МИНОБРНАУКИ РОССИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

(ФГБОУ ВО «ВГУ»)

Факультет Компьютерных наук

Кафедра программирования и информационных технологий

Разработка Web-приложения «CookBook»

Курсовая работа

09.03.04 – Программная инженерия

Зав. кафедрой _____ д-р физ.-мат. наук, профессор С.Д. Махортов

Выполнили _____ Е.С. Боровкова и К.И. Крылова

Руководитель _____ ассистент В.С. Тарасов

Воронеж 2019

Оглавление

Введение

В современном мире очень высокий темп жизни. Иногда человек забывает зайти в магазин за продуктами, иногда ему просто лень делать это из-за сильной усталости, но, придя домой, он понимает, что необходимо поесть, а чтобы поесть, надо сначала что-то приготовить и желательно из тех продуктов, что имеются в холодильнике. Также очень часто лень включать фантазию, какое блюдо можно приготовить из доступных продуктов.

Сервис, позволяющий пользователю вводить имеющиеся ингредиенты и выводящий список блюд, содержащих эти продукты вместе с пошаговым рецептом приготовления, актуален на сегодняшний день, так как многие процессы стараются автоматизировать, и автоматизация идей для приготовления блюд значительно бы упростила жизнь многим людям. Так же данную предметную область можно рассмотреть с научной точки зрения. Было бы полезно отследить статистику продуктов, которые вводят пользователи. С помощью этой статистики можно выяснить, какие продукты являются одними из самых распространенных и повседневных в нашей жизни, чтобы особо тщательно контролировать их качество и придумать как можно больше новых блюд, содержащих эти продукты.

Поэтому целью работы является создание такого сервиса, который позволит пользователю ввести ингредиенты, которые у него имеются, и система выведет список блюд, содержащих эти продукты вместе с пошаговым рецептом приготовления. Если в блюде все же имеются ингредиенты, которых нет в наличии, то система выдаст подсказку, что необходимо докупить.

1. Постановка задачи

Цель: разработать приложение для поиска блюд по имеющимся ингредиентам, которое позволит:

1. Уменьшить время на придумывание приготовления блюда.
2. Видеть альтернативы приготовления блюд из имеющихся ингредиентов.
3. Получать автоматически сгенерированный список необходимых продуктов для покупки.

Сфера использования: повседневная жизнь.

Задачи:

1. Реализовать просмотр всех таблиц.
2. Реализовать добавление, удаление, редактирование информации в таблицах.
3. Реализовать поиск блюд по ингредиентам.
4. Реализовать подсказку по недостающим ингредиентам.

2. Анализ предметной области

Блюдо – комбинация доведённых до готовности пищевых продуктов, которая кулинарно оформлена в виде порции и готова к употреблению.

Рецепт – руководство по приготовлению кулинарного изделия.

Ингредиент – составная часть чего-то (вещества, смеси).

Были проанализированы существующие приложения с кулинарными рецептами в целях выявления их недочетов. В ходе анализа были выявлены следующие минусы:

1. На сайте нет поиска блюда по ингредиентам совсем.
2. Поиск по ингредиентам есть, но результаты выводятся без учета продуктов, которых пользователю не хватает для приготовления найденного блюда.

Примеры сайтов, похожих по содержанию:

1. http://recepty-po-ingredientam.ru/
2. https://www.povarenok.ru/

В данной работе будут учтены недостатки существующих сервисов. При поиске по желательным ингредиентам пользователь будет получать подсказки, какие продукты придется докупить, причем вывод блюд будет в порядке возрастания недостающих продуктов. Это должно упростить поиск доступного для пользователя блюда.

2.1 Аппаратные и программные требования

Данный сервис является Web-приложением. Выбора пал именно на Web, так как оно имеет ряд плюсов:

1. Пользователь не нуждается в установке на свою машину тяжелого программного обеспечения. Все, что требуется для полноценной работы –

это браузер, обычно поставляемый вместе с операционной системой, и доступ в Интернет.

2. По сути в роли администраторов web-приложений выступают разработчики, которые работают в одном месте. В корпоративном секторе, например, это экономически гораздо выгоднее и эффективнее, чем содержание команды программистов и админов, занимающихся установкой и настройкой десктопных приложений на машинах пользователей.
3. Web-приложения не требовательны к ресурсам и не предъявляют никаких требований к аппаратной платформе. Это значит, что нет никакой разницы, сколько мегабайт оперативной памяти установлено на компьютере пользователя и из-под какой операционной системы он работает.
4. Нет проблем с поддержкой старых версий программ и обратной совместимостью. Когда появляется новая версия десктопного приложения, пользователям нередко приходится решать проблемы, связанные с обновлением уже установленной на их машине копии. В случае с браузерными приложениями таких проблем не возникает – существует только одна версия, в которой работают все пользователи, и в случае выхода новой все без исключения автоматически переходят на нее, порой даже не замечая этого.
5. Web-приложения позволяют своим пользователям быть по-настоящему мобильными. По сути, вы можете работать в сети, сохранять результаты своей работы на сервере и, в случае необходимости, иметь к ним доступ отовсюду.

Но существует и ряд минусов:

1. Интернет доступен не везде.
2. Существует огромное количество приложений, которые не могут быть заменены браузерными.

3. Многих пользователей смущает тот факт, что их данные будут храниться и обрабатываться где-то на чужом сервере. Ведь потенциально это может привести к утечке, потере или искажению информации.

Back-end реализован с помощью базы данных MySql, языка программирования Python и библиотек:

1. Flask
2. SQLAlchemy
3. Flask-SQLAlchemy
4. WTForms
5. WTForms_alchemy

2.2 Отличие Python и PHP

PHP и Python – это два наиболее популярных решения, которые выбирают компании и разработчики. Несмотря на то, что огромное количество веб-сайтов запрограммировано на PHP, в последние два года наблюдается резкая тенденция к росту разработки на Python. Оба языка имеют свои преимущества и недостатки, кроме некоторых отличительных особенностей. Тем не менее, в недавнем прошлом все большее число людей перешло на Python из PHP, благодаря нескольким преимуществам, таким как надежность, улучшенный дизайн и удобочитаемость, и это лишь некоторые достоинства, которые предоставляет язык программирования Python. Данное приложение написано на языке Python по ряду причин:

1. Продуманный дизайн. Гораздо проще использовать Python для написания отличного кода, чем PHP.
2. Python имеет архитектуру, благодаря которой его можно назвать продуманным и надежным языком. PHP не обладает такими качествами.
3. PHP не такой элегантный язык, как Python, хотя он полностью работоспособен. Любой разработчик, имеющий достаточный опыт создания

отличного программного обеспечения, может написать хороший скрипт на PHP. Однако, чтобы быть более гибким с PHP, требуется очень глубокое знание нюансов и особенностей языка.

4. В простоте обучения Python выигрывает. Python выигрывает у PHP не только потому, что его легко освоить, но и благодаря огромному количеству учебных пособий, доступных онлайн.
5. Python более читабельный стек. PHP широко документирован и придерживается классического подхода. С другой стороны, Python использует довольно строгие требования к отступам. Можно утверждать, что Python более читабелен не только по сравнению с PHP, но и с большинством других языков программирования. Сама философия Python—читабельность кода.
6. Синтаксис намного проще, чем PHP. Код Python проще для понимания и написания. Он не содержит фигурных скобок, как другие языки программирования, и его легко понять.
7. Более простые и легкодоступные инструменты отладки. Python имеет большую экосистему для разработчиков и легко доступные инструменты отладки для языка. Использовать эти инструменты довольно просто. Стек предоставляет разработчикам Python Debugger (PDB), довольно мощный отладчик, который прост в использовании и настолько хорошо документирован, что даже новички могут его понять. PHP также предлагает пакет отладчика под названием XDebug, который великолепен. Единственное преимущество, с которым Python побеждает, состоит в том, что ему требуется меньше средств отладки, чем PHP.
8. Явный победитель в управлении пакетами. Управление пакетами служит связующим звеном между различными проектами. С его помощью вы можете писать, создавать и обмениваться пакетами в таком формате, чтобы другие разработчики могли легко подключаться к другим приложениям. Управление пакетами существует в PHP, но, вероятно, нет такой базы кода,

который используется в той степени, в которой это делает PIP (инструмент для установки и управления пакетами Python).

9. Лямбды, предоставляемые Python, дают преимущество перед PHP. Лямбда (Lambdas)—это подпрограмма или функция, которая определяется и вызывается без привязки к идентификатору. Это блок кода, который можно передать и выполнить позже, один или несколько раз. Используя лямбда-выражения можно объявлять функции в любом месте кода, они просты в создании.
10. Python более универсален, чем PHP. Python—это универсальный язык программирования, практически бесконечный. Разработка веб-сайтов на Python—это не единственное, на что способен стек. Машинное обучение, NLP, наука о данных, обработка изображений, а также разработка настольных и мобильных приложений—это лишь несколько примеров использования Python. PHP может использоваться для других целей, но он предназначен именно для создания веб-страниц, и это то, что он делает лучше всего. Это сложный язык программирования, предназначенный для создания сложных веб-программ.
11. Python помогает создавать более надежный код, который можно использовать в различных сферах.

2.3 Выбор СУБД

PostgreSQL и MySQL – это две популярные БД с открытым исходным кодом. Они поддерживают все основные операции SQL, которые нужны пользователю. Когда дело доходит до более продвинутых задач, необходимо воспользоваться определенными функциями, поддерживаемыми каждой БД, такими как материализованные представления или частичные индексы. Например, PostgreSQL поддерживает материализованные представления, а MySQL - нет.

Если необходимо создать приложение, ориентированное на потребителя, целью которого является масштабирование с более миллионом активных пользователей – MySQL лучший выбор. Можно рассмотреть следующие аспекты сравнения PostgreSQL vs MySQL:

1. С точки зрения надежности:

   В PostgreSQL 10 добавили логическую репликацию, что делает ее эквивалентной MySQL. Исторически репликация была одной из причин выбора второй БД, но теперь по этому критерию первая с ней сравнялась. Обе базы данных имеют параметры, позволяющие соотносить производительность и долговечность MySQL vs PostgreSQL для веб- приложений.
2. С точки зрения скорости:

   Для приложений с длительным временем выполнения SELECT для аналитики, PostgreSQL работает лучше благодаря возможности параллельного запроса. Для небольших SELECT, охватывающих простой и кластеризованный индекс, отлично работает MySQL. Для приложений с большим количеством малых MySQL лучше подходит. Для приложений с тяжелыми UPDATE MySQL работает намного лучше. Для приложений с тяжелыми DELETE на эфемерных данных обе поддерживают раздел, обе работают хорошо, если пользователь тщательно использует эту функцию.
3. С точки зрения масштабируемости:

   Обе БД масштабируются довольно хорошо (вверх и вниз). PostgreSQL более скромная. Если у пользователя 1000 подключений, ей нужно более 10 ГБ дополнительной памяти.

Но все же MySQL обладает большим количеством преимуществ перед другими системами:

1. СУБД MySQL является одной из самых быстрых баз данных среди имеющихся на современном рынке.

2. СУБД MySQL является высокопроизводительной и относительно простой в использовании СУБД, которую значительно проще инсталлировать и администрировать, чем многие другие большие системы.
3. СУБД MySQL распространяется бесплатно для домашнего использования.
4. Сервер позволяет подключаться одновременно неограниченному количествльзователей. Доступ к серверу можно осуществить в интерактивном режиме с помощью различных интерфейсов, позволяющих вводить запросы и просматривать полученные результаты: это программы-клиенты, работающие с командной строкой, Web-браузеры, программы-клиенты, работающие в системе Windows, это, наконец, программные интерфейсы для языков C, Perl, Java, PHP и Python. Так что можно использовать как готовое клиентское программное обеспечение, так и создавать свое собственное.
5. MySQL предназначена для работы в сети, может быть доступна через Интернет, но при этом снабжена развитой системой защиты от несанкционированного доступа.
6. Дистрибутив СУБД MySQL можно получить, воспользовавшись Web-броузером. Что самое важное – можно получить и исходный код и внести в него коррективы.

2.4 Отличие Flask и Django

Flask и Django – два самых популярных веб-фреймворка для Python. Ключевая разница между Flask и Django это:

1. Flask реализуется с минимальными надстройками, которые всецело предоставлены аддонам или разработчику. Flask предоставляет простоту, гибкость и аккуратность в работе, позволяя пользователю самому выбирать, как реализовать те или иные вещи.

2. Django следует философии «все включено», и дает большой ассортимент для работы: есть панель админа, интерфейсы баз данных, ORM, и структура каталогов для ваших приложений и проектов.

Если рассматривать более детально, то можно выявить следующие различия:

1. Архитектура

   Flask:

   Нет жёсткой структуры. Можно поселить модели, контроллеры и инициализацию в одном файле, можно - в разных. Изначально проект состоит из пустой папки.

   Django:

   Состоит из проекта, который делится на приложения. Проект – корневая папка и глобальные настройки. Приложения – это функционал сайта, разбитый по разным модулям. У каждого проекта они свои. Примеры приложений: users отвечает за поведение и модели пользователей, API предоставляет методы для внешнего взаимодействия с сайтом и т.п. С непривычки такая система может показаться непонятной и сложной.
2. Расширение

   Flask:

   Устанавливается практически пустым, но легко расширяется сторонними батарейками. Обычно для решения одной задачи есть несколько батареек: выбор зависит от размера команды разработчиков и нагрузки на приложение.

   Django:

   Также можно установить сторонние батарейки. Однако в ней заменить что-то встроенное будет уже сложно. Поэтому иногда говорят, что в джанге всё приколочено гвоздями. Она годится для стандартного набора задач и если с реальностью это не сходится, от джанги надо уходить.

2.5 Object-Relational Mapping

ORM (англ. object-relational mapping, рус. объектно-реляционное отображение) — технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных». Существуют как проприетарные, так и свободные реализации этой технологии.

SQLAlchemy – это библиотека на языке Python для работы с реляционными СУБД с применением технологии ORM. Служит для синхронизации объектов Python и записей реляционной базы данных. SQLAlchemy позволяет описывать структуры баз данных и способы взаимодействия с ними на языке Python без использования SQL.

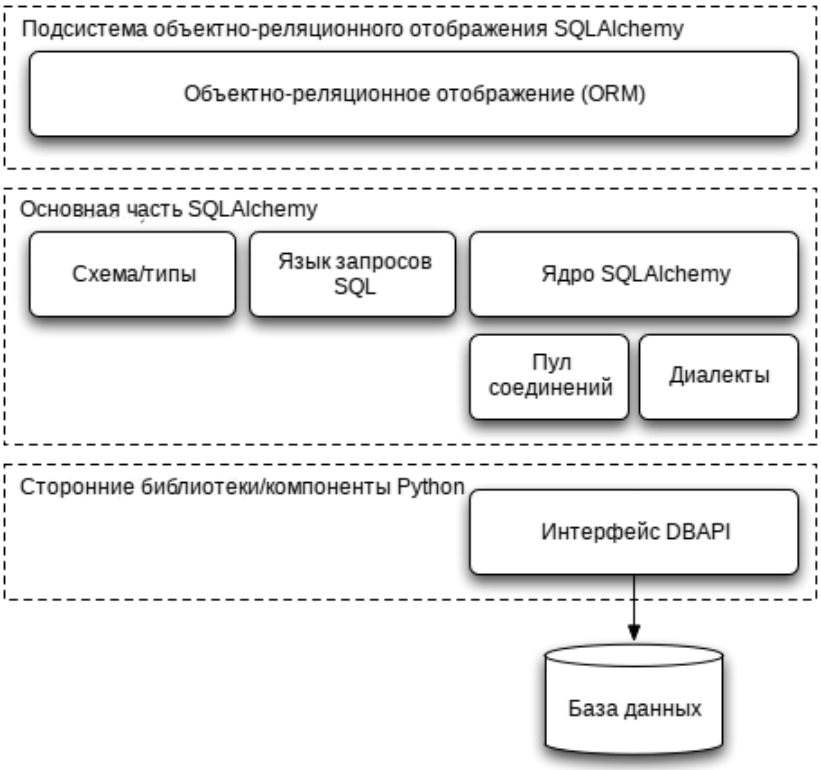

Рисунок 1 – Диаграмма уровней SQLAlchemy

Использование SQLAlchemy для автоматической генерации SQL-кода имеет несколько преимуществ по сравнению с ручным написанием SQL:

1. Безопасность. Параметры запросов экранируются, что делает атаки типа внедрение SQL-кода маловероятными.
2. Производительность. Повышается вероятность повторного использования запроса к серверу базы данных, что может позволить ему в некоторых случаях применить повторно план выполнения запроса.
3. Переносимость. SQLAlchemy, при должном подходе, позволяет писать код на Python, совместимый с несколькими back-end СУБД. Несмотря на стандартизацию языка SQL, между базами данных имеются различия в его реализации, абстрагироваться от которых и помогает SQLAlchemy.

Отличие SQLAlchemy от peewee:

1. Алхимия может строить совсем сложные запросы, гибкая
2. Peewee – это небольшое ORM, которое в данный момент поддерживает postgresql, mysql и sqlite. Хорошая сторона Peewee – это то, что он занимает мало места, его легко освоить, и можно приступить к работе с приложениями за несколько минут.

Выбор был сделан в пользу SQLAlchemy, так как:

1. Она позволяет создавать более сложные запросы.
2. Под нее существует библиотека WTForms-Alchemy, которая предоставляет вспомогательный класс, позволяющий создавать класс Form из модели SQLAlchemy.
3. Можно свободно перейти с одной БД на другую, если это потребуется.

3. Реализация

3.1 Описание логики приложения с помощью UML-диаграмм

1. Диаграмма классов

   Через класс Service осуществляется взаимодействие с другими классами: вывод, редактирование таблиц и поиск по таблицам. Остальные классы отправляют запросы к базе данных на вывод или изменение соответствующей таблицы.

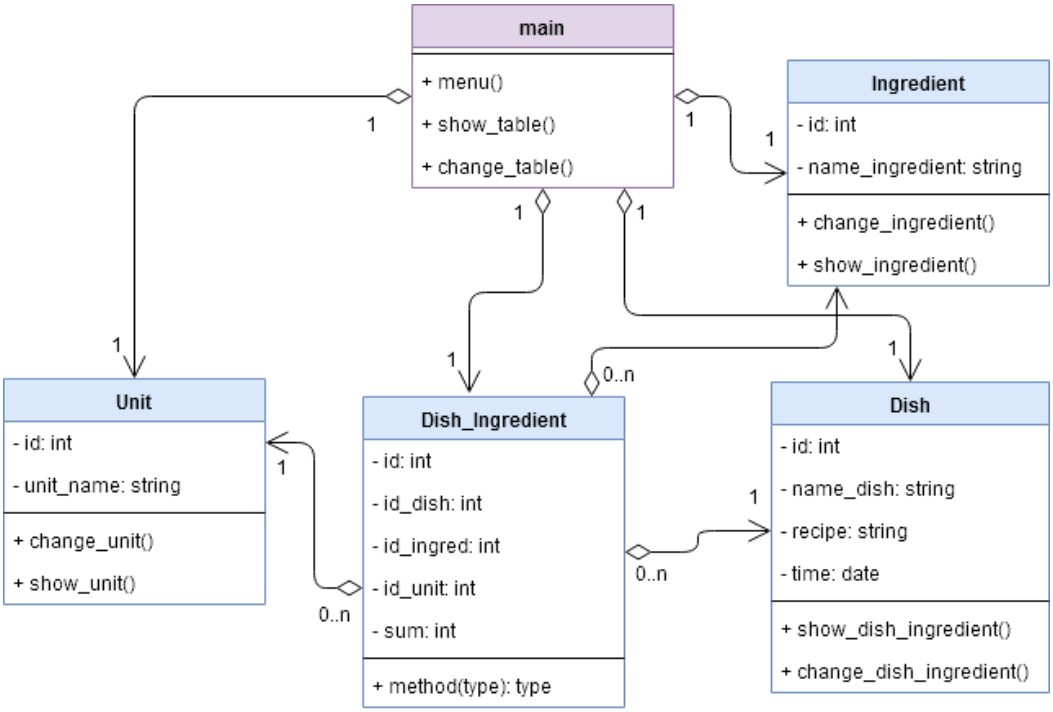

Рисунок 2 – Диаграмма классов

2. Диаграмма прецедентов (Use-case)

   Пользователь может выполнять следующие действия над системой: редактировать, удалять, добавлять записи в любую таблицу и осуществлять поиск по всем таблицам.

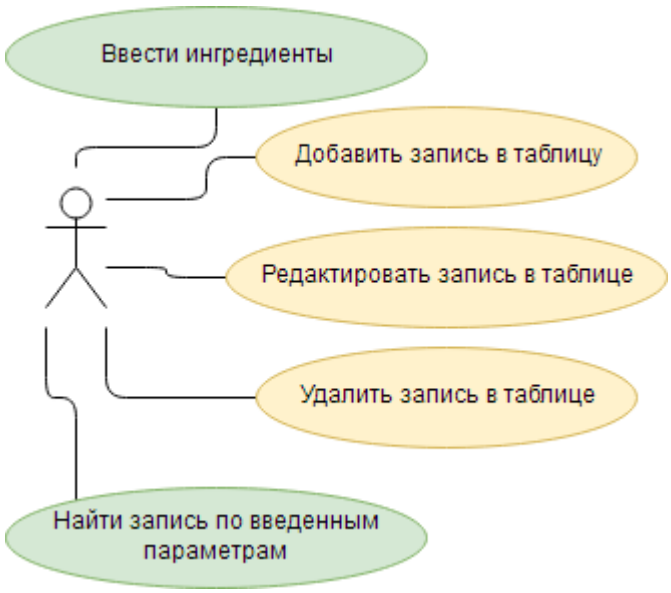

Рисунок 3 – Use-Case

3. Диаграмма объектов

В определенный момент времени в системе имеются следующие экземпляры объектов: пользователь, осуществляющий поиск по таблице Dish_Ingredient по ингредиенту “картошка”. Система вывела ответ на запрос пользователя с помощью класса Dish в виде блюда с рецептом “мясо по французски”. Единицы измерения картофеля - “кг”.

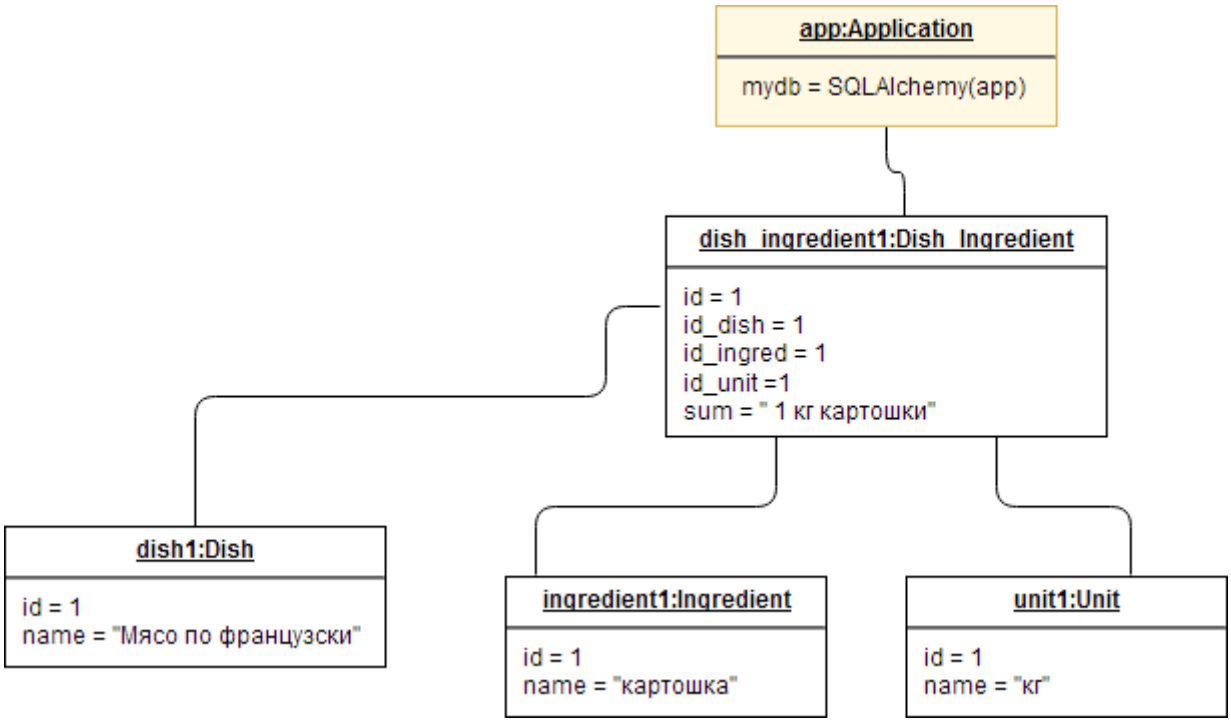

Рисунок 4 – Диаграмма объектов

4. Диаграмма последовательностей
    1. Пользователь вводит ингредиенты и нажимает “Поиск”.
    2. Система отправляет данные на сервер.
    3. Формируется запрос к БД.
    4. Сервер получает ответ и отправляет информацию сервису.
    5. Выводится таблица блюд с рецептами.

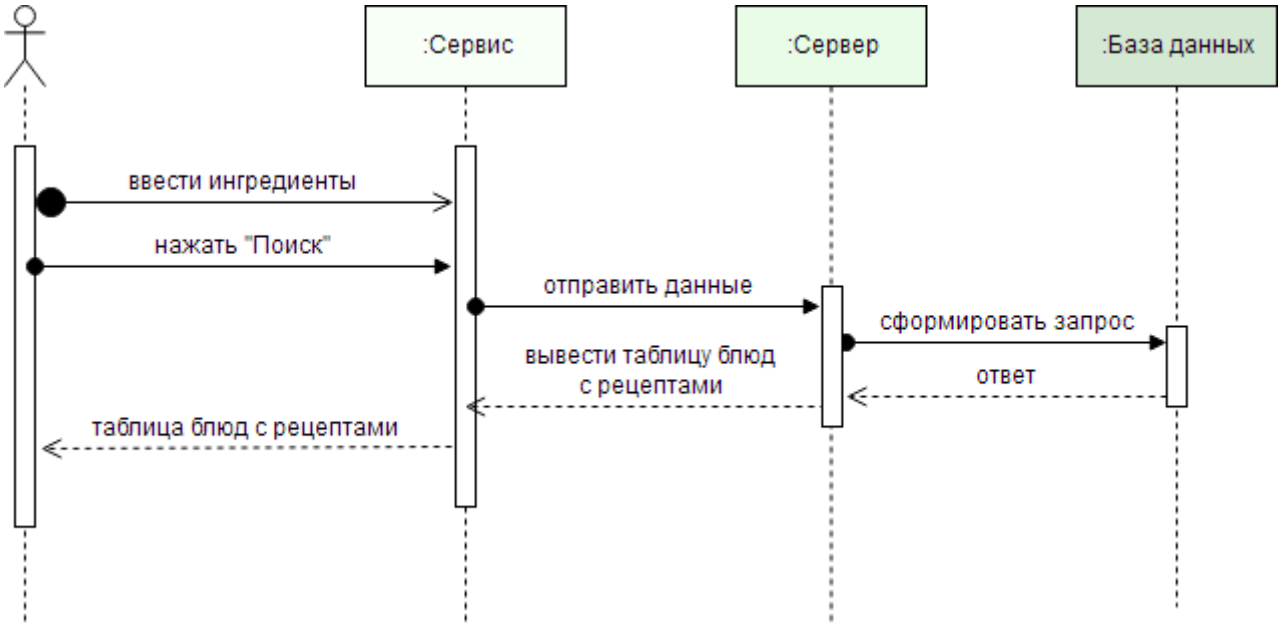

Рисунок 5 – Диаграмма последовательности

5. Диаграмма взаимодействия

    Данная диаграмма дублирует диаграмму последовательности, но только с другого ракурса. Здесь представлено взаимодействие объектов системы с аналогичным алгоритмом действий.

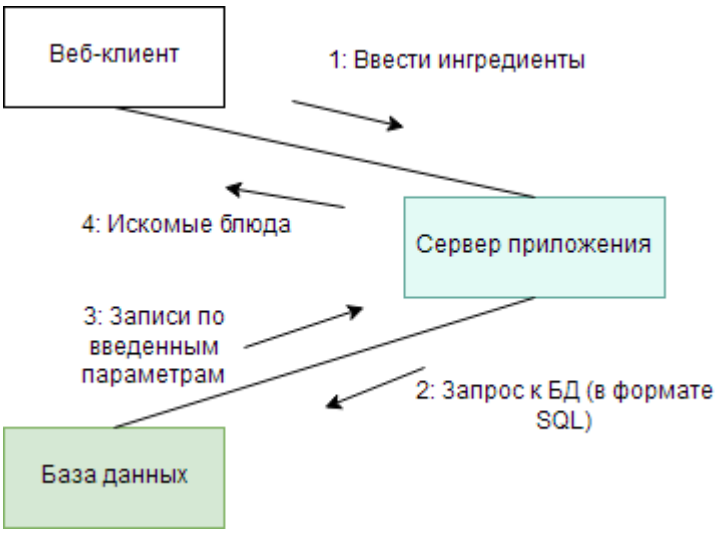

Рисунок 6 – Диаграмма взаимодействия

6. Диаграмма состояний

Здесь рассматривается состояние страницы «Блюда и ингредиенты». Если система не нашла записей по введенным данным, то можно проверить правильность или ввести другие данные.

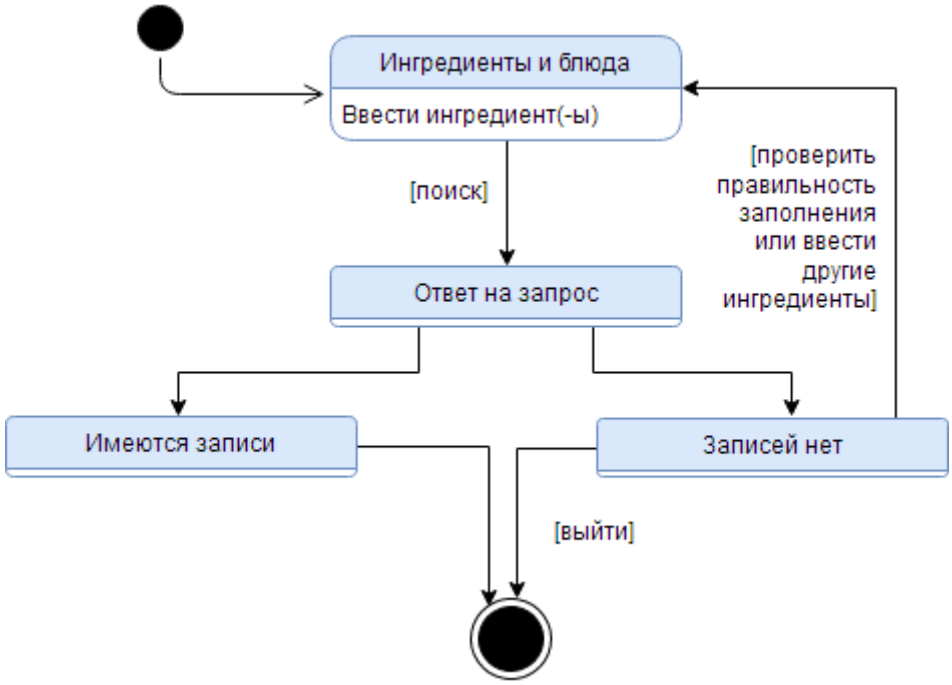

Рисунок 7 – Диаграмма состояний

7. Диаграмма активностей

На данной диаграмме показана последовательность действий пользователя по отношению к системе.

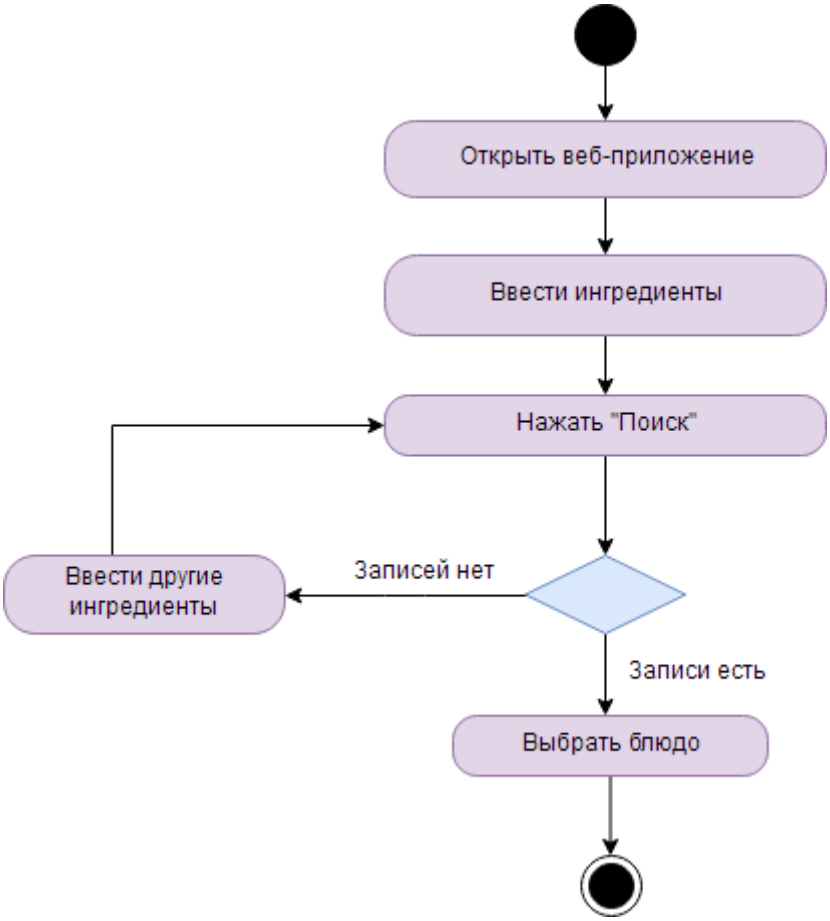

Рисунок 8 – Диаграмма активностей

8. Диаграмма развертывания

Сервер приложения подразумевает под собой логическую реализацию, связанную с базой данных. Взаимодействие между сервером и компьютером происходит посредством сети и браузером.

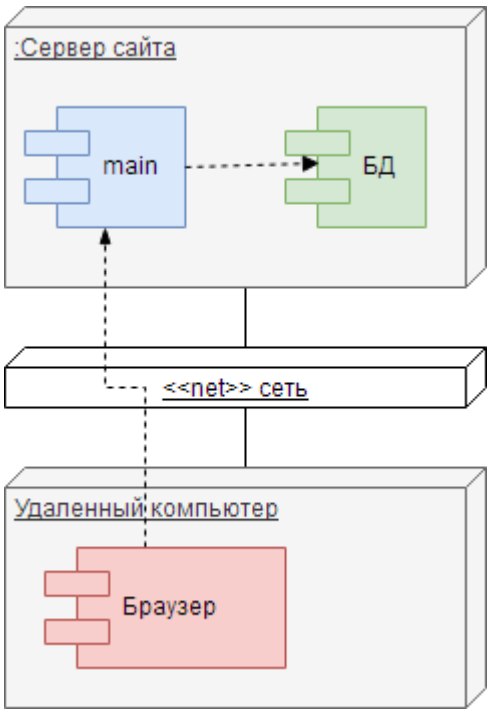

Рисунок 9 – Диаграмма развертывания

3.2 Back-end

В файле app_config.py происходит подключение к БД. Начинается с создания простого скрипта инициализации пакета app. Скрипт создает объект приложения, наследуя Flask. Затем происходит инициализация базы данных, с которой предстоит работать (Листинг 1).

Листинг 1 – app_config.py

```
from flask import Flask
from flask_sqlalchemy import SQLAlchemy

class Config:
    DEBUG = True
    SQLALCHEMY_DATABASE_URI =
"mysql+mysqlconnector://root:301098@localhost/mydb?auth_plugin=mysql_native_passwo
rd"
    SQLALCHEMY_COMMIT_ON_TEARDOWN = True
    SQLALCHEMY_TRACK_MODIFICATIONS = False

app = Flask(__name__)
app.config.from_object(Config())
```

```
mydb = SQLAlchemy(app)
```

Основная логика и взаимодействие с классами описано в файле main.py. Для получения полностью работающего веб-приложения надо написать скрипт, который стартует веб-сервер нашего приложения. Этот скрипт просто импортирует переменную app из нашего пакета app и вызывает метод run для того, чтобы запустить сервер. Переменная app – экземпляр класса Flask, созданный выше.

В этом же файле имеются методы, формирующие запрос на получение всех записей, содержащихся в каждой таблице. Одновременно с этим проверяется, есть ли условия для выборки записей, если нет, то выведутся все записи, если есть, то отфильтруются по введенным параметрам.

Листинг 2 – Вывод записей, содержащихся в таблице Dish

```
@app.route('/dish')
def dish():
    dishes = mydb.session.query(Dish).all()
    form = FormSearch(request.args)
    if form.button_search1.data:
        q = mydb.session.query(Dish)
        if form.dish_name.data != '':
            q = q.filter(Dish.dish_name == form.dish_name.data)
        dishes = q.all()
    return render_template('dish.html', dishes=dishes, form=form)
```

Также для каждой таблицы есть методы, позволяющие изменять, добавлять или удалить записи в таблице.

Листинг 3 – Изменение записей в таблице Dish

```
@app.route('/change_dish/<id>/<do>', methods=['GET', 'POST'])
def change_dish(id, do):
    if do == "add":
        s = Dish()
    else:
```

```
        s = mydb.session.query(Dish).filter(Dish.id_dish == id).one_or_none()
    form = DishForm(request.form, obj=s)
    if do == "delete":
        mydb.session.delete(s)
        mydb.session.flush()
        return redirect(url_for('dish'))
    if form.button_save.data:
        form.populate_obj(s)
        mydb.session.add(s)
        if s.id_dish != id:
            return redirect(url_for('dish', id=s.id_dish))
    return render_template('change_dish.html', form=form)
```

Работу back-end можно описать следующим образом. После того, как клиент совершил какое-либо действие с системой, вызывается соответствующий метод, в котором формируется запрос к БД. Ответ возвращается в этот же метод и данные перенаправляются на страницу.

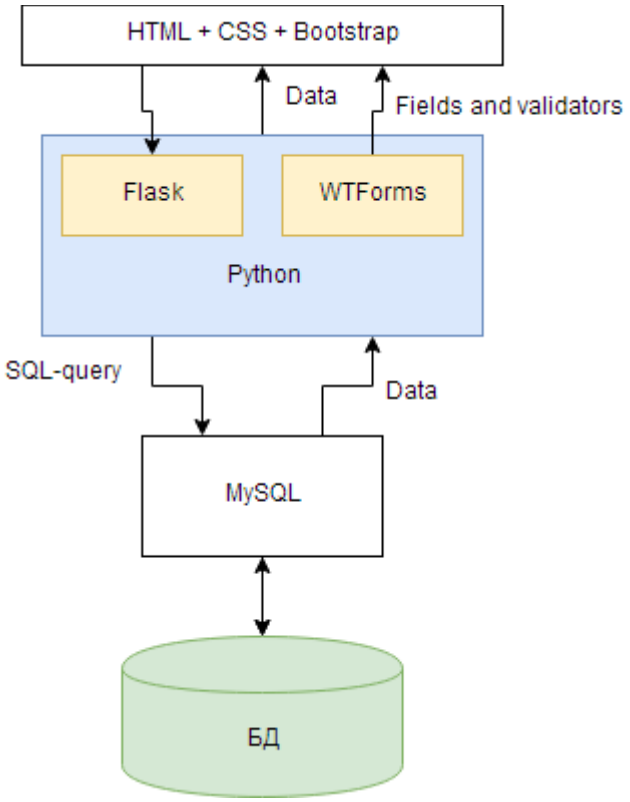

Рисунок 10 – Структура взаимодействия элементов приложения

3.3 Front-end

Протокол передачи гипертекста HTTP предназначен для взаимодействия между клиентами и серверами и работает как протокол запроса-ответа. Веб-браузер может быть клиентом, а приложение на компьютере, на котором размещен веб-сайт, – сервером. Клиент (браузер) отправляет HTTP-запрос серверу, сервер возвращает ответ, который содержит информацию о состоянии запроса и может также содержать запрошенный контент.

HTTP – это протокол, используемый для передачи данных через Интернет. Является частью пакета интернет-протокола и определяет команды и службы, используемые для передачи данных веб-страницы. HTTP использует модель server-client. Клиент может быть домашним компьютером, ноутбуком или мобильным устройством. HTTP-сервер, как правило, является веб-хостом с программным обеспечением веб-сервера, таким как Apache или IIS. Когда пользователь получает доступ к веб-сайту, браузер отправляет запрос на соответствующий веб-сервер и отвечает кодом состояния HTTP. Если URL-адрес действителен и соединение предоставлено, сервер отправит браузеру веб-страницу и связанные файлы.

Два часто используемых метода для запроса-ответа между клиентом и сервером:

1. GET – запрашивает данные из указанного ресурса.
2. POST – отправляет данных, подлежащие обработке, на указанный ресурс.

Перевод GET и POST в буквальном смысле означает получение и постобработку.

POST также является методом передачи переменных формы HTML с одной веб-страницы на другую, не отображая их в адресной строке. Альтернативный метод — GET, который добавляет значения в URL. Запросы HTTP POST предоставляют дополнительные данные от клиента (браузера) на сервер в теле

сообщения. Напротив, запросы GET включают все необходимые данные в URL. Формы в HTML могут использовать любой метод, указав метод = POST или method = GET (по умолчанию) в элементе. Указанный метод определяет, как данные формы передаются на сервер. Когда используется метод GET, все данные формы кодируются в URL-адрес в качестве параметров строки запроса. С POST данные формы появляются в теле сообщения HTTP-запроса. В рамках запроса POST произвольный объем данных любого типа может быть отправлен на сервер в теле сообщения запроса. Поле заголовка в запросе POST обычно указывает тип интернет-носителя тела сообщения. Основное различие между запросами GET и POST заключается в том, что они соответствуют различным HTTP-запросам, как определено в спецификациях HTTP.

Работа front-end реализована с помощью HTML, CSS и Bootstrap.

Во-первых, при запуске приложения надо показать главную страницу, содержащую все таблицы. Flask ищет файлы шаблонов внутри папки templates. В папке приложения была создана папка под названием templates. Внутри templates создан файл menu.html, в котором прописан следующий код (листинг 4):

Листинг 4 – Страница меню

```
<script type="text/javascript" src="/static/jquery/jquery-
3.3.1.min.js"></script>
<script type="text/javascript"
src="/static/bootstrap/bootstrap.min.js"></script>
<link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
<a class="menu" href="{{ url_for('beauty_dish') }}">Красивое блюдо</a>
<a class="menu" href="{{ url_for('dish') }}">Блюда</a>
<a class="menu" href="{{ url_for('ingredient') }}">Ингредиенты</a>
<a class="menu" href="{{ url_for('dish_ingred') }}">Блюда и ингредиенты</a>
<a class="menu" href="{{ url_for('unit') }}">Единицы измерения</a>
```

В main.py необходимо импортировать render_template, который используется для рендеринга файлов шаблонов. А основной метод будет выглядеть следующим образом (листинг 5):

Листинг 5 – Основной метод, возвращающий созданный файл шаблона

```
@app.route('/')
def menu():
    return render_template('./menu.html')
```

По аналогичному принципу работают остальные страницы и методы main.py. При изменении данных используется метод передачи данных POST, а при необходимости вывода существующих данных – метод GET.

Для каждой таблицы существует 2 страницы:

1. Страница, реализующая поиск, вывод данных и удаление записей.
2. Страница, реализующая изменение или добавление данных.

4. Тестирование

Таблица 1 – Тест-кейсы

ID	Name	Type	Preconditions	Steps	Expected result
1	Редактирование записи	Smoke	Открыта страница с таблицей, в которой имеются записи	1. Нажать на "Изменить" рядом с записью 2. Изменить данные в одном из полей 3. Нажать "Сохранить" 4. Проверить изменилась ли запись	1. Открылась страница для изменения параметров 2. Изменение данных возможно 3. Перенаправление на предыдущую страницу 4. Запись изменилась
2	Добавление записи	Smoke	Открыта страница с таблицей	1. Нажать на "Добавить" 2. Ввести данные 3. Нажать "Сохранить" 4. Проверить добавилась ли запись	1. Открылась страница для добавления параметров 2. Введенние данных возможно 3. Перенаправление на предыдущую страницу 4. Запись добавлена
3	Удаление записи	Smoke	Открыта страница с таблицей, в которой имеются записи, не связанные с другими таблицами	1. Нажать на "Удалить" рядом с записью	1. Запись удалена

Таблица 2 – Тест-план

Имя / Тип	ID Test Cases	QA Engineer	Планируемые затраты времени	03.06.2019
Smoke тестирование				
Редактирование записи	1	Боровкова Е.С.	1 мин	pass
Добавление записи	2	Боровкова Е.С.	1 мин	pass
Удаление записи	3	Боровкова Е.С.	1 мин	pass

5. Результаты

Приложение работает успешно и выполняет основные функции. Пример работы приложение показан на рис. 11-12:

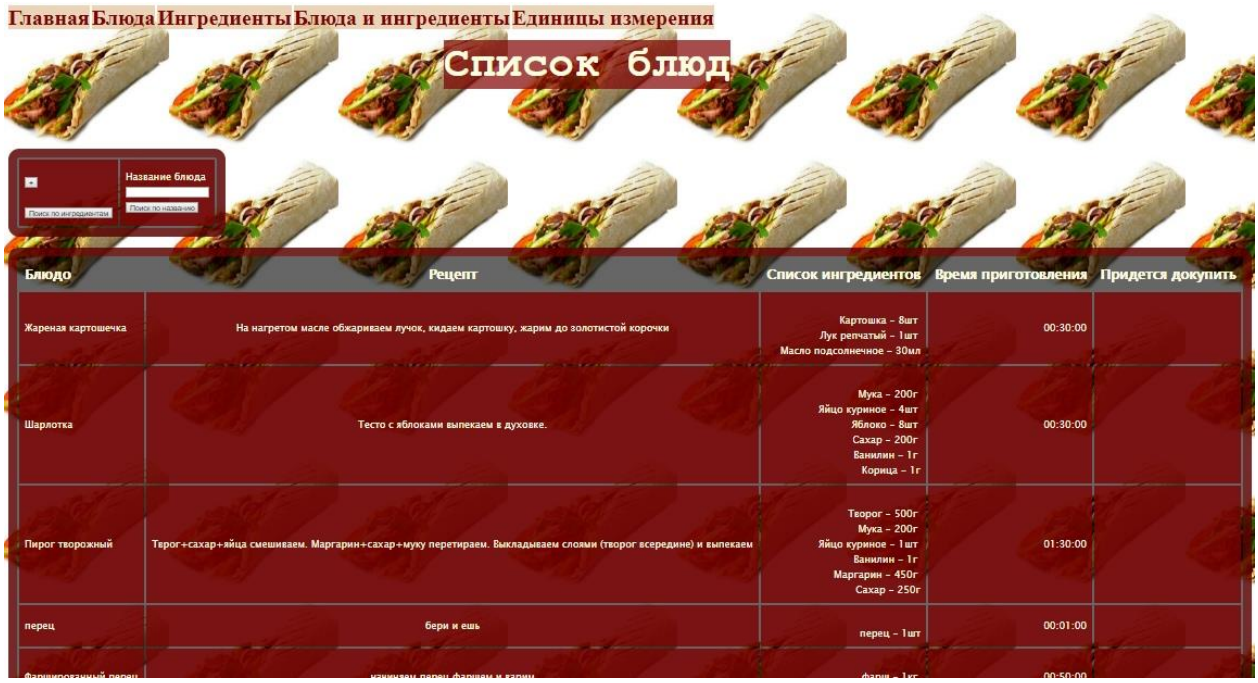

Рисунок 11 – Таблица вывода блюд с рецептами

Рисунок 12 – Пример вывода блюд по введенным ингредиентам

Заключение

Целью данной работы являлась разработка приложения для поиска блюд по имеющимся ингредиентам, которое позволит пользователям уменьшить время на придумывание приготовления блюда, видеть альтернативы приготовления блюд из имеющихся ингредиентов и получать автоматически сгенерированный список необходимых продуктов для покупки.

В ходе разработки были реализованы следующие задачи: просмотр всех таблиц, добавление, удаление, редактирование информации в таблицах, поиск блюд по ингредиентам. С помощью решения этих задач была достигнута цель работы. Приложение полностью соответствует ТЗ, что можно считать успешным завершением проекта.

Список использованных источников

1. Flask против Django: почему Flask может быть лучше: [сайт]. – URL: https://python-scripts.com/flask-vs-django
2. Web-приложения - преимущества и недостатки: [сайт]. – URL: https://mydiv.net/arts/view-web-prilozhenija_preimuxhestva_i_nedostatki.html
3. SQLAlchemy ORM: [сайт]. – URL: https://lectureswww.readthedocs.io/6.www.sync/2.codding/9.databases/2.sqlalchemy/
4. Peewee ORM манипуляция базами данных: [сайт]. – URL: https://python-scripts.com/peewee
5. Преимущества и недостатки MySQL: [сайт]. – URL: http://lectmania.ru/1x5c7e.html
6. Сравнение MySQL и PostgreSQL: [сайт]. – URL: https://losst.ru/sravnenie-mysql-i-postgresql
7. POST, GET: чем отличаются запросы друг от друга: [сайт]. – URL: http://fb.ru/article/367974/post-get-chem-otlichayutsya-zaprosyi-drug-ot-druga
8. Создание веб-приложения с использованием Python Flask и MySQL: [сайт]. – URL: https://code.tutsplus.com/ru/tutorials/creating-a-web-app-from-scratch-using-python-flask-and-mysql--cms-22972