# 1. Интерфейсы для ролей

Владелец:

* + Достаточно стандартной Django-админки (/admin).
  + Почему: Используется редко (только для глобальных настроек/назначений), не требует кастомизации.
  + Что видит: Все данные (пользователи, заказы, финансы), журнал действий, настройки доступа.

Менеджеры:

* + Кастомный интерфейс на Vue с упором на:
    - Управление заказами (фильтры, статусы, назначение рабочих).
    - Просмотр аналитики (графики загрузки, прибыль).
    - Чат/уведомления (если нужно).

Рабочие:

* + Упрощённый Vue-интерфейс с:
    - Формой добавления заказа (выпадающие списки, шаблоны).
    - Просмотром своих задач.
    - Минимум действий (только CRUD для своих объектов).

# 2. Архитектура (Django-DRF-PostgreSQL-Vue)

* Django:
  + Ядро системы, ORM для работы с БД.
  + Стандартная админка для владельца.
* DRF (Django REST Framework):
  + REST API для фронтенда (Vue).
  + CRUD для всех моделей (заказы, пользователи и т.д.).
  + Авторизация: JWT-токены (или сессии).
* PostgreSQL:
  + Хранение всех данных (пользователи, заказы, логи).
  + Настроена на хостинге (SpaceWeb).
* Vue:
  + Единственный фронтенд для менеджеров/рабочих.
  + Шаблоны Django не используются (чистый SPA).
  + Общение с БД: Только через DRF (никаких прямых запросов к PostgreSQL).

# 3. Схема запросов

Vue (Frontend) → Axios → DRF (API) → Django (ORM) → PostgreSQL

↑ ↑

(JWT-токен) (Проверка прав доступа)

# 4. Ключевые принципы

Разделение интерфейсов:

* + Владелец → Django Admin.
  + Менеджеры/Рабочие → Vue (разные UI для ролей).

Безопасность:

* + Нет прямого доступа фронтенда к БД.
  + Все проверки прав на стороне DRF.

Гибкость:

* + Vue-интерфейс можно дорабатывать без изменений в Django.

# Новая тема

**1. Календарь заказов для владельца**

* **Решение**:
  + Добавить в Django Admin **кастомную страницу** с календарём (используя библиотеку django-admin-interface или jet-admin).
  + Или дать владельцу доступ к API (через DRF) и вывести календарь в отдельном Vue-интерфейсе (если нужен сложный функционал).
* **Где данные**:
  + Заказы хранятся в таблице orders\_order, даты в поле created\_at или deadline.

**2. Учётные записи в БД**

* **Все пользователи** хранятся в auth\_user (стандартная таблица Django).
* **Различие ролей**:
  + is\_superuser=1 → Владелец.
  + is\_staff=1 + группа Managers → Менеджеры.
  + is\_staff=0 + группа Workers → Рабочие.
* **Где смотреть**:
  + В DBeaver: SELECT username, is\_superuser, is\_staff FROM auth\_user;.

**3. Дизайн интерфейсов (эскизы)**

**Менеджер:**

* **Дашборд**:
  + Календарь заказов (неделя/месяц).
  + Графики загрузки.
  + Кнопки: "Добавить заказ", "Назначить рабочего".
* **Список заказов**:
  + Фильтры (статус, клиент, дата).
  + Сортировка по срочности.

**Рабочий:**

* **Минималистичный интерфейс**:
  + Форма: "Выбрать шаблон заказа" → "Указать количество" → "Отправить".
  + Список своих задач на сегодня.

**4. Основные DRF Endpoints**

python

Copy

Download

*# urls.py*

urlpatterns = [

path('api/auth/', include('rest\_framework.urls')), *# JWT-логин*

path('api/orders/', OrderViewSet.as\_view({'get': 'list', 'post': 'create'})),

path('api/orders/<int:pk>/', OrderViewSet.as\_view({'patch': 'update'})),

path('api/calendar/', CalendarView.as\_view()), *# Календарь заказов*

path('api/users/', UserListAPI.as\_view()), *# Только для владельца*

]

**5. JWT-авторизация**

1. Установка:

bash

Copy

Download

pip install djangorestframework-simplejwt

1. Настройка (settings.py):

python

Copy

Download

REST\_FRAMEWORK = {

'DEFAULT\_AUTHENTICATION\_CLASSES': [

'rest\_framework\_simplejwt.authentication.JWTAuthentication',

]

}

1. Endpoints:
   * /api/auth/token/ — получить токен.
   * /api/auth/token/refresh/ — обновить токен.

**6. Первоначальная настройка Django для хостинга**

1. **Настройки (settings.py)**:

python

Copy

Download

DEBUG = False

ALLOWED\_HOSTS = ['ваш-домен.spaceweb.ru', '127.0.0.1']

STATIC\_ROOT = '/home/ваш-логин/www/static' *# Путь на хостинге*

1. **Пакеты для установки**:

bash

Copy

Download

pip install gunicorn psycopg2-binary

pip freeze > requirements.txt

1. **Заливка на SpaceWeb**:
   * Через FTP: загрузите весь проект (кроме venv).
   * В панели хостинга:
     + Создайте PostgreSQL БД.
     + Настройте WSGI (укажите путь к wsgi.py).

# Новая тема 2

**🔹 2. Настройка Django (Backend)**

**Шаг 1: Локальная подготовка**

1. Установите зависимости:

bash

Copy

Download

pip install django psycopg2-binary djangorestframework django-cors-headers

1. Создайте проект Django:

bash

Copy

Download

django-admin startproject backend

cd backend

python manage.py startapp api

**Шаг 2: Настройка settings.py**

1. Добавьте в INSTALLED\_APPS:

python

Copy

Download

INSTALLED\_APPS = [

...

'rest\_framework',

'corsheaders',

'api',

]

1. Настройте БД:

python

Copy

Download

DATABASES = {

'default': {

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

'NAME': 'mycrm',

'USER': 'ваш\_пользователь',

'PASSWORD': 'пароль',

'HOST': 'хост\_spaceweb', *# Например, pgsqlXX.spaceweb.ru*

'PORT': '5432',

}

}

1. Добавьте CORS (для Vue):

python

Copy

Download

CORS\_ALLOWED\_ORIGINS = [

"https://ваш-сайт.spaceweb.ru",

"http://localhost:8080", *# Для разработки*

]

MIDDLEWARE = [

...

'corsheaders.middleware.CorsMiddleware',

]

**Шаг 3: Создайте API для авторизации**

1. В api/models.py:

python

Copy

Download

from django.contrib.auth.models import User

from django.db import models

class Profile(models.Model):

user = models.OneToOneField(User, on\_delete=models.CASCADE)

phone = models.CharField(max\_length=20)

1. Выполните миграции:

bash

Copy

Download

python manage.py makemigrations

python manage.py migrate

1. Создайте сериализатор и ViewSet (DRF):

python

Copy

Download

*# api/serializers.py*

from rest\_framework import serializers

from django.contrib.auth.models import User

class UserSerializer(serializers.ModelSerializer):

class Meta:

model = User

fields = ['id', 'username', 'email']

*# api/views.py*

from rest\_framework.viewsets import ModelViewSet

from .serializers import UserSerializer

from django.contrib.auth.models import User

class UserViewSet(ModelViewSet):

queryset = User.objects.all()

serializer\_class = UserSerializer

1. Добавьте URLs в backend/urls.py:

python

Copy

Download

from django.urls import path, include

from rest\_framework.routers import DefaultRouter

from api.views import UserViewSet

router = DefaultRouter()

router.register(r'users', UserViewSet)

urlpatterns = [

path('api/', include(router.urls)),

]

# Новая тема 3