Техническое задание (ТЗ) на бэкенд проекта "Автотовары"

1. Описание проекта

Проект представляет собой платформу для продажи автотоваров. Пользователи могут просматривать каталог, добавлять товары в корзину, оформлять заказы, а также получать информацию о доставке. Платформа доступна через веб-сайт, мобильное приложение и Telegram-бот.

2. Технологический стек

- **Backend**: Python (Django, Django Rest Framework)
- **Frontend**: React JS
- **База данных**: PostgreSQL (ORM - Django ORM)
- **Аутентификация**: JWT (JSON Web Token), Google OAuth
- **Развертывание**: Docker, Nginx, Gunicorn
- **Хранение файлов**: Amazon S3 или локальное хранилище
- **Платежные системы**: Payme, Click, Stripe

3. Функциональные требования

3.1. Пользовательская аутентификация и управление аккаунтом

- Регистрация пользователей (по email, Google OAuth)
- Авторизация и управление сессиями
- Восстановление пароля через email
- Профиль пользователя (редактирование данных, история заказов)

3.2. Роли пользователей

- **Администратор**: управление пользователями, заказами, товарами, платежами
- **Продавец**: добавление товаров, управление заказами
- **Покупатель**: просмотр каталога, добавление товаров в корзину, оформление заказов, оставление отзывов

3.3. Каталог товаров

- CRUD-операции для товаров (создание, обновление, удаление, просмотр)
- Категоризация товаров (Категория → Подкатегория)
- Фильтрация и поиск по товарам
- Добавление изображений к товарам
- Характеристики товаров (ProductProperty)

3.4. Корзина и оформление заказов

- Добавление и удаление товаров из корзины
- Оформление заказа (указание данных для доставки, выбор оплаты)
- Управление статусами заказов (в обработке, отправлен, доставлен)

3.5. Оплата и доставка

- Интеграция с платежными системами (Payme, Click, Stripe)
- Расчет стоимости доставки
- Отслеживание статуса доставки

3.6. Административная панель

- Управление пользователями
- Управление заказами
- Управление товарами и категориями
- Аналитика и отчёты

4. Архитектура базы данных (на основе диаграммы)

- **CustomUser** (id, phone_number, city_id)
- **City** (id, name, region_id)
- **Region** (id, name)

- **Cart** (id, user, product, amount)
- **Order** (id, user, phone_number, total_price, city, address, floor)
- **OrderProduct** (id, order, user, product, amount)
- **Category** (id, name, image)
- **SubCategory** (id, name, category)
- **Product** (id, name, brand, description, price, amount, guaranty, status, country, subCategory, owner)
- **ProductImage** (id, image, product)
- **ProductProperty** (id, name, context, product)

5. API-эндпоинты

5.1. Аутентификация

- POST /api/auth/register/ - регистрация
- POST /api/auth/login/ - вход в систему
- POST /api/auth/logout/ - выход из системы
- POST /api/auth/password-reset/ - восстановление пароля

5.2. Товары

- GET /api/products/ - список товаров
- GET /api/products/{id}/ - детальная информация о товаре
- POST /api/products/ - добавление товара (продавец, админ)
- PUT /api/products/{id}/ - обновление товара (продавец, админ)
- DELETE /api/products/{id}/ - удаление товара (продавец, админ)

5.3. Заказы

- POST /api/orders/ - оформление заказа

- GET /api/orders/ - список заказов пользователя
- GET /api/orders/{id}/ - детали заказа
- PATCH /api/orders/{id}/ - обновление статуса заказа (админ, продавец)

5.4. Корзина

- GET /api/cart/ - содержимое корзины
- POST /api/cart/add/ - добавить товар в корзину
- DELETE /api/cart/remove/ - удалить товар из корзины

5.5. Админ-панель

- GET /api/admin/orders/ - управление заказами
- GET /api/admin/products/ - управление товарами
- GET /api/admin/users/ - управление пользователями

6. Развертывание

1. Настроить сервер (Ubuntu, Docker, Nginx, PostgreSQL)
2. Развернуть Django-проект
3. Настроить хостинг для изображений
4. Настроить CI/CD (GitHub Actions, Docker Compose)

ТЗ обновлено с учетом переданных данных. Доработки и дополнения возможны в процессе разработки.