ДОМАШКА 0. РЕШЕНИЕ.

Основные модули:

1. **Клиентский модуль (Client):**
   - Запрос услуг, просмотр инвойсов, отслеживание статусов.
   - История заказов (выполненные, отмененные, активные).
   - Уведомления о статусах заказов (почтовые нотификации).
2. **Модуль исполнителей (Worker):**
   - Управление пулом исполнителей.
   - Задачи для воркеров: активные, завершенные, отмененные.
   - Логика подтверждения начала и завершения работы.
   - Учёт и выдача расходников исполнителям.
3. **Модуль подбора и распределения задач (Matching):**
   - Подбор воркеров к задачам с учетом их доступности.
   - Автоматическое распределение заказов между воркерами.
4. **Административный модуль (Admin Panel):**
   - Управление услугами (создание/редактирование услуг).
   - Проверка заказов менеджерами (информация о заказе, связь с клиентом).
   - Система ставок (выполнен/провален заказ).
   - Анализ отмененных/проваленных заказов.
   - Формы проверки качества заказов и связи с клиентами.
5. **Финансовый модуль (Finance):**
   - Расчет стоимости услуг после подбора исполнителя.
   - Еженедельные инвойсы для клиентов (cron).
   - Ежемесячное начисление зарплат исполнителям (cron).
   - Управление списаниями, штрафами и балансами.
   - История финансовых операций (клиенты и исполнители).
6. **Модуль уведомлений (Notification):**
   - Отправка email-уведомлений клиентам и исполнителям.
7. **Модуль качества (Quality):**

- Проверка отмененных и проваленных заказов.
- Анализ работы воркеров.

Обоснование архитектурных решений:

1. Выбор модулей:

Каждый модуль решает конкретную задачу:

- **Client:** нужен для взаимодействия с пользователями системы.
- **Worker:** позволяет управлять исполнителями и отслеживать их задачи.
- **Matching:** автоматизирует подбор подходящего исполнителя для заказа.
- **Admin Panel:** предоставляет интерфейс для управления услугами и анализа статистики заказов и отработки проваленных заявок.
- **Finance:** обрабатывает все финансовые операции и инвойсы.
- **Notification:** обеспечивает доставку уведомлений.
- **Quality:** улучшает работу системы через анализ проваленных и отмененных заказов.
- **Scheduler:** выполняет регулярные задачи (cron), такие как инвойсы или начисление зарплат.
- **Платежный сервис:** используется для интеграции с внешними провайдерами оплаты.

2. Выбор структуры:

- **Монолитная архитектура:**
  Выбрана для быстрого старта, упрощённой поддержки и более дешёвого развертывания. Монолит позволяет избежать сложностей в межсервисных коммуникациях и уменьшить время на настройку CI/CD для каждого сервиса.
- Дальнейший переход на микросервисы возможен, если нагрузка возрастёт или отдельные модули потребуют независимого масштабирования (например, Notification или Matching).

3. Коммуникации:

- Внутри монолита модули взаимодействуют через вызовы функций/методов (синхронные операции). Это проще для дебага и управления.
- **Scheduler:** запускает задачи по расчету ЗП через планировщик.
- **Платежный сервис:** взаимодействие через HTTP API (асинхронные запросы с обработкой ответов).
- **Модуль уведомлений:** отправляет email через внешние SMTP-сервисы или API.

4. Критические и спорные моменты:

1. **Узкое место базы данных:**
   Единая база данных может стать точкой отказа и узким местом при увеличении нагрузки.
   Решение: на этапе роста — переход на шардирование или разбивка на микросервисы где у каждого будет своя бд.

2. **Отсутствие брокера сообщений:**
   Пока нет необходимости во внедрении брокера, так как задачи синхронны. Однако это может потребоваться при масштабировании.
3. **Глобальная зависимость между модулями:**
   Монолит может усложнить рефакторинг или миграцию в микросервисы.
4. **Уведомления:**
   Если объем отправки email резко увеличится, система может столкнуться низкой производительностью.