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 “Goldenfish”*

*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**

1. Введение
2. Постановка задачи
3. Анализ предметной области
4. Реализация
5. Тестирование
6. Дальнейшее развитие проекта
7. Заключение

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.

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).

Анализ предметной области

**3.1. Глоссарий**

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

*Подарок* - осуществление желания другого человека.

**3.2 Анализ существующих решений**

3.2.1. LesterWish

[LesterWish](http://lesterwish.com/)— 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*

3.2.2. MyWishList

[MyWishList](http://mywishlist.ru/)- 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;

- добавление пользователей в друзья;

- возможность оставлять комментарии;

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

Недостатки:

- непрезентабельный интерфейс.



*Рисунок 2. Страница авторизованного пользователя сайта* MyWishList

**3.3. Анализ предметной области**

3.3.1. 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 главную страницу → Авторизоваться → Посетить раздел «Мои желания» → Добавить желание

* + - 1. Воронка добавления пользователя в друзья:

Посетить главную страницу → Авторизоваться → Ввести юзернейм пользователя в поисковой строке → Нажать кнопку «Поиск» → Нажать кнопку «Добавить в друзья»

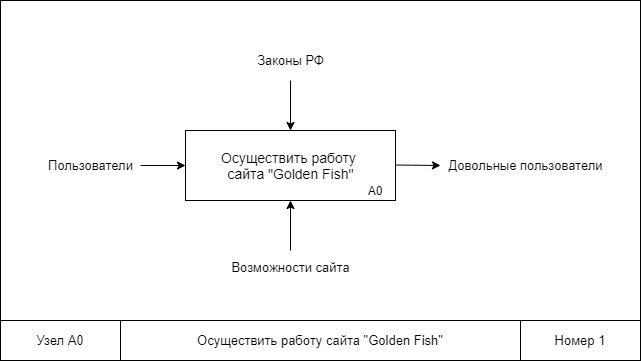
* + - 1. Воронка добавления подарка:

Посетить главную страницу → Авторизоваться → Перейти в раздел «Друзья» → Перейти во вкладку «Мои друзья» → Перейти на страницу друга → Нажать кнопку «Хочу подарить»

3.3.2. IDEF0

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

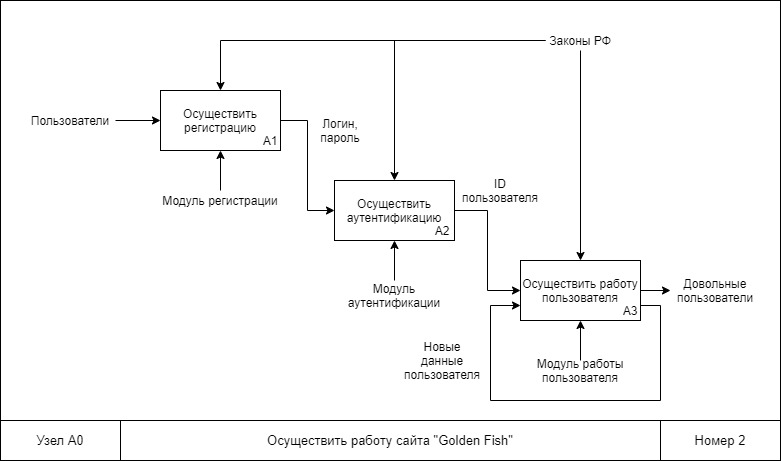
На рисунке 3 представлена контекстная диаграмма разрабатываемого приложения.



*Рисунок 3. Контекстная диаграмма разрабатываемого приложения*

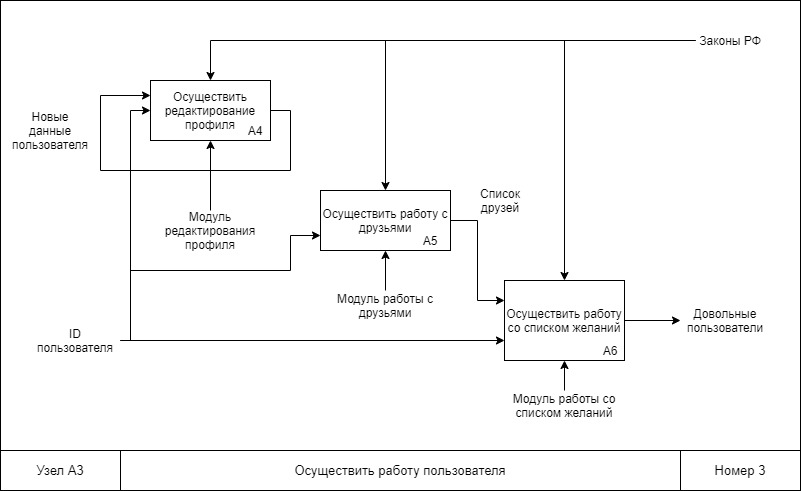
Работу сайта регулируют законы РФ, а именно законы о деятельности сайтов. На вход в Систему поступают пользователи, которые используют возможности сайта и получают от этого удовлетворение. На выходе Система получает довольных пользователей, которые снова захотели бы посетить сайт.

Детальное представление диаграммы верхнего уровня представлено на рисунке 4.



*Рисунок 4. Детальное представление диаграммы верхнего*

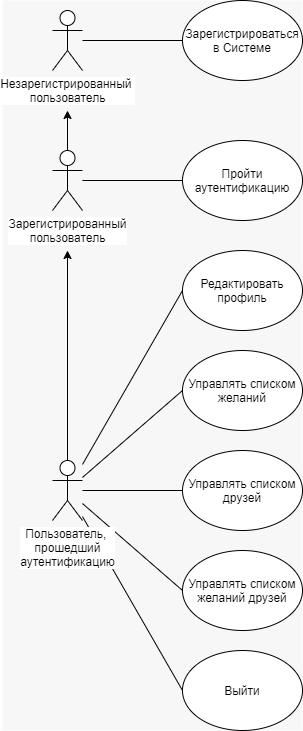
Блок А3 нуждается в более глубокой детализации. Дочерняя диаграмма изображена на рисунке 5.



*Рисунок 5. Диаграмма блока А3*

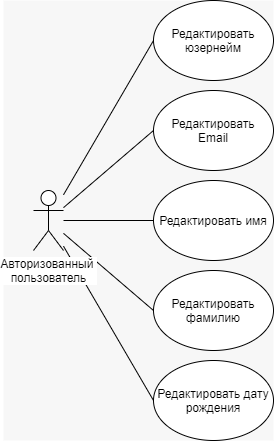
4.2. Диаграммы прецедентов

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*

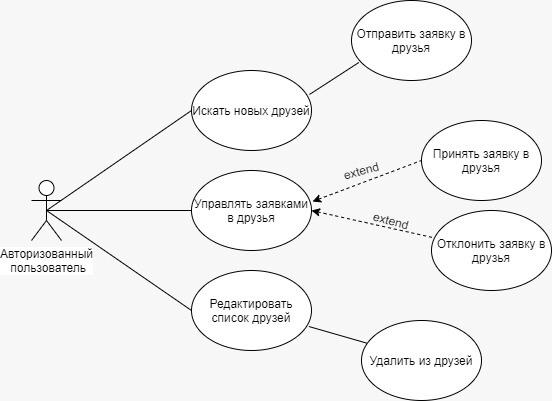
Далее подробнее рассмотрим варианты использования пользователя, прошедшего аутентификацию.



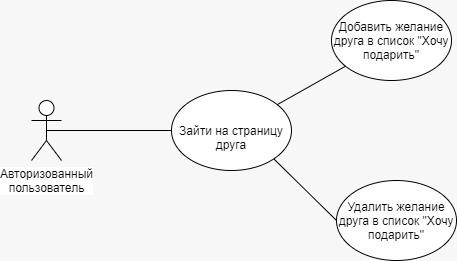
*Рисунок 7 – Диаграмма прецедентов редактирования профиля*



*Рисунок 8 – Диаграмма прецедентов управления списком желаний*



*Рисунок 9 – Диаграмма прецедентов управления списком друзей*



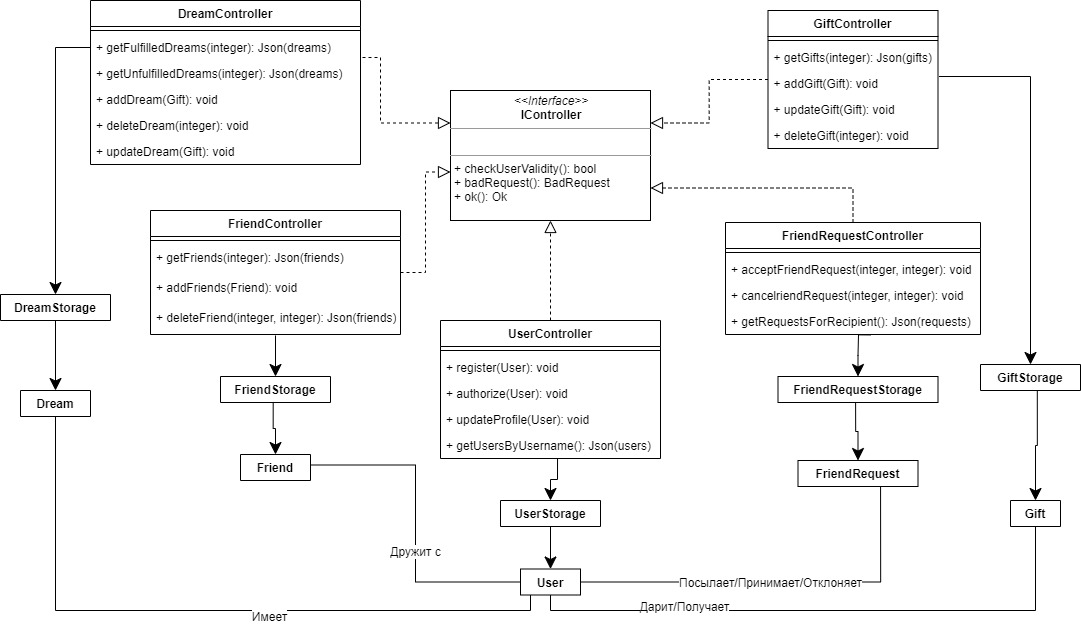
*Рисунок 10 – Диаграмма прецедентов управления списком желаний друзей*

4.3. Диаграмма классов

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

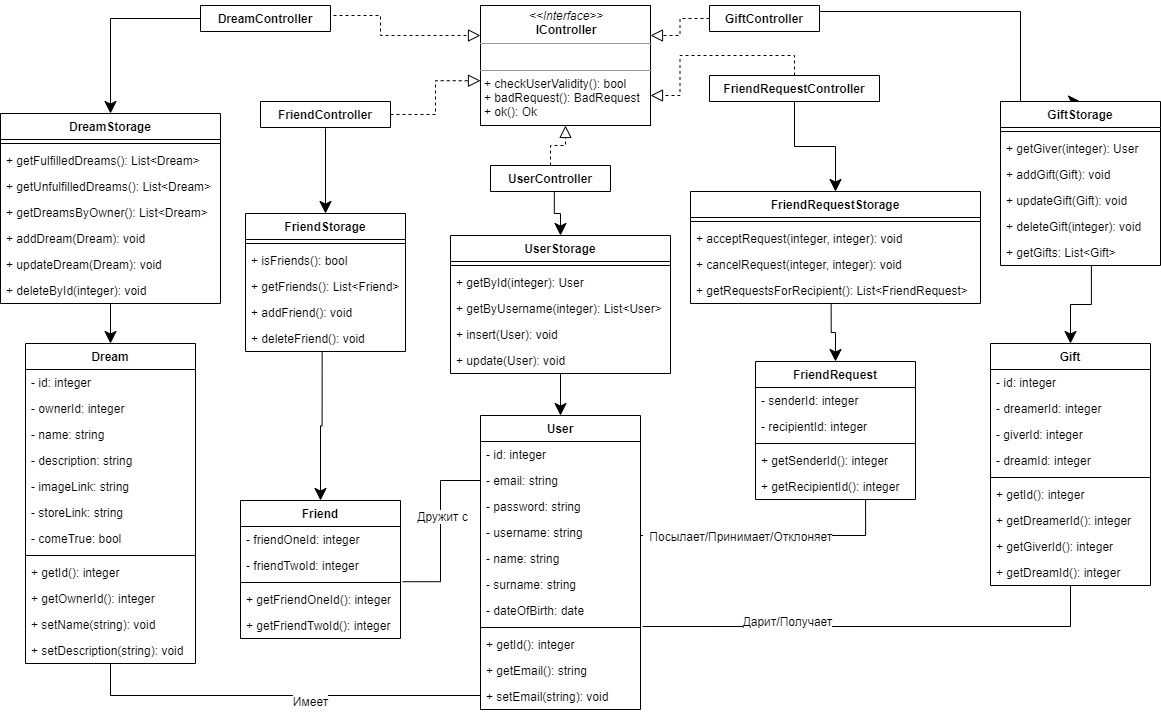
Диаграмма классов разрабатываемой Системы с подробным устройством классов-контроллеров расположена ниже. Стоит отметить, что данный проект строится на шаблоне проектирования MVC. Подробнее об этом написано в пункте «Обоснование архитектуры проекта».

**­­­­**



*Рисунок 11 - Диаграмма классов. Классы-контроллеры*

Та же диаграмма, но с детальным устройством классов, относящихся к модели, находится на рисунке 6.

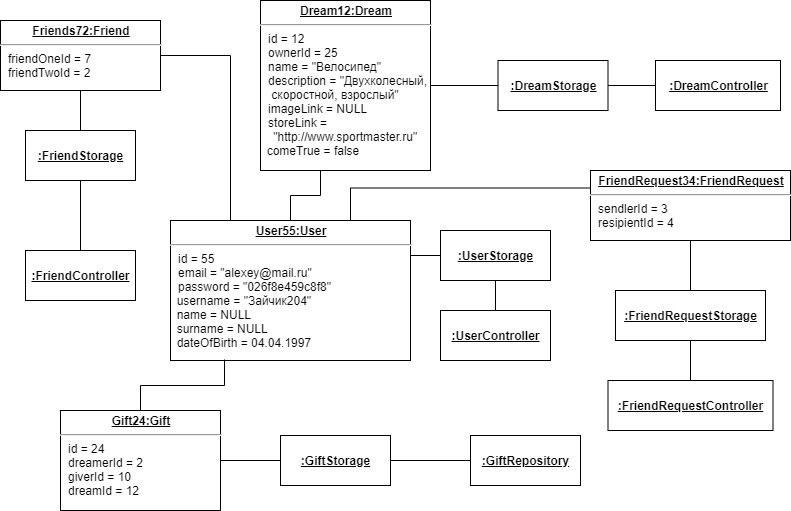


*Рисунок 12 - Диаграмма классов. Классы, относящиеся к модели*

4.4. Диаграмма объектов

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

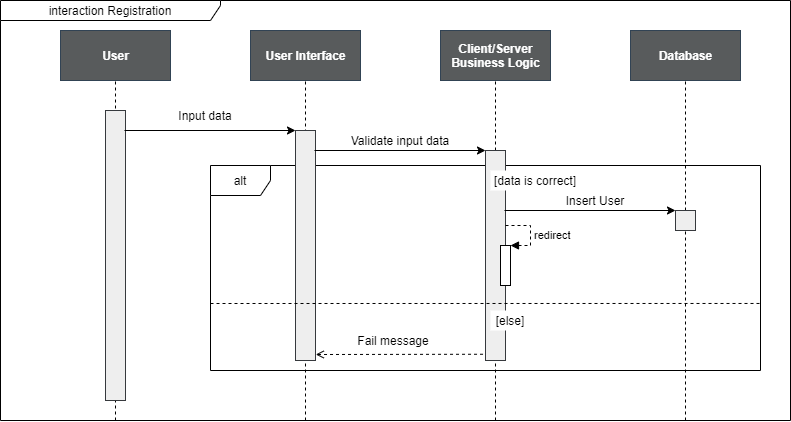
В разрабатываемой системе есть много классов, реализующих только методы. Они соответственно представлены прямоугольниками без атрибутов.



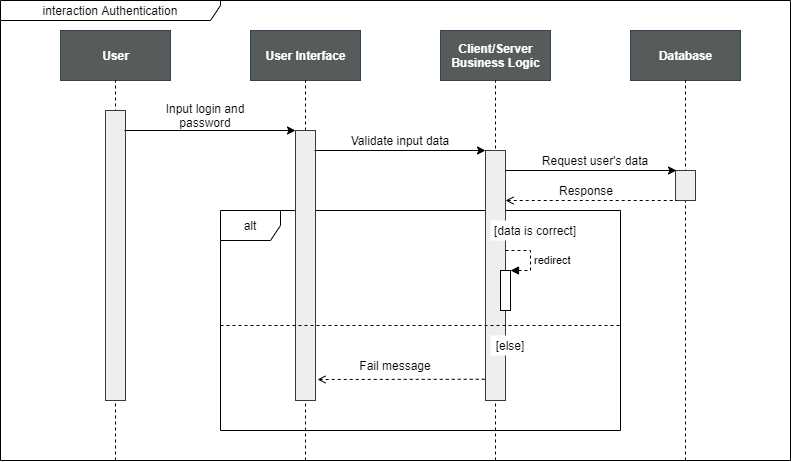
*Рисунок 13 - Диаграмма объектов*

4.5. Диаграммы последовательности

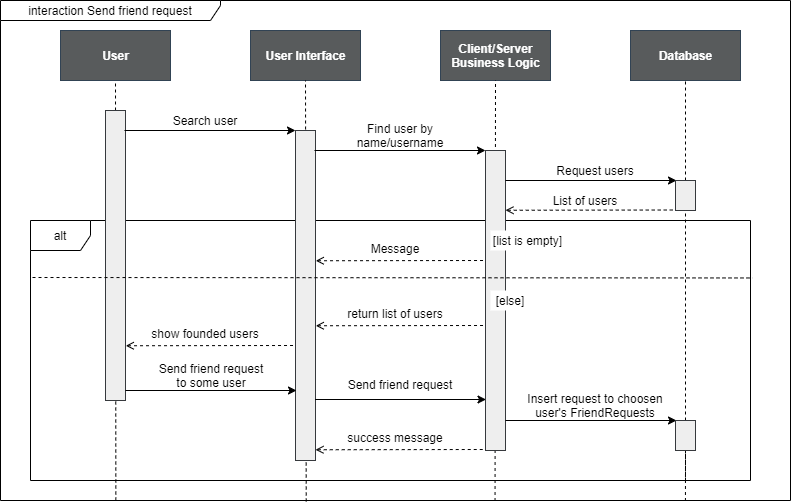
Для того чтобы показать, как различные объекты задействованы в приложении и как они взаимодействуют друг с другом, использованы диаграммы взаимодействия. В частности – диаграммы последовательности, которые акцентируют внимание на то, как тот или иной объект изменяется во времени, какие действия совершены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


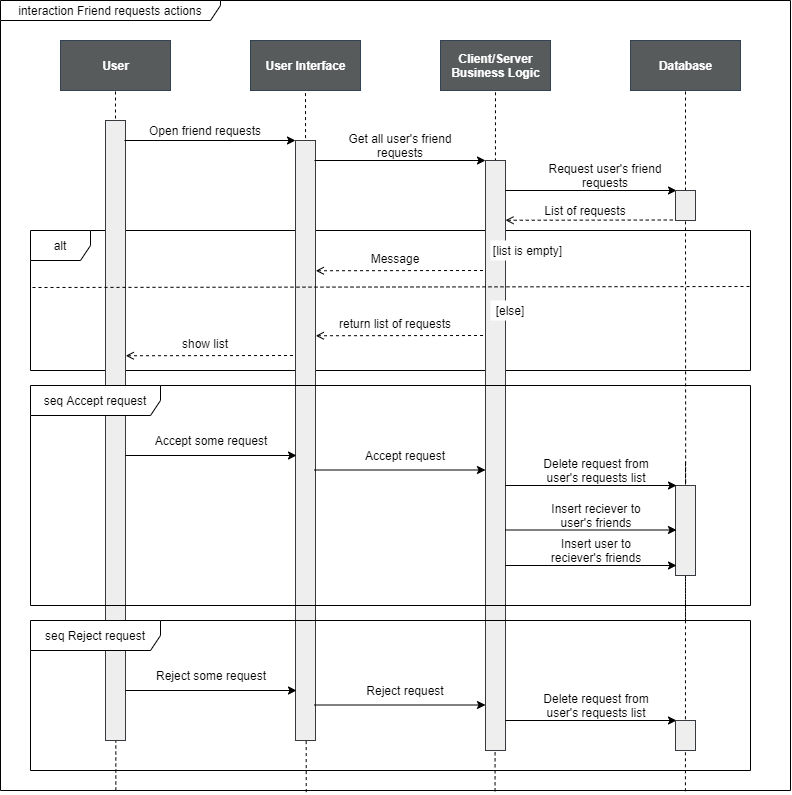
*Рисунок 14 - Диаграмма последовательности. Регистрация*

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.

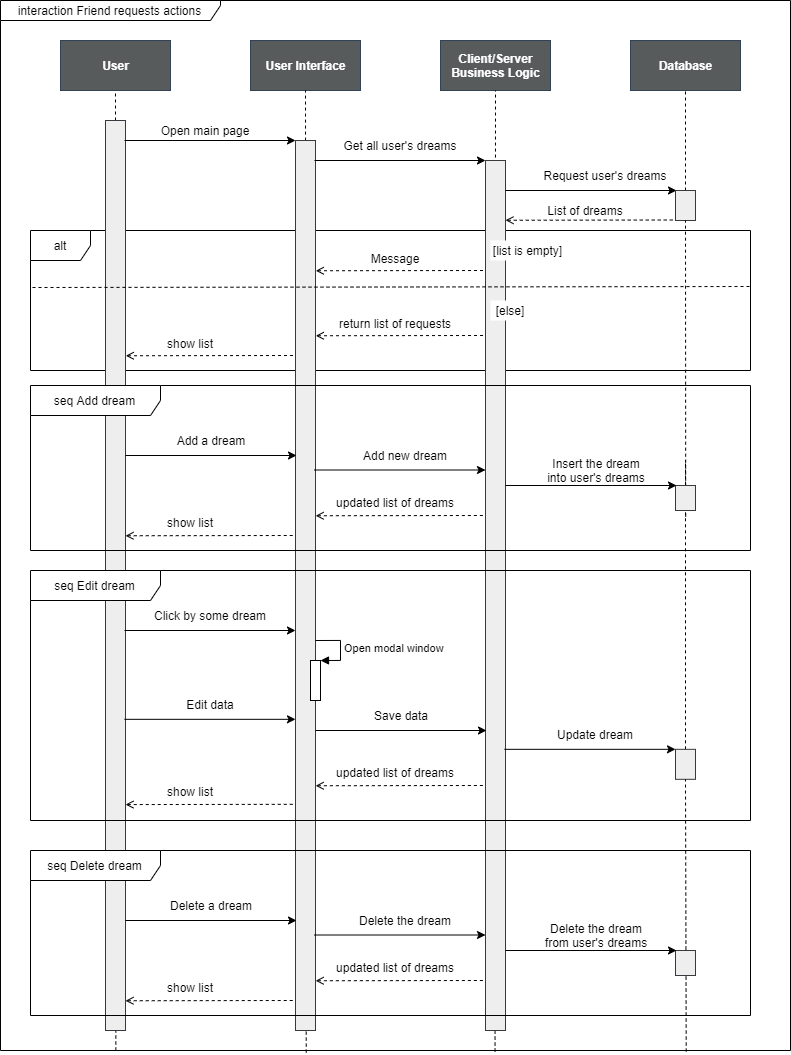
*Рисунок 15 - Диаграмма последовательности. Аутентификация*

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 FriendRequest соответствующего пользователя появляется новая запись. 

*Рисунок 16 - 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 17 - Диаграмма последовательности. Действия с заявками «в друзья»*

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

*Рисунок 18 - Диаграмма последовательности. Действия с желаниями*

4.6. 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.

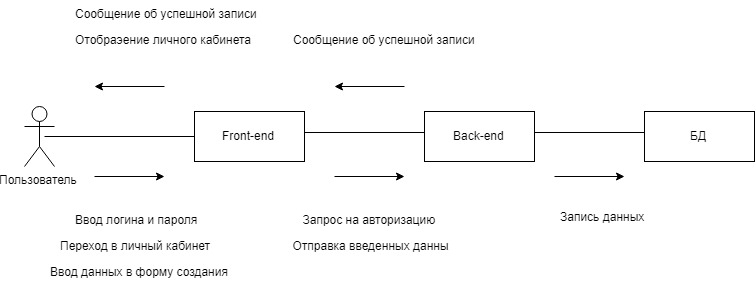


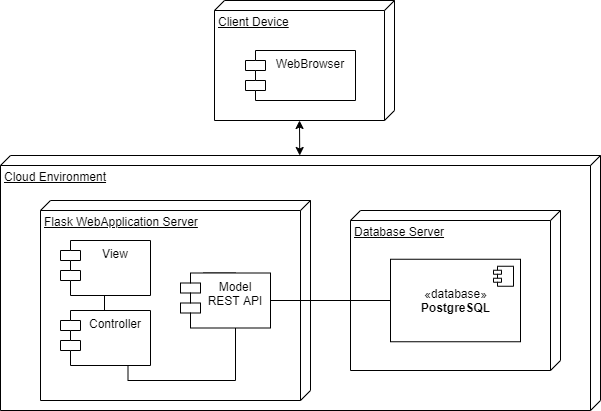
Рисунок 19 – Диаграмма коммуникации

4.7. Диаграмма состояний

4.8. Диаграмма активностей

4.9. 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 REST API. Все в совокупности планируется развернуть на одном из облачных серверов.



*Рис. Диаграмма развертывания*

4.10. 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.



*Рис. 10. Схема шаблона проектирования MVC*

К плюсам можно отнести:

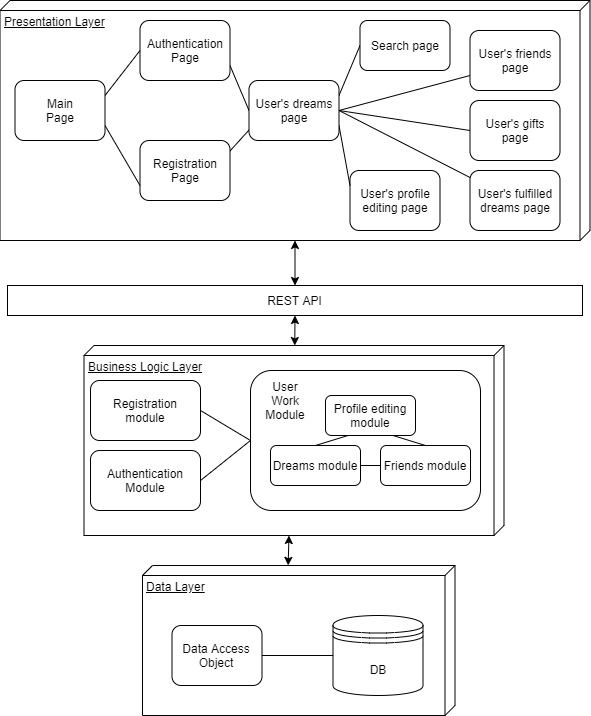
* Структурированный, гибкий код
* Хорошая расширяемость
* Легкость в отладке и тестировании

Минусами данной концепции являются:

* Увеличение объема кода
* Необходимость использования большего количества ресурсов

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

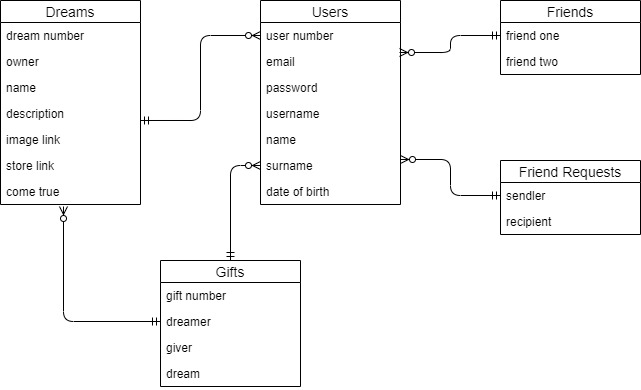
Архитектура разрабатываемого приложения схематично представлена на рисунке ниже.



*Рисунок – Архитектура разрабатываемого приложения*

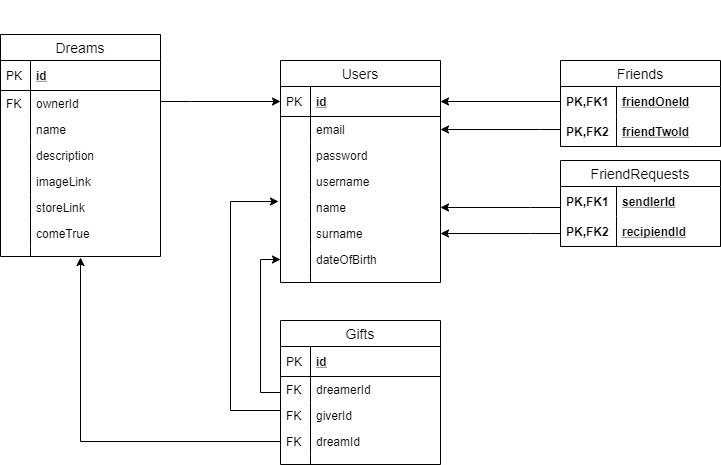
4.11. ER-диаграмма и схема базы данных

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



*Рис. .ER-диаграмма приложения*

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



*Рис. .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 (необязательно поле)
* Фамилия (необязательное поле)
* Дата рождения (необязательное поле)

1. 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.

1. 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.

1. 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

1. Gifts

* Уникальный идентификатор подарка
* Идентификатор пользователя, мечтающего о данном подарке (ссылается на уникальный идентификатор в таблице Users)
* Идентификатор пользователя, планирующего подарить данный подарок (ссылается на уникальный идентификатор в таблице Users)
* Идентификатор мечты (ссылается на уникальный идентификатор в таблице Dream)