AOzonJobTracker

Фича: Telegram-уведомления в личные сообщения

Техническое задание и гайд по внедрению (Android / POCO). Дата: 2026-02-07

Что делаем	Телефон при событии «появился слот» отправляет сообщение в ваш личный чат Telegram через Bot API.
Сервер нужен?	Нет. Достаточно интернета на телефоне (HTTPS-запросы в Telegram Bot API).
Куда слать	Личный чат (chat_id = 5494039093).
Токен бота	Использовать ваш token (в документе отображён маской: 7997058910…H-aBxg).
Когда слать	По изменениям состояния (dedupe), плюс опционально алерт на ошибки/паузы/восстановление.

Важно: в личку бот сможет писать только после того, как вы ему уже написали (обычно достаточно нажать Start / отправить любое сообщение).

1) Цель и границы

• Добавить канал оповещений через Telegram (личные сообщения).
• Не меняем логику мониторинга слотов: Telegram-уведомления реагируют на события, которые уже формируются трекером.
• Без сервера: все отправки делаются прямо с Android-девайса через HTTPS.
• Фича должна быть устойчивой к отсутствию интернета, 429 rate-limit и временным сбоям.

Не делаем: управление ботом через входящие команды Telegram, вебхуки, кнопки, меню. Только отправка событий (outgoing).

2) Архитектура интеграции в существующий проект

Встраиваем уведомления как отдельный компонент, чтобы он не зависел от конкретного экрана Ozon Job и не ломал твой «start from any screen» роутер.
• TelegramNotifier: единая точка отправки сообщений (API-клиент + ретраи + троттлинг).
• TelegramSettings: хранение chat_id/token/включено-ли/тихий режим/какие события слать.
• NotificationPolicy: правила дедупликации (не спамить одинаковым) + ограничения по частоте.
• EventSink: слой, который принимает события мониторинга (slot_found, slot_gone, error, recovery) и решает, надо ли слать в телегу.

Рекомендуемая связь:
SlotWatcher / StateMachine → EventBus → EventSink → TelegramNotifier

3) Данные и настройки

Минимальные настройки для лички:
• chat_id: 5494039093.
• bot_token: хранить строкой (в настройках приложения).
• enabled: тумблер «слать в Telegram».
• quiet_mode: «тихо» (disable_notification=true).
• events_mask: SLOT_FOUND / SLOT_GONE / ERROR / HEARTBEAT / RECOVERY.

UI-минимум: экран Telegram + кнопка «Отправить тестовое сообщение» + поле «показать последнюю ошибку Telegram».

3.1) Если chat_id потеряется: как восстановить

Простой путь без сервера:
• Открой чат с ботом в Telegram и отправь любое сообщение (или /start).
• С телефона сделай запрос Bot API методом getUpdates (long polling не нужен — можно один раз).
• В ответе будет объект message.chat.id — это и есть нужный chat_id для личного чата.

Документация: https://core.telegram.org/bots/api#getupdates

4) Telegram Bot API: как отправлять сообщение

Отправка — это запрос методом sendMessage. Поддерживаются GET/POST и передача параметров как query string, form-url-encoded или JSON.

Endpoint:

```
https://api.telegram.org/bot/sendMessage
```

Минимальный payload:

```
{ "chat_id": 5494039093, "text": "Слот появился: Петровское / Производство непрофиль"
}
```

Ключевые параметры sendMessage: chat_id, text, parse_mode (опционально), disable_notification (опционально).

Лучше всего: первый релиз — plain text без parse_mode (меньше граблей с экранированием).
Если вы заблокировали бота, Telegram вернёт 403 — в этом случае показывай ошибку и прекращай отправку, пока не разблокируешь.

5) Ограничения Telegram и защита от 429

Чтобы Telegram не начал возвращать «429 Too Many Requests», держи ограничения:
• В одном чате старайся не слать больше 1 сообщения в секунду (возможны короткие всплески, но затем 429).
• Сообщения слать по событию/изменению состояния, а не «каждую минуту одно и то же».
• Если пришёл 429 — делать backoff и повторять отправку позже (по возможности учитывать retry_after).

Троттлинг и дедуп обязательны: «ошибка сети → ретрай → дубликаты» быстро превращается в спам.

6) Дедупликация: что считать «новым слотом»

Нужно отличать «просто снова проверили» от «реально изменилось». Поэтому фиксируем последнюю картину слотов.
• Храни в Room last_seen_signature по ключу (warehouse=Петровское, job=Производство непрофиль).
• Сигнатура = отсортированный список дат/смен/окон (что уже умеешь вытащить из UI) + хеш строки.
• Отправка: только если signature изменилась (или если включён heartbeat и прошло N часов).
• Cooldown на одинаковые ERROR-алерты (например, не чаще 1 раза в 30 минут).

7) Шаблоны сообщений (чтобы было удобно читать)

Примеры читаемого plain text:

```
[OK] Слот появился Склад: Петровское Работа: Производство непрофиль Даты/окна: 10.02, 11.02 (пример) Время: 02:48 Источник: Ozon Job

[WARN] Мониторинг в паузе Причина: нет интернета / Ozon Job закрыто / не распознан экран Действие: recovery-шаг X Следующая попытка: через 2 минуты
```

8) Android-реализация: сеть и клиент

На Android достаточно обычного HTTP-клиента. Практические варианты:
• OkHttp (минимум магии).
• Retrofit поверх OkHttp (если уже используешь).
• Ktor client (если проект на Kotlin и хочется единый клиент).

Обязательное в манифесте: permission INTERNET.
Таймауты: connect 10–15s, read 15–20s, write 15–20s.
Потоки: отправка — из background dispatcher/coroutine, чтобы не блокировать main и не тормозить Accessibility-цикл.

8.1) Очередь сообщений (устойчивость)

Если хочешь «железобетон»: делай outbox-очередь.
• Сохраняй каждое сообщение в Room со статусом PENDING.
• Отдельный воркер/корутина пробует отправить; при успехе помечает SENT, при провале — увеличивает попытки и ставит next_retry_at.
• Плюс правило: не слать очередь, пока стоит блок на 429 (до времени unblock_at).

9) Работа в фоне: что важно на Android 12+ / 14+

Твоя основная логика мониторинга уже решает вопрос «как жить 24/7» (обычно Foreground Service или WorkManager). Telegram-отправка должна быть совместима с этим режимом.
• Если мониторинг идёт через Foreground Service — Telegram-запросы делай из того же процесса/сервиса.
• Если мониторинг иногда запускается задачей — отправку делай без UI, синхронно к событию.
• При отсутствии сети: очередь + один алерт «сеть вернулась» по восстановлению.

Если используешь Foreground Service: на новых Android важно корректно декларировать сервис/типы FGS и разрешения (иначе система может агрессивно ограничивать).

10) Обработка ошибок и «самовосстановление»

Сценарии и ожидаемое поведение:
• Нет интернета: статус OFFLINE, редкие проверки, один алерт при восстановлении.
• 401/403 от Telegram: токен неверный/отозван или бот заблокирован пользователем. Показать ошибку и выключить отправку до исправления.
• 429: backoff/пауза, учитывать retry_after если есть, не слать новые сообщения до конца паузы.
• Timeout/5xx: ретраи с экспоненциальной задержкой (например: 5s → 15s → 45s → 2m, с пределом).

11) Критерии готовности (acceptance)

• Кнопка «Тест» отправляет сообщение в личку (при верных token/chat_id).
• При появлении нового слота (изменение signature) приходит уведомление ≤ 1 цикл проверки.
• При повторных проверках без изменений сообщений нет (dedupe работает).
• При 429/таймаутах приложение не падает, а делает ретраи и не спамит.
• Логи: видно попытку отправки, результат, код/ошибку, причину дедуп-отмены и текущую паузу из-за 429.

12) Справка и источники

• Telegram Bot API Manual (sendMessage, getUpdates, форматы параметров): https://core.telegram.org/bots/api
• Telegram Bots FAQ (rate limits и 429 рекомендации): https://core.telegram.org/bots/faq
• Android Foreground Services (декларация/типы/разрешения): https://developer.android.com/develop/background-work/services/fgs/declare