



# Начинающий Python разработчик

Дэшборд моделирования функций.

## Алитикс, 2021 г.

### Задание

Необходимо разработать приложение, которое позволит пользователю моделировать работу функций  $y=f(t)$  и анализировать соответствующие графики.

Пример функции:  $y=t*t + 2/t$ , где

$t$  - unixtime в интервале  $[\text{datetime.now()} - \text{interval}, \text{datetime.now()}]$  с шагом  $dt$ ,

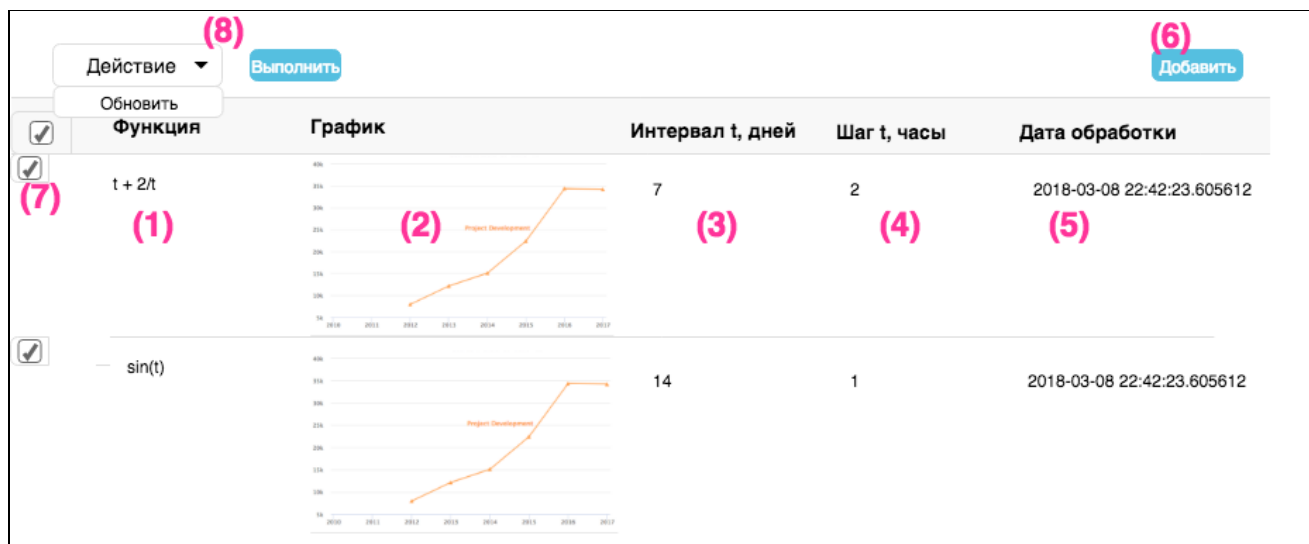
**Interval** - глубина периода моделирования в днях



**datetime.now()** - дата обработки

**dt** - шаг в часах

### User story

1. Пользователь заходит в админку, видит список ранее сохраненных функций (1), соответствующие графики (2), дату последней обработки (5), параметры (3) и (4), а также кнопку добавить (6).



Функция	График	Интервал t, дней	Шаг t, часы	Дата обработки
$t + 2/t$		7	2	2018-03-08 22:42:23.605612
$\sin(t)$		14	1	2018-03-08 22:42:23.605612

\* графики на скриншоте представлены для примера и не отражают представленные функции

2. Пользователь жмет кнопку добавить (6), вводит функцию (1) в текстовом виде, а также значение  $dt$  и  $interval$ . Введенная информация сохраняется в БД. Записи в БД уникальны только по ID.
3. Приложение генерирует изображение для введенной функции и сохраняет результат в БД. Если возникает ошибка - то сохраняется текст исключения (он выводится вместо графика). После обработки обновляется поле дата обработки (5).

4. По завершению обработки вновь открывается список (1), в которой появляется строка с только что введенной функцией, пользователь видит в строке с функцией график или текст исключения, введенные параметры и дату обновления.

## System story

Приложение состоит из одного django приложения. Сценарий при добавлении новой функции в админке:

1. Админка сохраняет инфу о функции в БД и запускает фоновую операцию генерации изображения.
2. Запуск операции происходит через асинхронный вызов [операции celery с ожиданием результата](#).
3. Операция выполняется в фоновой задаче celery.task в отдельном процессе celery worker.
4. Операция принимает в аргументах идентификатор строки с инфой о функции из БД.
5. Операция выбирает инфу о функции из БД по переданному айдишнику, генерирует точки для графика по выбранной инфе.
6. Операция генерирует изображение по точкам и сохраняет в БД, актуализирует дату обновления.
7. Админка дожидается генерации изображения и редиректит пользователя на список функций, в которой отображается график с строке с только что добавленной функции.

## Требования к реализации

1. Приложение разместить в одном репозитории github или bitbucket. Название репозитория должно состоять из рандомного хэша.
2. Для БД использовать Postgresql.
3. Для реализации приложения использовать django.
4. Для фоновых задач использовать celery, шину сообщений RabbitMQ, бэкенд результатов Redis.
5. Графики можно генерить самостоятельно, например через matplotlib.

## Желательно

6. Будет плюсом, если для развертывания приложения и необходимых инфраструктурных сервисов вы воспользуетесь docker-compose.
7. В этом случае для графиков лучше использовать возможности готового образа [highcharts](#).

## Работа над заданием

- Перед стартом работ сделать оценку по времени и срокам выполнения и отправить оценку письмом на адреса [db@alytics-team.com](mailto:db@alytics-team.com), [mikhailarev@alytics.ru](mailto:mikhailarev@alytics.ru)
- В теме письма необходимо указать название вакансии, а в письме свои данные и оценку по срокам выполнения задания
- прислать ссылку на электронную почту на адреса [db@alytics-team.com](mailto:db@alytics-team.com), [mikhailarev@alytics.ru](mailto:mikhailarev@alytics.ru)

## Если возникают вопросы

- По всем вопросам пишите в цепочку с подтверждением задания [db@alytics-team.com](mailto:db@alytics-team.com), [mikhailarev@alytics.ru](mailto:mikhailarev@alytics.ru)