Техническое задание (предсобеседовательное) Вакансия: Senior Fullstack-разработчик (Growth-команда / MVP) Компания: Posiflora

---

1. Цель задания

Проверить на практике способность кандидата быстро запускать MVP-фичи в Growth-формате: - сквозное fullstack-мышление (UI → API → DB → внешняя интеграция), - аккуратность в данных и идемпотентности, - умение работать с внешними API, - минимальные DevOps-навыки (упаковка в Docker).

Оцениваются скорость, корректность бизнес-логики и качество инженерных решений.

---

2. Ожидаемое время

До 2–4 часов.

---

3. Технологии (на выбор кандидата)

- **Frontend:** React + TypeScript (желательно).
- **Backend:** PHP (Symfony-style) **или** Go.
- **DB:** PostgreSQL / MySQL / SQLite (допустимо упрощение).
- **Инфраструктура:** Docker (желательно).

---

4. Контекст

Мы делаем MVP интеграции с Telegram для магазинов Posiflora.

Гипотеза Growth:

> «Если магазин сможет быстро подключить Telegram-бота и получать уведомления о новых заказах — это повысит ежедневную активность и удержание владельцев».

Нужно собрать минимально рабочую версию: - магазин подключает Telegram-бота (вводит токен и chat_id), - система отправляет уведомления о новых заказах, - есть панель статуса интеграции.

---

5. Данные / модель

Минимальная схема данных (можно использовать SQL или in-memory, но с понятными интерфейсами):

```
shops(id, name)

telegram_integrations(
  id, shop_id,
  bot_token text,
  chat_id text,
  enabled boolean,
  created_at, updated_at,
  unique(shop_id)
)

orders(
  id, shop_id,
  number text,
  total numeric,
  customer_name text,
  created_at timestamp
)

telegram_send_log(
  id, shop_id, order_id,
  message text,
  status enum('SENT','FAILED'),
  error text,
  sent_at timestamp,
  unique(shop_id, order_id) -- ключ идемпотентности
)
```

Сид: минимум 1 магазин, 5–10 тестовых заказов.

---

6. Backend API

6.1. Подключение Telegram-интеграции

POST /shops/{shopId}/telegram/connect

Payload:

```
{
  "botToken": "123456:ABC-DEF...",
  "chatId": "987654321",
  "enabled": true
}
```

Правила: - upsert по shopId (если интеграция уже есть — обновить); - валидировать, что `botToken` и `chatId` не пустые; - обновлять `updated_at`.

Ответ: сохранённая конфигурация интеграции.

6.2. Создание заказа (эмуляция)

POST /shops/{shopId}/orders

Payload:

```
{
  "number": "A-1005",
  "total": 2490,
  "customerName": "Анна"
}
```

Логика: 1. Создать заказ в таблице `orders`. 2. Если Telegram-интеграция включена — отправить уведомление в Telegram: - текст: `Новый заказ {number} на сумму {total} ₽, клиент {customerName}` 3. **Идемпотентность отправки:** - если для пары `(shopId, orderId)` уже есть запись в `telegram_send_log` — повторно не отправлять. 4. Логировать попытку отправки (SENT/FAILED) в `telegram_send_log`.

Ответ: заказ + статус отправки (`sent` / `failed` / `skipped`).

6.3. Статус интеграции

GET /shops/{shopId}/telegram/status

Возвращает: - `enabled` - `chatId` (допустимо частичное маскирование) - `lastSentAt` - `sentCount` за 7 дней - `failedCount` за 7 дней

---

7. Telegram отправка

Использовать Telegram Bot API:

`POST https://api.telegram.org/bot{token}/sendMessage`

body:

```
{ "chat_id": "...", "text": "..." }
```

Если кандидат не хочет дергать реальный Telegram — допускается мок-режим, но: - должен быть интерфейс `TelegramClient`; - в README описать, как включить реальную отправку.

---

8. Frontend (1 экран)

Страница: **/shops/:shopId/growth/telegram**

Содержит: - поле `botToken` - поле `chatId` - тумблер `enabled` - кнопка **«Сохранить»** (POST /telegram/connect) - блок статуса (GET /telegram/status): enabled, lastSentAt, sent/failed за 7 дней - краткая подсказка «как узнать chatId» (текстом)

UI может быть максимально простым, дизайн не оценивается.

---

9. Нефункциональные требования

- Разделение слоёв: сервис/хендлер → репозиторий → `TelegramClient`.
- Идемпотентность обеспечивается уникальным ключом или проверкой репозитория.
- Ошибки Telegram **не должны** ронять создание заказа: заказ создаётся, ошибка логируется.

---

10. Мини-инфраструктура

Обязательное минимальное оформление: - `Dockerfile` для backend - `docker-compose.yml` для backend + db

(или Dockerfile + инструкция запуска без compose)

Цель — показать, как быстро упаковывается MVP.

---

11. Тесты

Минимум 3 backend-теста: 1. При создании заказа и включённой интеграции вызывается `TelegramClient` и пишется лог **SENT**. 2. Повторная отправка/повторное создание заказа не создаёт дублей `telegram_send_log` и не шлёт сообщение повторно (идемпотентность). 3. При ошибке `TelegramClient` пишется лог **FAILED**, а заказ всё равно создаётся.

---

12. Что нужно прислать в результате

1. Ссылка на репозиторий (GitHub/GitLab) или архив.
2. README:
3. как запустить backend (docker/без docker)
4. как запустить frontend
5. как сидятся тестовые данные
6. как прогнать тесты
7. реальная Telegram-отправка или мок-режим
8. список допущений/упрощений

---

13. Как будем разбирать на собеседовании

На интервью (30–40 минут) обсудим: - где проведены границы MVP; - как обеспечена идемпотентность; - как бы перевели отправку в async (очередь/ретраи/cron); - безопасность токенов (env, шифрование, маскирование); - что бы добавили для мониторинга интеграции.

Удачи! Ждём ваше решение и короткое описание допущений.