Ай-ТИ Фреш

ТЕХНИЧЕСКИЙ РАЗБОР

AI-маршрутизация курьеров: ACO + Яндекс Карты

Матрица расстояний, кэш геокодера 30 суток, свой солвер и выгрузка из 1С УТ 11.5

Июль 2026

itfresh.ru · ИТ-аутсорсинг для юридических лиц

О ЧЁМ ЭТОТ РАЗБОР

Суть проблемы

Диспетчер раскидывает 30-40 заказов на 4 курьеров по карте и памяти: окна доставки срываются, пробег и бензин растут, а решением выглядит наём пятого курьера. Мы собираем маршрутизатор на матрице времени Яндекса и собственном ACO-солвере, забирая заявки из 1С УТ через HTTP-сервис. Оптимизируем не километры, а время в пути с учётом пробок: клиенту важно «во сколько привезут».

Почему это важно бизнесу

- Ручная раскидка не масштабируется: рост заказов на 20% закрывается наймом курьера, а не порядком обхода точек
- Сорванное окно доставки клиент запоминает лучше самой доставки — повторные заказы уходят туда, где есть точный ETA
- Бензин и переработки — прямые деньги: 10-15% лишнего пробега в месяц сопоставимы с половиной ставки курьера
- Знание маршрутов живёт в головах двух опытных курьеров: отпуск или увольнение сразу роняет качество доставки
- Без цифр по времени в пути руководитель не планирует смены и не спорит с подрядчиком по ставке за доставку

ЧТО МЫ ЗАКЛАДЫВАЕМ

Ключевые параметры реализации

100

элементов — предел синхронной матрицы за один запрос: 10×10 точек, не больше

документация API Матрицы расстояний

40 RPS

потолок синхронного режима; асинхронный даёт до 25 млн элементов, но только 5 RPS

документация API Матрицы расстояний

30 суток

разрешённый срок кэширования ответов геокодера — ровно столько держим TTL в Redis

условия использования API Геокодера

1000/сут

лимит HTTP-геокодера на бесплатном ключе при общей квоте 25 000 запросов в сутки

условия использования API Геокодера

9.15.6755

версия ortools в pip: держим как офлайн-эталон качества, боевой контур на нём не строим

PyPI, актуальный релиз

60 мин

интервал пересчёта матрицы пробок: чаще выигрыша не даёт, а счёт за API растёт линейно

наш стандарт

РЕАЛИЗАЦИЯ 1 ИЗ 3

Выгрузка заявок из 1С и геокодирование адресов

Что настраиваем

Торговая компания 28 РМ: 1С УТ 11.5 на платформе 8.3.27, PostgreSQL, отдельная VM маршрутизатора в той же подсети

Как мы это делаем

1. В расширении конфигурации поднимаем HTTP-сервис: /<база>/hs/routes/orders?date=YYYY-MM-DD отдаёт JSON — номер, адрес, вес, окно доставки, приоритет
2. Где расширение ставить нельзя, берём штатный REST: /odata/standard.odata/...?$format=json, состав задаём УстановитьСоставСтандартногоИнтерфейсаOData()
3. Адреса геокодируем через https://geocode-maps.yandex.ru/v1/ с apikey, ответ кладём в Redis по ключу geo:sha1(адрес) с TTL 30 суток — ровно по условиям API
4. Ответы с precision ниже street уходят диспетчеру на ручную модерацию: иначе курьер приедет в геометрический центр квартала
5. Координаты держим в реквизитах контрагента 1С с датой актуальности и обновляем фоновым заданием раз в 30 суток — как требуют условия API

РЕЗУЛЬТАТ

Утренняя выгрузка 38 заявок отрабатывает за секунды, геокодер расходуется на 5-10 новых адресов в день вместо сотни, суточный лимит 1000 запросов не приближается вплотную. Адрес правится один раз и в одном месте — в 1С, а не в трёх системах сразу.

КЛЮЧЕВОЙ НЮАНС

Кэшировать геокодер дольше 30 суток запрещено условиями API, поэтому TTL выставляем ровно и обновляем координаты фоновым заданием, а не при утреннем расчёте — иначе сбой геокодера тормозит выезд склада.

РЕАЛИЗАЦИЯ 2 ИЗ 3

Матрица времени: нарезка, кэш, деградация

Что настраиваем

Расчёт на 38-45 точек и 4 машины: сервис matrix-builder на Python 3.12, requests + бэкофф, Redis 7 под кэш матриц

Как мы это делаем

1. Запрос к https://api.routing.yandex.net/v2/distancematrix с параметрами apikey, origins, destinations, mode=driving и departure_time в unix для прогноза пробок
2. Матрица 40×40 = 1600 элементов в синхронный лимит 100 не влезает: режем на 16 блоков 10×10, собираем numpy-массив и держимся ниже потолка 40 RPS
3. Читаем rows[].elements[].status: FAIL по отдельной паре расчёт не роняет — ячейку добиваем оценкой по прямой с городским коэффициентом и помечаем флагом
4. 429 и 5xx ретраим с экспоненциальным бэкоффом 1-2-4-8 с и джиттером; на 401 сразу алерт в Telegram — ключ отозван либо ещё не активирован (до 15 минут)
5. Готовую матрицу кладём в Redis на 60 минут; для фургонов считаем mode=truck с weight, height и length, иначе маршрут уводит под низкий мост

РЕЗУЛЬТАТ

Вместо тысяч вызовов в сутки укладываемся в 12-14 пересчётов, счёт за API становится предсказуемой строкой бюджета, а сбой отдельной ячейки или короткая недоступность сервиса не срывают утреннюю раскладку заказов.

КЛЮЧЕВОЙ НЮАНС

В v2/distancematrix нет параметра waypoints — только origins и destinations, до 100 точек в каждом и не более 100 элементов в матрице. Превышение и опечатки прилетают как HTTP 400 с текстом в массиве errors.

РЕАЛИЗАЦИЯ 3 ИЗ 3

ACO-солвер, ограничения и маршрутный лист

Что настраиваем

Ядро в /opt/courier-routes на Python 3.12 в venv, numpy, systemd unit + timer, бот курьера на python-telegram-bot 22.x

Как мы это делаем

1. Профиль солвера: 40 муравьёв, 120 итераций, alpha 1.0, beta 2.5, испарение 0.5, подкрепление феромона по 5 лучшим маршрутам итерации
2. Ограничения зашиваем штрафами в целевую функцию: вместимость машины, окно доставки клиента, сервисное время 120 с на точку, приоритет VIP-заказов
3. Еженедельно сверяемся с OR-Tools 9.15 (PATH_CHEAPEST_ARC + GUIDED_LOCAL_SEARCH, time_limit.seconds 30) на архиве дней — это офлайн-эталон
4. Запуск в 06:30 и пересчёт в 12:30 системным таймером; курьер получает порядок точек и ссылку на следующую точку в Яндекс Навигаторе
5. Солвер экспортирует в Prometheus время расчёта, число нераспределённых заказов и ошибки API; недельная сводка уходит в чат диспетчеров

РЕЗУЛЬТАТ

Курьер видит готовый порядок обхода до выхода на склад, диспетчер получает отдельным списком заказы, которые в смену не помещаются, а руководитель — недельную динамику времени в пути вместо ощущений и споров на планёрке.

КЛЮЧЕВОЙ НЮАНС

Без явного штрафа за возврат на склад в середине дня солвер охотно строит «дозагрузку»: формально дешевле, практически неисполнимо. Бытовые правила склада записываем в целевую функцию явно.

ЧТО МЫ ВЫУЧИЛИ НА ПРАКТИКЕ

Подводные камни

- ✕ **Оптимизация километров вместо времени**
  В городе 22 км по свободным улицам быстрее 14 км по стоящему проспекту. Целевая функция — суммарная длительность с departure_time, а не пробег.

- ✕ **Матрица одним большим запросом**
  Синхронный режим отдаёт максимум 100 элементов. Матрицу 40×40 режем на 16 блоков 10×10 и держимся в 40 RPS, иначе стабильный HTTP 429.

- ✕ **Геокодирование при каждом расчёте**
  Адреса меняются редко, а лимит HTTP-геокодера на бесплатном ключе — 1000 запросов в сутки. Без кэша квота выгорает уже к обеду.

- ✕ **Один ключ и один счётчик на все сервисы**
  JS API и HTTP-геокодер идут одним ключом с общей квотой, Матрица расстояний — своим, Маршрутизация — ключом из Настройки → Компания.

- ✕ **Нет режима работы без API**
  Недоступность внешнего сервиса не должна останавливать склад: держим вчерашнюю матрицу и печатную раскладку по зонам как деградированный режим.

- ✕ **Жёсткая диктовка маршрута курьеру**
  Право отклониться с фиксацией причины снимает саботаж. Долю отклонений считаем метрикой качества солвера, а не дисциплины сотрудника.

- ✕ **Тюнинг ACO вслепую**
  alpha, beta и коэффициент испарения подбираются на 15-20 архивных днях. Правка параметров по одному удачному утру даёт нестабильные маршруты.

- ✕ **Ключи и токены в репозитории**
  apikey карт и токен бота живут только в EnvironmentFile systemd с правами 600 и в менеджере паролей; в git уезжают имена переменных, не значения.

УРОВНИ РЕАЛИЗАЦИИ

Как правильно

МИНИМУМ

- Выгрузка заявок из 1С по расписанию и единый справочник координат клиентов
- Матрица времени с departure_time вместо расстояний по прямой линии
- Маршрутный лист курьеру в Telegram со ссылкой в Яндекс Навигатор
- Ключи API вне репозитория, в EnvironmentFile systemd с правами 600

НОРМАЛЬНО

- Кэш в Redis: геокодер 30 суток, матрица пробок 60 минут
- Вместимость, окна доставки и сервисное время как штрафы в солвере
- Ретраи с бэкоффом на 429 и 5xx, деградация на вчерашнюю матрицу
- Метрики в Prometheus: время расчёта, отклонения курьеров, ошибки API

ХОРОШО

- Еженедельная сверка солвера с OR-Tools 9.15 на архиве реальных дней
- MVRP Яндекс Маршрутизации там, где нужен SLA, а не свой код
- Дневной пересчёт по фактическим отказам и переносам, не только утренний
- Разбор dropped_locations и solver_status на еженедельной планёрке

ПРОВЕРЬТЕ У СЕБЯ

Чек-лист самопроверки

- [ ] Знаете ли вы, сколько часов в день курьеры реально проводят в пути, и откуда взята эта цифра?
- [ ] Есть ли у каждого клиента подтверждённые координаты в 1С, а не только текстовая строка адреса?
- [ ] Кэшируются ли ответы геокодера и не превышает ли TTL разрешённые условиями API 30 суток?
- [ ] Режется ли матрица на блоки в пределах лимита 100 элементов и потолка 40 запросов в секунду?
- [ ] Что происходит с утренней раскладкой, если API недоступен 30 минут: есть ли деградированный режим?
- [ ] Учтены ли в расчёте окна доставки, вместимость машины и сервисное время на точке?
- [ ] Хранятся ли apikey и токен бота вне репозитория, в EnvironmentFile с правами 600?
- [ ] Может ли курьер отклониться от маршрута с фиксацией причины и считаете ли вы долю таких отклонений?
- [ ] Видит ли руководитель недельную динамику времени в пути и стоимость API в рублях?
- [ ] Проверялось ли качество вашего солвера против эталонного решения на архиве прошлых дней?

Если хотя бы на два вопроса ответ «нет» или «не знаю» — тема требует внимания.

ЧТО МЫ ДЕЛАЕМ

Как поможет ITFresh

ITFresh — ИТ-аутсорсинг для юридических лиц до 50 рабочих мест в Москве и области. 15+ лет практики, собственная инфраструктура в дата-центре МТС (8 серверов Dell Xeon Platinum).

- Снимаем базовую линию по вашим данным из 1С за 2-3 недели: время в пути, пробег, срывы окон
- Разворачиваем маршрутизатор на вашем сервере: выгрузка из 1С, кэш, солвер, бот курьера, таймеры
- Настраиваем нарезку матрицы и кэш под ваш тариф карт, чтобы счёт за API был предсказуем
- Ставим мониторинг Prometheus и еженедельный отчёт диспетчерам с целевыми порогами
- Обучаем диспетчера и курьеров, вводим регламент отклонений и разбор нераспределённых заказов

15+
лет в ИТ-поддержке

50
рабочих мест — наш профиль

МТС
дата-центр, Москва

КОНТАКТЫ

Обсудить вашу задачу

Сайт	**itfresh.ru**
Телефон	**+7 903 729-62-41**
Telegram	**@ITfresh_Boss**

Бесплатно посмотрим вашу инфраструктуру по этому чек-листу и скажем, где тонко — без обязательств.

itfresh.ru

ДОКУМЕНТАЦИЯ И НАШ СТЕК

Техническая база

01 API Матрицы расстояний: формат запроса, лимит 100 элементов и 40 RPS (yandex.ru — 2026)

02 API Матрицы расстояний: ответ rows/elements, status, distance, duration (yandex.ru — 2026)

03 Яндекс Маршрутизация: add/mvrp, mvrp/fetch-result, solver_status (yandex.ru — 2026)

04 Яндекс Маршрутизация: модули интеграции с 1С, депо, транспорт, заказы (yandex.ru — 2026)

05 API Геокодера: суточные лимиты и кэширование результатов до 30 суток (yandex.ru — 2026)

06 OR-Tools Routing Options: стратегии старта и метаэвристики поиска (developers.google.com — 9.15)

07 1С:Предприятие 8.3.27: HTTP-сервисы /hs/ и стандартный интерфейс OData (its.1c.ru — 8.3.27)

08 Наш шаблон courier-routes: systemd unit, таймеры, профиль ACO (itfresh.ru — 2026)

Основано на официальной документации продуктов и нашей практике внедрения.