# Разработка Web-блога для любительского кондитерского клуба

# Техническое задание

# Назначение разработки

Приготовление домашних кондитерских изделий – занятное и интересное дело для людей всех возрастов, а также для всех членов семьи. Настоящий многопользовательский сервис в своей web реализации должен предоставить удобный интерфейс для обмена знаниями в кондитерском деле. Основополагающими объектами сервиса должны стать рецепты, как обособленные записи о приготовлении конкретного лакомства, и клиенты, как записи о пользователях блога.

# Цель и задачи разработки

Целью разработки является непосредственное создание многопользовательского блога, как готового продукта.

Задачи, решаемые в процессе разработки:

* определение особенностей предметной области;
* детальное проектирование системы;
* уточнение требований к системе (при необходимости);
* программная реализация системы;
* тестирование итогового продукта;
* создание руководства для пользователя.

# Требования к приложению

## Требования к функциональным характеристикам

Система должна обеспечить хранение данных о пользователях, рецептах кондитерских изделий и прикреплённых к ним изображениях, тегах и ингредиентах, о комментариях к рецептам, оценках, выставленных пользователями к рецептам и комментариям, а также данные стандартных тегов.

Функционал, предоставляемый клиентам приложения (серверные функции):

* регистрация;
* авторизация;
* выход (со сбросом токена авторизации);
* добавление рецепта;
* просмотр данных обо всех рецептах;
* просмотр данных о рецептах:
* по идентификатору;
* по названию;
* по логину автора;
* по тегу.
* редактирование рецепта;
* удаление рецепта;
* просмотр данных авторизованного пользователя;
* просмотр публичной части данных пользователя;
* редактирование данных пользователя (за исключением пароля);
* смена пароля пользователя;
* добавление комментария к рецепту;
* удаление комментария к рецепту;
* добавление/редактирование оценки к рецепту;
* получение оценки к рецепту;
* удаление оценки к рецепту;
* добавление/редактирование оценки к комментарию;
* получение оценки к комментарию;
* удаление оценки к комментарию;

## Модель данных системы

База данных системы должна соответствовать инфологической модели, представленной на рисунке 1. Ниже приведено описание таблиц базы данных.

Таблица **client** - данные пользователя.

* id\_client - идентификатор (PK), целочисленное, автоматически генерируемое значение;
* username - логин, символьная строка;
* first\_name - имя, символьная строка;
* last\_name - фамилия, символьная строка;
* email - электронная почта, символьная строка;
* password - пароль, символьная строка;
* avatar - системный путь к аватар-изображению на сервере, символьная строка;
* is\_staff - возможность использовать администраторское приложение, булево значение;
* is\_active - возможность выполнять вход в систему, булево значение;
* is\_superuser - возможность использовать все привилегии сразу, булево значение;
* last\_login - дата и время последнего присоединения, дата со временем;
* date\_joined - дата и время создания записи, дата со временем.
* patronymic - отчество, символьная строка;

Участвует в отношении многие-ко-многим с моделями django.contrib.auth.models.Group и django.contrib.auth.models.Permission, что позволяет реализовать разделение пользователей на группы и выдачу им привилегий.

Таблица **client\_avatar** - данные аватар-изображения пользователя.

* client - пользователь, которому принадлежит аватарка (FK), целочисленное значение;
* id\_cavatar - идентификатор (PK), целочисленное, автоматически генерируемое значение;
* path - системный путь к аватар-изображению на сервере, символьная строка.
* format - расширение файла изображения.

Таблица **recipe** - данные о выкладываемом рецепте:

* creator - пользователь, выложивший рецепт (FK), целочисленное значение;
* id\_recipe - идентификатор (PK), целочисленное, автоматически генерируемое значение;
* title - название рецепта, символьная строка;
* portions - количество порций, целочисленное;
* time\_cook - время приготовления в минутах, целочисленное;
* weight - вес, целочисленное;
* avatar - системный путь к аватар-изображению на сервере, символьная строка;
* status - текущий статус рецепта, короткое целочисленное;
* date\_init - дата добавления, дата.

Таблица **ingredient** – ингредиент рецепта.

* recipe - рецепт, за которым закрепляется ингредиент (FK), целочисленное значение;
* id\_ingredient - идентификатор (PK), целочисленное, автоматически генерируемое значение;
* name - наименование, символьная строка;
* measure - мера ингредиента, символьная строка.

Таблица **cook\_stage** – стадия приготовления рецепта.

* recipe - рецепт, за которым закрепляется ингредиент (FK), целочисленное значение;
* id\_cookstage - идентификатор (PK), целочисленное, автоматически генерируемое значение;
* description - описание стадии приготовления, текст;
* picture - системный путь к изображению стадии приготовления на сервере, символьная строка.

Таблица **comment** - данные о комментариях к рецепту:

* creator - пользователь, выложивший рецепт (FK), целочисленное значение;
* recipe - рецепт, к которому написан комментарий (FK), целочисленное значение;
* id\_comment - идентификатор (PK), целочисленное, автоматически генерируемое значение;
* body - основное содержание комментария, текст;
* status - текущий статус комментария, короткое целочисленное значение;
* date\_init - дата добавления, дата.

Таблица **tag** – данные о теге:

* recipe - рецепт, за которым закрепляется тег (FK), целочисленное значение;
* id\_tag - идентификатор (PK), целочисленное, автоматически генерируемое значение;
* name - наименование, символьная строка.

Таблица **comment\_grade** – данные об оценивании комментариев пользователями:

* evaluator - пользователь, оценивающий комментарий (FK), целочисленное значение;
* comment - оцениваемый пользователем комментарий (FK), целочисленное значение;
* grade - значение оценки, короткое целочисленное значение.
* id\_comgrade - идентификатор (PK), целочисленное, автоматически генерируемое значение.

Таблица **recipe\_grade** – данные об оценивании комментариев пользователями:

* evaluator - пользователь, оценивающий рецепт (FK), целочисленное значение;
* recipe - оцениваемый пользователем рецепт (FK), целочисленное значение;
* grade - значение оценки, короткое целочисленное значение.
* id\_recgrade - идентификатор (PK), целочисленное, автоматически генерируемое значение.

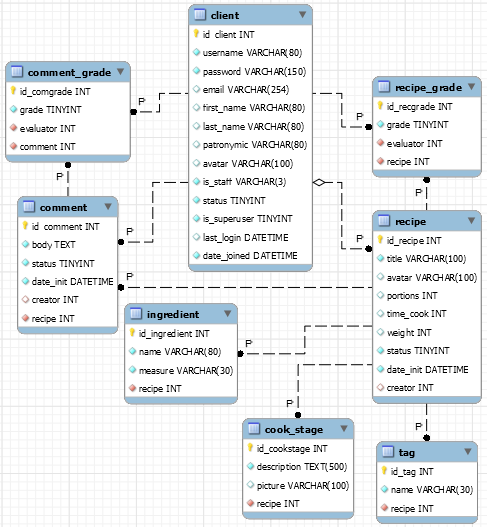


Рисунок 1 – Инфологическая модель базы данных

## Структура пользовательского интерфейса приложения

**Главная страница**

Общая «обёртка» для остальных компонентов

Все описанные далее страницы содержат toolbar и footer, характерные для главной страницы. Toolbar содержит логотип и кнопку «на главную» (оба элемента обеспечивают переход на главную страницу из любой другой), а также нижеописанные выпадающие меню «мои рецепты» и «мой кабинет», чувствительные к авторизованности пользователя.

Для неавторизованного пользователя:

Состоит из строки поиска рецептов (с радиокнопками «по названию», «по тегам», «по авторам»), выпадающего меню «мой кабинет» с кнопками «войти» (на страницу авторизации) и «регистрация» (на страницу регистрации), выпадающего меню «мои рецепты» с кнопками «список рецептов» (выдаётся сообщение с просьбой авторизоваться) и «добавить рецепт» (выдаётся сообщение с просьбой авторизоваться), общего списка рецептов (с фильтром по умолчанию «по рейтингу ↓»). Имеется выпадающий список для смены фильтра («по рейтингу ↓»/«по рейтингу ↑»/«по дате создания ↓»/«по дате создания ↑»/«в случайном порядке»).

Для авторизованного пользователя:

Состоит из строки поиска рецептов (с радиокнопками «по названию», «по тегам», «по авторам»), выпадающего меню «мой кабинет» с кнопками «мой профиль» (на страницу данных пользователя для владельца страницы) и «выйти» (на главную страницу для неавторизованного пользователя), выпадающего меню «мои рецепты» с кнопками «список рецептов» (общий список карточек рецептов (данные: аватар, наименование, рейтинг) для текущего пользователя с возможностью перехода на страницу конкретного рецепта для создателя рецепта) и «добавить рецепт» (на страницу создания рецепта), общего списка рецептов (с фильтром по умолчанию «по рейтингу ↓»). Имеется выпадающий список для смены фильтра («по рейтингу ↓»/«по рейтингу ↑»/«по дате создания ↓»/« по дате создания ↑»/«в случайном порядке»).

Общее:

* Общий писок рецептов оформлен страницами, по 15 карточек рецептов (данные: аватар, наименование, логин автора, рейтинг).
* При нажатии на карточку рецепта открывается страница рецепта (если пользователь авторизован, и рецепт принадлежит ему, то открывается страница рецепта для создателя рецепта, иначе – для обычного пользователя).
* При нажатии на логин автора открывается страница данных пользователя (если пользователь авторизован, и рецепт принадлежит ему, то открывается страница данных пользователя для владельца страницы, иначе – для обычного пользователя).
* Поиск осуществляется по всем активным (проверка статуса активности на сервере) на данный момент рецептам на сайте. При поиске по авторам учитывается статус активности автора. Результаты выводятся внутрь общего списка рецептов.

**Страница регистрации**

Страница с формой регистрации. Поля: логин\*, пароль\*, почта\*, Ф, И, О. Кнопки: «зарегистрироваться». При заполнении и подтверждении формы происходит проверка данных по существующим пользователям с идентичным логином или почтой. В случае прохождения проверки создаётся запись о новом пользователе и происходит переход на главную страницу для неавторизованного пользователя (для авторизации необходимо дополнительно открыть форму авторизации), иначе - выдаётся сообщение с просьбой подкорректировать необходимые данные формы.

**Страница авторизации**

Страница с формой авторизации. Поля: логин/почта\* (возможна авторизация по любому из полей), пароль\*. Кнопки: «войти». В случае прохождения проверки откроется главная страница для авторизованного пользователя с уведомлением об успешной авторизации, в противном случае выдаётся сообщение с просьбой подкорректировать необходимые данные формы.

**Страница данных пользователя**

Для обычного пользователя:

Страница, которая содержит следующие данные: аватар, логин, Ф, И, О, статус активности, дата создания записи, дата последнего вхождения в систему.

Для владельца страницы:

Страница, которая содержит следующие данные: аватар, логин, Ф, И, О, адрес электронной почты. Данные представлены заполненной формой и предоставляют функционал для редактирования. Имеется кнопка «применить», которая становится активной в случае изменения данных. В случае прохождения проверки редактируется запись о настоящем пользователе и происходит переход на главную страницу для авторизованного пользователя.

Общее:

* Авторизованный пользователь не имеет возможности открыть собственную страницу данных пользователя, как страницу данных пользователя для обычного пользователя (только как для владельца страницы).

**Страница рецепта**

Для обычного пользователя:

На странице отображены следующие данные: аватар рецепта, наименование, логин создателя, дата создания, рейтинг, описания этапов приготовления с изображениями, теги, комментарии к рецепту.

Имеется кнопки для оценивания рецепта (+/-, влияет на формирование рейтинга рецепта) и добавления комментария (открывается форма по созданию комментария (без перехода на другую страницу), содержащая поле содержимого комментария и кнопок «отправить» и «отмена»; случае отправки, создаётся запись о комментарии). Если пользователь не авторизован, то выводится сообщение с просьбой авторизоваться.

Комментарии оформлены снизу от рецепта, в качестве списка карточек (данные: аватар и логин автора, текст комментария, рейтинг; имеется кнопка для оценивания комментария (подобно оцениванию рецепта)) в обратном хронологическом порядке по дате создания. Сначала отображены первые 10 комментариев, далее присутствует кнопка для загрузки всех остальных имеющихся комментариев.

Для создателя рецепта:

Содержит те же данные и возможности, что и страница рецепта для обычного пользователя, однако данные (аватар рецепта, наименование, описание, прикреплённые изображения и теги) представлены заполненной формой и представляют функционал для редактирования. Имеется кнопка «применить», которая становится активной в случае изменения данных. В случае прохождения проверки редактируется запись о настоящем рецепте и происходит переход на главную страницу для авторизованного пользователя.

Общее:

* Авторизованный пользователь не имеет возможности открыть собственный рецепт, как страницу рецепта для обычного пользователя (только как для создателя рецепта, даже через открытие страницы рецепта через главную страницу).

**Страница создания рецепта**

**Содержит пустую форму с полями:** аватар рецепта, наименование, описание, прикреплённые изображения и теги; имеются кнопки: «добавить рецепт», «отмена» (на главную страницу). В случае прохождения проверок добавляются записи о рецепте и прикреплённых к нему данных, происходит переход на главную страницу и выводится сообщение об успешности добавления рецепта, иначе – выводится сообщение с просьбой скорректировать данные формы.

## Эргономические и технико-эстетические требования

Взаимодействие пользователей с функционалом приложения должно осуществляться через удобный и понятный графический интерфейс, реализованный с использованием стандартных графических элементов web-приложений (кнопки, радиокнопки, чекбоксы, комбобоксы, изображения, формы, текстовые поля и блоки и т.п.), выполненными в единой стилистике.

Интерфейс приложения должен быть логически распределён между пунктами меню, расположенного в тулбаре, и соответствующими страницами, не должен быть перегружен графическими элементами. Переход между окнами должен осуществляться без задержек.

Требования к оформлению графических элементов:

* ограниченная цветовая палитра: сиреневый, тёмно-сиреневый, бирюзовый;
* использование унифицированных графических элементов, связанных с выполнением сходных задач;
* повсеместное соблюдение общих пропорций размеров и расположения элементов;
* отсутствие теней у элементов.

Требования к оформлению текстовых элементов:

* использование кодировки текста UTF-8;
* полная русификация текста всех управляющих элементов и сообщений;
* использование шрифтов без засечек (Arial, Bahnshrift, Trebuchet)
* ограниченная цветовая палитра: чёрный, белый, сиреневый (возможно расширение палитры для особых текстовых блоков).

Серверное приложение должно обеспечивать безопасное и полноценное взаимодействие с функционалом, предоставляемым операционной системой и СУБД, и корректную обработку всех типов аварийных ситуаций (логических и арифметических ошибок на уровне переменных программы, ошибок логики работы приложения, ошибок взаимодействия с операционной системой и т.д.) с отправкой при необходимости соответствующих текстовых уведомлений на сторону клиентского приложения.

Клиентское приложение должно обеспечивать понятный и удобный графический интерфейс пользователя с корректной обработку всех типов аварийных ситуаций (как появляющихся в результате работы клиентского приложения, так и зависящих от работы серверного приложения) с отправкой при необходимости соответствующих уведомлений конечному пользователю.

В любой момент взаимодействия с клиентским приложением у конечного пользователя не должно возникать дискомфорта, связанного с непониманием состояния, в котором находится система или конкретные элементы графического интерфейса. Необходимо обеспечить вывод соответствующих уведомлений и оповещений, интуитивную понятность логики переходов между элементами приложения и логики состояний графических элементов.

## Требования к применению систем управления базами данных

Применяемая в разработке система управления базами данных должна удовлетворять следующим требованиям:

* реляционная модель данных;
* поддержка стандарта SQL не менее SQL 3 (1999);
* наличие встроенных средств:
* защиты данных от несанкционированного доступа;
* разграничения параллельного многопользовательского доступа;
* резервного копирования данных.
* поддержка стандарта SQL не менее SQL 3 (1999);
* быстродействие при работе с данными;
* свободная лицензия.

## Требования к защите информации от несанкционированного доступа

Для конечных пользователей необходимо обеспечить идентификацию (авторизацию) на уровне серверного приложения и установить соответствующие ограничение на доступ к данным. Смена пароля пользователя должна производиться с использованием отдельной формы, в которой пользователю необходимо подтвердить текущий пароль личного аккаунта. При вводе пароля пользователем содержимое поля должно быть сокрыто.

## Требования к сохранности информации при авариях

На стороне сервера:

Должно быть предусмотрено ручное создание резервной копии базы данных в облачном хранилище один раз в неделю. В случае аварийной ситуации программное обеспечение должно полностью восстанавливать свою функциональность с использованием актуальной копии. В облачном хранилище достаточно хранить две последние копии.

На стороне конечного пользователя:

Требования к сохранности сформированных, но не отправленных и не сохранённых локально данных в случае аварийных ситуаций не предъявляются.

## Требования к технологиям разработки

При разработке системы должны использоваться технология реляционных баз данных, процедурная и объектно-ориентированная парадигмы программирования. Система должна быть реализована в соответствии с трёхзвенной моделью архитектуры «клиент-сервер» (Web-браузер на стороне клиента, клиентское и серверное приложения на обособленных ПК или серверных станциях).

## Требования к составу и конфигурации программных средств

При разработке и использовании компонентов системы на рабочих станция должны быть установлены соответствующие лицензированные программные средства:

1. ПК для разработки серверного приложения:

* Python 3.7 и пакет необходимых библиотек и фреймворков:
* asgiref 3.2.10;
* Django 3.0.3;
* django-cors-headers 3.5.0;
* djangorestframework 3.11.1;
* djangorestframework-jsonapi 3.2.0;
* djangorestframework-jwt 1.11.0;
* djoser 2.0.5;
* inflection 0.5.1;
* mysql 0.0.2;
* mysqlclient 2.0.1;
* PyJWT 1.7.1;
* pytz 2020.1;
* sqlparse 0.3.1.
* IDE PyCharm 2019.3.2;
* MySQL Server 8.0;
* MySQL Workbench 8.0 C;
* Postman.

1. Компьютер, выполняющий роль сервера :

* Python 3.7 и описанный ранее пакет необходимых библиотек и фреймворков;
* MySQL Server 8.0.

1. ПК для разработки клиентского приложения:

* NodeJS и пакет необходимых библиотек и фреймворков:
* react;
* react-router-dom;
* react-redux;
* react-images-uploading;
* redux-devtools-extension;
* axios-react-redux;
* redux-thunk;
* react-dom.
* IDE WebStorm 2020.1.1;
* Postman.

1. ПК для разработки дизайна клиентского приложения:

* Adobe Photoshop CC;
* Adobe Illustrator CС;
* Adobe XD CC.

1. ПК конечного пользователя:

* Web-браузер, способный корректно отображать используемые элементы графического интерфейса.

# Требования к методам тестирования

Начальный этап работ по тестированию работоспособности системы должен быть представлен тестированием модульного и интеграционного уровней методом «белого ящика» на этапе разработки программных компонентов системы. На этом этапе необходимо определить корректность и полноценность работы каждого программного модуля в отдельности и в составе локальных функциональных объединений.

На втором этапе, после объединения программных модулей системы, необходимо обеспечить полное функциональное тестирование системного уровня методом «белого ящика», в том числе с целью выявления ошибок переходов между состояниями системы, недочётов в работе её временных зависимостей. Дополнительно необходимо воспользоваться помощью стороннего эксперта, задачей которого будет дублирование испытаний второго этапа без знания структуры системы (методом «чёрного ящика»).

На завершающем этапе тестирования работы системы необходимо воспользоваться помощью закрытого круга независимых тестировщиков, обладающих различным уровнем навыков в тестировании и выступающих в роли конечных пользователей (без знания структуры продукта), с целью определения удобства интерфейса приложения. Дополнительной задачей настоящего этапа должно стать подвержение системы различным сценариям агрессивного поведения пользователей для определения невыявленных до этого уязвимостей.