Содержание

[Введение 4](file:///\\PC-232-00\network\Диплом%20и%20курсовик\Курсач\Курсовой%20Проект.docx#_heading=h.gjdgxs)

[1 Экономическая характеристика объекта 5](file:///\\PC-232-00\network\Диплом%20и%20курсовик\Курсач\Курсовой%20Проект.docx#_heading=h.30j0zll)

[2 Постановка задачи 6](file:///\\PC-232-00\network\Диплом%20и%20курсовик\Курсач\Курсовой%20Проект.docx#_heading=h.1fob9te)

[2.1 Экономическое содержание задачи 6](file:///\\PC-232-00\network\Диплом%20и%20курсовик\Курсач\Курсовой%20Проект.docx#_heading=h.3znysh7)

[2.2 Входная информация 6](file:///\\PC-232-00\network\Диплом%20и%20курсовик\Курсач\Курсовой%20Проект.docx#_heading=h.2et92p0)

[3 Технические и инструментальные средства для решения задачи 7](file:///\\PC-232-00\network\Диплом%20и%20курсовик\Курсач\Курсовой%20Проект.docx#_heading=h.tyjcwt)

[3.1 Обоснование выбора языка программирования 7](file:///\\PC-232-00\network\Диплом%20и%20курсовик\Курсач\Курсовой%20Проект.docx#_heading=h.3dy6vkm)

[3.2 Технические требования к компьютеру для решения задачи 9](file:///\\PC-232-00\network\Диплом%20и%20курсовик\Курсач\Курсовой%20Проект.docx#_heading=h.1t3h5sf)

[4 Информационное обеспечение задачи 10](file:///\\PC-232-00\network\Диплом%20и%20курсовик\Курсач\Курсовой%20Проект.docx#_heading=h.4d34og8)

[4.1 Классификаторы, используемые для решения задачи 10](file:///\\PC-232-00\network\Диплом%20и%20курсовик\Курсач\Курсовой%20Проект.docx#_heading=h.2s8eyo1)

[5 Программное обеспечение задачи 13](file:///\\PC-232-00\network\Диплом%20и%20курсовик\Курсач\Курсовой%20Проект.docx#_heading=h.17dp8vu)

[5.1 Описание методов 13](file:///\\PC-232-00\network\Диплом%20и%20курсовик\Курсач\Курсовой%20Проект.docx#_heading=h.3rdcrjn)

[5.2 Алгоритм решения задачи 15](file:///\\PC-232-00\network\Диплом%20и%20курсовик\Курсач\Курсовой%20Проект.docx#_heading=h.26in1rg)

[5.3 Описание блок – схемы 15](file:///\\PC-232-00\network\Диплом%20и%20курсовик\Курсач\Курсовой%20Проект.docx#_heading=h.lnxbz9)

[Заключение 19](file:///\\PC-232-00\network\Диплом%20и%20курсовик\Курсач\Курсовой%20Проект.docx#_heading=h.35nkun2)

[Список использованных источников 20](file:///\\PC-232-00\network\Диплом%20и%20курсовик\Курсач\Курсовой%20Проект.docx#_heading=h.1ksv4uv)

[Приложение А 21](file:///\\PC-232-00\network\Диплом%20и%20курсовик\Курсач\Курсовой%20Проект.docx#_heading=h.44sinio)

[Приложение Б 22](file:///\\PC-232-00\network\Диплом%20и%20курсовик\Курсач\Курсовой%20Проект.docx#_heading=h.2jxsxqh)

Введение:

Рабочее время  проводит на оплачиваемом [труде](https://translated.turbopages.org/proxy_u/en-ru.ru.d27a9040-626a5b1a-ae5423d0-74722d776562/https/en.wikipedia.org/wiki/Wage_labour). [Неоплачиваемый труд](https://translated.turbopages.org/proxy_u/en-ru.ru.d27a9040-626a5b1a-ae5423d0-74722d776562/https/en.wikipedia.org/wiki/Unpaid_work), такой как личная работа по дому или уход за детьми или домашними животными, не считается частью рабочей недели.

Многие страны регулируют [рабочую неделю](https://translated.turbopages.org/proxy_u/en-ru.ru.d27a9040-626a5b1a-ae5423d0-74722d776562/https/en.wikipedia.org/wiki/Work_week) законодательно, например, устанавливая минимальные ежедневные [периоды отдыха](https://translated.turbopages.org/proxy_u/en-ru.ru.d27a9040-626a5b1a-ae5423d0-74722d776562/https/en.wikipedia.org/wiki/Break_(work)), ежегодные [праздники](https://translated.turbopages.org/proxy_u/en-ru.ru.d27a9040-626a5b1a-ae5423d0-74722d776562/https/en.wikipedia.org/wiki/Civic_holiday) и [максимальное количество рабочих часов](https://translated.turbopages.org/proxy_u/en-ru.ru.d27a9040-626a5b1a-ae5423d0-74722d776562/https/en.wikipedia.org/wiki/Eight-hour_day) в неделю. Рабочее время может варьироваться от человека к человеку, часто в зависимости от экономических условий, местоположения, культуры, выбора образа жизни и прибыльности средств к существованию человека. Например, кто-то, кто поддерживает детей и платит большую ипотеку, возможно, должен работать больше часов, чтобы удовлетворить основные [расходы на жизнь](https://translated.turbopages.org/proxy_u/en-ru.ru.d27a9040-626a5b1a-ae5423d0-74722d776562/https/en.wikipedia.org/wiki/Cost_of_living), чем кто-то с той же [доходной способностью](https://translated.turbopages.org/proxy_u/en-ru.ru.d27a9040-626a5b1a-ae5423d0-74722d776562/https/en.wikipedia.org/wiki/Earning_power) с более низкими затратами на жилье. В развитых странах, таких как Великобритания, некоторые работники работают неполный рабочий день, потому что они не могут найти работу на полный рабочий день, но многие выбирают сокращение рабочего времени для ухода за детьми или другой семьей; некоторые выбирают его просто для увеличения свободного времени.

Стандартное рабочее время (или нормальное рабочее время) относится к законодательству, ограничивающему рабочее время в день, в неделю, в месяц или в год. Работодатель платит более высокие ставки за сверхурочные часы, как того требует закон. Стандартное рабочее время стран мира составляет от 40 до 44 часов в неделю (но не везде: от 35 часов в неделю во Франции до 112 часов в неделю в северокорейских трудовых лагерях), а дополнительные сверхурочные выплаты примерно на 25-50% выше нормы почасовая оплата. рабочее время относится к максимальному рабочему времени сотрудника. Работник не может работать больше уровня, указанного в законе о максимальном рабочем времени

[Всемирная организация здравоохранения](https://translated.turbopages.org/proxy_u/en-ru.ru.477a08bd-627036a3-e0010bc9-74722d776562/https/en.wikipedia.org/wiki/World_Health_Organization) и [Международная организация труда](https://translated.turbopages.org/proxy_u/en-ru.ru.477a08bd-627036a3-e0010bc9-74722d776562/https/en.wikipedia.org/wiki/International_Labour_Organization) подсчитали, что во всем мире в 2016 году каждый десятый работник работал 55 и более часов в неделю, а 745 000 человек умерли в результате сердечных заболеваний или инсульта, связанных с работой в течение этих долгих часов, что делает воздействие длительного рабочего времени профессиональным заболеванием. Фактор риска с наибольшим бременем заболеваний.[[](https://translated.turbopages.org/proxy_u/en-ru.ru.477a08bd-627036a3-e0010bc9-74722d776562/https/en.wikipedia.org/wiki/Working_time#cite_note-5)

[Промышленная революция](https://translated.turbopages.org/proxy_u/en-ru.ru.477a08bd-627036a3-e0010bc9-74722d776562/https/en.wikipedia.org/wiki/Industrial_revolution) позволила большей части населения работать круглый год, потому что этот труд не был привязан к сезону, а искусственное [освещение](https://translated.turbopages.org/proxy_u/en-ru.ru.477a08bd-627036a3-e0010bc9-74722d776562/https/en.wikipedia.org/wiki/Lighting) позволяло работать дольше с каждым днем. [Крестьяне](https://translated.turbopages.org/proxy_u/en-ru.ru.477a08bd-627036a3-e0010bc9-74722d776562/https/en.wikipedia.org/wiki/Peasants) и сельскохозяйственные рабочие переехали из сельской местности на работу на городские [заводы](https://translated.turbopages.org/proxy_u/en-ru.ru.477a08bd-627036a3-e0010bc9-74722d776562/https/en.wikipedia.org/wiki/Factory), и рабочее время в течение года значительно увеличилось. До [коллективных переговоров](https://translated.turbopages.org/proxy_u/en-ru.ru.477a08bd-627036a3-e0010bc9-74722d776562/https/en.wikipedia.org/wiki/Collective_bargaining) и [законов о защите работников](https://translated.turbopages.org/proxy_u/en-ru.ru.477a08bd-627036a3-e0010bc9-74722d776562/https/en.wikipedia.org/wiki/Labour_law) существовал финансовый стимул для компании максимизировать отдачу от дорогостоящей техники, имея долгие часы. Записи показывают, что графики работы до двенадцати-шестнадцати часов в день, от шести до семи дней в неделю практиковались на некоторых промышленных площадках.

1. Экономическая характеристика объекта

Расписание для курсирующих по постоянному [маршруту](https://dic.academic.ru/dic.nsf/ruwiki/657283) [повозок](https://dic.academic.ru/dic.nsf/ruwiki/237720) появилось в [Великобритании](https://dic.academic.ru/dic.nsf/ruwiki/86) в [1784 году](https://dic.academic.ru/dic.nsf/ruwiki/1884), но лишь для времени отправления, а не прибытия. Расписание с временем прибытия появилось уже во времена паровозов в [1840 году](https://dic.academic.ru/dic.nsf/ruwiki/1810).  
С расписанием связано и [Среднее время по Гринвичу](https://dic.academic.ru/dic.nsf/ruwiki/925): в [1847 году](https://dic.academic.ru/dic.nsf/ruwiki/473) британские паровозные компании объединились и для всебританского расписания было определено единое время по времени [Гринвичской обсерватории](https://dic.academic.ru/dic.nsf/ruwiki/384683).

2. Постановка задачи

2.1 Экономическое содержание задачи

Цель курсового проекта = создание мобильного приложения на базе Android для AO «Расписание Предприятия».

Разработанный модуль позволяет производить различные действия над Расписанием с помощью смартфона.

Программа создана на языке С#, графический интерфейс простой и понятный, сама программа проста в использовании и не требует специального образования или прохождение курсов.

2.2. Входная информация

Входная информация представляет собой сбор данных о человеке, который находиться на предприятии.

2.3. Выходная информация

Выходной информацией является уже обновленное расписание, над которым были произведены какие-либо из доступных действий.

3 Технические и инструментальные средства для решения задачи

3.1 обоснование выбора языка программирования

Для написания программы была выбрана среда программирования Visual Studio 2012 C#, основанная на языке программирования C#. Данная среда выгодно отличается эффективностью и надежностью.

Для стабильного функционирования программы необходим компьютер фирмы IBM или совместимый с ним, с объёмом оперативной памяти не менее 128 Мб., процессор с частотой не менее 600мГц.

Для решения поставленной задачи необходимо использовать функциональную, эффективную и удобную платформу для разработки, позволяющую применять принципы объектно-ориентированного программирования. В качестве такой платформы была выбрана среда .NET.

Среда разработки Visual Studio, поставляемая вместе с .NET, предоставляет необходимый инструментарий для эффективного и быстрого создания приложений с графическим интерфейсом.

Появление технологии .NET повлекло за собой массовую реконструкцию некоторых языков программирования, стремящихся использовать те или иные возможности платформы, такие как C++ и Visual Basic. Microsoft решили предложить разработчикам альтернативу - язык, ориентированный специально .NET и создали C#. Сами разработчики языка описывают его, как простой, современный, объектно-ориентированный и безопасный язык программирования. Синтаксически C# напоминает C++ и Java, что позволяет за достаточно короткое время изучить тонкости нового языка.

Несмотря на то, что C# и .NET предназначены в первую очередь для веб-разработки, их также активно применяют для создания приложений, которые должны устанавливаться на машине конечного пользователя, где и будет выполняться вся обработка данных. Разработку таких приложений обеспечивает библиотека Windows Forms, позволяющая проектировать графический интерфейс. Система, описанная в данной работе, разработана именно с помощью библиотеки Windows Forms.

Язык программирования C# претендует на подлинную объектную ориентированность.

Язык программирования C# призван реализовать компонентно-ориентированный подход к программированию, который способствует меньшей машинно-архитектурной зависимости результирующего программного кода, большей гибкости, переносимости и легкости повторного использования программ.

Принципиально важным отличием от предшественников является изначальная ориентация на безопасность кода.

Расширенная поддержка событийно-ориентированного программирования.

Язык программирования C# является «родным» для создания приложений в среде Microsoft .NET, поскольку наиболее тесно и эффективно интегрирован с ней.

Microsoft Visual Studio- это версия Visual Studio и .NET Framework, которая поддерживает новые и улучшенные объекты, включает среду разработки с обновленным интерфейсом и отличается интегрированной поддержкой Microsoft SQL Server, позволяя создавать и развертывать проекты с применением сервера баз данных. Из инструмента программиста, пишущего и отлаживающего код, Microsoft Visual Studio, превратилась в полноценное инструментальное средство, позволяющее автоматизировать деятельность всех членов команды, работающих над проектом.

3.2 Технические требования к компьютеру для решения задачи

Рекомендуемые системные требования к компьютеру на которой будет установлена IDE Android Sudio следующие:

* Операционная система – Microsoft® Windows® 7/8/10 (32- или 64-bit);
* оперативная память – 3 ГБ минимум, рекомендуется 8 ГБ;
* пространство на жестком диске – рекомендуется 4 ГБ;
* монитор – 1280 х 800 минимальное разрешение экрана.

4 Информационные обеспечение задачи

4.1 Классификаторы, используемые для решения задачи

Классификатор представляет собой набор таблиц (справочников).

Объект – некоторая сущность в цифровом пространстве, обладающая определённым состоянием и поведением в данной предметной области.

Атрибут – элемент, описывающий любую характеристику объекта, важную для данной предметной области. Объектом в данной базе данных будет: информация о работнике приведенная в таблице 4.1, документе показанных на таблице 4.2 и местонахождении документа на таблице 4.3.

Таблица 4.1 – Атрибуты работника

|  |  |
| --- | --- |
| Объект | Атрибуты |
| Работник | Идентификационный код  Имя  Сколько лет  Фамилия |

Таблица 4.2 – Атрибуты документа

|  |  |
| --- | --- |
| Объект | Атрибуты |
| Расписание | Идентификационный код  День  Работник |

Таблица 4.3 – Атрибуты местонахождения документа

|  |  |
| --- | --- |
| Объект | Атрибуты |
| День | Идентификационный код  Имя дня  Время прихода  Время ухода |

Для выполнения проекта была выбрана СУБД SQL Server. Схема БД представлена на рисунке 4.1.

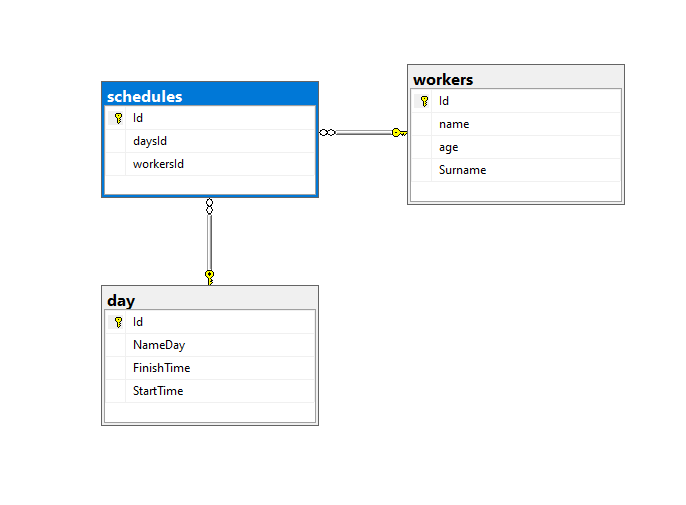


Рисунок 4.1 – Связи таблиц в БД

Структура таблиц Расписания, местонахождения и работников базы данных представлена на рисунках 4.2 – 4.4 соответственно.

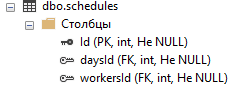


Рисунок 4.2 – Таблица «Работников»

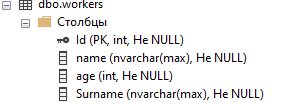
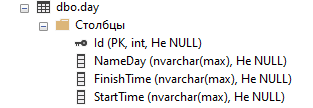


Рисунок 4.3 – Таблица «Дни недели»



5 Программное обеспечение задачи

5.1 Описание методов

Для примера взята часть кода для входа в приложение под своей учетной записью. При нажатии на кнопку вызывается активити, для дальнейшей работы с приложением.

<Window x:Class="Рабочее\_время\_Работиков.Add"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:Рабочее\_время\_Работиков"

mc:Ignorable="d"

Title="Add" Height="450" Width="491">

<Grid HorizontalAlignment="Center" Width="491" Background="Beige" >

<StackPanel RenderTransformOrigin="0.497,0.516" Margin="65,63,65,0">

<Label Height="25" Content="Имя"/>

<TextBox Background="White" BorderBrush="Black" TextChanged="TextBox\_TextChanged" x:Name="a"/>

<Label Height="25" Content="Фамилия"/>

<TextBox Background="White" BorderBrush="Black" x:Name="b" Cursor="IBeam"/>

<Label Height="25" Content="Возраст"/>

<TextBox Background="White" BorderBrush="Black" x:Name="c"/>

</StackPanel>

<Button Margin="138,379,138,23" Background="Black" Content="Добавить Работника" Click="Button\_Click"/>

</Grid>

</Window>

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

using Рабочее\_время\_Работиков.Data;

using Рабочее\_время\_Работиков.Models;

namespace Рабочее\_время\_Работиков

{

/// <summary>

/// Логика взаимодействия для Add.xaml

/// </summary>

public partial class Add : Window

{

public Add()

{

InitializeComponent();

}

private void Button\_Click(object sender, RoutedEventArgs e)

{

DataBaseContext context=new DataBaseContext();

var worker = new Workers(a.Text,Convert.ToInt32(c.Text),b.Text);

context.Add(worker);

context.SaveChanges();

addTime addTime = new addTime();

addTime.Show();

}

private void TextBox\_TextChanged(object sender, TextChangedEventArgs e)

{

}

private void b\_TextChanged(object sender, TextChangedEventArgs e)

{

}

}

}

5.2 Алгоритм решения задачи

Для проведения каких-либо действий с Расписанием сначала необходимо выбрать раздел: Добавить работника, Время, добавить, ознакомиться или проверить Расписание.

5.3 Описание блок – схемы

Последовательное описание блок – схемы приложения приведено ниже:

Шаг 1. Запуск программы.

Шаг 2. Вводим имя, возраст, Время. Потом нажимаем добавить.

Шаг 3. Вход в приложение под своей учетной записью.

Шаг 4. Вводим, когда приходит Работник и уходит это действие добавляется в Базу данных.

Шаг 5. Выводится через Базу данных на экран Расписание каждого работника.

Шаг 6. Выполнение

Шаг 7. Закрытие приложения.

Блок-схема приведена в приложении А.

Листинг программы приведен в приложении Б.

5.4 Описание интерфейса Программы

В активити входа в приложение, представленном на рисунке 5.1, находятся три текстовых поля Имя, Фамилия и возраст, кнопка добавить работника. Для дальнейшей работы нужно Вести данные работника, после этого откроется основное активити, представленное на рисунке 5.2 с возможностью выбора действия над расписанием. При нажатии на кнопку происходит открытие нужного нам активити, которая показана на рисунке



Рисунок 5.1 – Активити Добавления



Рисунок 5.2 – Основное активити

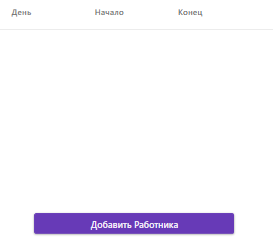


Рисунок 5.3 – Активити согласования

Для успешной работы с программой необходимо:

1. Версия Android 4.0 и выше.
2. Подключение к интернету.
3. Учетная запись.

Заключение

Программа разработана на языке С# в интегрированной среде разработки Android Studio с использованием СУБД SQL Server. Приложение интуитивно понятно даже неопытному пользователю и не затрудняет его работу с Расписанием с помощью смартфона

В дальнейшем в приложение будет обновляться и будут добавляться новые функции необходимые для работы с Расписанием.

В процессе написания кода для данной программы были приобретены практические навыки в работе в СУБД SQL Server и языке программирования С#.

Цель курсового проекта достигнута. Задачи выполнены.

# Список использованных источников

Интернет ресурсы:

1 Википедия. Язык программирование СИ [Электронный ресурс] – режим доступа: https://ru.wikipedia.org/wiki/Си\_(язык\_программирования)

2 Microsoft. Документация по C#. [Электронный ресурс] – режим доступа: <https://docs.microsoft.com/ru-ru/dotnet/csharp/>

Приложение А

(Обязательно)

Блок – схема программы

Запуск программы

Начало

Введены верные Данные работника?

Нет

Да

Вход в приложение

Выбрано действие?

Нет

Закрытие

Выполнение

Да

Сохранение документа

Рисунок А.1–Блок схема программы

Приложение Б

(обязательно)

namespace Рабочее\_время\_Работиков.Migrations

{

public partial class a : Migration

{

protected override void Up(MigrationBuilder migrationBuilder)

{

migrationBuilder.CreateTable(

name: "schedules",

columns: table => new

{

Id = table.Column<int>(type: "int", nullable: false)

.Annotation("SqlServer:Identity", "1, 1")

},

constraints: table =>

{

table.PrimaryKey("PK\_schedules", x => x.Id);

});

migrationBuilder.CreateTable(

name: "day",

columns: table => new

{

Id = table.Column<int>(type: "int", nullable: false)

.Annotation("SqlServer:Identity", "1, 1"),

NameDay = table.Column<string>(type: "nvarchar(max)", nullable: false),

ScheduleId = table.Column<int>(type: "int", nullable: true)

},

constraints: table =>

{

table.PrimaryKey("PK\_day", x => x.Id);

table.ForeignKey(

name: "FK\_day\_schedules\_ScheduleId",

column: x => x.ScheduleId,

principalTable: "schedules",

principalColumn: "Id");

});

migrationBuilder.CreateTable(

name: "time",

columns: table => new

{

Id = table.Column<int>(type: "int", nullable: false)

.Annotation("SqlServer:Identity", "1, 1"),

name = table.Column<string>(type: "nvarchar(max)", nullable: false),

ScheduleId = table.Column<int>(type: "int", nullable: true)

},

constraints: table =>

{

table.PrimaryKey("PK\_time", x => x.Id);

table.ForeignKey(

name: "FK\_time\_schedules\_ScheduleId",

column: x => x.ScheduleId,

principalTable: "schedules",

principalColumn: "Id");

});

migrationBuilder.CreateTable(

name: "workers",

columns: table => new

{

Id = table.Column<int>(type: "int", nullable: false)

.Annotation("SqlServer:Identity", "1, 1"),

name = table.Column<string>(type: "nvarchar(max)", nullable: false),

age = table.Column<int>(type: "int", nullable: false),

ScheduleId = table.Column<int>(type: "int", nullable: true)

},

constraints: table =>

{

table.PrimaryKey("PK\_workers", x => x.Id);

table.ForeignKey(

name: "FK\_workers\_schedules\_ScheduleId",

column: x => x.ScheduleId,

principalTable: "schedules",

principalColumn: "Id");

});

migrationBuilder.CreateIndex(

name: "IX\_day\_ScheduleId",

table: "day",

column: "ScheduleId");

migrationBuilder.CreateIndex(

name: "IX\_time\_ScheduleId",

table: "time",

column: "ScheduleId");

migrationBuilder.CreateIndex(

name: "IX\_workers\_ScheduleId",

table: "workers",

column: "ScheduleId");

}

protected override void Down(MigrationBuilder migrationBuilder)

{

migrationBuilder.DropTable(

name: "day");

migrationBuilder.DropTable(

name: "time");

migrationBuilder.DropTable(

name: "workers");

migrationBuilder.DropTable(

name: "schedules");

}

}

}

namespace Рабочее\_время\_Работиков.Migrations

{

public partial class Otec2 : Migration

{

protected override void Up(MigrationBuilder migrationBuilder)

{

migrationBuilder.DropForeignKey(

name: "FK\_schedules\_time\_timesId",

table: "schedules");

migrationBuilder.DropTable(

name: "time");

migrationBuilder.DropIndex(

name: "IX\_schedules\_timesId",

table: "schedules");

migrationBuilder.DropColumn(

name: "timesId",

table: "schedules");

migrationBuilder.AddColumn<string>(

name: "FinishTime",

table: "day",

type: "nvarchar(max)",

nullable: false,

defaultValue: "");

migrationBuilder.AddColumn<string>(

name: "StartTime",

table: "day",

type: "nvarchar(max)",

nullable: false,

defaultValue: "");

}

protected override void Down(MigrationBuilder migrationBuilder)

{

migrationBuilder.DropColumn(

name: "FinishTime",

table: "day");

migrationBuilder.DropColumn(

name: "StartTime",

table: "day");

migrationBuilder.AddColumn<int>(

name: "timesId",

table: "schedules",

type: "int",

nullable: false,

defaultValue: 0);

migrationBuilder.CreateTable(

name: "time",

columns: table => new

{

Id = table.Column<int>(type: "int", nullable: false)

.Annotation("SqlServer:Identity", "1, 1"),

name = table.Column<string>(type: "nvarchar(max)", nullable: false)

},

constraints: table =>

{

table.PrimaryKey("PK\_time", x => x.Id);

});

migrationBuilder.CreateIndex(

name: "IX\_schedules\_timesId",

table: "schedules",

column: "timesId");

migrationBuilder.AddForeignKey(

name: "FK\_schedules\_time\_timesId",

table: "schedules",

column: "timesId",

principalTable: "time",

principalColumn: "Id",

onDelete: ReferentialAction.Cascade);

}