МИНОБРНАУКИ РОССИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

**“ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ”**

Факультет *компьютерных наук*

Кафедра *Информационных технологий управления*

*Веб-приложение “Туса Juice’а”*

*Курсовой проект*

*По дисциплине*

*Технологии программирования*

*09.03.02 Информационные системы и технологии*

Обучающийся \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*Солонин А.А., 3 курс, д/о*

Обучающийся \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*Гнездилов К.В., 3 курс, д/о*

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*В.С. Тарасов, преподаватель*

Воронеж 2020

**СОДЕРЖАНИЕ**

[Введение 4](#_Toc42689429)

[1. Постановка задачи 5](#_Toc42689430)

[2. Анализ предметной области 6](#_Toc42689431)

[2.1. Глоссарий 6](#_Toc42689432)

[2.2. Анализ существующих решений 6](#_Toc42689433)

[2.2.1. vzboltay.com 6](#_Toc42689434)

[2.2.2. russianfood.com 7](#_Toc42689435)

[2.2.3. CocktailsArt 8](#_Toc42689436)

[2.3. Обоснование выбора продуктовых воронок 9](#_Toc42689437)

[2.4. Диаграммы 10](#_Toc42689438)

[2.4.1. Диаграмма вариантов использования 10](#_Toc42689439)

[2.4.2. Диаграммы последовательности 11](#_Toc42689440)

[2.4.3. Диаграммы коммуникаций 14](#_Toc42689441)

[2.4.4. Диаграмма состояний 16](#_Toc42689442)

[2.4.5. Диаграмма развертывания 17](#_Toc42689443)

[2.4.6. Диаграмма классов 18](#_Toc42689444)

[2.4.7. Диаграмма объектов 19](#_Toc42689445)

[2.4.8. Диаграммы активности 20](#_Toc42689446)

[2.4.9. Диаграммы IDEF0 24](#_Toc42689447)

[3. Реализация 26](#_Toc42689448)

[3.1.Средства реализации 26](#_Toc42689449)

[3.2.Клиентская часть 26](#_Toc42689450)

[3.3. Серверная часть 26](#_Toc42689451)

[3.4. Развертывание 26](#_Toc42689452)

[4. Интерфейс 26](#_Toc42689453)

[4.1. Стартовая страница 26](#_Toc42689454)

[4.2. Страница авторизации 27](#_Toc42689455)

[4.3. Страница регистрации 27](#_Toc42689456)

[4.4. Страница поиска 28](#_Toc42689457)

[4.5. Страница добавления коктейля 28](#_Toc42689458)

[5. Тестирование 29](#_Toc42689459)

[Заключение 31](#_Toc42689460)

[Приложение 32](#_Toc42689461)

[Приложение 1. Результаты тестирования 32](#_Toc42689462)

Список используемых источников………………………........................................33

# Введение

Каждому человеку приходится участвовать во множестве вечеринок. При этом организация вечеринки — это всегда ответственная и трудная деятельность. Одним из важных составляющих успешной вечеринки является наличие напитков. Организатор вечеринки может испытывать значительные трудности в подборе алкогольных напитков, так как компания вечеринки может сложится разная: от девушек, предпочитающих легкие коктейли до взрослых мужчин, отдающих предпочтение крепкому алкоголю. Если организатор вечеринки не обладает высокими компетенциями в выборе алкоголя, ему приходится обращаться к дополнительным источникам информации. При этом люди всегда хотят попробовать что-то новое, возникает необходимость иметь ресурс, который подскажет, какие напитки точно должны понравится.

На данный момент сайтов и мобильных приложений, предоставляющих знания о напитках, не отвечают требованиям пользователя. Они могут иметь сложный интерфейс, замедляющий доступ к информации, или являться платными, что тоже отталкивает пользователя от использования такого сервиса.

В данной курсовой работе рассматривается разработка веб-приложения, оптимизирующего процесс поиска инструкций изготовления коктейлей, а также снижающего временные затраты на обучение технологиям изготовления напитков.

# Постановка задачи

Основным назначением Web - приложение «Туса Juice’a» является формирование стандартной коктейльной базы с минимальными продуктовыми требованиями необходимыми для создания коктейлей как в заведениях, так и в домашних условиях.  
В рамках проекта автоматизируется деятельность в следующих сферах:  
1. Обогащение информационной сферы знаний пользователя;  
2. Анализ предпочтений пользователя.

Приложение должно позволять пользователю:

1. Искать напитки
2. Получать рейтинг напитков
3. Регистрироваться в системе
4. Добавлять новые рецепты напитков

# Анализ предметной области

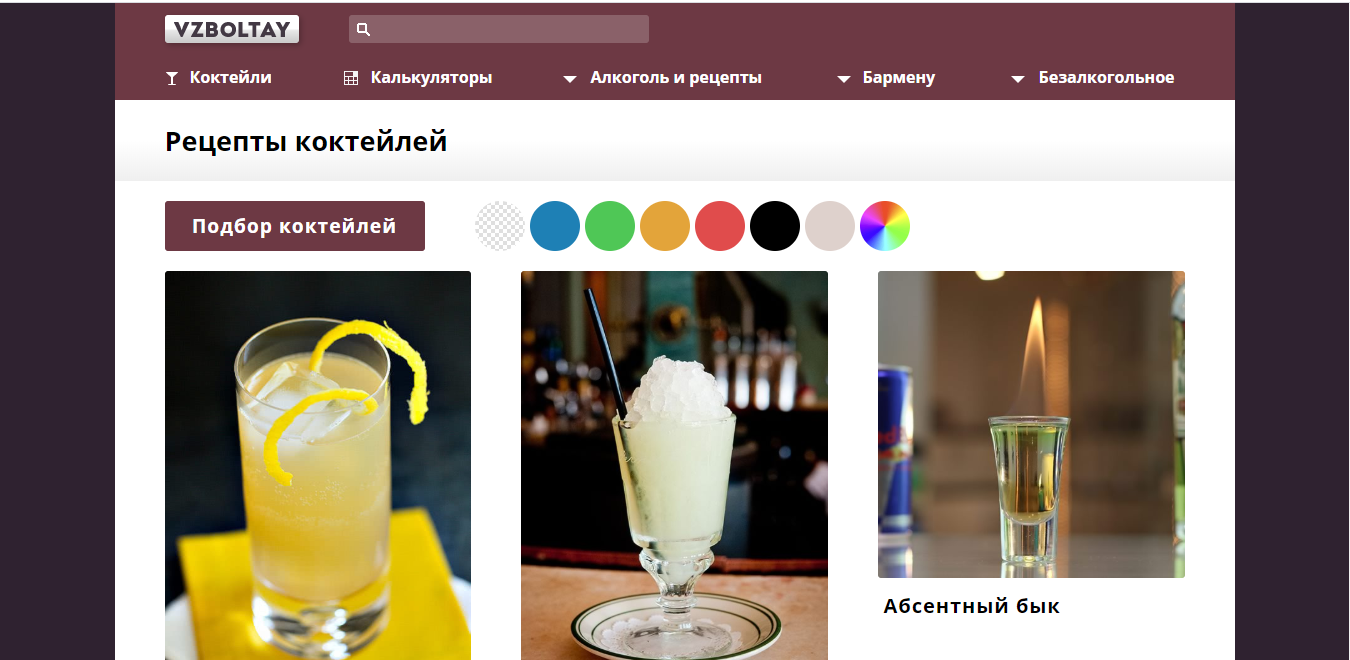
## 2.1. Глоссарий

* Напиток – это коктейль из нескольких ингредиентов.
* Рейтинг – список напитков, отсортированный в порядке уменьшения средней оценки пользователей.

## 2.2. Анализ существующих решений

Рассмотрим существующие решения.

### 2.2.1. vzboltay.com



*Рисунок 1. Главная страница сайта vzboltay.com*

Сайт vzboltay.com содержит большую подборку алкогольных напитков, а также на сайте содержится много информации о различных видах алкоголя.

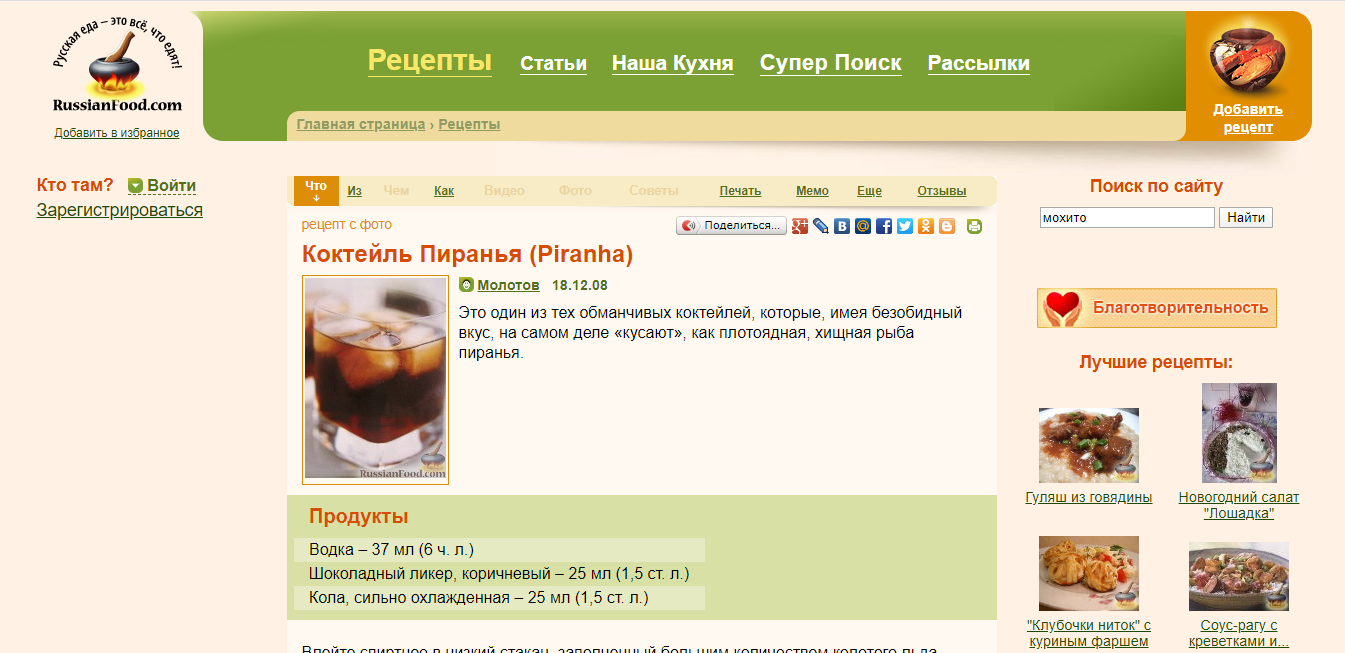
Достоинства:

* Большая база коктейлей
* Калькуляторы дистилляции и производства алкоголя
* Дополнительно на сайте размещены научно-познавательные статьи на тему алкоголя
* Подбор коктейлей по типу алкоголя

Недостатки:

* Неудобный фильтр коктейлей по цветам на главной странице
* Нет возможности добавления собственных коктейлей
* Нет рейтинговой системы

### 2.2.2. russianfood.com



*Рисунок 2. Страницанапитка сайта russianfood.com*

Сайт russianfood.com специализируется на рецептах еды, а также алкогольных и неалкогольных напитках.

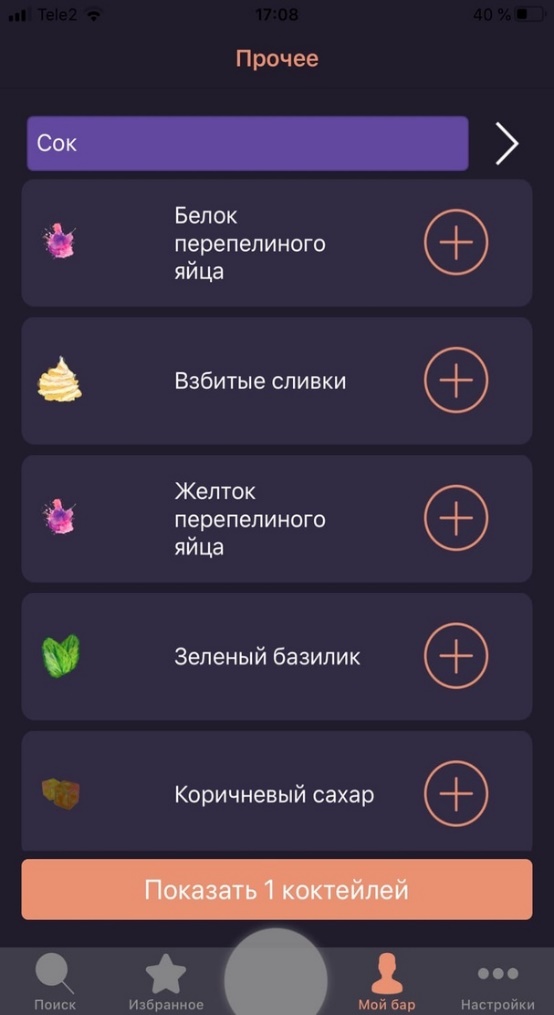
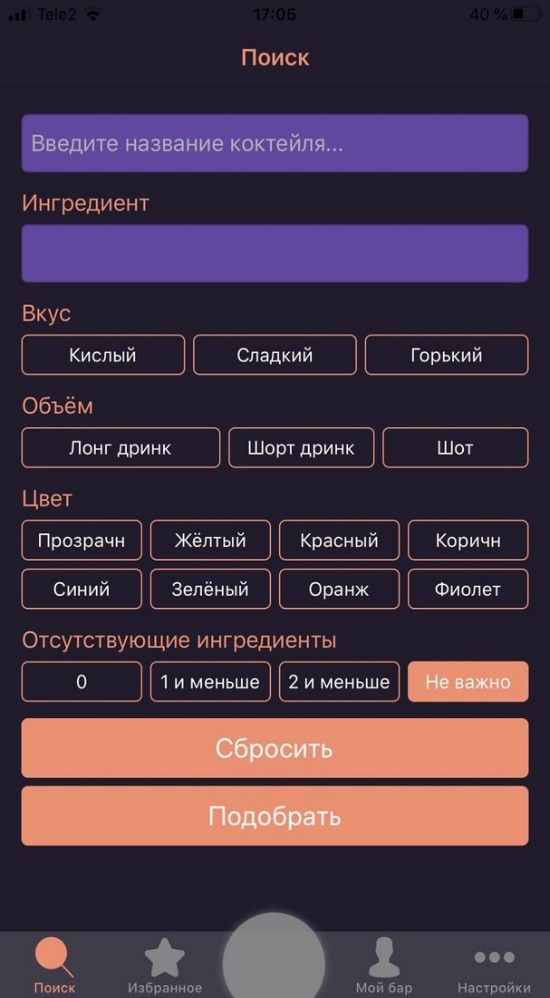
Достоинства:

* Возможность добавлять рецепты
* Возможность сохранять рецепты в записную книжку, чтобы не потерять
* Комментарии

Недостатки:

* Нет рейтинговой системы
* Одинаковые рецепты добавлены несколько раз
* Устаревший дизайн

### 2.2.3. CocktailsArt



*Рисунок 3. Интерфейс приложения CocktailsArt*

Мобильное приложение Cocktails Artсодержит информацию более чем о 300 напитках и имеет продвинутый поиск по ингредиентам и другим параметрам.

Достоинсва:

* Интуитивно-понятный интерфейс
* Поиск с множеством характеристик
* Возможность добавлять рецепты в избранное
* Возможность ввести ингредиенты и получить список коктелей,которые можно сделать из этих ингредиентов

Недостатки:

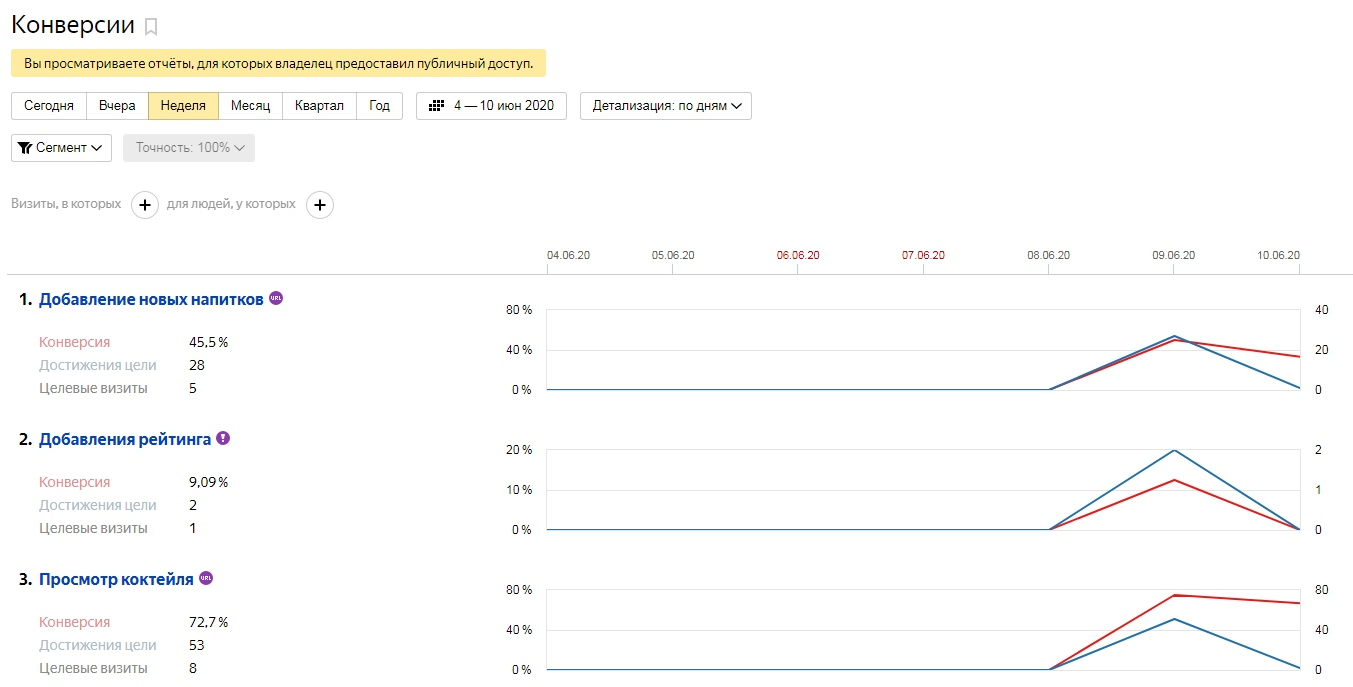
* Нет рейтинговой системы
* Большинство рецептов платные

## 2.3. Обоснование выбора продуктовых воронок

С помощью сервиса Яндекс. Метрика было принято решение создать продуктовых воронки, основанных на целевых действиях пользователя Сервиса. ([https://metrika.yandex.ru/dashboard?id=64750285](https://vk.com/away.php?to=https%3A%2F%2Fmetrika.yandex.ru%2Fdashboard%3Fid%3D64750285&cc_key=)) Также принимая во внимание анализ предметной области и личный интерес разработчиков Сервиса, выделены два сценария пользователя.

1. Добавление новых напитков  
 2.Рейтинговая система  
Причины выбора сценариев:  
 1. Улучшение функциональных возможностей (возможность добавления новых напитков, а так их оценивание, а также возможность просмотра всех напитков по представленном рейтингу)  
 2.Понимание характера выбора коктейлей среднестатистического зарегистрированного/незарегистрированного пользователя Сервиса (какие коктейли люди чаще всего просматривают, что это за люди (какой возраст, какие устройства))

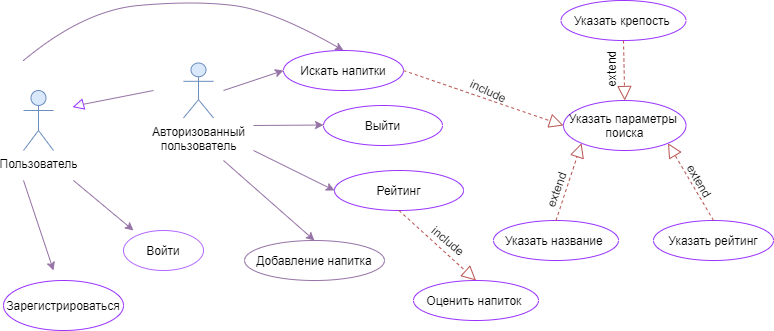
3.Улучшение графического интерфейса (было выяснено, что пользователям не нравится дизайн сайта, т.к. он является устаревшим.)

Воронки:   


*Рисунок 4. Демонстрация воронок*

## 2.4. Диаграммы

### 2.4.1. Диаграмма вариантов использования

****

*Рисунок 5. Диаграмма вариантов использования*

Рассмотрим диаграмму вариантов использования на рисунке5.

В системе есть два актера: Пользователь и Авторизованный пользователь.

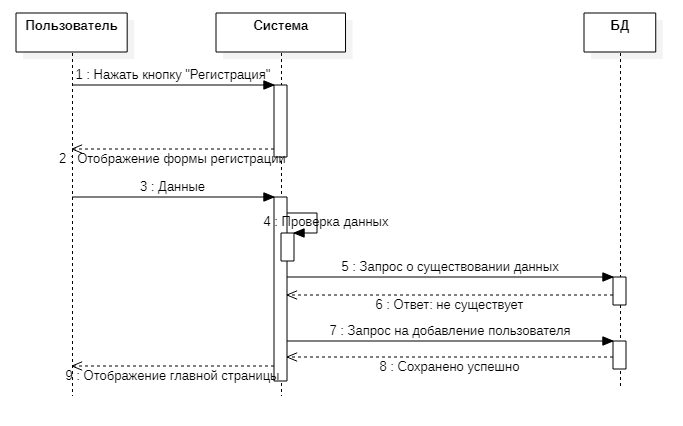
Действия, который может совершить Пользователь:

* Зарегистрироваться
* Войти

Действия, который может совершить Авторизованный пользователь:

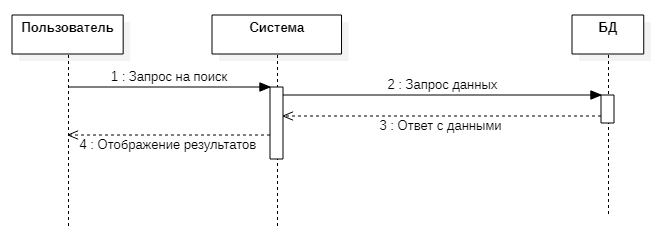
* Добавить напиток
* Искать напитки
* Просмотреть рейтинг
* Выйти из системы

### 2.4.2. Диаграммы последовательности

Диаграмма последовательности для процесса регистрации, рисунок 6

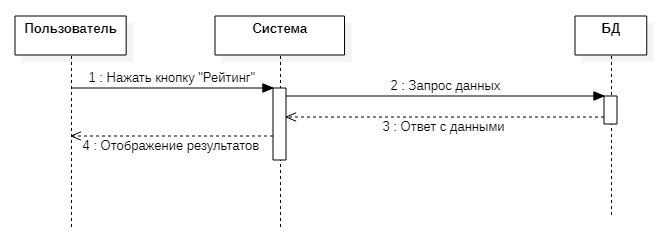
*Рисунок 6. Диаграмма последовательности для процесса регистрации*

Чтобы произвести регистрацию в системе пользователь нажимает кнопку «Регистрация». Затем система отображает форму для заполнения, куда пользователь вводит логин и пароль. Система проверяет, что необходимые поля введены корректно, Пользователя с такими данными еще нет в Системе. Если проверка прошла успешно, система делает запрос на добавление нового пользователя. Когда данные сохранены, БД присылает ответ об успешном сохранении, и система отображает главную страницу.

Диаграмма последовательности для поиска напитков, рисунок7

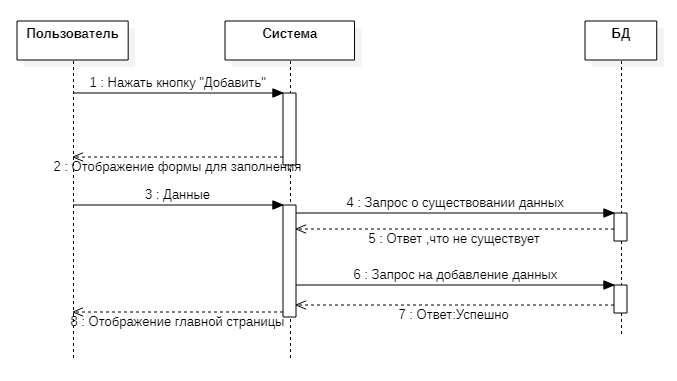
*Рисунок 7. Диаграмма последовательности для поиска напитков*

Чтобы произвести поиск напитков пользователь должен ввести название напитка или выбрать критерии поиска, затем система вместе с запросом на поиск получает эти данных и делает запрос к базе данных. База данных отправляет результаты поиска и система отображает результат.

Диаграмма последовательности для получения рейтинга, рисунок8

*Рисунок8. Диаграмма последовательности для получения рейтинга*

Чтобы получить рейтинг напитков пользователь должен нажать кнопку «Рейтинг», затем система делает запрос к базе данных. База данных отправляет отсортированные результаты, и система отображает результат.

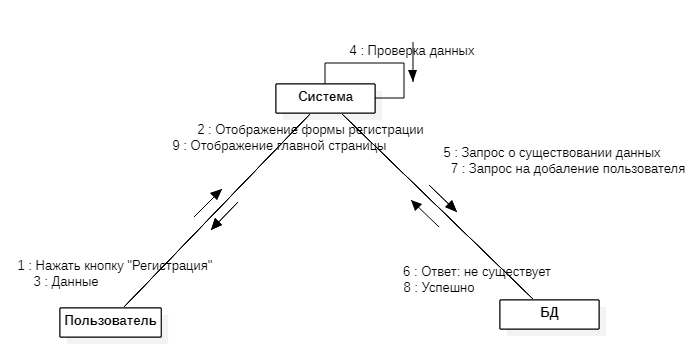
Диаграмма последовательности для добавления напитка, рисунок9

*Рисунок9. Диаграмма последовательности для добавления напитка*

Чтобы добавить новый напиток пользователь нажимает кнопку «Добавить». Затем система отображает форму для заполнения, куда пользователь вводит информацию о напитке. Система проверяет, что напитка с такими параметрами не существует в Системе, затем делает запрос на добавление нового напитка. Когда данные сохранены, БД присылает ответ об успешном сохранении, и система отображает главную страницу.

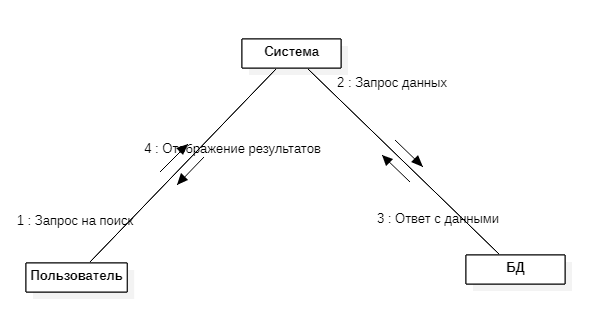
### 2.4.3. Диаграммы коммуникаций

Диаграмма коммуникации для процесса регистрации, рисунок10



*Рисунок10. Диаграмма коммуникации для процесса регистрации*

Диаграмма коммуникации для поиска напитков, рисунок 11



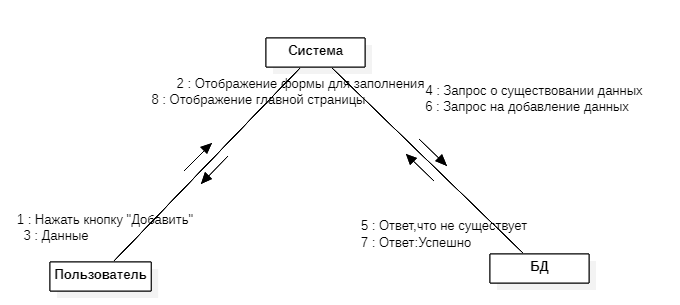
*Рисунок 11. Диаграмма коммуникации для поиска напитков*

Диаграмма коммуникации для получения рейтинга, рисунок 12



*Рисунок 12. Диаграмма коммуникации для получения рейтинга*

Диаграмма коммуникации для добавления напитка, рисунок 13

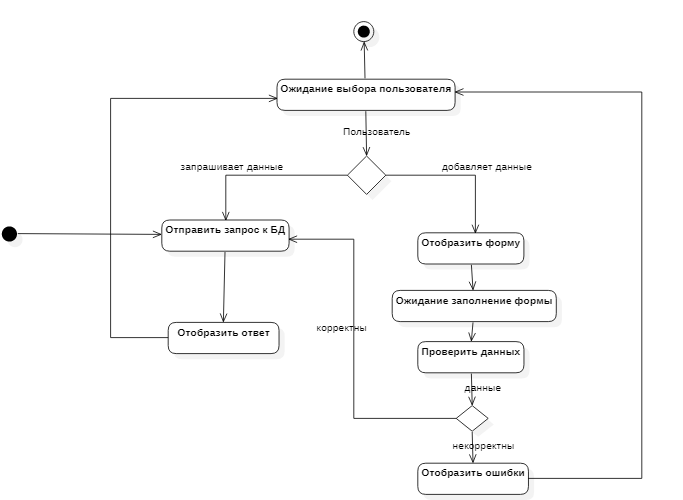


*Рисунок 13. Диаграмма коммуникации для добавления напитка*

Сообщения в диаграммах коммуникаций такие же, что и в диаграммах последовательности, поэтому их повторное описание не приводится.

### 2.4.4. Диаграмма состояний

На рисунке 14 изображена диаграмма состояний системы.



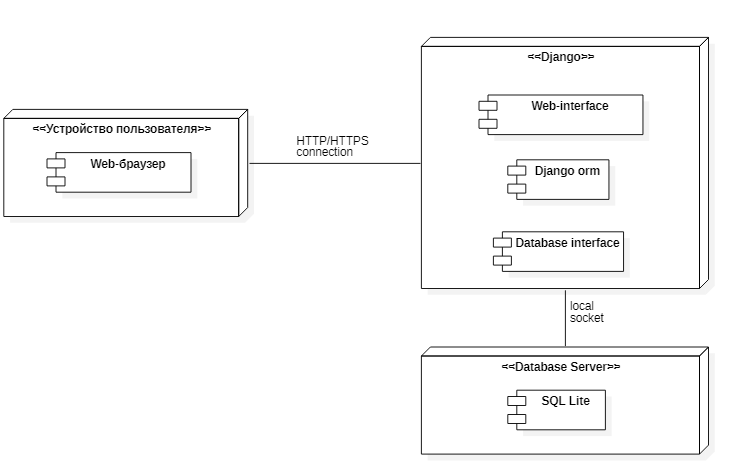
*Рисунок 14. Диаграмма состояний*

Диаграмма состояний отражает возможные состояния системы. Начальным состоянием системы, можно считать, отправление запроса к БД при отображении главной страницы.В зависимости от выбора пользователя возможны 2 основные цепочки состояний, связанных с отображение данных, необходимых пользователю, и с добавлением данных пользователем.

Если пользователь запрашивает данные, система отправляет запрос данных к базе данных, затем отображает ответ от базы данных и переходит в состояние ожидания выбора пользователя.

Если пользователь хочет добавить данные, система отображает форму для ввода данных, затем проверяет введенные данные. Если данные верны, система отправляет запрос к БД, иначе отображает ошибки и переходит в состояние ожидания выбора пользователя.

### 2.4.5. Диаграмма развертывания



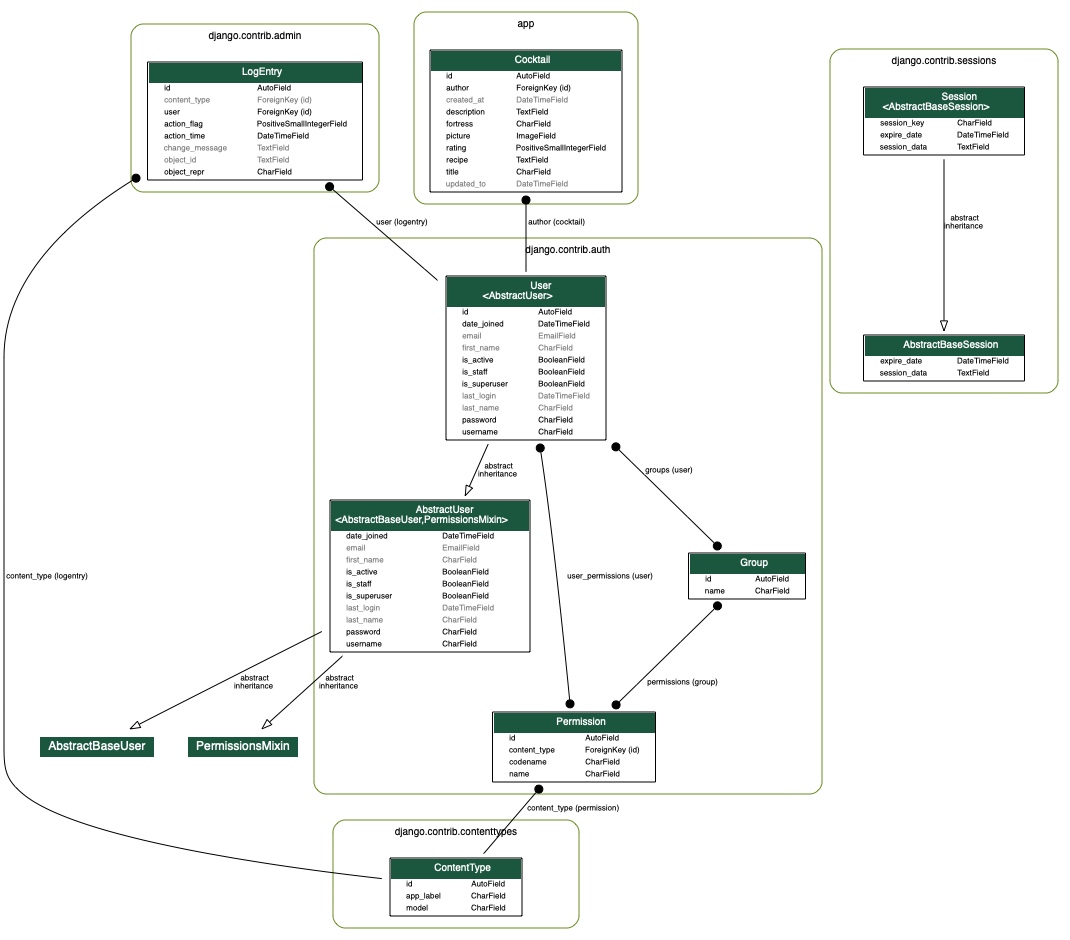
*Рисунок15. Диаграмма развертывания*

На рисунке15 представлена диаграмма развертывания, чтобы определить какие аппаратные компоненты («узлы») существуют, какие программные компоненты работают на каждом узле, и как различные части этого комплекса соединяются друг с другом.

Для данной системы узлами являются устройство пользователя,web-сервер наDjangoи сервер базы данных. На узле устройство пользователя работает web‑браузер, на узле сервера расположеныweb-интервейс, Djangoormи интерфейс базы данных, на узле сервера базы данных хранится СУБД SQLLite.

### 2.4.6. Диаграмма классов

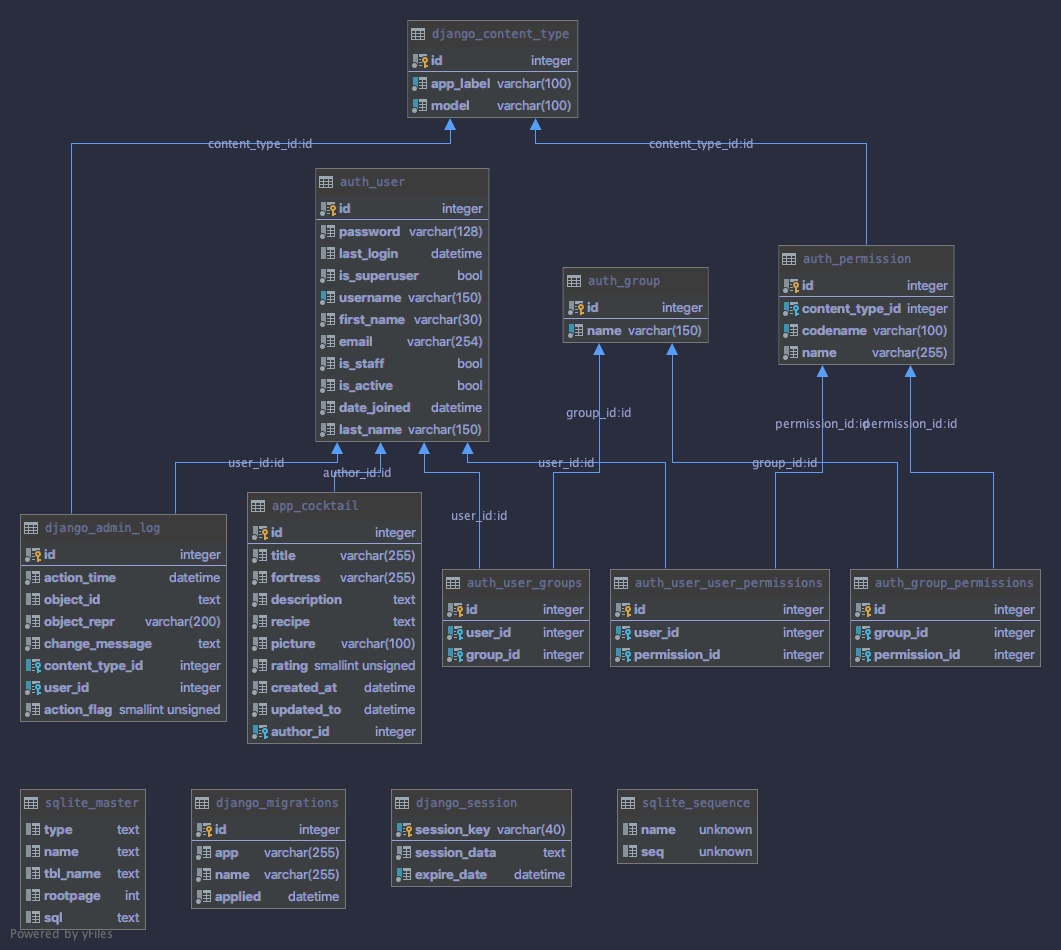
На рисунке 16 представлена диаграмма классов.



*Рисунок 16. Диаграмма классов*

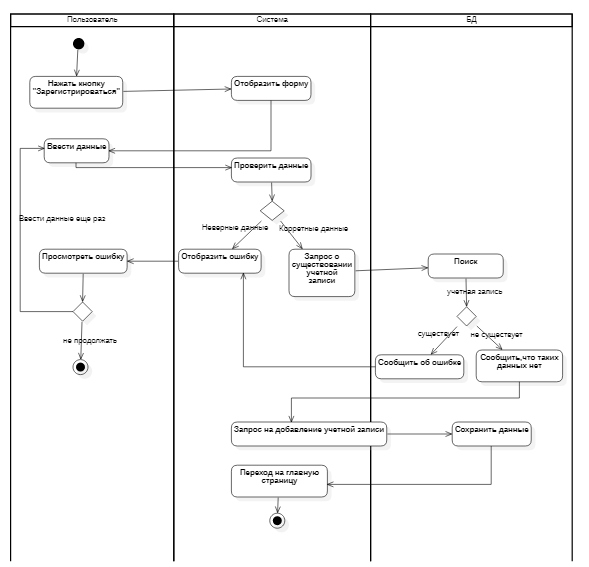
### 2.4.7. Диаграмма объектов

На рисунке 17 представлена диаграмма объектов.



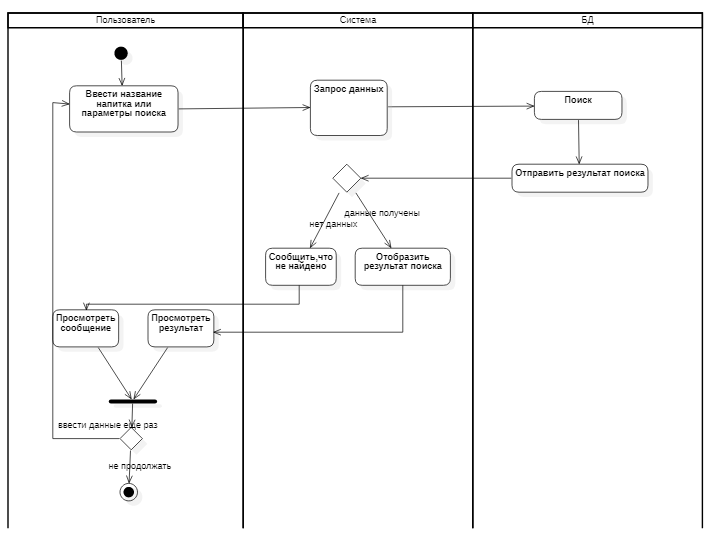
*Рисунок 17. Диаграмма объектов*

### 2.4.8. Диаграммы активности



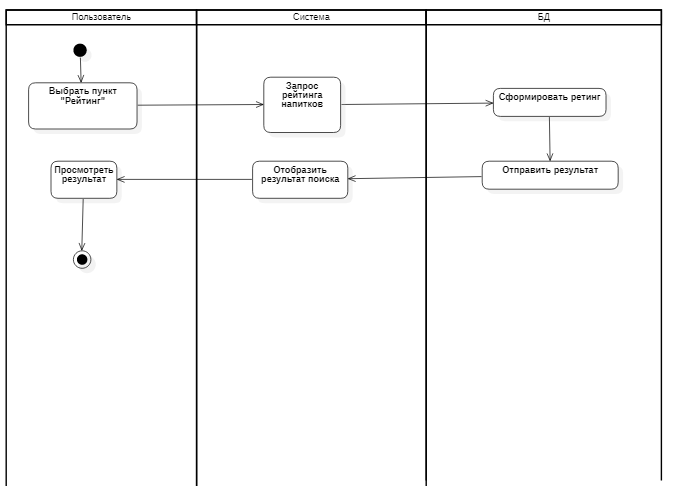
*Рисунок18. Диаграмма активности для процесса регистрации*

Чтобы зарегистрироваться в системе пользователь нажимает кнопку «Регистрация», система отображает форму для ввода информации, пользователь вводит данные для регистрации. Затем система проверяет введенные данные, если данные корректны, система проверяет наличие такой учетной записи в системе. Если такой учетной записи в системе нет, система отправляет запрос на добавление учетной записи и база данных сохраняет новые данные, затем система переходит на главную страницу. В случае возникновения ошибок, системаотображает эти ошибки пользователю и он может ввести данные еще раз или завершить действие.



*Рисунок 19. диаграмма активности для поиска напитков*

Чтобы произвести поиск напитков пользователь должен ввести название напитка или выбрать критерии поиска, затем система вместе с запросом на поиск получает эти данных и делает запрос к базе данных. База данных отправляет результаты поиска, если ничего не найдено, система сообщает, что данных не найдено, пользователь просматривает сообщение и может ввести данные заново. Если по запросу пользователя существуют данные в базе, система отображает результаты, пользователь может ввести данные еще раз или завершить действие.



*Рисунок 20. Диаграмма активности для получения рейтинга*

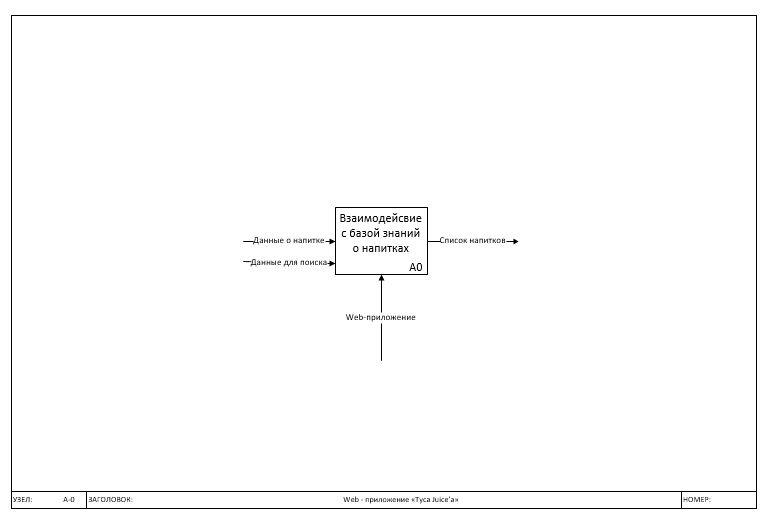
Чтобы получить рейтинг напитков пользователь выбрать пункт «Рейтинг», затем система делает запрос к базе данных. База данных формирует рейтинг и отправляет отсортированные результаты, и система отображает результат. Затем пользователь просматривает результаты и завершает действие.



*Рисунок 21. Диаграмма активности для добавления напитка*

Чтобы добавить напиток пользователь выбирает пункт «Добавить», система отображает форму для ввода информации, пользователь вводит данные о напитке. Затем система проверяет введенные данные, если данные корректны, система проверяет существование такого напитка в системе. Если такого напитка нет, система отправляет запрос на добавление напитка и база данных сохраняет новые данные, затем система переходит на главную страницу. В случае возникновения ошибок, система отображает эти ошибки пользователю, и пользователь может ввести данные еще раз или завершить действие.

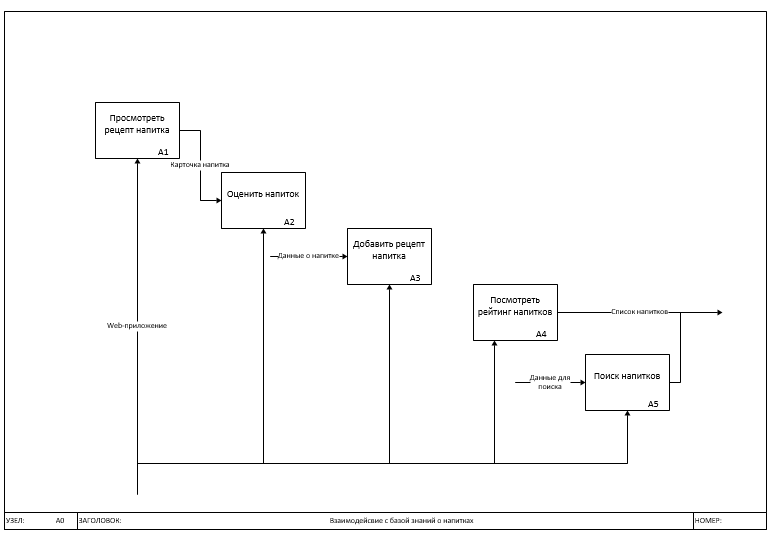
### 2.4.9. Диаграммы IDEF0



*Рисунок 22. Диаграмма IDEF0 верхнего уровня*

Данная диаграмма является наиболее общим представлением системы.

Входных данными для функционального блока «Взаимодействие с базой знаний о напитках» являются: данные о напитке, данные для поиска.Выходным потоком является список напитков. Механизмом для данной модели являетсяweb-приложение, на котором производятся все процессы этой системы.



*Рисунок 23. Диаграмма IDEF0 уровень А0*

Данная диаграмма раскрывает функциональный блок диаграммы верхнего уровня«Взаимодействие с базой знаний о напитках». Он состоит из 5 функциональных блоков: А1 «Просмотреть рецепт напитка», А2 «Оценить напиток»,А3 «Добавить рецепт напитка», А4 «Посмотреть рейтинг напитков», А5 «Поиск напитков».

# Реализация

## 3.1. Средства реализации

Для разработки в данной области использовалась среда разработки pyCharm от компании JetBrains.

Его особенностями является статический анализ кода, навигация по коду (файловой структуре), поддержка систем контроля версий, множество инструментов для веб-разработки приложений на языке программирования python.

Для написания данной работы использовался язык программирования высокого уровня Python, так как он позволяет создать мощные приложения в короткие сроки. Благодаря обширной стандартной библиотеке нет необходимости реализовывать функционал приложения с нуля.

В качестве инструмента для написания серверного приложения, позволяющего использующего готовые решения, был выбран фреймворк Django.

## 3.2. Клиентская часть

HTML – язык гипертекстовой разметки, являющийся первичной основой разрабатываемого web-приложения. Разметка строится при помощи дескрипторов (тегов), для их выделения используются угловые скобки.

Теги могут быть парными, то есть для каждого открывающего тега требуется соответствующий закрывающий, и непарными, когда этого не требуется.

У тегов существует принадлежность к классам, которая задается при помощи атрибута class:

<div class="class-name"></div>

Также классам можно устанавливать идентификаторы, обозначающие уникальные имена для элемента:

<div id="id-name"></div>

Структурно все теги страницы располагаются в теге тега <html>, внутри которого располагаются два основных тега <head>, который содержит информацию для всей страницы, например, подключение стилей, шрифтов или указания имени страницы, и <body>, внутри которого описывается содержимое непосредственно самой страницы.

В конце страницы тега <body>можно указать скрипты языка JavaScript, при их наличии, в теге <script>:

<script src='script.js'></script>

## 3.3. Серверная часть

Приложения на фреймворке Django конструируется из нескольких приложений для построения более сложной системы, при этом данные каждое приложение решает одну конкретную задачу.

Чтобы создать основу для Django приложения, необходимо прописать команду django-adminstartprojecttyca. При создании приложения изначально создается одноименная папка с конфигурационными файлами проекта:

manage.py – скрипт взаимодействия с самим ядром Django

\_\_init\_\_.py – пустой файл, который указывает, что текущий каталог является пакетом Python

app.py содержит информацию о проекте

settings.py – настройки/конфигурация проекта

urls.py – содержит информацию о всех доступных URL адресах

models.py – содержит информацию о моделях, используемых приложением

views.py – содержит информацию о представлениях

Взаимодействие с базой данных в Django осуществляется при помощи объектно-ориентированного отображения (Object-RelationalMapper или ORM). Учитывая, что Django старается упростить любую операцию при работе с проектом, стандартные SQL-инструкции не используются. Вместо этого фреймворк оперирует особой структурой – моделью.

Модель представляет собой одну таблицу в БД и содержит набор полей и поведение данных, которые хранит приложение. При ее создании формируется программный интерфейс приложения (API) для работы с базами данных. Важно уточнить, что в ее состав включены операции создания, чтения, изменения и удаления.

Например, в проекте tyca в файле app/models.py реализовано множество моделей. Рассмотрим часть модели Evaluation: в ней описаны два поля user, cocktail и evaluation, где первый имеет тип CharField(255), а второй является внешним ключом на первичный ключ модели User, а третий является внешним ключом на первичный ключ модели Cocktail:

class Evaluation(models.Model):

user = models.ForeignKey(User, models.CASCADE,

related\_name=‘evaluation\_user')

cocktail = models.ForeignKey(Cocktail, models.CASCADE,

related\_name=‘evaluation\_cocktail')

evaluation = models.PositiveSmallIntegerField()

Связи между сущностями аналогичны связям обычных баз данных. Так models.CASCADE подразумевает SQL правило ONDELETECASCADE (каскадное удаление).

После того, как модель была изменена, необходимо применить миграцию. При этом изменения модели переносятся в БД.

Одним из других важных механизмов Django является взаимодействие URL и Views (представления или классы представлений). Когда пользователь вводит определенный URL, Django использует URLconf и начинает поиск по массиву urlpatterns, состоящему из элементов списка путей url. Далее он проходит по каждому url шаблону, пока не определится корректный. Если такой был найден, то вызывается соответствующее представление, в противном случае выводится ошибка.

Представлением в Django является обычная функция языка python, которая определяет логику дальнейшей работы. Это может быть запрос информации из модели и передача ее в шаблон.

## 3.4. Развертывание

Основной платформой для развертывания Django является WSGI, это фактически стандарт для веб-серверов и приложений на Python.

Команда startproject установит простую WSGI конфигурацию по-умолчанию, которую вы можете впоследствии изменить под нужды вашего проекта и использовать с любым WSGI-совместимым веб-сервером.

Когда WSGI сервер загружает ваше приложение, Django необходимо импортировать модуль с настройками.

Django использует переменную окружения DJANGO\_SETTINGS\_MODULE для определения расположения модуля настроек. Она должна содержать путь для импорта этого модуля. Вы можете использовать разные значения при разработке и на боевом сервере, все зависит от организации настроек в вашем проекте.

Если переменная не определена, wsgi.py использует значение tyca.settings, где tyca название вашего проекта. Вот как runserver определяет расположение файла настроек для вашего проекта.

## 3.5. Тестирование

Автоматизированное тестирование - чрезвычайно полезный инструмент для современного веб-разработчика. Вы можете использовать набор тестов - набор тестов - для решения или избежания ряда проблем:

Когда вы пишете новый код, вы можете использовать тесты, чтобы подтвердить, что ваш код работает так, как ожидается.

При рефакторинге или модификации старого кода можно использовать тесты, чтобы убедиться, что изменения не повлияли неожиданно на поведение приложения.

Тестирование веб-приложения - сложная задача, так как веб-приложение состоит из нескольких уровней логики - от обработки HTTP-запросов, до проверки и обработки форм, рендеринга шаблонов. С помощью фреймворка тестирования и разнообразных утилит Django можно симулировать запросы, вставлять тестовые данные, проверять вывод приложения и в целом проверять, что ваш код делает то, что должен.

Предпочтительным способом написания тестов в Django является использование модуля юнит-тестов, встроенного в стандартную библиотеку Python. Подробно об этом говорится в документе "Написание и запуск тестов».

Разберем пример теста работоспособности детального просмотра коктейля.

def test\_detail():

User.objects.create\_user('foo', password='bar')

Cocktail.objects.create(

title='Test',

fortress='ttt',

description='fbcvbcvbc',

recipe='gnfghg',

picture='tуса\_juice/static/1.jpg',

author=User.objects.first()

)

cocktail = Cocktail.objects.first()

assert (

reverse("detailed\_view", kwargs={"pk": cocktail.id})

== f"/cocktail/{cocktail.id}"

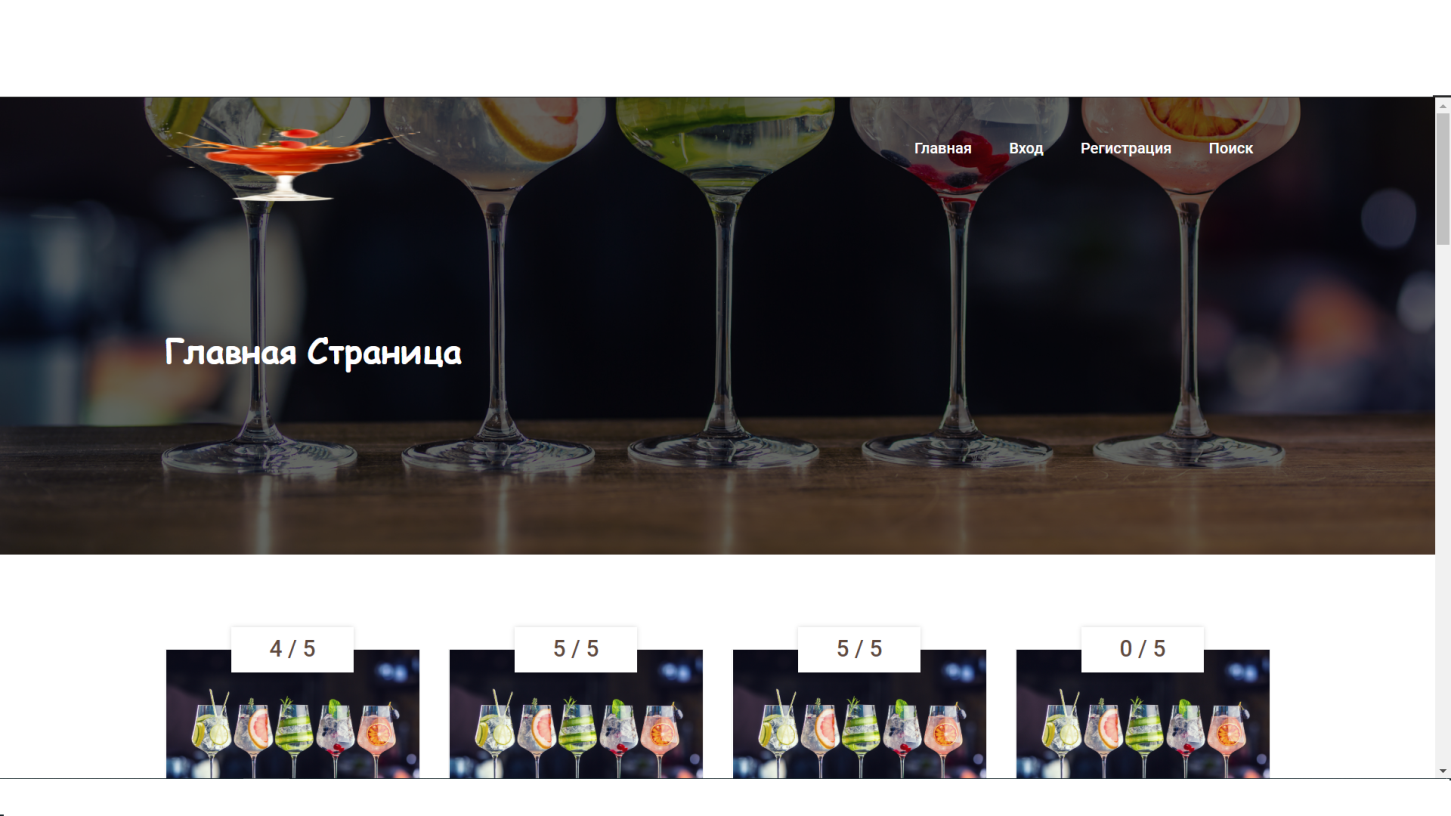
)

assert resolve(f"/cocktail/{cocktail.id}").view\_name == «detailed\_view»

В тесте происходит создание записи, получения этой записи, а затем проверки работы view.

# Интерфейс

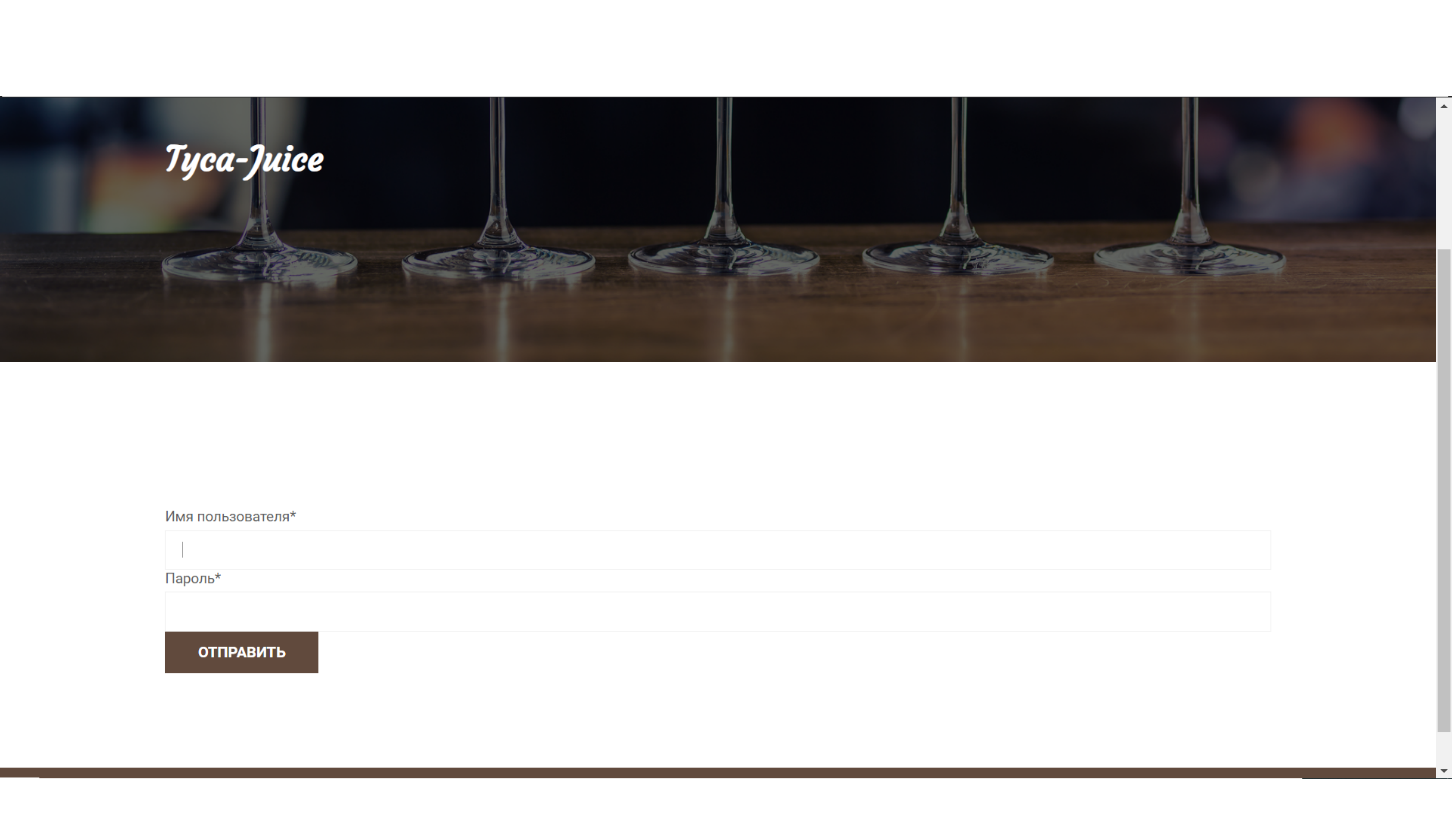
## 4.1. Стартовая страница



*Рисунок 24. Стартовая страница*

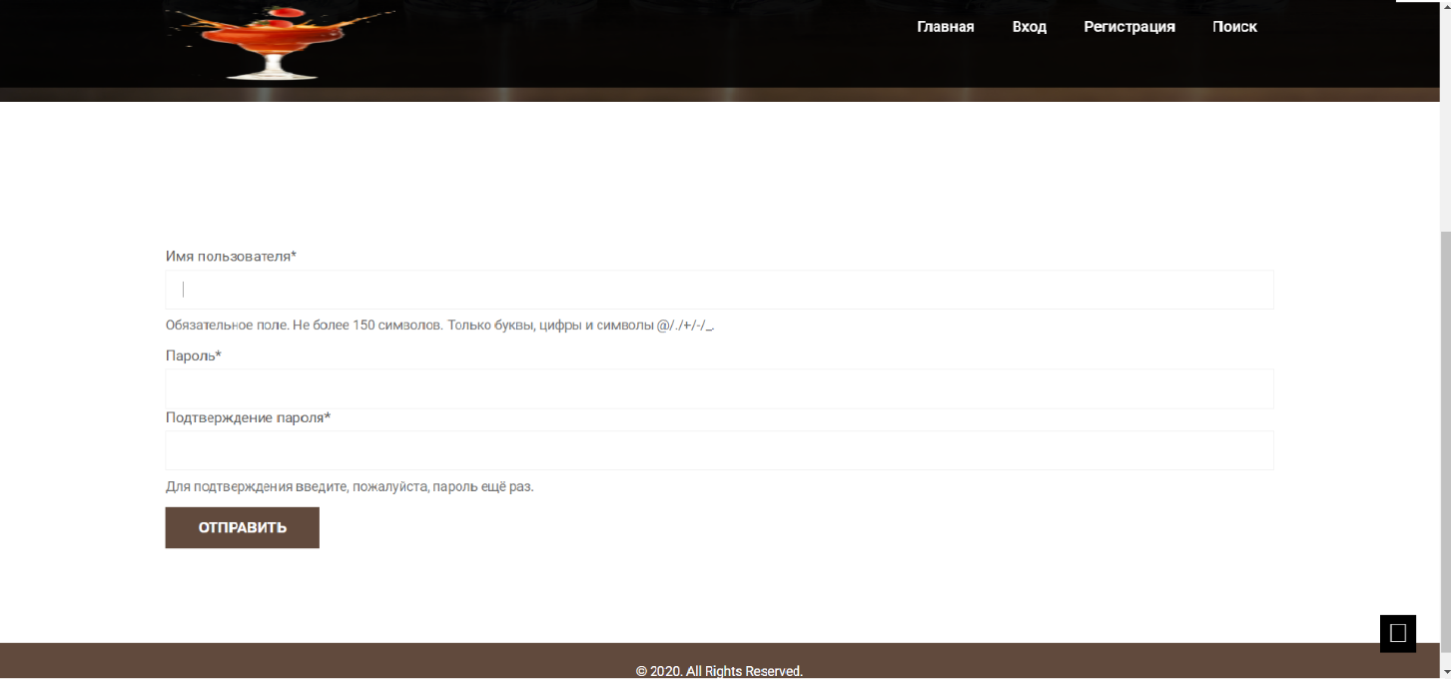
На стартовой странице можно увидеть навигационное меню, логотип, а также часть базы коктейлей.

## 4.2. Страница авторизации



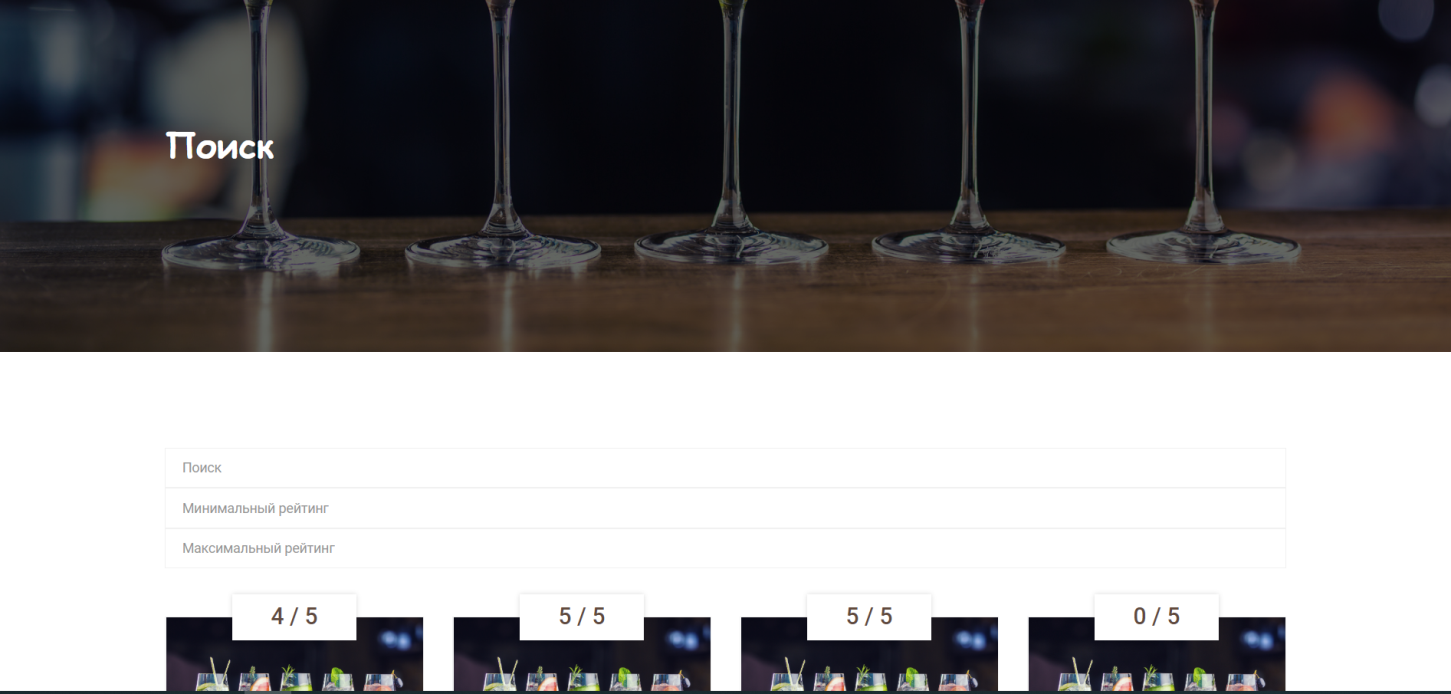
*Рисунок 25. Страница авторизации в Туса Juice’a*

## 4.3. Страница регистрации



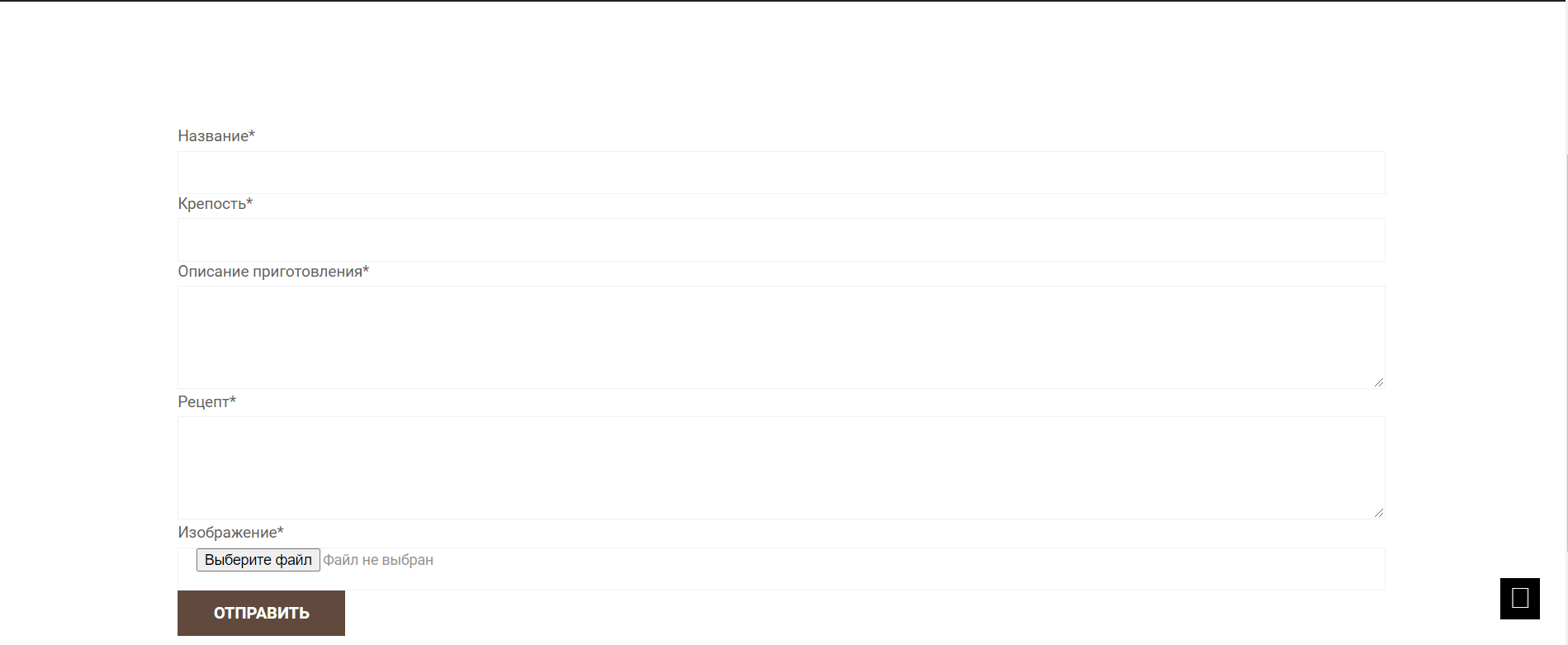
*Рисунок 26. Страница регистрации в Туса Juice’a*

## 4.4. Страница поиска



*Рисунок 27. Страница поиска коктейлей в Туса Juice’a*

## 4.5. Страница добавления коктейля



*Рисунок 28. Страница добавления коктейля в Туса Juice’a*

# Тестирование

Основные виды тестирования, которые были проведены:

* 1. Smoke testing – поверхностное тестирование системы на предмет работоспособности. В рамках этого тестирования были проверены возможности:
* запуска клиентского приложения;
* отображение пользовательского интерфейса;
* соединение и получение ответа от сервера.
  1. Sanitytesting – тестирование работы конкретных функций системы для доказательства того, что система работает согласно требованиям в Техническом Задании. В рамках данного теста были проверены функциональные возможности системы:
* добавление пользователя;
* авторизация пользователя;
* выход из системы;
* поиск коктейля;
* добавление данных о коктейле;
* изменение данных о коктейле;
* удаление данных о коктейле;
* возможность оценки коктейля.
  1. Negative testing – тестирование поведения системы, при попытке ввода некорректных данных. Было протестировано поведение системы, при попытке ввода некорректных данных в различных функциях. Тест-кейсы представлены в приложении 2.
  2. Usability testing – тестирование того, насколько легко конечный пользователь сможет освоить систему. В рамках этого тестирования были выбраны 3 человека, которые ранее не были знакомы с системой. Импредлагалось попробовать выполнить основные функции системы. С результатами тестирования можно ознакомиться в Приложении 2.

В результате тестирования оказалось 100 % пройденных тест-кейсов.

# Заключение

В результате работы было реализовано web-приложение, оптимизирующего процесс поиска инструкций изготовления коктейлей (Туса Juice’a), которое позволяет пользователю искать нужные коктейли, оценить существующие коктейли на сайте, а также добавлять свои коктейли в базу web-приложения. Стоит отметить, что приложение имеет визуально понятный интерфейс, который позволяет без затруднений пользоваться данным приложением.

В процессе реализации были выполнены следующие задачи:

1. Проведен анализ рынка с целью выявления достоинств и недостатков схожих по функционалу систем;

2. Спроектировано приложение с учетом информации, полученной ранее в ходе анализа;

3. Разработана Front-end часть приложения, находящаяся в браузере пользователя;

4. Разработана Back-end часть приложения, развернутая на удаленном сервере;

5. Разработана базы данных, расположенная на удаленном сервере;

6. Описан процесс разработки и результат.

# Приложение

## Приложение 1. Результаты тестирования

Таблица 1 – Результаты Smoke, sanity и negative тестирования.

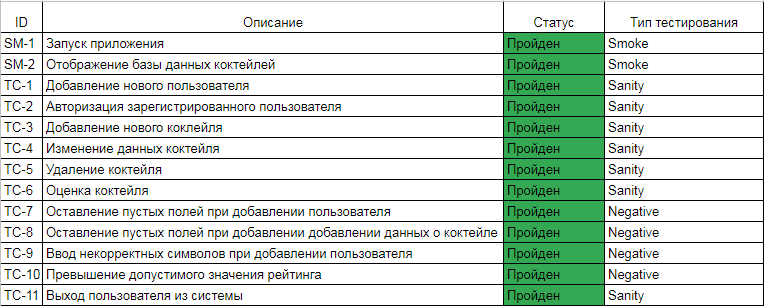


Таблица 2 – Результаты Usability тестирования.

|  |  |  |
| --- | --- | --- |
| Функция | Пользователь 1 | Пользователь 2 |
| Добавление нового пользователя | Пройден | Пройден |
| Добавление нового коктейля | Пройден | Пройден |
| Поиск коктейля | Пройден | Пройден |
| Изменение данных о коктейле | Пройден | Пройден |
| Оценка коктейля | Пройден | Пройден |
| Выход из системы | Пройден | Пройден |

Список использованных источников

1. Paul Barry Head First Python A Brain-Friendly Guide / Paul Barry – O’REILLY, 2017.
2. Платформа YOUTUBE: <https://www.youtube.com/>
3. Swagger – умная документация вашего RESTful web-API — обзор Junior back-end developer-а для новичков<https://habr.com/ru/post/434798/>
4. <https://fastapi.tiangolo.com/>