Backend Developer | Test Task

Описание задания

Необходимо разработать небольшой REST-сервис для управления заявками на выплату средств. Каждая заявка создаётся через API и должна обрабатываться асинхронно с использованием Celery. Задание небольшое по объёму, но важно продемонстрировать аккуратный, чистый и поддерживаемый код.

Технологический стек

Обязательные технологии:

- Python 3.10+
- Django 4.2+
- DRF
- Celery + Redis или RabbitMQ
- PostgreSQL

Разрешено использовать любые дополнительные инструменты и современные практики проектирования, менеджеры пакетов, типизацию, слои и т.д.

Требования к реализации

1. Модель данных

Требуется реализовать сущность «Заявка на выплату» с минимальным набором полей, например:

- идентификатор;
- сумма выплаты;
- валюта;
- реквизиты получателя;
- статус заявки;
- дата создания;
- дата обновления;
- опциональное описание или комментарий.

Конкретные типы полей, внутренние ограничения и правила — на ваше усмотрение. Важно, чтобы модель была реалистичной для задачи выплаты средств.

---

2. REST API

Необходимо реализовать следующие эндпоинты:

- GET /api/payouts/ — список заявок
- GET /api/payouts/{id}/ — получение заявки по идентификатору
- POST /api/payouts/ — создание новой заявки
- PATCH /api/payouts/{id}/ — частичное обновление заявки (прежде всего — статуса)
- DELETE /api/payouts/{id}/ — удаление заявки

Формат ошибок и ответов должен быть корректным и предсказуемым.

---

3. Асинхронная обработка Celery

При создании заявки должна запускаться Celery-задача, которая:

- принимает идентификатор созданной заявки;
- имитирует обработку заявки (например, задержка, логирование, простая проверка);
- изменяет статус заявки после обработки.

Цель — продемонстрировать корректную интеграцию Celery и понимание асинхронных процессов.

---

4. Валидация

При создании заявки необходимо проверить:

- обязательные поля;
- корректность значений;
- положительность и тип суммы;
- длину или формат строковых полей (где это имеет смысл).

Способ реализации валидации — на ваше усмотрение.

---

5. Тесты

Минимальный набор тестов:

- тест успешного создания заявки;
- тест проверки вызова Celery-задачи при создании (например, через mock).

Остальные тесты — по вашему усмотрению.

Оценка результата

Качество кода

- структура проекта;
- чистота и читаемость;
- грамотная работа с Django ORM;
- отсутствие дублирования;
- аккуратность архитектуры и разбиения на слои;
- корректная работа API и обработка ошибок.

Работа Celery

- корректная настройка;
- связность API и фоновых задач;
- надёжность выполнения задачи.

Минимальный функционал

- работоспособность REST API;
- базовая валидация;
- наличие тестов.

Мы обращаем внимание на зрелость инженерного подхода и современные практики разработки.

Считайте логику работы очень важными и опастныи. Тк этот финтех. Хочется увидеть максимально надёжный подход в элегантном исполнении.

Makefile и инструкция по запуску

В репозитории необходимо предоставить:

- README.md с краткой инструкцией по запуску проекта:
    - установка зависимостей;
    - запуск приложения;
    - запуск миграций;
    - запуск Celery worker;
    - запуск тестов.
- Makefile с базовыми командами:
    - запуск сервиса;
    - запуск тестов;
    - запуск worker;
    - миграции;
    - дополнительные команды — по вашему усмотрению.

Описание деплоя

В README.md нужно кратко описать:

- как вы представляете деплой проекта в прод;
- какие сервисы необходимы;
- как бы вы запускали Django и Celery в реальной системе;
- минимальные шаги по подготовке окружения.

Достаточно текстового описания без детальной инфраструктуры.

Дополнительные возможности (необязательно, но повысит оценку)

- Dockerfile и docker-compose (web, база, брокер, worker);
- CI (GitHub Actions / GitLab CI) с тестами и линтерами;
- использование современных пакетных менеджеров (uv, poetry и т.д.);
- линтеры и форматтеры;
- документация API (Swagger / Redoc);
- скриншоты работы API (Postman, Swagger UI);
- минимальная типизация.

Эти пункты не обязательны, но помогут лучше понять ваш инженерный уровень.

Что предоставить

- Ссылку на репозиторий GitHub/GitLab;
- Краткую инструкцию по запуску;
- Краткое описание деплоя;
- (Необязательно) скриншоты работы API.

Рекомендуемое время выполнения: около 1.5 часа(обязательной части). Допускается сделать больше, если хотите показать качество и подход(что только положительно оценивается, чтобы выделиться среди большого кол-ва кандидатов).
Можете использовать ваши любимые boilerplate проектов. Чтобы оценить ваш архитектурный подход.

Примечание: если проект будет готов к продакшн среде - будет большой плюс. Но лучше описать от и до как запустить проект на сервере (на сколько хорошо вы это умеете).

Желаем удачи! Покажите “Do your best”!!!