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

Место для задания

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](#_Toc122914454)

[1 Анализ технического задания](#_Toc122914455)

[2 Разработка моделей данных](#_Toc122914456)

[3 Проектирование работы системы](#_Toc122914457)

[4 Разработка и реализация системы](#_Toc122914458)

[5 Тестирование системы](#_Toc122914459)

[Заключение](#_Toc122914461)

[Список литературы](#_Toc122914462)

[Приложение 1. Модели данных](#_Toc122914463)

[Приложение 2. Текст программы](#_Toc122914463)

[Приложение 2. Снимки окон программы](#_Toc122914463)

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.

2 Разработка моделей данных

Для проекта было разработано несколько версий моделей баз данных, которые видо изменялись и упрощались.

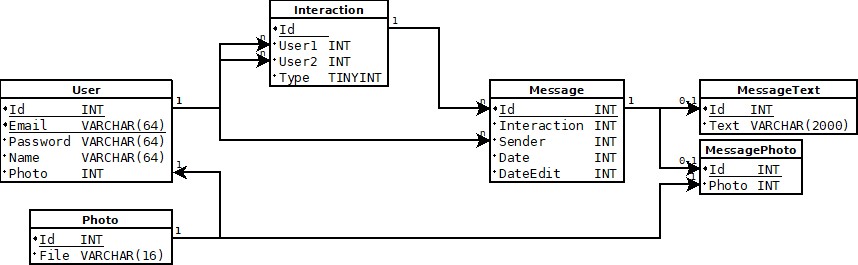


Рисунок 1 - первоночальный вид модели

В рисунке 1, продемонстрированна структура хранения данных, и данная модель будет гораздо правильной. Таблицы не допускают null полей. Так-же

таблицы Interaction, Message, Photo, MessagePhoto имеют одинаковый размер на одну запись, что во многих СУБД оптимизирует выборку. Но из-за некоторых сложностей в реализации в ASP.NET, и из-за отсутсвия опыта работы с моделями баз данных в данной среде, схему пришлось упрощать.

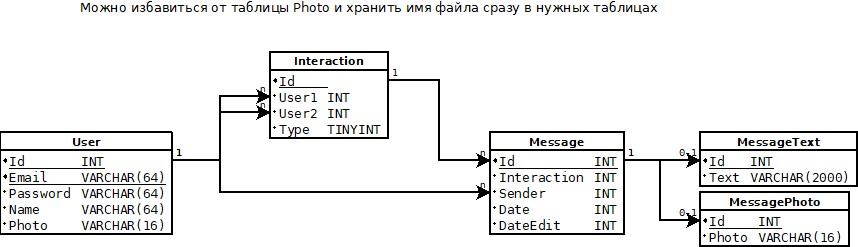


Рисунок 2 - упрощенная база данных.

В рисунке 2, уже можно заметить что была убрана таблица Photo, и данные из нее, уже напрямую храняться в MessagePhoto и User. Данное решение было принято, чтобы не гонять запросы в базу данных, для объеденения таблиц, для выборок данных. Обычно такие объеденения делают специально написанные функции на SQL языках, и из-за отсутсвия возможности и навыков работы в данном фреймворке.

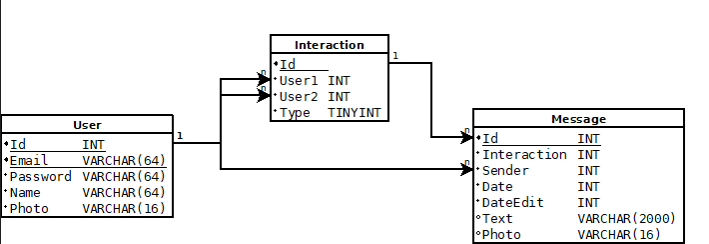


Рисунок 3 - упрощенная таблица

В рисунке 3, так-же была упрощенна таблица Message, из-за причин описанных выше. Теперь в ней хранится Text и Photo, это записи которые могут допускать null.

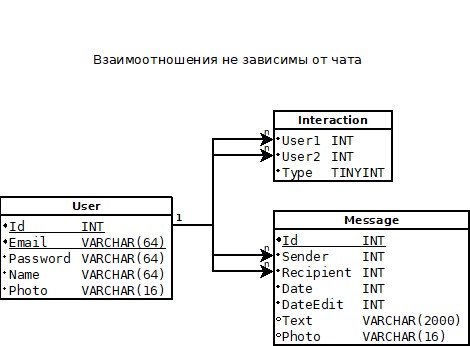


Рисунок 4 - Упрощенная база данных.

В рисунке 4, так-же пришлось упростить схему, теперь в отображаеммых моделях, сразу будет видно кто и кому отправлял письмо, в предыдущей же версии, мы могли бы видеть только id отношений, и кто отправил сообщение.

using System.ComponentModel.DataAnnotations;

namespace SimpleMessenger.Models

{

public class User

{

[Key]

public int Id { get; set; }

[Required(ErrorMessage = "Введите почту пользователя")]

[DataType(DataType.EmailAddress)]

[Display(Name = "Почта")]

public string Email { get; set; }

[Required(ErrorMessage = "Введите пароль пользователя")]

[DataType(DataType.Password)]

[MaxLength(32), MinLength(8)]

[Display(Name = "Пароль")]

public string Password { get; set; }

[Required(ErrorMessage = "Введите имя пользователя")]

[MaxLength(32), MinLength(3)]

[Display(Name = "Имя")]

public string Name { get; set; }

[Display(Name = "Фотография")]

public string? Photo { get; set; } // URL фотографии пользователя

}

}

using System.ComponentModel.DataAnnotations.Schema;

using System.ComponentModel.DataAnnotations;

using System.Xml.Linq;

namespace SimpleMessenger.Models

{

public class Message

{

[Key]

public int Id { get; set; }

[Column(Order = 1)]

[Required(ErrorMessage = "Укажите отправителя")]

[Display(Name = "Отправитель")]

public int SenderId { get; set; }

[ForeignKey(nameof(SenderId))]

public User Sender { get; set; }

[Column(Order = 2)]

[Required(ErrorMessage = "Укажите получателя")]

[Display(Name = "Получатель")]

public int RecipientId { get; set; }

[ForeignKey(nameof(RecipientId))]

public User Recipient { get; set; }

[DataType(DataType.DateTime)]

[Display(Name = "Время отправления")]

public DateTime Date { get; set; }

[DataType(DataType.DateTime)]

[Display(Name = "Время изменения")]

public DateTime DateEdit { get; set; }

[MaxLength(1000)]

[DataType(DataType.MultilineText)]

[Display(Name = "Текст сообщения")]

public string? Text { get; set; }

[Display(Name = "Фотография")]

public string? Photo { get; set; } // URL фотографии сообщения

}

}

using System.ComponentModel.DataAnnotations.Schema;

using System.ComponentModel.DataAnnotations;

using System.Xml.Linq;

namespace SimpleMessenger.Models

{

public class Interaction

{

[Key]

[Column(Order = 1)]

[Required(ErrorMessage = "Укажите пользователя 1")]

[Display(Name = "Пользователь 1")]

public int User1Id { get; set; }

[ForeignKey(nameof(User1Id))]

public User User1 { get; set; }

[Key]

[Column(Order = 2)]

[Required(ErrorMessage = "Укажите пользователя 2")]

[Display(Name = "Пользователь 2")]

public int User2Id { get; set; }

[ForeignKey(nameof(User2Id))]

public User User2 { get; set; }

[Required(ErrorMessage = "Укажите тип отношений")]

[Display(Name = "Отношения")]

[EnumDataType(typeof(TypeInteraction))]

public TypeInteraction Type { get; set; }

}

public enum TypeInteraction

{

[Display(Name = "Обоюдный бан")]

Enemy = -3,

[Display(Name = "Пользователь 1 заблокировал 2")]

Blocked = -2,

[Display(Name = "Пользователь 2 заблокировал 1")]

Blocker = -1,

[Display(Name = "Нейтральные")]

None = 0,

[Display(Name = "Пользователь 1 подписан на 2")]

Subscriber = 1,

[Display(Name = "Пользователь 2 подписан на 1")]

Subscription = 2,

[Display(Name = "Друзья")]

Friend = 3,

}

}

3 Проектирование работы системы

В данном разделе содержится информация о проектировании работы ИС, разработанной в текущей курсовой работе. В первую очередь было необходимо определить основные компоненты системы и их взаимодействие.

При проектировании работы системы нужно уделить особое внимание архитектуре приложения и его функциональности. В соответствии с техническим заданием, система должна быть создана с использованием технологии ASP.NET Core MVC и базs данных SQL Server.

Первоначальным этапом проектирования является разработка моделей данных. Для хранения отношений, самих пользователей и для хранения сообщений.

Далее следует создание CRUD операций для администратора и пользователя системы. Это позволит администратору управлять данными, а пользователю работать с другими пользователями В рамках создания CRUD функционала важно предусмотреть возможности добавления, просмотра, обновления и удаления записей.

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 в AppDbContext:

public class AppDbContext : DbContext

{

public AppDbContext(DbContextOptions<AppDbContext> options) : base(options)

{

}

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

modelBuilder.Entity<Interaction>().HasKey(e => new { e.User1Id, e.User2Id });

modelBuilder.Entity<Interaction>().HasQueryFilter(filter => filter.User1Id < filter.User2Id);

modelBuilder.Entity<User>().HasIndex(i => i.Email).IsUnique();

}

public DbSet<User> Users { get; set; }

public DbSet<Interaction> Interactions { get; set; }

public DbSet<Message> Messages { get; set; }

}

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.

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.

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.

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.

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 С# 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

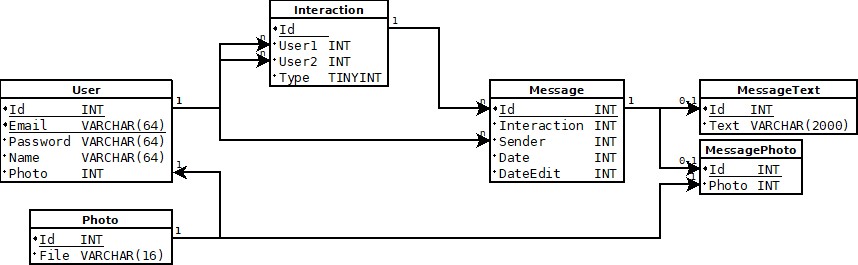


Рисунок 1 - первоночальный вид модели

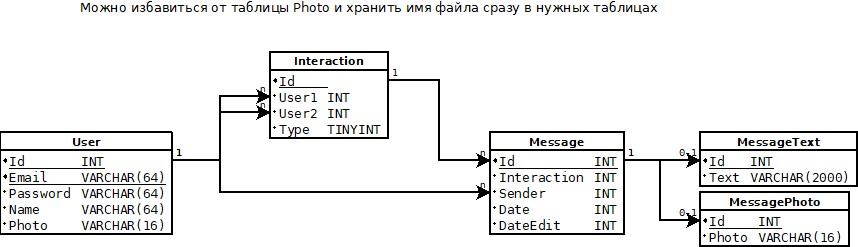


Рисунок 2 - упрощенная база данных.

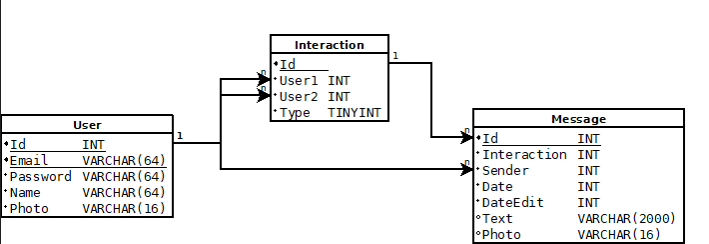


Рисунок 3 - упрощенная таблица

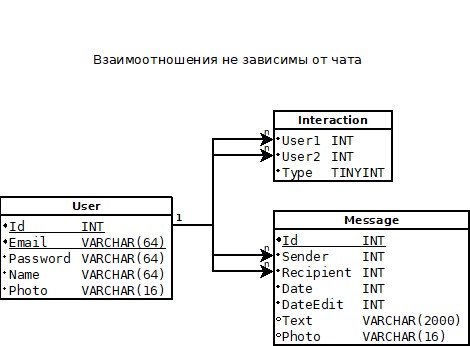


Рисунок 4 -Итоговый вид базы данных.

Приложение 2

Приложение 3