**Технический проект:** **Программная система для управления кулинарными рецептами и автоматизации формирования списка продуктов.**

Настоящий документ описывает и обосновывает архитектурные, технологические и проектные решения для создания веб-приложения, обеспечивающего публикацию и управление кулинарными рецептами, взаимодействие пользователей (избранное, подписки), а также автоматическое формирование и экспорт агрегированного списка покупок.

Система ориентирована на конечных пользователей (домашние кулинары, авторы рецептов) и администратора платформы. Решение реализуется как веб-приложение и включает два ключевых слоя:

1. Клиентский слой (Frontend/UI): веб-интерфейс, предоставляющий навигацию по каталогу рецептов, детальный просмотр, действия «добавить в избранное», «подписаться на автора», работу со списком покупок и экспорт в TXT/PDF, а также формы аутентификации/регистрации и управления профилем (аватар, имя).
2. Серверный слой (Бэкенд): приложение на Python/Django с DRF (REST API) и Djoser (аутентификация), реализующее бизнес-логику: CRUD рецептов, управление тегами и ингредиентами, генерацию коротких ссылок, агрегирование ингредиентов по выбранным рецептам, формирование и выдачу файлового экспорта. Данные хранятся в PostgreSQL. Публикация и статические ресурсы обслуживаются через Nginx, приложение запускается под Gunicorn, развёртывание контейнеризировано с помощью Docker/Docker Compose.

**Функциональная структура**

Отражает модульное устройство веб-приложения и взаимосвязи между его компонентами. Разрабатываемый продукт включает следующие основные функциональные блоки:

* управление учетными записями пользователей: регистрация, аутентификация, смена пароля, редактирование профиля и аватара;
* управление рецептами: создание, редактирование, удаление и просмотр рецептов, а также формирование каталога рецептов с тегами и изображениями;
* взаимодействие пользователей: добавление рецептов в избранное, подписка и отписка от авторов, формирование ленты публикаций;
* список покупок: добавление рецептов в список покупок, агрегирование ингредиентов по выбранным рецептам и экспорт списка в формате .txt или .pdf;
* администрирование: управление тегами и ингредиентами, модерация пользователей и рецептов, ведение журнала событий.

Такое разделение обеспечивает логическую целостность, независимость компонентов и возможность масштабирования системы.

Для графического представления структуры применяются диаграммы IDEF0, IDEF3 и DFD, демонстрирующие потоки данных и взаимосвязи между подсистемами.

Изображение выглядит как снимок экрана, текст, диаграмма

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 1 – IDEF0

Изображение выглядит как диаграмма, снимок экрана

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 2 – IDEF0 декомпозиция

Изображение выглядит как снимок экрана, черный, линия

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 3 – IDEF3

Изображение выглядит как диаграмма, снимок экрана, линия, План

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 4 – Диаграмма потоков данных

**Спецификация процессов**

Поведение системы описывается через основные пользовательские сценарии и взаимодействие между модулями.

Для описания логики применяются диаграммы прецедентов (Use Case), диаграммы последовательностей и блок-схемы процессов (BPMN), отражающие пошаговую работу компонентов.

Пример основного сценария:

* пользователь входит в систему → создаёт рецепт → добавляет ингредиенты и изображение → публикует → другие пользователи просматривают рецепт, добавляют его в избранное или в список покупок → система формирует агрегированный список ингредиентов → пользователь экспортирует его в формате .txt или .pdf.
* Для отдельных операций (например, аутентификация, создание рецепта, агрегация ингредиентов, экспорт) строятся BPMN-схемы, показывающие последовательность шагов, включая валидацию данных и обработку ошибок.
* Для объектов, имеющих жизненный цикл — рецепт, список покупок, профиль пользователя — создаются диаграммы состояний, отображающие переходы: создание → редактирование → публикация → архивирование (или удаление).

Изображение выглядит как диаграмма, текст, линия, шаблон

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 5 – Use Case диаграмма

**Структура хранимой информации**

Данные системы организованы вокруг ключевых сущностей, отражающих логику работы веб-приложения:

* Пользователь: хранит данные учётной записи, аватар, настройки профиля и роль (пользователь или администратор);
* Рецепт: включает автора, название, описание, изображение, время приготовления и связи с ингредиентами и тегами;
* Ингредиент: содержит наименование и единицы измерения;
* Тег: классифицирует рецепты по тематике (например, «Завтрак», «Вегетарианское», «Десерт»);
* Избранное: хранит связи между пользователями и рецептами, добавленными в избранное;
* Подписки: описывает отношения между пользователями (подписчик ↔ автор);
* Список покупок: связывает пользователя с выбранными рецептами и агрегирует ингредиенты для экспорта;
* Короткие ссылки: обеспечивают доступ к рецептам через постоянные уникальные идентификаторы.

Связи между сущностями реализованы в виде отношений:

* один пользователь может иметь множество рецептов;
* один рецепт содержит множество ингредиентов (отношение M:N через промежуточную таблицу);
* пользователь может подписываться на других пользователей и добавлять рецепты в избранное (отношения M:N).

Информация хранится в реляционной базе данных PostgreSQL, спроектированной с использованием ER-диаграммы, отражающей сущности, атрибуты и связи между ними.

Такой подход обеспечивает логическую целостность данных, надёжность хранения и возможность масштабирования при росте объёма контента.

Изображение выглядит как текст, снимок экрана, диаграмма, дизайн

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 6 – ER диаграмма

**Архитектурно-структурное решение**

Разрабатываемая система реализует двухуровневую клиент–серверную архитектуру с чётким разделением пользовательского интерфейса и серверной логики.

Архитектура включает два основных уровня:

1. Клиентский уровень (Frontend/UI)

Представлен веб-интерфейсом, доступным через браузер. Реализует взаимодействие с пользователем:

* + регистрация и авторизация;
  + просмотр и фильтрация рецептов;
  + добавление в избранное, подписка на авторов;
  + формирование и экспорт списка покупок;
  + редактирование профиля и загрузка аватара.

1. Серверный уровень (Backend/API)

Реализован на Python (Django + Django REST Framework) и обеспечивает обработку запросов, работу с базой данных и логику приложения. Основные функции:

* + CRUD-операции с рецептами, ингредиентами и тегами;
  + управление пользователями и правами доступа (Djoser);
  + генерация коротких ссылок и экспорт списка покупок;
  + администрирование и модерация контента.

Данные хранятся в реляционной СУБД PostgreSQL, доступ к которой осуществляется через ORM Django.

Для обеспечения стабильной работы и масштабируемости используется связка Gunicorn + Nginx, а развертывание выполняется в контейнерах Docker / Docker Compose.

Такое архитектурное решение обеспечивает:

* независимость клиентской и серверной частей;
* гибкость масштабирования и обновлений;
* высокую отказоустойчивость и удобство сопровождения.

Изображение выглядит как снимок экрана, диаграмма, линия

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 7 – Диаграмма развертывания

**Интерфейсы**

Пользовательский интерфейс (UI) разработан с учётом потребностей целевой аудитории — домашних кулинаров, авторов рецептов и администраторов системы. Основные принципы, заложенные в архитектуру интерфейса, направлены на простоту, визуальную чистоту и удобство взаимодействия:

1. Фокус на контенте.

Интерфейс минимизирует отвлекающие элементы и акцентирует внимание на главном — рецептах, ингредиентах и фотографиях блюд. Структура страниц интуитивна: карточки рецептов, фильтры по тегам, кнопки «добавить в избранное» и «в список покупок» расположены в логически ожидаемых местах.

1. Прозрачное взаимодействие.

Каждый элемент интерфейса сопровождается визуальной обратной связью — подтверждением действий (добавление, удаление, подписка, экспорт). Пользователь видит результат своих операций мгновенно без перезагрузки страницы благодаря REST API.

1. Прямое манипулирование объектами.

Пользователь может создавать, редактировать и удалять рецепты напрямую из личного кабинета, управлять списком покупок и загружать изображения блюд.

1. Последовательность и логика рабочего процесса.

Весь пользовательский путь выстроен последовательно:

Регистрация / Вход → Просмотр рецептов → Добавление в избранное / подписка → Формирование списка покупок → Экспорт списка в TXT / PDF.

1. Доступность и адаптивность.

Интерфейс адаптирован под различные устройства (ПК, планшеты, смартфоны), обеспечивая одинаково комфортный опыт взаимодействия на всех экранах.

Макет интерфейса главной станицы для авторизованного пользователя представлен на рисунке 8.

Изображение выглядит как снимок экрана, Прямоугольник, диаграмма

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 8 – Интерфейс главной страницы авторизованного пользователя

Макет интерфейса главной станицы для неавторизованного пользователя представлен на рисунке 9.

Изображение выглядит как Прямоугольник, снимок экрана, диаграмма

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 9 – Интерфейс главной страницы неавторизованного пользователя

Макет интерфейса регистрации представлен на рисунке 10.

Изображение выглядит как Прямоугольник

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 10 – Интерфейс регистрации

Макет интерфейса авторизации представлен на рисунке 11.

Изображение выглядит как Прямоугольник, снимок экрана, диаграмма

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 11 – Интерфейс авторизации

Макет интерфейса карточки рецепта представлен на рисунке 12.

Изображение выглядит как текст, снимок экрана, Прямоугольник, диаграмма

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 12 – Интерфейс карточки рецепта

Макет интерфейса просмотра рецепта для авторизованного пользователя представлен на рисунке 13.

Изображение выглядит как чек, текст

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 13 – Интерфейс просмотра рецепта для авторизованного пользователя

Макет интерфейса просмотра рецепта для неавторизованного пользователя представлен на рисунке 14.

Изображение выглядит как чек, текст, линия

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 14 – Интерфейс просмотра рецепта для неавторизованного пользователя

Макет интерфейса страницы 404 представлен на рисунке 15.

Изображение выглядит как Прямоугольник, снимок экрана, диаграмма

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 15 – Интерфейс страницы 404

Макет интерфейса просмотра подписок представлен на рисунке 16.

Изображение выглядит как Прямоугольник, дизайн

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 16 – Интерфейс просмотра подписок

Макет интерфейса просмотра избранного представлен на рисунке 17.

Изображение выглядит как диаграмма, Прямоугольник, снимок экрана, текст

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 17 – Интерфейс просмотра избранного

Макет интерфейса создания рецепта представлен на рисунке 18.

Изображение выглядит как текст, снимок экрана, чек

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 18 – Интерфейс создания рецепта

Макет интерфейса редактирования рецепта представлен на рисунке 19.

Изображение выглядит как текст, снимок экрана

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 19 – Интерфейс редактирования рецепта

Макет интерфейса изменения пароля представлен на рисунке 20.

Изображение выглядит как текст, снимок экрана, Прямоугольник, диаграмма

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 20 – Интерфейс изменения пароля

Макет интерфейса изменения аватара пользователя представлен на рисунке 21.

Изображение выглядит как текст, снимок экрана, круг, диаграмма

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 21 – Интерфейс изменения аватара пользователя