**СОДЕРЖАНИЕ**

# Анализ предметной области

## Обзор предметной области

## Анализ существующих решений

## Описание функциональности программного средства

## Разработка функциональных требований

# Планирование

## Создание бэклога

## Работа с git

# Проектирование программного средства

## Эргономическое обеспечение

## Информационное обеспечение

# Технико-экономическое обоснование эффективности разработки программного средства

## Характеристика программного средства

## Стоимостная оценка затрат

## Расчет экономического эффекта

## Расчет показателей экономической эффективности разработки и реализации программного средства на рынке

## Вывод об экономической эффективности

## Список использованных источников

# **АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ**

## **Обзор предметной области**

Кулинария представляет собой важную часть современной культуры. Это не только способ приготовления еды, но и отличное хобби, которое может приносить много радости и удовольствия. Кроме того, это хобби, которое имеет множество преимуществ и может привести к улучшению качества жизни.

Целевая аудитория разрабатываемого продукта представляет собой пользователей, объединенных общим интересом к кулинарии, который может быть обусловлен следующими факторами:

– желание открыть для себя новые рецепты блюд;

– желание найти рецепт конкретного блюда;

– желание поделиться своими авторскими рецептами.

Чтобы удовлетворить потребности пользователей, приложение должно включать большой каталог рецептов. Расширение каталога рецептов могут осуществить сами пользователи. Для этого необходимо добавить возможность добавлять собственные рецепты и поддерживать обратную связь с авторами, что позволит создать вокруг него сетевое сообщество. Сетевое сообщество представляет собой группу людей, которые участвуют в социальном взаимодействии и связаны с другими членами интернет-группы [1]. В качестве связи в таком сообществе выступает общий интерес к кулинарии, а социальные взаимодействия происходят посредством обратной связи внутри самого приложения. Сообщество может стать общей средой, где участники могут взаимодействовать с разработчиками, вносить свой вклад в развитие приложения.

Таким образом, пользователи разрабатываемого приложения будут частью сообщества, объединенного общими интересами. Их взаимодействие включает следующие основные процессы:

– поиск рецепта;

– приготовление блюда по рецепту;

– публикация авторских рецептов;

– обратная связь.

Следовательно, для успешного поддержания сообщества приложение должно фокусироваться на этих процессах, обеспечивая удобство использования, прозрачность взаимодействия и возможность постоянного расширения функционала за счёт идей пользователей.

## **Анализ существующих решений**

На сегодняшний день существуют программные средства, предоставляющие пользователю схожие функции: Kitchen Stories, Простые рецепты, Tasty.

«Kitchen Stories» – это мультимедийная кулинарная платформа, ориентирующаяся на видео контент [2]. Платформа предоставляет большой выбор профессиональных рецептов. Пример интерфейс мобильного приложения «Kitchen Stories» представлен на рисунке 1.1.

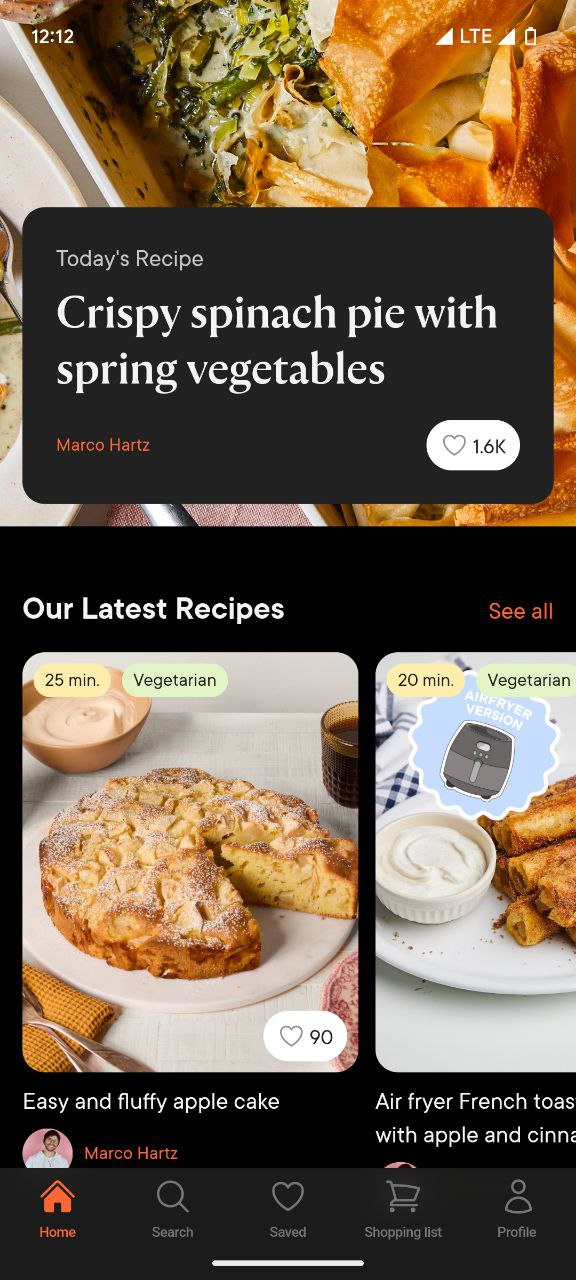


Рисунок 1.1 – Пример интерфейса «Kitchen Stories»

«Kitchen Stories» предлагает пользователю:

– поиск рецептов с помощью фильтров по ингредиентам и категориям;

– каталог профессиональных рецептов;

– система отзывов и комментариев.

Плюсы приложения «Kitchen Stories»:

– большой каталог профессиональных рецептов;

– удобная система фильтров,

– подробная информация о каждом рецепте, разложенная по шагам с использованием изображений.

Минусы приложения «Kitchen Stories»:

– отсутствие встроенного конструктора рецептов;

– отсутствие поддержки русского языка.

«Простые рецепты» - мобильное приложение рецептов, ориентированное на российский рынок. Пример интерфейса программы «Простые рецепты» представлен на рисунке 1.2.

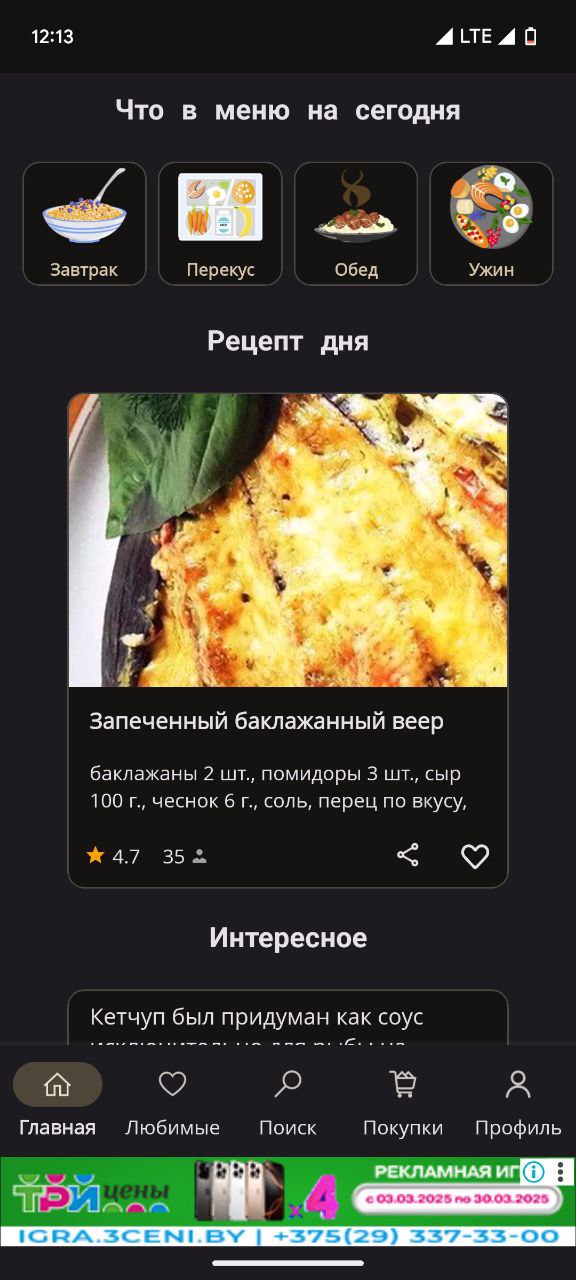


Рисунок 1.2 – Пример интерфейса «Простые рецепты»

Приложение «Простые рецепты» предлагает:

– большой каталог пользовательских рецептов;

– фильтры для поиска рецептов по категориям;

– встроенный конструктор рецептов;

– система отзывов и комментариев.

Плюсы приложения «Простые рецепты»:

– возможность быстро публиковать свои рецепты внутри приложения;

– поддержка русского языка.

Минусы приложения «Простые рецепты»:

– отсутствие гибких фильтров для поиска рецептов (есть только фильтры по категориям);

– редактор рецептов не позволяет подробно описать каждый шаг создания рецепта с прикреплением изображений;

– навязчивая реклама.

Приложение «Tasty» делает ставку на приятный интерфейс и красочные видеорецепты. Пример интерфейса приложения представлен на рисунке 1.3.

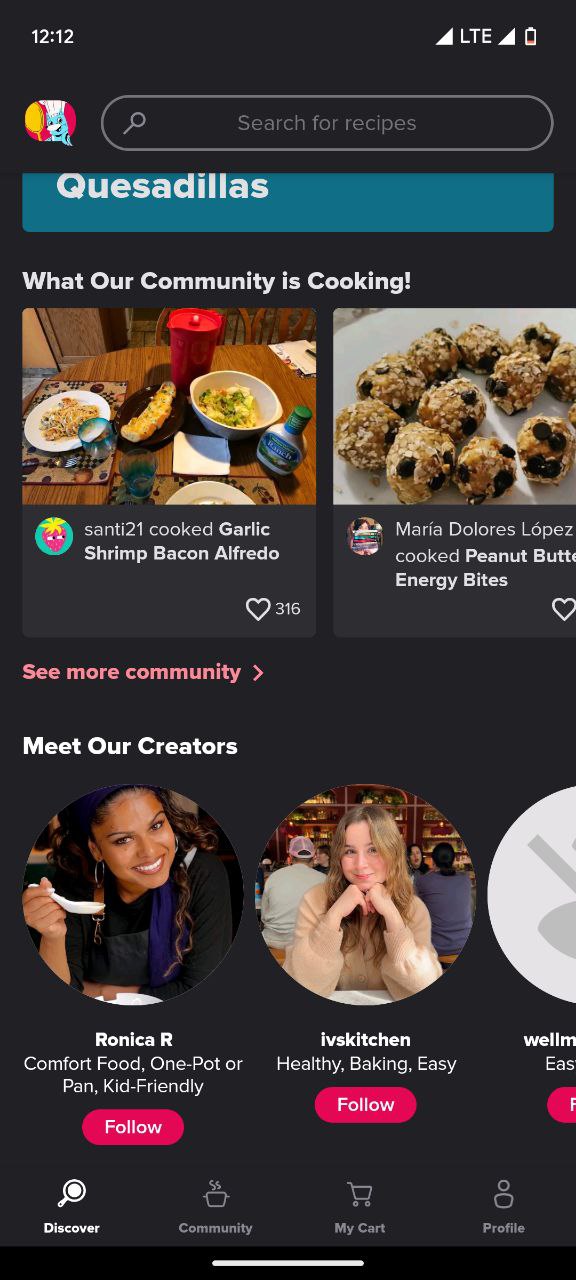


Рисунок 1.3 – Пример интерфейса «Tasty»

«Tasty» предлагает пользователю:

– каталог из более 3000 различных рецептов, включая пользовательские;

– фильтры для поиска рецептов по категориям;

– возможность подписаться на любимых авторов;

– система отзывов и комментариев.

Плюсы программы «Tasty»:

– приятный интерфейс;

– подробное описание процесса приготовления блюда с использованием видеоматериалов.

Минусы программы «Tasty»:

– отсутствие встроенного конструктора;

– отсутствие фильтра по ингредиентам;

– отсутствие поддержки русского языка.

В таблице 1.1 показан сравнительный анализ рассмотренных выше приложений по функциональным и пользовательским критериям. Оценивание проведено по двухбалльной шкале:

– 0 баллов – характеристика отсутствует;

– 1 балл – характеристика присутствует частично или имеет недостатки

– 2 балла – характеристика присутствует полностью.

Таблица 1.1 – Сравнительный анализ кулинарных приложений

|  |  |  |  |
| --- | --- | --- | --- |
| Критерий | Приложение | | |
| Kitchen Stories | Tasty | *Простые рецепты* |
| Конструктор рецептов внутри приложения | 0 | 0 | 1 |
| Гибкие фильтры для поиска рецептов | 2 | 1 | 1 |
| Пошаговое описание процесса приготовления | 2 | 2 | 1 |
| Поддержка русского языка | 0 | 0 | 2 |
| Система отзывов и комментариев | 2 | 2 | 2 |
| ИТОГО | 6 | 5 | 7 |

Таким образом, у каждой из рассмотренных программ есть свои недостатки. К тому же большинство из них не ориентированы на белорусский и российский рынки. Следовательно, необходимо создать программное средство, которое будет учитывать приведенные недостатки.

## **Описание функциональности программного средства**

Для описания функциональности программного средства была выбрана UML диаграмма вариантов использования, которая отражает отношения между актерами и прецедентами и позволяет описать систему на концептуальном уровне. Основные элементы диаграммы – актер и прецедент.

UML предназначен для определения, визуализации, проектирования и документирования программных систем.

На основе технического была составлена диаграмма вариантов использования, изображённая на рисунке 1.4.

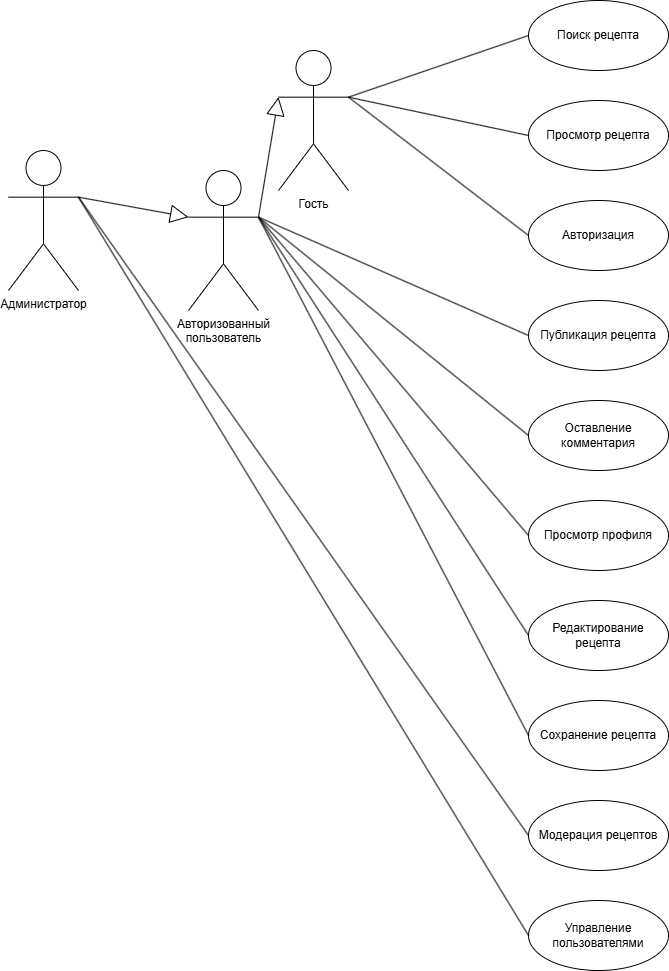


Рисунок 1.4 – Диаграмма вариантов использования

В соответствии с рисунком 1.4 можно выделить два основных составляющих элемента: актер и прецедент.

Актер – роль внешнего объекта программного средства, прямо взаимодействующая с ее частями. Участником может быть человек, другая система, подсистема или класс.

Прецедент – описание множества последовательных событий, выполняемых программным средством, а также отображение взаимодействия между участниками и системой.

На основании представленной диаграммы вариантов использования можно выделить следующих основных актеров:

– гость;

– авторизованный пользователь;

– администратор.

Гостю доступен базовый функционал программы. Он может просматривать каталог рецептов, пользоваться поиском или авторизироваться. После авторизации он становится авторизованным пользователем.

Авторизированным пользователям открываются новые возможности. Они могут сохранять понравившиеся рецепты, оставлять комментарии. Им также доступен конструктор рецептов, с помощью которого они могут делиться своими кулинарными идеями. В процессе авторизации каждый пользователь также указывает информацию (имя пользователя, изображение), которая будет отображаться в его профиле и доступна для просмотра другим пользователям.

Администраторам доступны дополнительные возможности. Они могут управлять аккаунтами других пользователей (блокировать пользователей), а также заниматься модерацией рецептов.

## **Разработка функциональных требований**

Функциональные требования можно разделить на несколько типов по ролям в приложении. Рассмотрим требования для каждой роли по отдельности.

Для гостей:

– просмотр каталога рецептов;

– просмотр страницы рецепта;

– просмотр комментариев и отзывов других пользователей;

– просмотр профилей других пользователей;

– поиск рецептов по названию, ингредиентам;

– фильтрация по категориям, времени приготовления, популярности;

– возможность регистрации или авторизации с помощью сторонних сервисов (Google).

Для авторизованных пользователей:

– добавление и удаление рецептов из избранного;

– просмотр списка сохраненных рецептов;

– возможность оценить рецепт;

– возможность оставить комментарий;

– использование конструктора для создания и публикации собственных рецептов;

– просмотр и редактирование своего профиля.

Для администраторов:

– редактирование, одобрение, отклонение рецептов;

– управление комментариями (удаление при нарушении правил);

– блокировка и разблокировка аккаунтов других пользователей.

# **ПЛАНИРОВАНИЕ**

## **Создание бэклога**

Бэклог продукта — это перечень задач, расположенных в порядке важности. Этот инструмент является основополагающим для эффективного планирования и приоритизации работы в рамках как крупных команд, так и индивидуальных проектов. Наиболее важные элементы расположены в начале бэклога, чтобы разработчики понимали, чем следует заняться в первую очередь. Бэклог может быть бесконечным: постоянно возникают новые идеи, как улучшить функциональность [[?](https://books.yandex.ru/reader/mbmDjL93?resource=book)].

Бэклог обеспечивает единый, централизованный источник информации обо всей запланированной работе, устраняя необходимость поиска задач в разрозненных заметках или списках. Бэклог помогает структурировать общий объем работ, разбивая его на более мелкие и управляемые элементы, а также способствует определению ключевых приоритетов еще до того, как будет потрачено значительное время на детализацию отдельных задач. Кроме того, бэклог облегчает отслеживание прогресса разработки, позволяя разработчику видеть, какие задачи уже выполнены и что еще предстоит сделать, что способствует своевременному выявлению и устранению потенциальных задержек или проблем.

При работе над проектом в одиночку, четкое определение границ проекта приобретает первостепенное значение. Отсутствие ясного понимания того, что входит и что не входит в объем проекта, может легко привести к "разрастанию" функциональности, когда разработчик постоянно добавляет новые возможности, не предусмотренные первоначальным планом. Это, в свою очередь, чревато увеличением сроков разработки, потерей фокуса на основных целях и, в конечном итоге, может привести к выгоранию и незавершенности проекта. Установка контрольных точек и реалистичных сроков для ключевых этапов проекта позволит отслеживать прогресс и избежать затягивания разработки [[?](https://dev.to/danieldevi/from-idea-to-launch-a-guide-to-building-software-projects-for-solo-devs-and-teams-17a5)].

В качестве инструментов для ведения бэклога индивидуальный разработчик может использовать самые простые методы и средства. Текстовые файлы представляют собой очень простое решение для ведения списка задач. Они легко обновляются и просматриваются, не требуют установки дополнительного программного обеспечения и могут быть доступны на любом устройстве. Электронные таблицы (например, Excel или Google Sheets) предоставляют более широкие возможности для структурирования задач, позволяя добавлять столбцы с информацией о статусе, приоритете, оценке трудозатрат и других параметрах. Также существуют специальные инструменты предоставляющие шаблоны для создания бэклога, такие как Trello, Asana или Notion. Для индивидуального разработчика часто бывает достаточно именно таких простых и интуитивно понятных инструментов, которые не требуют специального обучения или сложной настройки. Основная цель - эффективно управлять задачами, а не тратить время на освоение сложных систем управления проектами.

В рамках данного проекта для создания бэклога использовался Notion. В начальном бэклоге были выделены следующие задачи:

1 UI design. Подготовить визуальную и интерактивную часть всех страниц приложения перед разработкой.

2 Catalogue page. Создать главную страницу, содержащую каталог всех рецептов.

3 Recipe page. Создать страницу, предоставляющую подробную и пошаговую информацию о рецепте.

4 Search page. Создать страницу поиска с возможностью выставить фильтры.

5 Profile page. Создать страницу профиля пользователя, отображающую информацию о пользователе и опубликованные им рецепты.

6 Login page. Создать страницу авторизации с использованием сервисов авторизации, таких как google.

7 Favorite page. Создать страницу, отображающую сохраненные рецепты пользователя.

8 Comment and review system. Добавить интерфейс для возможности публикации комментария или отзыва к рецепту.

9 Authentication endpoints. Реализовать функционал аутентификации.

10 Authorization. Реализовать функционал авторизации пользователей с различными ролями.

11 Recipe endpoints. Реализовать CRUD для приложения на стороне сервера.

12 Review and comment endpoints. Реализовать CRUD для публикации отзывов и комментариев.

13 Moderation page. Создать страницу, позволяющую администраторам производить модерацию рецептов.

Созданный бэклог представлен на рисунке 2.1.

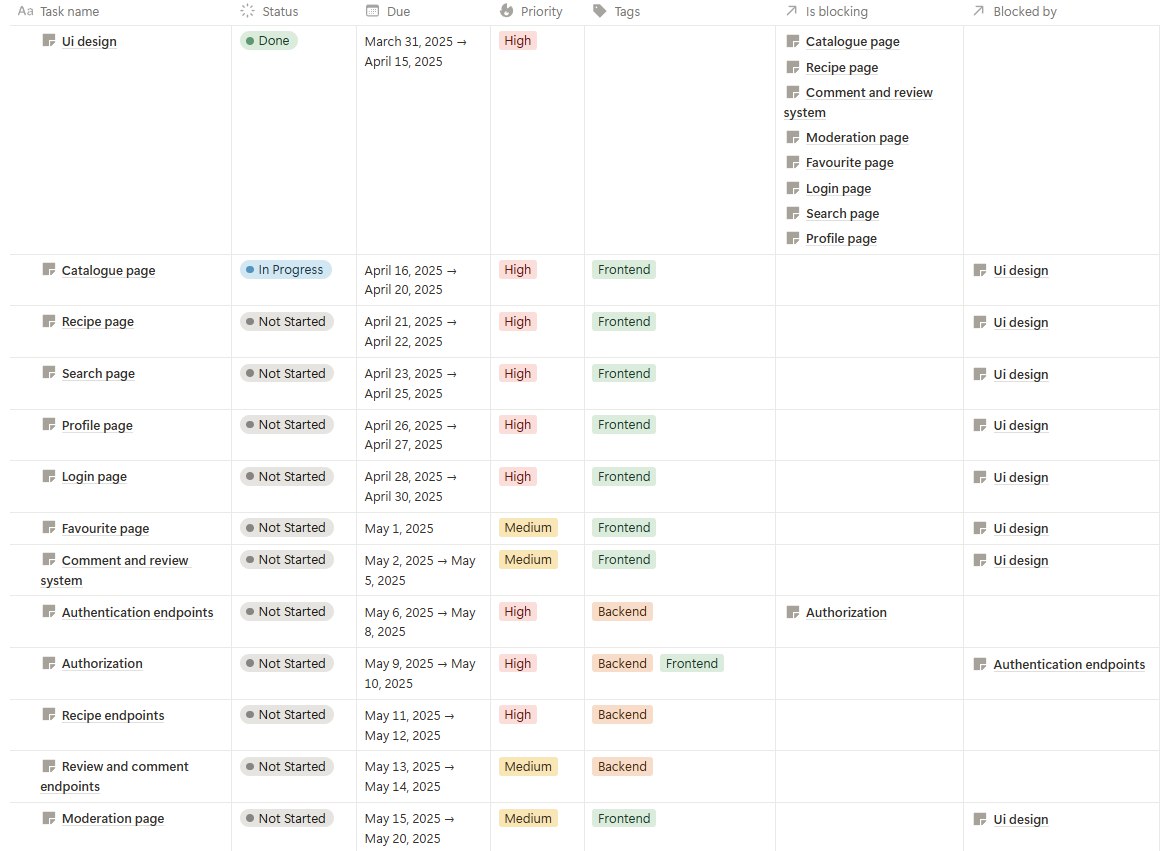


Рисунок 2.1 - Бэклог

Таким образом, бэклог является ключевым инструментом для организации и эффективного управления проектом. Хорошо управляемый бэклог помогает индивидуальному разработчику оставаться сфокусированным на основных целях проекта, эффективно управлять объемом работы, отслеживать свой прогресс и оптимально использовать свое время и ограниченные ресурсы. Принципы гибкой разработки, такие как итеративность и готовность к изменениям, могут быть успешно адаптированы для индивидуальной работы, а бэклог является центральным элементом этого подхода, обеспечивая необходимую гибкость и прозрачность на протяжении всего жизненного цикла проекта. Таким образом, бэклог выступает в качестве своеобразной дорожной карты, которая ведет индивидуального разработчика к успешному завершению его проекта.

## **Работа с git**

Контроль версий является фундаментальной практикой в разработке программного обеспечения, и его важность не уменьшается, когда разработчик работает в одиночку. Управление изменениями в коде, отслеживание истории проекта и возможность возврата к предыдущим состояниям имеют решающее значение для успеха любого программного проекта, независимо от его размера или количества участников. Использование системы контроля версий, такой как Git, предоставляет индивидуальным разработчикам ряд преимуществ, включая полную историю проекта с момента его создания, возможность вернуться к любой предыдущей версии или увидеть различия между ними, а также ценный опыт работы с широко используемым в отрасли инструментом [[?](https://sdlambert.github.io/2015/04/09/git-workflow-for-solo-development/)].

Используемый рабочий процесс предполагает создание новой ветки для каждой новой функции или исправления ошибки, изолированную работу в этой ветке и последующее слияние обратно в основную ветку по завершении. Это позволяет удерживать баланс между простотой и эффективным управлением кодом. Изолируя каждую функцию или исправление ошибки в отдельной ветке, индивидуальный разработчик может работать более методично, предотвращать случайное внесение ошибок в основную кодовую базу и легко отменять изменения при необходимости.

Каждый коммит должен представлять собой одно логическое изменение. Хорошо написанные сообщения коммитов имеют решающее значение для понимания истории проекта и облегчения будущей отладки или модификаций. Каждый коммит должен отражать одну законченную задачу — добавление новой функциональности, исправление ошибки или рефакторинг существующего кода. Для стандартизации сообщений рекомендуется использовать подход Conventional Commits, при котором заголовок сообщения состоит из трёх частей: типа изменений (например, feat, fix, docs, style, refactor, perf, test или chore), области применения (auth, ui, api, db и т. д.) в круглых скобках и краткого описания в повелительном наклонении без точки на конце, длиной до 50 символов.

Использование такого подхода облегчит навигацию по истории репозитория, упростит поиск нужных изменений и повысит прозрачность разработки.

# **ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА**

## **Эргономическое обеспечение**

При первом запуске приложения пользователь видит экран приветствия, прототип которого представлен на рисунке 2.1. Дизайн выполнен в красно-белой цветовой гамме, привлекающей внимание. Крупный заголовок сообщает пользователю о назначении приложения, а большая кнопка «Let's Go» приглашает перейти к следующему этапу.

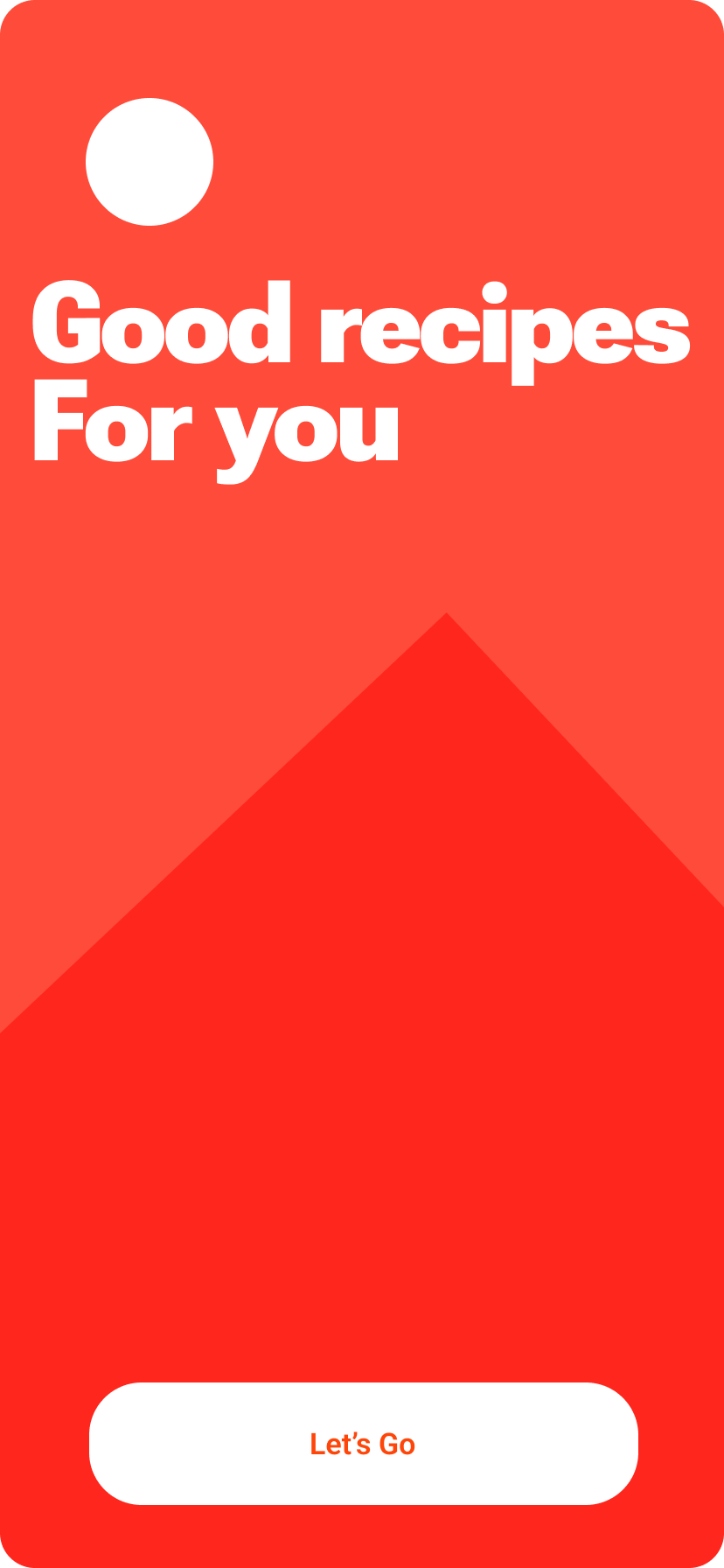


Рисунок 2.1 – Прототип экрана приветствия

После экрана приветствия пользователя встречает главный экран (рисунок 2.2). На главном экране пользователю предлагается выбрать, что он хочет попробовать сегодня. Пользователь может выбрать интересующую его категорию для просмотра блюд. Для быстрого поиска присутствует строка поиска.

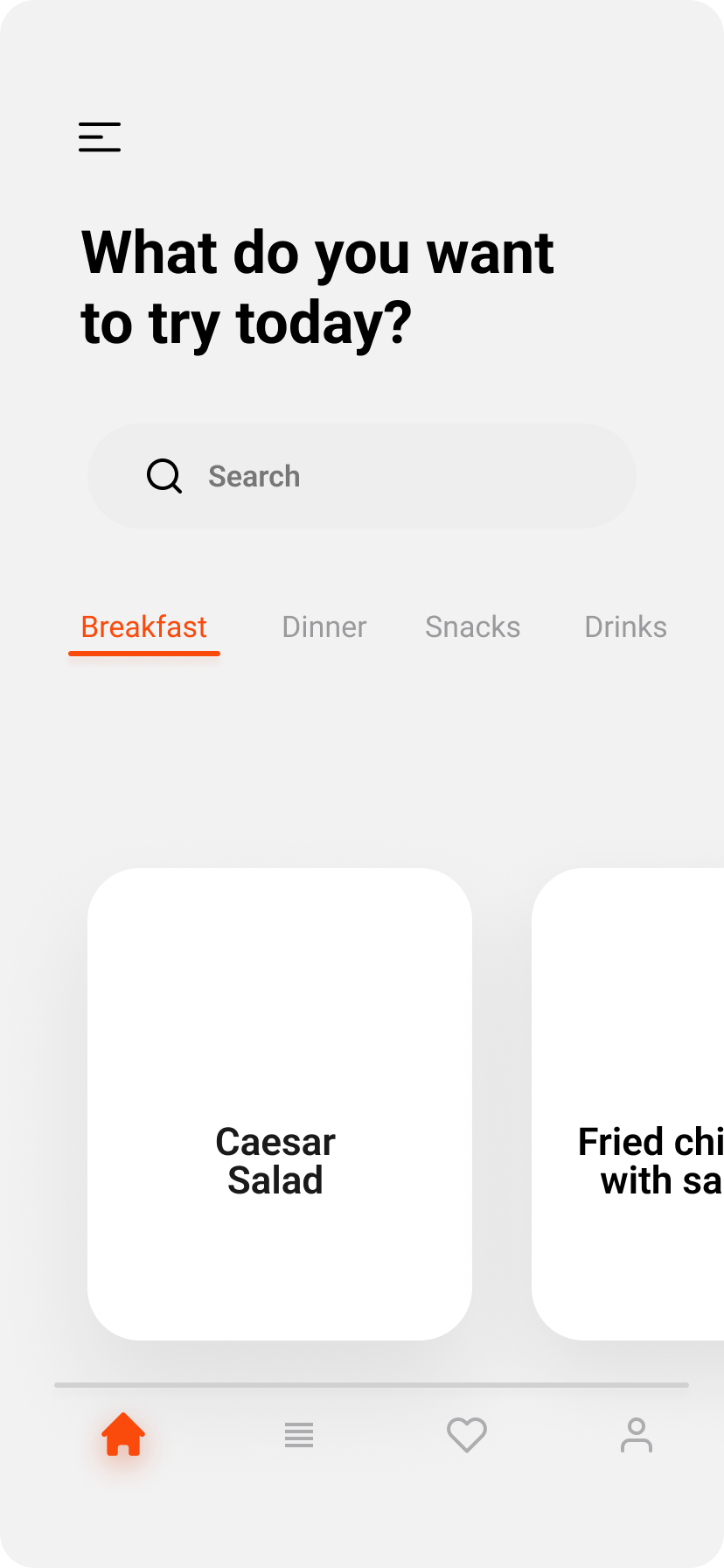


Рисунок 2.2 – Прототип главного экрана

Экран профиля содержит информацию о пользователе и доступ к основным функциям, связанным с его учетной записью. В верхней части располагаются имя и электронная почта, рядом с которыми находится кнопка для редактирования этих данных. Ниже представлены разделы: «History», где отображается список ранее просмотренных рецептов; «Reviews», позволяющий просматривать собственные отзывы и комментарии; «FAQ», включающий ответы на часто задаваемые вопросы; и «Help», предоставляющий справочную информацию по использованию приложения. Прототип экрана профиля представлен на рисунке 2.3.

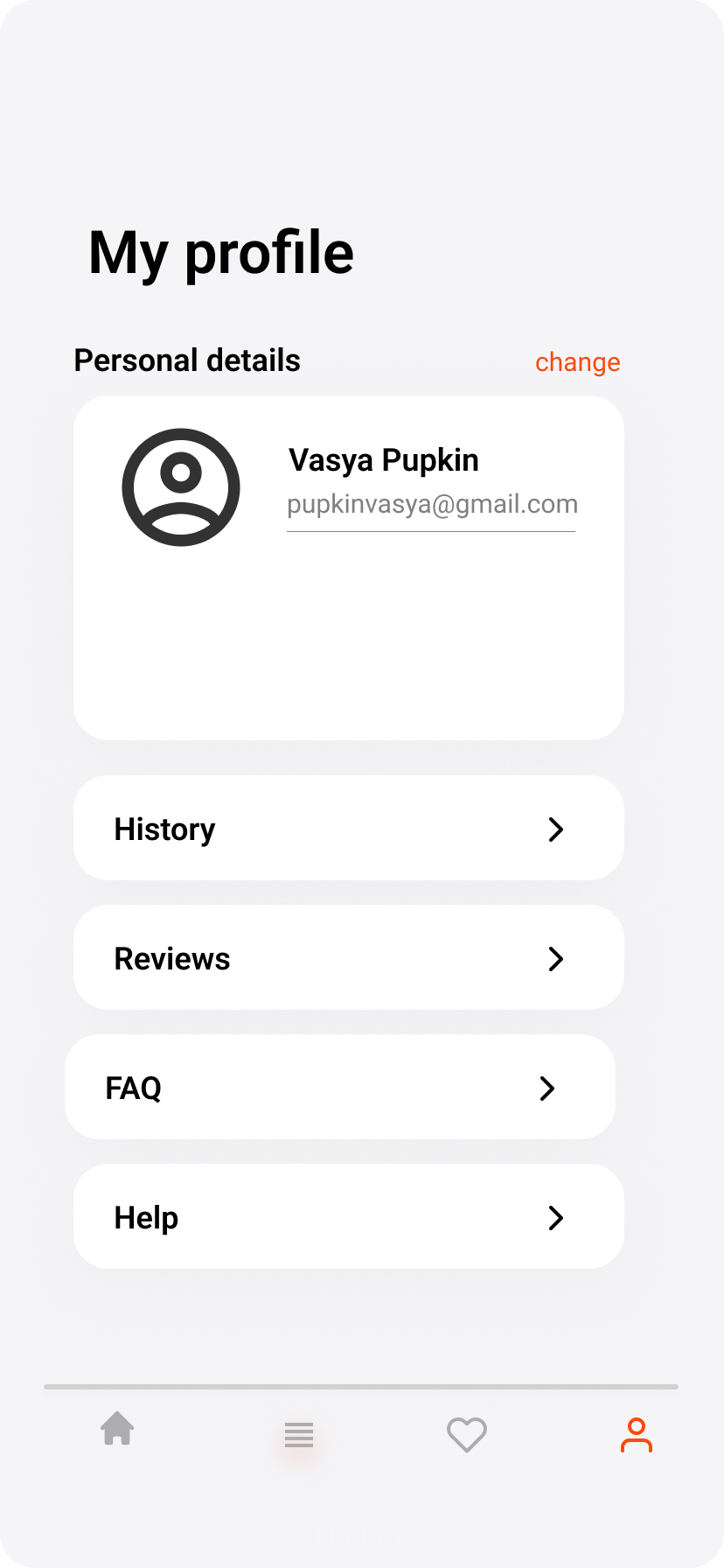


Рисунок 2.3 – Прототип экрана профиля

## **Информационное обеспечение**

Схема логической модели базы данных, включающая атрибуты сущностей предметной области программного средства и связи между ними представлена на рисунке 2.4.

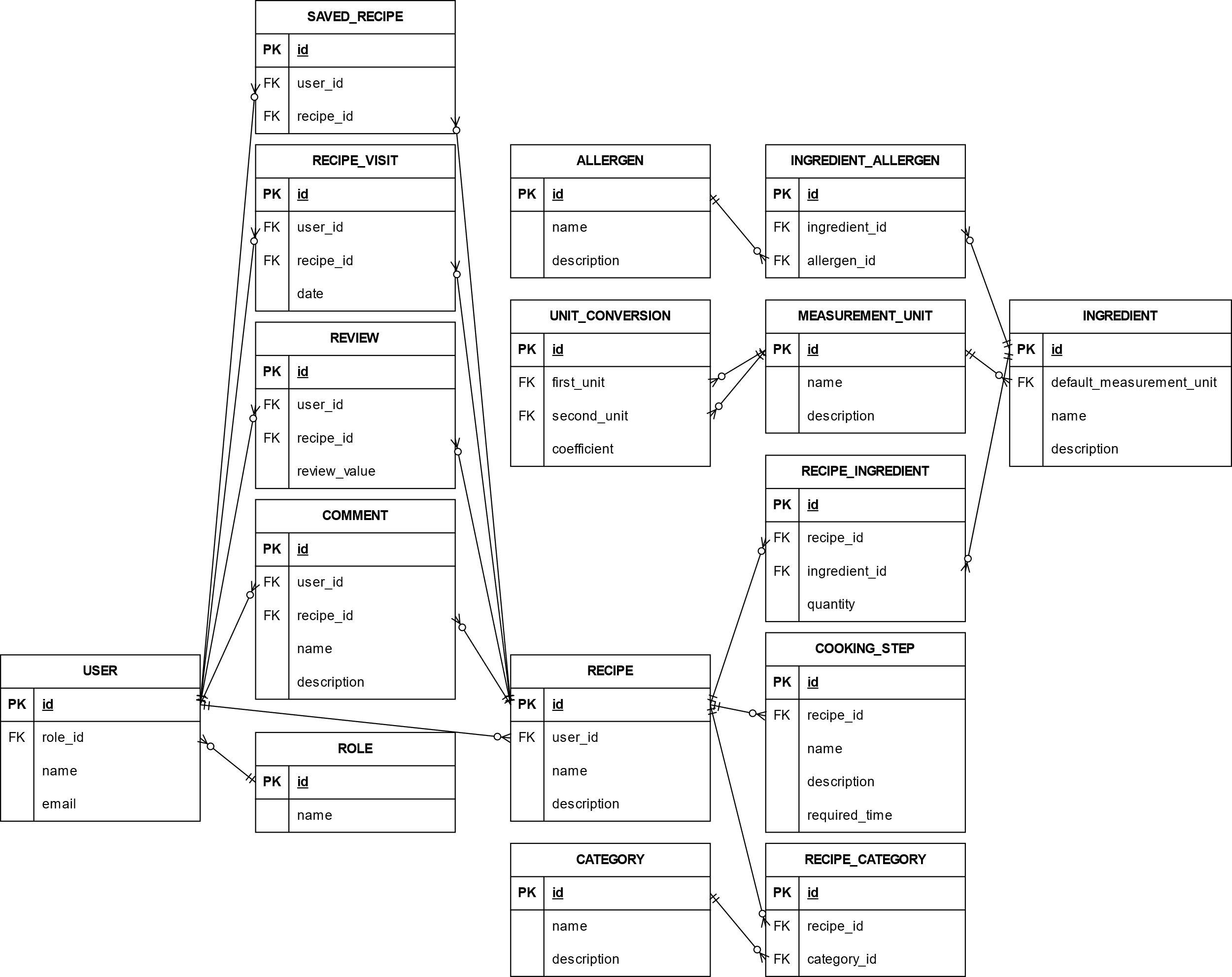


Рисунок 2.4 – Схема логической модели базы дынных

После анализа функциональных требований к веб-приложению определен следующий набор сущностей:

– user – данные о пользователе;

– role – данные о роли пользователя;

– recipe – данные о рецепте, связана с сущностью «category» связью «многие ко многим» с помощью таблицы «recipe\_category»;

– category – данные о категории рецептов, связана с сущностью «recipe» связью «многие ко многим» с помощью таблицы «recipe\_category»;

– recipe\_ingredient – данные об ингредиенте в конкретном рецепте;

– ingredient – данные об ингредиенте, связана с сущностью «allergen» связью «многие ко многим» с помощью таблицы «ingredient\_allergen».;

– measurement\_unit – данные о единице измерения для ингредиента в рецепте;

– unit\_conversion – данные для перевода из одной единицы измерения в другую (коэффициент для перевода единиц измерения);

– cooking\_step – данные о шаге приготовления блюда;

– allergen – данные об ингредиентах, которые могут вызывать аллергию, связана с сущностью «ingredient» связью «многие ко многим» с помощью таблицы «ingredient\_allergen»;

– saved\_recipe – данные о сохраненном пользователем рецепте;

– recipe\_visit – данные о просмотре рецепта пользователем;

– review – отзыв пользователя о рецепте;

– comment – комментарий пользователя к рецепту.

Сущность «user» включает следующие атрибуты:

– id – уникальный идентификатор пользователя;

– role\_id – роль пользователя, обеспечивает связь «многие к одному» с сущностью «role»;

– name – имя пользователя;

– email – адрес электронной почты пользователя.

Сущность «role» включает в себя следующие атрибуты:

– id – уникальный идентификатор роли;

– name – название роли.

Сущность «recipe» включает в себя следующие атрибуты:

– id – уникальный идентификатор рецепта;

– user\_id – автор рецепта, обеспечивает связь «многие к одному» с сущностью «user»;

– name – название рецепта;

– description – описание рецепта.

Сущность «category» включает в себя следующие атрибуты:

– id – уникальный идентификатор категории;

– name – название категории;

– description – описание категории.

Сущность «recipe\_ingredient» включает в себя следующие атрибуты:

– id – уникальный идентификатор ингредиента рецепта;

– recipe\_id – рецепт, обеспечивает связь «многие к одному» с сущностью «recipe»;

– ingredient\_id – ингредиент, обеспечивает связь «многие к одному» с сущностью «ingredient»;

– quantity – количество ингредиента в единицах измерения по умолчанию, необходимое для приготовления блюда по рецепту.

Сущность «ingredient» включает в себя следующие атрибуты:

– id – уникальный идентификатор ингредиента;

– default\_measurement\_unit – единица измерения по умолчанию для данного ингредиента, обеспечивает связь «многие к одному» с сущностью «measurement\_unit»;

– name – название ингредиента;

– description – описание ингредиента.

Сущность «measurement\_unit» включает в себя следующие атрибуты:

– id – уникальный идентификатор единицы измерения;

– name – название единицы измерения;

– description – описание единицы измерения.

Сущность «unit\_conversion» включает в себя следующие атрибуты:

– id – уникальный идентификатор коэффициента для перевода из первой единицы измерения во вторую;

– first\_unit – первая единица измерения, обеспечивает связь «многие к одному» с сущностью «measurement\_unit»;

– second\_unit – вторая единица измерения, обеспечивает связь «многие к одному» с сущностью «measurement\_unit»;

– coefficient – коэффициент перевода.

Сущность «cooking\_step» включает в себя следующие атрибуты:

– id – уникальный идентификатор шага приготовления;

– recipe\_id – рецепт, к которому относится данный шаг, обеспечивает связь «многие к одному» с сущностью «recipe»;

– name – название шага приготовления;

– description – описание шага приготовления;

– required\_time – время выполнение шага приготовления.

Сущность «allergen» включает в себя следующие атрибуты:

– id – уникальный идентификатор аллергена;

– name – название аллергена;

– description – описание аллергена.

Сущность «saved\_recipe» включает в себя следующие атрибуты:

– id – уникальный идентификатор сохраненного рецепта;

– user\_id – пользователь, сохранивший рецепт, обеспечивает связь «многие к одному» с сущностью «user»;

– recipe\_id – рецепт, который был сохранен, обеспечивает связь «многие к одному» с сущностью «recipe»;

Сущность «recipe\_visit» включает в себя следующие атрибуты:

– id – уникальный идентификатор просмотра рецепта;

– user\_id – пользователь, просмотревший рецепт, обеспечивает связь «многие к одному» с сущностью «user»;

– recipe\_id – рецепт, который был просмотрен, обеспечивает связь «многие к одному» с сущностью «recipe»;

– date – дана просмотра.

Сущность «review» включает в себя следующие атрибуты:

– id – уникальный идентификатор отзыва;

– user\_id – пользователь, оставивший отзыв, обеспечивает связь «многие к одному» с сущностью «user»;

– recipe\_id – рецепт, о котором был оставлен отзыв, обеспечивает связь «многие к одному» с сущностью «recipe»;

– review\_value – оценка рецепта.

Сущность «comment» включает в себя следующие атрибуты:

– id – уникальный идентификатор комментария;

– user\_id – пользователь, оставивший комментарий, обеспечивает связь «многие к одному» с сущностью «user»;

– recipe\_id – рецепт, о котором был оставлен комментарий, обеспечивает связь «многие к одному» с сущностью «recipe»;

– name – заголовок комментария;

– description – тело комментария.

Схема физической модели базы данных приведена на рисунке 2.5.

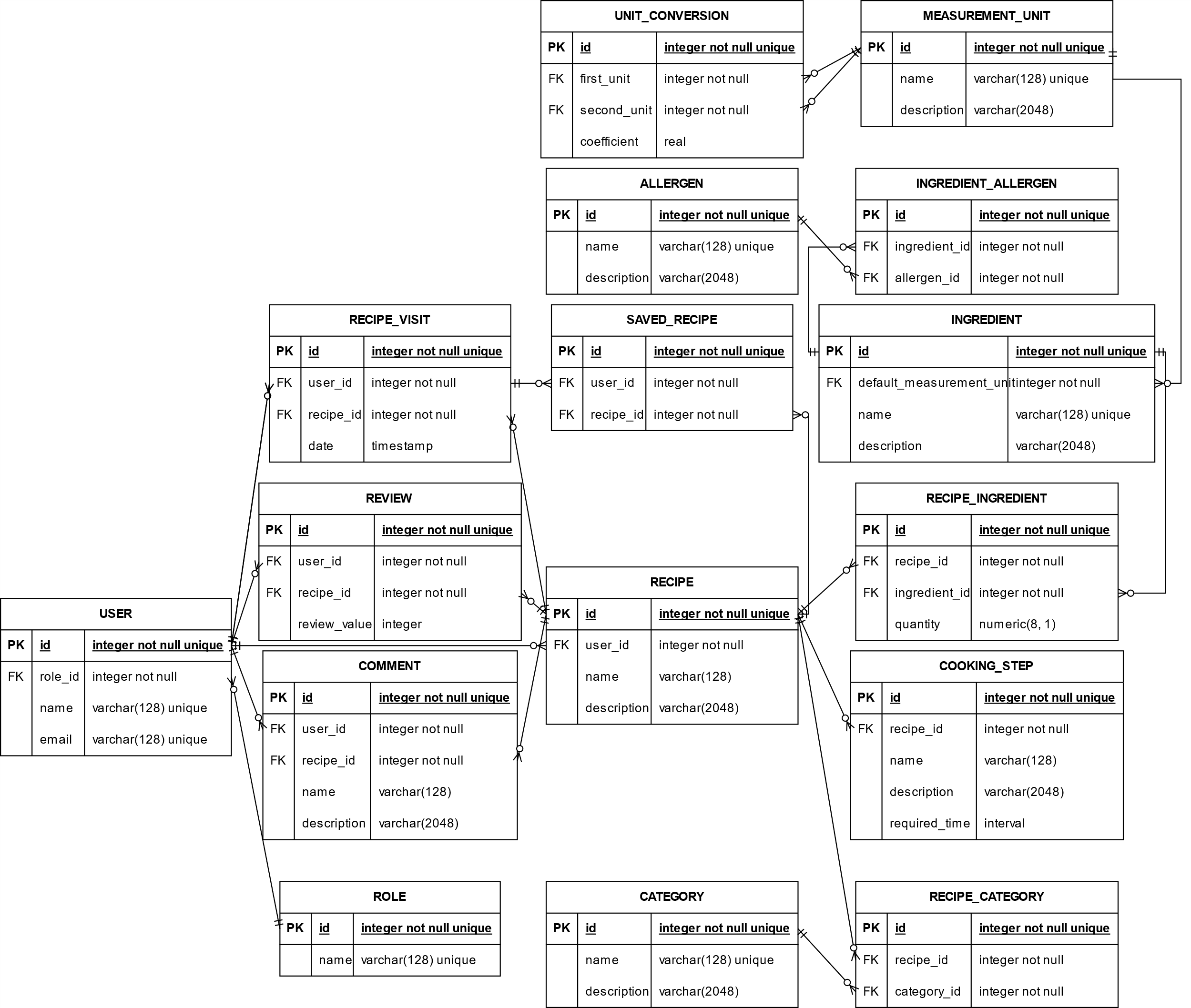


Рисунок 2.5 – Схема физической модели базы данных

Схема и ее описание, приведенные выше, показывают таблицы реляционной базы данных, их поля и типы данных, первичные и внешние ключи, а также связи между таблицами. Разработанная база данных соответствует третьей нормальной форме.

# **ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ ЭФФЕКТИВНОСТИ РАЗРАБОТКИ ПРОГРАММНОГО СРЕДСТВА**

## **Характеристика программного средства**

Цель разработки заключается в создании мобильного приложения, которое позволит пользователям быстро и удобно находить рецепты интересующих их блюд, а также делиться своими кулинарными идеями с другими пользователями.

Приложение призвано решить проблему пользователей, которые пользуются используют поисковые системы (Google, Yandex) для поиска рецептов. При таком подходе пользователи не могут воспользоваться фильтрами или единым удобным интерфейсом. Также мобильное приложение позволит пользователям создать свое сообщество, в котором они могут обмениваться своим опытом. Например, отзывы других пользователей могут помочь начинающим кулинарам в выборе подходящего рецепта.

Приложение ориентировано на следующие группы пользователей:

1. Домохозяйки и семьи. Люди, которые готовят дома и ищут новые рецепты для разнообразия семейного меню.
2. Начинающие кулинары. Пользователи, желающие научиться готовить, которым нужны пошаговые инструкции и советы.
3. Профессиональные повара и кулинары. Эксперты, которые хотят делиться своими авторскими рецептами и набираться вдохновения у коллег.
4. Любители правильного питания. Те, кто следит за своим рационом и ищет полезные и сбалансированные рецепты.
5. Люди с диетическими ограничениями. Те, кто страдает от аллергий или непереносимости определенных продуктов (например, безглютеновые, безлактозные рецепты).
6. Молодежь и студенты. Те, кто ищет быстрые и бюджетные рецепты.

В качестве модели монетизации будет использоваться freemium. Она даёт бесплатный доступ к основным функциям приложения для широкой аудитории, но при этом мотивирует активных пользователей оплачивать дополнительные полезные функции (отключение рекламы, эксклюзивные рецепты). Это даёт баланс между охватом и доходом, не отпугивая потенциальных пользователей.

В качестве канала продаж программного продукта будет использоваться платформа Google Play.

## **Стоимостная оценка затрат**

Инвестиции в разработку программного модуля включают в себя основную заработную плату разработчиков, дополнительную заработную плату разработчиков, отчисления на социальные нужды и прочие расходы.

Расчет основной заработной платы сотрудников осуществляется по формуле (1.1):

|  |  |  |
| --- | --- | --- |
|  |  | (3.1) |

где – коэффициент премий и иных стимулирующих выплат; – количество исполнителей, занятых разработкой веб-ресурса, чел.; – часовая заработная плата -го исполнителя, р.; – трудоемкость работ, выполняемых -м исполнителем, ч.

Разработкой всего приложения занимается инженер-программист (Full-stack разработчик), обязанности тестирования приложения лежат на инженере-тестировщике. Задачами инженера-программиста, который занимается являются создание модели данных, графического интерфейса, связи между моделью данных и графическим интерфейсом. Инженер-тестировщик занимается выявлением неработоспособных частей приложения, а также оценивает пользовательский опыт, получаемый от использования приложения.

Месячная заработная плата основана на медианных показателях для Junior инженера-программиста в период с декабря по февраль 2025 года по Республике Беларусь, которая составляет 1100 Долларов США в месяц, а для Junior инженера-тестировщика – 900 Долларов США [3]. По состоянию на 5 марта 2025 года 1 Доллар США по курсу Национального Банка Республики Беларусь составляет 3,2345 Белорусских рублей [4].

В перерасчёте на Белорусские рубли месячные оклады для инженера-программиста и инженера-тестировщика соответственно составляют составляет 3 557,95 и 2 911,05 Белорусских рублей соответственно.

Часовой оклад исполнителей высчитывается путём деления их месячного оклада на количество рабочих часов в месяце, которое принято равным 168 ч.

За трудоемкость работ инженера-программиста и инженера-тестировщика принято соответственно 196 и 32 часа.

Коэффициент премии приравнивается к единице, так как она входит сумму заработной платы. Затраты на основную заработную плату приведены в таблице:

Таблица 3.1 – Затраты на основную заработную плату

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Категория исполнителя | Месячный оклад, р | Часовой оклад, р | Трудоёмкость работ, ч | Итого, р |
| Инженер-программист | 3 557,95 | 21,18 | 196 | 4151,28 |
| Инженер-тестировщик | 2 911,05 | 17.33 | 32 | 554,56 |
| Итого | | | | 4705,84 |
| Премия и иные стимулирующие выплаты (0%) | | | | 0 |
| Всего затраты на основную заработную плату разработчиков | | | | 4705,84 |

Расчет дополнительных выплат, предусмотренных законодательством о труде, определяется по формуле (1.2)

|  |  |  |
| --- | --- | --- |
|  |  | (3.2) |

где – норматив дополнительной заработной платы.

Значение норматива дополнительной заработной платы принято за 10 %.

Отчисления на социальные нужды (в Фонд Социальной Защиты Населения и на обязательное страхование определяются в соответствии с действующими законодательными актами по формуле (1.3)

|  |  |  |
| --- | --- | --- |
|  |  | (3.3) |

где – норматив отчислений в ФСЗН и Белгосстрах.

Норматив отчислений в ФСЗН и Белгосстрах примем равным 34,6%, т.е. = 34,6%.

Прочие расходы, включаемые в себестоимость разработки программного обеспечения, рассчитываются как процент от затрат на основную заработную плату команды разработчиков по формуле (1.4).

|  |  |  |
| --- | --- | --- |
|  |  | (3.4) |

где – норматив прочих расходов.

Значение норматива прочих расходов принято за 30%.

Расчет расходов на реализацию определяется с помощью норматива расходов на реализацию по формуле (1.5).

|  |  |  |
| --- | --- | --- |
|  |  | (3.5) |

где Нр – норматив расходов на реализацию.

Значение норматива расходов на реализацию принято за 3%.

Определяем общую сумму затрат на разработку и реализацию как сумму ранее вычисленных расходов: на основную заработную плату разработчиков, дополнительную заработную плату разработчиков, отчислений на социальные нужды, расходы на реализацию и прочие расходы. Значение определяется по формуле:

|  |  |  |
| --- | --- | --- |
|  |  | (3.6) |

Таким образом, величина затрат на разработку программного средства высчитывается по указанной выше формуле и указана в таблице:

Таблица 3.2 – Затраты на разработку

|  |  |  |
| --- | --- | --- |
| Название статьи затрат | Формула/таблица для расчёта | Значение, р. |
| 1. Основная заработная плата разработчиков | См. таблицу 1.1 | 4705,84 |
| 2. Дополнительная заработная плата разработчиков |  | 470,58 |
| 3. Отчисление на социальные нужды |  | 1791,04 |
| 4. Прочие расходы |  | 1411,74 |
| 5. Расходы на реализацию |  | 141,17 |
| 6. Общая сумма затрат на разработку и реализацию |  | 8520,37 |

## **Расчет экономического эффекта**

Для расчёта экономического эффекта организации-разработчика программного средства, а именно чистой прибыли, необходимо знать такие параметры как объем продаж, цену реализации и затраты на разработку.

Соответственно необходимо создать обоснование возможного объёма продаж, количества проданных лицензий расширенной версии программного средства, купленного пользователями.

Согласно данным портала президента РБ, 1 января 2024 года население Беларуси составило 9,1 миллиона человек, из которых 91,5% пользуются интернетом [5]. Это значит, что около 8,3 миллиона человек являются потенциальными пользователями мобильных приложений.

На данный момент отсутствует актуальная статистика по использованию кулинарных приложений в Беларуси. В связи с этим целесообразно опираться на данные российского рынка, поскольку он схож с белорусским по структуре и потребительским предпочтениям.

Согласно исследованию, в 2023 году 36% россиян увлекались кулинарией [6]. Соответственно, количество потенциальных пользователей кулинарного приложения в Беларуси можно рассчитать следующим образом:

Предположим, что удастся занять 4% потенциального рынка в первый год. Тогда ожидаемое количество пользователей приложения:

Для расчета количества платных подписок используется среднее значение конверсии, принятое за 5%. Тогда ожидаемое количество купленных платных подписок составит:

.

Для оценки стоимости подписки было проведено сравнение с аналогичными приложениями. Например, цена на подписку в приложении Kitchen Stories cоставляет 6,0 евро, что, в пересчете на белорусские рубли по состоянию на 5 марта 2025 года по курсу национального банка составит около 20,4 белорусского рубля. Цена подписки в приложении «Простые рецепты» составляет 2,4 доллара или 7,8 белорусского рубля. Однако в эту подписку входит только отключение рекламы. На основе этих данных было принято решение установить стоимость подписки на уровне 10 белорусских рублей.

Для расчёта прироста чистой прибыли необходимо учесть налог на добавленную стоимость, который высчитывается по следующей формуле:

|  |  |  |
| --- | --- | --- |
|  |  | (3.7) |

где – отпускная цена копии программного средства, р.; – ставка налога на добавочную стоимость, %.

Ставку налога на добавленную стоимость примем равной 20%. Используя данное значение, посчитаем НДС:

|  |  |  |
| --- | --- | --- |
|  |  |  |

Посчитав налог на добавленную стоимость, можно рассчитать прирост чистой прибыли, которую получит разработчик от продажи программного продукта. Для этого используется формула:

|  |  |  |
| --- | --- | --- |
|  |  | (3.8) |

где Нп – ставка налога на прибыль, %; – рентабельность продаж копий.

Ставка налога на прибыль примем равной 20%. Рентабельность продаж копий принята за 40%. Зная ставку налога и рентабельность продаж копий (лицензий), рассчитывается прирост чистой прибыли для разработчика:

|  |  |  |
| --- | --- | --- |
|  | . |  |

## **Расчёт показателей экономической эффективности разработки и реализации программного средства на рынке**

Для того, чтобы оценить экономическую эффективность разработки и реализации программного средства на рынке, необходимо рассмотреть результат сравнения затрат на разработку данного программного продукта, а также полученный прирост чистой прибыли за год.

Сумма затрат на разработку меньше суммы годового экономического эффекта, поэтому можно сделать вывод, что такие инвестиции окупятся менее, чем за один год.

Таким образом, оценка экономической эффективности инвестиций производится при помощи расчёта рентабельности инвестиций (Return on Investment, ROI). Формула для расчёта ROI:

|  |  |  |
| --- | --- | --- |
|  |  | (3.9) |

Тогда рентабельность инвестиций составит:

|  |  |  |
| --- | --- | --- |
|  |  |  |

## **Вывод об экономической эффективности**

Проведённые расчёты технико-экономического обоснования позволяют сделать предварительный вывод о целесообразности разработки данного программного продукта. Общая сумма затрат на разработку и реализацию составила 8520,37 белорусских рублей, а отпускная цена была установлена на уровне 10 белорусских рублей.

Прирост чистой прибыли за год, исходя из предполагаемом объеме продаж 6000 лицензий, составляет 16000,00 Белорусских рублей. Рентабельность инвестиций за год составляет 87,79%.

Это означает, что разработка данного программного продукта является целесообразной и реализация программного средства по установленной цене имеет смысл.

Однако, следует учитывать возможные риски, связанные с конкуренцией со стороны аналогов, что может привести к незамеченности продукта на рынке. Кроме того, высокая рентабельность сопровождается определёнными рисками, а расчётные показатели были получены при предполагаемом объёме продаж 6000 лицензий в год.

Тем не менее, при поддержке проект может получить долгосрочное и успешное развитие и масштабироваться на другие регионы.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

[1] Патяева, Е. Ю. Интернет-сообщества как новая социальная реальность: психологические задачи «человека сетевого» / Е. Ю. Патяева // Социальная и экономическая психология. – 2018. – Т. 3. – №2(9) – С. 74–109.

[2] Kitchen Stories – About [Электронный ресурс]. – Режим доступа : https://pages.kitchenstories.com/en/about.

[3] Зарплата в ИТ | dev.by [Электронный ресурс]. – Режим доступа : https://salaries.devby.io.

[4] Официальные курсы белорусского рубля по отношению к иностранным валютам, устанавливаемые Национальным банком Республики Беларусь ежедневно, на 05.03.2025 [Электронный ресурс]. – Режим доступа : https://www.nbrb.by/statistics/rates/ratesdaily.

[5] Беларусь в цифрах | Официальный интернет-портал Президента Республики Беларусь [Электронный ресурс]. – Режим доступа : https://president.gov.by/ru/belarus/numbers.

[6] Исследование ОК: каждый второй россиянин старается найти единомышленников в соцсетях [Электронный ресурс]. – Режим доступа : https://insideok.ru/blog/issledovanie-ok-kazhdyj-vtoroj-rossijanin-staraetsja-najti-edinomyshlennikov-v-socsetjah.