`

`

ВВЕДЕНИЕ

В мире, где каждый миг становится ценным, связь и обмен информацией становятся сердцем нашей повседневности. Мы представляем новое мобильное приложение - социальную сеть, где общение преображается в нечто более увлекательное и насыщенное. Наше приложение не просто соединяет людей, оно создает целый мир возможностей для выражения идеи, обмена новостями и оценки того, что нас окружает.

Цель нашей социальной платформы - вдохновлять вас на связь и общение в формате, который соответствует вашему образу жизни. Мы предлагаем простой и удобный способ делиться моментами, мыслями и достижениями с теми, кто вам дорог. Вы можете не только выкладывать свои новости, но и находить новых друзей, оценивать и поддерживать их идеи.

Наше приложение стремится быть вашим персональным пространством для творчества, общения и вдохновения. Мы создали инновационную среду, где вы можете выражаться, находить интересные контент и получать заслуженные признания за свои успехи. Добро пожаловать в нашу социальную сеть, где каждый ваш пост - это шаг к обогащению вашей жизни и жизни тех, кто вас окружает.

`

1. ОБЩАЯ ЧАСТЬ

1.1. Цель разработки

Целью разработки мобильного приложения социальной сети "DelWiz" является создание инновационной и привлекательной платформы для упрощения общения, совместного творчества и обмена идеями между пользователями. Наша задача - предоставить удобный и простой в использовании интерфейс, способствующий свободному общению и мгновенному обмену контентом, таким как фотографии, текстовые сообщения и другие формы выражения.

1.2. Средства разработки

Для проектирования, разработки и тестирования программного обеспечения, а также оформления документации по проекту использовались программные средства, представленные в таблице 1.

Таблица 1 - Программные средства разработки

№	Тип средства	Название средства	Назначений
1	2	3	4
1	Операционная система	Microsoft Windows 11 PRO 21H2	Организация взаимодействия программ и пользователя
2	Среда разработки мобильного приложения	Intellij IDEA 2023.2.1	Разработка приложения для телефонов на базе операционной системы Android и IOS.
3	Построение схем БД	SqlDBM	Построение логической и физической модели базы данных
4	СУБД	Microsoft SQL Server	Создание и манипуляция базой данных, обеспечение безопасности, надежности хранения и целостности данных, предоставления средств администрирования БД.
5	Интерфейс взаимодействия	Mircosoft SQL Server Management Studio 18	Для взаимодействия СУБД
6	Хостинг		Хранение базы данных на удаленном сервере.

В качестве средств вычислительной техники при разработке ПО использовался ноутбук Asus TUF Gaming F15. Характеристики

`

представлены в таблице 2. В качестве средства для тестирования программы использовался телефон Oneplus 9RT 5G

Таблица 2 - Технические характеристики устройств используемых при разработке ПО

№	Тип средства	Название средства
1	2	3
Ноутбук Asus TUF Gaming F15		
1	Размер экрана:	15,6
2	Разрешение экрана:	1920x1080
3	Линейка процессора:	Intel i5 11400f
4	Количество ядер процессора:	6
5	Оперативная память:	16 ГБ
6	Тип видеокарты:	Дискретная
7	Видеокарта:	RTX 3050
8	Конфигурация накопителей:	SSD
9	Общий объем всех накопителей:	2 ТБ
Телефон Oneplus 9RT 5G		
1	Размер экрана:	6,2
2	Разрешение экрана:	2400x1080
3	Линейка процессора:	Snapdragon 888 Gen 1
4	Количество ядер процессора:	8
5	Оперативная память:	8 ГБ
6	Общий объем всех накопителей:	128 ГБ

`

2. СПЕЦИАЛЬНАЯ ЧАСТЬ

2.1. Постановка задачи

Разработать социальную сеть под названием "DelWiz", представляющую мобильное приложение для Android, способное обеспечить пользователей возможностью создавать и делиться контентом, общаться и оценивать публикации других пользователей. Данное приложение будет поддерживать создание и публикацию различных форматов контента, таких как изображения, видео и текстовые сообщения. Также предусмотрены функции подключения к другим пользователям, оценки и комментирования контента.

2.1.1. Входные данные

Входные данные представлены в следующем виде:

- Регистрация пользователя (Фамилия, Имя, Пароль, Повтор пароля, Логин);
- Авторизация пользователя (Логин, Пароль);
- Поиск пользователя (Логин);
- Написание, изменение сообщения (Текст сообщения);
- Оценка новостей (Оценка);
- Комментирование новостей (Комментарий);
- Загрузка, обновление, редактирование фотографий (Фотография);
- Редактирование профиля (Имя, Фамилия, Возраст, Логин).

2.1.2. Выходные данные

Выходные данные представлены в следующем виде:

- Зарегистрированные пользователи (Фамилия, Имя, Логин)
- Авторизованные пользователи (Логин)
- Информация о пользователе (Имя, Фамилия, Возраст, Логин)
- Сообщения (Текст сообщения)
- Оценки новостей (Оценка)

`

- Комментарии к новостям (Комментарий)
- Фотографии (Фотография)

2.1.3. Подробные требования к проекту

В данной информационной системе должны быть реализованы такие функции как:

- Авторизация пользователя
- Регистрация пользователя
- Выход из приложения
- Изменение персональных данных
- Загрузка фото (аватарка)
- Создание новости
- Изменение новости
- Удаление новости
- Поиск пользователей по имени и логину клиента
- Фильтрация пользователей (Друзья, Все пользователи, Заявки)
- Создание достижения
- Изменение достижения
- Удаление достижения
- Поиск достижений по названию
- Отправка сообщений
- Изменение сообщений
- Удаление сообщения
- Вывод статистики зарегистрированных пользователей
- Создание и сохранение резервных копий базы данных

2.2. Внешняя спецификация

2.2.1. Описание задачи

Мобильное приложение:

При открытии приложения пользователю должен открываться экран авторизации, если пользователь был зарегистрирован ранее, он

`

может войти под своими данными. Если же пользователя нет в системе, он должен пройти по кнопке «регистрация», после чего откроется экран регистрации. После регистрации пользователя перенесет на окно «авторизации» для дальнейшей авторизации. Если ранее пользователь авторизовался в приложении, его автоматически при запуске приложения будет переносить на экран своего личного профиля.

В нижней части экрана должны быть расположены кнопки, для перемещения по вкладкам приложения (профиль, пользователи, мессенджер, новости).

Приложения для компьютера:

При открытии приложения пользователю должно быть открыто окно ASP.NET.

В ходе разработки был проведен анализ предметной области, чтобы понять, как организация работает. В результате были выявлены процессы, которые можно автоматизировать. Они представлены на рисунках 3-6.

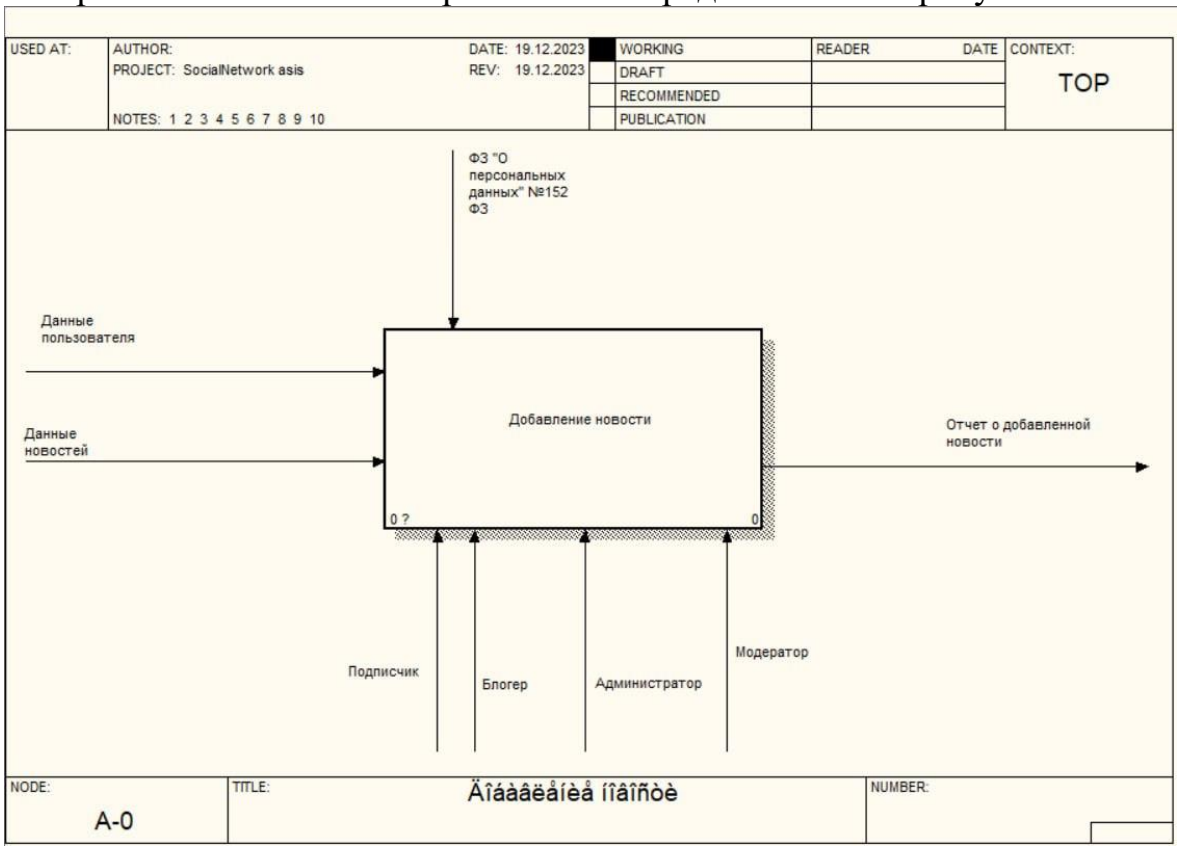


Рисунок 1 - Диаграмма бизнес процессов уровень A0 (AS-IS)

`

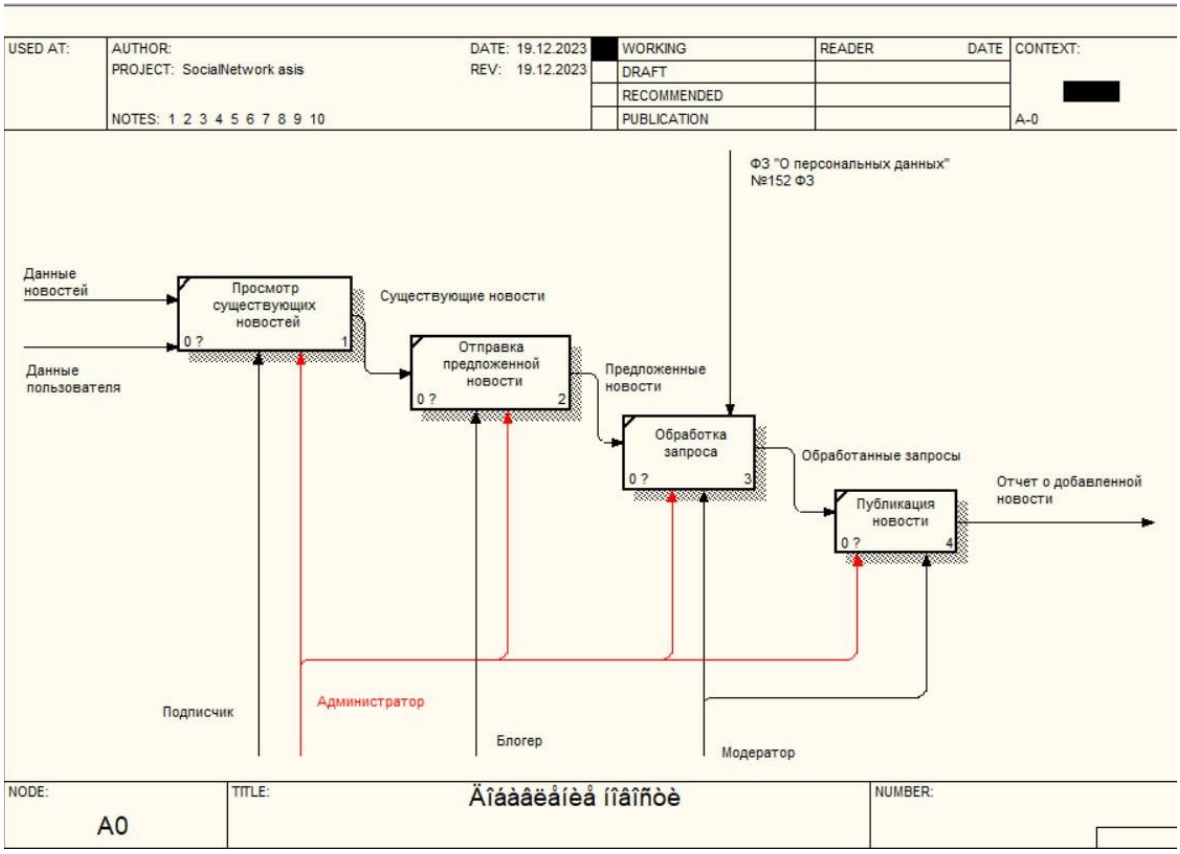


Рисунок 2 – Диаграмма уровня A1(AS-IS).

Для автоматизации был выбран процесс публикации фотографий. Это позволит значительно упростить и ускорить процесс публикации фотографий.

`

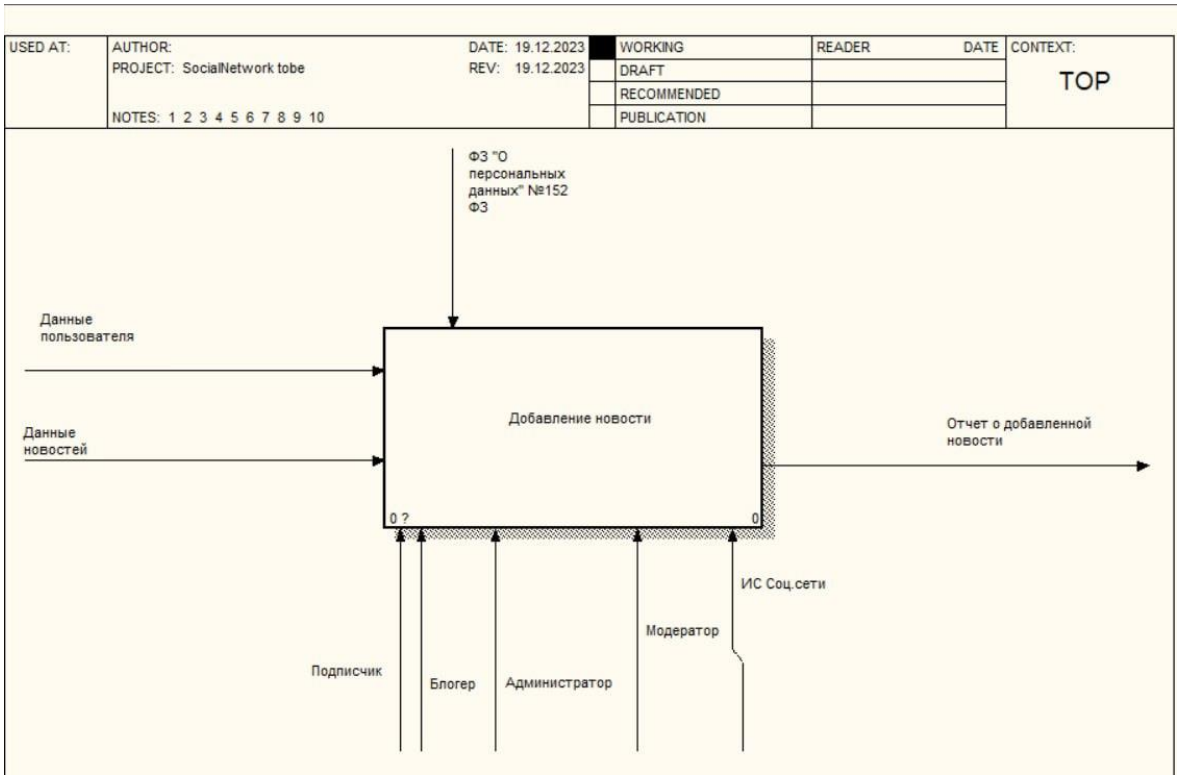


Рисунок 3 - Диаграмма бизнес процессов уровень A0(To-Be).

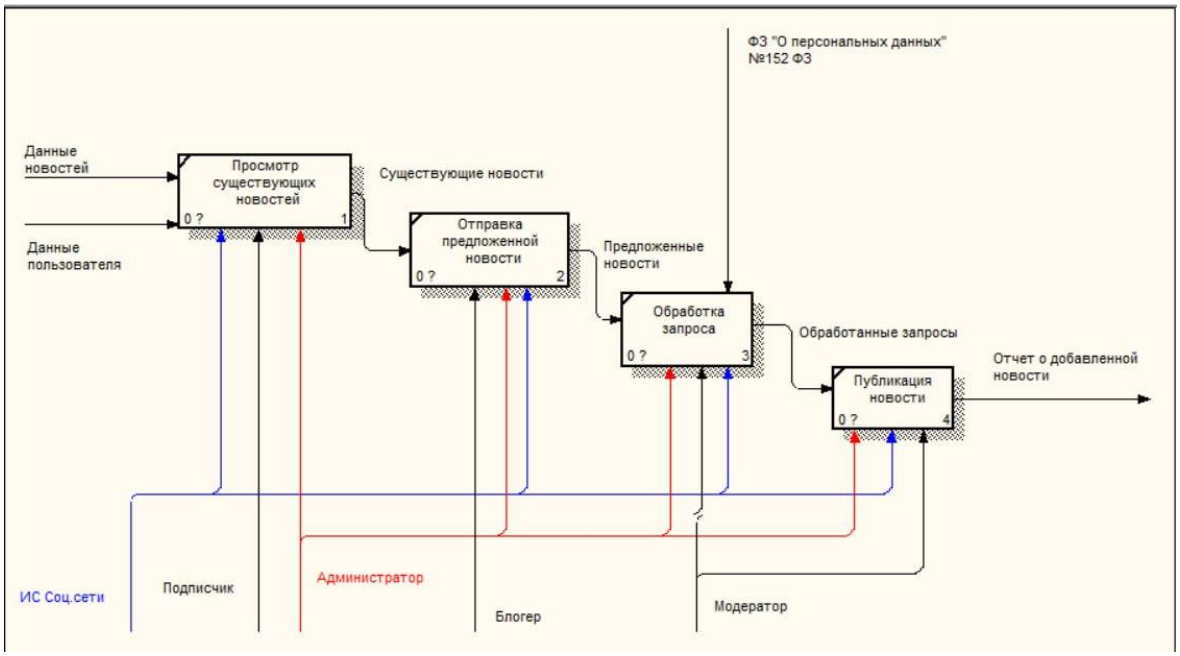


Рисунок 4 - Диаграмма уровня A1(To-Be).

2.2.2. Потоки данных

На рисунках 5-8 представлены диаграммы потоков данных в приложении

`

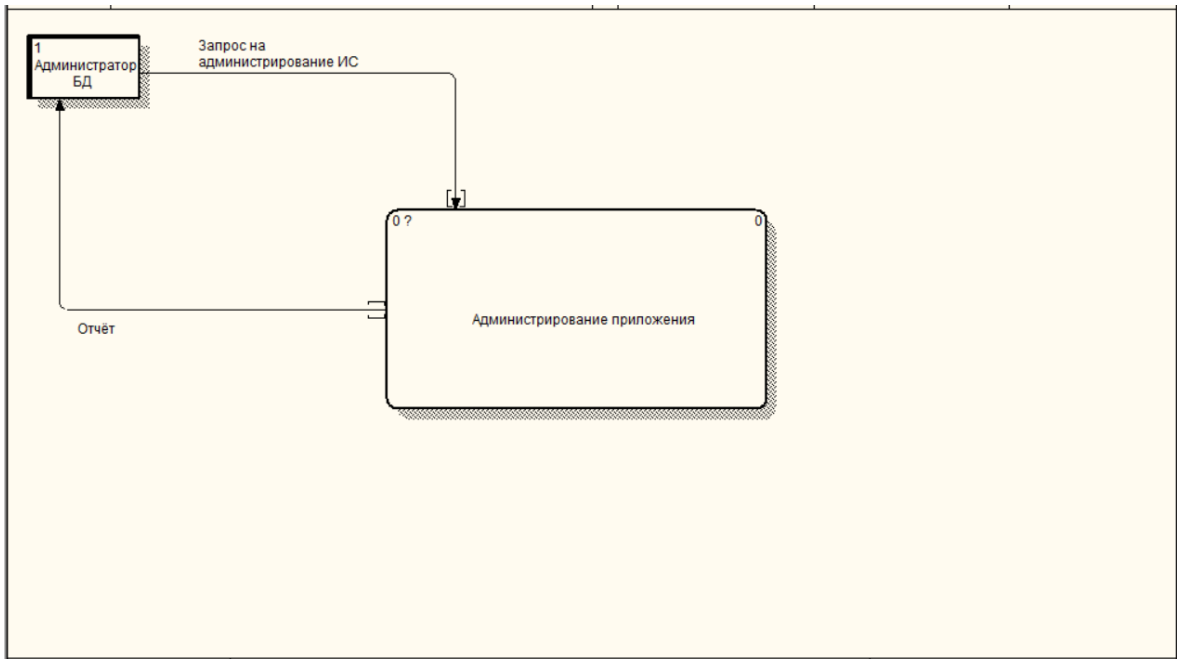


Рисунок 5 - Диаграмма потоков данных

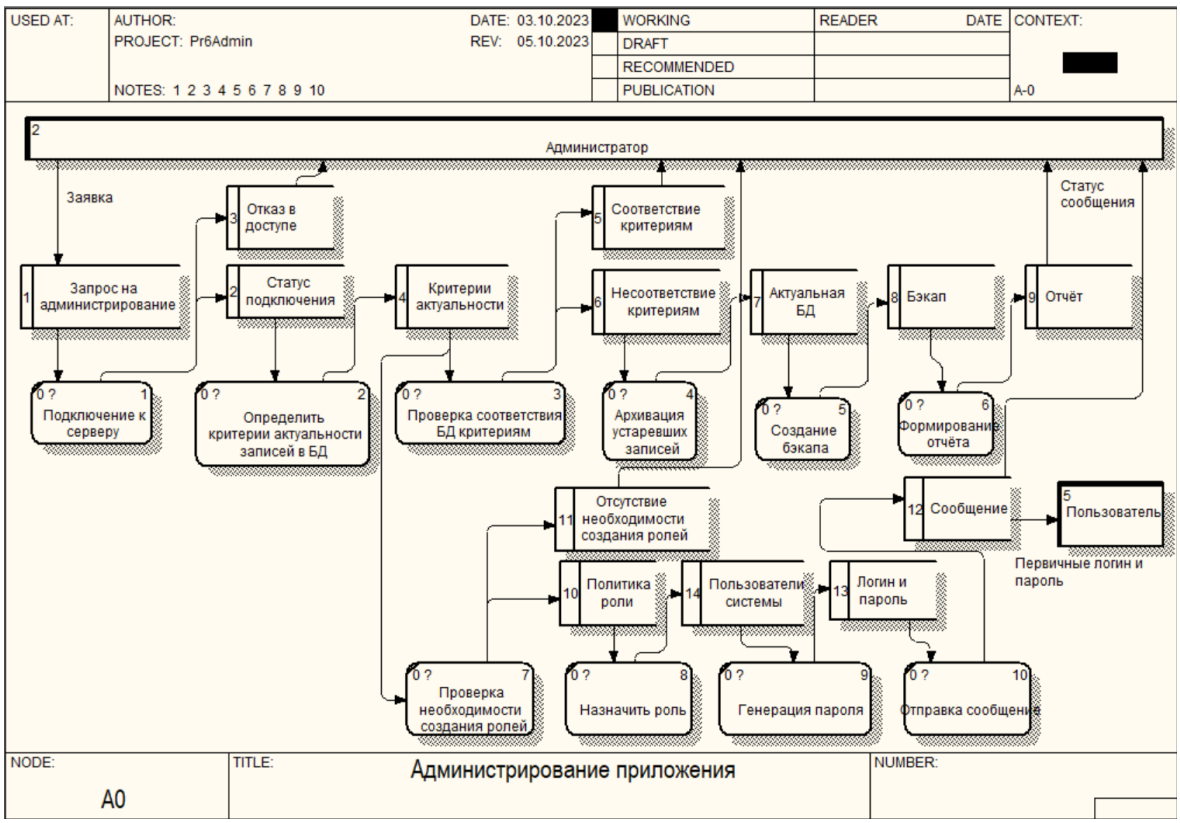


Рисунок 6 - Диаграмма потоков данных

`

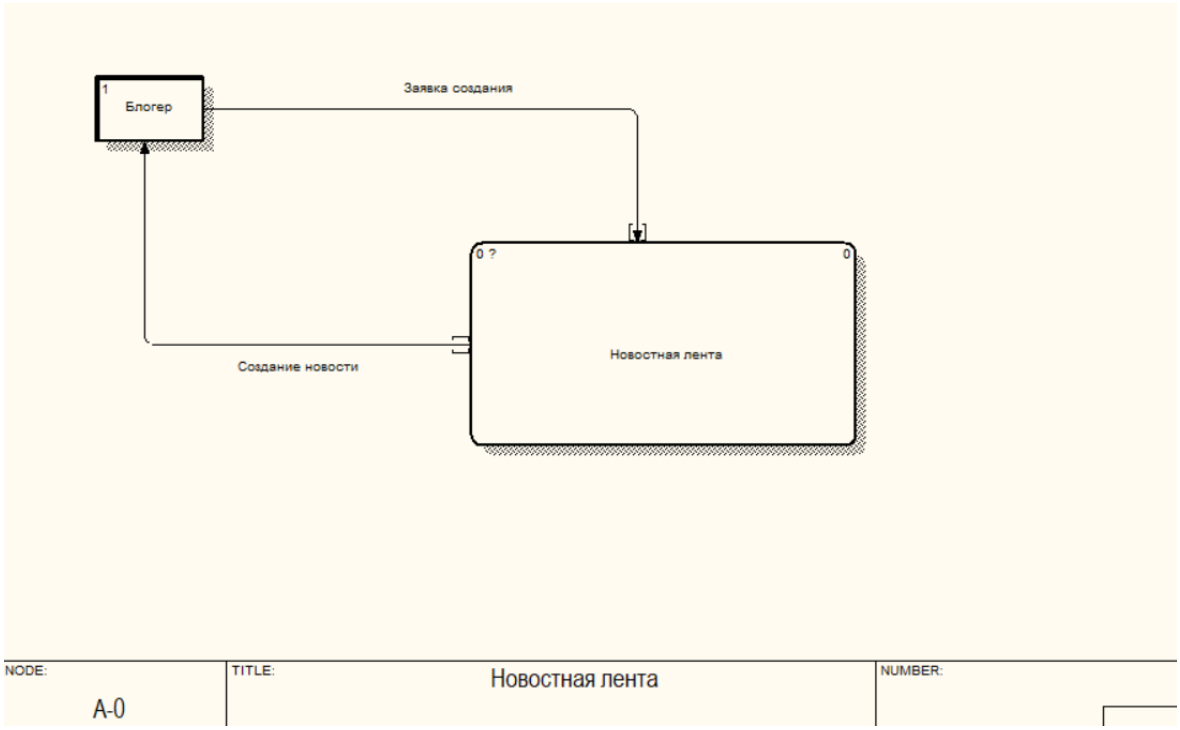


Рисунок 7 - Диаграмма потоков данных

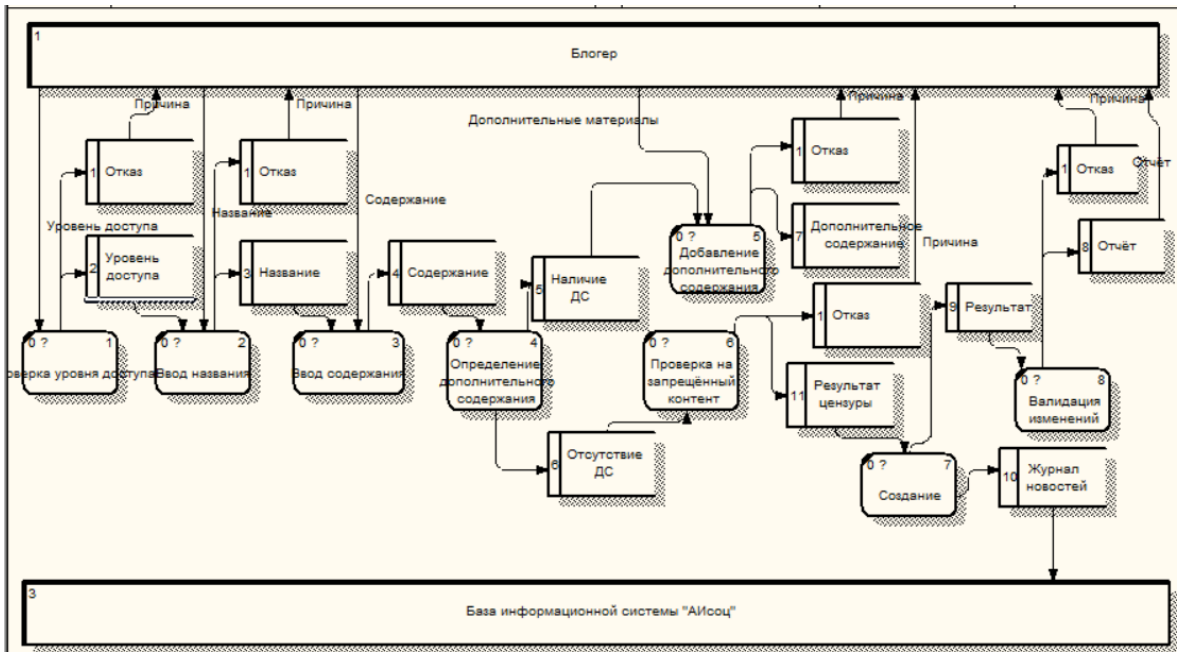


Рисунок 8 - Диаграмма потоков данных

`

2.2.3. Диаграмма классов UML

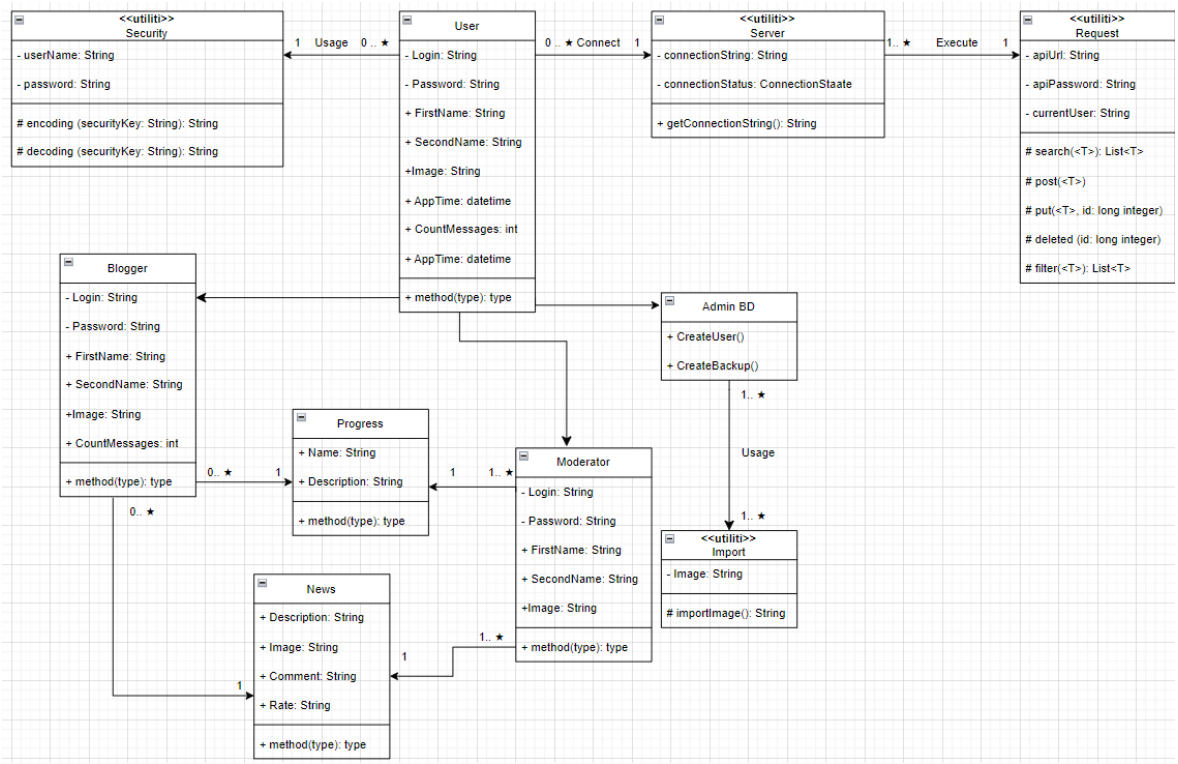


Рисунок 9 - UML

2.2.4. Диаграмма прецедентов

На рисунке 7 представлена диаграмма прецендентов.

`

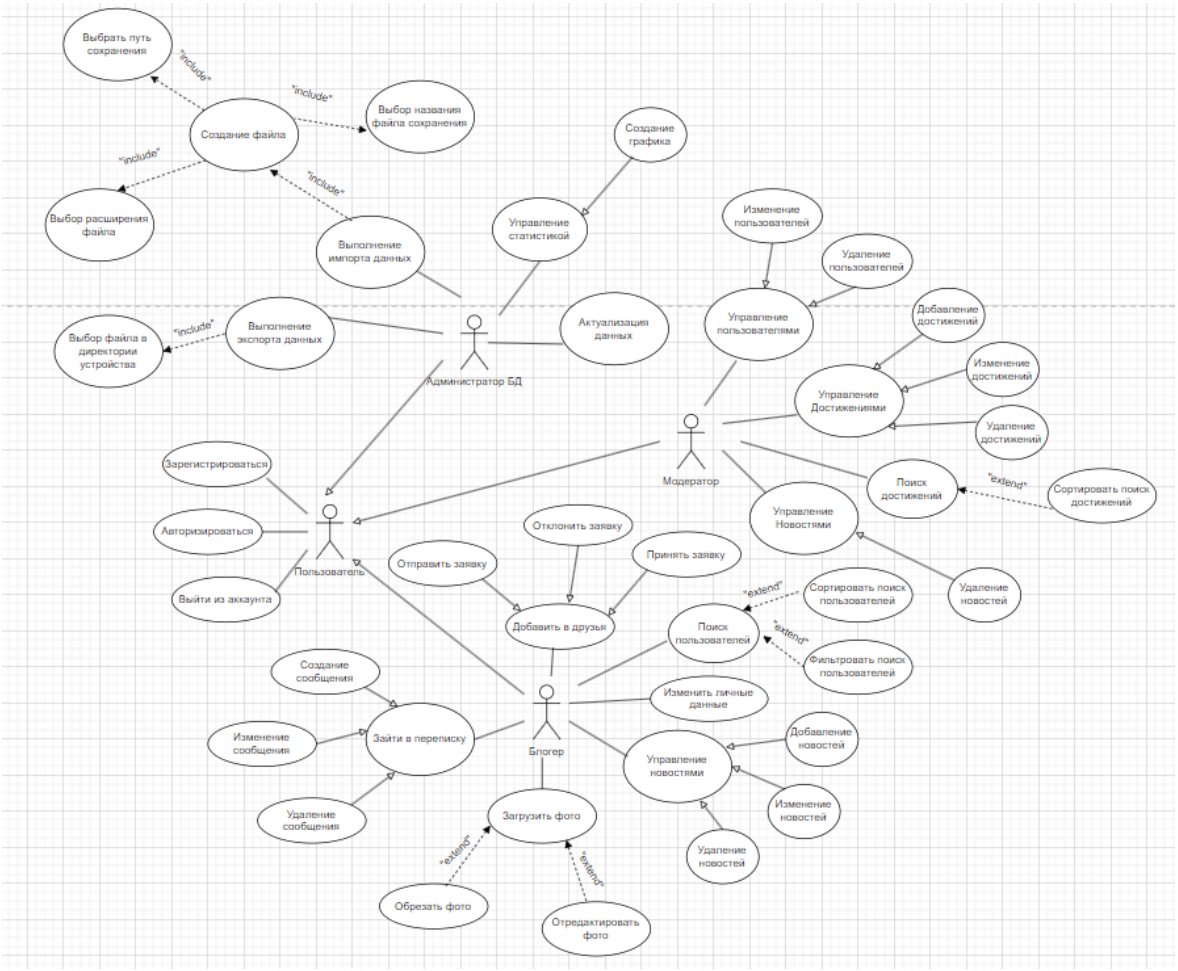


Рисунок 10 - Диаграмма прецендентов.

2.2.5. Входные и выходные данные

В таблицах под номерами 3-4 представлены входные и выходные данные приложения.

Таблица 3 - Входные данные приложения

Поле	Ограничение	Тип	Форма ввода	Описание
1	2	3	4	5
Авторизация				
Логин пользователя	{5,255}	Строка	Поле ввода	Поле ввода для логина пользователя при авторизации
Пароль пользователя	{6,255}	Строка	Поле ввода	Поле ввода для пароля пользователя при авторизации
Регистрация				
Имя	{5,255}, Только буквы	Строка	Поле ввода	Поле ввода для имени

`

Поле	Ограничение	Тип	Форма ввода	Описание
1	2	3	4	5
	латинского алфавита			пользователя при регистрации
Фамилия	{5,255}, Только буквы латинского алфавита	Строка	Поле ввода	Поле ввода для фамилии пользователя при регистрации
Логин пользователя	{5,255}	Строка	Поле ввода	Поле ввода для логина пользователя при авторизации
Пароль	{6,255}, Только буквы латинского алфавита	Строка	Поле ввода	Поле ввода для пароля пользователя при регистрации
Повторный пароль	{6,255}, Только буквы латинского алфавита	Строка	Поле ввода	Поле ввода для повторного пароль пользователя при регистрации
Поиск пользователя				
Логин пользователя	{5,255}	Строка	Поле ввода	Поле ввода для логина пользователя при поиске
Написание, изменение, удаление сообщения				
Текст сообщения	Ограничения не предоставляются	Строка	Поле ввода	Поле ввода для сообщения пользователя
Оценка новостей				
Оценка	Bool	Строка	кнопка	Поле для оценки новости в виде булл переменной
Комментирование новостей				
Комментарий	Ограничения не предоставляются	Строка	Поле ввода	Поле ввода для комментария пользователя
Загрузка, обновление, редактирование фотографий				

`

Поле	Ограничение	Тип	Форма ввода	Описание
1	2	3	4	5
Фотография	Ограничения не предоставляются	Строка	Поле ввода	Поле ввода фотографии

Таблица 4 - Выходные данные приложения

Поле	Ограничение	Тип	Форма ввода	Описание
1	2	3	4	5
Авторизация				
Логин пользователя	{5,255}	Строка	Поле ввода	Поле ввода для логина пользователя при авторизации
Пароль пользователя	{6,255}	Строка	Поле ввода	Поле ввода для пароля пользователя при авторизации
Регистрация				
Имя	{5,255}, Только буквы латинского алфавита	Строка	Поле ввода	Поле ввода для имени пользователя при регистрации
Фамилия	{5,255}, Только буквы латинского алфавита	Строка	Поле ввода	Поле ввода для фамилии пользователя при регистрации
Логин пользователя	{5,255}	Строка	Поле ввода	Поле ввода для логина пользователя при авторизации
Пароль	{6,255}, Только буквы латинского алфавита	Строка	Поле ввода	Поле ввода для пароля пользователя при регистрации
Повторный пароль	{6,255}, Только буквы латинского алфавита	Строка	Поле ввода	Поле ввода для повторного пароль пользователя при регистрации
Поиск пользователя				
Логин пользователя	{5,255}	Строка	Поле ввода	Поле ввода для логина

`

Поле	Ограничение	Тип	Форма ввода	Описание
1	2	3	4	5
				пользователя при поиске
Написание, изменение, удаление сообщения				
Текст сообщения	Ограничения не предоставляются	Строка	Поле ввода	Поле ввода для сообщения пользователя
Оценка новостей				
Оценка	Bool	Строка	кнопка	Поле для оценки новости в виде булл переменной
Комментирование новостей				
Комментарий	Ограничения не предоставляются	Строка	Поле ввода	Поле ввода для комментария пользователя
Загрузка, обновление, редактирование фотографий				
Фотография	Ограничения не предоставляются	Строка	Поле ввода	Поле ввода фотографии

2.2.6. Методы

При разработке приложений использовались среды разработки Intellij IDEA и Visual Studio. Visual studio использовалось для написания API при использовании языка C#. Intellij IDEA использовалось для написания мобильного приложения с использованием языка программирования Dart.

Данные приложения были написаны с использованием основных парадигм объектно-ориентированного программирования, таких как, инкапсуляция (будет использоваться для ограничения доступа одних компонентов программы к другим); наследование (необходимо для повторного использования кода и способствует независимому расширению программного обеспечения и интерфейсов); полиморфизм (для обработки различных типов данных).

Ниже на рисунке 11 приведён пример инкапсуляции.

`

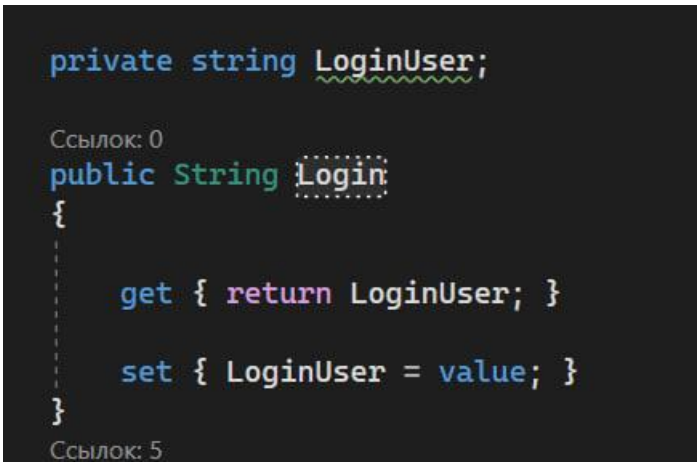


Рисунок 11 - Пример инкапсуляции

Следующий рисунок 12 представляет наследование.

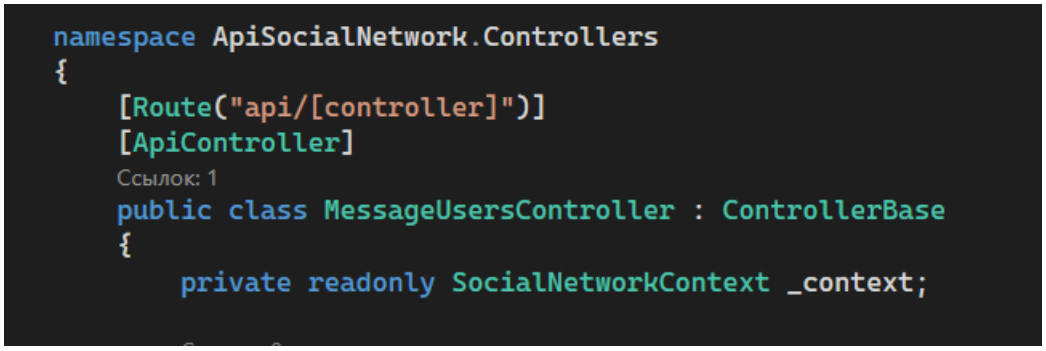


Рисунок 12 - Пример наследования

В создании API использовался паттерн MVC (Model-view-controller). Из представления (View) пользователь осуществляет вызов методов контроллера. Из контроллера осуществляется обновление модели в соответствии с действиями пользователями. Затем данная модель передаётся в представление, в свою очередь представление может осуществлять запрос данных из модели.

На рисунке 13 представлена модель.

`

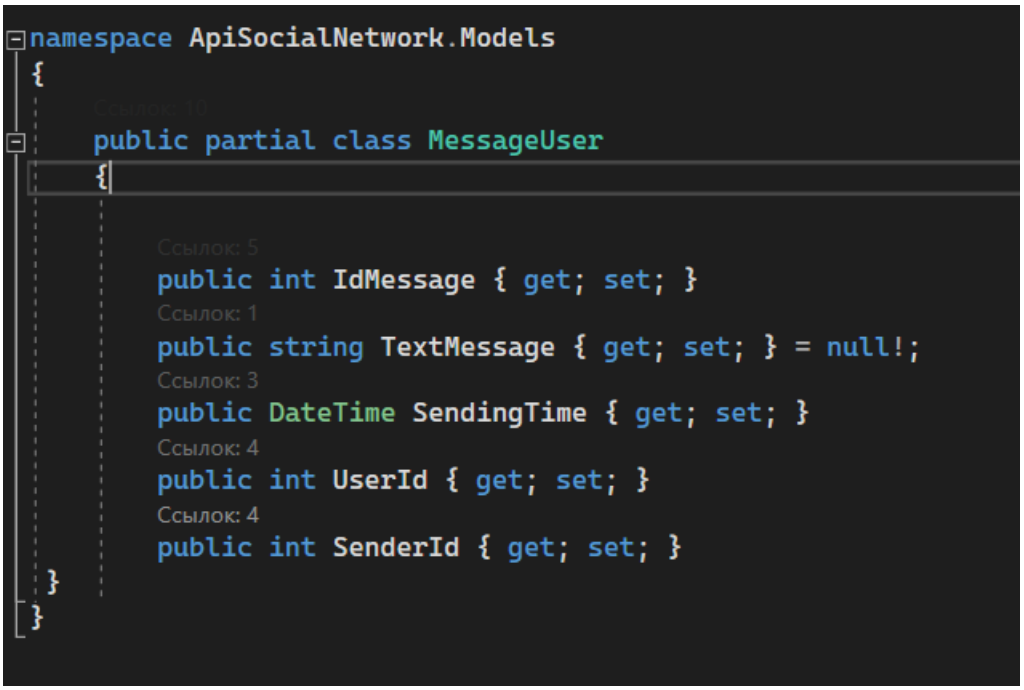


Рисунок 13 - Пример модели из парадигмы MVC

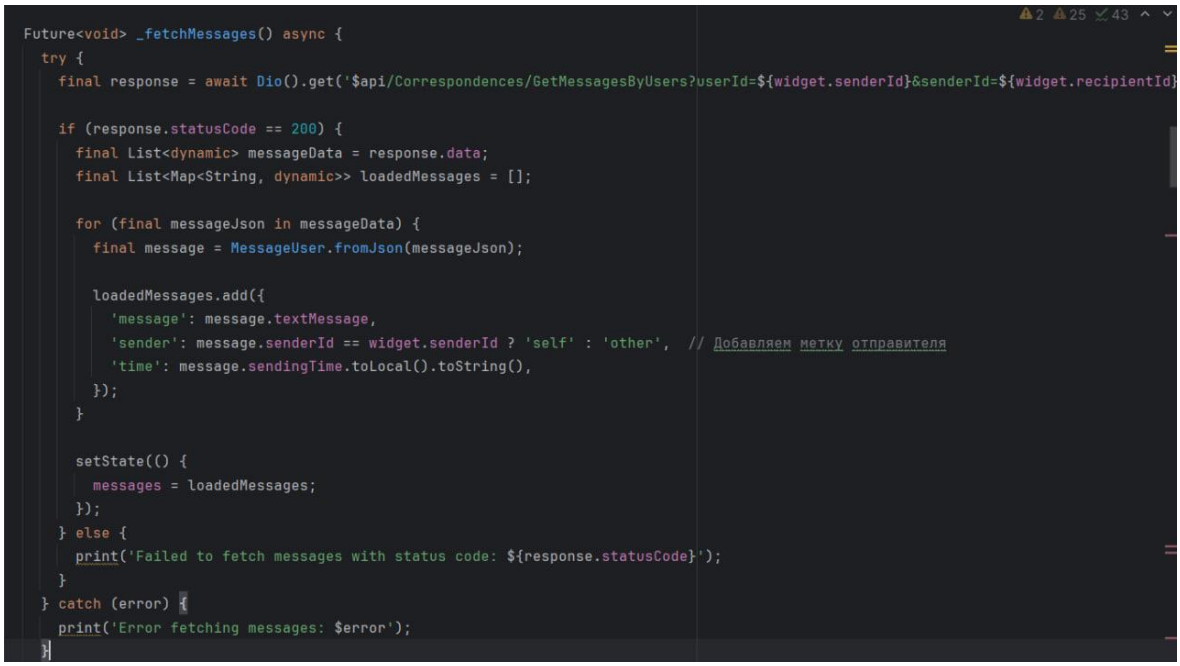
На рисунке 14 представлен контроллер.

Рисунок 14 - Пример контроллера из парадигмы MVC

Рефакторинг и оптимизация являются важными этапами в разработке программного обеспечения. Рефакторинг помогает улучшить качество кода, устранить ошибки и упростить его структуру, что в свою очередь повышает производительность, облегчает поддержку и расширение кода в будущем. Оптимизация обеспечивает эффективную работу приложения, ускоряет его выполнение и уменьшает потребление ресурсов, что позволяет получать более высокую производительность и экономить ресурсы. Ниже на рисунках 15-16 приведен пример

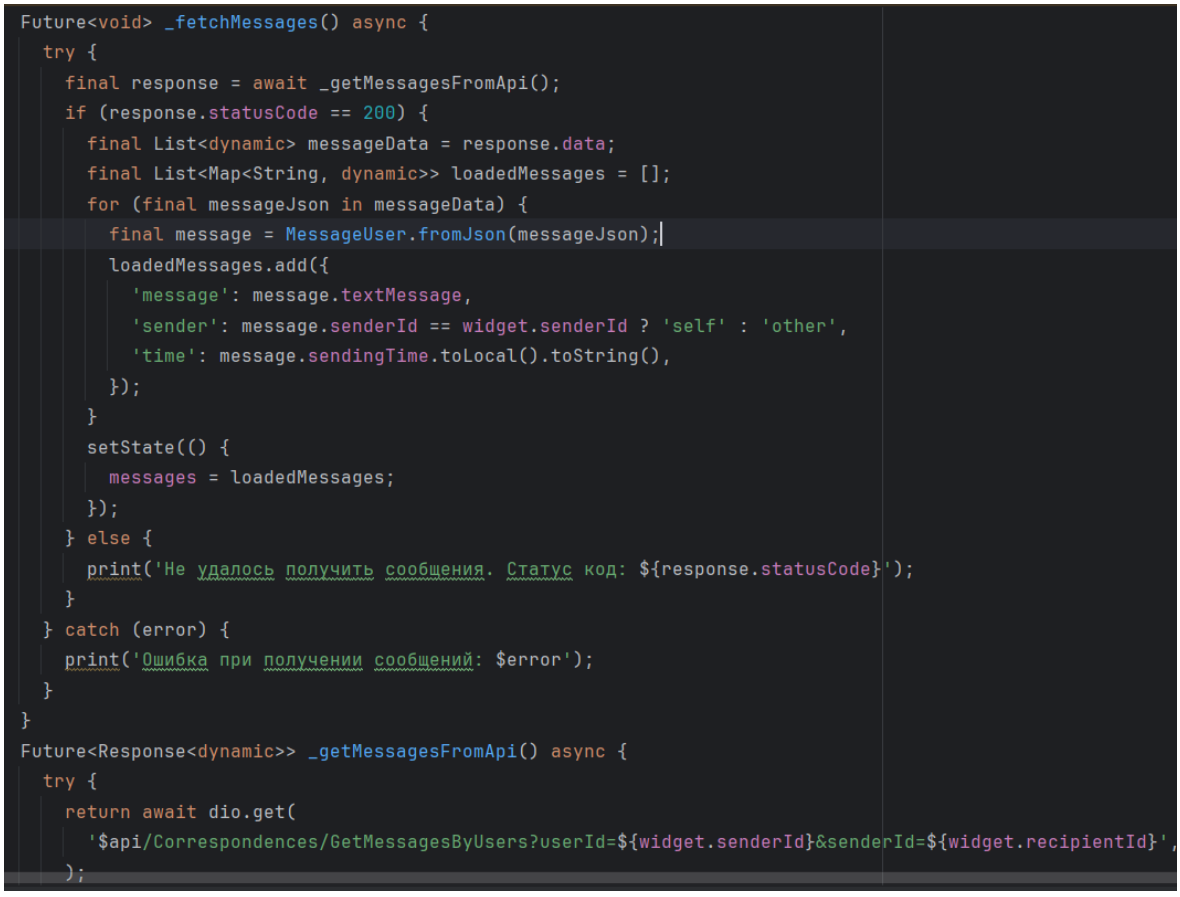
`

рефакторинга кода, путем вынесения функций в отдельные методы что в последствии даст возможность модернизации кода.



```
Future<void> _fetchMessages() async {
  try {
    final response = await Dio().get('$api/Correspondences/GetMessagesByUsers?userId=${widget.senderId}&senderId=${widget.recipientId}

    if (response.statusCode == 200) {
      final List<dynamic> messageData = response.data;
      final List<Map<String, dynamic>> loadedMessages = [];

      for (final messageJson in messageData) {
        final message = MessageUser.fromJson(messageJson);

        loadedMessages.add({
          'message': message.textMessage,
          'sender': message.senderId == widget.senderId ? 'self' : 'other',  // Добавляем метку отправителя
          'time': message.sendingTime.toLocal().toString(),
        });
      }

      setState(() {
        messages = loadedMessages;
      });
    } else {
      print('Failed to fetch messages with status code: ${response.statusCode}');
    }
  } catch (error) {
    print('Error fetching messages: $error');
  }
```

Рисунок 15 - Пример кода до рефакторинга



```
Future<void> _fetchMessages() async {
  try {
    final response = await _getMessagesFromApi();
    if (response.statusCode == 200) {
      final List<dynamic> messageData = response.data;
      final List<Map<String, dynamic>> loadedMessages = [];
      for (final messageJson in messageData) {
        final message = MessageUser.fromJson(messageJson);
        loadedMessages.add({
          'message': message.textMessage,
          'sender': message.senderId == widget.senderId ? 'self' : 'other',
          'time': message.sendingTime.toLocal().toString(),
        });
      }
      setState(() {
        messages = loadedMessages;
      });
    } else {
      print('Не удалось получить сообщения. Статус код: ${response.statusCode}');
    }
  } catch (error) {
    print('Ошибка при получении сообщений: $error');
  }
}
Future<Response<dynamic>> _getMessagesFromApi() async {
  try {
    return await dio.get(
      '$api/Correspondences/GetMessagesByUsers?userId=${widget.senderId}&senderId=${widget.recipientId}',
    );
```

Рисунок 16 - Пример кода после рефакторинга

`

2.2.7. Тесты

Для тестирования программы будет использоваться ручное тестирование по тестовым сценариям, также будет применено нагрузочное тестирование на разработанное API.

Сценарий и результаты тестовых испытаний представлены в Приложении В.

2.2.8. Контроль целостности данных

В ходе разработки информационной системы были применены методы валидации входных данных, которые обеспечивают целостность этих данных. Этот подход включает контроль целостности данных, который определяет ситуации и действия приложения при выполнении задач, связанных с сохранением, изменением или удалением данных.

Таблица 5 - Контроль целостности данных

№	Ситуация	Аномалия	Реакция программы
1	2	3	4
1	Регистрация пользователя	Введен существующий логин	Программа после попытки регистрации сообщает о провале регистрации.
		Введен некорректный пароль	Программа после попытки регистрации сообщает о провале регистрации.
2	Добавление, изменение, удаление сообщения	Введено пустое сообщение при изменении	Программа после попытки изменения данных сообщает о провале.

2.3. Проектирование

2.3.1. Схема архитектуры приложения

Приложение отправляет запросы на сервер, где находится база данных, с помощью API. Запросы и ответы осуществляются по определенному протоколу и формату данных, которые заранее согласовываются между приложением и сервером.

`

Приложение может отправлять запросы на чтение, изменение или удаление данных из базы данных через API. Соответственно, сервер обрабатывает запросы и отправляет обратно ответы с нужной информацией в формате, понятном приложению. Это позволяет приложению получать и обрабатывать данные из базы данных без необходимости обращения к ней напрямую.

Ниже на рисунке 17 представлена схема связи базы данных и приложения через API.

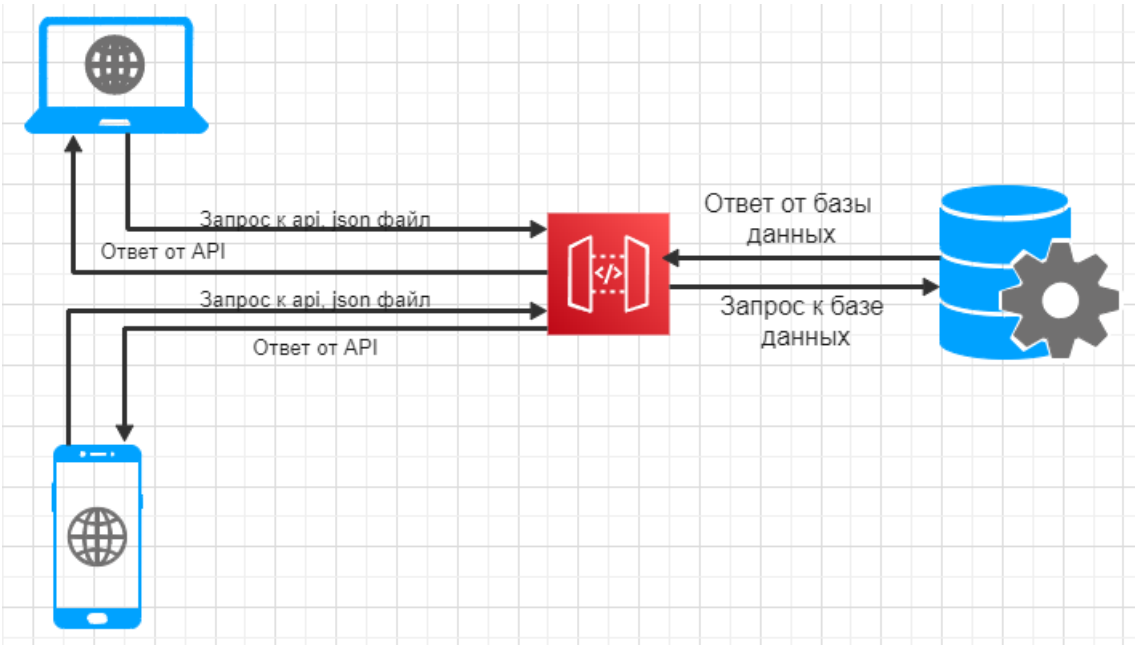


Рисунок 17 - Схема архитектуры приложения

2.3.2. Логическая схема данных

В базе данных находится 13 таблиц это: Пользователь, переписка, сообщения, поддержка, роль, друзья, заявка, новость, фото новости, фото, фото пользователя, достижения, достижения пользователей. Таблицы между собой преимущественно имеют связь МКМ

Ниже на рисунке 18 представлена логическая схема данных разработанная под приложение.

`

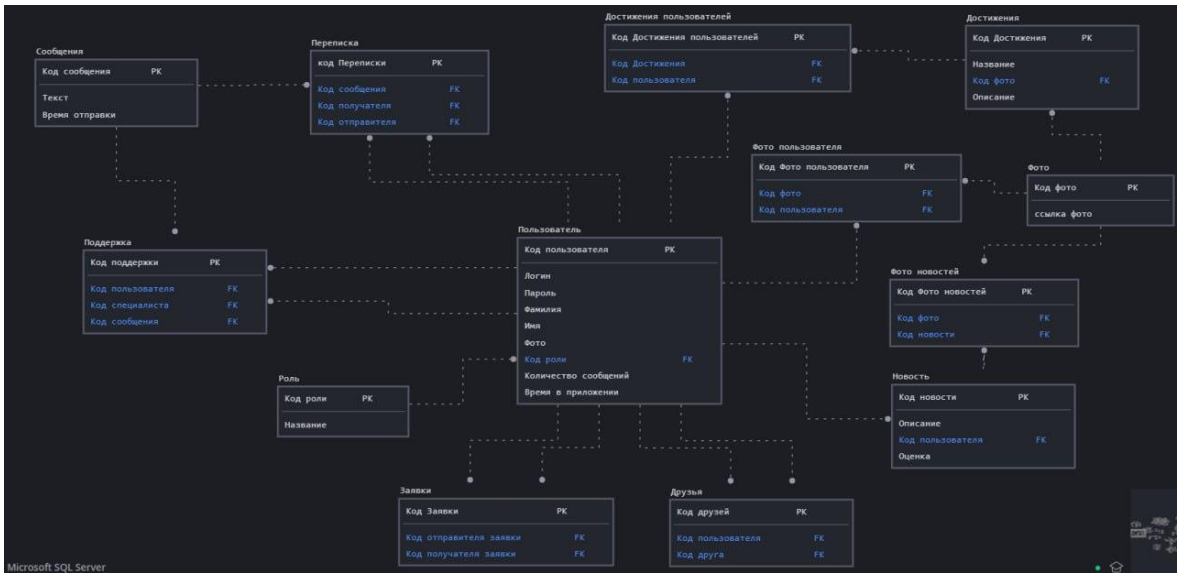


Рисунок 18 - Логическая схема базы данных

2.3.3. Физическая схема данных

В физической схеме данных показано какие типы данных имеют поля в таблицах. Также на ней стрелочками показано какие таблицы имеют связь между собой.

Ниже на рисунке 19 представлена физическая схема базы данных

`

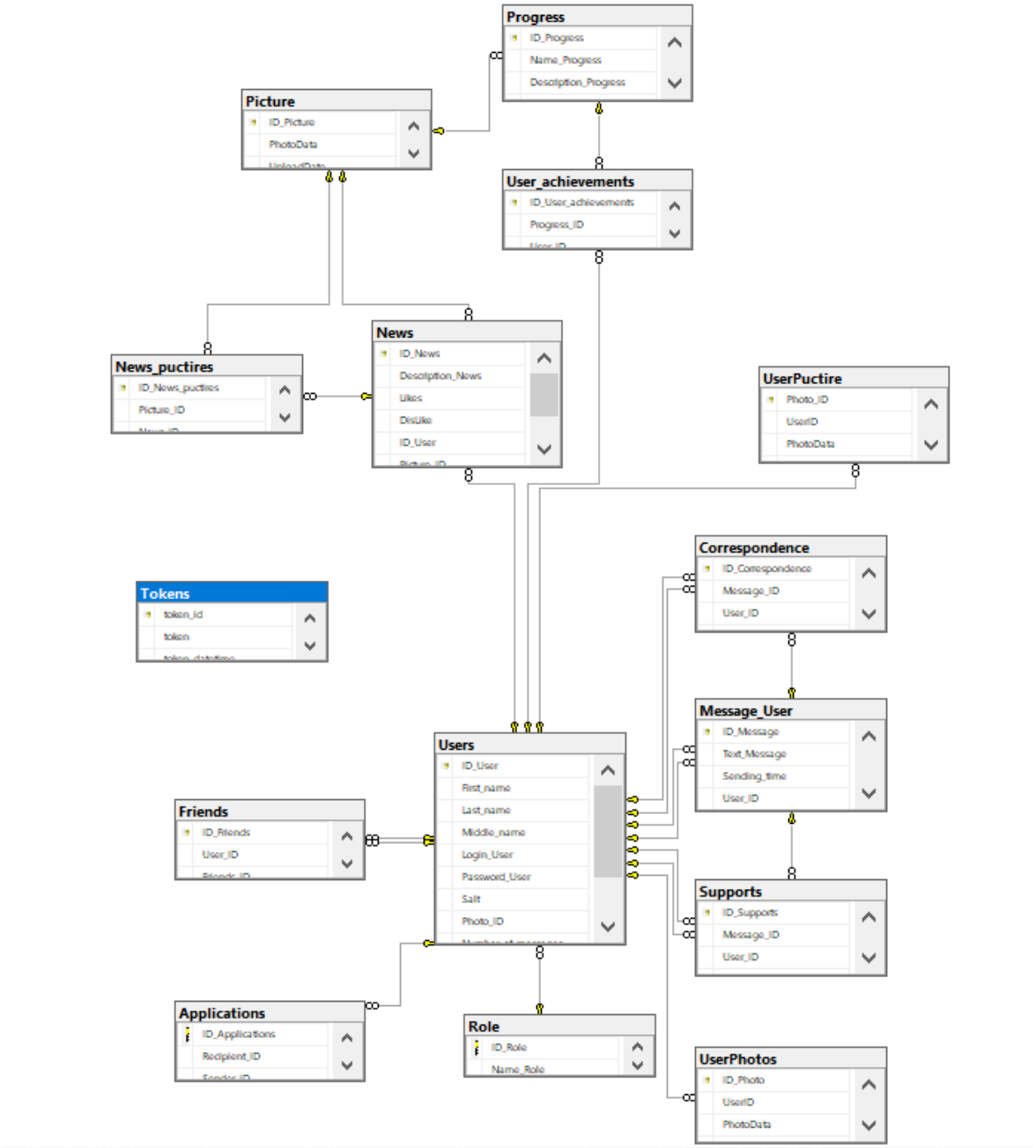


Рисунок 19 - Физическая схема базы данных

Словарь данных представлен в таблице ниже.

Таблица 6 - Словарь данных

Ключ	Наименование	Тип данных	Обязательность заполнения	Описание
ID_User	Идентификатор пользователя	INT	Да	Уникальный идентификатор пользователя

`

First_name	Имя	NVARCHAR(50)	Да	Имя пользователя
Last_name	Фамилия	NVARCHAR(50)	Да	Фамилия пользователя
Middle_name	Отчество	NVARCHAR(50)	Нет	Отчество пользователя
Login_User	Логин пользователя	VARCHAR(50)	Да	Логин пользователя для входа в систему
Password_User	Пароль пользователя	VARCHAR(100)	Да	Пароль пользователя для входа в систему
Salt	Соль	VARCHAR(256)	Нет	Дополнительный уровень защиты пароля
Photo_ID	Идентификатор фотографии пользователя	INT	Нет	Ссылка на фотографию пользователя
Number_of_messages	Количество сообщений	INT	Нет	Количество сообщений пользователя
Time_in_the_app	Время в приложении	TIME	Нет	Время, проведенное пользователем в приложении
Role_ID	Идентификатор роли пользователя	INT	Да	Ссылка на роль пользователя
ID_Role	Идентификатор роли	INT	Да	Уникальный идентификатор роли
Name_Role	Наименование роли	VARCHAR(50)	Да	Название роли пользователя
ID_Message	Идентификатор сообщения	INT	Да	Уникальный идентификатор сообщения
Text_Message	Текст сообщения	VARCHAR(max)	Да	Текст сообщения
Sending_time	Время отправки сообщения	DATETIME	Да	Время отправки сообщения

`

User_ID	Идентификатор пользователя	INT	Да	Ссылка на пользователя, связанного с сообщением
Sender_ID	Идентификатор отправителя	INT	Да	Ссылка на пользователя, отправившего сообщение
ID_Picture	Идентификатор изображения	INT	Да	Уникальный идентификатор изображения
PhotoData	Бинарные данные изображения	VARBINARY(max)	Да	Двоичные данные изображения
UploadDate	Дата загрузки изображения	DATETIME	Да	Дата и время загрузки изображения
ID_Progress	Идентификатор прогресса	INT	Да	Уникальный идентификатор прогресса
Name_Progress	Название прогресса	NVARCHAR(50)	Да	Название прогресса
Description_Progress	Описание прогресса	NVARCHAR(100)	Да	Описание прогресса
ID_News	Идентификатор новости	INT	Да	Уникальный идентификатор новости
Description_News	Описание новости	VARCHAR(100)	Да	Описание новости
Likes	Количество лайков	INT	Нет	Количество лайков новости
DisLike	Количество дизлайков	INT	Нет	Количество дизлайков новости
Picture_ID	Идентификатор изображения	INT	Да	Ссылка на изображение, связанное с новостью
ID_News_puctires	Идентификатор связи изображения и новости	INT	Да	Уникальный идентификатор связи изображения и новости
ID_Friends	Идентификатор дружбы	INT	Да	Уникальный идентификатор дружбы

`

Friends_ID	Идентификатор друга	INT	Да	Уникальный идентификатор друга
ID_Applications	Идентификатор заявки	INT	Да	Уникальный идентификатор заявки
Recipient_ID	Идентификатор получателя	INT	Да	Ссылка на пользователя-получателя заявки
Sender_ID	Идентификатор отправителя	INT	Да	Ссылка на пользователя-отправителя заявки
ID_Supports	Идентификатор поддержки	INT	Да	Уникальный идентификатор запроса поддержки
Specialist_ID	Идентификатор специалиста	INT	Да	Ссылка на специалиста поддержки

2.3.4. Структурная схема

В данной структурной схеме, также известной как диаграмма классов, представлена графическая модель, показывающая связи, зависимости и отношения между классами в программном проекте. Классы, как объекты программирования, могут иметь свойства, методы и поля. Диаграмма классов обычно используется для представления сложных систем и облегчения понимания связей между классами, что упрощает разработку и сопровождение кода в будущем.

Диаграмма классов приложения:

На рисунке 20 представлена схема приложения для персонала.

`

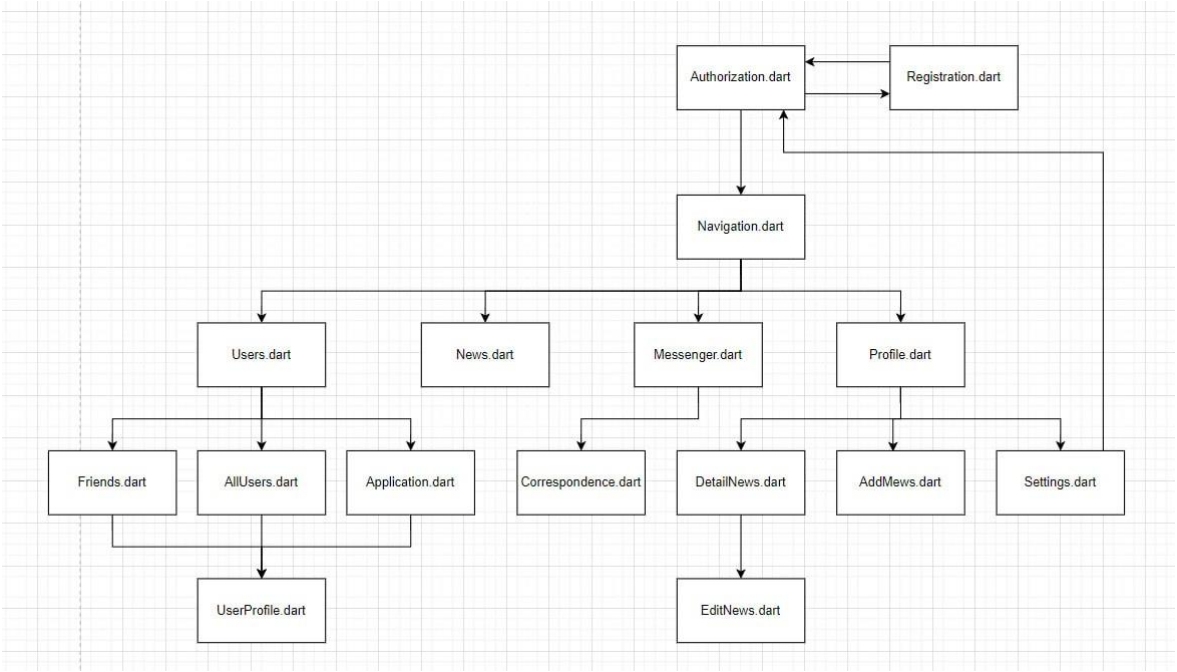


Рисунок 20 - Структурная схема приложения персонала.

Ниже представлена таблица с названием модулей и описание, в котором кратко описано за что отвечает модуль.

Таблица 7 Модули приложения клиента

Номер	Название модуля	Описание модуля
1	Authorization.dart	Данный модуль содержит логику и разметку окна авторизации, отвечает за вход пользователя в систему.
2	Registration.dart	Данный модуль содержит логику и разметку окна регистрации, отвечает за добавление пользователя в систему.
3	Navigation.dart	Данный модуль содержит логику и разметку окна заявок авторизованного пользователя.
4	Messenger.dart	Данный модуль содержит логику и разметку окна добавления заявки пользователя.
5	News.dart	Данный модуль содержит логику вывода списка заявок пользователя в лист.
6	Users.dart	Данный модуль отвечает за отображение списка пользователей.
7	Profile.dart	Модуль, реализующий функционал управления профилем пользователя.
8	Friends.dart	Модуль для работы с друзьями пользователя.
9	AllUsers.dart	Отображение списка всех пользователей приложения.
10	Application.dart	Основной модуль, инициализирующий приложение и управляющий роутингом.
11	UserProfile.dart	Отображение информации о профиле конкретного пользователя.
12	Correspondence.dart	Модуль для ведения переписки между пользователями.
13	DetailNews.dart	Подробный просмотр новости.
14	EditNews.dart	Редактирование существующих новостей.

`

15	AddNews.dart	Создание и добавление новостей в приложение.
16	Settings.dart	Модуль для управления настройками приложения.

2.3.5. Функциональная схема

На функциональной схеме представлены функции, находящие на окнах приложений. Данная схема позволяет легче прослеживать связи между компонентами приложения и оценивать работу системы в целом.

На рисунке 21 представлена функциональная схема приложение для персонала.

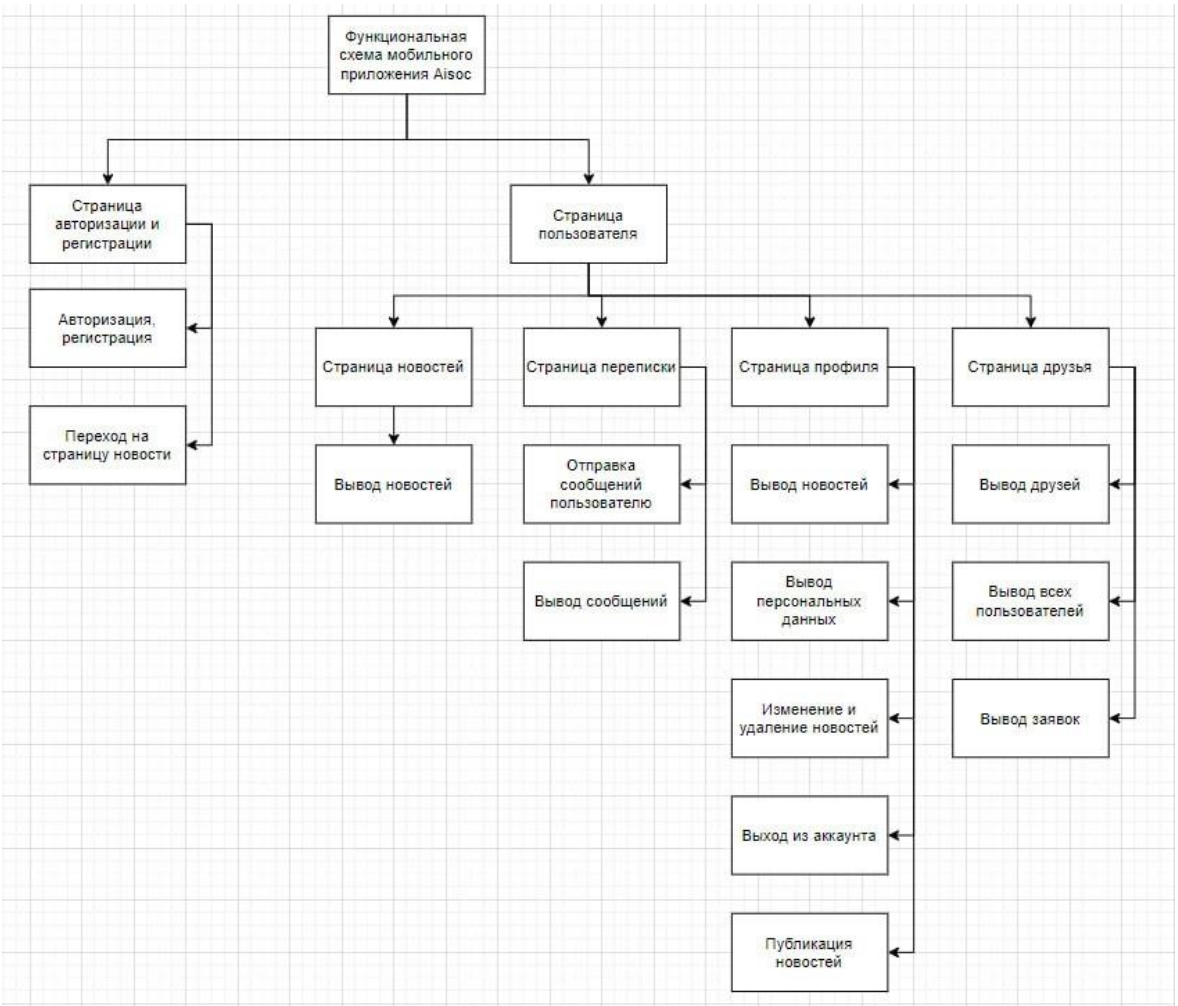


Рисунок 21 - Функциональная схема приложения для персонала.

2.3.6. Диаграмма классов

Ниже представлено диаграмма классов проекта.

`

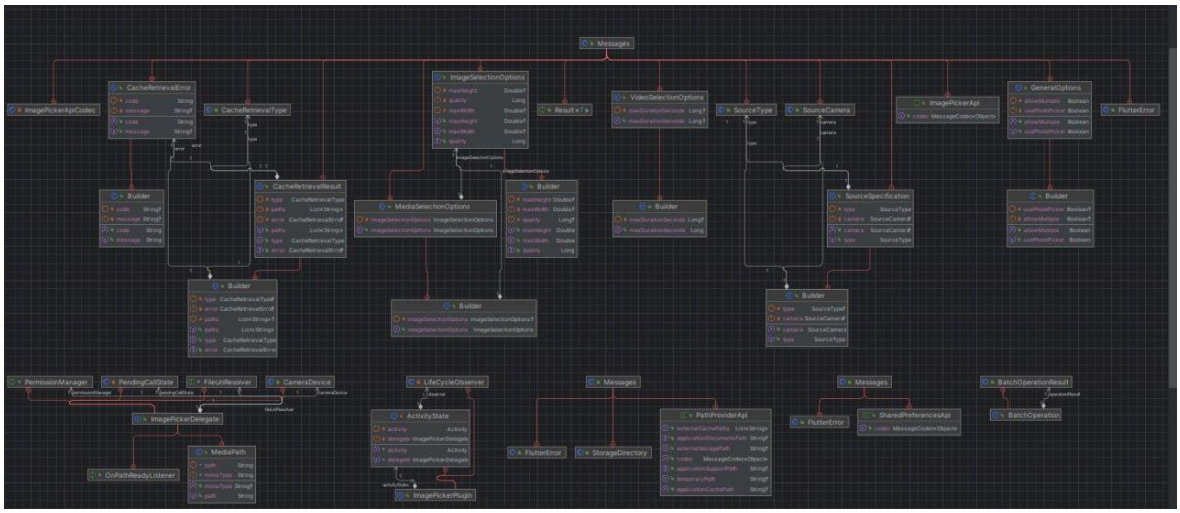


Рисунок 22 - Диаграмма классов приложения для Персонала

2.3.7. Схема тестирования

Схема тестирования показана в приложении В Сценарий и результаты тестовых испытаний.

2.3.8. Схема пользовательского интерфейса

Ниже представлена схема пользовательского интерфейса, которая показывает, как осуществляется переход между окнами приложений.

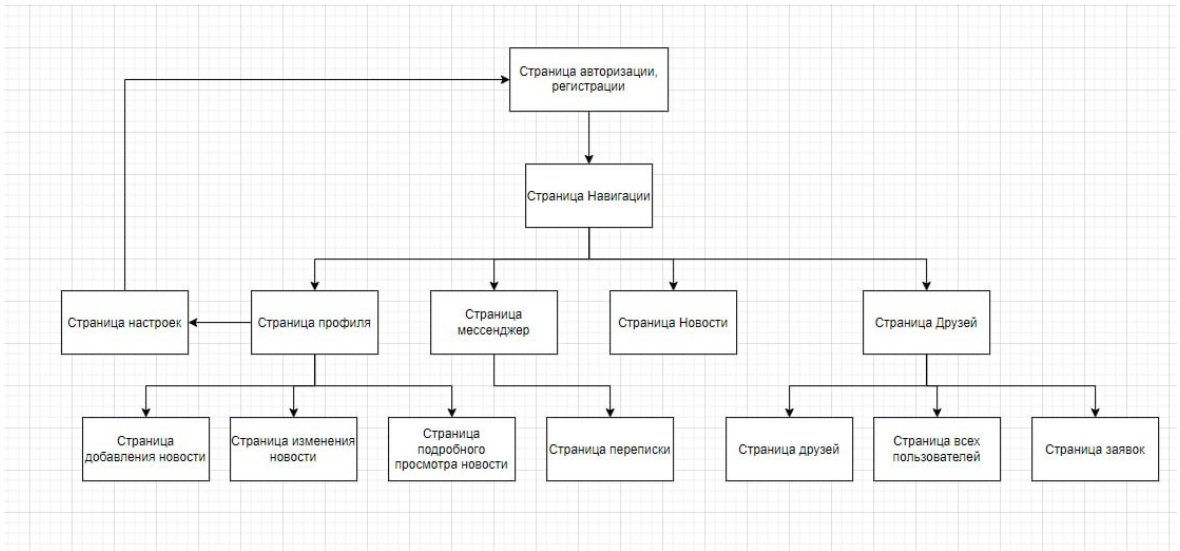


Рисунок 23 - Схема пользовательского интерфейса

2.4. Результат работы программы

В результате выполнения поставленной задачи, было разработано решение, которое является мобильным приложением для социальной сети «DelWiz».

`

На рисунках 24-25 представлens результаты работы программы. Больше информации о результат работы программы описан в приложении Г «Руководство пользователя».

Ниже представлены фотографии приложения для пользователя:

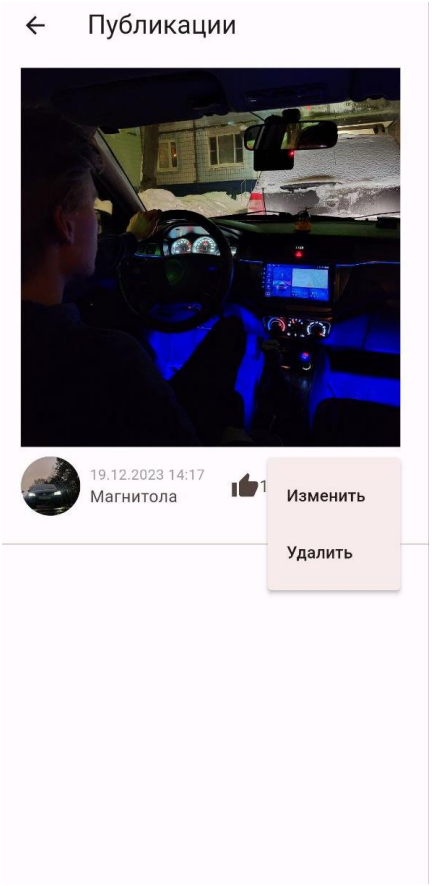


Рисунок 24 – Новости пользователя

`

Рисунок 25 – Сообщения пользователя

3. ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ

3.1. Инструментальные средства

MsSQL была выбрана в качестве основы для базы данных из-за её удобного ядра и использования SQL Server Management Studio для разработки. Postgres, хоть и бесплатная и открытая, обладает богатым функционалом, но настройка и управление могут быть сложными. MySQL, также бесплатная и популярная, уступает MSSQL и Postgres в определенных возможностях, таких как поддержка транзакций, полнотекстовый поиск и работа с большими объемами данных. Именно поэтому MS SQL Server была предпочтительнее при выборе ядра для этой базы данных.

Были выбраны инструменты разработки, такие как Intellij IDEA для написания мобильного приложения, а также Microsoft Visual Studio 2022 для создания API и WPF приложения. Intellij IDEA была выбрана в качестве оптимальной среды разработки приложений под Android на языке Dart. Microsoft Visual Studio 2022, с фреймворком ASP.NET и архитектурой MVC, использовалась для удобства разработки API. Архитектура MVC предоставляет значительные преимущества, включая возможность генерировать стандартные запросы к API. Кроме того, Visual Studio выбрана благодаря возможности создавать приложения для операционной системы Windows.

3.2. Отладка программы

Отладка приложения и API была осуществлена с помощью встроенных средств отладки в Intellij IDEA и в IDE Visual Studio. Отладка использовалась при включенном приложении и приостанавливалась, доходя до точки остановки.

В ходе работы над проектом было распространено несколько ошибок. Первая ошибка была проблемой с пользователями, происходила

`

неверная инициализация, из-за чего пользователь был пустым и его персональные данные.

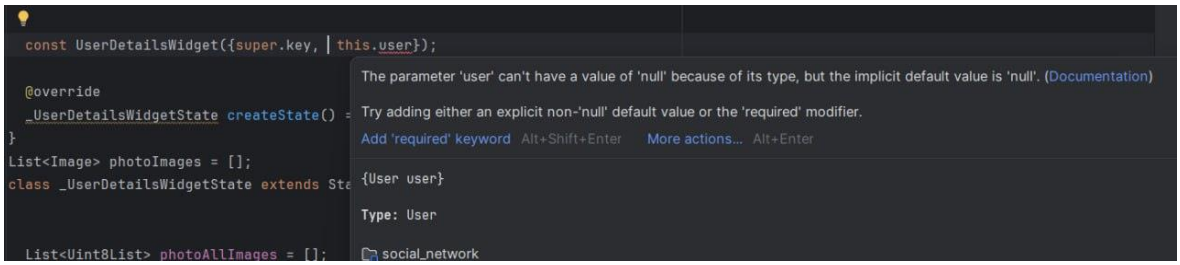


Рисунок 26 - Ошибка "нет подключение к IP-адресу"

Второй распространенной проблемой было не правильный запрос, к методу, который вызывал ошибку «400». Ошибка исправляется верным заполнением данных и перезапуском запроса.

Рисунок 27 - Ошибка не правильного обращение к методу

3.3. Защитное программирование

Механизм try catch в программе и в API помогает обнаруживать ошибки во время выполнения кода и предпринимать соответствующие действия для их обработки, что повышает надежность и безопасность системы.

В случае возникновения ошибок, механизм try_catch позволяет изолировать проблемный участок кода и обработать ошибку локально, не прерывая выполнение всего скрипта или приложения, что улучшает производительность и снижает риски сбоев.

Ниже представлен пример его использования:

`

```
Future<dynamic> request(String login, String password) async {
/*  login = _loginController.text;
  password = _passwordController.text;*/
  Response response;
  try {
    Response response = await dio.get('$api/Users/$login/$password');
    print(response.data.toString());
    // Handle the response here
  } catch (e) {
    if (e is DioError) {
      print("Dio Error: ${e.message}");
      print("Dio Response: ${e.response?.data}");
    }
  }
}
```

Рисунок 28 - Блок try catch

Так же для безопасности использовалось хеширования пароля пользователя с помощью соли.

```
public static byte[] GenerateSalt(int length)
{
    byte[] salt = new byte[length];
    using (RandomNumberGenerator rng = RandomNumberGenerator.Create())
    {
        rng.GetBytes(salt);
    }
    return salt;
}
```

Рисунок 29 - Методы хеширования пароля.

Так же для безопасности был использован токен, который генерируется при авторизации пользователя. В токене шифруется номер учетной записи пользователя и его роль. После чего у каждого запроса в API есть проверка на то какие роли имеют к ней доступ.

`

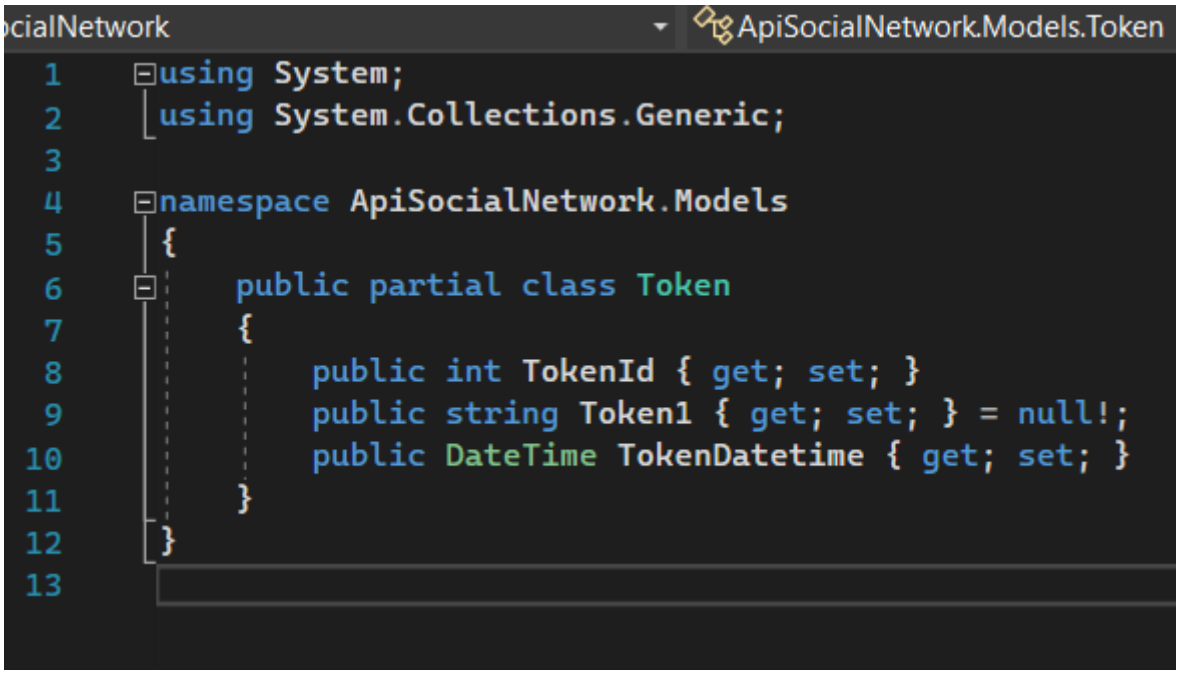


Рисунок 30 - Методы для работы с токеном.

3.4. Характеристики программы

Разработанное мобильное приложение «DelWiz» может запускаться на операционной системе Android и IOS. Основным критерием считается версия ОС Android, на котором запускается разработанный ресурс, версия должна быть 8.0 или выше.

Приложения администратора и персонала могут быть запущены на операционных системах Windows 10 и выше.

Характеристики программы представлены в приложении Б «Текст программы» в таблице 1 «Модули».

ЗАКЛЮЧЕНИЕ

"В результате разработки социальной сети "DelWiz" на платформах Dart и Flutter была создана уникальная и инновационная платформа, предлагающая пользователю не просто возможность общения, но и мир бесконечных возможностей для самовыражения и обмена впечатлениями. Это приложение является пространством для творчества, вдохновения и совместного обогащения жизни каждого участника.

Мы успешно реализовали и выпустили приложение, которое удовлетворяет ожидания пользователя, обеспечивая безупречную работу, максимальную удобство и высокую производительность. Сочетание Dart и Flutter дало нам возможность создать удивительно интуитивный интерфейс и функционал, который соответствует современным стандартам и ожиданиям наших пользователей.

В ходе разработки были использованы передовые технологии и методы, что позволило нам создать приложение, сочетающее в себе качество, надежность и инновационный дизайн. Этот проект стал не только удачным шагом в области социальных сетей, но и плодотворным опытом для нашей команды, давшим нам ценные знания и навыки для будущих проектов.

"DelWiz" представляет собой не только социальную сеть, но и платформу для интеллектуального обмена идеями, поддержки друг друга и создания уникального сообщества. Наше приложение - это шаг к более насыщенной и интересной жизни, где каждый пост и обмен моментами добавляют ценности не только в вашу жизнь, но и в жизнь окружающих.

Мы видим потенциал для дальнейшего развития "DelWiz", и одним из ключевых направлений является улучшение взаимодействия пользователей через добавление чата, что поможет улучшить коммуникацию и обогатить опыт пользователей.

`

Разработка "DelWiz" стала важным этапом нашего творческого пути, и мы уверены, что это приложение окажет позитивное влияние на повседневную жизнь пользователей, поддерживая их в их идеях, творчестве и общении."

`

СПИСОК ИСПОЛЬЗУЕМЫХ МАТЕРИАЛОВ

1. Microsoft Docs - SQL Server Documentation: https://docs.microsoft.com/en-us/sql/ (Дата использования: 01.10.23)
2. Flutter Documentation: https://flutter.dev/docs (Дата использования: 02.10.23)
3. Dart Documentation: https://dart.dev/guides (Дата использования: 03.10.23)
4. ASP.NET Documentation: https://docs.microsoft.com/en-us/aspnet/ (Дата использования: 04.10.23)
5. Microsoft SQL Server Official Website: https://www.microsoft.com/en-us/sql-server/ (Дата использования: 05.10.23)
6. C# Programming Guide: https://docs.microsoft.com/en-us/dotnet/csharp/ (Дата использования: 06.10.23)
7. Flutter and Dart Package Repository - pub.dev: https://pub.dev/ (Дата использования: 07.10.23)
8. ASP.NET Core Documentation: https://docs.microsoft.com/en-us/aspnet/core/ (Дата использования: 08.10.23)
9. SQL Server Central - Community and Resources: https://www.sqlservercentral.com/ (Дата использования: 09.10.23)
10. Flutter YouTube Channel: https://www.youtube.com/c/FlutterDev (Дата использования: 10.10.23)
11. Dart GitHub Repository: https://github.com/dart-lang (Дата использования: 11.10.23)
12. C# GitHub Repository: https://github.com/dotnet/csharplang (Дата использования: 12.10.23)
13. SQL Server Blog - Official Microsoft Blog for SQL Server: https://techcommunity.microsoft.com/t5/sql-server-blog/bg-p/SQLServer (Дата использования: 13.10.23)
14. Flutter Cookbook: https://flutter.dev/docs/cookbook (Дата использования: 14.10.23)
15. .NET Blog - Official Blog for the .NET development community: https://devblogs.microsoft.com/dotnet/ (Дата использования: 15.10.23)
16. Flutter GitHub Repository: https://github.com/flutter/flutter (Дата использования: 16.10.23)
17. DartPad - Online Dart Editor: https://dartpad.dev/ (Дата использования: 17.10.23)
18. SQL Server Books Online: https://technet.microsoft.com/en-us/sqlserver/bb671430.aspx (Дата использования: 18.10.23)
19. ASP.NET Community Forums: https://forums.asp.net/ (Дата использования: 19.10.23)
20. C# Corner - Online Community for Software Developers: https://www.c-sharpcorner.com/ (Дата использования: 20.10.23)

`

21. FlutterFire - Firebase Integration for Flutter: https://firebase.flutter.dev/ (Дата использования: 21.10.23)
22. .NET Foundation - Supporting Open Source .NET Projects: https://dotnetfoundation.org/ (Дата использования: 22.10.23)
23. Dart SDK GitHub Repository: https://github.com/dart-lang/sdk (Дата использования: 23.10.23)
24. SQL Server Performance Tuning Tips: https://www.mssqltips.com/ (Дата использования: 24.10.23)
25. Xamarin - Cross-platform App Development with .NET: https://dotnet.microsoft.com/apps/xamarin (Дата использования: 25.10.23)
26. Entity Framework Documentation: https://docs.microsoft.com/en-us/ef/ (Дата использования: 26.10.23)
27. Flutter Medium Publication: https://medium.com/flutter (Дата использования: 27.10.23)
28. SQL Server Query Optimization Techniques: https://www.red-gate.com/simple-talk/sql/performance/sql-query-optimization-techniques-in-sql-server/ (Дата использования: 28.10.23)
29. Blazor Documentation - Building Interactive Web Applications with .NET: https://docs.microsoft.com/en-us/aspnet/core/blazor/ (Дата использования: 29.10.23)
30. Stack Overflow - Community-driven Question and Answer Platform: https://stackoverflow.com/ (Дата использования: 30.10.23)

`