



Executor Balancer: многопоточная система распределения заявок в реальном времени

Создать легковесное решение для справедливого распределения поступающих заявок между исполнителями на основе параметров с конструктором параметров и условий для удобной настройки.

Суть кейса:

1. Заявки поступают большим потоком в автоматизированную информационную систему (АИС) из различных источников.
2. Сервис должен принимать заявки посредством интеграции с АИС.
4. Сервис должен определить активных исполнителей, соответствующих параметрам заявки, с учётом возможности изменения параметров в любой момент времени.
5. Сервис автоматически назначает подходящего исполнителя на заявку.

Описание кейса



- 1 У исполнителя более 20 параметров, влияющих на подбор подходящей заявки, регулярно появляются новые параметры
- 2 Каждая заявка включает более чем 15 параметров, определяющих её пригодность для конкретного исполнителя, список параметров постоянно расширяется
- 3 Параметры являются переменными и зависят от конфигурации как исполнителей, так и самих заявок. Прямой подход ("в лоб") к обработке всех возможных комбинаций усложняет разработку и масштабирование приложения, поскольку простое добавление нового параметра может нарушить существующий алгоритм и привести к ошибкам
- 4 Добавление новых параметров традиционным способом увеличивает объём кода и усложняет структуру программы. Это создаёт трудности при разработке, тестировании и поддержке проекта, повышает риск ошибок и замедляет процесс внесения изменений
- 5 Количество поступающих заявок достигает 4000 штук в час. Характер потока варьируется от высоких всплесков активности до стабильного уровня. Важно обеспечить точное распределение заявок между доступными исполнителями в режиме реального времени
- 6 Процесс назначения исполнителя выполняется асинхронно, фиксируется в финальной базе данных внешнего АИС спустя 2-10 секунд после завершения операции
- 7 Исполнители параллельно обрабатывают и завершают заявки, освобождая свои рабочие слоты
- 8 Существует ограничение на максимальное число обработанных заявок одним исполнителем в сутки, которое можно настроить индивидуально или сделать неограниченным
- 9 Отсутствует учёт важности («веса») исполнителей и заявок, что мешает внедрить справедливый механизм распределения. Например, необходимо учитывать разницу между квалифицированными и менее опытными специалистами, равно как и различия в уровне сложности обработки отдельных заявок.
- 10 Настройки исполнителей хранятся и изменяются во внешней информационной системе (АИС). Сами заявки также сохраняются и подвергаются изменениям в той же внешней АИС

Описание кейса

Цель:

Создать сервис, обеспечивающий справедливые и эффективные механизмы распределения заявок между исполнителями с поддержкой динамических настроек и анализа ключевых показателей эффективности (KPI).



Легкость настройки: предоставить удобный интерфейс конструктора для быстрого и легкого добавления новых параметров, используемых в процессе распределения.

Производительность: обеспечить высокую пропускную способность при приемлемой точности, обработку параллельных потоков данных без возникновения конфликтов ("race conditions")

Поддержка внешних интеграций: синхронизироваться с внешней АИС, где хранятся параметры исполнителя и заявки

Универсальность: возможность легко добавлять новые параметры для заявки и исполнителя в алгоритм распределения

«Вес» заявки и исполнителя: реализовать поддержку расчета значимости каждой заявки и квалификации исполнителей

Справедливое распределение: распределять заявки равномерно между исполнителями, допускается погрешность $\pm 1-2\%$.

Отчеты и графики: предоставлять интерактивные дашборды с отчетами и графиками по заявкам и исполнителям.

Документация и схема аналитики: создать документацию в виде BPMN-диаграммы, sequence-диаграммы и БД (ERD), а также описать ключевые проблемы и возможные альтернативные подходы в решении проблематики

Будет плюсом:

1. Выгрузка метрик дашборда в Excel.
2. Выгрузка метрик дашборда через API.
3. Хранение сводных и агрегированных метрик в отдельной таблице.
4. Обновление данных в реальном времени или с минимальной задержкой.
5. Акцент на точность расчетов, а не исключительно на скорость обработки.

Для кого:

- автоматизированные информационные системы (АИС);
- системы управления взаимоотношениями с клиентами (CRM);
- системы планирования ресурсов предприятия (ERP);
- службы контроля качества продукции и услуг;
- центры технической поддержки и колл-центры;
- интернет-магазины и e-commerce проекты;
- компании, оказывающие услуги населению и бизнесу;
- логистические центры и склады товаров;
- отделы разработки программного обеспечения;
- розничные торговые сети (ритейл);
- предприятие, где необходима ручная валидация человеком.

Стек (может быть любым)

Нет ограничений в использовании технологии и языков программирования. Допускается реализация с помощью машинного обучения и искусственного интеллекта

Сервис бекенд:

- принимает запросы с заявками из внешнего АИС, сохраняет в SQLite/Redis/MQ для обработки;
- получает сохраняет «кеш» исполнителей из внешнего из внешней АИС;
- получает сохраняет «кеш» изменения по одному исполнителю с настройками из внешнего АИС;
- динамическая настройка распределения и параметров веса;
- вычисляет исполнителя для заявки, отправляет запрос в внешний АИС с id исполнителя и заявки;
- конструктор условий алгоритма распределения, поддержка базовых операторов сравнения полей и типов данных, поддержка строковых массивов из справочников.

Демо

1. Реализовать эмуляцию внешнего АИС (CRUD):
 - 1.1. Методы чтения, записи и изменения настроек пользователя.
 - 1.2. Методы чтения, записи и изменения заявок.
2. Скрипты для многопоточной обработки данных и эмуляции работы внешнего АИС:
 - 2.1. Скрипт многопоточного заполнения заявок.
 - 2.2. Скрипт многопоточной смены статусов заявок, симулирующий работу исполнителей и освобождение слотов.
3. Сервис распределения заявок.

Веб-интерфейс:

- настройка параметров распределения конструктора;
- отображение дашборда в разных разрезах и таймлайнах отчета распределения.

План демонстрации работы use-case:

1. Настройка конструктора условий алгоритма распределения.
2. Примеры заявки и настроек пользователя в внешней АИС.
3. Запуск сценария наполнения внешней АИС: запуск многопоточных заданий для создания заявок с изменением их статусов.
4. Работа сервиса распределения:
 - 4.1. Внешняя АИС направляет каждую новую заявку в сервис распределения.
 - 4.2. Сервис мгновенно фиксирует заявку и выбирает исполнителя
5. Отражение динамики распределения на UI-дашборде в реальном времени.
6. Объяснение математики и логики работы алгоритма, включая его сильные стороны и ограничения.

Описание кейса



Условия распределения

Исполнитель: специалист обрабатывающий заявку

Заявка: электронное обращение в компанию

Первичная заявка: новое обращение статус «рассмотрение», может быть завершена, либо вернуться заявителю в статус «доработка»

Вторичная заявка: обращение вернулось после статуса «доработка» исполнителю в статус «рассмотрение»

Условия распределения:

1. Только активные исполнители участвуют в распределении.
2. Исполнитель назначается, если у него меньше открытых заявок, чем у остальных кандидатов.
3. Перераспределение вторичной заявки при изменении статуса на «рассмотрение». Если исполнитель становится неактивным или перестаёт соответствовать параметрам, он теряет право обработку обращения, заявка должна быть перераспределена.
4. Назначаемый исполнитель должен соответствовать всем необходимым параметрам конкретной заявки.
5. Суточный лимит по количеству заявок не превышен.
6. Распределение родительской заявки. Если parent_id не пустой и выполняются условия параметров (исключить суточный лимит исполнителя) – назначаем исполнителя с заявки по parent_id, если пуст назначаем нового.
7. Учёт веса заявки и исполнителя. Каждый объект обладает определённым весом, отражающим его важность или сложность. Предлагаем ввести собственные критерии классификации (например, высокая/средняя/низкая).