МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
“ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ”

Факультет *компьютерных наук*

Кафедра *программирования и информационных технологий*

LikeFood

Курсовой проект

09.03.04 Информационные системы и технологии
Программная инженерия в информационных системах

Допущен к защите

Обучающийся _____*М.С. Верещагина, 3 курс, д/о*

Обучающийся _____*В.С. Желтухин, 3 курс, д/о*

Обучающийся _____*Е.И. Олейник, 3 курс, д/о*

Руководитель _____*В.С. Тарасов, ассистент*

Руководитель _____*А.В. Нужных, ассистент*

Руководитель _____*В.А. Рыжков, ассистент*

Воронеж 2020

Содержание

Введение

По данным глобального исследования Nielsen, 67% потребителей в России активно следят за своим рационом, чтобы предотвратить различные болезни. Для того, чтобы исключить из своего употребления все вредные продукты, им приходится заходить на сайты с кулинарными рецептами в поисках вкусных и полезных блюд. Однако на большинстве кулинарных сайтов «диетические рецепты» - лишь один из разделов, соседствующий с калорийными «десертами» и сдобной «выпечкой».

Поэтому наша система полностью посвящена одной узконаправленной тематике - диетические рецепты, чтобы у пользователей нашего сайта не возникло соблазна приготовить и съесть что-нибудь вкусное и вредное, тем самым нарушив свой режим.

Все рецепты в систему будут добавлять приглашенные извне фуд-блогеры/диетологи/шеф-повара, досконально разбирающиеся в этой теме, и пользователям нашего сайта будет предоставлена возможность оценки этих рецептов.

Для того, чтобы уменьшить время, затрачиваемое на принятие решения о приготовлении того или иного блюда, пользователю будет предоставлена возможность фильтрации рецептов по имени автора, названию рецепта, категории блюда с сортировкой по рейтингу или дате добавления рецептов.

На основе рейтинга рецептов будет строиться рейтинг авторов, что позволит им соревноваться между собой за популярность и доверие читателей. Они смогут рекламировать в своих социальных сетях наш сайт и приглашать своих поклонников оценить их рецепты, а сторонние пользователи, не знакомые ранее с деятельностью данного автора смогут затем найти его в других социальных сетях и подписаться там на него.

Интересен наш сайт и с научной точки зрения, в особенности различным маркетологам и аналитикам. Можно отследить статистику самых популярных рецептов (отсортировав все рецепты по рейтингу) и выяснить, какие диетические блюда являются одними из самых популярных, чтобы затем включать в меню различных ресторанов и кафе. Также можно будет проанализировать, рецепты каких авторов пользуются наибольшей популярность и к кому люди охотнее прислушиваются в выборе рецептов.

1. Постановка задачи

Цель: перед нами стоит задача создания веб-приложения, предназначенного для формирования базы данных диетических рецептов от различных знаменитостей, специализирующихся в этой сфере, с возможность пользовательской оценки для построения рейтинга рецептов, на основании которого строится рейтинг авторов.

Задачи, решаемые приложением:

1. Формирование списка рецептов диетических блюд
2. Предоставление пользовательского рейтинга рецептов
3. Построение рейтинга авторов

2. Анализ предметной области

2.1 Глоссарий

Система - информационная система «LikeFood» с Web-интерфейсом, требования к которой указаны в данном документе.

Автор - авторизованный пользователь, обладающий правами читателя, а также правами создания и удаления (только своих) рецептов, участвующий в рейтинге авторов.

Читатель - авторизованный пользователь, не участвующий в рейтинге, обладающий правом просматривать и оценивать рецепты авторов.

Администратор - авторизованный пользователь, обладающий правами удаления любого рецепта, а также правом создавать аккаунты авторов.

Рецепт - запись, созданная автором, в которой описан процесс приготовления диетического блюда.

Блюдо - приготовленная пища, процесс создания которой описан в рецепте.

Топ авторов - рейтинг авторов, расположенных по убыванию общего количества оценок пользователей на их рецептах.

2.2 Анализ существующих решений

По информации сервиса Яндекс.Подбор слов ежемесячно ключевое слово “диетические рецепты” вводится в поисковый запрос более чем 142 903 раз. Если добавить к этому результаты поиска слова «диета» и «диетические блюда», то количество запросов будет в разы выше. Для сравнения: «рецепты десертов» вводят в поисковый запрос ежемесячно 77 294 раза, а «рецепты фаст-фуда» - 976 раз. Не стоит забывать и о других поисковых системах, которыми пользуются в нашей стране. Это говорит о том, что поиск диетических рецептов с помощью средств сети Интернет популярен в РФ.

Были проанализированы существующие сайты с диетическими рецептами в целях выявления их недочетов.

Одни из первых результатов в Яндексе представлены на рисунках 1-3.

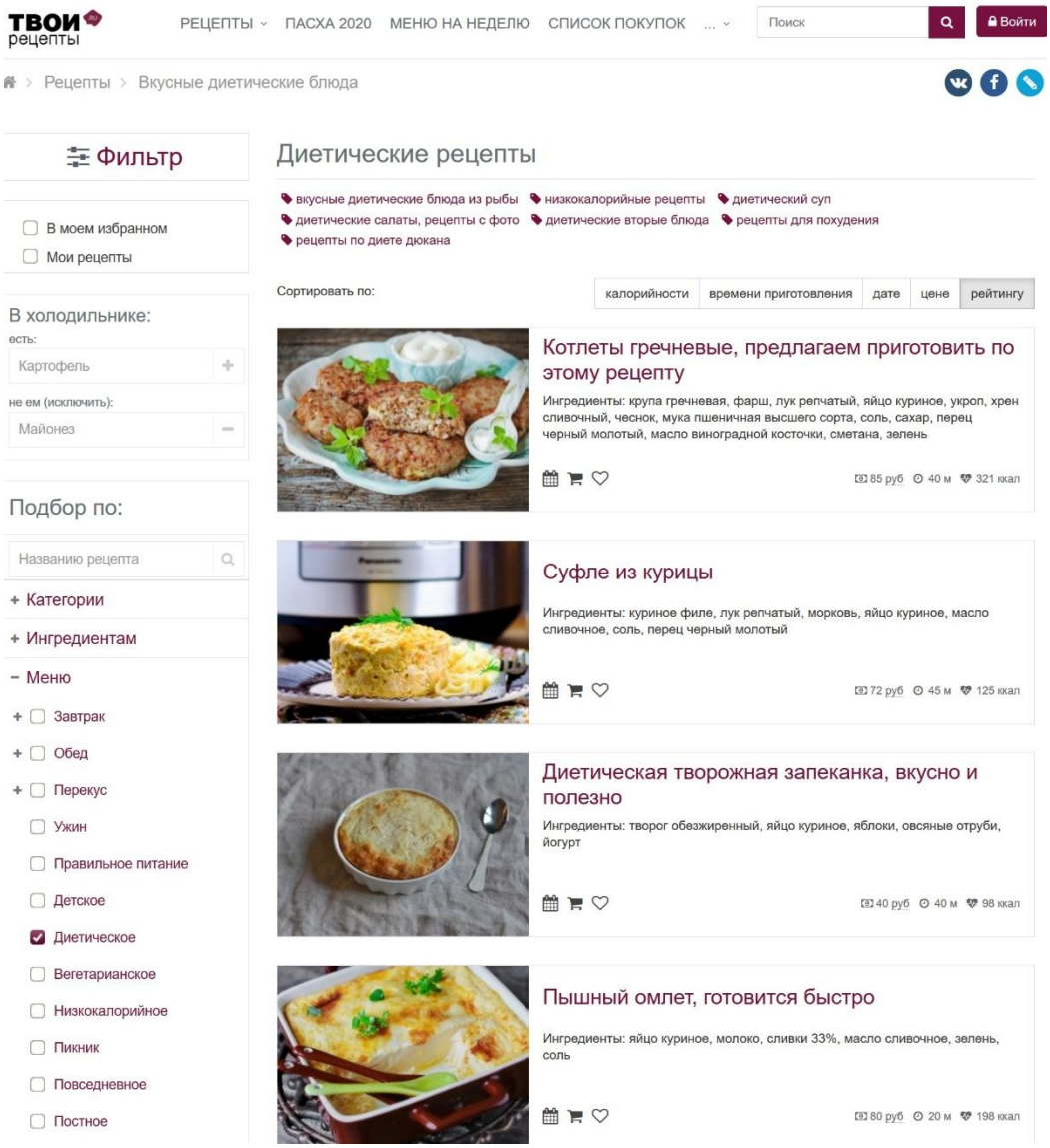

Рис.1. Внешний вид сайта «Твои рецепты».

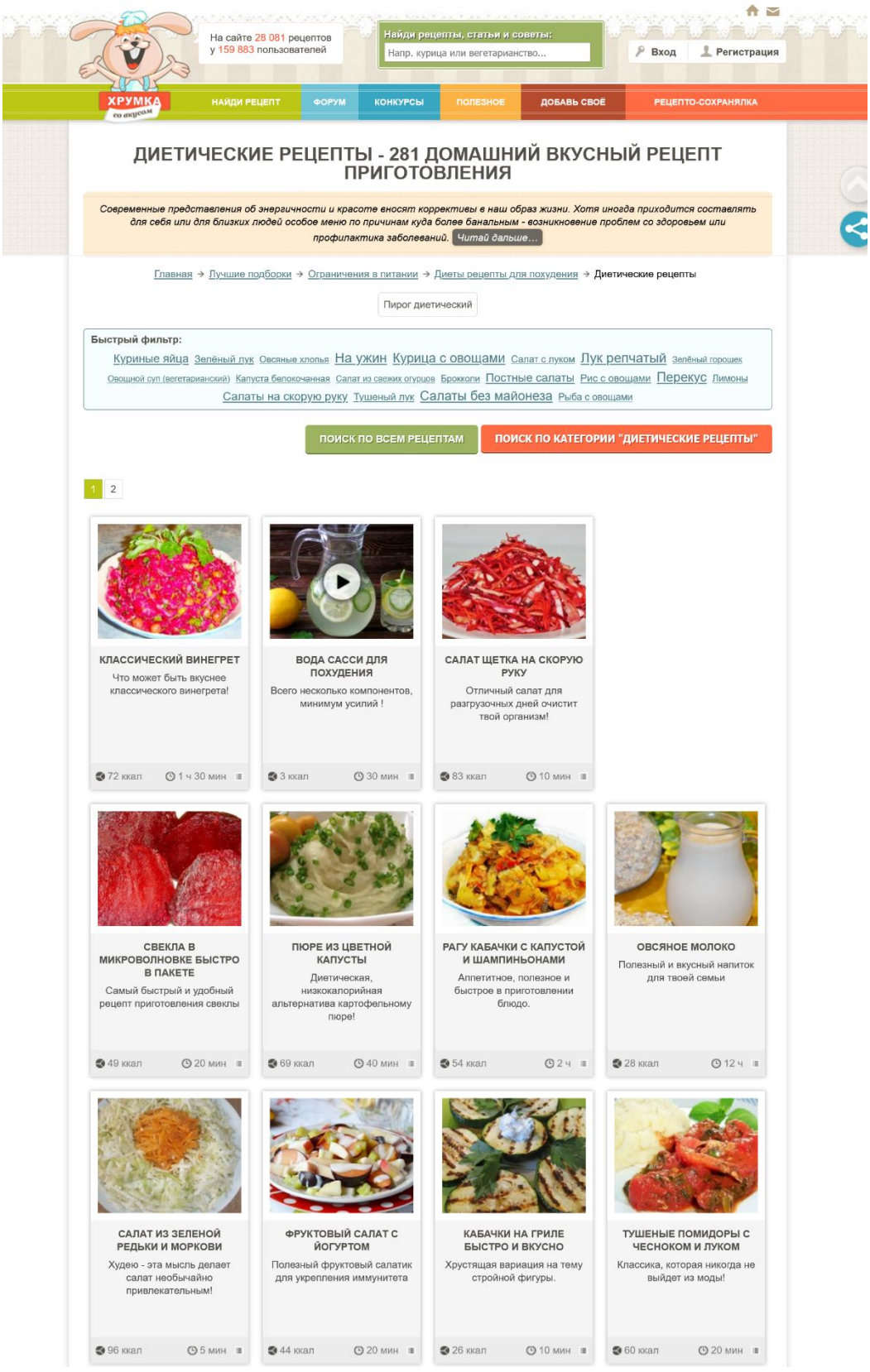

Рис.2. Внешний вид сайта «1000.menu».

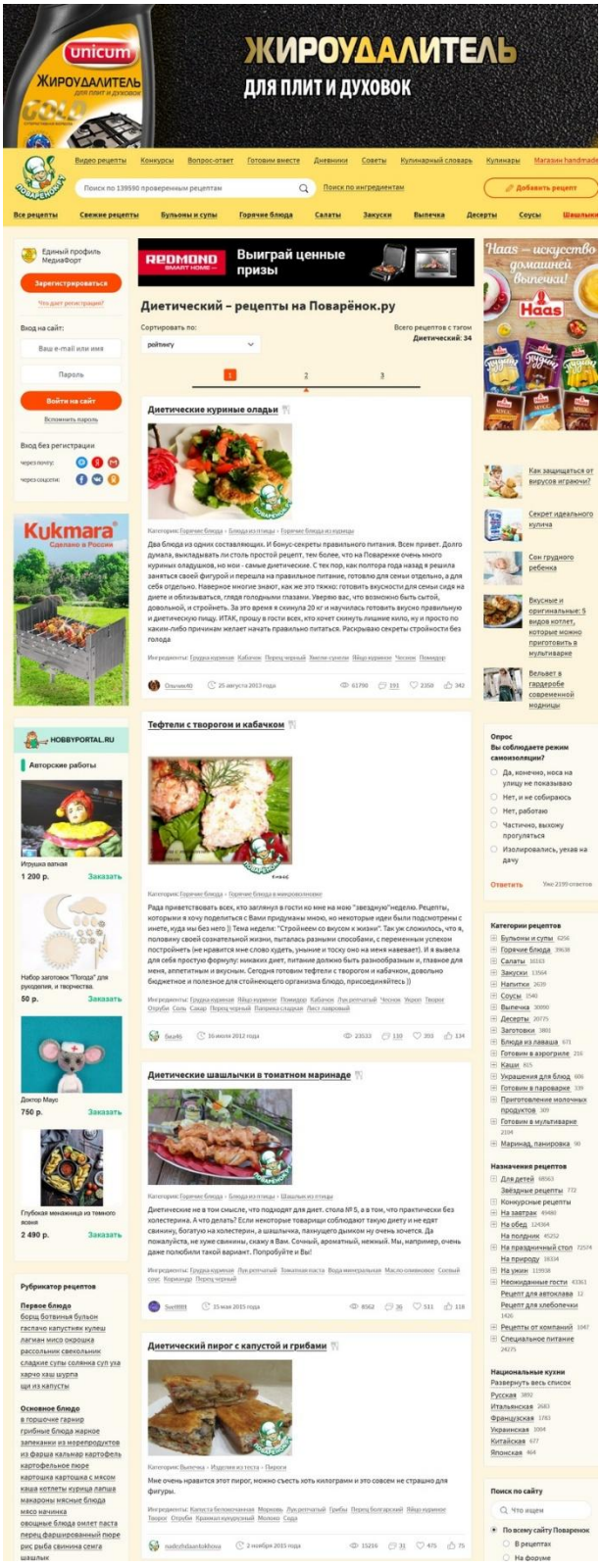

Рис.3. Внешний вид сайта «Поваренок».

Как было замечено в введении, «диетические рецепты» в данном случае являются лишь одни из разделов на сайте (в некоторых случаях еще и представленным малым количеством рецептов). Возможностью добавление рецептов обладает каждый зарегистрировавшийся пользователь, из-за чего некоторые рецепты становятся нечитабельными. Очень часто личность таких авторов остается неизвестна, поэтому пользователю очень тяжело определить можно ли доверять автору/рецепту. Ни один из этих сайтов не предоставляет возможность построения рейтинга авторов рецептов.

Таким образом, тяжело назвать эти сайты аналогами нашей системы.

2.3 Анализ задачи

2.3.1 Варианты использования системы

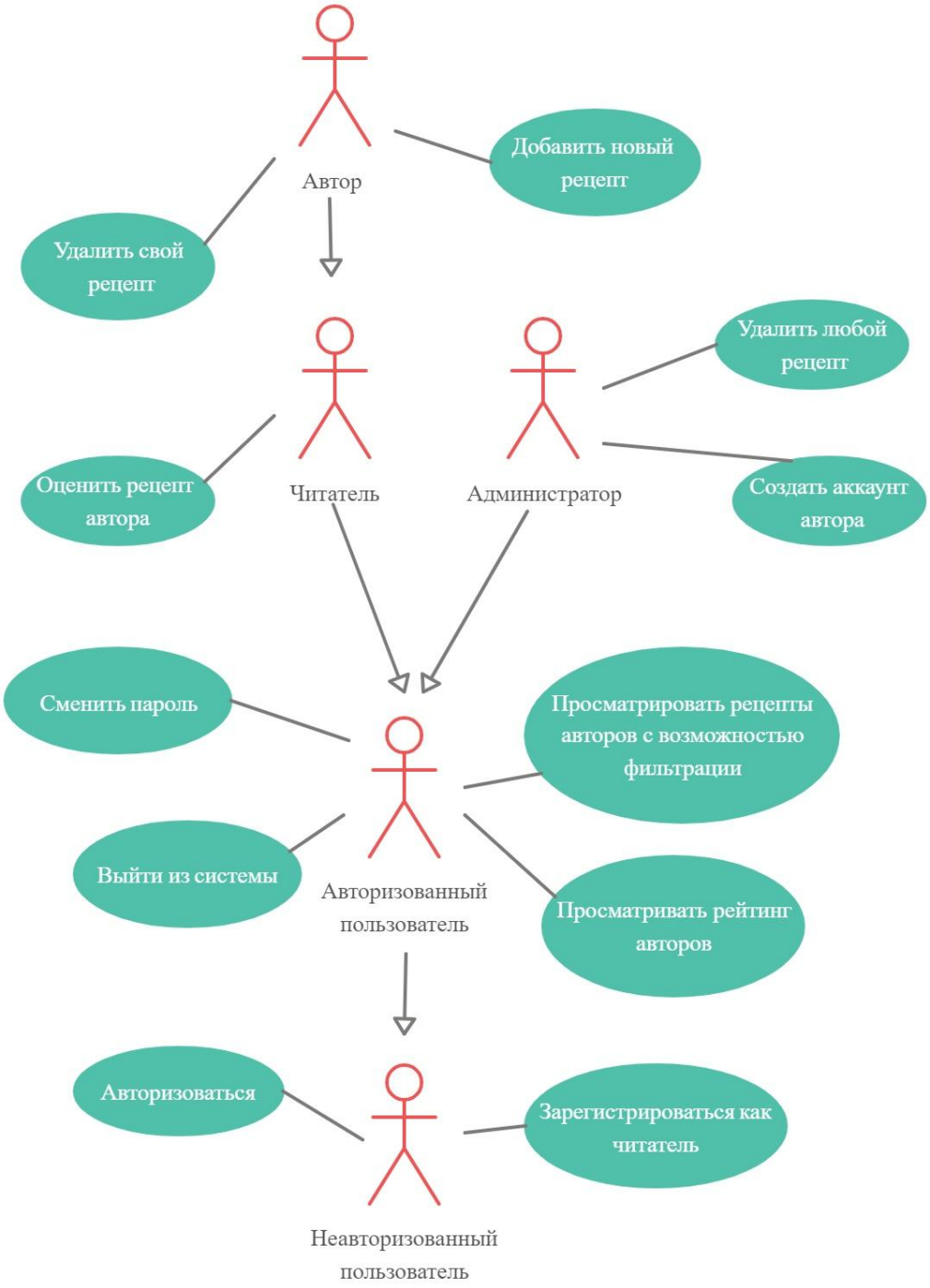

Рис.4. Диаграмма прецедентов.

При взаимодействии с приложением у пользователя, входящего в определенную группу доступа, есть определенный список возможностей, который более наглядно изображен на рисунке 4.

В Системе выделяется четыре группы пользователей — Неавторизованный (анонимный) пользователь, Читатель, Автор, Администратор.

Приложение должно предоставлять следующие возможности:

Неавторизованный пользователь может зарегистрироваться (будет создан новый аккаунт с правами читателя, зарегистрировать нового автора в Системе может только администратор) и авторизоваться в Системе.

Читатель может просматривать рейтинг авторов, кулинарные рецепты авторов (с возможностью их фильтрации по названию, автору, категории и сортировки по рейтингу или дате добавления), а также с возможностью оценки рецептов, сменой пароля и выхода из системы.

Автор - продвинутый читатель, поэтому он обладает всеми функциями читателя и так же может создавать новые рецепты, удалять те рецепты, автором которых он является, просматривать свои рецепты.

Администратор Системы не участвует в построении рейтинга авторов и не имеет права создавать/оценивать рецепты. Он может регистрировать новых авторов (т.е. создавать для них аккаунты), просматривать рейтинг авторов, кулинарные рецепты авторов (с возможностью их фильтрации по названию, автору, категории и сортировки по рейтингу или дате добавления), менять свой пароль, удалять любые рецепты в Системе, добавленные авторами.

2.3.2 ER-диаграмма

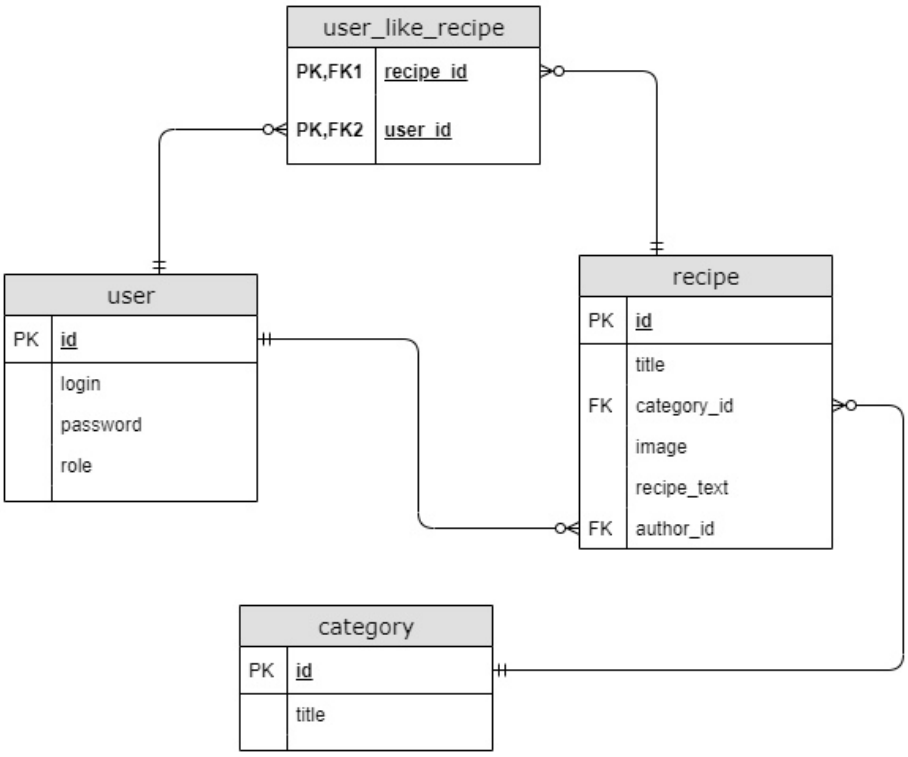

Рис.5. ER-диаграмма.

На рисунке 5 представлена схемы базы данных для разработанного приложения.

Данная база данных состоит из 4 таблиц:

- user
- recipe
- category
- user_like_recipe

В таблице "user" хранится информация о пользователях. Она содержит в себе следующие атрибуты:

- id - уникальный идентификатор пользователя, который является первичным ключом
- login - логин пользователя
- password - пароль пользователя

- role - роль пользователя (администратор, писатель, читатель)

В таблице "recipe" хранится информация о рецептах. Она содержит в себе следующие атрибуты:

- id - уникальный идентификатор рецепта, который является первичным ключом
- title - название рецепта
- category_id - уникальный идентификатор категории к которой относится рецепт, является вторичным ключом, ссылающимся на таблицу "category"
- image - ссылка на изображение, хранящееся на сервере приложения и сопровождающее рецепт
- recipe_text - текст рецепта
- author_id - уникальный идентификатор автора рецепта, является вторичным ключом, ссылающимся на таблицу "user"

В таблице "category" хранится информация о категориях, к которым относятся рецепты. Она содержит в себе следующие атрибуты:

- id - уникальный идентификатор категории к которой относится рецепт, является первичным ключом
- title - название категории

Между таблицами "user" и "recipe" имеется связь один-ко-многим. У каждого рецепта обязательно есть только один автор (связь обязательна), но у каждого автора может быть один и более рецептов, а может ни быть не одного (связь необязательная).

Между таблицами "user" и "category" имеется связь один-ко-многим. Каждый рецепт обязательно относится только к одной категории (связь обязательна), но в каждой категории может содержаться один и более рецептов, а может не быть ни одного (связь необязательная).

Также между таблицами "user" и "recipe" имеется еще одна связь многие-ко-многим, с помощью который реализуется функция оценки рецептов пользователями. Пользователь может оценить сколько угодно рецептов, а может ни одного (связь необязательная). Рецепт может быть оценен сколько угодно раз разными пользователями, а может - ни одного (связь необязательная).

Связь многие-ко-многим создается с помощью трех таблиц: две исходные таблицы ("user", "recipe") и одна соединительная таблица ("user_like_recipe").

Первичный ключ соединительной таблицы "user_like_recipe" – составной. Она состоит из двух полей, двух внешних ключей, которые ссылаются на первичные ключи таблиц "user" и "recipe":

- recipe_id
- user_id

2.3.3 Диаграмма классов

Рис.6. Диаграмма классов.

Рассмотрим классы, представляющие бизнес-логику, которые мы будем использовать для реализации нашего проекта.

Диаграмма классов представлена на рисунке 6.

Классы User, Recipe и Category наследуются от базового класса Model и сопоставляются с одноименными таблицами в нашей базе данных (user, recipe, category).

Классы UserLogic, RecipeLogic, CategoryLogic каждый содержат в себе атрибут entityClass, ссылающийся на соответствующий класс (User, Recipe, Category).

Все функции в них представляют соответствующие запросы к базе данных.

В классе CategoryLogic имеются следующие методы:

- __init__(self) - конструктор класса
- find_by_id(self, id) - находит по переданному id категории соответствующую категорию в базе данных
- get_categories(self) - возвращает все категории

В классе UserLogic следующие методы:

- __init__(self) - конструктор класса
- add_user(self, login, password, role) - добавляет нового пользователя в базу данных с заданными атрибутами и сохраняет изменения в базе данных
- find_by_id(self, id) - находит пользователя по переданному id в базе данных
- get_all_authors(self) - возвращает все записи из таблицы пользователей в базе данных с ролью автора
- find_by_login(self, login) - находит пользователя по переданному login в базе данных

- check_password(self, user: User, password) - проверяет переданный пароль на соответствие записанному в базе данных
- change_password(self, user: User, new_password) - изменяет пароль пользователя и сохраняет изменения в базе данных

В классе RecipeLogic имеются следующие методы:

- __init__(self) - конструктор класса
- add_recipe(self, title, category_id, file_path, recipe_text, author_id) - добавляет новый рецепт в базу данных с заданными атрибутами и сохраняет изменения в базе данных
- get_recipes(self) - получает все рецепты из базы данных
- get_recipe_likes(self, id) - возвращает количество общее количество оценок на рецепте по его id
- get_recipe_like_by_user(self, user_id, recipe_id) - возвращает информацию о: оценивал ли пользователь уже этот рецепт ранее или еще нет
- put_like(self, user: User, recipe:Recipe) - добавляет запись в таблицу user_like_recipe и сохраняет изменения в базе данных
- delete_like(self, user: User, recipe:Recipe) - удаляет запись из таблицы user_like_recipe и сохраняет изменения в базе данных
- get_author_recipes(self, author_id) - возвращает все записи из таблицы рецептов, в которых значение в столбце author_id соответствует переданному атрибуту
- find_recipe_by_id(self, id) - возвращает запись рецепта с переданным id
- filter_recipes_author(self, author_login, name, category, sorting) - фильтрация по имени автора

- filter_recipes_name(self, author_login, name, category, sorting) - фильтрация по названию рецепта
- filter_recipes_category(self, author_login, name, category, sorting) - фильтрация по категории
- filter_recipes_sorting(self, author_login, name, category, sorting) - фильтрация по сортировке рецептов
- filter_my_recipes(self, author_login, name, category, sorting, user: User) - фильтрация, доступная только для авторов среди их рецептов
- delete_recipe(self, id) - удаление рецепта по переданному id и сохранение изменений в базе данных

Класс LikeFood - главный класс, выступающий в роли фасада. Связывает все остальные классы воедино.

Он имеет следующие атрибуты:

- user_logic - экземпляр класса RecipeLogic, создающийся в конструкторе класса
- recipe_logic - экземпляр класса UserLogic, создающийся в конструкторе класса
- category_logic - экземпляр класса CategoryLogic, создающийся в конструкторе класса

Также в классе LikeFood определены следующие методы:

- __init__(self) - конструктор класса
- login(self, login) - функция авторизации пользователя в системе
- register(self, login, password, role) - функция создания нового аккаунта читателя или автора

- gen_password(self, length=8, method=["lowercase", "uppercase", "digits", "punctuation"]) - метод генерации случайного пароля с заданными характеристиками (для добавления автора)
- show_top_raiting(self) - возвращает рейтинг пользователей
- show_recipes(self, value) - возвращает все рецепты
- check_user(self, login) - проверка пользователя с таким логином на существование (используется для регистрации и авторизации)
- get_recipe_info(self, recipe_id) - возвращает информацию о рецепте по id для его отображения
- add_like_to_recipe(self, recipe_id) - добавляет оценку на рецепт
- delete_like_from_recipe(self, recipe_id) - убирает поставленную оценку с рецепта
- delete_recipe(self, recipe_id) - функция удаления рецепта
- get_filter_recipes(self, author_login, name, category, sorting, isAuthor) - функция выбора фильтрации
- create_recipe(self, title, category_id, image, recipe_text) - функция создания рецепта
- allowed_file(self, filename) - возвращает разрешенные форматы для загрузки картинки рецепта на сервер
- show_categories(self) - функция, возвращающая все категории для отображения
- change_password(self, login, new_password) - функция смены пароля
- current_user_role(self) - возвращает роль авторизованного пользователя
- current_user_is_authenticated(self) - возвращает информацию о том, авторизован ли пользователь в системе

- logout(self) - выход текущего пользователя из системы

2.3.4 Диаграмма объектов

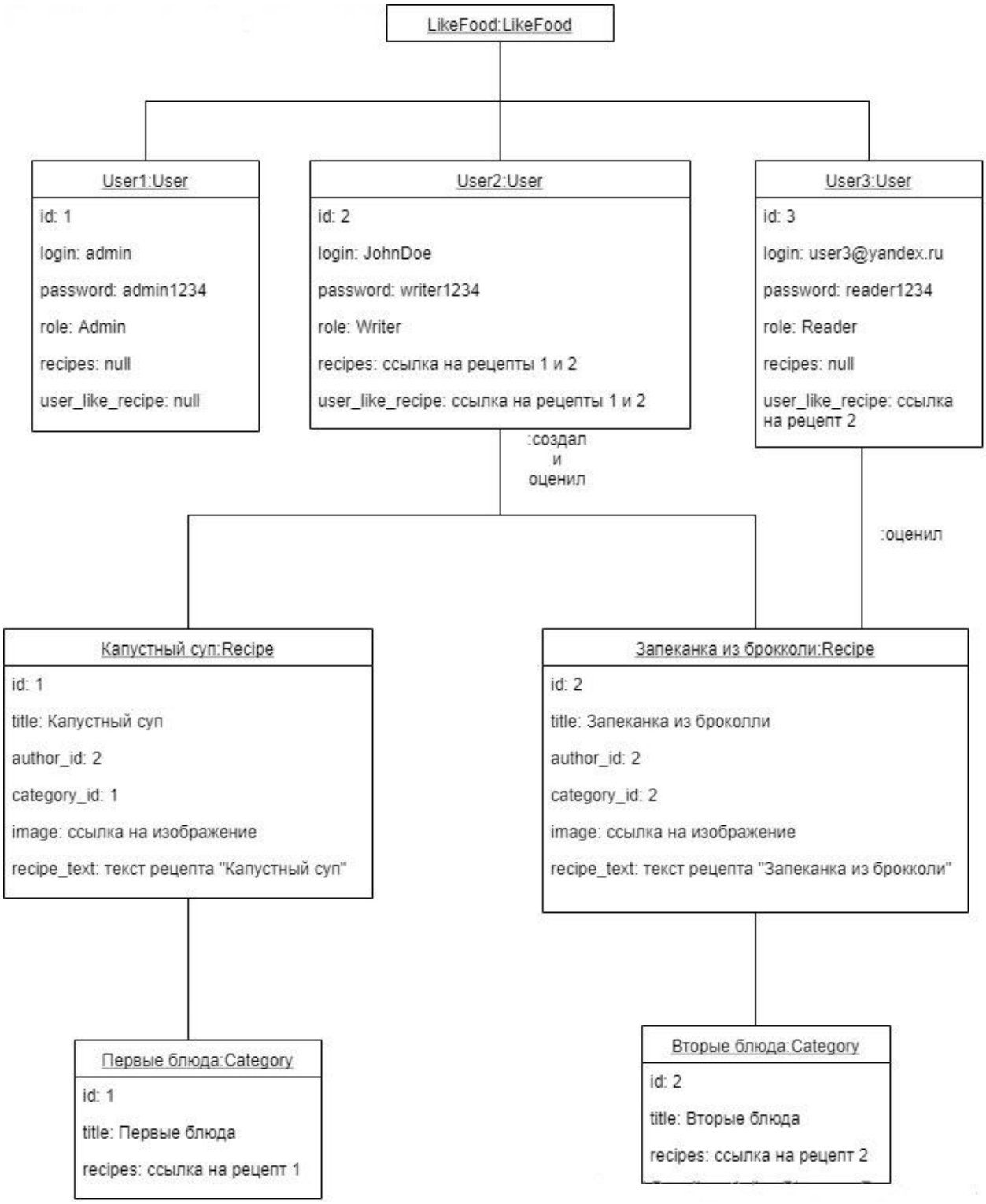

Рис.7. Диаграмма объектов.

Диаграмма объектов показывает множество объектов - экземпляров классов и отношений между ними в некоторый момент времени. То есть диаграмма объектов — это своего рода снимок состояния системы в определенный момент времени, показывающий множество объектов, их состояния и отношения между ними в данный момент.

В момент времени, зафиксированный на рисунке 7, у нас в системе находится три экземпляра класса User со следующим распределением ролей: User1 - администратор, User2 - автор и User3 - читатель. User2 добавил два рецепта в систему: капустный суп и запеканка из брокколи. Капустный суп относится к категории "Первые блюда", запеканка - "Вторые блюда". User3 оценил блюдо "Запеканка из брокколи".

2.3.5 Диаграммы последовательностей и взаимодействий

Ниже мы рассмотрим, как разнообразные компоненты системы взаимодействуют друг с другом, их отношения друг к другу, в том числе и сообщения, которым они обмениваются.

На рисунке 8 показана диаграмма коммуникации, на которой явно указываются отношения между объектами при авторизации пользователей в приложении, а на рисунке 9 изображена диаграмма последовательности, на которой изображено упорядоченное во времени взаимодействие объектов.

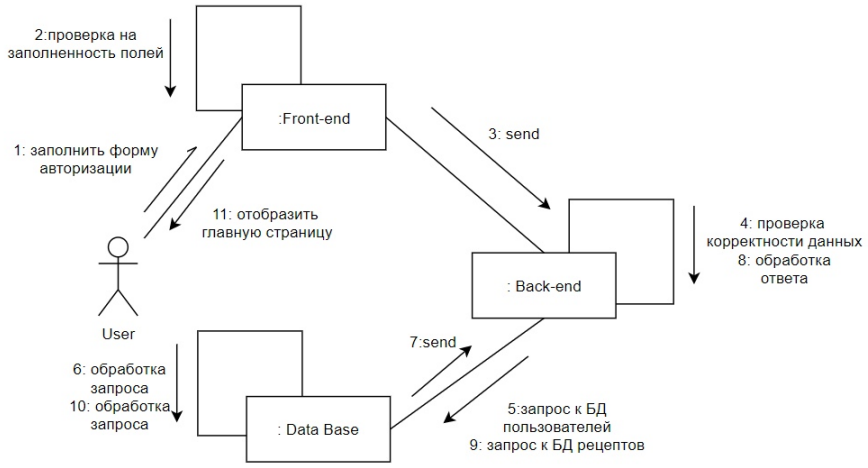

Рис. 8. Диаграмма коммуникаций для авторизации пользователя

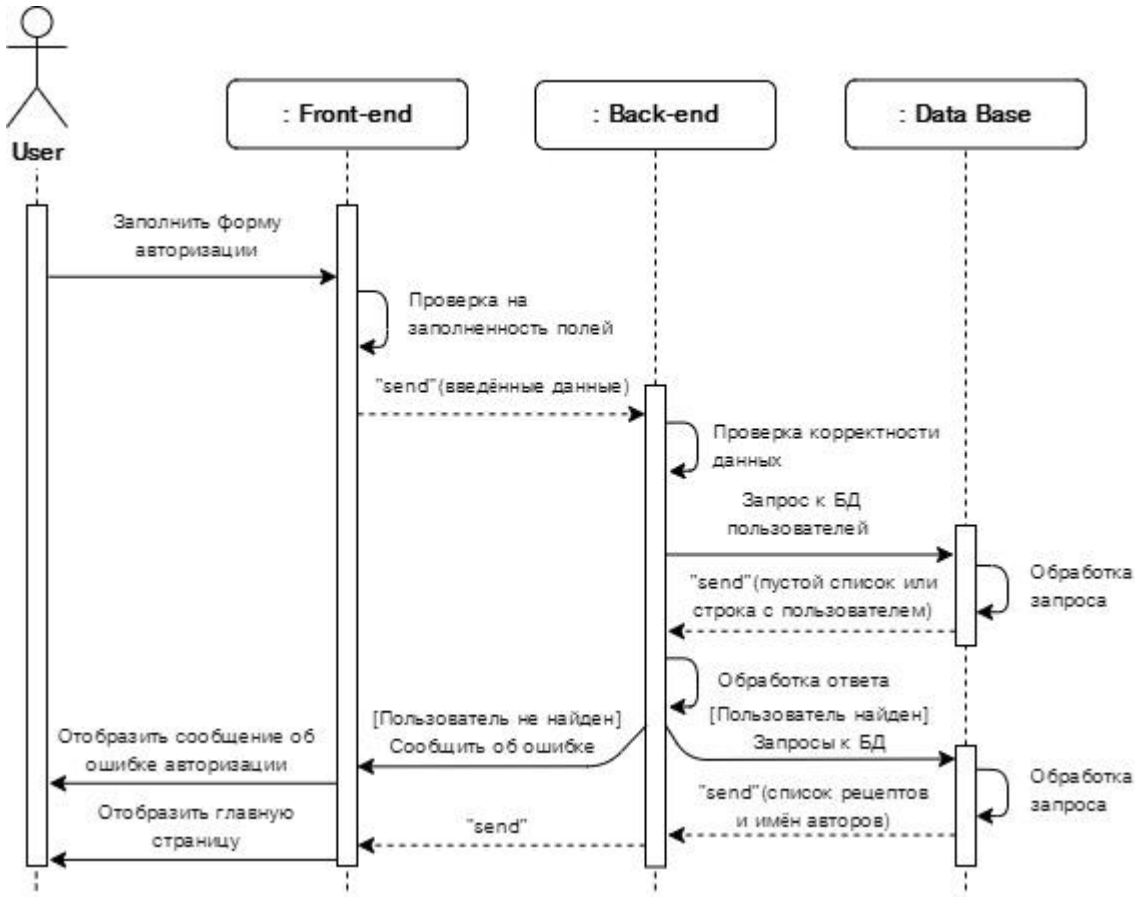

Рис. 9. Диаграмма последовательностей для авторизации пользователя

На рисунке 10 показана диаграмма коммуникации, на которой явно указываются отношения между объектами при регистрации читателей в приложении, а на рисунке 11 изображена диаграмма последовательности, на которой изображено упорядоченное во времени взаимодействие объектов.

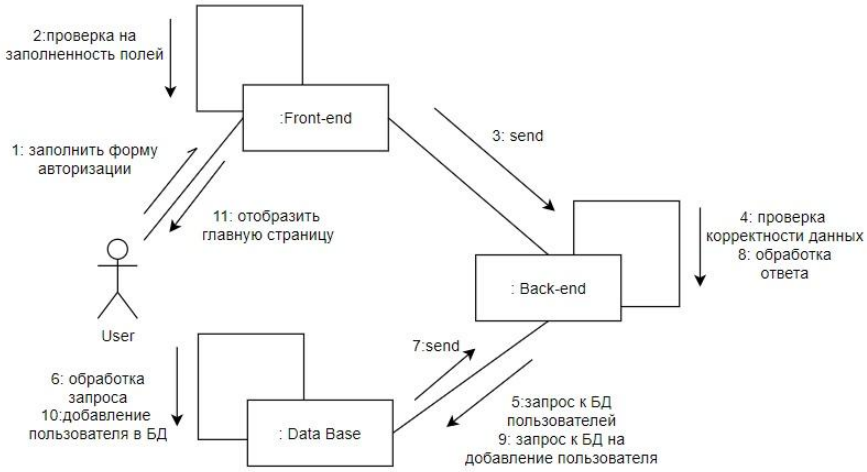

Рис. 10. Диаграмма коммуникаций для регистрации читателей

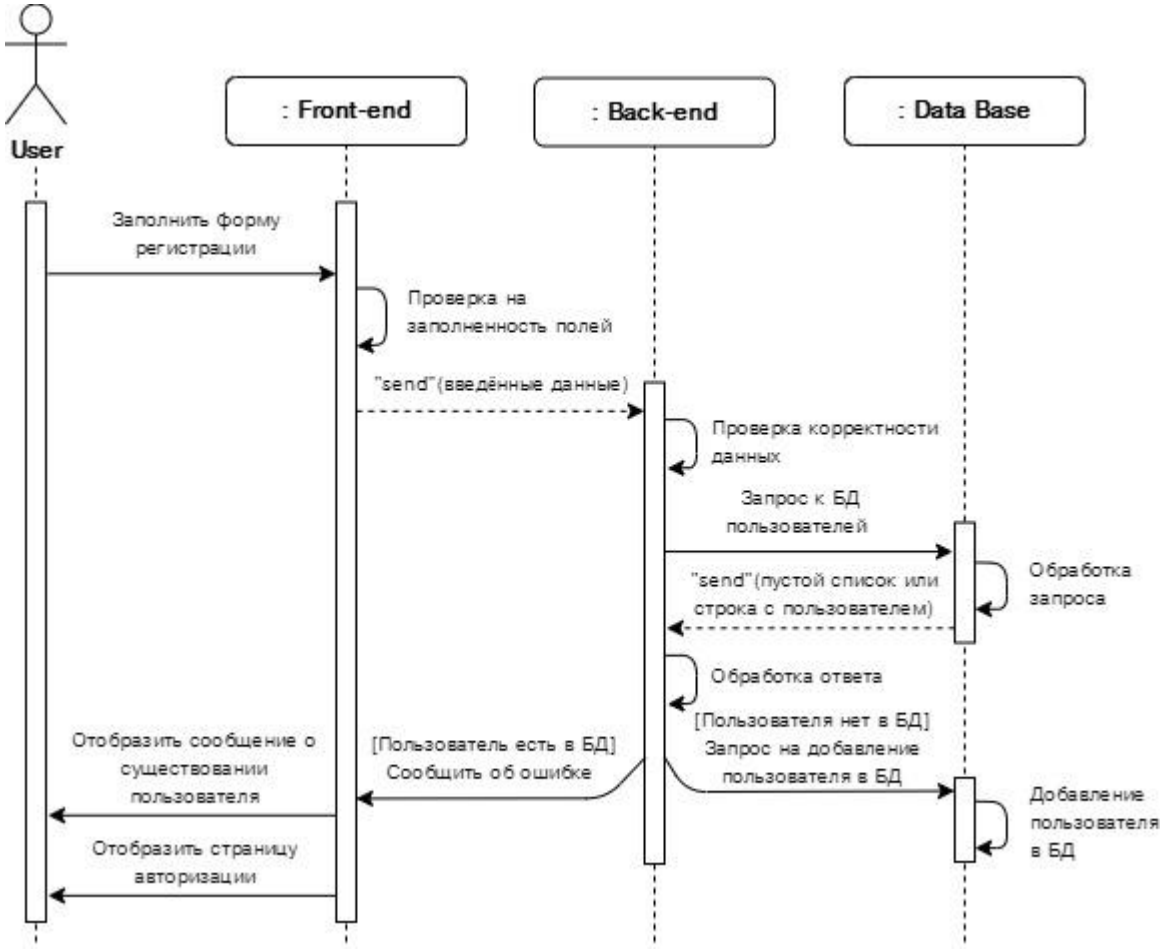

Рис. 11. Диаграмма последовательностей для регистрации читателей

На рисунке 12 показана диаграмма коммуникации, на которой явно указываются отношения между объектами при оценивании рецепта пользователем (читателем или автором) в приложении, а на рисунке 13 изображена диаграмма последовательности, на которой изображено упорядоченное во времени взаимодействие объектов.

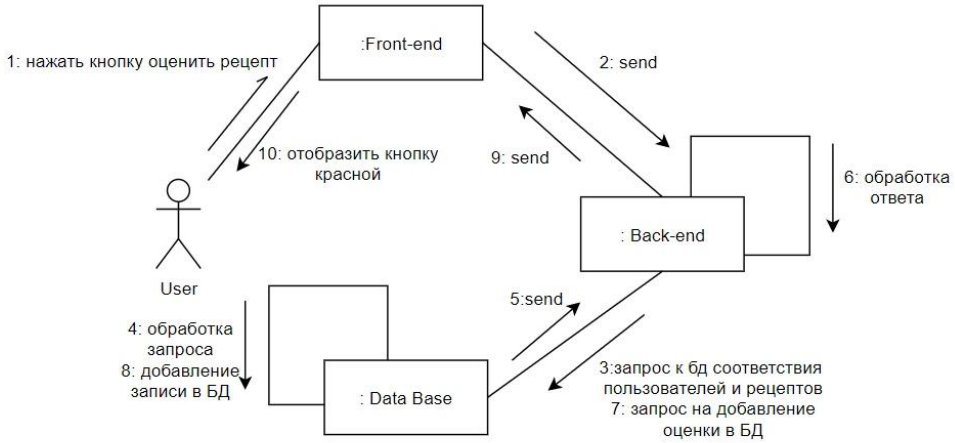

Рис. 12. Диаграмма коммуникаций для оценивания рецепта

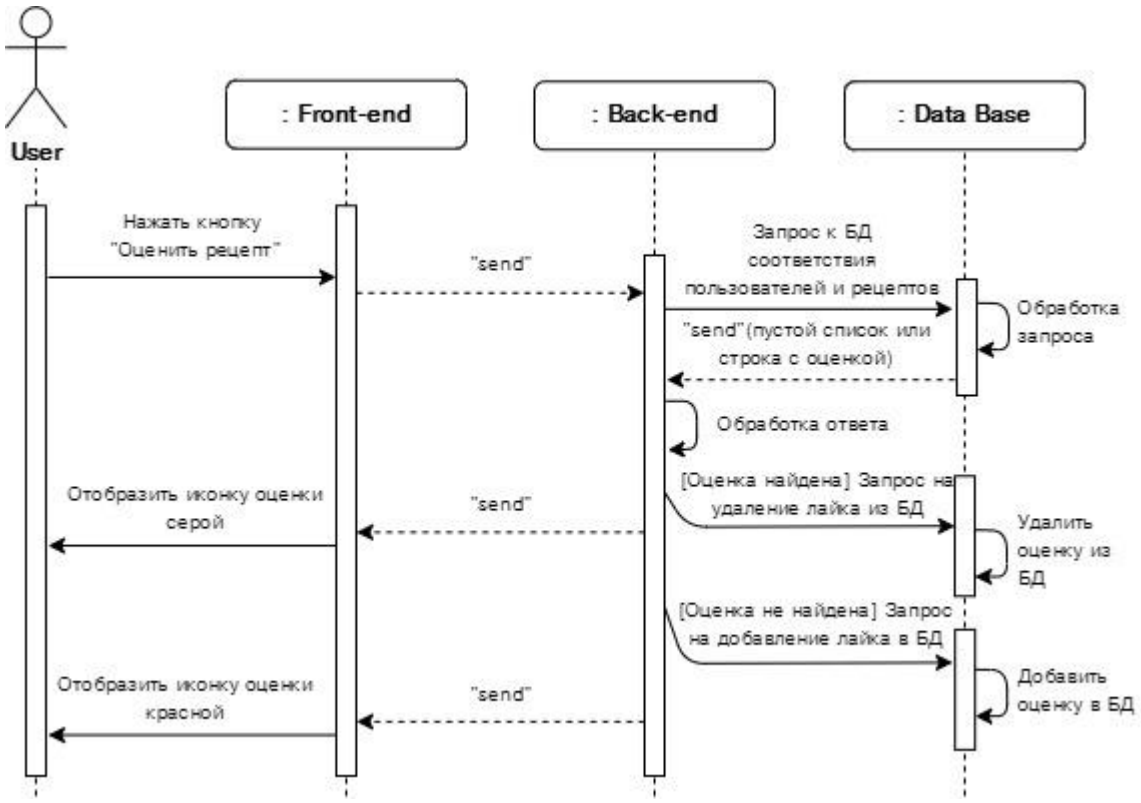

Рис. 13. Диаграмма последовательностей для оценивания рецепта

На рисунке 14 показана диаграмма коммуникации, на которой явно указываются отношения между объектами при добавлении рецепта пользователем (автором) в приложение, а на рисунке 15 изображена диаграмма последовательности, на которой изображено упорядоченное во времени взаимодействие объектов.

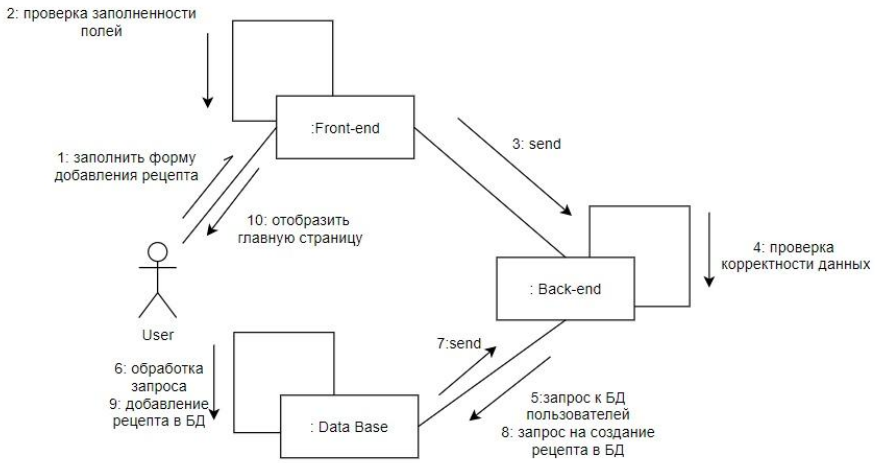

Рис. 14. Диаграмма коммуникаций для добавления рецепта

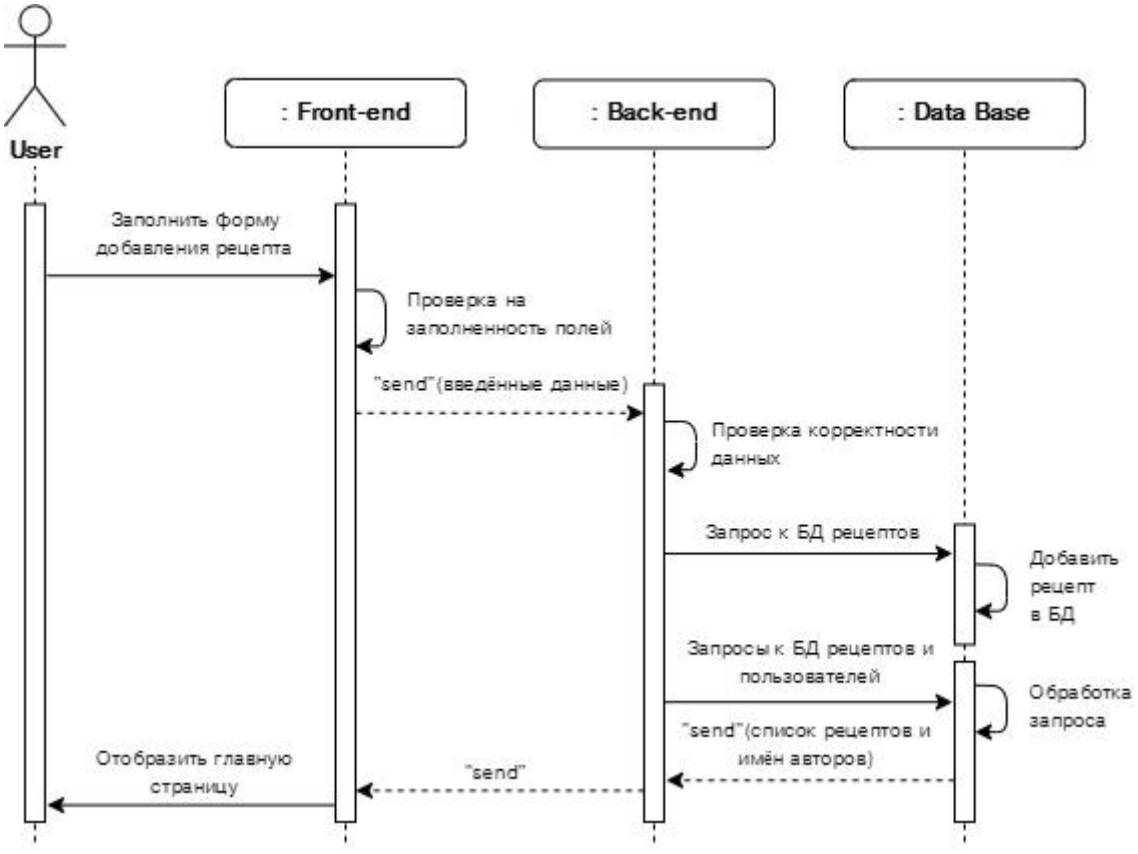

Рис. 15. Диаграмма последовательностей для добавления рецепта

На рисунке 16 показана диаграмма коммуникации, на которой явно указываются отношения между объектами при удалении рецепта пользователям в приложении, а на рисунке 17 изображена диаграмма последовательности, на которой изображено упорядоченное во времени взаимодействие объектов.

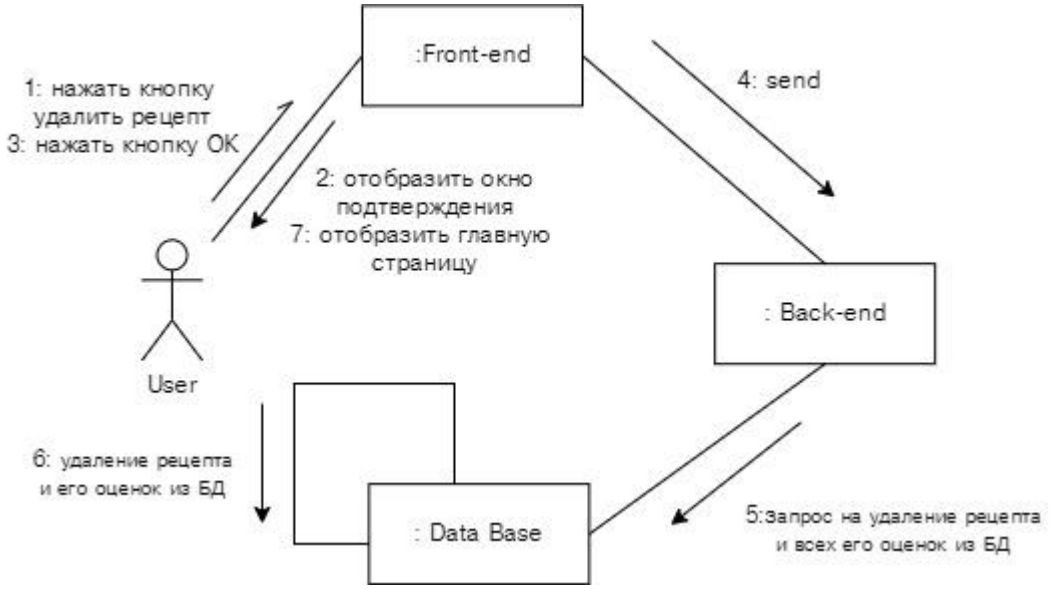

Рис. 16. Диаграмма коммуникаций для удаления рецепта

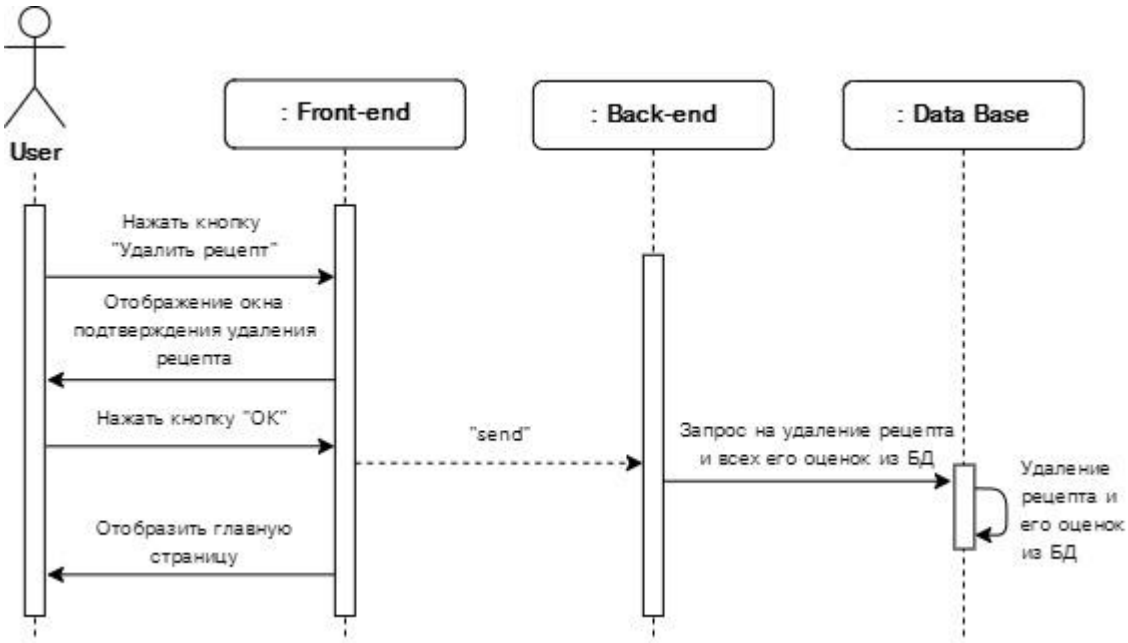

Рис. 17. Диаграмма последовательностей для удаления рецепта

На рисунке 18 показана диаграмма коммуникации, на которой явно указываются отношения между объектами при отображении топа авторов в приложении, а на рисунке 19 изображена диаграмма последовательности, на которой изображено упорядоченное во времени взаимодействие объектов.

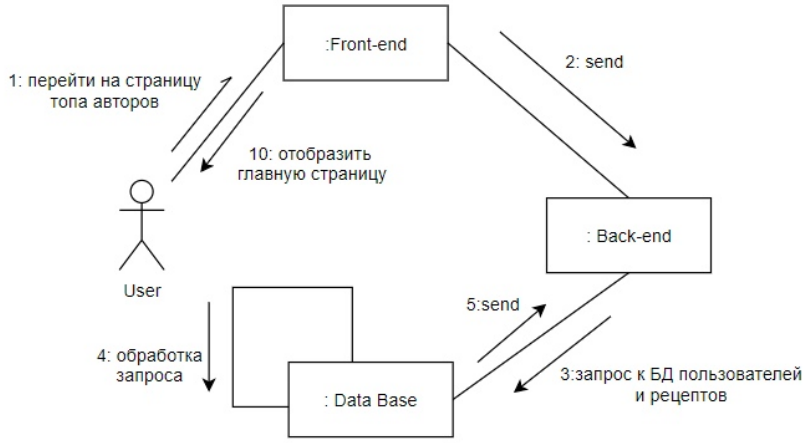

Рис. 18. Диаграмма коммуникаций для отображения топа авторов

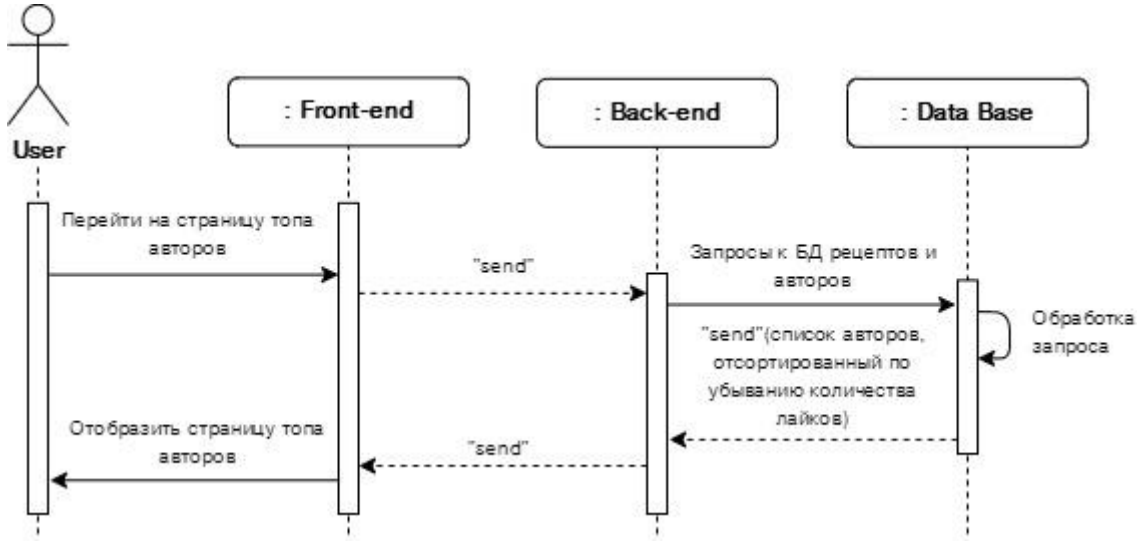

Рис. 19. Диаграмма последовательностей для отображения топа авторов

На рисунке 20 показана диаграмма коммуникации, на которой явно указываются отношения между объектами при добавлении авторов администратором в приложении, а на рисунке 21 изображена диаграмма

последовательности, на которой изображено упорядоченное во времени взаимодействие объектов.

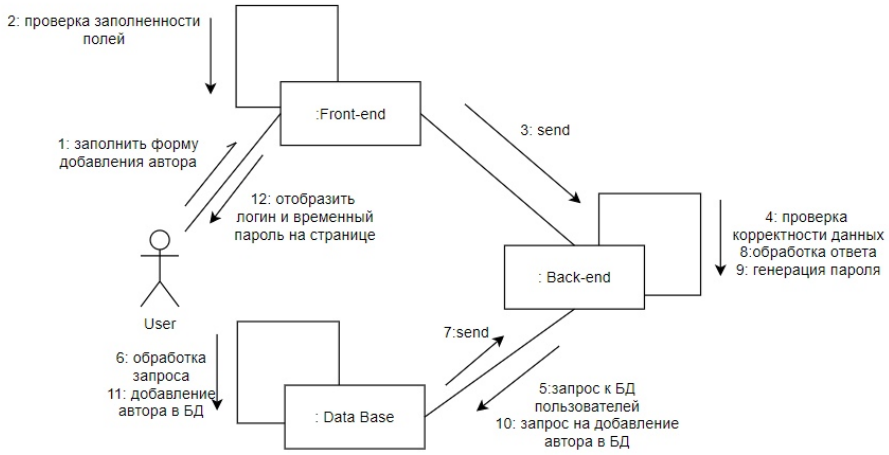

Рис. 20. Диаграмма коммуникаций для добавления авторов администратором

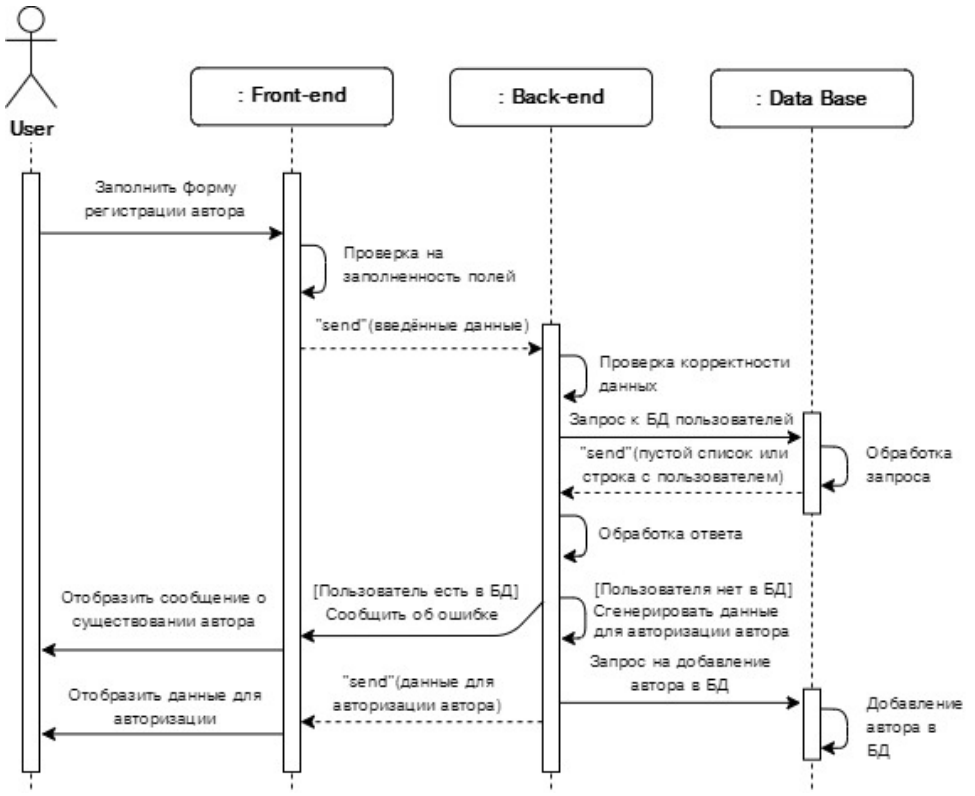

Рис. 21. Диаграмма последовательностей для добавления авторов администратором

2.3.6 Диаграммы состояний

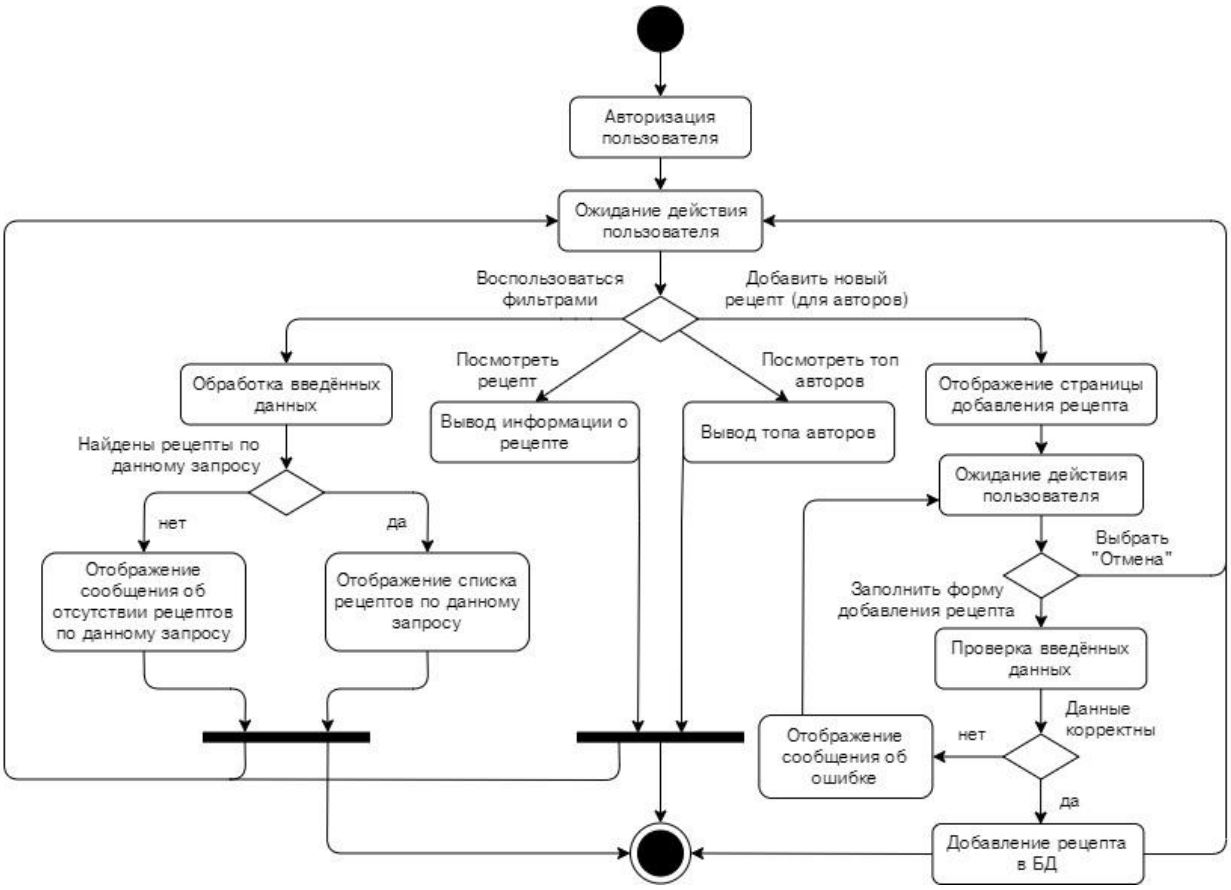

Рис. 22. Общая диаграмма состояний

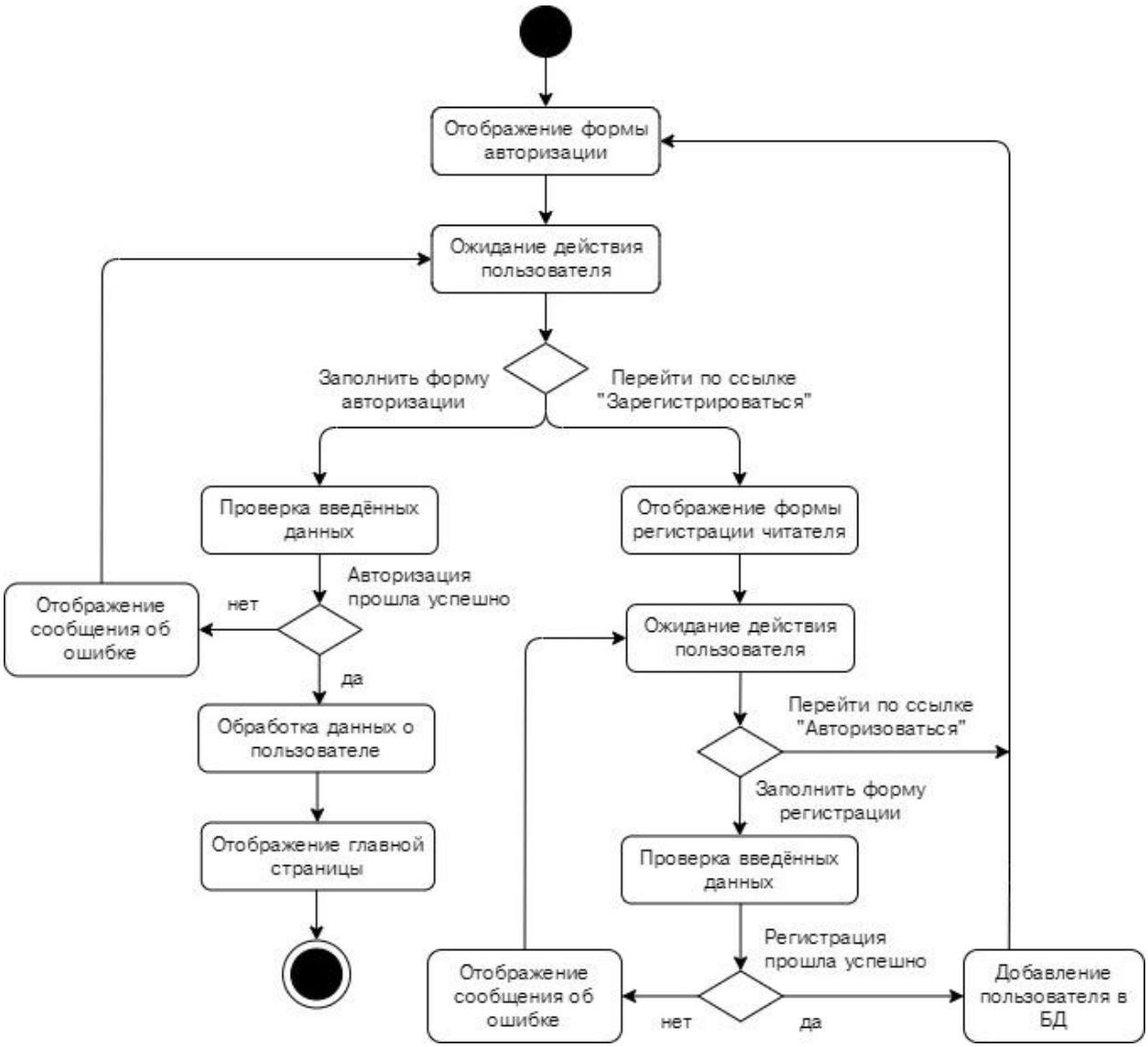

Рис. 23. Диаграмма состояний для процесса авторизации и регистрации читателя

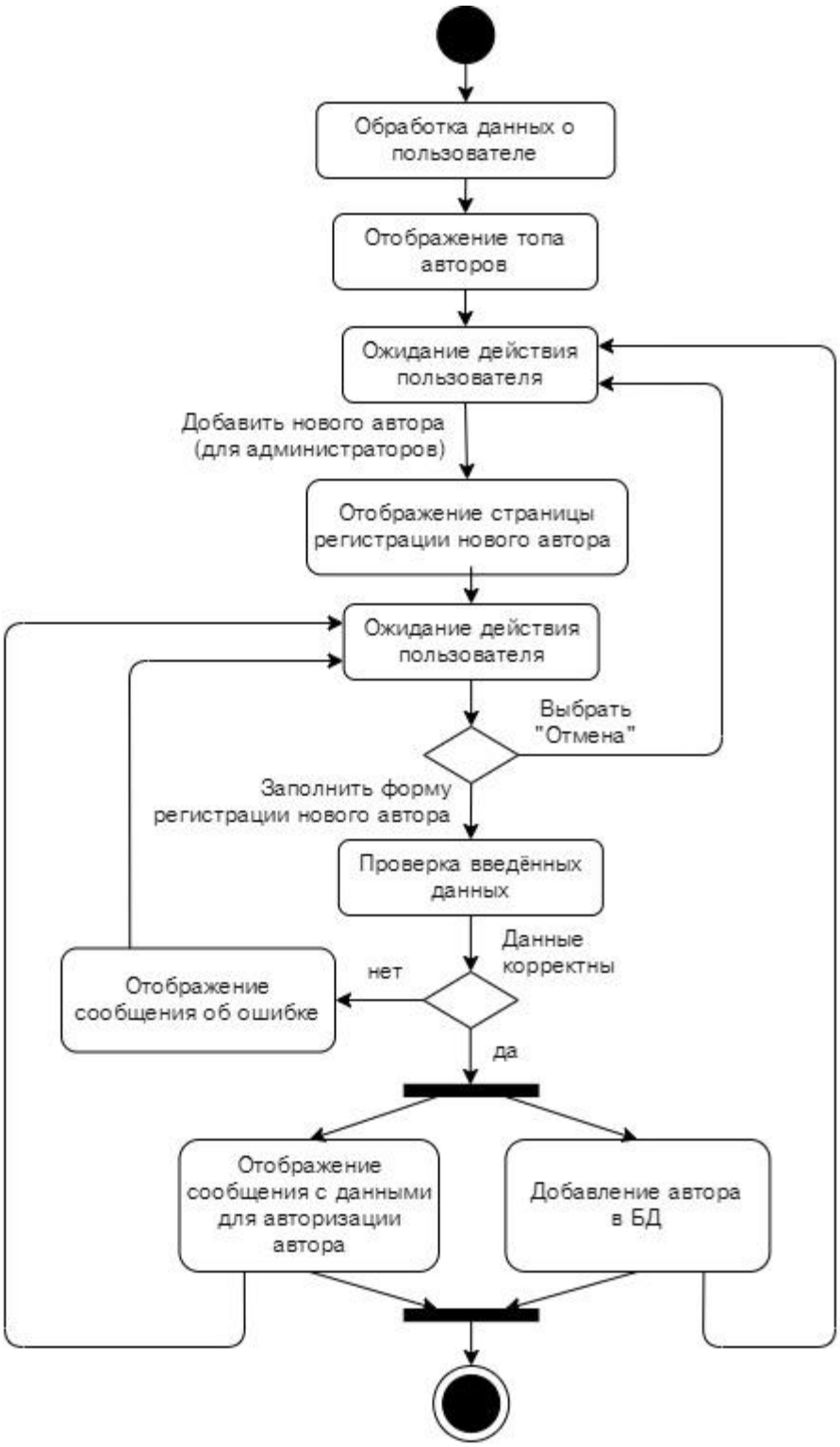

Рис. 24. Диаграмма состояний для процесса вывода рейтинга авторов

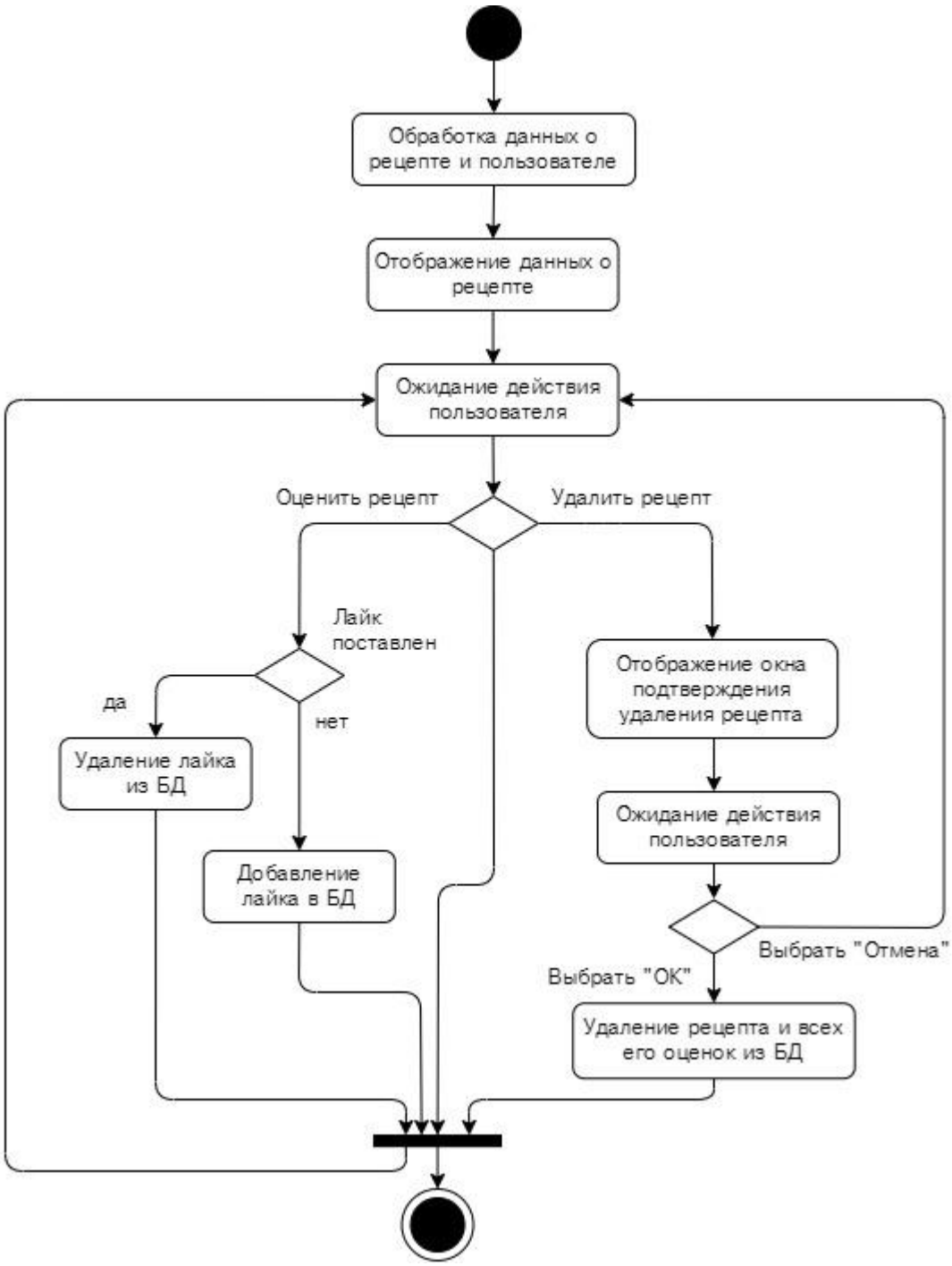

Рис. 25. Диаграмма состояний для процесса вывода информации о рецепте

Диаграмма состояний, изображенная на рисунке 22, отражает возможные состояния системы. При запуске приложения система начинается с отображения формы авторизации и ожидании выбора действия

пользователя. Диаграмма состояний для процесса авторизации изображена на рисунке 23.

В зависимости от выбора пользователя возможны следующие основные цепочки состояний:

- Заполнить формы авторизации
- Перейти по ссылке "Зарегистрироваться"

Если пользователь выбирает опцию "заполнить форму авторизации", то система переходит в состояние проверки введенных данных. Если все данные, введенные пользователем, прошли проверку на соответствие требованиям успешно, то система переходит в состояние отображения главной страницы. Если же данные не прошли проверку, то система переходит в состояние отображения сообщения об ошибке.

Если пользователь выбирает опцию "перейти по ссылке "Зарегистрироваться"", то система переходит в состояние отображения формы регистрации читателя, потом снова в состояние ожидания действий пользователя. Далее пользователь может выбрать опцию "Перейти по ссылке "Авторизоваться"" и система вернется в состояние отображения формы авторизации, или выбрать опцию "Заполнить форму регистрации": тогда система перейдет в состояние проверки введенных данных. Если все данные, введенные пользователем, прошли проверку на соответствие требованиям успешно, то система возвращается в состояние отображения формы авторизации. Если же данные не прошли проверку, то система переходит в состояние отображения сообщения об ошибке и возвращается в состояние ожиданий действия пользователя.

После прохождения процесса авторизации система снова переходит в состояние ожиданий действия пользователя. В зависимости от выбора пользователя возможны следующие основные цепочки состояний:

- Воспользоваться фильтрами

- Посмотреть рецепт
- Посмотреть топ авторов
- Добавить новый рецепт (только для авторов)

Если пользователь выбрал опцию воспользоваться фильтрами, то система переходит в состояние обработки введенных данных. Если были найдены рецепты по введенному запросу, то система переходит в состояние отображения списка рецептов по данному запросу. Если таких рецептов не оказалось, система переходит в состояние отображения сообщения об отсутствии рецептов по данному запросу.

Если пользователь выбрал опцию "добавить новый рецепт", то система переходит в состояние отображения страницы добавления рецепта, затем ожидания действия пользователя. Если пользователь выбирает "отмена", система возвращается в состояние ожидания действия пользователя. Если пользователь заполняет форму добавления рецепта, то система переходит в состояние проверки введенных данных. Если введенные пользователем данные не проходят проверку на соответствие требованиям, то система возвращается в состояние ожидания действия пользователя. Если данные прошли проверку, то система переходит в состояние добавление рецепта в Базу данных.

Если пользователь выбрал опцию "посмотреть рецепт" (рисунок 25), то система переходит в состояние обработки данных о рецепте и пользователе, затем в состояния отображения данных о рецепте и ожидания действия пользователя. Далее пользователь может выбрать опцию "Оценить рецепт" (только для читателей и авторов) и, если оценка пользователем уже была поставлена ранее система перейдет состояние удаление оценки из базы данных, если нет - в состояние добавления оценки в базу данных, или выбрать опцию "Удалить рецепт" (только для авторов рецепта и администраторов): тогда система перейдет в состояния отображения окна подтверждения

удаления рецепта и ожидание действия пользователя. Если пользователь выбрал “Ок” система перейдет в состояние удаления рецепта из базы данных. Если он выбрал отмена, система вернется в состояние ожидания действий пользователя.

Если пользователь выбрал опцию "посмотреть топ авторов" (рисунок 24), то система переходит в состояния обработки данных о пользователе, отображения топа авторов, ожидание действий пользователя. Если пользователь является администратором, то он может добавить нового автора в систему. Тогда она перейдет в состояния отображения страницы регистрации нового автора и ожидание действий пользователя. У пользователя есть две опции: "заполнить форму регистрации нового пользователя" или "выбрать "Отмена"". Если пользователь выбрал опцию "выбрать «Отмена"" система вернется к отображению топа авторов и ожиданию действия пользователей. Если пользователь выбрал "заполнить форму регистрации нового пользователя", система перейдет в состояние проверки введенных данных, затем, если данные некорректны - вернется в состояние ожиданий действий пользователя, если корректны - в состояния добавления автора в базу данных и отображения сообщения с данными для авторизации автора.

После прохождения всех основных цепочек состояний система переходит в состояние ожиданий действия пользователей.

Пользователь в любой момент может завершить работу с системой путем выхода из своего аккаунта.

2.3.7 Диаграммы активности

На рисунках 26 - 29 изображены диаграммы активности для основных процессов в системе. Они отражают возможные действия, состояния которых описаны на диаграммах состояний.

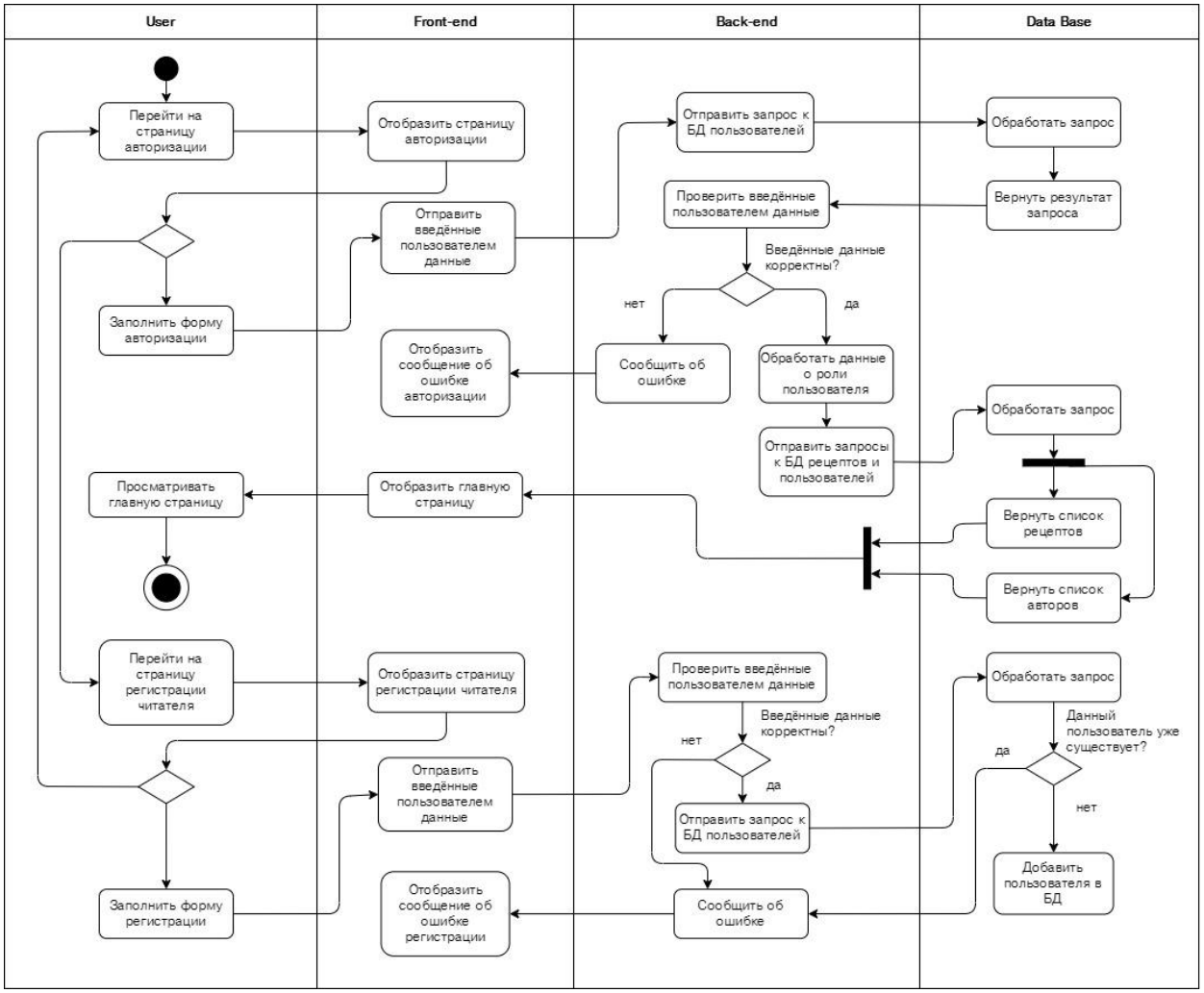

Рис. 26 Диаграмма активности для процессов авторизации/регистрации

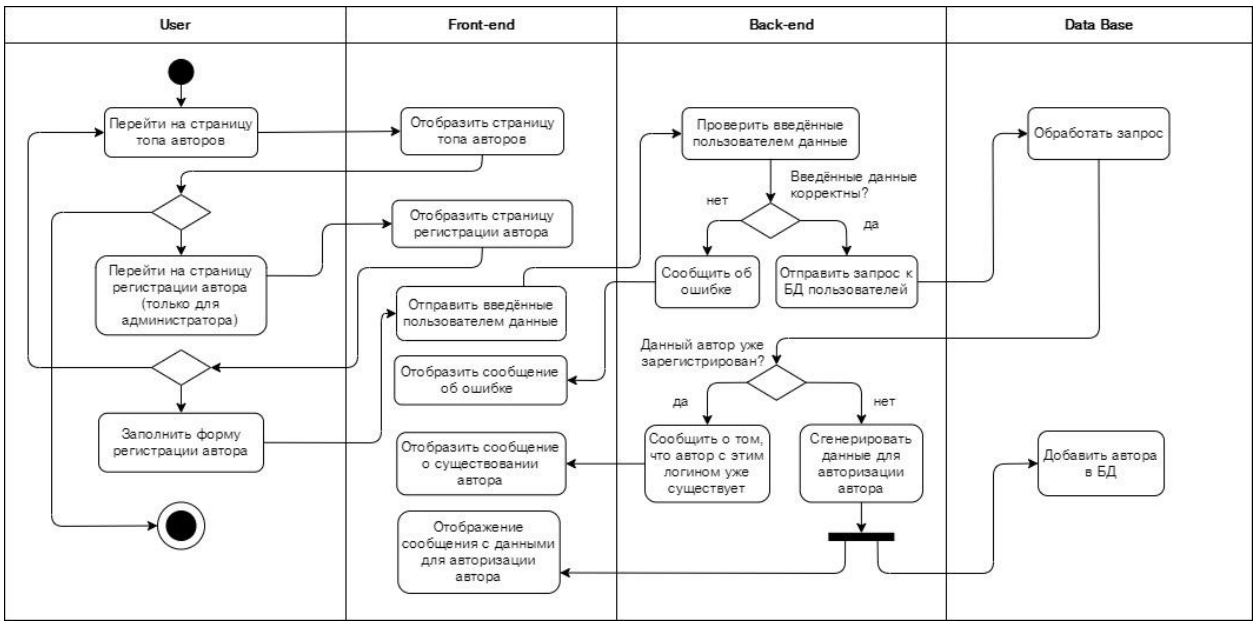

Рис. 27 Диаграмма активности для процессов просмотра топа авторов/добавления нового автора

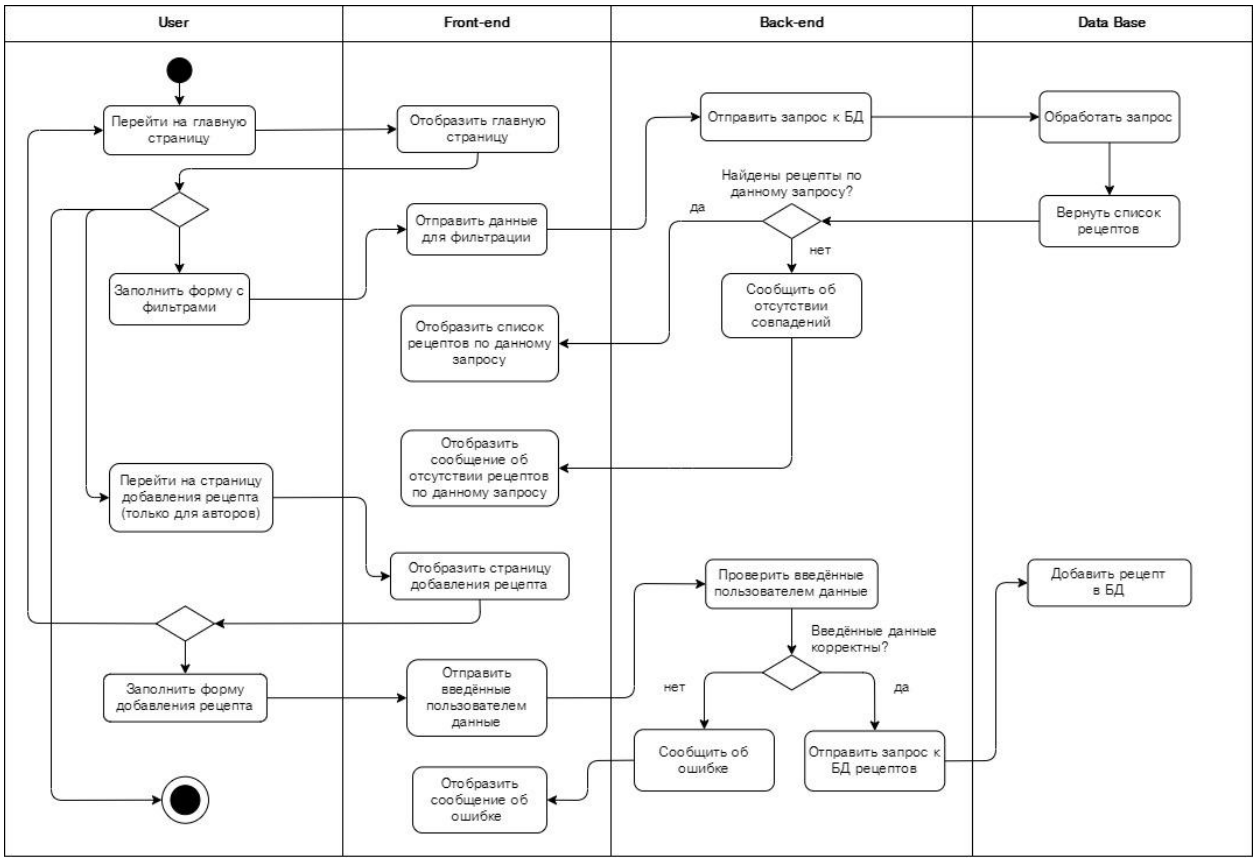

Рис. 28 Диаграмма активности для процессов просмотра главной страницы/добавления нового рецепта

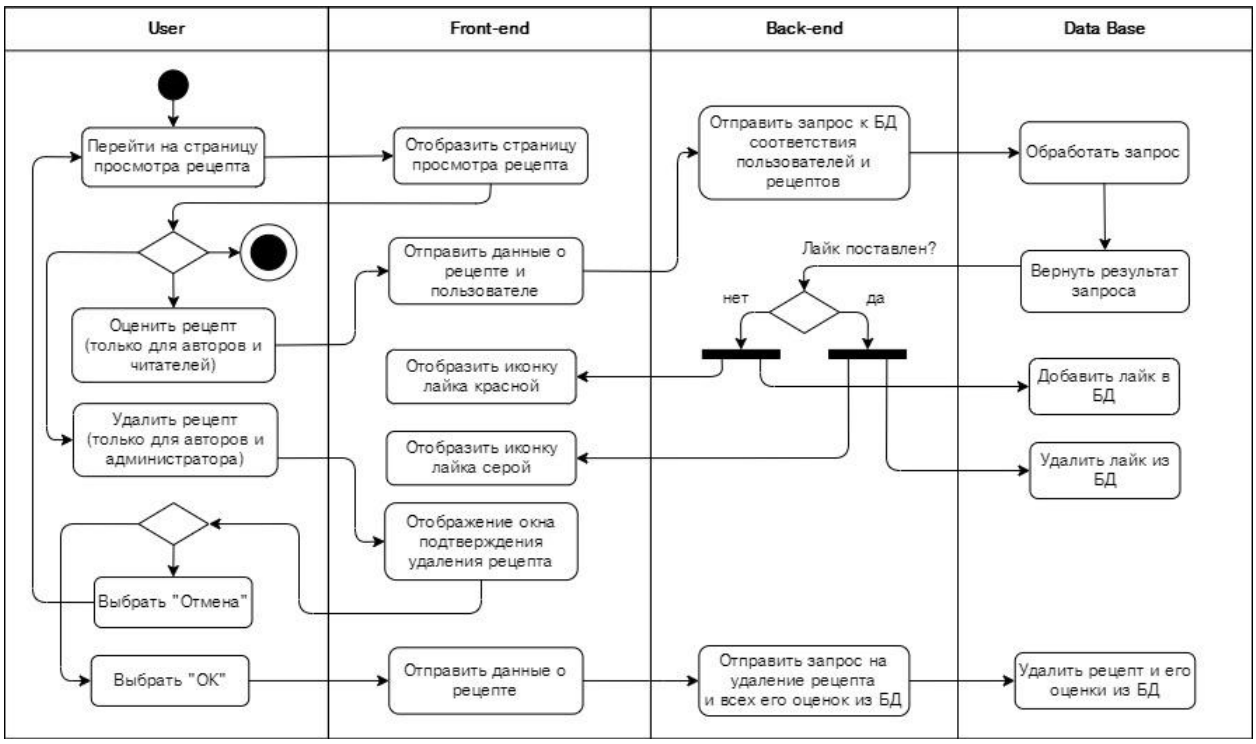

Рис. 29 Диаграмма активности для процесса просмотра рецепта

2.3.8 Диаграмма развертывания

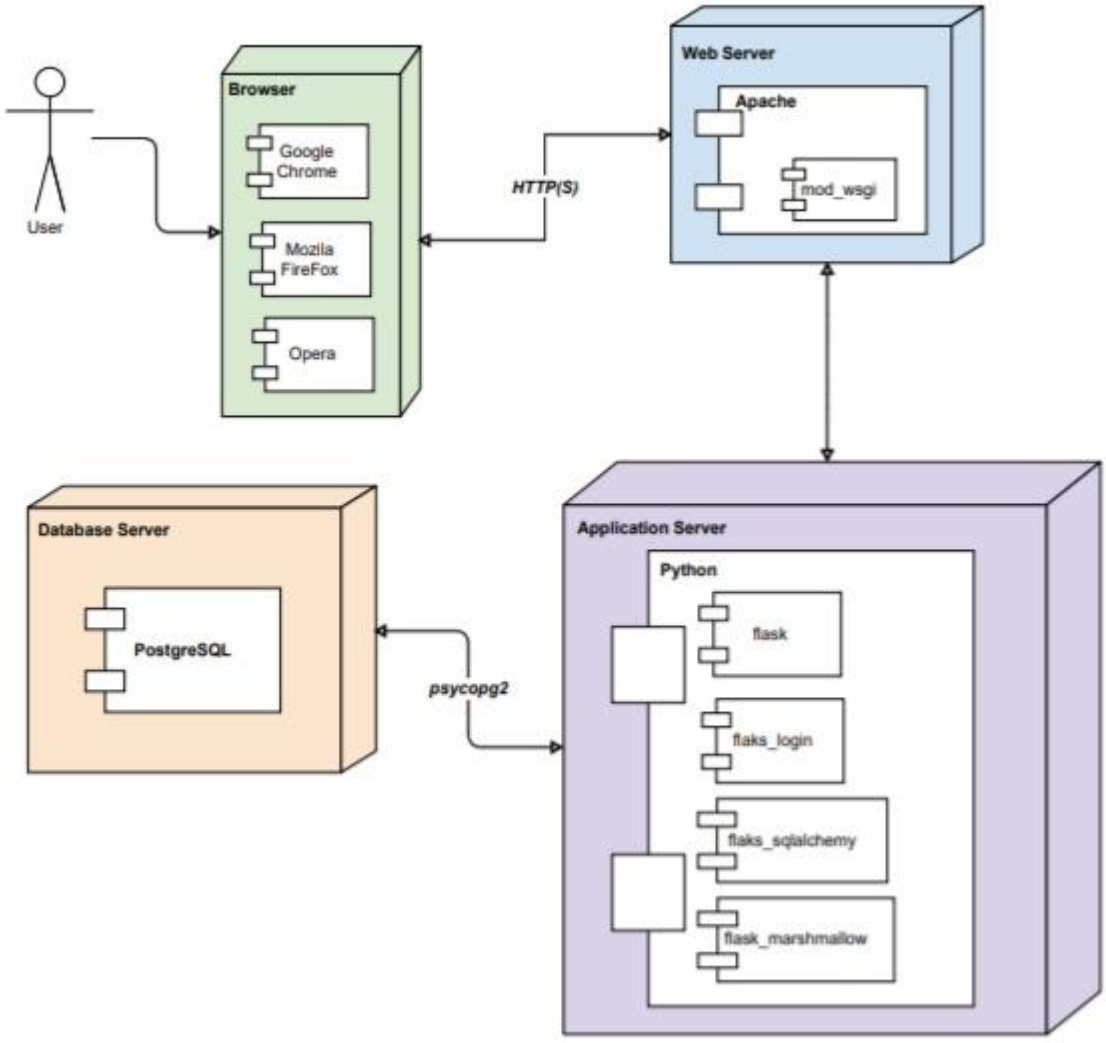

Рис. 30. Диаграмма развертывания.

На рисунке 30 представлена диаграмма развертывания, чтобы определить какие аппаратные компоненты ("узлы") существуют, какие программные компоненты ("артефакты") работают на каждом узле и как различные части этого комплекса соединяются друг с другом.

Для разрабатываемого web-приложения узлом устройства являются браузер, web-сервер, сервера приложения и базы данных. В качестве узла среды выполнения выступают браузеры Google Chrome, Mozilla Firefox и

Opera. front-end приложения развернут на web-сервере Apache, back-end находится на сервере приложения, и база данных на сервере базы данных.

2.3.9 Сценарии воронок конверсии

Для LikeFood главная цель состоит в привлечении как можно большего числа пользователей (авторов и читателей), которые будут добавлять и оценивать диетические рецепты. Поэтому ключевым показателем эффективности будет являться количество добавленных и просмотренных рецептов, количество поставленных оценок на рецептах.

- Воронка конверсии для LikeFood №1

  Посетил главную страницу -> Авторизовался -> Перешел на страницу добавления нового рецепта -> Нажал "Добавить новый рецепт"
- Воронка конверсии для LikeFood №2:

  Посетил главную страницу -> Зарегистрировался как читатель -> Авторизовался -> Перешел на страницу определенного рецепта -> Оценил рецепт
- Воронка конверсии для LikeFood №3:

  Посетил главную страницу -> Зарегистрировался как читатель -> Авторизовался -> Перешел на страницу рейтинга авторов

2.3.10 Архитектура системы

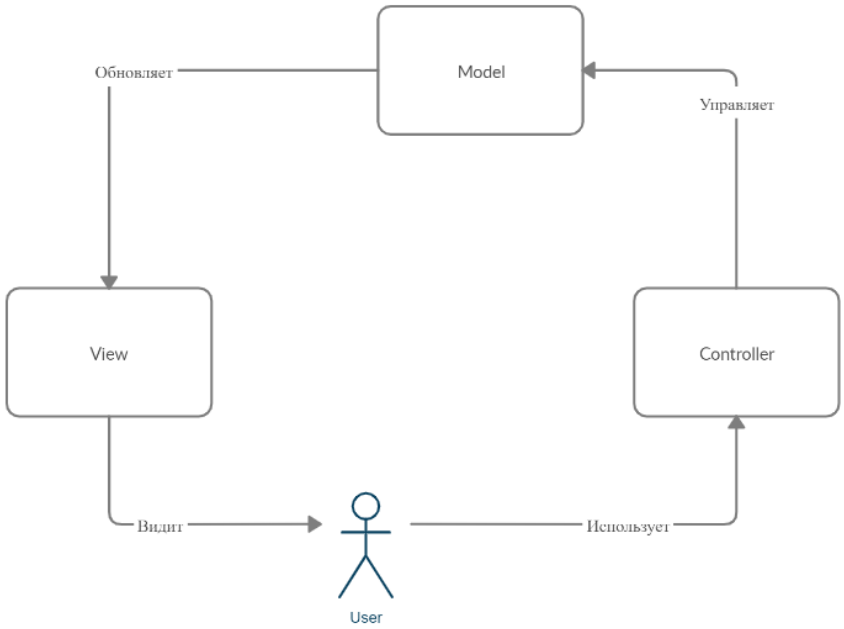

Рис. 31. Архитектурный шаблон MVC

В основе архитектуры нашего приложения лежит MVC-шаблон (Model-View-Controller), который описывает способ построения структуры нашего приложения, сферы ответственности и взаимодействие каждой из частей в данной структуре.

Идея шаблона проста: нужно четко разделять ответственность за различное функционирование в нашем приложение. Таким образом, наш проект разделяется на три основным компонента, каждый из которых отвечает за различные задачи. Идея этого шаблона изображена на рисунке 31.

Контроллер обрабатывает действия пользователя, проверяет полученные данные и передает их модели. То есть это обработчик событий, инициируемых пользователем (нажатие на кнопку, переход по ссылке, отправка формы).

Модель получает данные от контроллера, выполняет необходимые операции и передает их в вид. То есть это метод, который запускается обработчиком и выполняет все основные операции (получение записей из базы данных, проведение вычислений).

Вид (или представление) получает данные от модели и выводит их для пользователя. То есть это интерфейс приложения.

Почему мы используем именно этот шаблон?

Основное сфера использования этого шаблона — построение архитектуры WWW-приложений, написанных на основных языках программирования.

Самое очевидное преимущество, которое мы получаем от использования концепции MVC — это четкое разделение логики представления (интерфейса пользователя) и логики приложения.

Таким образом, на выходе мы получаем независимые блоки кода, которые можно как угодно менять, не затрагивая другие. Это позволяет эффективно работать в команде - каждый занимается своим компонентом. При этом не нужно вникать в чужой код, так как твои действия никак не повлияют на другие фрагменты приложения.

С использованием шаблона MVC код получается более структурированным, и, тем самым, облегчается поддержка, тестирование и повторное использование решений.

Как это реализуется на практике?

В качестве основы реализуемого проекта был выбран веб-фреймворк Flask. Flask не имеет жесткой структуры и позволяет выбирать модули под конкретные задачи и устанавливать их мере необходимости, что хорошо подходит для небольших и средних сайтов (форумы и личные блоги), что и послужило основой для нашего выбора.

Flask — это микрофреймворк состоящий всего из двух компонентов - Werkzeug и Jinja 2.

Werkzeug — это простой маршрутизатор, который обрабатывает запросы и направляет ответы. Также он обеспечивает совместимость с WSGI для использования фреймворка с веб-сервером.

Jinja2 — это шаблонизатор который подставляет сгенерированные на сервере данные в сверстанную HTML страницу.

У Flaska нет ни слоя для работы с базами данных, ни иной функциональности, предоставляемых другими фреймворками.

В рамках архитектуры MVC Werkzeug охватывает Controller (C), a Jinja2 - View (V). Flask не предоставляет интегрированный слой Model (M) и позволяет самостоятельно выбрать решение для базы данных. Мы выбрали

Flask-SQLAlchemy с ORM (Object-Relational Mapper) над реляционной базой данных PostgreSQL. И это и будет нашей Model (M).

Таким образом, Flask нельзя назвать MVC-фреймворком, однако он поддерживает архитектурный шаблон MVC благодаря своей гибкости и расширяемости.

3. Средства разработки

В качестве средств реализации нашего проекта были выбраны следующие технологии:

1. В качестве основного языка программирования для серверной стороны был выбран язык программирования Python. Причины: наличие огромного количестве библиотек, высокая расширяемость, а также его широкое распространение в среде разработчиков.
2. Веб-фреймворк Flask был выбран в качестве основы реализуемого проекта. Flask предоставляет гибкость в работе, не имеет жесткой структуры, позволяет выбирать модули под конкретные задачи и устанавливать их по мере необходимости.
3. Язык Python очень удобно использовать для работы с базой данных. Так как в данной системе используется веб-фреймворк Flask, то для организации работы с базой данных была выбрана популярная в среде разработчиков библиотека SQLAlchemy.
4. В качестве СУБД была выбрана PostgreSQL, обладающую полной объектно-ориентированной функциональностью, справляющейся с обработкой одновременно нескольких заданий.
5. При разработке клиентской стороны были использованы широко распространенные HTML и CSS. Реализация на языке Python позволила легко организовать связь серверной стороны (функционирование базы данных, логика проекта) с клиентской (web-формы).