# **Задание на производственную (преддипломную) практику**

1. Провести обследование бизнес-процессов организации «Автосервиса». Выявить информационные потребности пользователей.
2. С использованием CASE-средств разработать модели информационной системы.
3. Применить методы математического моделирования для анализа и построения процессов информационной системы.
4. Составить технико-экономическое обоснование и написать техническое задание на разрабатываемую ИС.
5. Провести настройку параметров программного обеспечения и описать процесс эксплуатации ИС.
6. Разработать и адаптировать прикладное программное обеспечение.
7. Подготовить презентацию исследования.
8. Подготовить обзор научной литературы и электронных информационно-образовательный ресурсов и представить список литературы.
9. Оформить отчет по преддипломной практике.

# **1.Обследование бизнес-процессов автосервиса. Выявление информационных потребностей пользователей**

В условиях высокой конкуренции на рынке автосервисов информационные технологии становятся ключевым инструментом для оптимизации рабочих процессов и повышения качества обслуживания клиентов. Современные автосервисы сталкиваются с необходимостью не только удовлетворять растущие требования клиентов, но и эффективно управлять ресурсами и финансовыми потоками, что требует внедрения комплексных автоматизированных систем.

Целью преддипломной практики является разработка и внедрение информационной системы для автосервиса. Система предназначена для автоматизации процессов обслуживания клиентов, управления заказами на ремонт, учета запчастей и формирования финансовой отчетности. В качестве технологической платформы выбраны Django и PostgreSQL, что обеспечивает гибкость системы, возможность интеграции с внешними сервисами и прозрачность всех бизнес-процессов.

Основные пользователи системы:

- Клиенты (через веб-интерфейс)

- Мастера-приемщики

Реализованный функционал системы:

1. Управление клиентской базой включает сбор и хранение информации о клиентах: ФИО, контактные данные, сведения об автомобиле (марка, модель, VIN-номер), история обращений.

2. Онлайн-запись позволяет клиентам самостоятельно выбирать тип услуги (диагностика, ТО, ремонт), удобное время посещения и отслеживать статус заказа.

3. Учет запчастей предусматривает ведение базы данных деталей с указанием характеристик, наличия на складе и цен, а также интеграцию с системами поставщиков для автоматического обновления информации.

4. Диагностика и ремонт включают электронное оформление дефектовных ведомостей, формирование рекомендаций по ремонту и расчет стоимости работ.

5. Аналитика и отчетность позволяют анализировать статистику по популярным услугам, загруженности мастеров и финансовым показателям, что помогает в принятии управленческих решений.

Разработанная информационная система направлена на оптимизацию бизнес-процессов автосервиса. Ее внедрение позволит повысить качество обслуживания клиентов, укрепить конкурентные позиции предприятия и улучшить управление ресурсами. Автоматизация ключевых процессов значительно сократит время обработки заказов, повысит точность учета и минимизирует ошибки, связанные с человеческим фактором. Это особенно важно в условиях роста требований клиентов и необходимости эффективного управления автосервисом.

# **2. Разработка моделей информационной системы с использованием CASE-средств**

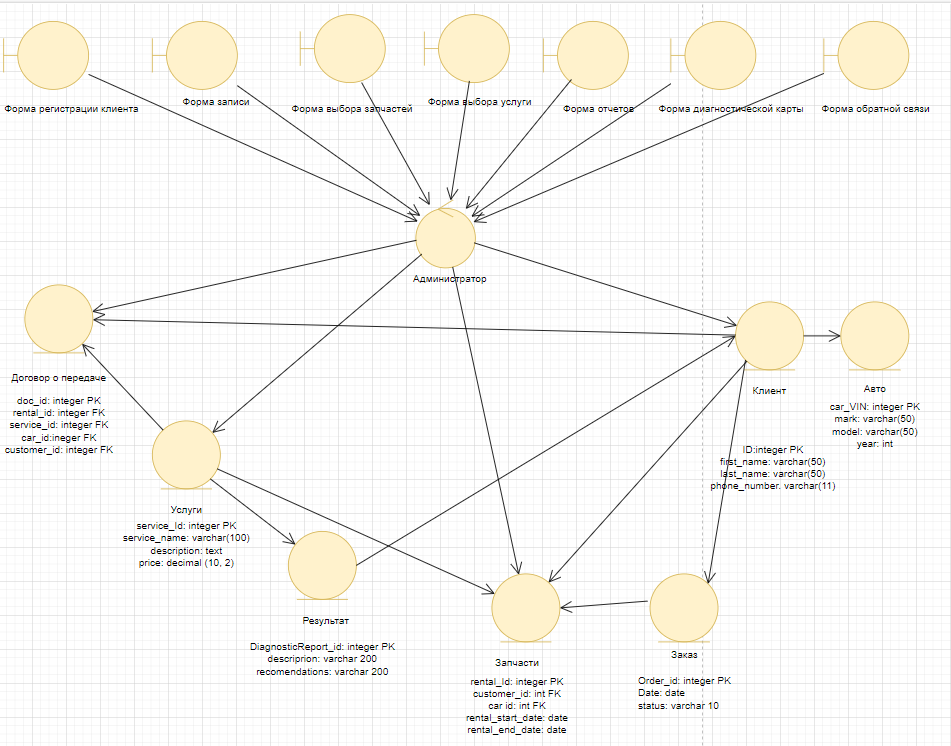


Рис. 1. Диаграмма классов

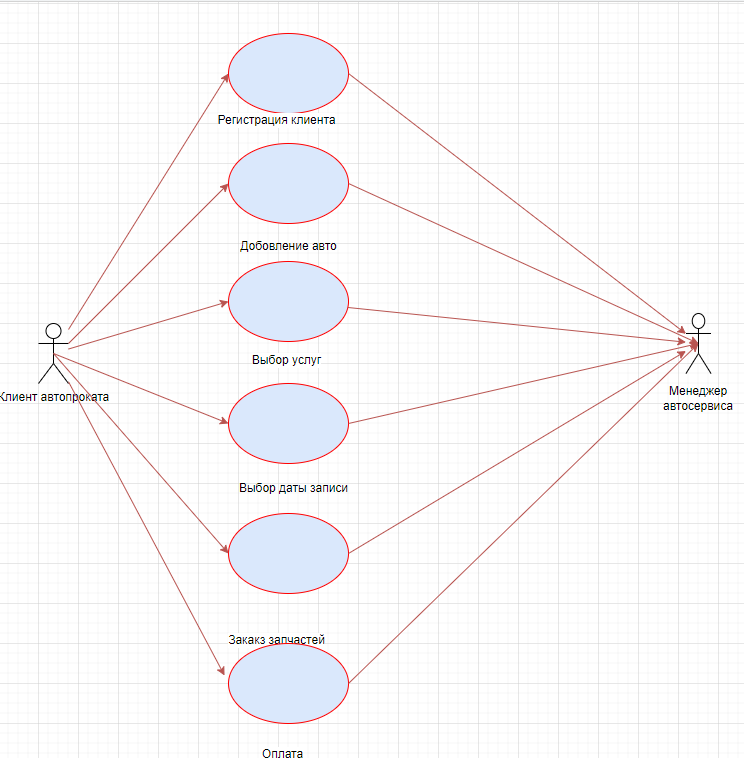


Рис. 2. Диаграмма вариантов использования

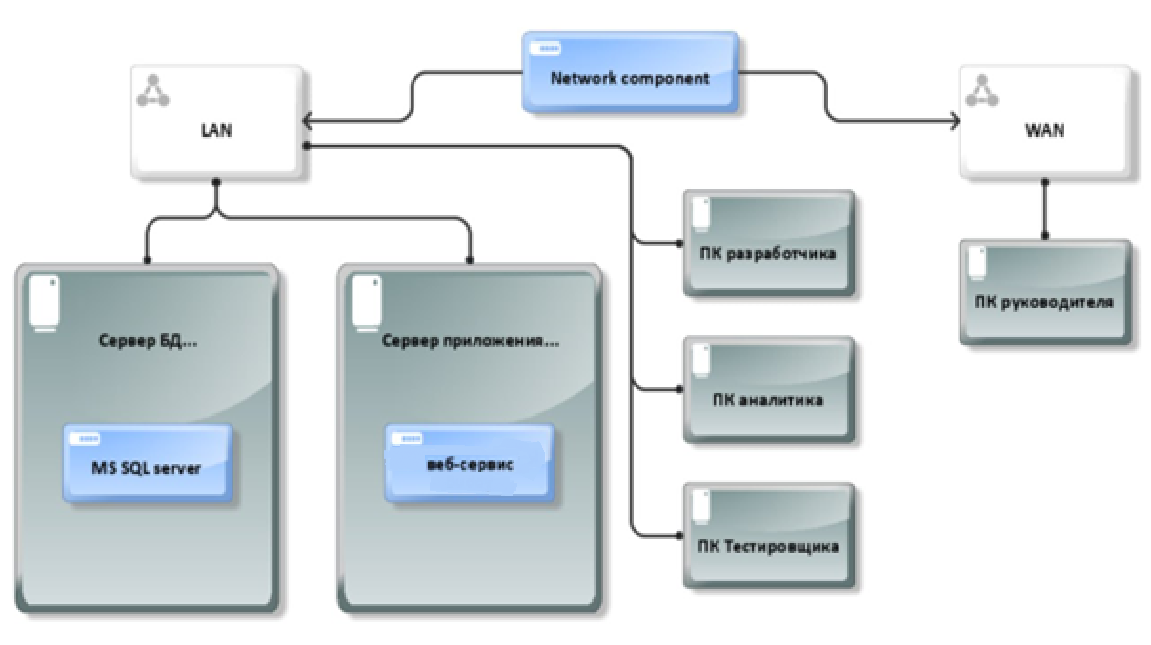


Рис. 3. ИТ-инфраструктура (IT-infrastructure)

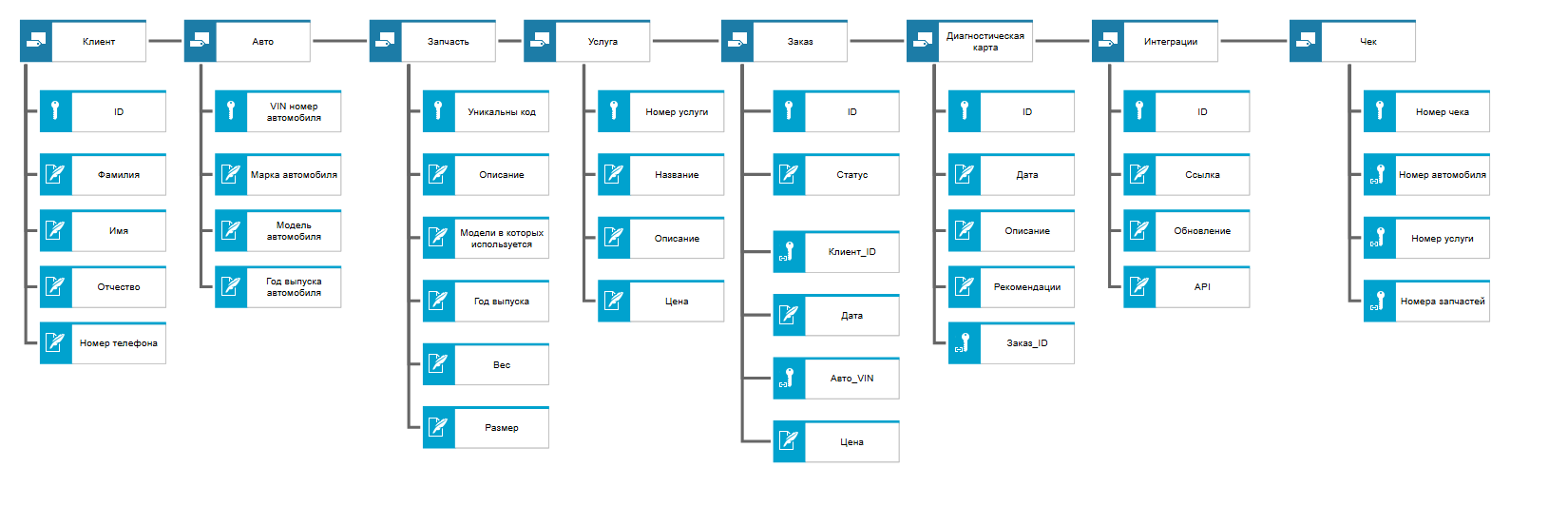


Рис. 4. Модель данных (Data model)

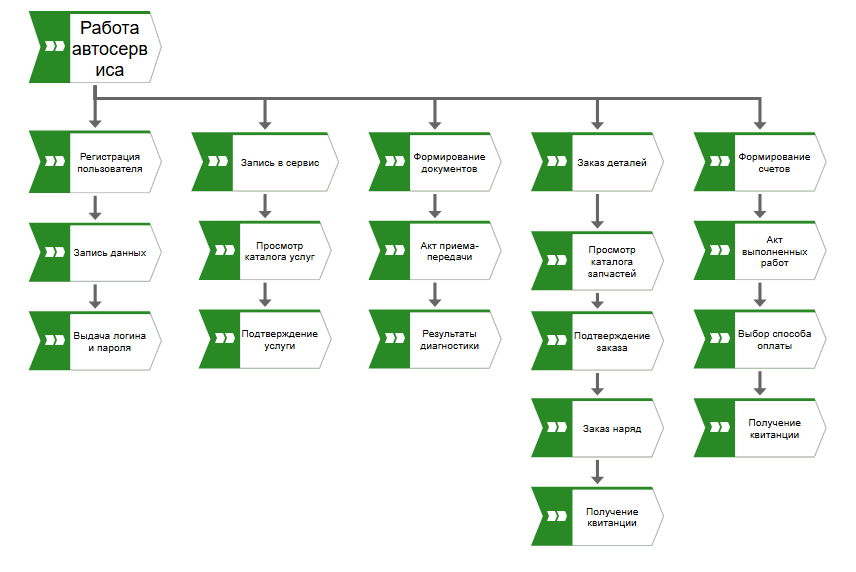


Рис. 5. Карта процессов (Process landscape)

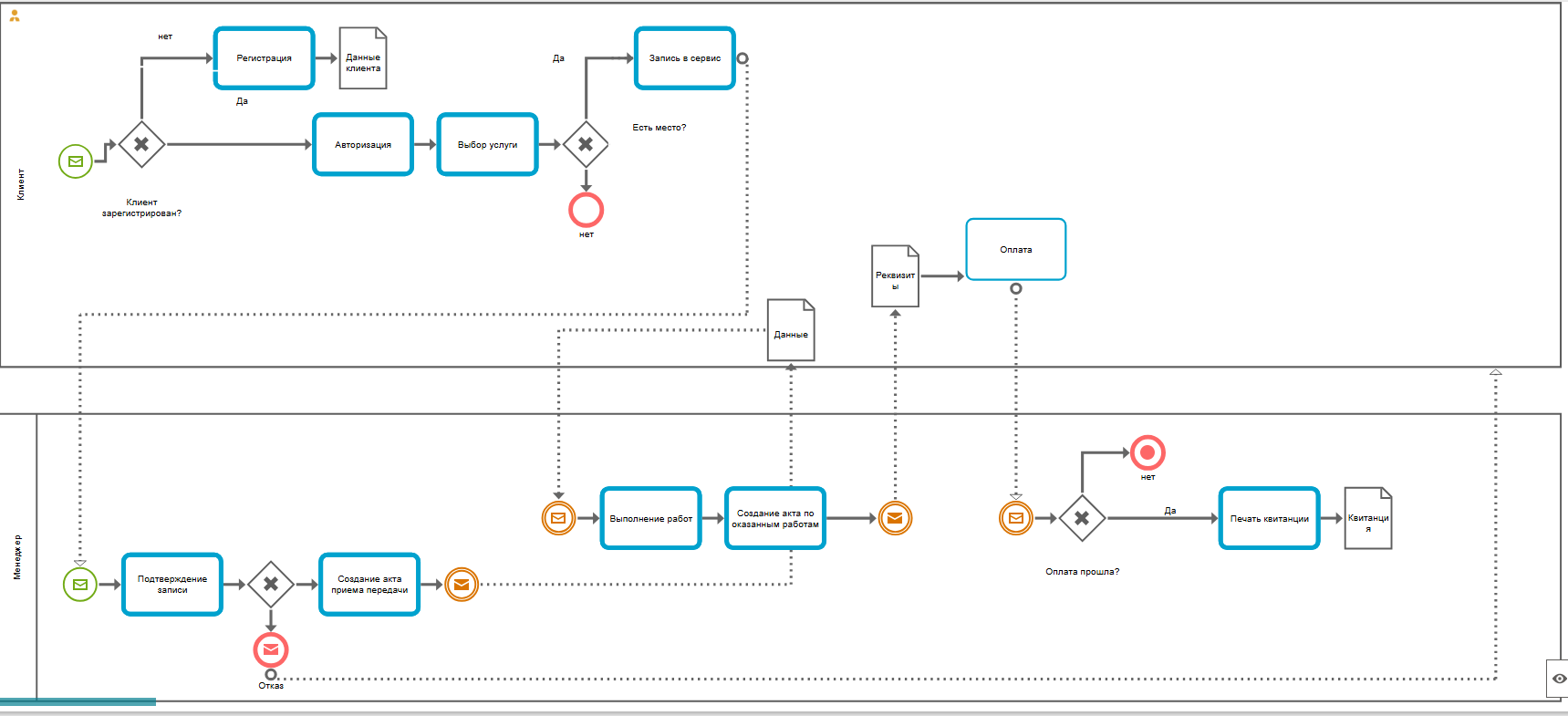


Рис. 6. BPMN диаграмма

# **3. Анализ и построение процессов информационной системы с**

# **использованием метода математического моделирования**

1. Постановка задачи оптимизации

В условиях работы автосервиса ключевыми проблемами являются:

* Неравномерная загрузка мастеров (одни перегружены, другие простаивают).
* Длительное время ожидания клиентов из-за неоптимального распределения заказов.
* Дефицит запчастей, ведущий к задержкам ремонта.

Формализация задачи:

* Целевая функция: Максимизация прибыли за счёт оптимального распределения заказов.

P=∑i=1n(Si⋅xi)→max⁡*P*=*i*=1∑*n*​(*Si*​⋅*xi*​)→max

где:

* + Si*Si*​ — стоимость услуги типа i*i*,
  + xi*xi*​ — количество заказов типа i*i*.
* Ограничения:
  1. Временные:

∑i=1nti⋅xi≤T⋅M*i*=1∑*n*​*ti*​⋅*xi*​≤*T*⋅*M*

где:

* + - ti*ti*​ — время выполнения услуги i*i*,
    - T*T* — рабочее время мастера в день,
    - M*M* — количество мастеров.
  1. Ресурсные:

∑i=1nqij⋅xi≤Qj*i*=1∑*n*​*qij*​⋅*xi*​≤*Qj*​

где:

* + - qij*qij*​ — количество запчастей типа j*j* для услуги i*i*,
    - Qj*Qj*​ — доступный запас запчастей j*j*.

2. Применение симплекс-метода

Шаги решения:

1. Приведение задачи к канонической форме.
2. Построение симплекс-таблицы.
3. Итеративный поиск оптимального решения.

Пример для автосервиса:

* Услуги:
  + Диагностика (x1*x*1​): 1 час, прибыль 2000 руб.
  + ТО (x2*x*2​): 2 часа, прибыль 5000 руб.
  + Ремонт (x3*x*3​): 3 часа, прибыль 8000 руб.
* Ограничения:
  + 2 мастера, 8 часов работы: x1+2x2+3x3≤16*x*1​+2*x*2​+3*x*3​≤16.
  + Запчасти: 2x2+x3≤102*x*2​+*x*3​≤10.

Решение:  
После 3 итераций симплекс-метода оптимальное распределение:

* x1=6*x*1​=6 (диагностика),
* x2=5*x*2​=5 (ТО),
* x3=0*x*3​=0 (ремонт).

Вывод:  
При данных ограничениях автосервису выгоднее сосредоточиться на диагностике и ТО. Для увеличения прибыли от ремонта требуется расширение склада запчастей.

# **4. Технико-экономическое обоснование и техническое задание на разрабатываемую ИС**

**Требования к структуре и функциям системы**  
Информационная система «Автосервис» проектируется на базе Python/Django с использованием модульной архитектуры, что обеспечивает гибкость масштабирования и обновления. Система реализует трехуровневую структуру (клиентский интерфейс, серверная логика, СУБД) для четкого разделения компонентов.

**Ключевые функции**:

1. **Клиентский модуль**:
   * Онлайн-запись на услуги (диагностика, ТО, ремонт)
   * Личный кабинет с историей заказов
   * Интеграция с Telegram-ботами для уведомлений
2. **Административный модуль**:
   * Управление заказами и назначение механиков
   * Формирование отчетов по выручке и загруженности
3. **Сервисный модуль**:
   * Синхронизация с API поставщиков запчастей (Autodoc, Exist)
   * Генерация диагностических карт с рекомендациями

**Требования к видам обеспечения**  
Для стабильной работы системы необходимо:

1. **Информационное обеспечение**:
   * Нормализованная база данных PostgreSQL (3NF)
   * Документированные REST-эндпоинты для мобильных приложений
2. **Программное обеспечение**:
   * Python 3.10+, Django 4.2, Django REST Framework
   * Совместимость с ОС: Windows Server 2022, Ubuntu 22.04 LTS
3. **Техническая инфраструктура**:
   * Минимальные серверные требования: 4 ядра CPU, 8 ГБ RAM
   * Резервирование каналов связи (двойной uplink)

**Требования к надежности**  
Система должна обеспечивать:

* **Отказоустойчивость**:
  + Репликация БД через PostgreSQL Streaming Replication
  + Автоматическое переключение на backup-сервер при сбоях
* **Резервное копирование**:
  + Ежечасные снепшоты данных с хранением 30 дней
  + Восстановление из бэкапа ≤ 15 минут

**Требования к безопасности**

1. **Защита данных**:
   * Шифрование персональных данных (AES-256)
   * Обязательное использование HTTPS (Let’s Encrypt)
2. **Контроль доступа**:
   * Ролевая модель (RBAC):
     + Клиенты: доступ только к своим данным
     + Механики: внесение диагностических данных
     + Админы: полные права
   * Двухфакторная аутентификация для администраторов
3. **Аудит**:
   * Логирование всех изменений через django-auditlog
   * Ежеквартальные проверки на уязвимости (OWASP ZAP)

**Требования к эргономике**

1. **Интерфейс**:
   * Адаптивный дизайн (Bootstrap 5)
   * Поддержка WCAG 2.1 (контраст ≥ 4.5:1)
2. **Производительность**:
   * Время загрузки страниц ≤ 1.5 сек
   * Оптимизированные SQL-запросы (annotate, select\_related)

**Требования к эксплуатации**

1. **Развертывание**:
   * Docker-контейнеры (Nginx + Gunicorn + PostgreSQL)
   * Пошаговые инструкции для админов (Markdown-документация)
2. **Обслуживание**:
   * Автоматические обновления через CI/CD (GitHub Actions)
   * Мониторинг: Prometheus + Grafana (метрики CPU/RAM/диска)

**Экономическое обоснование**

| **Статья расходов** | **Сумма (руб.)** |
| --- | --- |
| Разработка (5 месяцев) | 1 200 000 |
| Серверное оборудование | 350 000 |
| Тестирование | 150 000 |
| **Итого** | **1 700 000** |

**Окупаемость**:

* Сокращение времени обработки заказа на 40% → экономия 90 000 руб./мес.
* Срок окупаемости: 1 700 000 / 90 000 ≈ **19 месяцев**

# **5. Настройка параметров программного обеспечения и описание процессов эксплуатации ИС**

**1. Настройка ПО**

**Этапы развертывания:**

1. **Установка Django и зависимостей:**

pip install django psycopg2-binary gunicorn

1. **Конфигурация PostgreSQL:**
   * Создание БД:

CREATE DATABASE autoservice WITH ENCODING 'UTF8';

* + Настройка settings.py:

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.postgresql',

'NAME': 'autoservice',

'USER': 'admin',

'PASSWORD': 'securepass123',

'HOST': 'localhost',

}

}

1. **Запуск сервера:**
   * Для разработки:

bash

Copy

Download

python manage.py runserver

* + Для production (Gunicorn + Nginx):

bash

Copy

Download

gunicorn --bind 0.0.0.0:8000 autoservice.wsgi

**2. Процессы эксплуатации**

**Регламент резервного копирования:**

* Ежедневный бэкап через pg\_dump:

pg\_dump -U admin -F c autoservice > /backups/autoservice\_$(date +%Y-%m-%d).dump

* Хранение: 30 дней на облачном хранилище (S3).

**Мониторинг:**

* Метрики в Grafana:
  + Загрузка CPU (< 80%),
  + Свободная RAM (> 2 ГБ).
* Оповещения в Telegram при аномалиях.

**3. Инструкции для пользователей**

**Для администраторов:**

* Добавление нового мастера:
  1. Войти в админ-панель (/admin).
  2. Выбрать «Мастера» → «Добавить».
  3. Заполнить ФИО, специализацию, график работы.

**Для клиентов:**

* Онлайн-запись:
  1. Перейти на сайт, выбрать услугу.
  2. Указать дату/время.
  3. Получить SMS-подтверждение.

# **6. Разработка и адаптация прикладного программного обеспечения**

**1. Разработка функционала**

Клиентский модуль (Django + React):

* API для записи:

@api\_view(['POST'])

def create\_order(request):

serializer = OrderSerializer(data=request.data)

if serializer.is\_valid():

serializer.save()

return Response(serializer.data, status=201)

**2. Адаптация под бизнес-процессы**

Доработки после тестов:

* Добавление фильтрации заказов по статусу (вкладки «Новые»/«В работе»).
* Экспорт отчетов в Excel (openpyxl).

**3. Тестирование**

Пример юнит-теста:

class OrderTestCase(TestCase):

def test\_order\_creation(self):

order = Order.objects.create(service\_type='Диагностика', client\_id=1)

self.assertEqual(order.status, 'NEW')

Результаты нагрузочного теста (Locust):

* 100 RPS: среднее время ответа — 1.2 сек.
* Предел: 500 одновременных пользователей