Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Рязанский государственный радиотехнический университет имени В.Ф. Уткина»

Кафедра АСУ

Дисциплина: Учебная практика

Отчёт о практической работе

«Проектирование бизнес-процесса»

Выполнил: ст. гр. 135

Бардин М.С

Проверил:

Костиков М.Г.

Рязань 2022

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Рязанский государственный радиотехнический университет

имени В.Ф. Уткина»

«УТВЕРЖДАЮ»

Заведующий кафедрой АСУ

\_\_\_\_\_\_\_\_\_\_\_\_ С.И. Холопов

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 20*22* г.

ЗАДАНИЕ НА УЧЕБНУЮ ПРАКТИКУ БАКАЛАВРА

Обучающемуся *Бардину Максиму Сергеевичу*  группа *135*

Направление подготовки 09.03.02 – Информационные системы и технологии

Направленность Информационные системы и технологии

Трудоемкость практики 72 часа

Научный руководитель *Костиков Максим Геннадьевич* *старший преподаватель кафедры АСУ Рязанского государственного радиотехнического университета имени В. Ф. Уткина*

Тема практики: Создание ER моделей и программ для работы с БД.

Цель практики: Получение навыков создания ER моделей и программ для работы с Entity Framework Core.

Задачи практики

1. Разработать ER модель заказа товара из магазина электроники. ­­­­­­­­

2. Осуществить программную реализацию разработанной модели.

Планируемые результаты

Получение навыков создания ER моделей и программ для работы с Entity Framework Core.

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Костиков М. Г.

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Бардин М.С.

Оглавление

[Описание предметной области 4](#_Toc122355853)

[Описание бизнес-логики. 6](#_Toc122355854)

[Описание работы сервиса. 9](#_Toc122355855)

[Use case 10](#_Toc122355856)

[ER модель 11](#_Toc122355857)

[Физическая модель 14](#_Toc122355858)

[Реализация модели. 15](#_Toc122355859)

[Заключение 22](#_Toc122355860)

# Описание предметной области

В последние годы стало особенно заметно проявление интереса широкого круга людей к занятиям различными видами спорта, двигательной активности для отдыха и восстановления сил, для обеспечения хорошей спортивной формы и состояния здоровья. Современный фитнес - является символом хорошего «качества» жизни.

Главной задачей фитнес-центра является - получение прибыли за счет продажи оказания спортивно-оздоровительных услуг, обеспечение комфортной среды для поддержки здоровья и улучшения физической формы.

Важный фактор, влияющий на развитие фитнес-центра - целевая аудитория, то есть тот круг клиентов, фактически или потенциально являющихся заинтересованными в получении данного вида услуг. Именно направленность на целевую аудиторию, интерес в увеличении прибыли и качестве оказании услуг обеспечивает эффективную работу фитнес-центра.

В самом фитнес клубе ведутся списки клиентов с указанием их личных данных и тренеров, с которым они индивидуально занимаются. Помимо ведения списков клиентов ведется учёт сотрудников, и услуг, которые они предоставляют, также расписания, по которым они работают, и благодаря которым можно отслеживать загруженности каждого сотрудника. Обслуживание клиентов ведется по абонементной системе, то есть клиенту требуется приобрести клубную карту на срок, требуемый клиенту, после чего за ним будет закреплён специалист, с которым будет обговорено расписание клиента.

В основу деятельности фитнес-центра входят следующие принципы:

1. Индивидуальный подход к каждому клиенту;
2. Использование современных методик и качественного оборудования;
3. Сотрудники фитнес-центра имеют высокий профессиональный уровень и подготовку.

Широкий выбор направлений предоставляет возможность клиенту подобрать именно ту фитнес-программу, подходящую конкретно ему, исходя из пола, возраста, состояния здоровья и общей физической подготовки клиента. Сотрудникам фитнес-центра необходимо делать акцент на индивидуальный подход к каждому, предлагая ему рассмотреть весь спектр предоставляемых услуг.

Основные бизнес-процессы, которые можно выделить, это: продажа фитнес-услуг, регистрация клиентов и увеличению клиентской базы. Под регистрацией клиентов понимается часть учета, когда необходимо занести нового клиента в список и вести дальнейшую работу с ним.

Вследствие повышения интереса людей к здоровому образу жизни, распространения фитнес программ, разнообразия всевозможных направлений в области фитнеса, наблюдается постоянное увеличение числа клиентов. Что приводит к проблемам, связанным с учетом и управлением. Наибольший интерес для фитнес-клуба представляют постоянные клиенты, то есть те клиенты, которые покупают не разовые тренировки, а абонементы на длительные сроки. Это обусловлено также улучшением репутации самого фитнес-центра. Таким образом, необходимы также действия, направленные на их удержание (например, скидки или бонусы).

В данном случае наибольший интерес представляет работа с клиентами, как один из основных бизнес-процессов центра, который требует оптимизации. Также этот процесс является одним из наиболее важных, так как именно от него зависит не только репутация центра, но и продуктивность его работы.

# Описание бизнес-логики.

Пользователь вводит логин или пароль, с помощью базы данных мы проверяем существует ли такой логин и пароль. Если существует, то пользователь входит в аккаунт, иначе идет переадресация клиента к регистрации. Пользователь регистрируется, следом данные заносятся в базу данных.

Далее проверяется: есть ли у пользователя подписка на сервис или если она имеется, то не истек ли срок. Если подписка имеется то, спрашиваем у пользователя хочет он внести новые продукты в аллергический список. Если подписки нет, переадресуем на страницу выбора подписки. Пользователь выбирает желаемый тариф и получает доступ к сервису.  
 Указав аллергические продукты информационная система составляет запрос на добавление в БД новых продуктов. Запрос выполняется внутри БД, далее уведомляем пользователя об успешном добавлении продукта в карту аллергика.   
 Следующем шагом является добавление новой цели – того результата, который хочет клиент. Информационная система также создает запрос на добавление новой цели и привязывает ее к клиенту.  
 После того как клиент внес все добавления (изменения), ментор, прикрепленный к клиенту, указывает рекомендованные продукты для клиента и его диеты. Информационная система создает запрос на добавление новых продуктов в рекомендованный список, привязанный к клиенту.

Выше пересказанный алгоритм представлен на рисунках 1 и 2.

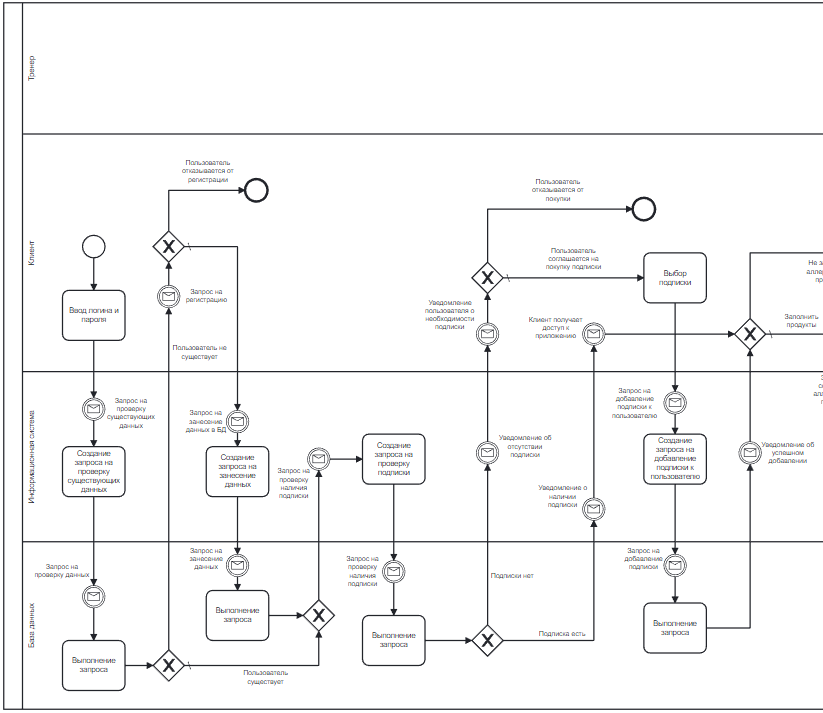


Рисунок 1 – Первая часть BPMN модели

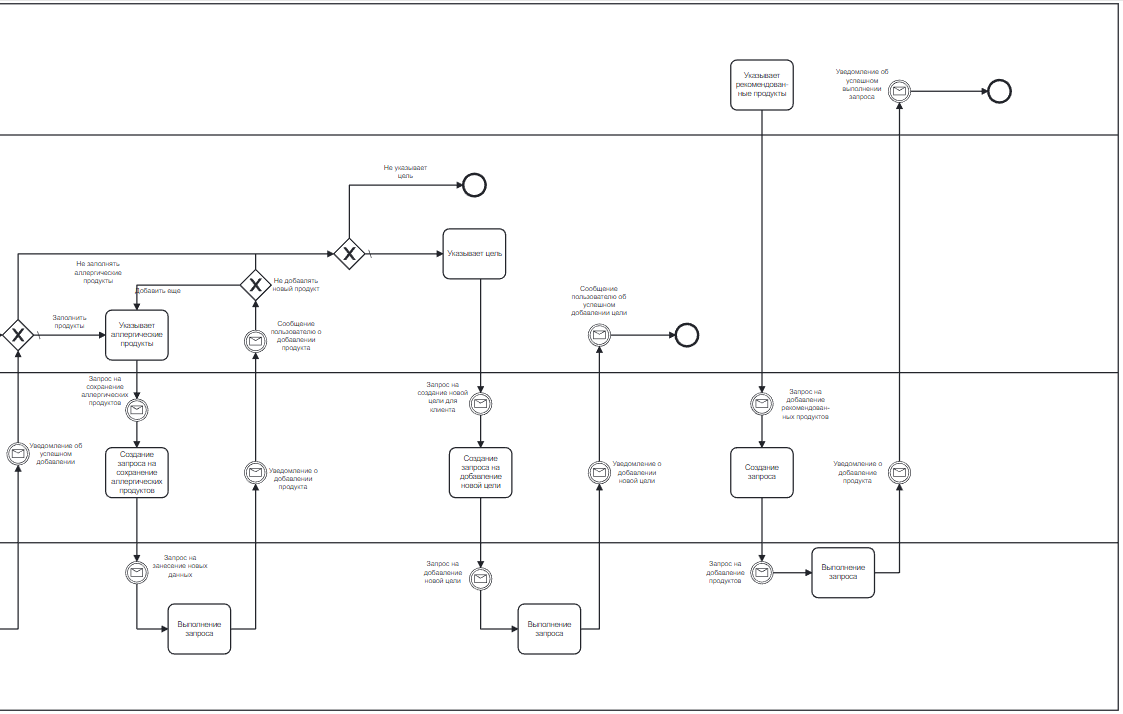


Рисунок 2 – Вторая часть BPMN модели

# Описание работы сервиса.

В самом начале клиент указывает свои начальные данные (массу тела, продукты, на которые у пользователя аллергия) и желаемые результаты (массу тела).

Ментор (тренер) – специально обученный человек, следящий за определенной группой клиентов и направляющий их на достижение указанной цели.

Задачи тренера:

1. Составить рацион питания.

2. Определить предполагаемую дату достижения цели.

3. Составить план тренировок.

4. Поддерживать связь с пользователем.

5. Отвечать на задаваемые клиентом вопросы.

За взаимодействие клиента и ментора отвечает специально разработанный чат, к которому есть постоянный доступ.

Вышеописанные действия изображены на рисунке 3.

# Use case

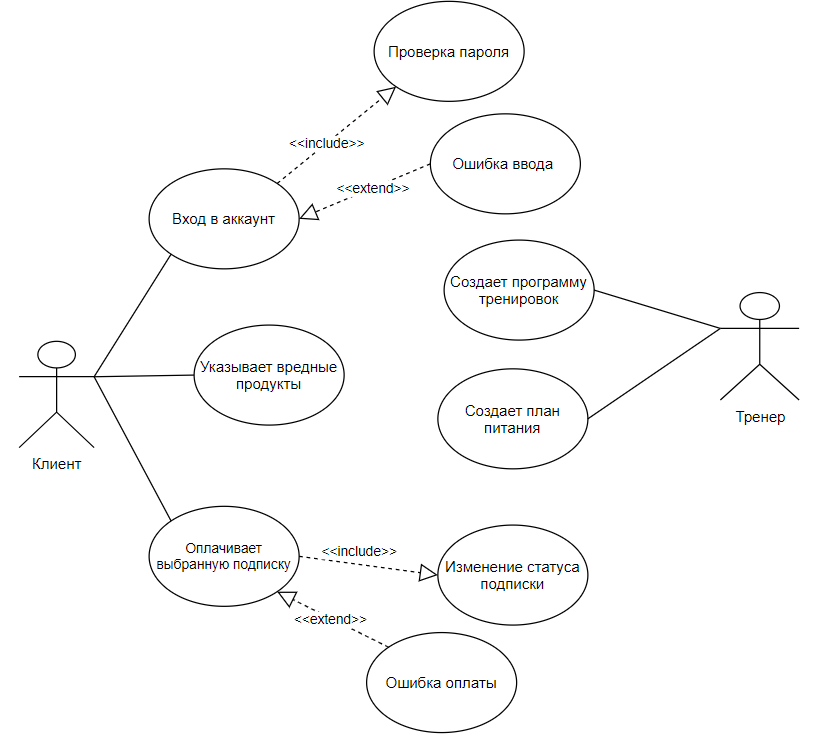


Рисунок 3 – Use case

# ER модель

Для разработки информационной системы фитнес приложения была создана ER диаграмма, которая показывает предполагаемую для программной реализации структуру базы данных.

Описание таблиц

Таблица №1. Информация о клиенте

Данная таблица предполагает хранение ФИО клиента, его номер телефона, которые предоставляется его личному ментору, ID ментора для ссылки на ментора и пароль для входа в приложение.

Таблица №2. Информация о цели клиента.

Данная таблица предполагает хранения информации о поставленной клиентом целью, то есть его начальный и конечный вес. Остальные поля таблицы заполняет ментор, то есть предполагаемый срок выполнения, количество разрешенных БЖУ (Белки Жиры Углеводы). Также в таблице содержится ссылка на Id Клиента.

Таблица №3. Информация о подписке.

Данная таблица содержит информацию о приобретенной подписки клиентом, то есть ее наименование, дата начала, дата окончания, стоимость. Также в таблице содержится ссылка на клиента.

Таблица №4. Информация о менторе.

Данная таблица содержит основную информацию о менторе, то есть его ФИО и должность.

Таблица №5. Карта аллергика.

Данная таблица является связывающим звеном, чтобы избежать связи N:N. В таблице содержится Id клиента, Id продукта.

Таблица №6. Каталог разрешенных продуктов.

Данная таблица является связывающим звеном, чтобы избежать связи N:N. В таблице содержится Id цели, Id продукта. Таблица является списком тех продуктов, которые можно употреблять одному клиенту

Таблица №7. Каталог продуктов.

Данная таблица является описанием продукта, который в дальнейшем будет распределен на разрешенный или запрещенный. Таблица содержит такие поля как: название продукта, количество белков, количество жиров, количество углеводов в продукте.

Таблица №8. Каталог графика занятий.

Данная таблица содержит в себе информацию об общем количестве тренировок, количестве силовых и кардио тренировок. Также в таблице имеется ссылка на клиента.

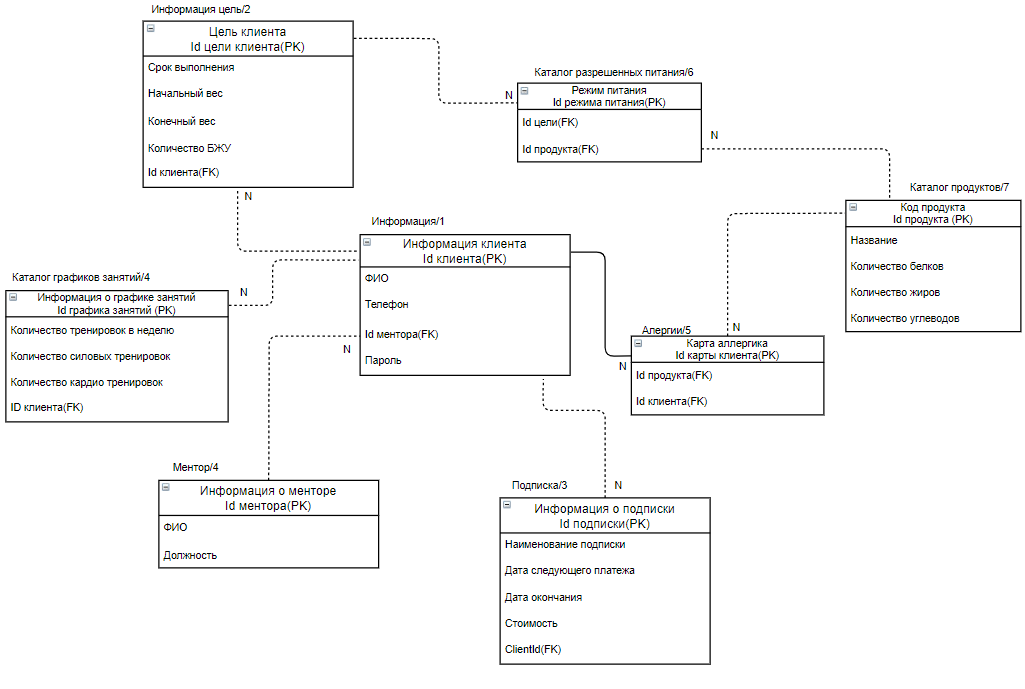


Рисунок 3- ER модель

# Физическая модель

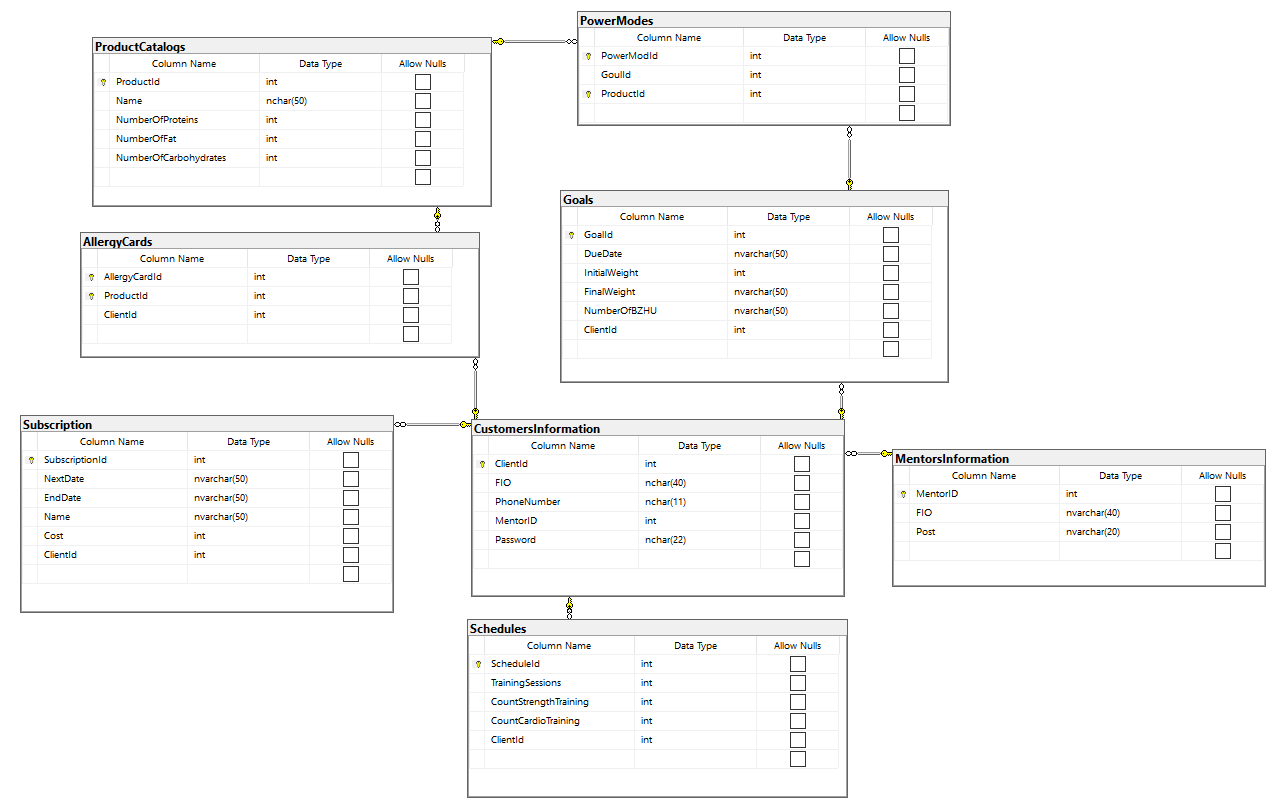


Рисунок 4 – Физическая модель

# Реализация модели.

На основе ER модели была создана соответствующая ей база данных. Для работы с этой базой была создана программа, включающая в себя:

1. Json-файл конфигурации для соединения с базой данных.

{

"ConnectionStrings": {

"DefaultConnection": "Server=localhost\\SQLEXPRESS;Database=FitnessApp;Trusted\_Connection=True;"

}

}

2. Класс Client

public class Client

{

[Key]

[Column("ClientId")]

public int Id { get; set; }

public string FIO { get; set; }

public string PhoneNumber { get; set; }

public int MentorID { get; set; }

public string Password { get; set; }

public Mentor Mentor { get; set; }

public LinkedList<Goals> Goals { get; set; } = new LinkedList<Goals>();

public LinkedList<AllergyCards> AllergyCards { get; set; } = new LinkedList<AllergyCards>();

public LinkedList<Subscription> Subscriptions { get; set; } = new LinkedList<Subscription>();

public LinkedList<Schedules> Schedules { get; set; } = new LinkedList<Schedules>();

}

3. Класс Mentor

public class Mentor

{

[Key]

[Column("MentorId")]

public int Id { get; set; }

public string FIO { get; set; }

public string Post { get; set; }

public LinkedList<Client> Clients { get; set; } = new LinkedList<Client>();

}

4. Класс AllergyCards

public class AllergyCards

{

[Column("AllergyCardId")]

public int Id { get; set; }

[Column("ProductId")]

public int Product { get; set; }

[Column("ClientId")]

public int Clients { get; set; }

public Client Client { get; set; }

public ProductCatalogs ProductCatalogs { get; set; }

}

5. Класс Goals

public class Goals

{

[Key]

[Column("GoalId")]

public int Id { get; set; }

public string DueDate { get; set; }

public int InitialWeight { get; set; }

public int FinalWeight { get; set; }

public string NumberOfBZHU { get; set; }

public int ClientId { get; set; }

public Client Client { get; set; }

public LinkedList<PowerModes> PowerModes { get; set; } = new LinkedList<PowerModes>();

}

6. Класс PowerModes

public class PowerModes

{

[Column("PowerModId")]

public int Id { get; set; }

[Column("ProductId")]

public int Product { get; set; }

[Column("GoulId")]

public int Goul { get; set; }

public Goals Goals { get; set; }

public ProductCatalogs ProductCatalogs { get; set; }

}

7. Класс ProductCatalogs

public class ProductCatalogs

{

[Key]

[Column("ProductId")]

public int Id { get; set; }

public string Name { get; set; }

public int NumberOfProteins { get; set; }

public int NumberOfFat { get; set; }

public int NumberOfCarbohydrates { get; set; }

public LinkedList<PowerModes> PowerModes { get; set; } = new LinkedList<PowerModes>();

public LinkedList<AllergyCards> AllergyCards { get; set; } = new LinkedList<AllergyCards>();

}

8. Класс Schedules

public class Schedules

{

[Key]

[Column("ScheduleId")]

public int Id { get; set; }

public int TrainingSessions { get; set; }

public int CountStrengthTraining { get; set; }

public int CountCardioTraining { get; set; }

public int ClientId { get; set; }

public Client Client { get; set; }

}

9. Класс Subscriptions

public class Subscription

{

[Key]

[Column ("SubscriptionId")]

public int Id { get; set; }

public string NextDate { get; set; }

public string EndDate { get; set; }

public string Name { get; set; }

public int Cost { get; set; }

public int ClientId { get; set; }

public Client Client { get; set; }

}

10. Класс ApplicationContext

public class ApplicationContext : DbContext

{

public DbSet<Client> CustomersInformation { get; set; }

public DbSet<Mentor> MentorsInformation { get; set; }

public DbSet<Subscription> Subscription { get; set; }

public DbSet<ProductCatalogs> ProductCatalogs { get; set; }

public DbSet<Schedules> Schedules { get; set; }

public DbSet<PowerModes> PowerModes { get; set; }

public DbSet<AllergyCards> AllergyCards { get; set; }

public DbSet<Goals> Goals{ get; set; }

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

modelBuilder.Entity<AllergyCards>().HasKey(u => new { u.Id, u.Product });

modelBuilder.Entity<PowerModes>().HasKey(u => new { u.Id, u.Product });

}

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

var builder = new ConfigurationBuilder();

builder.SetBasePath(Directory.GetCurrentDirectory());

builder.AddJsonFile("jsconfig.json");

var config = builder.Build();

string connectionString = config.GetConnectionString("DefaultConnection");

optionsBuilder.UseSqlServer(connectionString);

}}

11. Класс Program.

//Вывод ФИО всех клиентов

using (ApplicationContext db = new ApplicationContext())

{

var clients = from p in db.CustomersInformation.ToArray()

select p.FIO;

foreach (var p in clients)

Console.WriteLine(p);

Console.ReadKey();

}

//Вывод Номера телефона клиента и Фамилии ментора

using (ApplicationContext db = new ApplicationContext())

{

var clients = from u in db.CustomersInformation.ToArray()

from c in db.MentorsInformation.ToArray()

select new

{

phoneNumber = u.PhoneNumber,

FioMentor = c.FIO

};

foreach (var p in clients)

Console.WriteLine(p.phoneNumber + " " + p.FioMentor);

Console.ReadKey();

}

//Вывести название подписки у клиента с введенным Id

using (ApplicationContext db = new ApplicationContext())

{

var clients = from c in db.CustomersInformation.ToArray()

select c;

foreach (var c in clients)

Console.WriteLine(c.Id + " " + c.FIO);

Console.WriteLine("Введите id клиента, которого хотите проверить");

int intId = int.Parse(Console.ReadLine());

var subscriptions = from p in db.Subscription.ToArray()

where p.ClientId == intId

select p;

foreach (var p in subscriptions)

Console.WriteLine(p.Name);

Console.ReadKey();

}

//Добавление клиента к ментору с введенным Id

using (ApplicationContext db = new ApplicationContext())

{

var mentors = from p in db.MentorsInformation.ToArray()

select p;

foreach (var p in mentors)

Console.WriteLine(p.Id + " " + p.FIO + " " + p.Post);

Console.WriteLine("Введите id ментора к которому привязать клиента");

int Id = int.Parse(Console.ReadLine());

int maxClientId = db.CustomersInformation.Max(c => c.Id);

Console.WriteLine("Введите данные пользователя, которого хотите добавить");

Console.WriteLine("Введите ФИО");

string fIO = Console.ReadLine();

Console.WriteLine("Введите пароль");

string password = Console.ReadLine();

Console.WriteLine("Введите номер телефона");

string phoneNumber = Console.ReadLine();

var clients = db.CustomersInformation.ToArray();

var mentor = db.MentorsInformation.Where(c => c.Id == Id).FirstOrDefault();

Client client = new Client { Id = maxClientId + 1, FIO = fIO, PhoneNumber = phoneNumber, MentorID = Id, Password = password, Mentor = mentor };

db.CustomersInformation.Add(client);

db.SaveChanges();

var GetClients = from p in db.CustomersInformation.ToArray()

where p.Id == maxClientId + 1

select p;

Console.WriteLine(GetClients);

Console.ReadKey();

}

//Вывести Id клиента с именем Максим

using (ApplicationContext db = new ApplicationContext())

{

SqlParameter name = new SqlParameter("@name", "%Максим%");

var clients = db.CustomersInformation.FromSqlRaw("SELECT \* FROM CustomersInformation WHERE FIO LIKE @name", name).ToArray();

foreach (var emp in clients)

Console.WriteLine($"{emp.FIO}");

Console.ReadKey();

}

//Добавление продукта в карту алергика у определенного клиента

using (ApplicationContext db = new ApplicationContext())

{

var getClients = db.CustomersInformation.ToArray();

Console.WriteLine("Список клиентов");

foreach (var p in getClients)

Console.WriteLine("Id: " + p.Id + " - Фамилия: " + p.FIO);

Console.WriteLine("Введите id клиента к которому хотите добавить продукт");

int nomerClient = int.Parse(Console.ReadLine());

Console.WriteLine("Список продуктов");

var products = db.ProductCatalogs.ToArray();

foreach (var p in products)

Console.WriteLine("Id: " + p.Id + " - Название: " + p.Name);

Console.WriteLine("Введите Id продукта, который хотите добавить");

int nomerProduct = int.Parse(Console.ReadLine());

int maxAlleryCardsId = db.AllergyCards.Max(c => c.Id);

AllergyCards allergyCards = new AllergyCards { Id = maxAlleryCardsId + 1, Product = nomerProduct, Clients = nomerClient };

db.AllergyCards.Add(allergyCards);

db.SaveChanges();

var getAllergyCards = from p in db.AllergyCards.ToArray()

where p.Clients == nomerClient

select p;

Console.WriteLine("Список продуктов у клиента");

foreach (var p in getAllergyCards)

Console.WriteLine(p.ProductCatalogs.Name);

Console.ReadKey();

}

//Проверка на существование логина в БД

using (ApplicationContext db = new ApplicationContext())

{

Console.WriteLine("Введите ФИО");

string fio = Console.ReadLine();

var getClientsIsFio = from p in db.CustomersInformation.ToArray()

where p.FIO == fio

select p.FIO;

Console.WriteLine(getClientsIsFio.Any());

Console.ReadKey();}

//Удаление ментора по Id

Mentor deluser = (from Mentor in db.MentorsInformation where Mentor.Id == 5 select Mentor).First();

if (deluser != null)

{

db.MentorsInformation.Remove(deluser);

db.SaveChanges();

}

var users = db.MentorsInformation.ToArray();

Console.WriteLine("Список объектов");

foreach (Mentor u in users)

Console.WriteLine(u.FIO + " " + u.Post + " " + u.Id);

Console.ReadLine();

//Вывод подписки по Id

using (ApplicationContext db = new ApplicationContext())

{

var getClients = from p in db.CustomersInformation.ToArray()

select p;

Console.WriteLine("Список клиентов");

foreach (var c in getClients)

Console.WriteLine("Id - " + c.Id + "; ФИО - " + c.FIO);

Console.WriteLine("Введите Id клиента, у которого хотите посмотреть подписки");

int id = int.Parse(Console.ReadLine());

var getSubcription = from c in db.Subscription.ToArray()

where c.ClientId == id

select c;

foreach (var i in getSubcription)

Console.WriteLine("Название: " + i.Name + "; Стоимость: " + i.Cost);

Console.ReadKey();

}

//Удаление подписки у клиента с определенным id

using (ApplicationContext db = new ApplicationContext())

{

var Client = db.Subscription.Include(u => u.Client).ToArray();

Console.WriteLine("Список клиентов и их Id подписки");

foreach (var p in Client)

Console.WriteLine("- " + p.Client.Id + ": " + p.Client.FIO + " - " + p.Id);

Console.WriteLine("Введите id подписки, которую хотите удалить");

int intId = int.Parse(Console.ReadLine());

var subscriptions = (from p in db.Subscription

where p.Id == intId

select p).First();

if (subscriptions != null)

{

db.Subscription.Remove(subscriptions);

db.SaveChanges();

}

var subscription = db.Subscription.ToArray();

Console.WriteLine("Список всех подписок");

foreach (Subscription u in subscription)

Console.WriteLine(u.Id + " - " + u.Name + " - " + u.Cost);

Console.ReadKey();

}

//Добавление нового продукта

using (ApplicationContext db = new ApplicationContext())

{

int maxId = db.ProductCatalogs.Max(p => p.Id);

Console.WriteLine($"Max id: {maxId}");

ProductCatalogs test = new ProductCatalogs { Id = maxId + 1, Name = "Макароны", NumberOfProteins = 12, NumberOfCarbohydrates = 2, NumberOfFat = 13 };

db.ProductCatalogs.Add(test);

db.SaveChanges();

var products = db.ProductCatalogs.ToArray();

Console.WriteLine("Список продуктов");

foreach (ProductCatalogs u in products)

Console.WriteLine(u.Id + " " + u.Name);

Console.ReadKey();

}

//Изменение количества тренировок в 2 раза

using (ApplicationContext db = new ApplicationContext())

{

var users = from p in db.Schedules.ToArray()

let trainingSessions = p.TrainingSessions - 2

select new

{

TrainingSessions = trainingSessions

};

db.SaveChanges();

foreach (var p in users)

{

Console.WriteLine(p.TrainingSessions);

}

Console.ReadKey();

}

//Изменение должности у ментора с определенным Id

using (ApplicationContext db = new ApplicationContext())

{

string post = Console.ReadLine();

Mentor upduser = (from Mentor in db.MentorsInformation where Mentor.Id == 5124 select Mentor).First();

if (upduser != null)

{

upduser.Post = post;

db.SaveChanges();

}

var users = db.MentorsInformation.ToArray();

foreach (Mentor u in users)

Console.WriteLine(u.FIO + " " + u.Post + " " + u.Id);

Console.ReadKey();

}

//Узнать Id клиента по продукту из карты аллергика

using (ApplicationContext db = new ApplicationContext())

{

var product = db.ProductCatalogs.ToArray();

Console.WriteLine("Все продукты");

foreach (var p in product)

Console.WriteLine(p.Id + " - " + p.Name);

Console.WriteLine("Введите Id продукта");

int id = int.Parse(Console.ReadLine());

var productId = from c in db.ProductCatalogs.ToArray()

where c.Id == id

select c;

var allergyCards = product.SelectMany(x => x.AllergyCards);

foreach (var p in allergyCards)

Console.WriteLine(p.Client.Id);

Console.ReadKey();

}

# Заключение

Была создана и реализована модель информационной системы организации концертов. В процессе работы были получены навыки создания ER моделей, баз данных и написания программ для работы с базами данных с помощью фреймворка EntityFrameworkCore.