Министерство науки и высшего образования Российской Федерации
Муромский институт (филиал)
федерального государственного бюджетного образовательного учреждения высшего образования
«Владимирский государственный университет
Имени Александра Григорьевича и Николая Григорьевича Столетовых»
(МИВлГУ)

Факультет ИТР

Кафедра ПИн

КУРСОВАЯ РАБОТА

По Распределенные системы обработки данных

Тема Распределенная ИС «Чаты социальных сетей»

Руководитель

_____ Белякова А.С.

(оценка) (фамилия, инициалы)

_____

(подпись) (дата)

Члены комиссии

Студент ПИн-121

(группа)

_____ Ермилов М.В.

(подпись) (Ф.И.О.) (фамилия, инициалы)

_____ _____

(подпись) (Ф.И.О.) (подпись) (дата)

Муром 2023

В данной курсовой работе представлен процесс создания простого веб-мессенджера с использованием языка программирования C# и технологий ASP.NET и Entity Framework

.

In this term paper, the process of creating a simple web messenger using the C# language with ASP.NET and Entity Framework technologies is presented.

Содержание

Введение

Быстрое развитие технологий и увеличивающийся спрос на мгновенное общение преобразовали наши способы взаимодействия и связи друг с другом. Веб-мессенджеры стали неотъемлемой частью нашей повседневной жизни, предлагая безупречные и мгновенные решения для общения как в личных, так и в профессиональных целях. Данная курсовая работа направлена на изучение процесса создания простого веб-мессенджера с использованием языка C# и мощных фреймворков ASP.NET и Entity Framework.

Основная цель этого проекта - создать простой веб-мессенджер, который позволит пользователям регистрироваться, управлять контактами, отправлять и получать сообщения, а также выполнять различные взаимодействия, такие как добавление в друзья и блокировка пользователей. Используя ASP.NET, надежный веб-фреймворк, и Entity Framework, передовой Object-Relational Mapper (ORM), этот проект нацелен на демонстрацию практического применения этих технологий в создании функционального и масштабируемого веб-приложения.
На протяжении данной работы мы рассмотрим проектирование и реализацию веб-мессенджера, включая архитектуру, схему базы данных и основные функциональные возможности. Мы также рассмотрим возникающие проблемы в процессе разработки и обсудим потенциальные улучшения и дальнейшие доработки. Завершение этого проекта предоставит ценные знания о тонкостях разработки веб-приложений и практическом применении современных веб-технологий.

1 Анализ технического задания

Разработать распределенную ИС для автоматизации предметной области по технологии ASP.NET Core MVC.

Программный продукт представляет собой систему чатов социальных сетей, где пользователи могут посылать друг-другу сообщения, обмениваться файлами. Пользователи так-же могут регистрироваться в системе и настраивать свой профиль в любое время. Так-же пользователю необходимо иметь возможность работы с «друзьями», а именно добавлять, удалять и блокировать людей.

Для, администраторов проекта, необходимо иметь возможность просмотра всех пользователей, их отношений и их сообщений. Так-же у администраторов должна быть возможность, выгружать данные с проекта в формат Excel.

Реализуемая программа должна разработана на технологии ASP.NET Core MVC. Технология ASP.NET Core MVC является одной из наиболее популярных и мощных платформ для разработки веб-приложений. Она предоставляет разработчикам широкий набор инструментов и возможностей для создания масштабируемых, высокопроизводительных и безопасных приложений. Процесс работы APS.Net MVC представлен на Следующем рисунке.

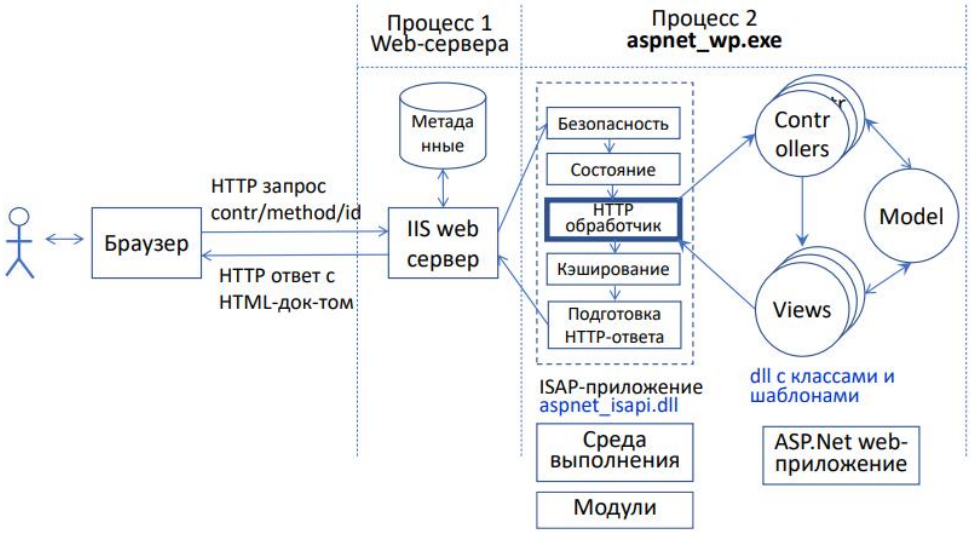

Рисунок 1 – Процесс работы APS.Net MVC

К достоинствам ASP.Net MVC относятся:

1) Разделение ответственности:

- MVC-приложение состоит из трех частей (контроллера, представления модели), каждая из которых выполняет свои специфичные функции (в итоге приложение будет легче поддерживать модифицировать в будущем);

- В силу разделения ответственности приложения MVC обладают лучшей тестируемостью (можно тестировать отдельные компоненты независимо друг от друга).

2) Соответствие протоколу HTTP:

- MVC-приложения в отличие от веб-форм не поддерживают объекты состояния (ViewState);

- Ясность и простота платформы позволяют добиться большего контроля над работой приложения.

3) Гибкость:

- можно настраивать различные компоненты платформы по своему усмотрению;

- можно изменять части конвейера работы MVC или адаптировать его к своим нуждам и потребностям.

Технология ASP.NET MVC – это фреймворк на основе которого разработчик создает свое web-приложение. Фреймворк включает набор совместно работающих классов, а также вспомогательные конструкции. Разработчику для создания приложения нужно понять логику работы фреймворка и способ подключения для выполнения требуемых работ. Подключение выполняется за счет создания производных классов и вызова нужных методов.

Компоненты MVC. Архитектурный шаблон Model-View-Controller (MVC) разделяет web-приложение на части, которые легче кодировать, тестировать и поддерживать.

Модель (model) состоит из файлов, которые содержат код доступа к данным, код бизнес-логики, код валидации данных.

Представление (view) состоит из файлов, которые используются для создания HTML документа, содержащего интерфейс пользователя.

Контроллер (controller) состоит из файлов, которые получают запросы пользователя, требуемые данные из модели, после чего передают эти данные нужному представлению.

Модель представления (view model) – необязательный компонент, который состоит из файлов, единственной работой которых является передача данных между контроллерами и представлениями. Технически это не является частью паттерна MVC, но view models часто используются при работеASP.NET MVC. Схема работы MVC-фреймворка отражена на рисунке 2. Обработка запросов к MVC приложению представлена на рисунке 3.

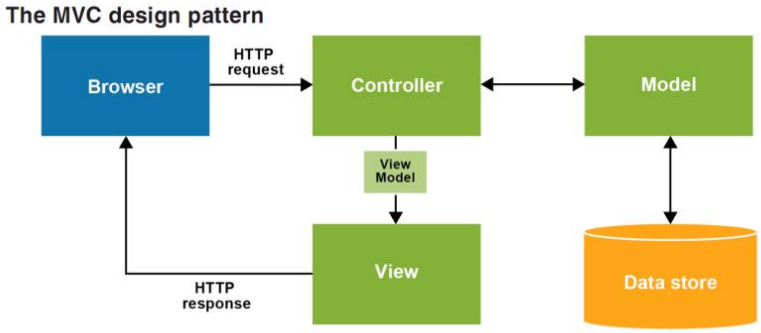

Рисунок 2 – Схема работы MVC-фреймворка

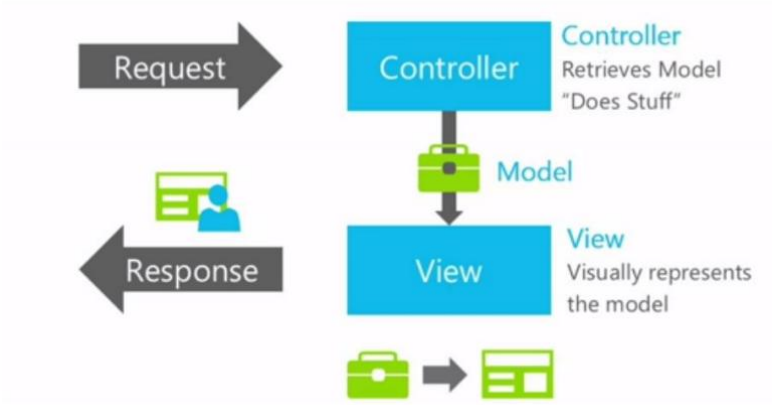

Рисунок 3 – Обработка запросов к MVC приложению

2 Разработка моделей данных

2.1 Концептуальная модель данных

Концептуальная модель хранилища данных представляет собой описание главных (основных) сущностей и отношений между ними. Концептуальная модель является отражением предметных областей, в рамках которых планируется построение хранилища данных.

При проектировании концептуальной модели структурируют данные и выявляют взаимосвязи между ними, без рассмотрения особенностей реализации и вопросов эффективности обработки, поэтому концептуальная модель не является полностью подходящей для дальнейшей разработки, все таблицы должны быть нормализованы для реляционной базы данных. Составленная концептуальная модель представлена на рисунке 1 приложения 1.

2.2 Логическая модель данных

Логическая модель расширяет концептуальную путем определения для сущностей их атрибутов, описаний и ограничений, уточняет состав сущностей и взаимосвязи между ними.

Концептуальная модель изменяется так, чтобы она могла быть обеспечена конкретной моделью данных.

В результате формируется логическая модель. Логическая модель отражает логические связи между элементами данных вне зависимости от их содержания и среды хранения.

Логическая модель может быть реляционной, иерархической или сетевой.

В качестве способа организации информационной базы выбрана реляционная база данных. Именно такой способ хранения всех данных является

наиболее подходящим для проектируемой информационной системы по следующим причинам:

− наглядность модели для пользователя: все данные в реляционной модели представлены в табличной форме;

− независимость данных от программного продукта для их обработки;

− реляционные базы данных являются наиболее распространенными среди разработчиков ПО, следовательно, использование этих баз позволит сэкономить время и бюджет на внедрение нового типа БД.

Составленная логическая модель представлена на рисунке 2 приложения 1.

1.3. Физическая модель данных

Физические модели данных служат для отображения моделей данных. Основными понятиями модели данных являются поле, логическая запись, логический файл. Слово "логический" введено, чтобы отличать понятия, относящиеся к логической модели данных, от понятий, относящихся к физической модели данных. Основными понятиями физической модели данных, используемыми для представления логической модели данных, являются поле, физическая запись, физический файл. В частности, логическая запись, состоящая из полей, может быть представлена в виде физической записи (из тех же полей), логический файл – в виде физического файла. Имена таблиц и колонок будут сгенерированы на основе сущностей и атрибутов логической модели, учитывая максимальную длину имени и другие синтаксические ограничения, накладываемые СУБД. Если в имени сущности или атрибута встречается пробел, он заменяется на символ «_».

Физическая модель описывает способ хранения данных в базе данных. В физической модели мы учитываем типы данных, индексы, ограничения целостности и другие технические детали.

Составленная физическая модель представлена на рисунке 3 приложения 1.

В соответствии с данными моделями с помощью APS.NET Core MVC были сгенерированы следующие три таблицы:

Таблица 1 - таблица Users

Название	Тип данных
Id	PRIMARY KEY, INTEGER, NOT NULL
Name	UNIQUE, NVARCHAR(max), NOT NULL
Email	UNIQUE, NVARCHAR(max), NOT NULL
Password	NVARCHAR(max), NOT NULL
Photo	NVARCHAR(max), NOT NULL

Таблица 2 - таблица Interactions

Название	Тип данных
User1	PAIR KEY, INTEGER, NOT NULL
User2	PAIR KEY, INTEGER, NOT NULL
Type	INTEGER, NOT NULL

Таблица 3 - таблица Messages

Название	Тип данных
Id	PRIMARY KEY, INTEGER, NOT NULL
Sender	PAIR KEY, INTEGER, NOT NULL
Recipient	PAIR KEY, INTEGER, NOT NULL
Date	INTEGER, NOT NULL
DateEdit	INTEGER, NOT NULL
Text	NVARCHAR(max)
Photo	NVARCHAR(max)

4 Разработка и реализация системы

Для создания проекта, и для правильной работы с ней, пришлось не использовать Identity, а написать его с нуля. Так-же для отношений пришлось использовать условие User1Id < User2Id, и делать эту пару - уникальным значением. Для этого пришлось прописывать большое кол-во условий и ограничений.

Для создания своей системы регистрации, пришлось использовать такую технологию как IHttpContextAccessor, которая позволяет сохранять текущую сессию пользователя, для того, чтобы правильно выдавать результаты для тех или иных действий.

Для администраторов, было решено сделать права для первых двух пользователях в системе. Так-же из контроллеров CRUD, был убран весь функционал, кроме просмотра данных и выгрузки их в Excel формат. Так-же, для безопасности пользователей, администраторы не могут видить пароль. Пароль пока-что храниться в базе в открытом виде, но это можно будет исправить, и хранить уже хэш пароля.

При миграции базы данных, возникла ошибка, из-за чего пришлось менять в файлах миграции все следующие параметры: с ReferentialAction.Cascade на ReferentialAction.NoAction. Эта ошибка заключалась в обновлении таблиц, после удаления какой то записи, из-за чего могла возникнуть рекурсия. Как решить данную проблему более правильно, пока неизвестно.

Для работы с профилем, было создано несколько страниц, а именно: Accaunt, Accaunt/User, Accaunt/Login и Accaunt/Registration. Вся логика была прописана в следующем классе: AccountController. Эта система позволила не перегружать проект лишними базами и таблицами и в случае необходимости, позволит безпрепядственно обновить проект.

Для разграничение ролей на пользователя и админа, было принято решение, что роль администратора имеют 2 первых пользователя в системе, для этого был

создан параметр Program.Admin = 3, который говорит о том, что с третьего пользователя в системе, будет отсчет на обычного пользователя.

5 Тестирование системы

Для тестирования системы, было создано несколько аккаунтов, которые позволяли симулировать поведения пользователей, а именно: блокировка и добавление в друзья, вход, регистрация и отправка сообщений. Так-же была проверка на возможность редактирования информации о пользователе, самим пользователем.

В ходе тестирования, была создана таблица, тестов основных функций, с помощью которой можно было своевременно находить бреши в проекте и быстро их устронять. Так-же тестирование проходило в двух временных промежутках, а именно при разработке проекта, и после окончательного его завершения. В первом случае, проверялся функционал отдельных частей проекта, во втором проверялась работоспособность всех модулей и их взаимосвязь.

Таблица 4 - тестирование продукта

Выполняемое действие	Ожидаемый результат	Результат
Запуск приложения	Отображение начальной страницы приложения	Успешно
Регистрация	Создание профиля в системе	Успешно
Вход	Вход в систему, под созданным паролем и указанной почтой	Успешно
Редактирование профиля	Изменение имени, на то, которое не занято и смена фотографии профиля	Успешно
Просмотр страниц	Просмотр страницы своей и других пользователей и в зависимости от того, страница друга, заблокированного человека или своей страницы, отображался разный интерфейс	Успешно

Работа с другим пользователем	Добавление в друзья, блокировка, отписка и подписка на него	Успешно
Отправление и просмотр сообщений	Отправление сообщений, тем людям которые не заблокированны пользователем. И просмотр истории чатов	Успешно
Списки людей	Отображение 5 списков, все люди, друзья, подписчики, подписки, заблокированные	Успешно
Ограничение видимости таблиц	Ограничение видимости таблиц от обычных пользователей	Успешно
Ограничение видимости списков людей	Ограничение видимости списков всех пользователей от незалогиненых людей	Успешно
Просмотр и выгрузка отчетов	Просмотр всех таблиц и выгрузка отчетов, только для администраторов	Успешно

Результаты, полученные в ходе тестирования разработанного приложения, позволяют сделать заключение в том, что разработанная программа соответствует требованиям технического задания.

Заключение

В ходе выполнения курсового проекта была разработана распределенная информационная система для мессенджера с использованием технологии ASP.NET Core MVC, созданы концептуальная, логическая и физическая модели данных. В результате работы создано приложение в котором пользователи могут общаться и добавлять друг друга в друзья.

Так-же, были получены навыки в разработке проектов на ASP.NET Core MVC, что позволит сократить время на разработку некоторых проектов или для создания прототипов.

Подводя итоги, можно считать, что разработанное приложение соответствует требованиям технического задания.

Список литературы

1) Столбовский, Д. Н. Разработка Web-приложений ASP.NET с использованием Visual Studio .NET : учебное пособие / Д. Н. Столбовский. - 3-е изд. - Москва, Саратов.

2) Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2020. - 375 с.

3) Фримен Адам. ASP. NET MVC 5 с примерами на C# 5.0 для профессионалов : Вильямс, 2018, 736с.

4) Brian L Gorman. Practical Entity Framework: Database Access for Enterprise Applications. - Apress, 2020 - 433pp.

5) Эспозито Дино Разработка современных веб-приложений. Анализ предметных областей и технологий. — Вильямс, 2017, 464с.

Приложение 1

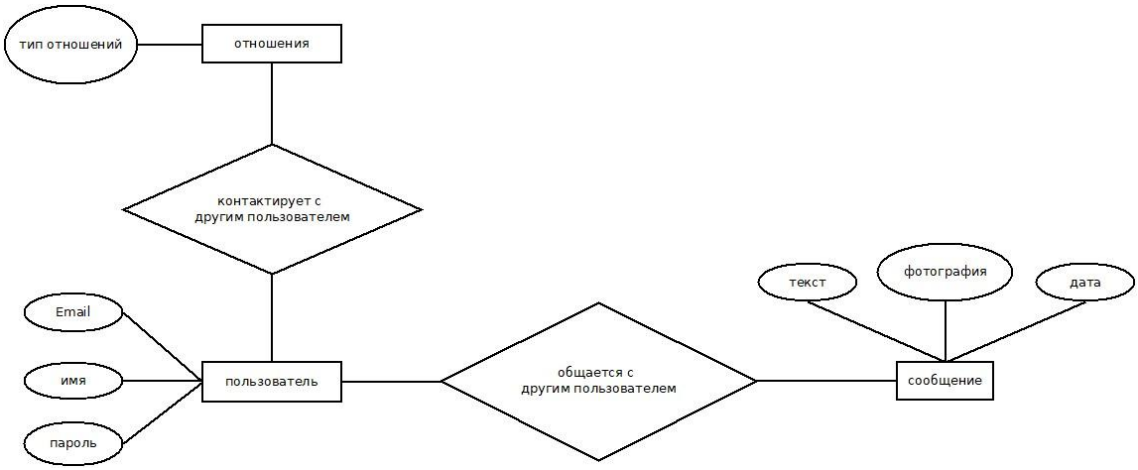

Рисунок 1 - Концептуальная модель данных

Рисунок 2 - Логическая модель данных.

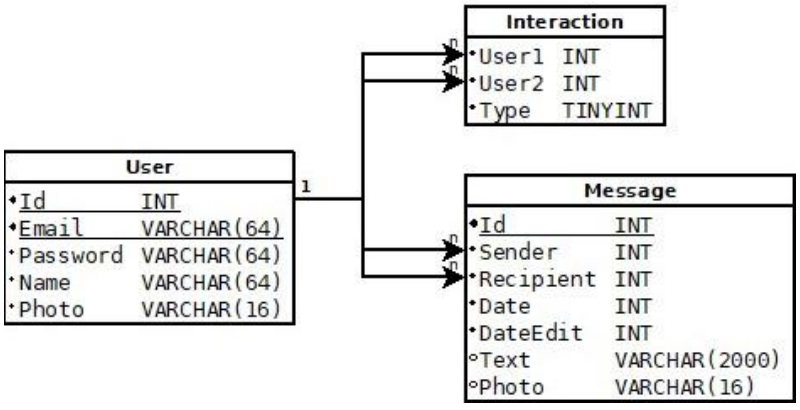

Рисунок 3 - Физическая модель данных

Приложение 2

Для подробного ознакомления с данным приложением можно использовать ссылку на репозиторий данного проекта: https://github.com/m9agrest/Messenger

Приложение 3

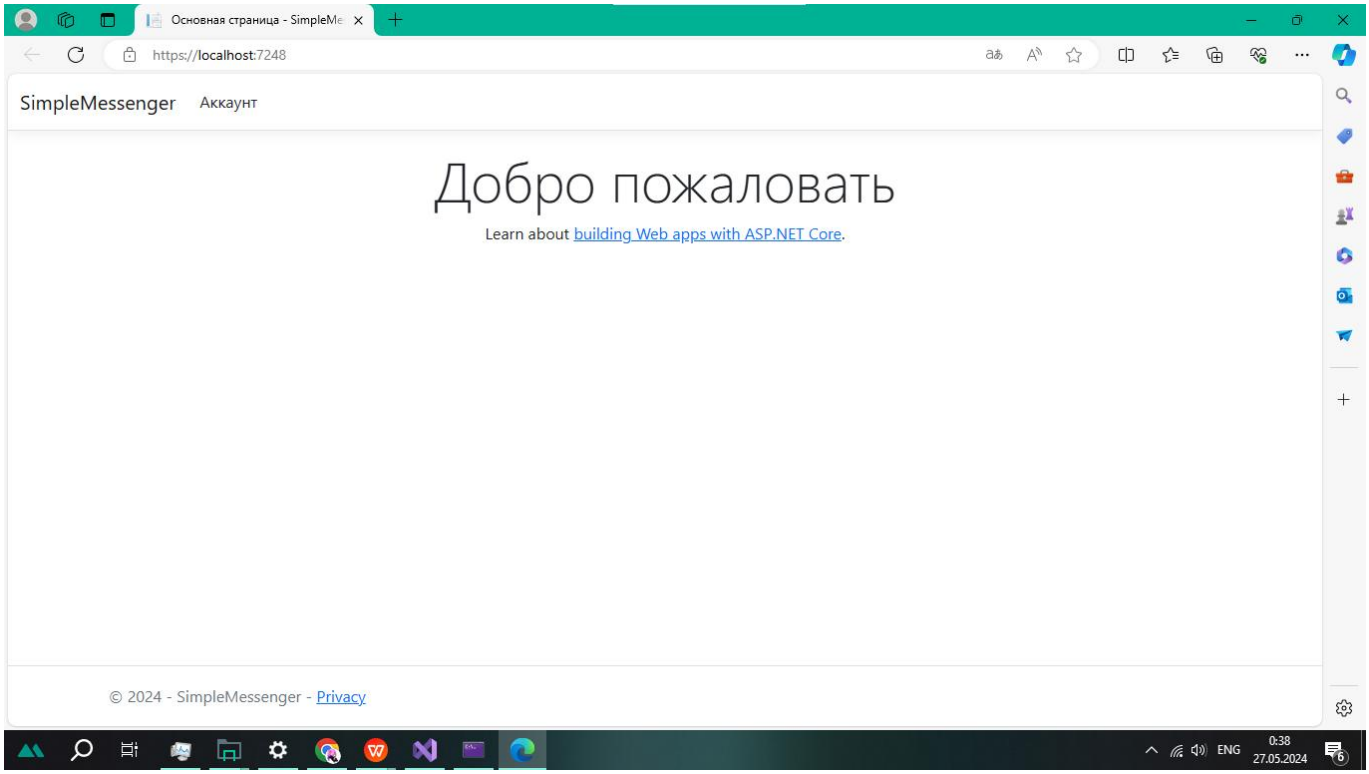

Рисунок 1 - начальное окно

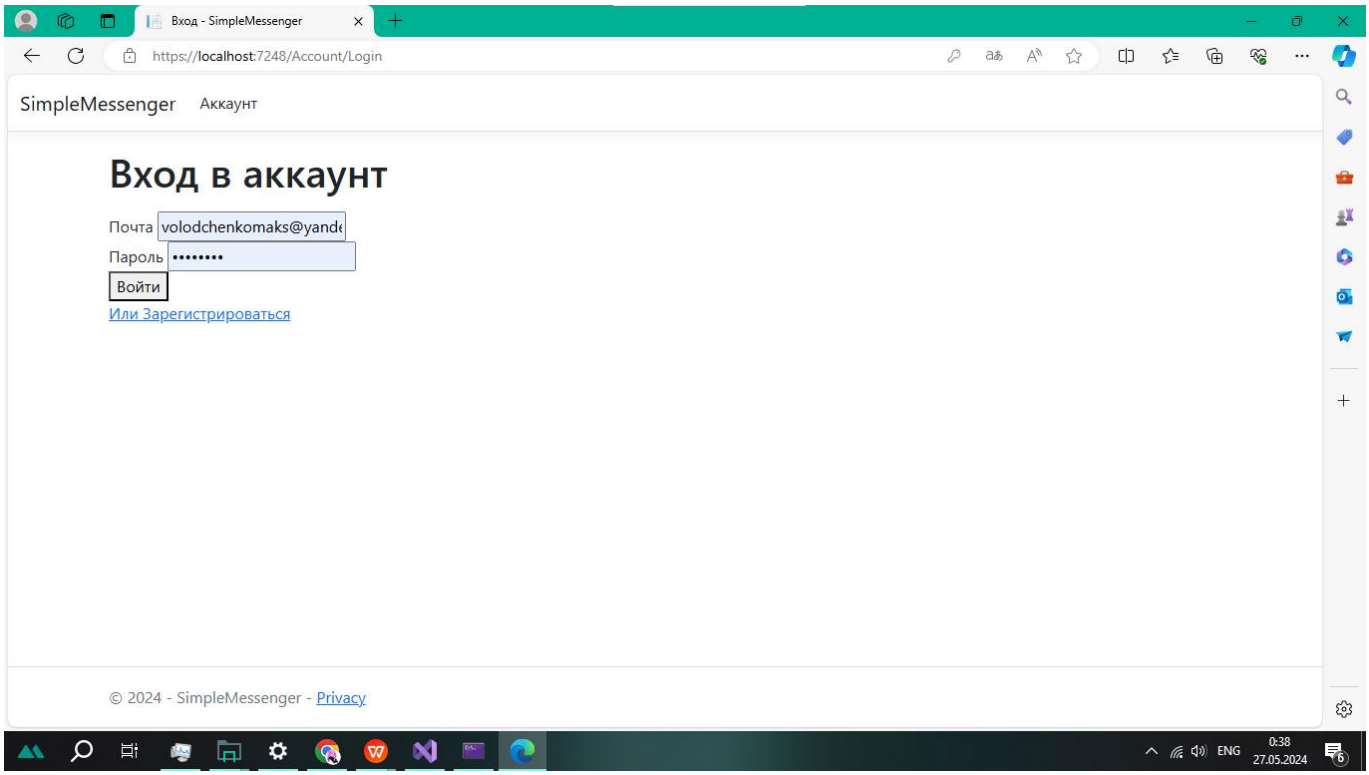

Рисунок 2 - вход в аккаунт

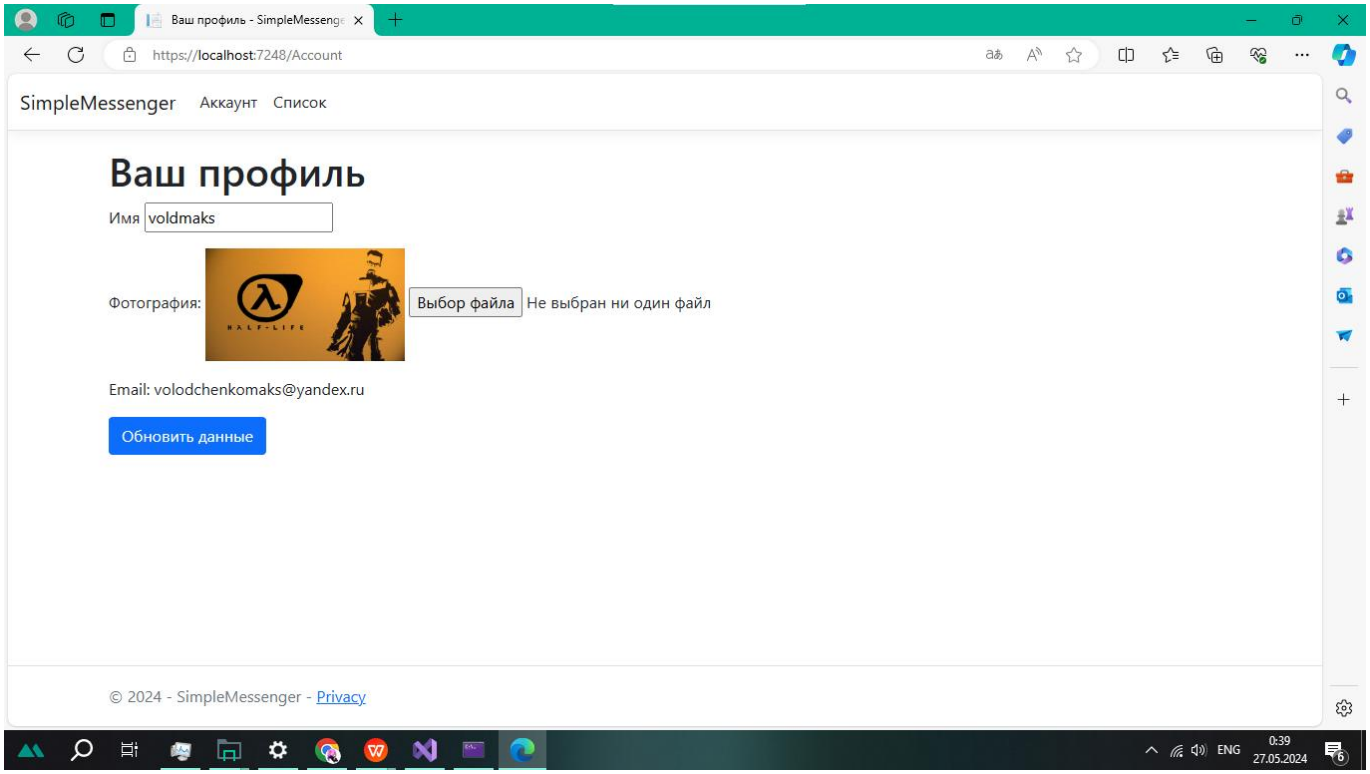

Рисунок 3 - профиль пользователя

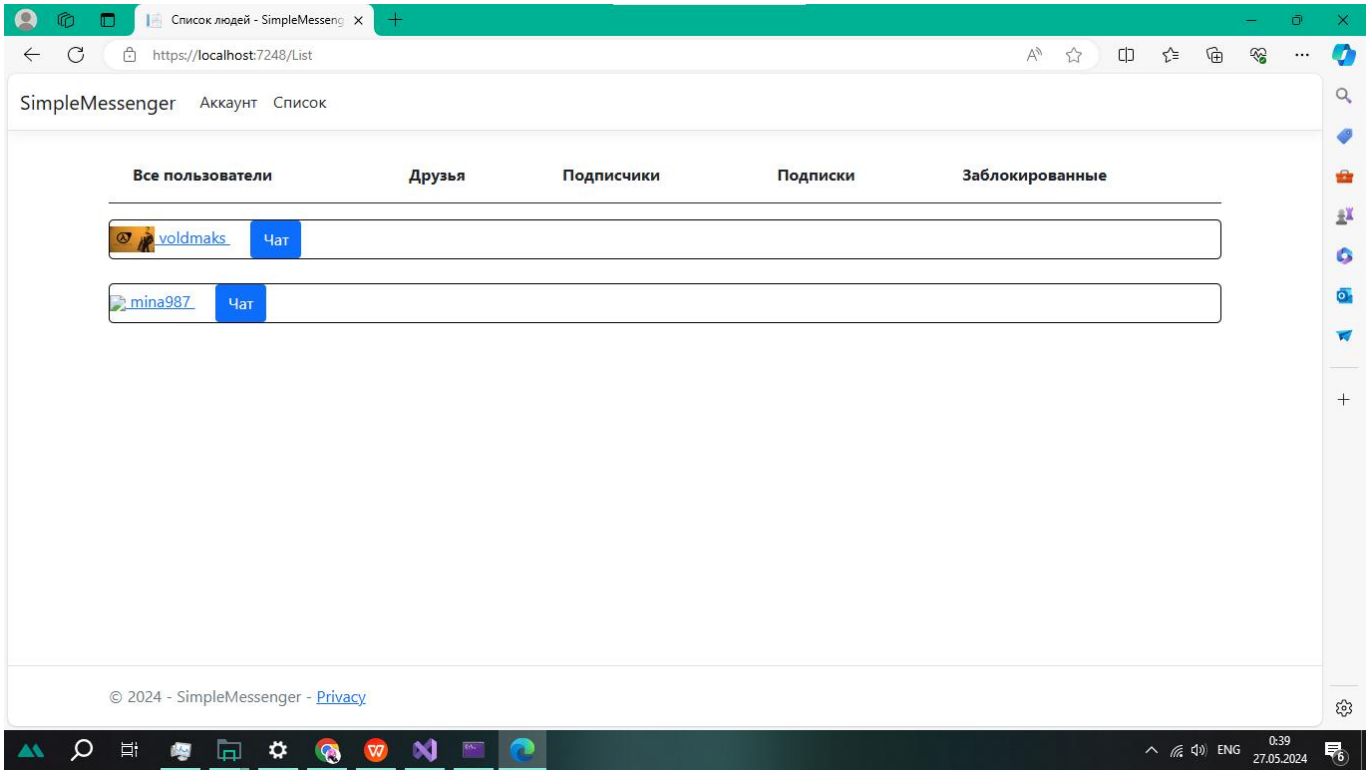

Рисунок 4 - спико всех пользователей программы

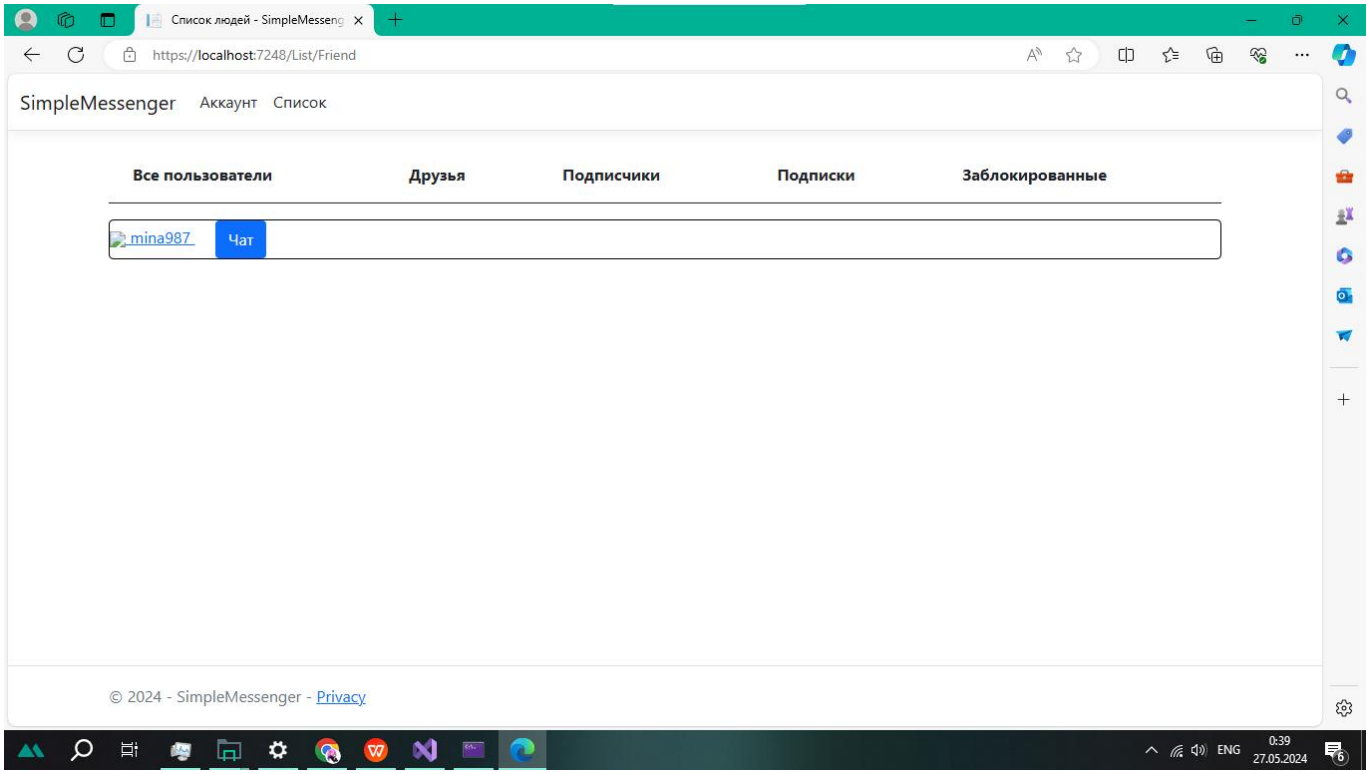

Рисунок 5 - список друзей

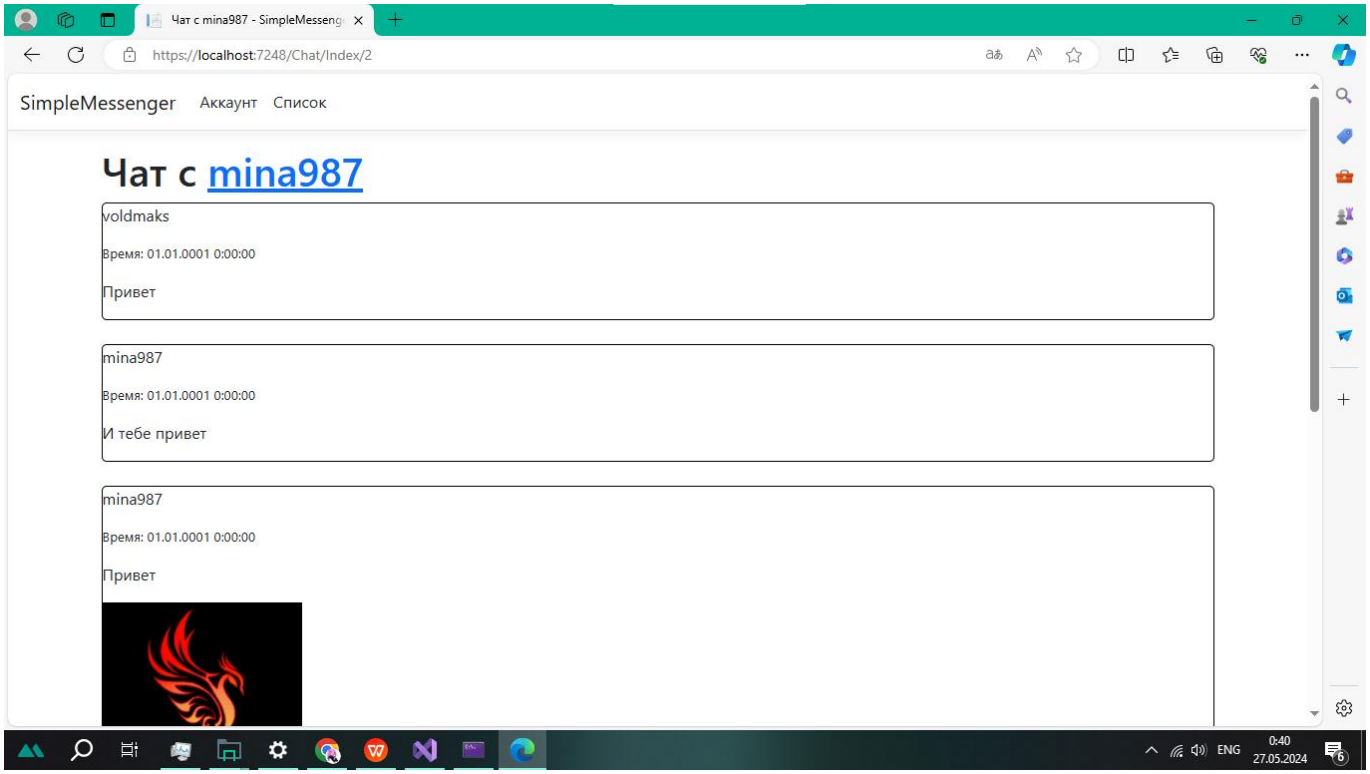

Рисунок 6 - чат с пользователем

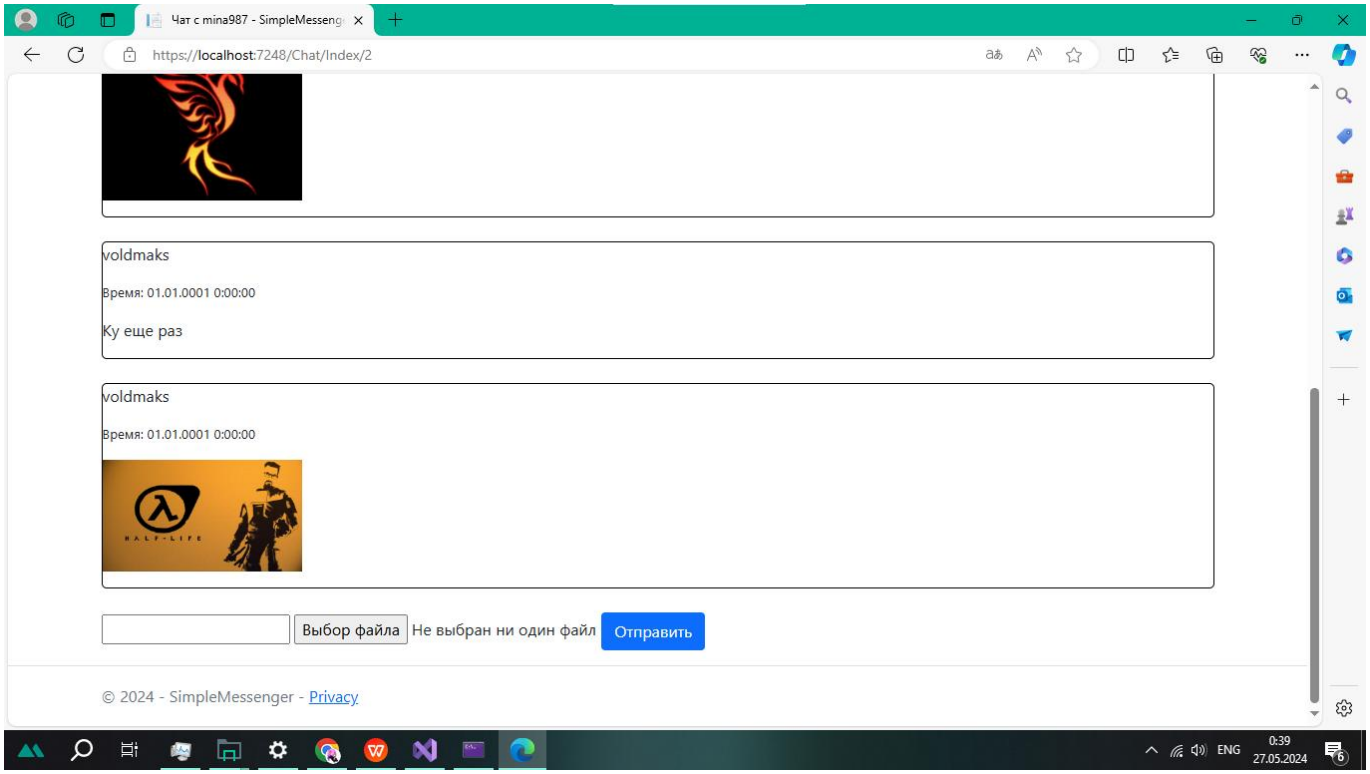

Рисунок 7 - чат с пользователем

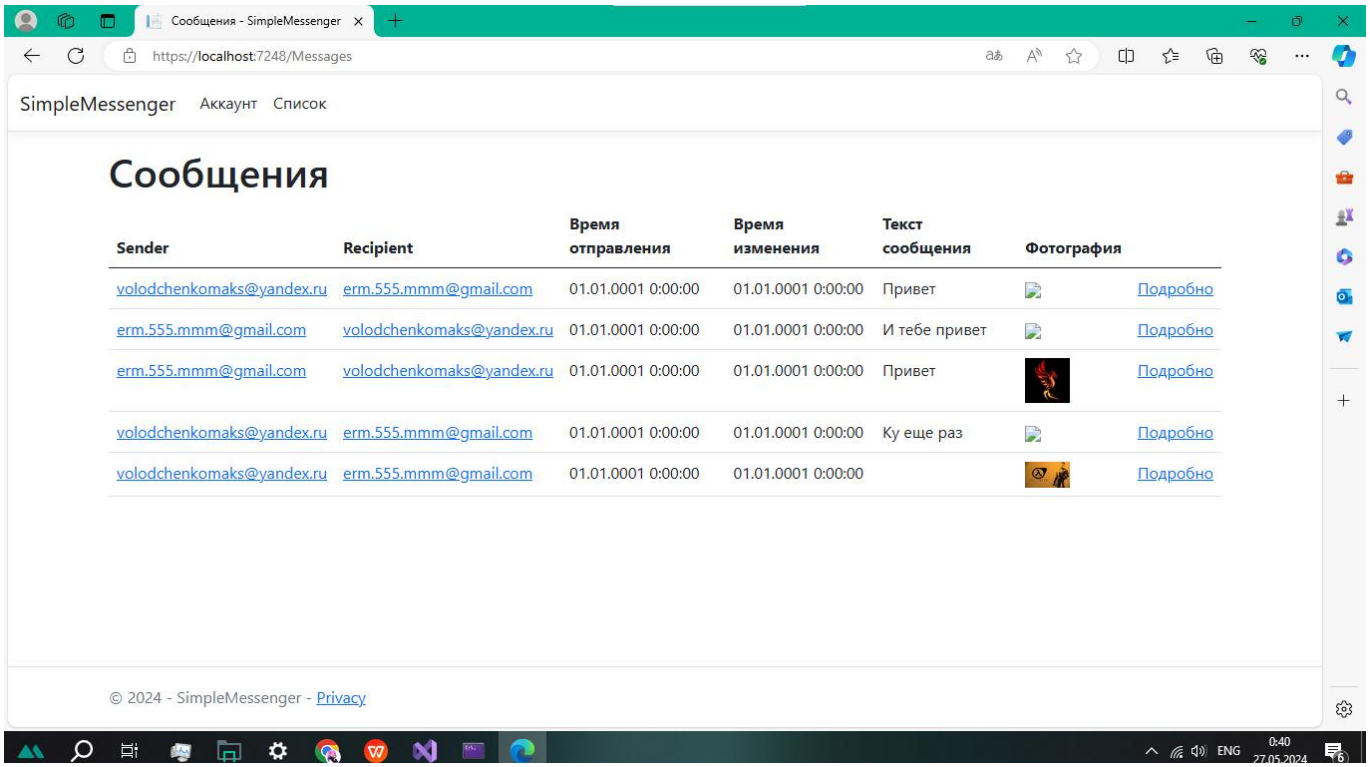

Рисунок 8 - панель админа

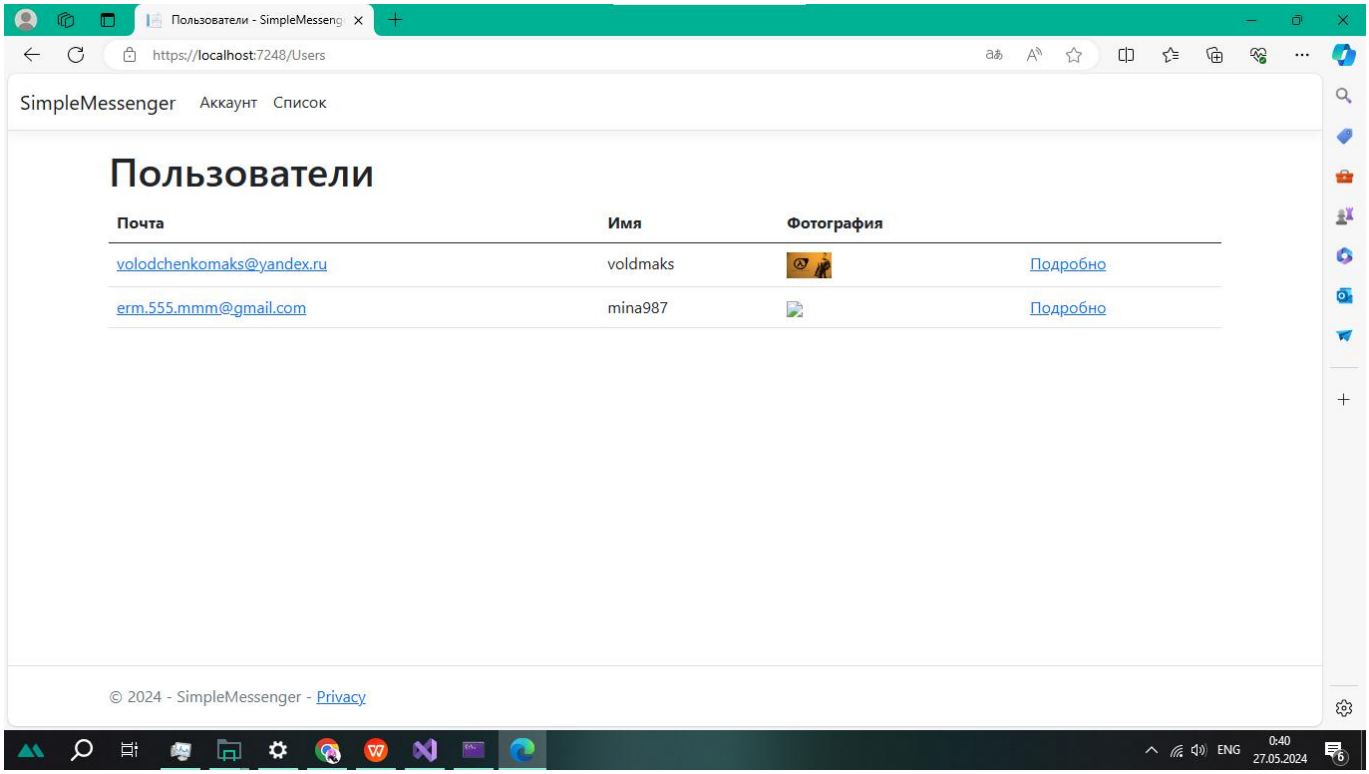

Рисунок 9 - панель админа

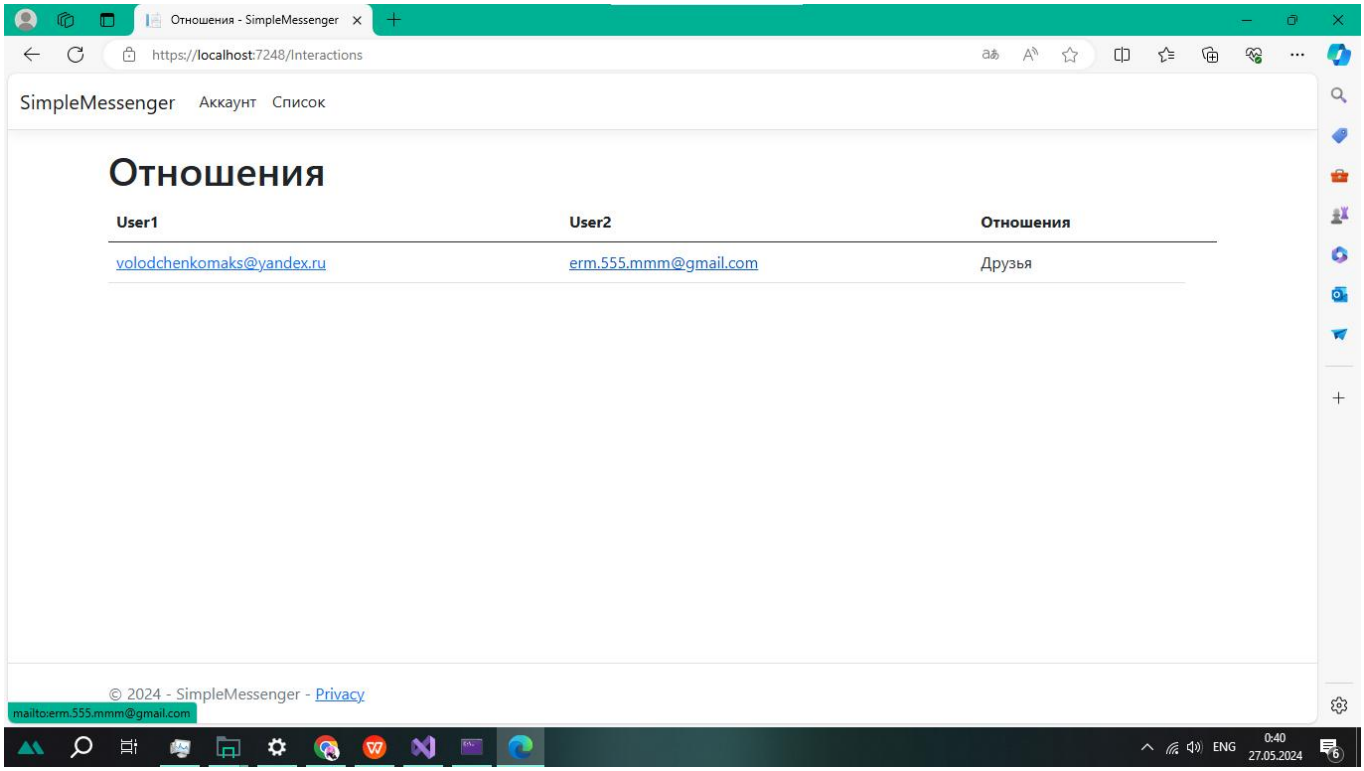

Рисунок 10 - панель админа