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

Факультет *компьютерных наук*

Кафедра *технологий обработки и защиты информации, информационных технологий управления*

Web-приложение учета подарков “Golden Fish”

Курсовой проект

09.03.02 Информационные системы и технологии

Допущен к защите

Обучающийся _____*С.А. Никонова, 3 курс, д/о*

Обучающийся _____*Е.М. Скворцова, 3 курс, д/о*

Обучающийся _____*А.А. Шурыгина, 3 курс, д/о*

Руководитель _____*В.С. Тарасов, преподаватель*

Воронеж 2020

Содержание

Введение

В современном мире человеку предъявляются высокие требования: он должен быть успешен во всех сферах, успевать всё и везде. За последнее столетие темп жизни серьезно возрос, и тенденции к его уменьшению не предвидится. В суете мы часто забываем о действительно важных вещах: порой люди не помнят о предстоящих праздниках или у них попросту не хватает времени к ним подготовится. Бывают ситуации, когда человек за день до события заходит в магазин и покупает первый попавшийся под руку подарок. Конечно, это не способствует гармоничным взаимоотношениям и часто заставляет человека испытывать чувство вины.

Данный курсовой проект посвящен созданию приложения, которое поможет человеку организовать процесс выбора подарков, позволит составлять список желаний, просматривать списки желаний друзей и решать, кто подарит тот или иной подарок.

При наличии такого приложения у дарителя отпадет необходимость мучительного выбора, а получатель всегда будет доволен полезным и нужным подарком, о котором он давно мечтал. Система поможет лучше узнать человека, а в некоторых моментах даже развлечься. Стоит отметить, что наличие данного приложения не означает полного отказа от спонтанных и неожиданных подарков, а его использование носит исключительно вспомогательный характер.

1. Постановка задачи

Целью данной курсовой работы является реализация веб-приложения, позволяющего зарегистрированным пользователям создавать свои списки желаний (подарков, которые они хотели бы получить), а также просматривать списки желаний других пользователей - друзей, отмечать те желания, которые они могли бы осуществить, планируя таким образом список подарков для своих друзей.

Приложение должно выполнять перечисленные функции и решать основную задачу — управление списками желаний. Для этого к приложению выдвинуты следующие требования:

- удобный, интуитивно понятный пользовательский интерфейс;
- минималистичный дизайн;
- стабильная работа в браузере GoogleChrome v.80.0.3987.

Архитектура разрабатываемого приложения должна отвечать требованиям шаблона проектирования MVC, иметь front-end и back-end части. Выбор данной архитектуры обусловлен стремлением отделить бизнес-логику и интерфейс, чтобы реализовать независимые друг от друга компоненты, позволяющие легко поддерживать проект в дальнейшем. В соответствии с данной архитектурой должны быть реализованы следующие компоненты приложения:

- база данных;
- пользовательский интерфейс, клиентская часть приложения;
- бизнес-логика проекта, обрабатывающая запросы к базе данных на удаленном сервере;
- взаимодействие между клиентской и серверной частью (RESP API).

2. Анализ предметной области

2.1. Глоссарий

Желание - стремление к осуществлению чего-либо или обладанию чем-либо.

Подарок - осуществление желания другого человека.

2.2 Анализ существующих решений

2.2.1. LesterWish

LesterWish - это портал организации подарков. Первый в списке запросов через поисковую систему Google и Yandex. Позволяет создать событие и составить список желаний для него, отправить ссылку друзьям через социальные сети, SMS или по электронной почте, и позволить им забронировать подарки, чтобы не было повторений.

Достоинства:

- создание событий;

- много полей для описания события;

- возможность добавления желания прямо из онлайн-каталога известных магазинов;

- возможность поделиться списком любым способом.

Недостатки:

-нельзя добавить свое желание (только выбрать товар из имеющихся каталогов);

Рисунок 1 - Главная страница сайта LesterWish

2.2.2. MyWishList

MyWishList - самый крупный российский сервис для составления списков желаний. Полезен пользователям, желающим подготовить список ожидаемых подарков, а также создать список для личного использования, для планирования собственных приобретений.

Достоинства:

- создание поздравлений;

- возможность оставлять комментарии;

- создание скрытых желаний (не видны никому, видны только друзьям, видны всем).

Недостатки:

- непрезентабельный интерфейс.

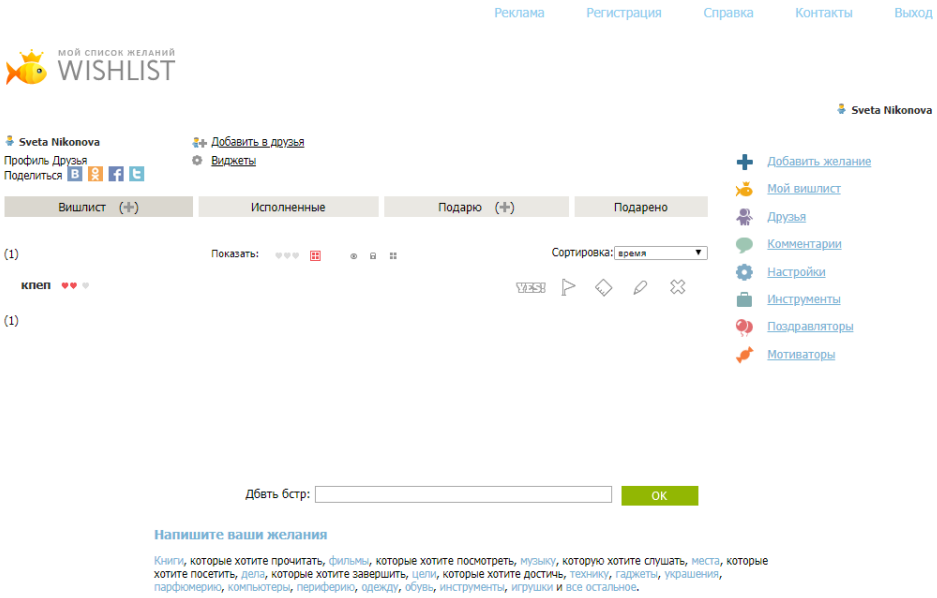

Рисунок 2 - Страница авторизованного пользователя сайта MyWishList

3. Анализ предметной области

3.1. Описание и обоснование выбора воронок конверсии

Задача разрабатываемого сайта – заставить как можно больше пользователей взаимодействовать между собой. Чем больше пользователи будут контактировать, тем больше людей захотят стать пользователями сайта. Например, если для компании друзей ни один праздник не обходится без сервиса «GoldenFish», то еще один друг просто не сможет остаться в стороне.

На основе вышесказанного можно выделить три ключевых сценария: «Добавление желания», «Добавление в друзья», «Добавление подарка». Без желаний невозможно взаимодействие пользователей, и разрабатываемый сайт теряет смысл. Чем больше друзей у пользователя, тем больше людей увидит его список желаний, а значит, выше вероятность взаимодействия. Ну и, конечно, добавление желания друга в список «Хочу подарить» - это конечная точка взаимодействия.

1. Воронка добавления желания:

Посетить раздел «Мои желания» → Нажать на кнопку «Добавить желание» → Нажать на кнопку «Сохранить»

2. Воронка добавления пользователя в друзья:

Нажать кнопку «Поиск друга» → Нажать кнопку «Добавить в друзья»

3. Воронка добавления подарка:

Перейти в раздел «Друзья» → Перейти на страницу друга → Нажать кнопку «Хочу подарить»

3.2. IDEF0

IDEF0 диаграмма предназначенная для формализации и описания бизнес-процессов.В IDEF0 рассматриваются логические отношения между функциями системы, а не их временная последовательность.

На рисунке 3 представлена контекстная диаграмма разрабатываемого приложения.

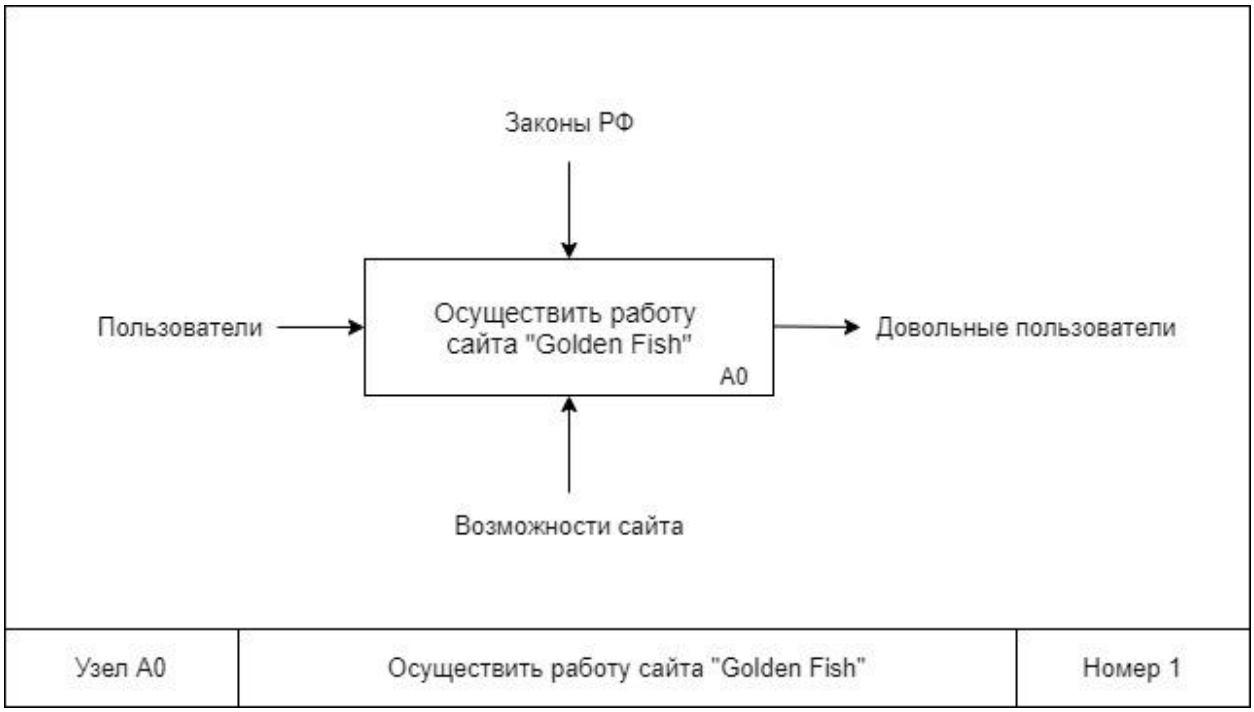

Рисунок 3 - Контекстная диаграмма разрабатываемого приложения

Работу сайта регулируют законы РФ, а именно законы о деятельности сайтов. На вход в Систему поступают пользователи, которые используют возможности сайта и получают от этого удовлетворение. На выходе Система получает довольных пользователей, которые снова захотели бы посетить сайт.

Детальное представление диаграммы верхнего уровня представлено на рисунке 4.

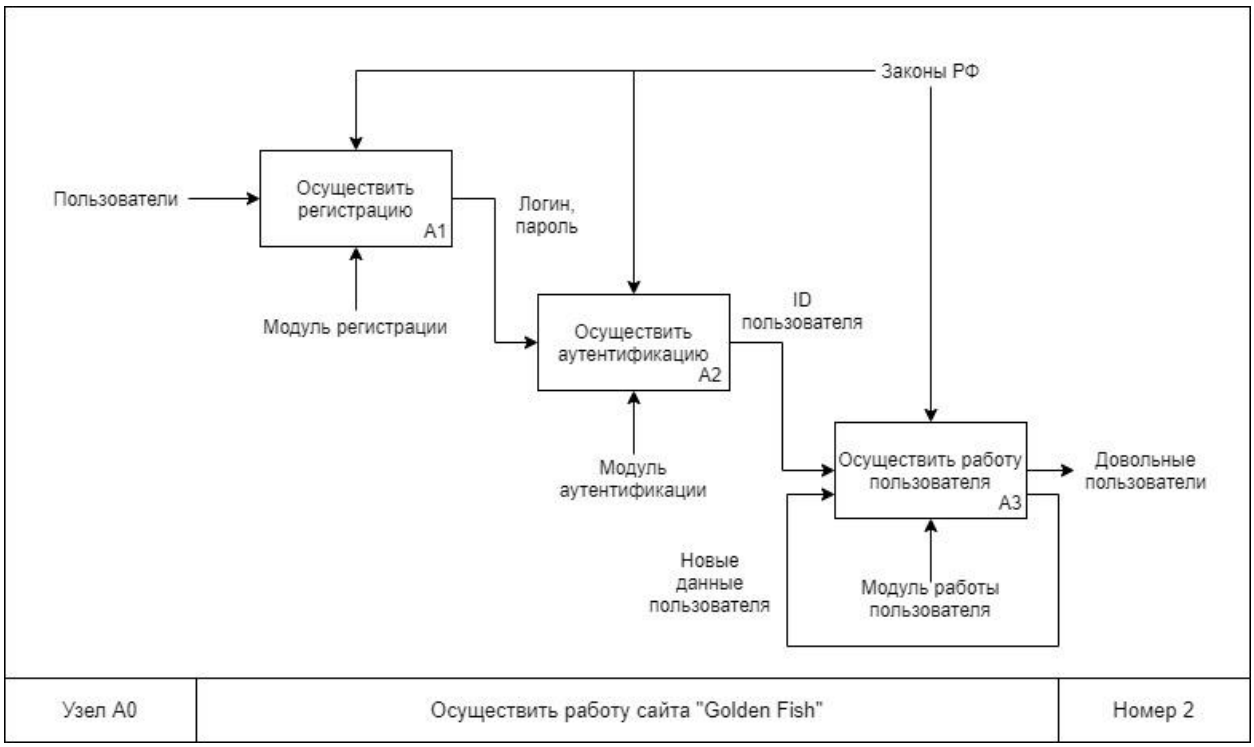

Рисунок 4 - Детальное представление диаграммы верхнего

Блок A3 нуждается в более глубокой детализации. Дочерняя диаграмма изображена на рисунке 5.

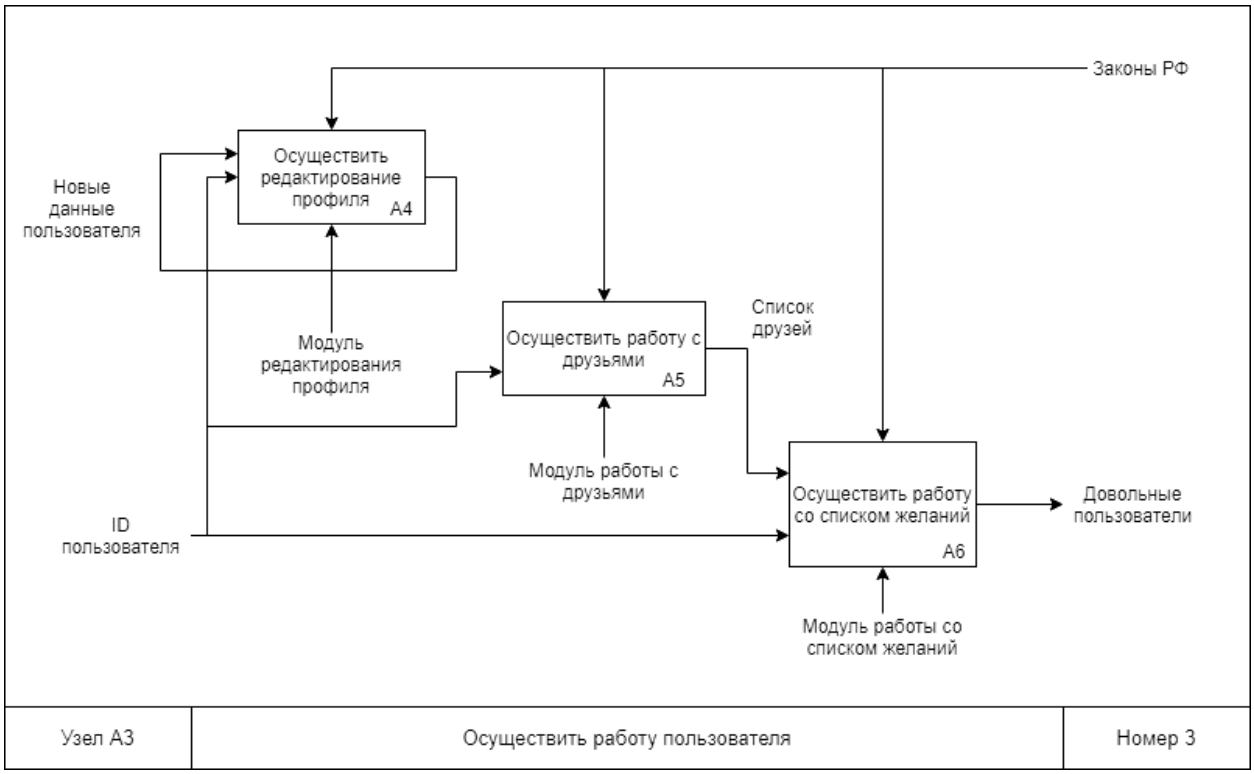

Рисунок 5 - Диаграмма блока A3

3.3. Диаграммы прецедентов

Диаграммы прецедентов играют основную роль в моделировании поведения системы. Каждая такая диаграмма показывает множество прецедентов, актеров и отношения между ними. Диаграммы прецедентов применяются для моделирования вида системы с точки зрения вариантов использования.

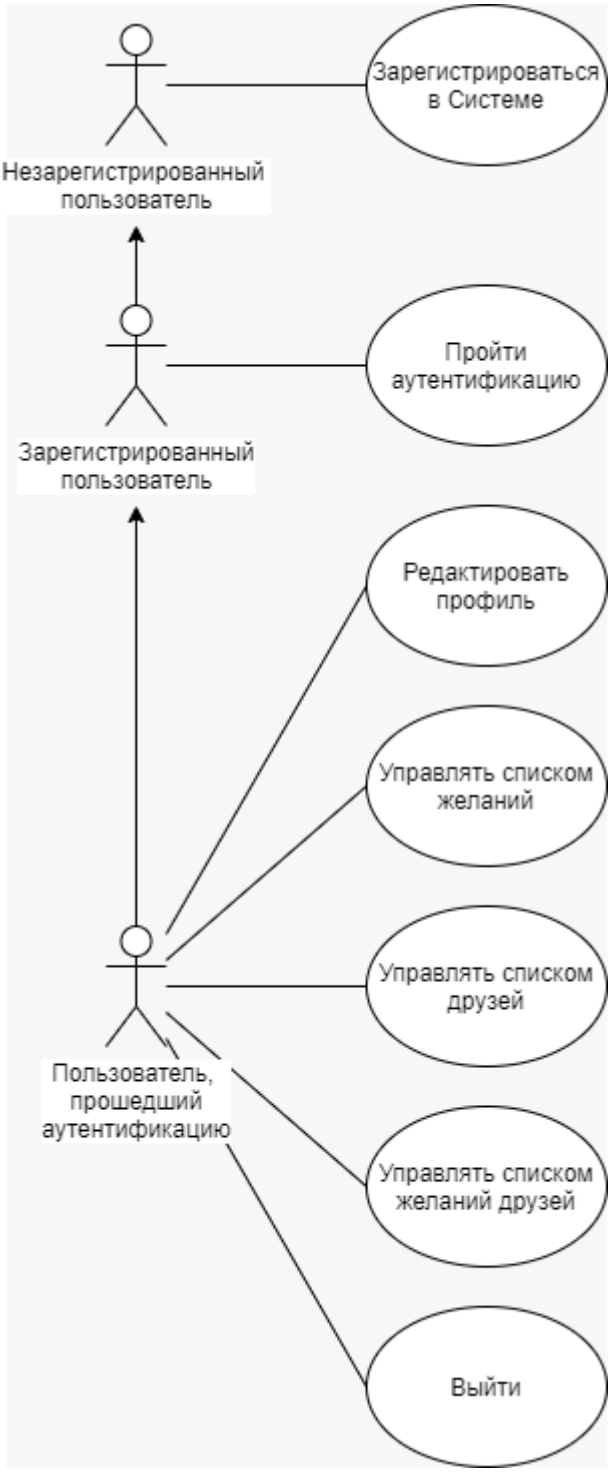

Рисунок 6 –Диаграмма прецедентов регистрации и аутентификации

Далее подробнее рассмотрим варианты использования пользователя, прошедшего аутентификацию.

Рисунок 7 – Диаграмма прецедентов редактирования профиля

Рисунок 8 – Диаграмма прецедентов управления списком желаний

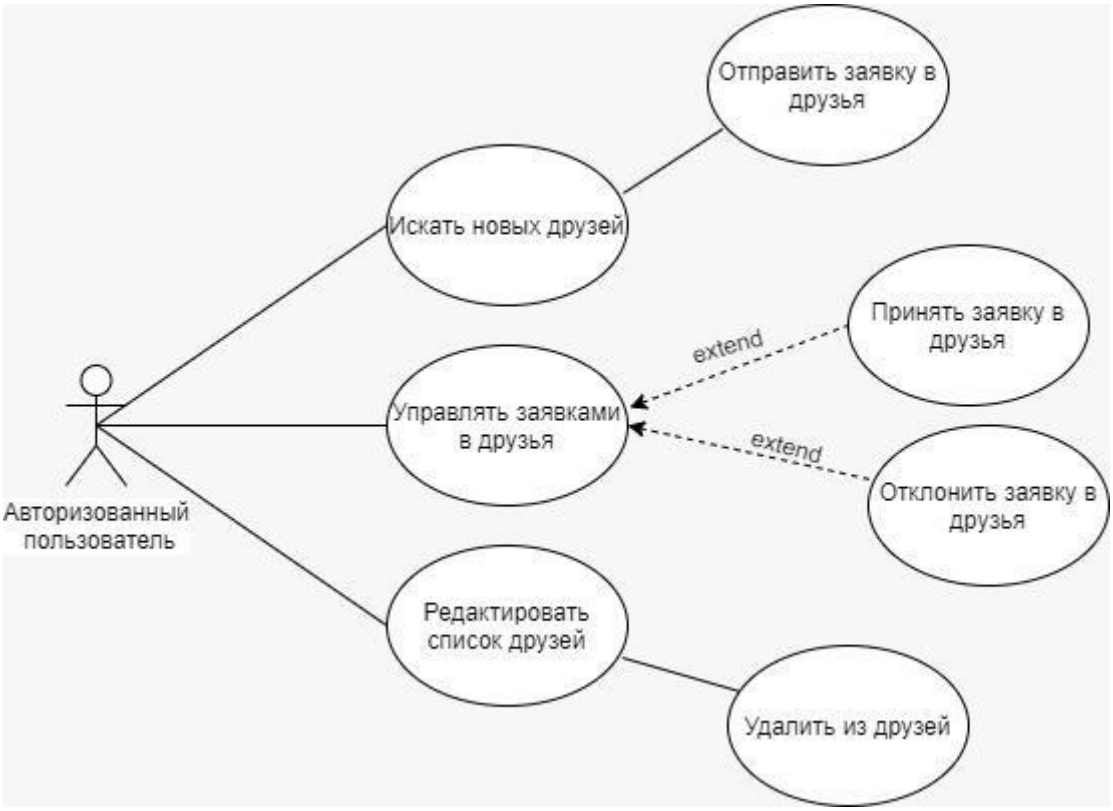

Рисунок 9 – Диаграмма прецедентов управления списком друзей

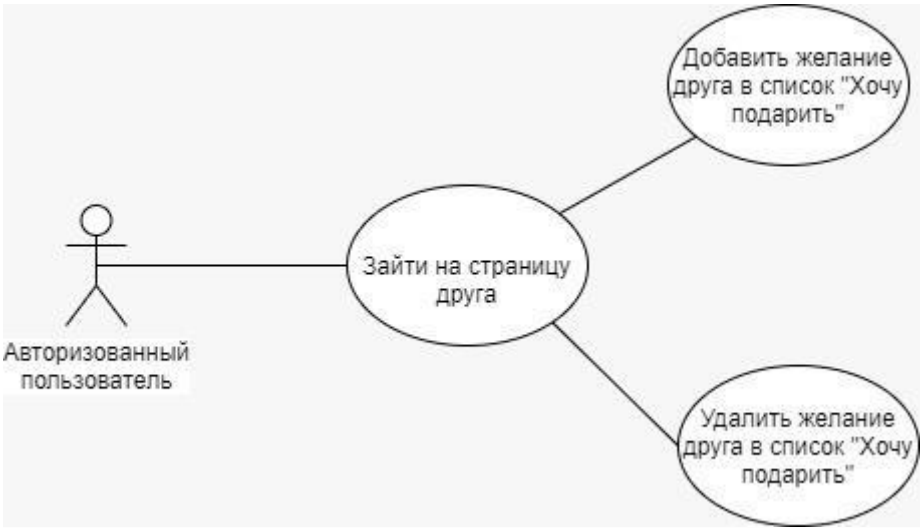

Рисунок 10 – Диаграмма прецедентов управления списком желаний друзей

3.4. Диаграмма классов

Диаграмма классов описывает систему с точки зрения ее проектирования, показывая ее структуру: строятся классы, интерфейсы и отношения между ними.

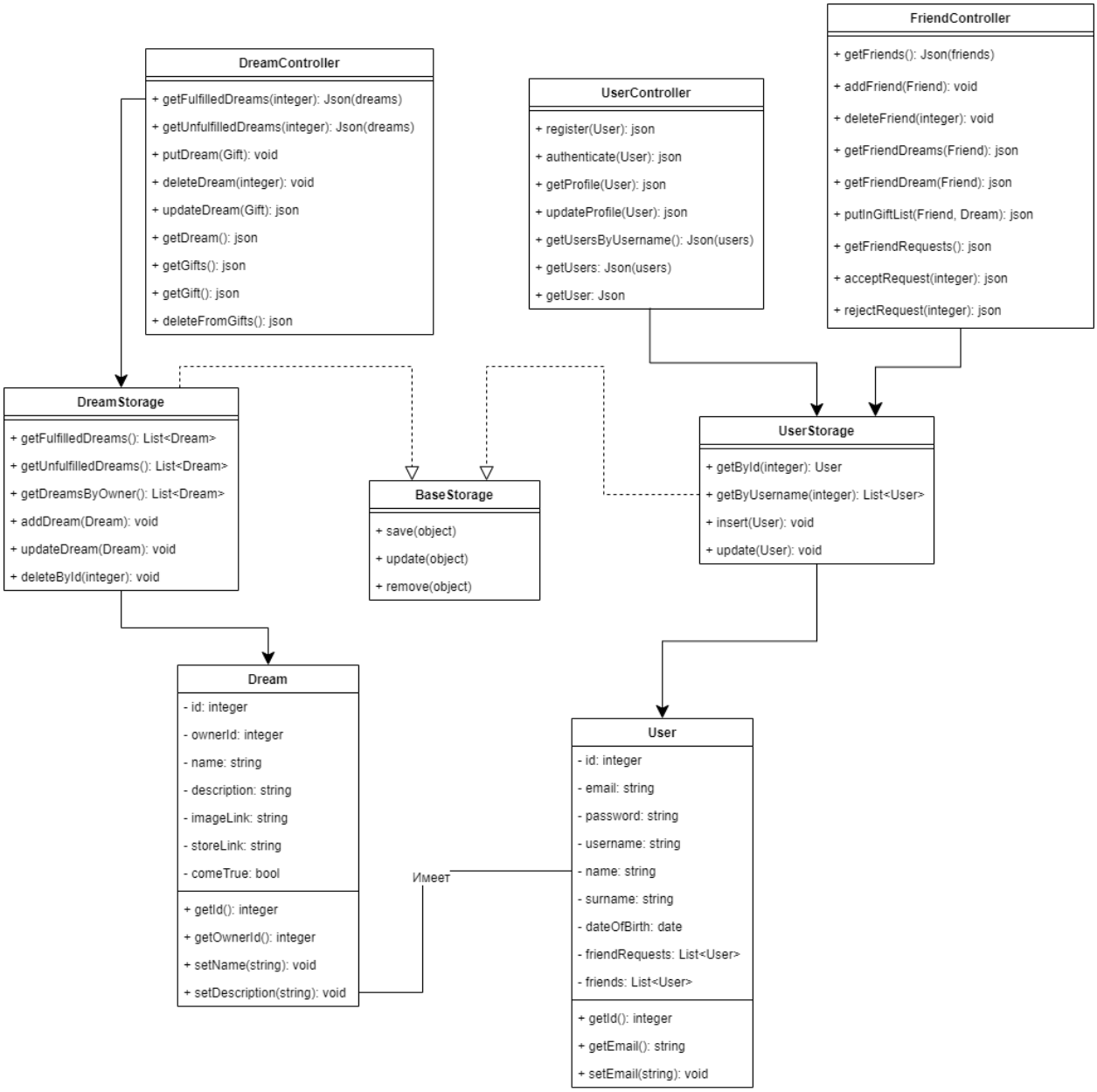

Рисунок 11 -Диаграмма классов

3.5. Диаграмма объектов

На диаграмме объектов отображаются экземпляры классов системы с указанием текущих значений их атрибутов и связей между объектами.

В разрабатываемой системе есть классы, реализующие только методы. Они соответственно представлены прямоугольниками без атрибутов.

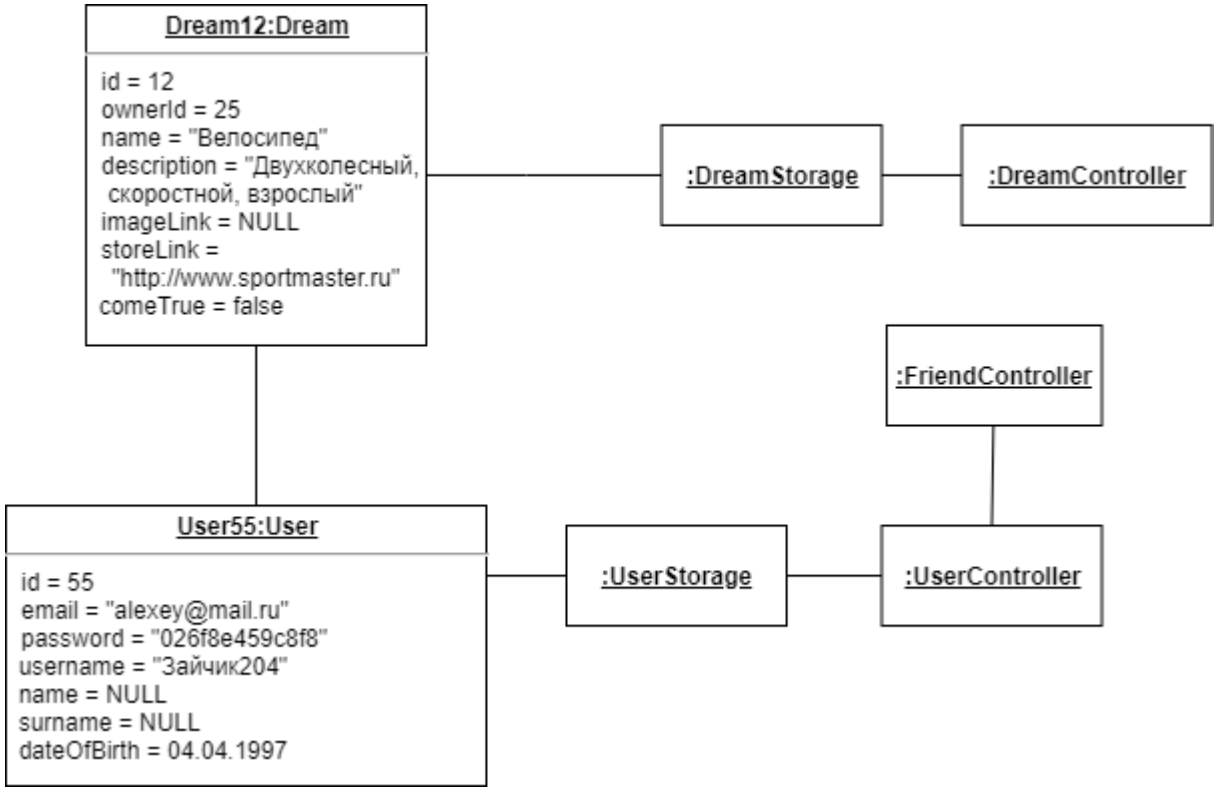

Рисунок 12 - Диаграмма объектов

3.6. Диаграммы последовательности

Для того чтобы показать, как различные объекты задействованы в приложении и как они взаимодействуют друг с другом, использованы диаграммы взаимодействия. В частности – диаграммы последовательности, которые акцентируют внимание на то, как тот или иной объект изменяется во времени, какие действия совершены и в какой последовательности.

Чтобы стать пользователем приложения, необходимо зарегистрироваться: заполнить форму запрашиваемыми данными. Если данные введены корректно, Система добавляет пользователя в базу данных и перенаправляет на главную страницу. В ином случае, пользователь получает сообщение обошибке.

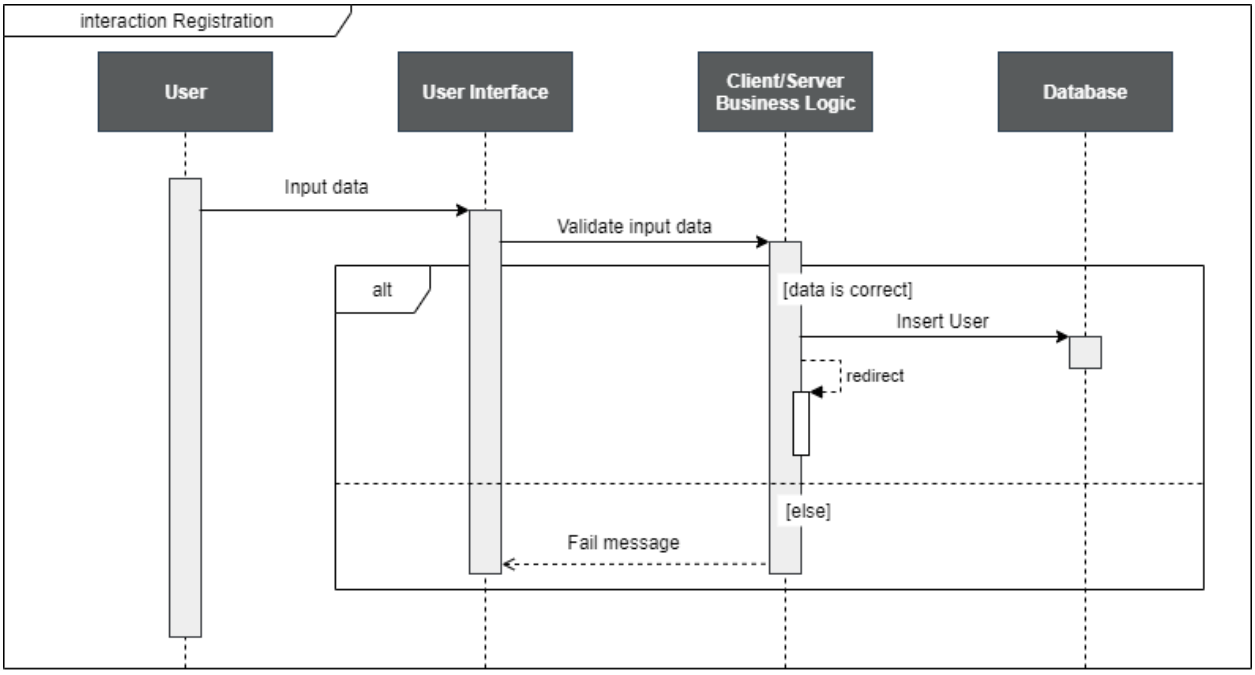

Рисунок13 -Диаграмма последовательности. Регистрация

В том случае, если пользователь уже был зарегистрирован, ему необходимо пройти аутентификацию, введя свои логин и пароль. Производится запрос к базе данных, существует ли пользователь с введенными данными. Если запрос вернул результат, пользователь попадает

на свою страницу, иначе получает сообщение об ошибке.

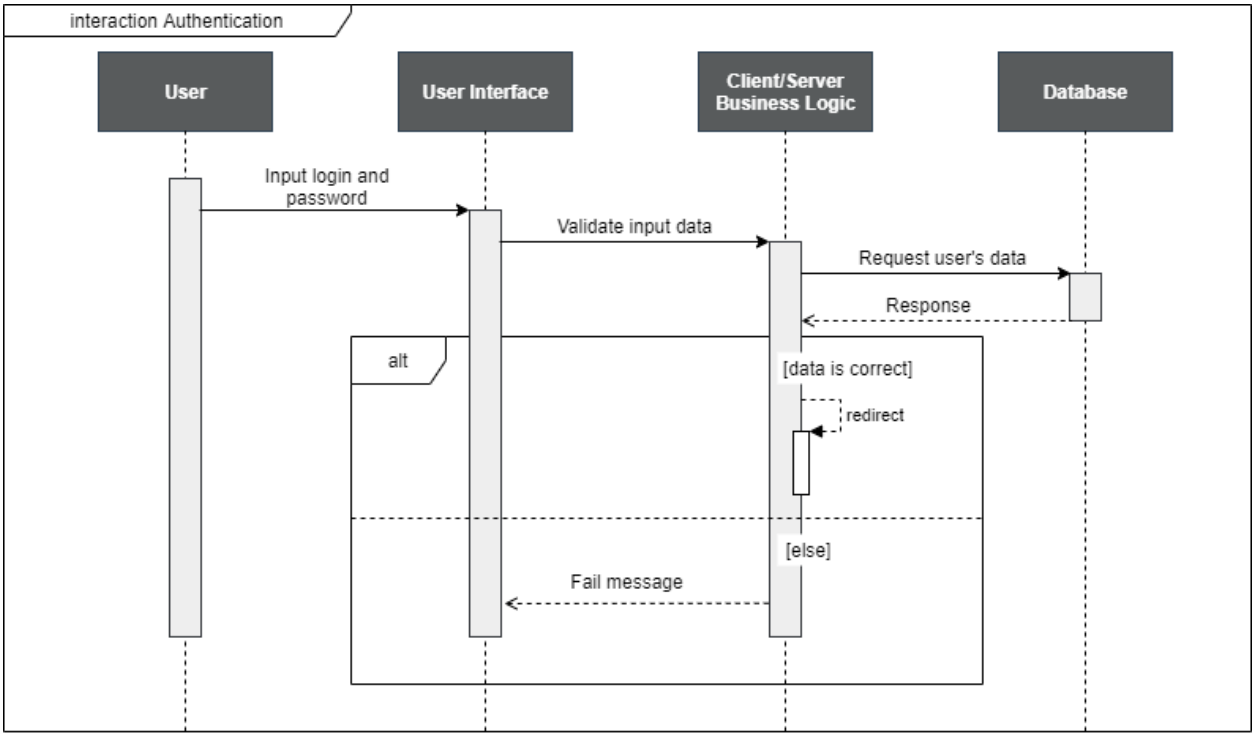

Рисунок 14 - Диаграмма последовательности. Аутентификация

Прежде всего, у авторизованного пользователя имеется возможность взаимодействия с другими пользователями приложения. Чтобы найти своего друга, зарегистрированного в Системе, необходимо выполнить поиск, в свою очередь на уровне доступа к данным производится очередной запрос. Если пользователь (пользователи) по запросу найден, ему можно отправить заявку в друзья. Если это сделано, в таблице FriendRequestсоответствующего

пользователя появляется новая запись.

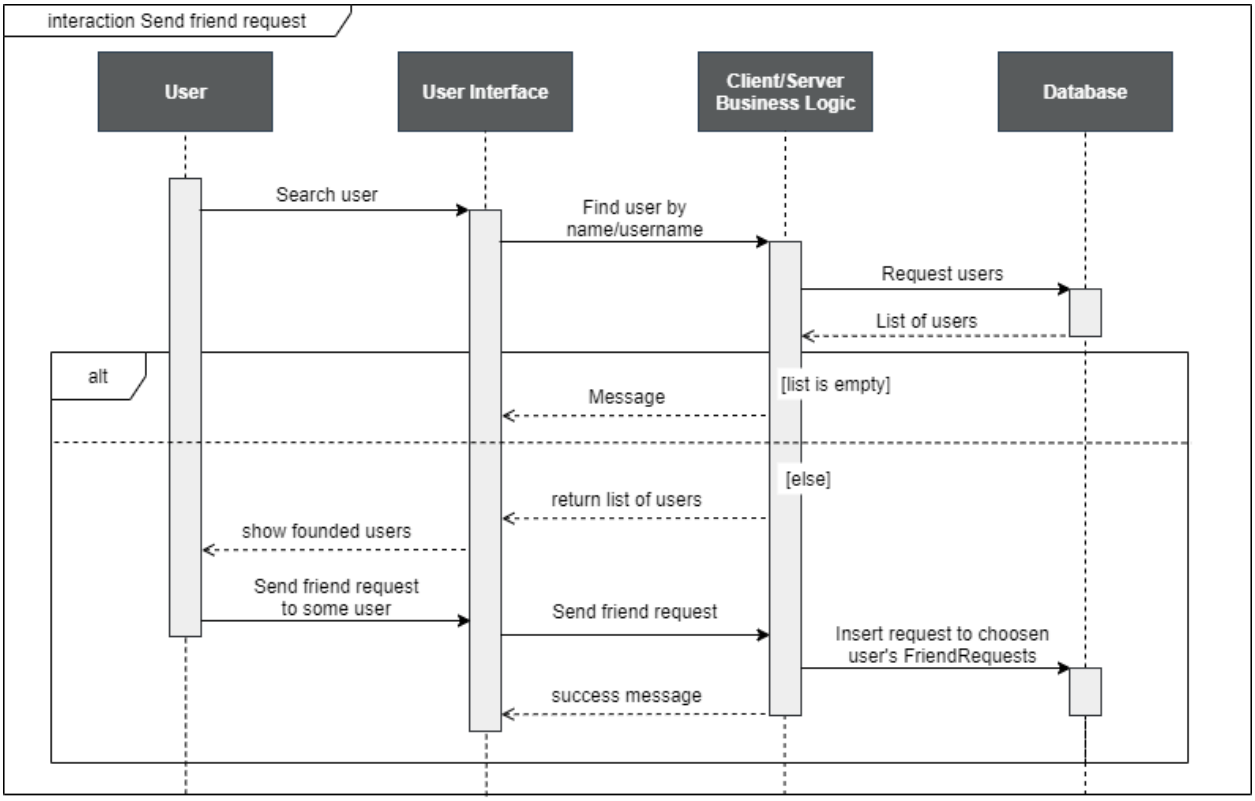

Рисунок 15 -Диаграмма последовательности. Отправление заявки «в друзья»

В том случае, если у пользователя тоже имеются активные заявки в друзья, он может принять или отклонить их, что показано на Рисунке 12.

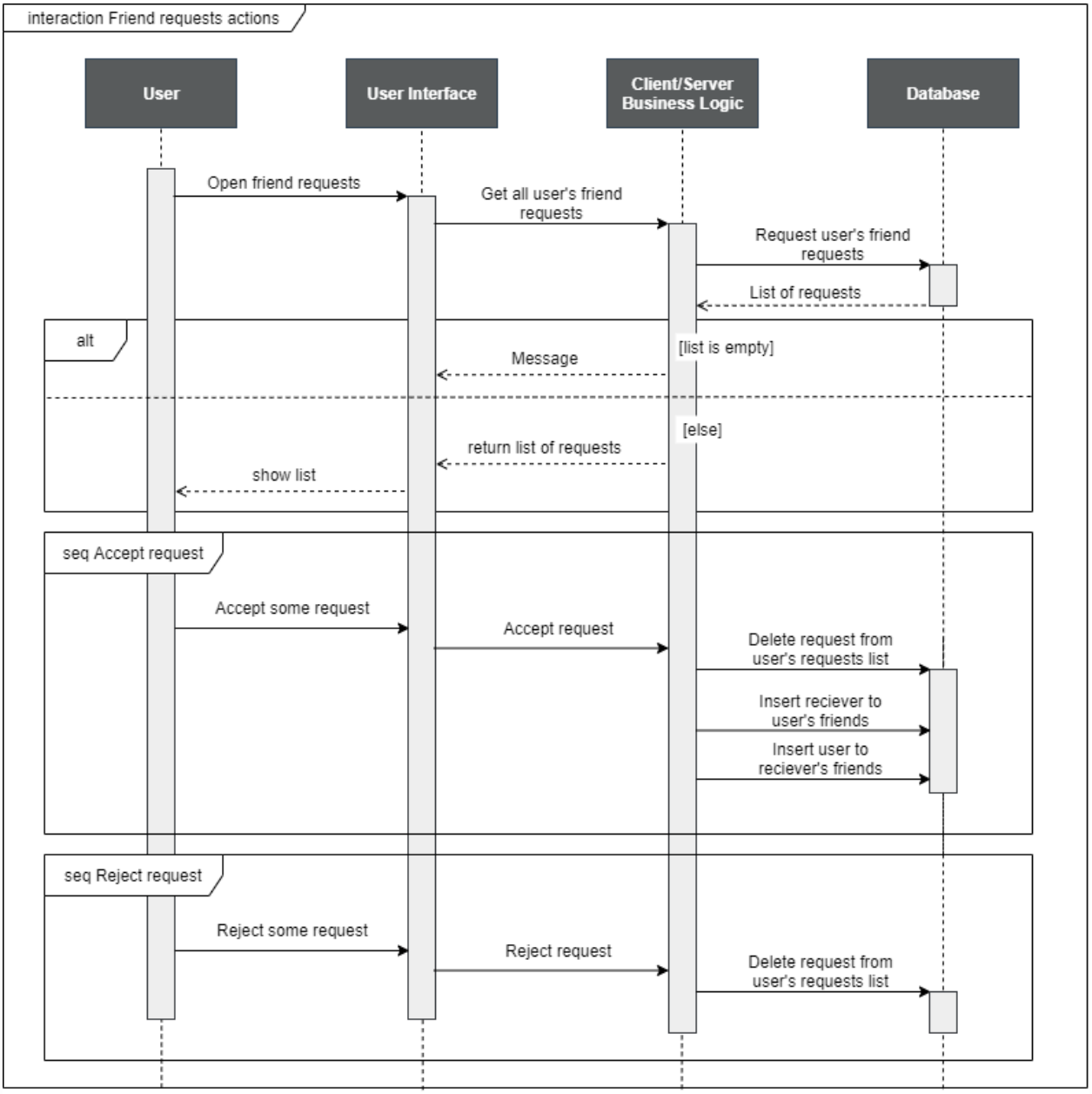

Рисунок 16 -Диаграмма последовательности. Действия с заявками «в друзья»

Наконец, пользователь может совершать действия со своими желаниями: добавлять, удалять или редактировать описание.

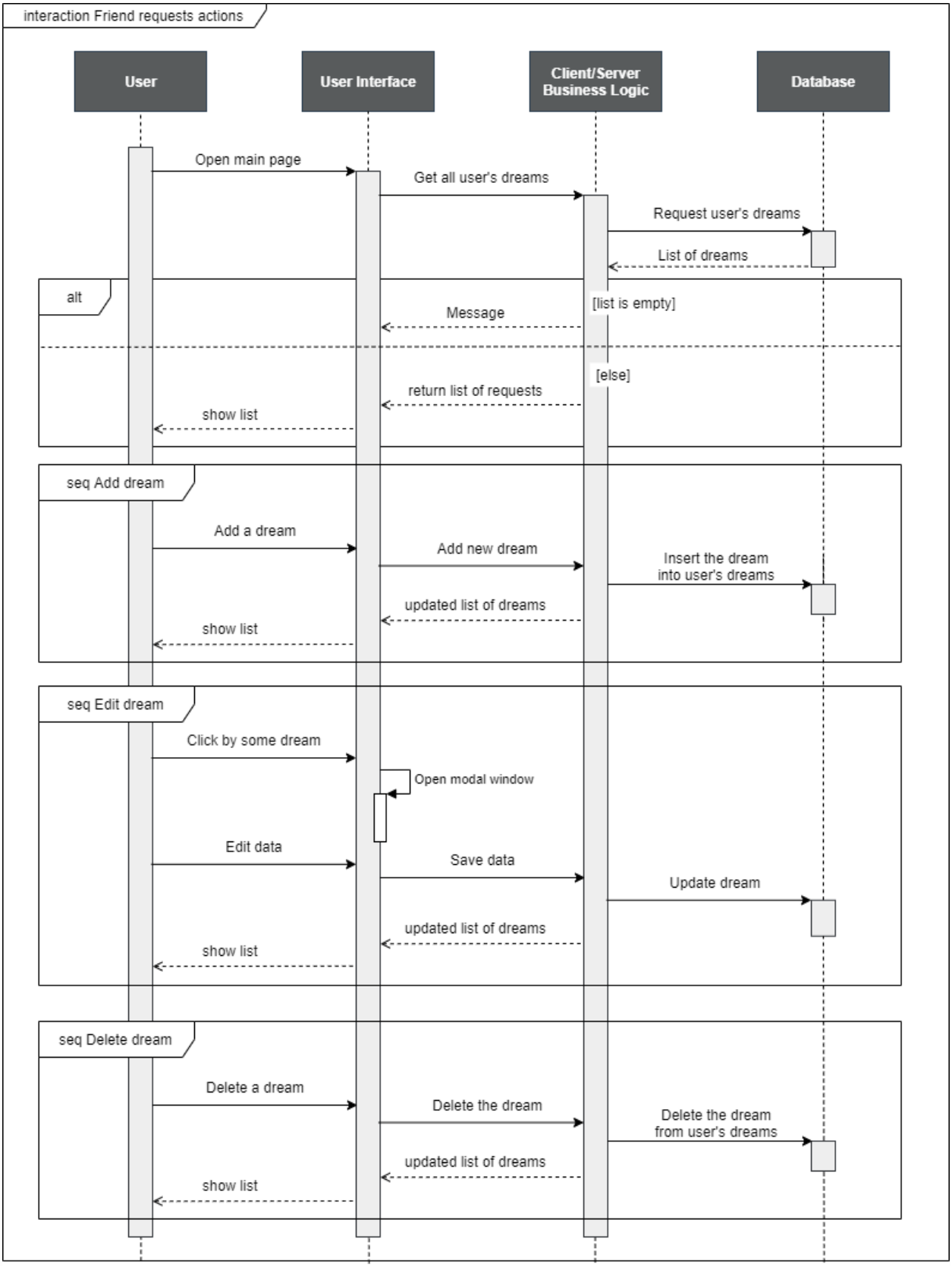

Рисунок 17 -Диаграмма последовательности. Действия с желаниями

3.7. Диаграмма коммуникации

Диаграмма коммуникации — диаграмма, на которой изображаются взаимодействия между частями композитной структуры или ролями кооперации. В отличие от диаграммы последовательности, на диаграмме коммуникации явно указываются отношения между объектами, а время как отдельное измерение не используется.

На рисунке 19 представлена диаграмма коммуникации разрабатываемой системы.

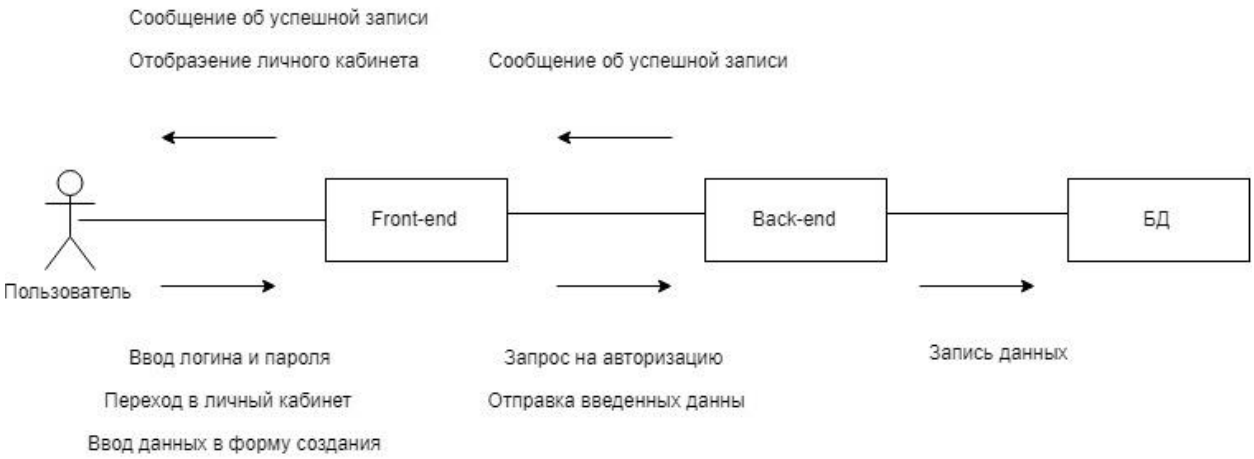

Рисунок 18 – Диаграмма коммуникации

3.8. Диаграмма состояний

Диаграмма состояний показывает, как объект переходит из одного состояния в другое. Эта диаграмма служит для моделирования динамических аспектов системы.

На рисунке 20 представлена диаграмма состояний разрабатываемой Системы.

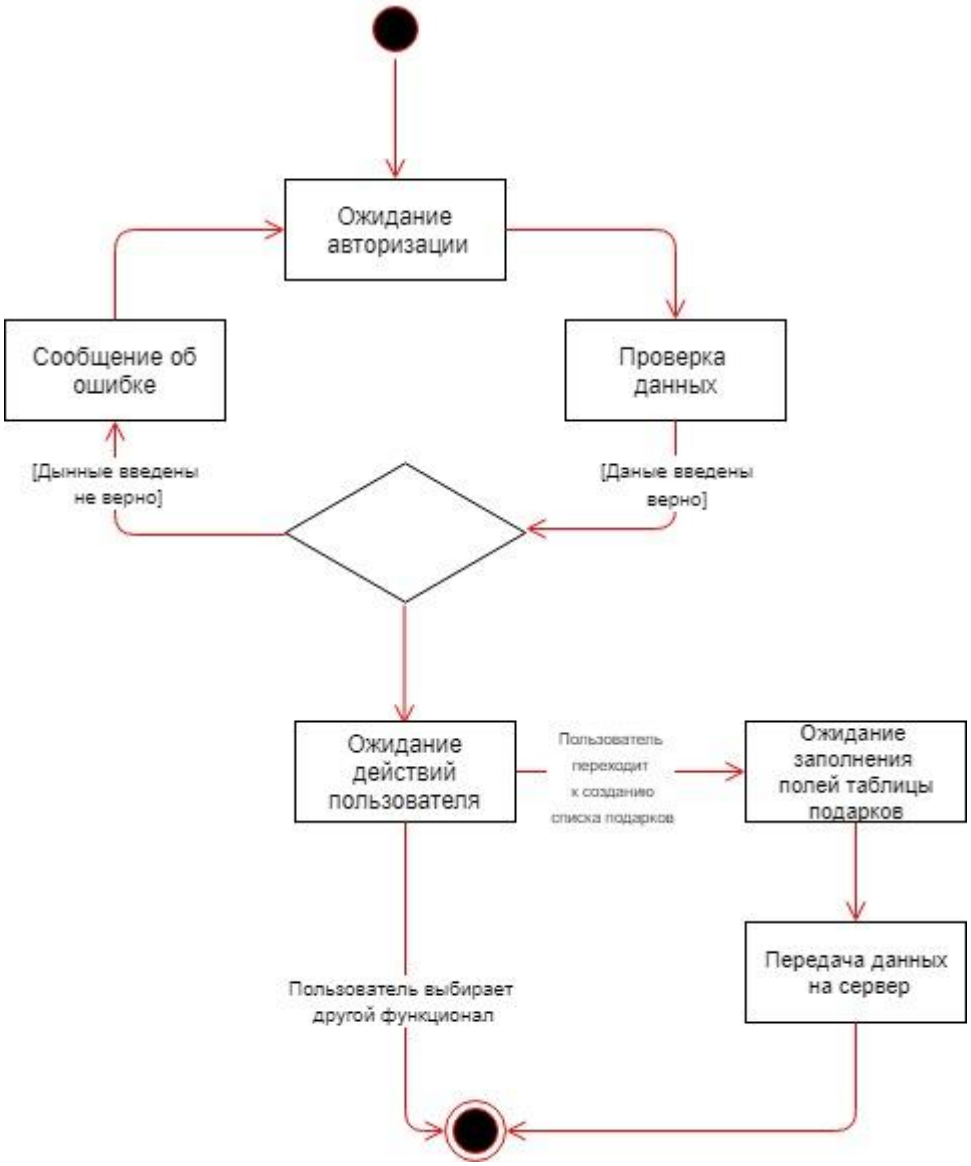

Рисунок 19 – Диаграмма состояний

3.9. Диаграмма активностей

Она, как и диаграмма состояний, отражает динамические аспекты поведения системы**.** Она представляет собой блок-схему, которая наглядно показывает, как поток управления переходит от одной деятельности к другой.

Активности на диаграмме “разбросаны” по беговым дорожкам, каждая из которых соответствует поведению одного из объектов. На диаграмме активностей разрабатываемой Системы представлены четыре дорожки (пользователь, приложение, сервер, БД). Взаимодействие между дорожками отображено на рисунке 21.

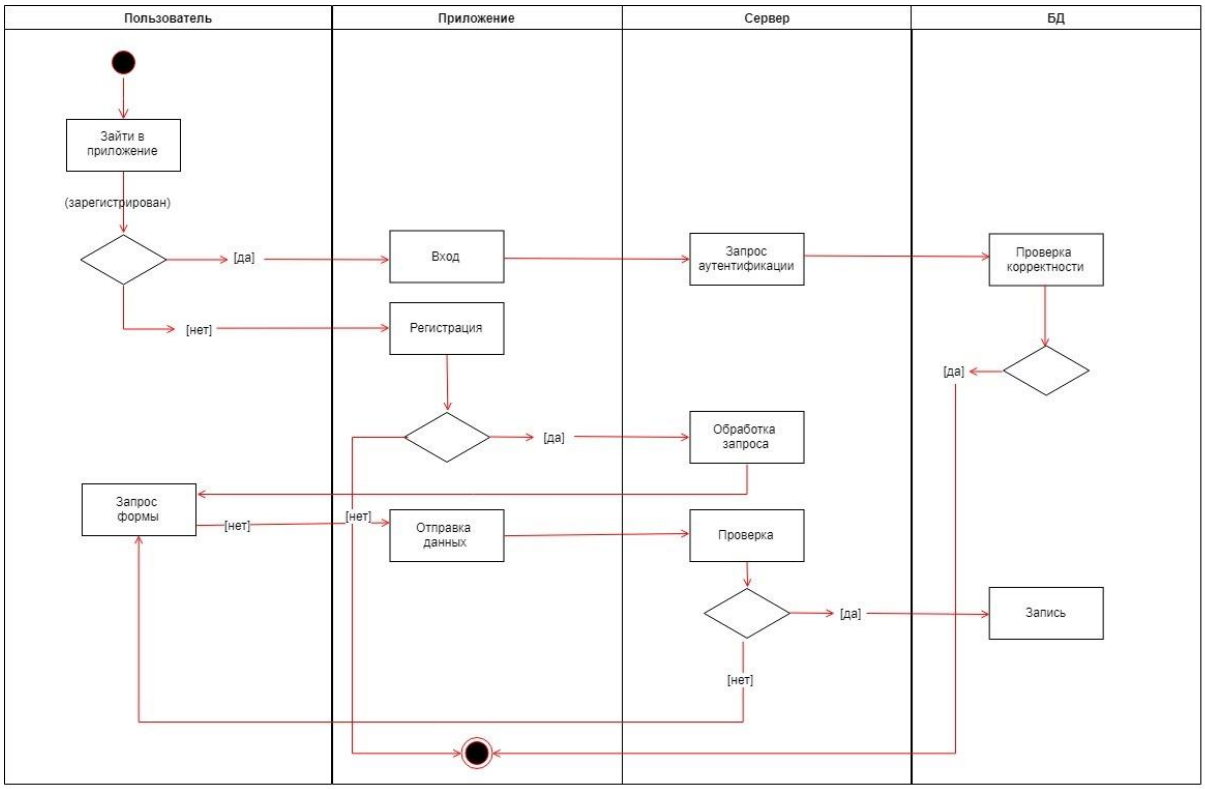

Рисунок 20 – Диаграмма активностей

3.10. Диаграмма развертывания

Диаграмма развертывания моделирует физическое развертывание компонентов приложения (артефактов) на различных аппаратных компонентах. Так, разрабатываемое web-приложение, архитектурные модули которого также показаны на следующей диаграмме, включает запуск сервера Flask-приложения и сервера базы данных PostgreSQL. Их взаимодействие обеспечивает RESTAPI. Все в совокупности планируется развернуть на одном из облачных серверов.

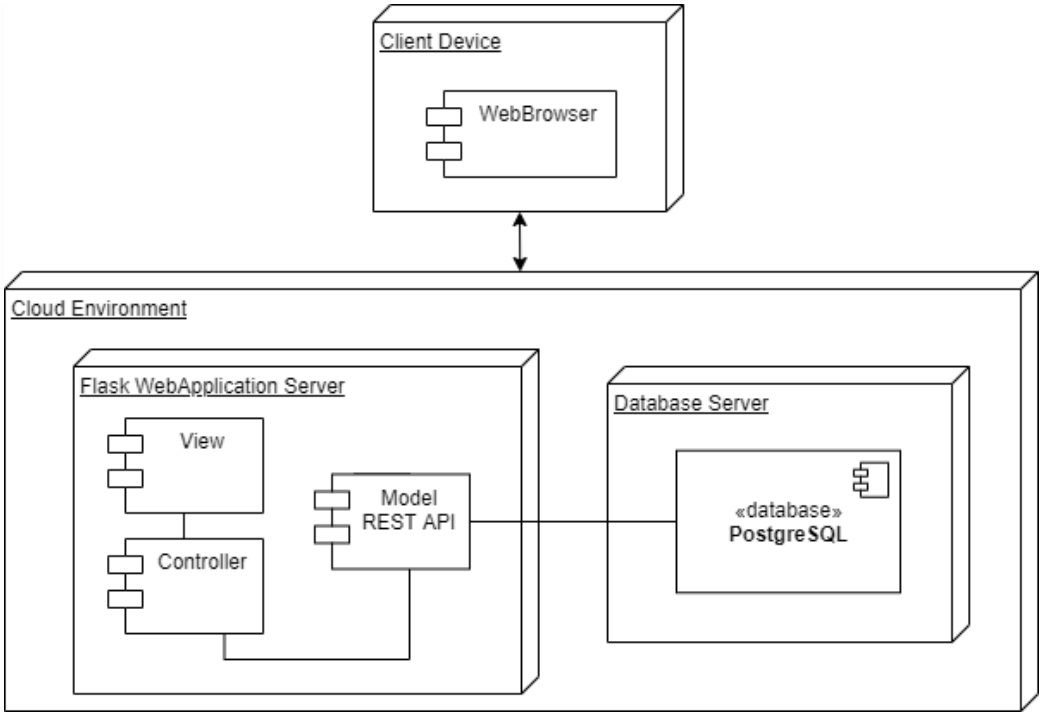

Рисунок 21 -Диаграмма развертывания

3.11. Обоснование архитектуры проекта

Архитектура приложения основана на шаблоне программирования MVC (Model-View-Controller), который предполагает разделение кода на 3 части:

– Model хранит данные, необходимые для работы Системы и предоставляет методы для работы с ними.

– Controller обеспечивает связь между пользователем и Системой: обращается к Model для получения или изменения данных и передает их View.

– View отвечает за отображение переданных данных в графическом интерфейсе.

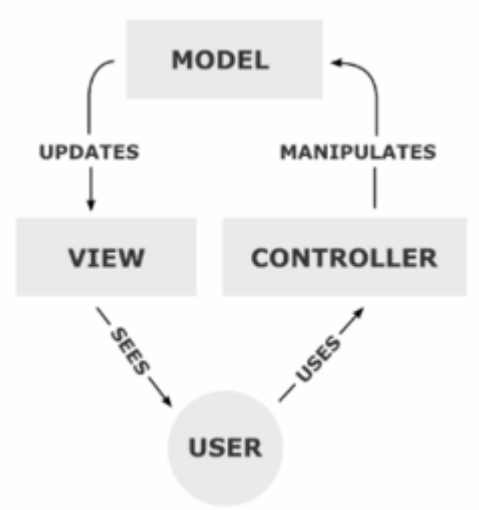

Рисунок 22 - Схема шаблона проектирования MVC

К плюсам можно отнести:

- Структурированный, гибкий код
- Хорошая расширяемость
- Легкость в отладке и тестировании

Минусами данной концепции являются:

- Увеличение объема кода
- Необходимость использования большего количества ресурсов

Выбор данного шаблона проектирования оправдан большим количеством плюсов и готовностью пренебречь минусами.

Архитектура разрабатываемого приложения схематично представлена на рисунке ниже.

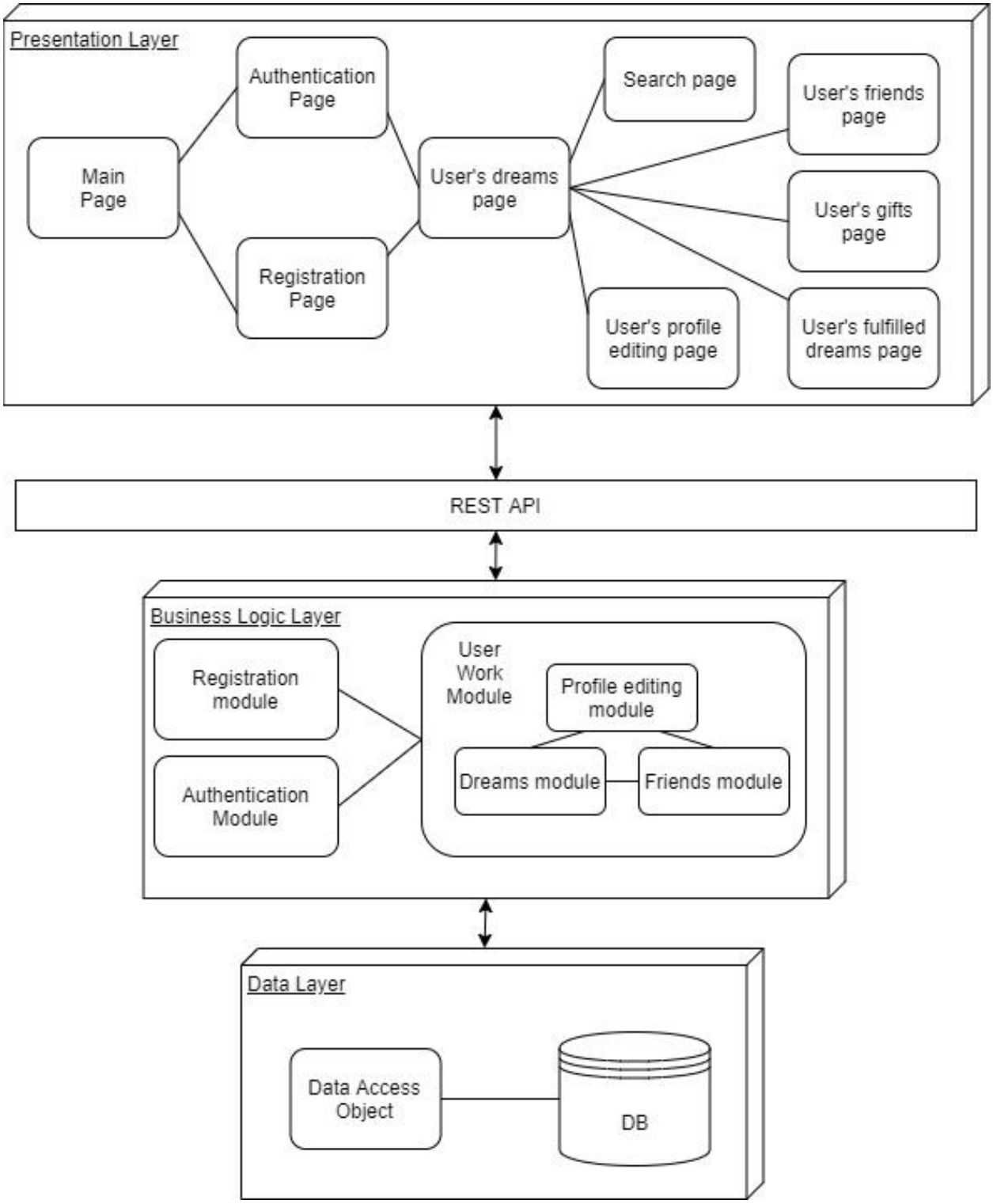

Рисунок 23 – Архитектура разрабатываемого приложения

3.12. ER-диаграмма и схема базы данных

ER-диаграмма показывает, как разные «сущности» (люди, объекты, концепции и так далее) связаны между собой внутри системы. На рисунке 25 представлена ER-диаграмма разрабатываемого приложения. Она состоит из пяти сущностей: пользователи, друзья, заявки в друзья, желания. Все связи на диаграмме в отношении «один-ко-многим».

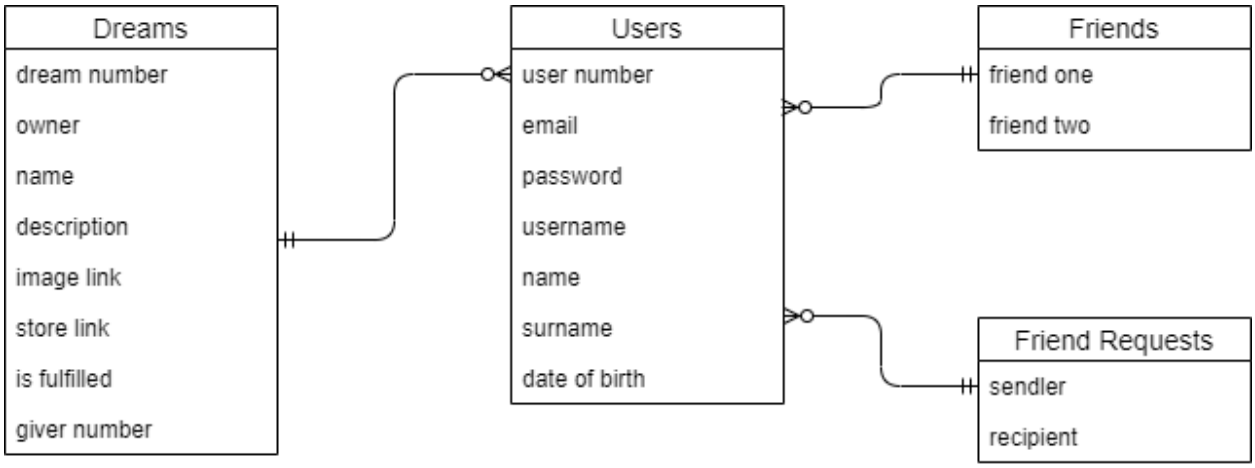

Рисунок 24 - ER-диаграмма приложения

Данная ER диаграмма преобразовывается в следующую схему базы, представленную на рисунке 26.

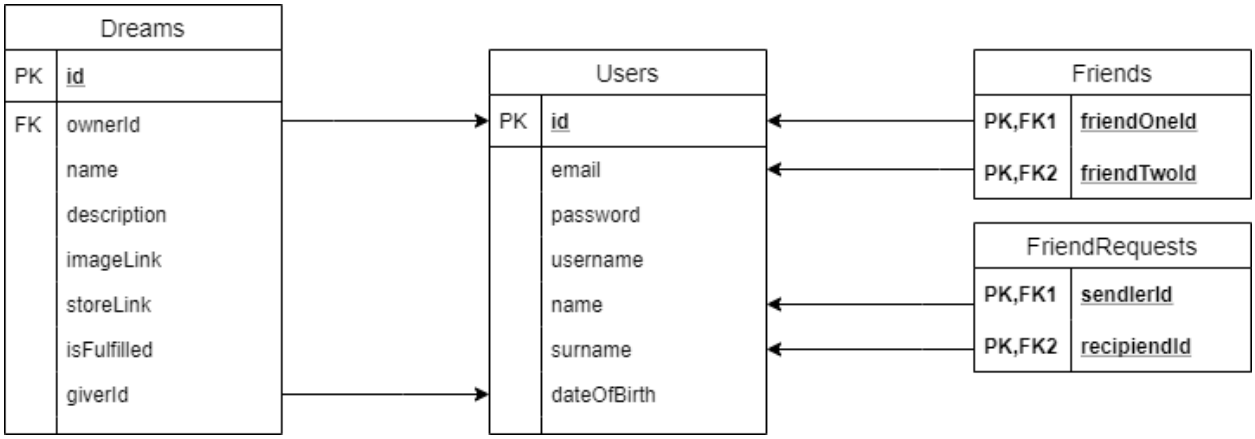

Рисунок 25 - Схема базы данных приложения

База данных состоит из пяти таблиц:

1. Users

- Уникальный идентификатор пользователя в Системе (первичный ключ)
- Электронная почта

- Пароль
- Псевдоним пользователя в Системе
- Имя
- Фамилия
- Дата рождения (необязательное поле)

2. Friends

- Идентификатор первого друга (ссылается на уникальный идентификатор в таблице Users)
- Идентификатор второго друга (ссылается на уникальный идентификатор в таблице Users)

(friendOneId, friendTwoId) – составной первичный ключ.

3. FriendsRequests

- Уникальный идентификатор пользователя, от которого исходит заявка (ссылается на уникальный идентификатор в таблице Users)
- Уникальный идентификатор пользователя, которому приходит заявка (ссылается на уникальный идентификатор в таблице Users)

(sendlerId, recipientId) – составной первичный ключ.

4. Dreams

- Уникальный идентификатор мечты
- Идентификатор пользователя, имеющего эту мечту (ссылается на уникальный идентификатор в таблице Users)
- Название мечты
- Описание (необязательно поле)
- Ссылка на картинку/фотографию желания (необязательное поле)
- Ссылка на интернет-магазин, в котором можно приобрести подарок (необязательное поле)
- Флаг, указывающий сбылась ли мечта

- Уникальный идентификатор пользователя, планирующего исполнить это желание (необязательное поле)

4. Реализация

4.1. Backend

Главной особенностью приложения является реализация его взаимодействия с клиентским устройством конечного пользователя по REST-протоколу. Это значит, что ресурсами, хранящимися в базе данных и открытыми для общего доступа, необходимо манипулировать в соответствии с тем запросом, который был послан клиентом. В REST-протоколе выделяют 4 основных метода: GET (получение ресурса), POST (создание нового ресурса), PUT (обновление существующего ресурса), DELETE (удаление ресурса). Конечно, в действительности этих методов много больше.

Такими ресурсами в Системе являются пользователи (/users, /friends) и желания (/mywishes, /fulfilled, /gifts) и несколько других узкоспециализированных, описанных ниже. Так называемые конечные точки созданы в зависимости от назначения того или иного ресурса:

- /users – список пользователей Системы
- /friends – список пользователей-друзей авторизованного в Системе пользователя
- /mywishes – список желаний авторизованного пользователя
- /fulfilled – желания авторизованного пользователя, отмеченные как «исполнившиеся»
- /gifts – список подарков – желаний друзей, отмеченных авторизованным пользователем
- /profile – информация об авторизованном пользователе
- /registration – создание нового пользователя
- /authentication – аутентификация пользователя в Системе

Итак, для каждого из ресурсов был реализован необходимый метод для манипулирования:

Модуль работы с пользователями

- GET /users – получение списка всех пользователей системы

- POST /users – поиск пользователя по юзернейму, который передается в теле запроса
- GET /profile – получение информации об авторизованном пользователе
- PUT /profile – обновление данных о пользователе, необходимые параметры (email, имя, фамилия, юзернейм, дата рождения) передаются в теле запроса
- POST /registration – создание нового пользователя
- POST /authentication – прохождение аутентификации

Модуль работы с друзьями

- PUT /users/{id} – отправление заявки «в друзья» пользователю с конкретным id
- GET /friends – получение списка друзей пользователя
- GET /friends/{id} – получение списка желаний друга с конкретным id
- DELETE /friends/{id} – удаление друга с конкретным id
- PUT /friends/{friend_id}/{dream_id} – добавление конкретного желания конкретного друга в собственный список «Хочу подарить»
- GET /friends/requests – получение списка заявок «в друзья» текущего пользователя
- PUT /friends/requests/{sender_id} – принятие заявки «в друзья»
- DELETE /friends/requests/{sender_id} – отклонение заявки «в друзья»

Модуль работы с желаниями

- GET /mywishes – получение списка желаний текущего пользователя
- GET /mywishes/{dream_id} – получение конкретного желания из списка
- POST /mywishes – создание нового желания

- PUT /mywishes/{dream_id} – обновление существующего желания
- DELETE /mywishes/{dream_id} – удаление существующего желания
- GET /fulfilled – получение списка исполненных желаний
- GET /gifts – получение списка желаний друзей, отмеченных текущим пользователем
- GET /gifts/{gift_id} – получение конкретного желания из списка «Хочу подарить»
- DELETE /gifts/{gift_id} – удаление желания из списка «Хочу подарить»

Поскольку приложение спроектировано согласно архитектуре MVC, реализация всех вышеперечисленных методов находится на уровне контроллера. Именно сюда приходят запросы с клиента, которые соответствующим образом обрабатываются. Для доступа к базе данных применена технология объектно-реляционного отображения (ORM). Основными сущностями в Системе являются User, Dream. База данных имеет соответствующие таблицы для сущностей, а также таблицы, отражающие отношения между пользователями – friend, friend_request. На уровне модели находятся объекты-сущности, а также специальные классы, реализующие CRUD-операции с данными: чтение, сохранение, обновление, удаление. Например, GET-запрос на получение желаний пользователя обращается к ресурсу /mywishes. Контроллер получает запрос, обращается с помощью объекта DreamStorage (класс для доступа к таблице dream) к базе данных, запрашивая список всех желаний пользователя с данным ID, которые не отмечены как исполнившиеся. Тот же объект доступа к данным возвращает список объектов Dream, которые контроллер преобразует в список JSON-объектов и возвращает в качестве ответа от сервера. Подобным образом реализованы все вышеперечисленные методы.

4.2 Frontend

Главная страница сайта ведет к страницам регистрации и аутентификации.

Рисунок 26 – Главная страница сайта

Форма для регистрации содержит обязательные и необязательное поле. Если не все обязательные поля заполнены, выводится соответствующее сообщение. При нажатии на кнопку «Зарегистрироваться» на сервер отправляется POST запрос к API-ресурсу /registration, тело содержит введенные данные в формате JSON. Данные обрабатываются: инициализируется объект User, который затем сохраняется в базу данных. На основе введенного пароля генерируется хэш, которые и сохраняется в базе данных. Генерируется уникальный токен доступа для авторизации пользователя в Системе, который возвращается клиенту в теле ответа.

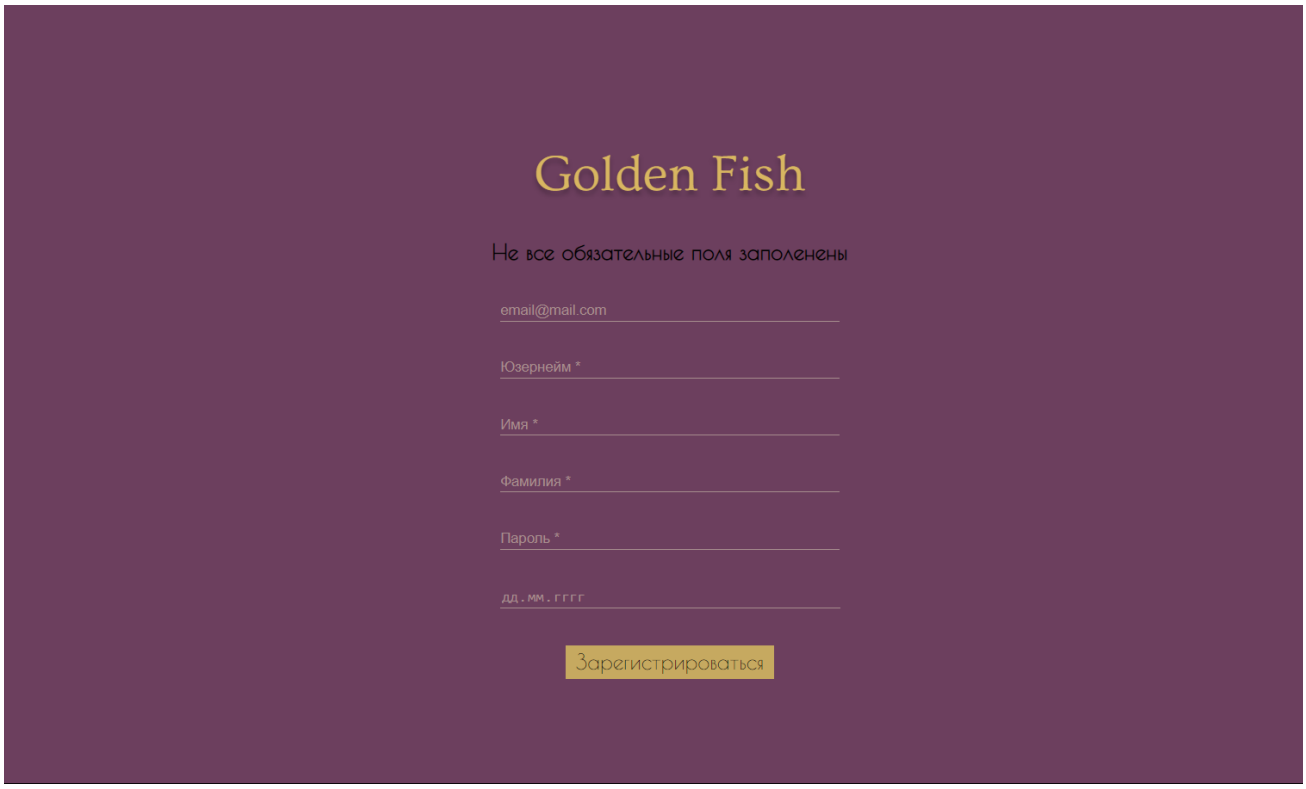

Рисунок 27 – Страница регистрации. Сообщение об ошибке

Форма аутентификации имеет стандартный вид: пользователь должен ввести email и пароль. Переданные серверу, данные сверяются с базой данных: проверяется их наличие, а также вновь сгенерированный хэш сверяется с уже хранящимся в базе. Если в базе данных найдена идентичная запись, статус ответа от сервера равен 200 и пользователь успешно авторизуется – тело ответа также содержит токен. Если нужная запись не найдена, сервер возвращает статус код ошибки по умолчанию – 400, а пользователь получает сообщение о неверно введенных данных.

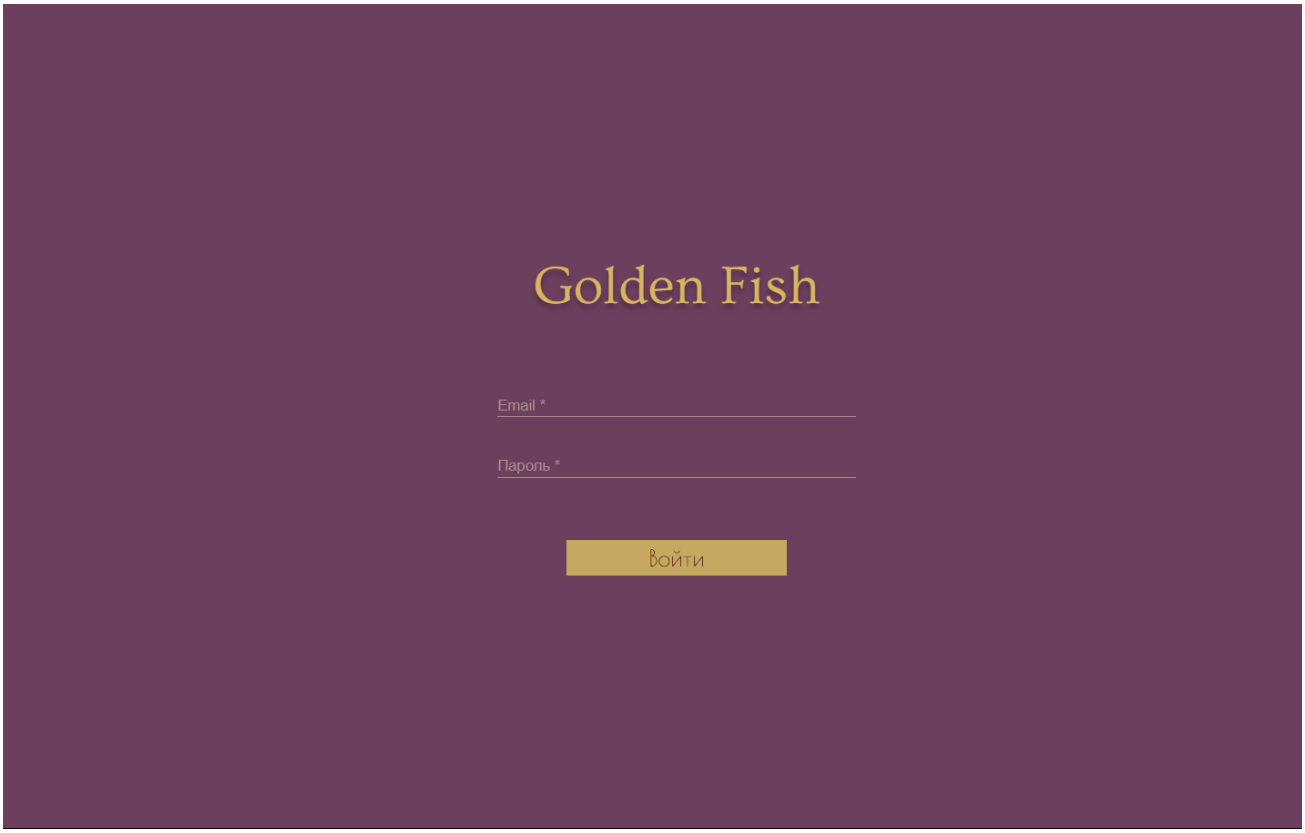

Рисунок 28 –Страница аутентификации

После успешной авторизации Система перенаправляет пользователя на его личную страницу редактирования своих желаний. Ресурс поддерживает четыре вида запроса: GET, POST, PUT, DELETE. Соответственно, свое желание на данной странице можно просмотреть, создать новое, обновить описание и удалить.

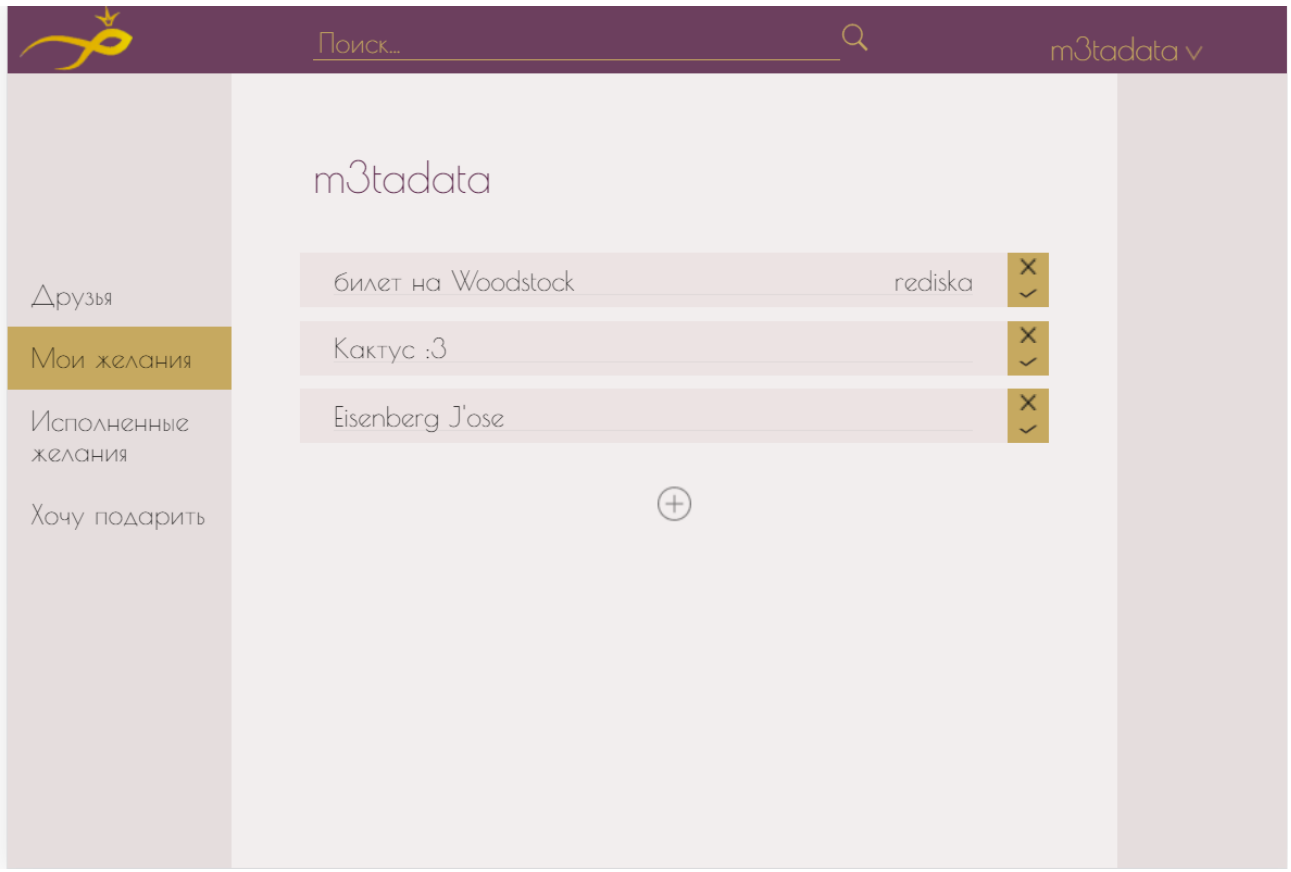

Рисунок 29 –Личная страница пользователя. Мои желания

При создании нового желания предлагается заполнить следующие поля (Рисунок 31). Только поле для ввода названия является обязательным. Пользователь нажимает на «Сохранить», к серверу отправляется POST запрос. Создается инициализированный объект Dream, который сохраняется в базу данных. В теле ответа содержится созданный объект в формате JSON.

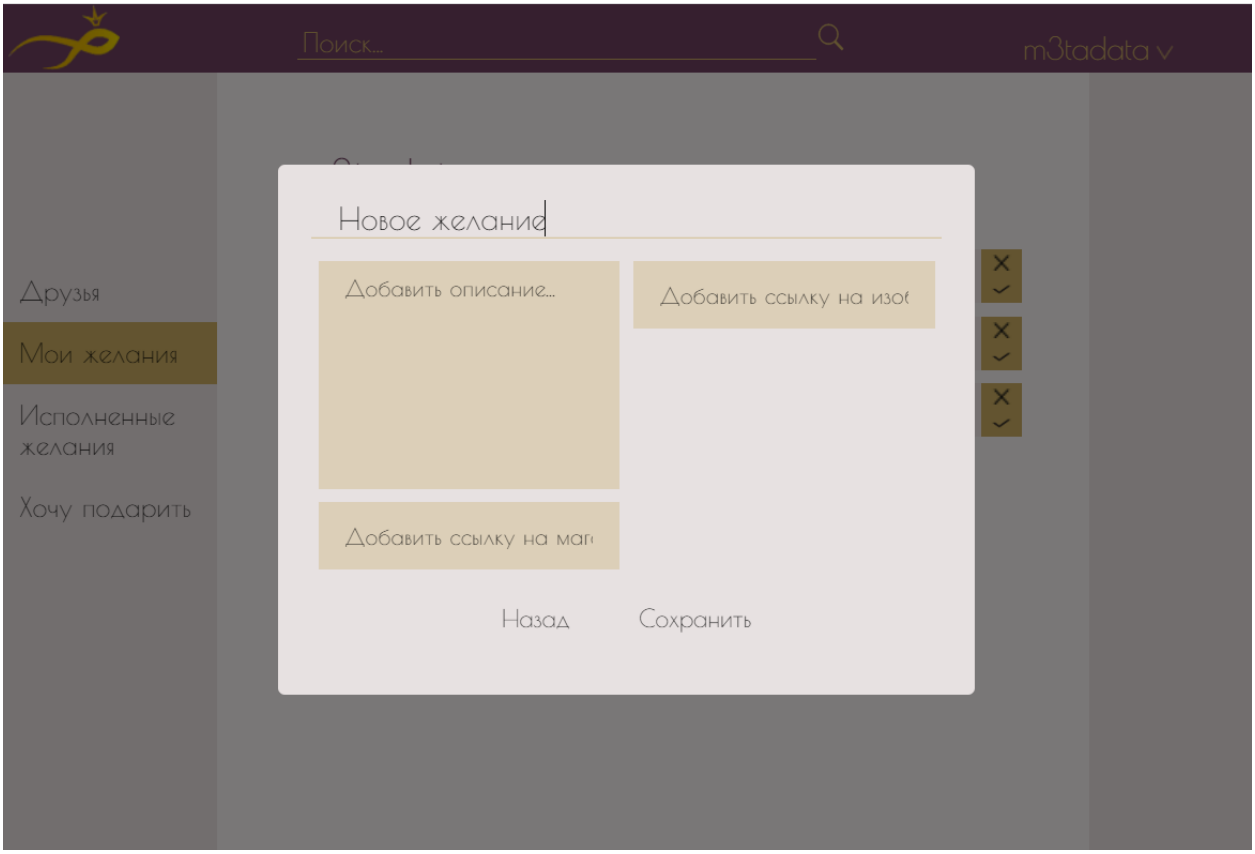

Рисунок 30 –Личная страница пользователя: создание нового желания

Развернуть сервер приложения и сервер базы данных было решено на облачном хостинге Heroku. Интерактивная документация на Swagger по разработанному REST API находится на домене приложения и позволяет наглядно увидеть, какие параметры принимает тот или иной запрос и что содержит тело ответа, а также отправить тестовые запросы к серверу.

4.3. Аналитика

К сайту была подключена система аналитики от Яндекс.Метрики. На ее основе будут изучаться данные о посещениях и посетителях сайта с целью улучшения и оптимизации работы ресурса.

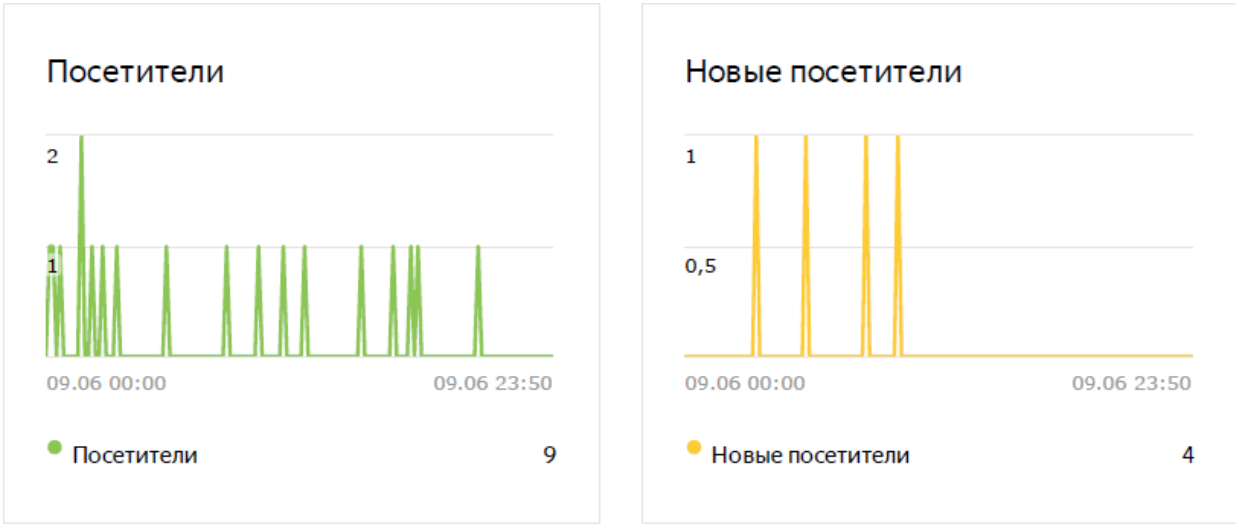

Рисунок 31 – Графики посетителей в системе аналитики

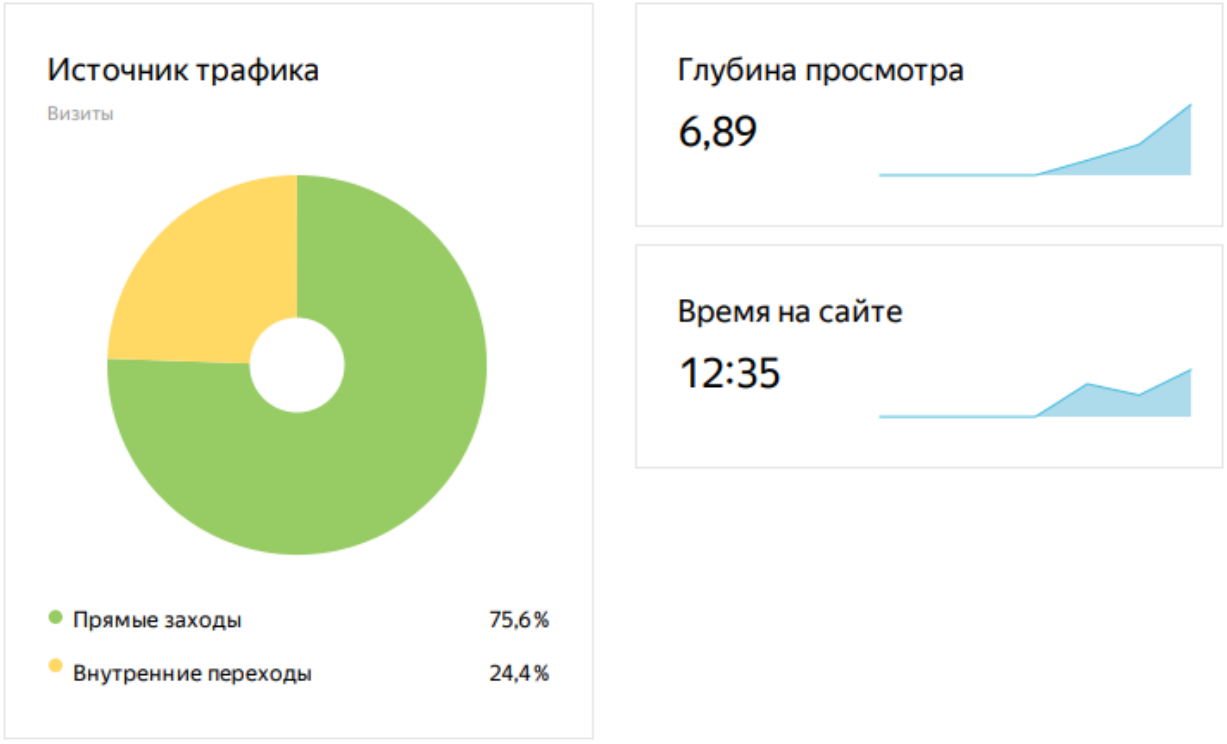

Рисунок 32 – Графики источника трафика, глубины просмотра и времени на сайте в системе аналитики

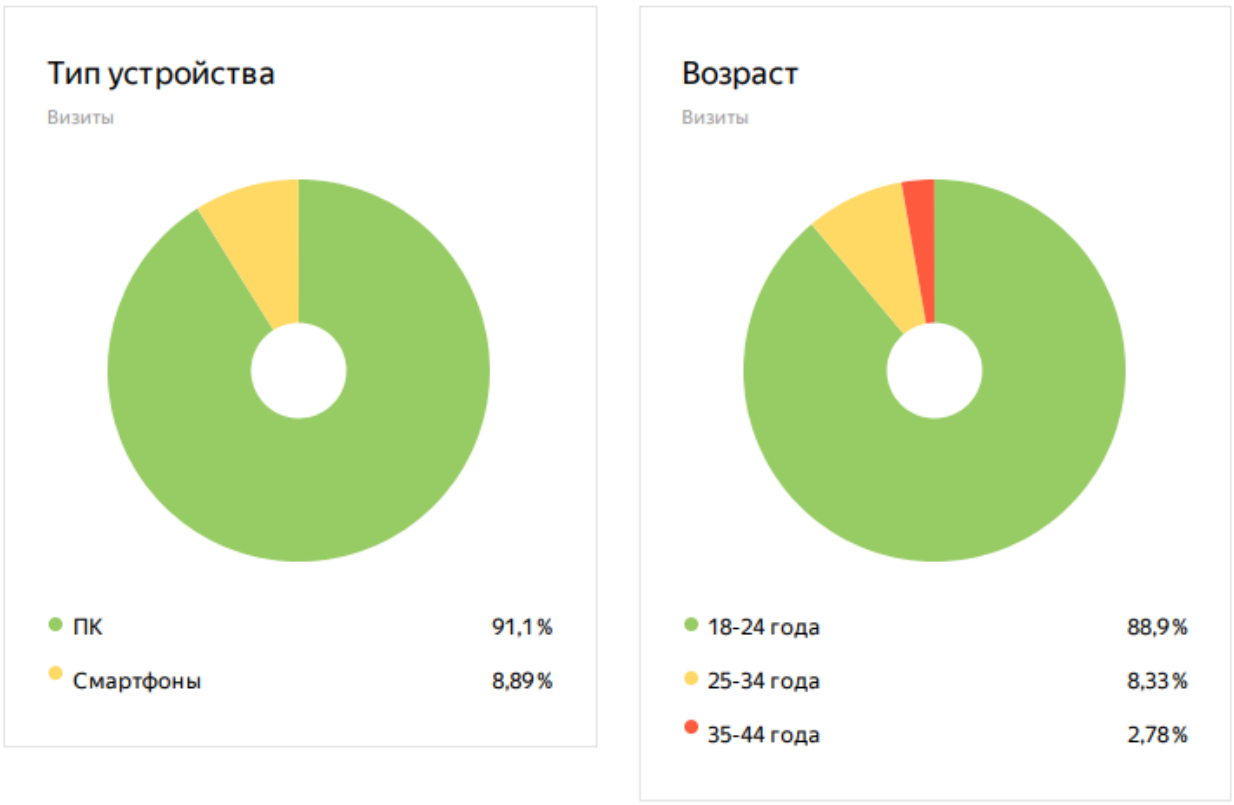

Рисунок 33 – Графики типа устройства и возраста в системе аналитики

Подключены три конверсионные воронки, описанные в пункте 3.1.

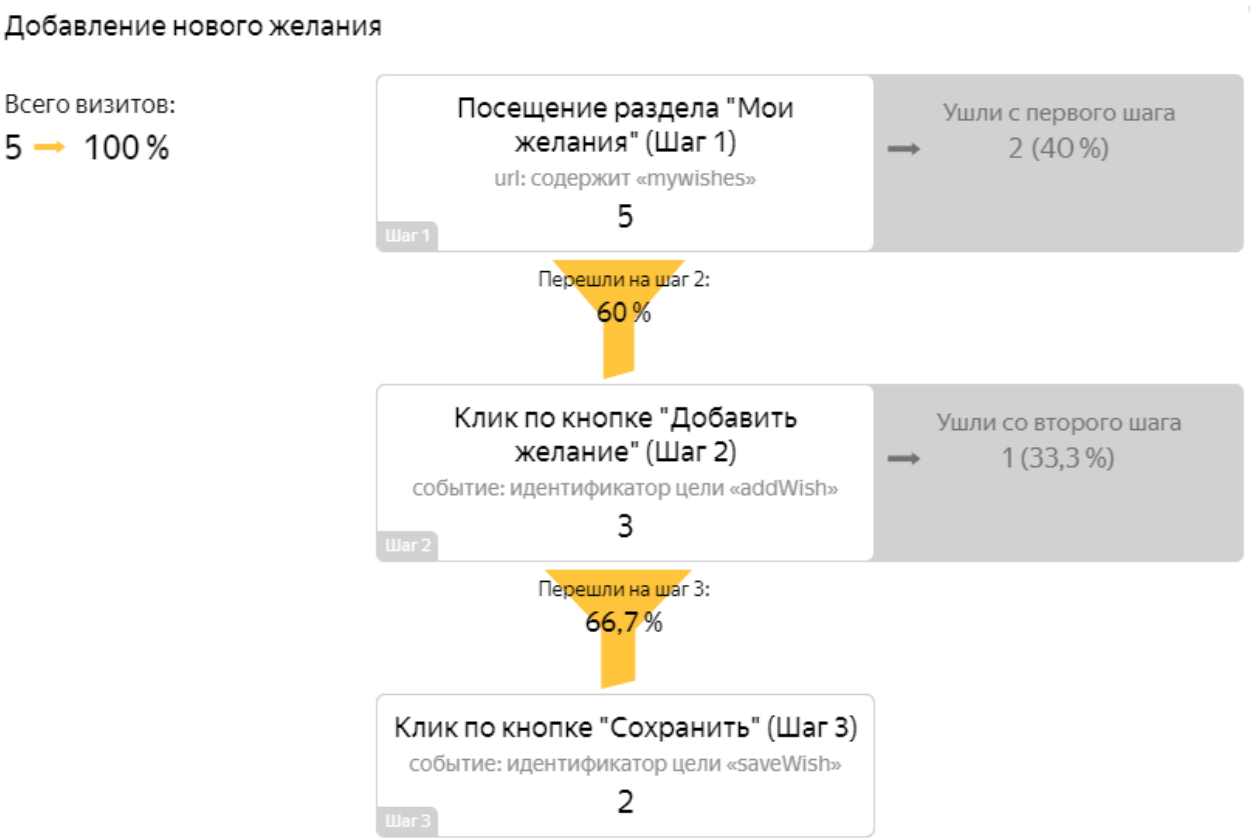

Рисунок 34 – Конверсионная воронка №1. Добавление нового желания

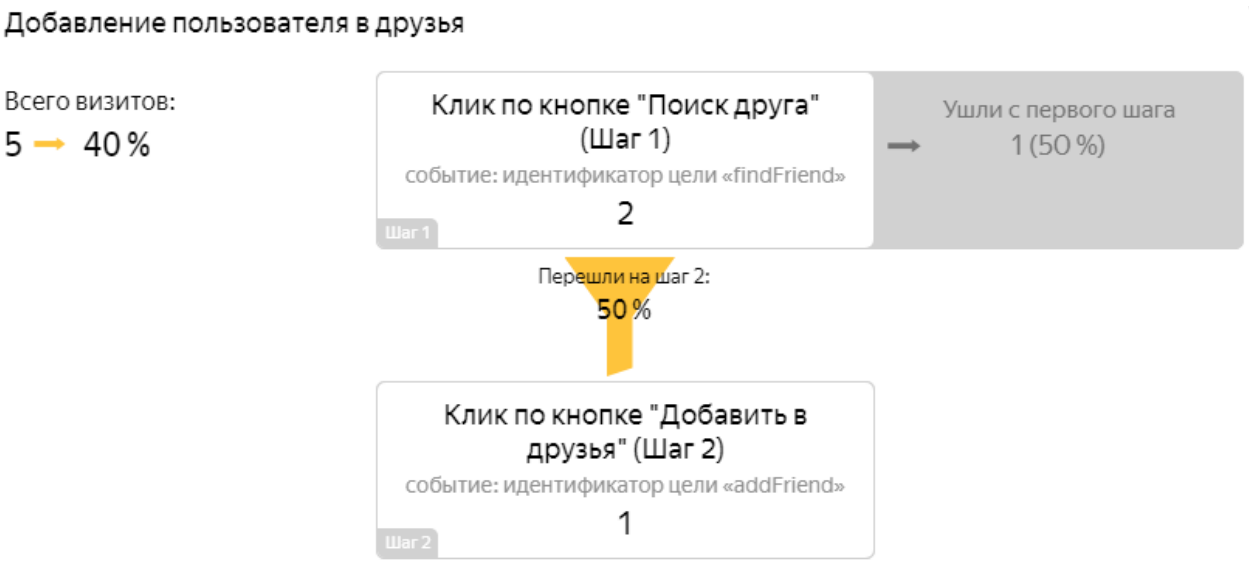

Рисунок 35 – Конверсионная воронка №2. Добавление пользователя в друзья

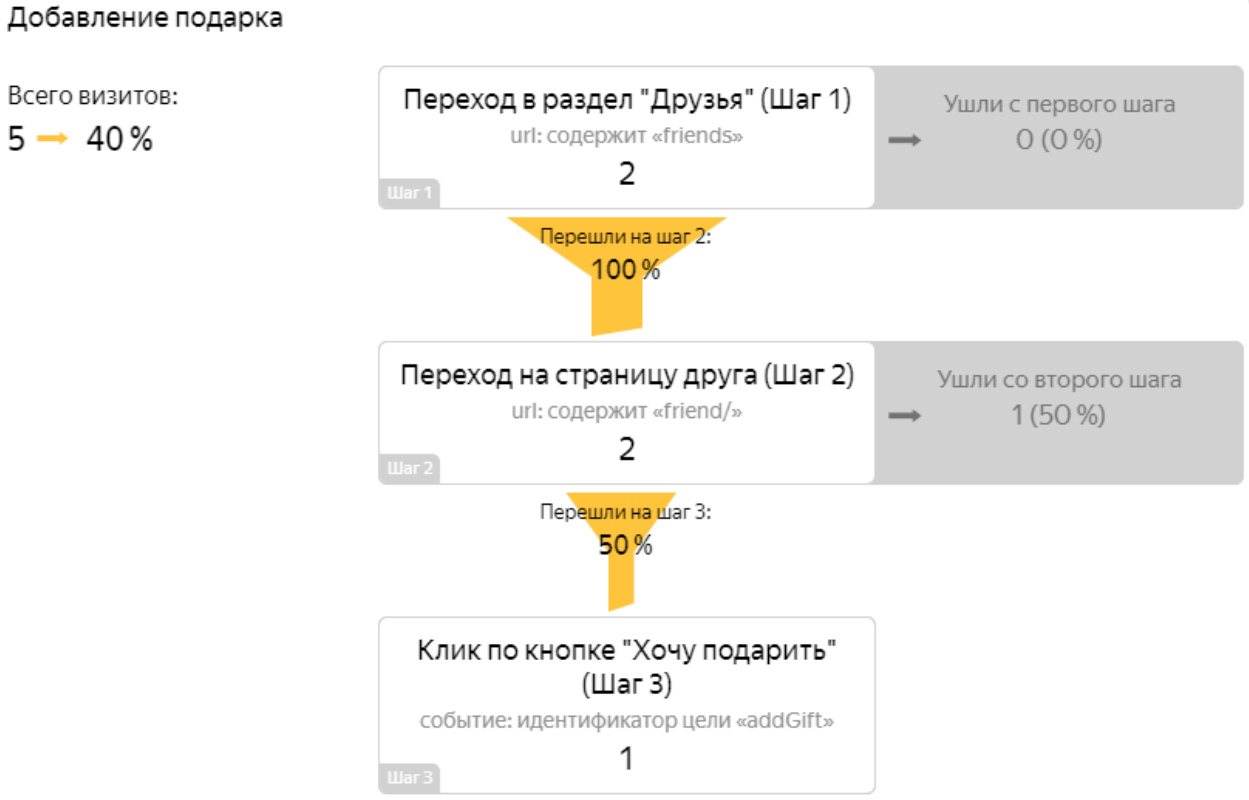

Рисунок 36 – Конверсионная воронка №3. Добавление подарка

5. Тестирование

Было проведено тестирование веб-приложения GoldenFish, которое было разделенно на 3 этапа:

- Подготовка: анализ исходных документов о проекте (ТЗ) и написание тест-кейсов

- Проведение тестирования: Функциональное тестирование ведется вручную по подготовленным заранее тестовым сценариям

- Отчет: после проведения тестирования все ошибки, найденные в работе приложения, описываются в отчете о тестировании

При проведении тестирования веб-приложения GoldenFish ошибок в работе функционала приложения выявлено не было.

6. Дальнейшее развитие проекта

Конечно, чтобы Система стала востребованной среди пользователей, ее необходимо дорабатывать, улучшать и добавлять новый функционал. Выделим пункты для дальнейшего развития:

- Сделать авторизацию через популярные социальные сети, почтовые сервисы
- Добавить модерацию
- Добавить подтверждение почты
- Добавить возможность изменения и восстановления пароля
- Добавить возможность создания событий (описание события, повод, место, время и т.д.), отправлять приглашение друзьям на эти праздники
- Добавить возможность добавления описания, фотографий, видео к прошедшим событиям, которые будут сохраняться в разделе «Воспоминания»
- Добавить оповещения (о том, что пришла заявка в друзья, что заявка отклонена, напоминания о событиях друзей, оповещение о том, что друг отметил желание как «Хочу подарить» и т.д.)
- Подключить каталоги из магазинов
- Добавить возможность отправления сообщений, создания групповых чатов из списка приглашенных на событие

Это лишь некоторые пункты из возможностей расширения Системы. В будущем этот проект может перерасти в подобие социальной сети с праздничной тематикой.

7. Заключение

В результате работы было реализовано Android приложение, которое позволяет осуществлять учет доходов и расходов, а также имеет визуально информативный интерфейс, который позволяет получать всю необходимую информацию в виде диаграммы с главного экрана.

Были выполнены следующие задачи:

1. Разработана клиентская часть приложения
2. Разработана серверная часть приложения
3. Была создана связь между ними с помощью REST API
4. Разработана база данных
5. Проведено тестирование

Приложение отвечает всем заявленным требованиям.