Требуется разработать микросервис для Службы международной доставки. Сервис должен получать данные о посылках и рассчитывать им стоимость доставки.

Сервис должен содержать роуты:

- **Зарегистрировать посылку**. Информация о посылке должна содержать: название, вес, тип, стоимость содержимого в долларах. Входные данные должны быть в формате json. При получении нужно их валидировать. Если валидация прошла успешно — выдать индивидуальный id посылки, доступный только пользователю, подавшему запрос на регистрацию посылки, в рамках его сессии.
- **Получить все типы посылок и id типов.** Посылки бывают 3х типов: одежда, электроника, разное. Типы должны храниться в отдельной таблице в базе данных.
- **Получить список своих посылок со всеми полями, включая *имя* типа посылки и стоимость доставки (если она уже рассчитана)**. Также должна быть пагинация и возможность фильтровать посылки по типу и факту наличия рассчитанной стоимости доставки.
- **Получить данные о посылке по ее id.** Данные включают: название, вес, тип посылки, ее стоимость, стоимость доставки.

Периодические задачи:

- Раз в 5 минут выставлять всем необработанным посылкам стоимости доставки в рублях.
  Стоимость доставки вычисляется по формуле:
  Стоимость = (вес в кг * 0.5 + стоимость содержимого в долларах * 0.01 ) * курс доллара к рублю
- Курс доллара к рублю брать тут https://www.cbr-xml-daily.ru/daily_json.js и кешировать в redis.

Примечание:

- Если стоимость доставки еще не рассчитана - выводить “Не рассчитано”.
- Добавить возможность запуска периодических задачи вне расписания для нужд отладки.

Обязательные требования:

- Приложение НЕ содержит авторизации.
- Приложение отслеживает пользователей по сессии, т.е. у каждого пользователя свой список посылок.
- Данные хранятся в MySQL.
- Реализация на выбор: на Django или на FastAPI (с валидацией через pydantic).
- Описание API через Swagger.
- Докеризация результата (docker-compose up для запуска сервиса и всех зависимостей).
- Особое внимание нужно уделить:

    - Стандартизации ошибок и ответов.
    - Логированию.
    - Кэшированию.
    - Обработке ошибок.
    - Чистоте и понятности кода.
    - Следованию общепринятым стандартам программирования на python.
    - Документированию основных методов.

Плюсом будет:

- В случае выбора Django - реализация через DjangoRestFramework.
- Использование асинхронности (если используем FastAPI).
- Покрытие API тестами.
- Веб-интерфейс к приложению.
- Выполнение одного из дополнительных заданий:
    - Предположим, у нас в день бывает миллион посылок. Есть техническое условие - хранить лог расчётов стоимостей доставок в MongoDB. Предложите и реализуйте способ подсчета суммы стоимостей доставок всех посылок по типам за день, используя информацию из этого лога.
    - Реализовать регистрацию посылок используя RabbitMQ. Посылки из роута регистрации попадают не напрямую в БД, а через брокер сообщений и воркеры, которые попутно сразу же рассчитают стоимость доставки. Периодическая задача расчета стоимость доставки больше не нужна.
    - Создать роут привязки посылки к транспортной компании (добавить id компании - любое положительное число - к записи посылки), которая ее будет доставлять. Одну посылку могут попытаться привязать разные компании, поэтому необходимо обеспечить гарантию, что 1-й обратившаяся компания закрепит посылку за собой. У нас высоконагруженная система - запросы могут прилетать несколько раз в секунду.

Если в процессе выполнения задания возникли какие-либо затруднения в решениях, то необходимо задокументировать обоснование этих решений, либо указать это в сопроводительном письме при передаче решения тестового задания.

Ориентировочное время выполнения - 10 часов. Максимальное - 16 часов. Тестовое задание необходимо предоставить на почту hr@ra-delta.ru в виде ссылки на открытый репозиторий. Заголовок письма должен иметь вид "Тестовое Python разработчика: Имя Фамилия". Срок сдачи обсуждается индивидуально с HR.