборов данных нажатием кнопки в интерфейсе

- если в процессе обработки возникает эксепшен - нужно сохранять его в БД

- сохранять в БД результат для каждого набора данных

- показывать общий статус последней проверки функции: если были эксепшены - True, иначе - False

- показывать последние результаты обработки для кажого набора данных

- показывать список эксепшенов

Требования к инструментарию:

1. Приложение должно быть реализовано на языке python с использованием фрэймворка django. Все зависимости должны быть установлены с помощью pip, в корне проекта хранить requirements

2. Для работы с БД использовать django-orm, бэкенд ORM - Postgres

3. Для организации работы фоновых задач использовать RabbitMQ и celery.

4. Для мониторинга фоновых задач использовать celery-flower

Требования к реализации:

1. Обработка одного набора данных должна производиться в цепочке из трех задач:

- вычитывание из БД и формирование JSON запроса

- вызов функции обработки JSON запроса и формирование JSON ответа

- сохранение JSON ответа в БД

2. Каждая задача цепочки должна выполняться в отдельном воркере, это позволит размещать их на разных серверах. Через интерфейс celery flower можно управлять количеством параллельных процессов в каждом воркере, это позволит равномерно распределять нагрузку.

3. Тестируемая функция должна быть задачей celery (т.е. быть второй задачей в цепочке из п.1)

Пример тестируемой функции:

def test\_func(json):

import json

d = json.loads(json)

return {‘result’: d[‘a’] + d[‘b’]}

Приложение разрабатывать в git репозитории