Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Уральский федеральный университет имени первого Президента России Б.Н. Ельцина» (УрФУ) Институт радиоэлектроники и информационных технологий РТФ

Кафедра « »

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель курсового проектирования:  
Мирвода Сергей Геннадьевич

Дата защиты \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ОТЧЁТ ПО КУРСОВОЙ РАБОТЕ

«ПРИЛОЖЕНИЕ ДЛЯ УЧЕТА РАБОЧЕГО ВРЕМЕНИ»

ПО ДИСЦИПЛИНЕ: ЯЗЫКИ И МЕТОДЫ ПРОГРАММИРОВАНИЯ

Студенты:

Крапивин Михаил Александрович

Лубин Евгений Игоревич

Бекбулатов Дмитрий Владиславович

Группа: РИ-311055

Екатеринбург 2023

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc155964023)

[ЦЕЛЬ 4](#_Toc155964024)

[УПРАВЛЕНИЕ СТАДИЯМИ ПРОЕКТА 5](#_Toc155964025)

[ПРОЕКТИРОВАНИЕ 6](#_Toc155964026)

[РАЗРАБОТКА 9](#_Toc155964027)

[АЛГОРИТМ РАБОТЫ 11](#_Toc155964028)

[ЗАКЛЮЧЕНИЕ 16](#_Toc155964029)

[ЛИТЕРАТУРА 17](#_Toc155964030)

[ПРИЛОЖЕНИЕ А 18](#_Toc155964031)

**Репозиторий в GitHub**: https://github.com/MgrDim/Time-tracking-app-project.git

# ВВЕДЕНИЕ

Давно не секрет, что время всегда считалось одним из самых главных ресурсов в мире. Поэтому многие организации заинтересованы в построении эффективной системы, деятельность которой будет направлена на оптимизацию управления временем в коллективе, дабы добиться правильного распределения приоритетов выполнения поставленных задач. Своим продуктом мы решили оптимизировать и облегчить реализацию концепции, описанной выше.

**Видение проекта** заключается в создании приложения, которое поможет пользователям эффективно управлять своим рабочим временем и развивать навыки тайм-менеджмента. Наше приложение будет предлагать уникальные функции, такие как учет времени и экспорт данных учета в Excel-файл. Они помогут пользователям отслеживать и анализировать потраченное на выполнение задач из разных категорий время. Результат работы будет достигнут долгим и упорным трудом, а также дружным коллективом, который будет исполнять задачи в срок.

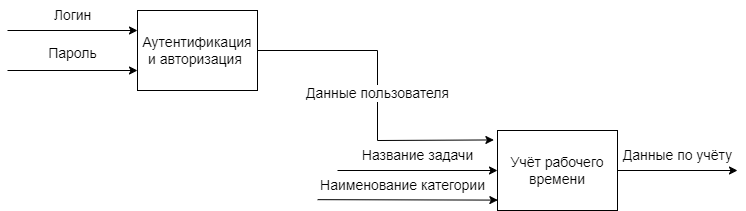
# ЦЕЛЬ

Основной и главной целью этого проекта является создание удобного и понятного приложения, которое будет учитывать и хранить в себе данные о затраченном времени на выполнение поставленных задач из разных категорий профессиональной деятельности.

Также для удобного анализирования полученной информации было решено реализовать возможность экспортировать данные из сервера базы данных и конвертировать их в удобную для чтения и понимания форму.

# УПРАВЛЕНИЕ СТАДИЯМИ ПРОЕКТА

Для лучшего понимания проекта было принято решение создать функциональную диаграмму (рис. 1).

Рисунок 1 – Функциональная диаграмма продукта (как он видится)

Чтобы облегчить работу с проектом, был создан репозиторий на GitHub, в котором публиковались текущие версии приложения и диаграммы. Также в нём публиковались новые задачи и требования по проекту, что крайне ускорило, как и сам процесс разработки, так и проводимый контроль качества, помогающий выявить ошибки и отклонения от изначальной задумки продукта.

# ПРОЕКТИРОВАНИЕ

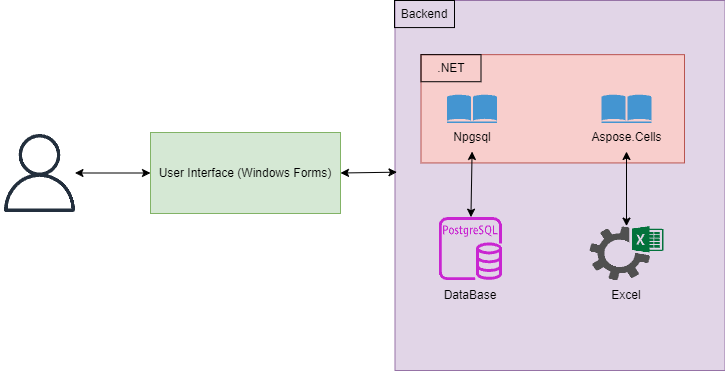
На первом этапе было предложено видение архитектуры приложения, представленное на рисунке 2.

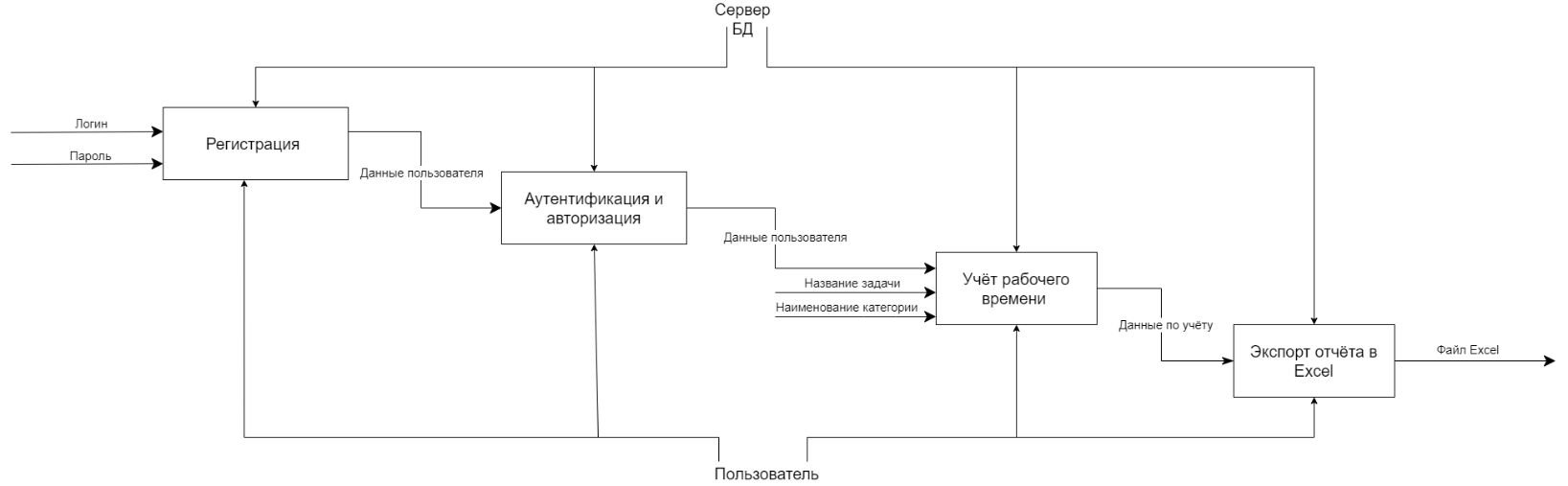
Рисунок 2 – Архитектурная диаграмма

По задумке пользователь взаимодействует с системой через пользовательский интерфейс, который выполнен при помощи технологии полнофункционального графического интерфейса Windows Forms, содержащегося в Visual Studio.

Также следует взять во внимание, что данные будут храниться в базе данных. В качестве СУБД будет использован PostgreSQL, так как она является для нас более знакомой и изученной. Помимо этого, была добавлена возможность экспортировать из базы данных в Excel-файл информацию об активности пользователя за выбранный промежуток времени.

Исходя из того, что для разработки приложения используется платформа Windows Forms .NET пользователю необходимо иметь на своем компьютере операционную систему семейства Windows. Так как база данных реализована на удаленном сервере, то пользовательскому компьютеру не обязательно иметь установленный PostgreSQL.

Была дополнена изначальная функциональная диаграмма (рис. 3), которая теперь полностью описывает основной функционал приложения. По диаграмме можно визуально отследить взаимодействие различных модулей приложения: какие данные поступают на каждый из модулей, и какие данные он возвращает.

Рисунок 3 – Функциональная диаграмма

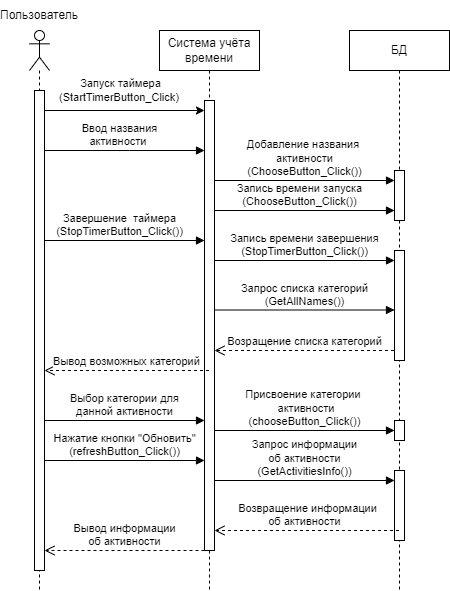
Чтобы описать взаимодействия между объектами, находящимися в одной системе, в виде сообщений, действий и операций, при этом соблюдая определенный порядок выполнения работы и обмена информации во времени, была создана диаграмма последовательностей (рис. 4).

Рисунок 4 – Диаграмма последовательностей

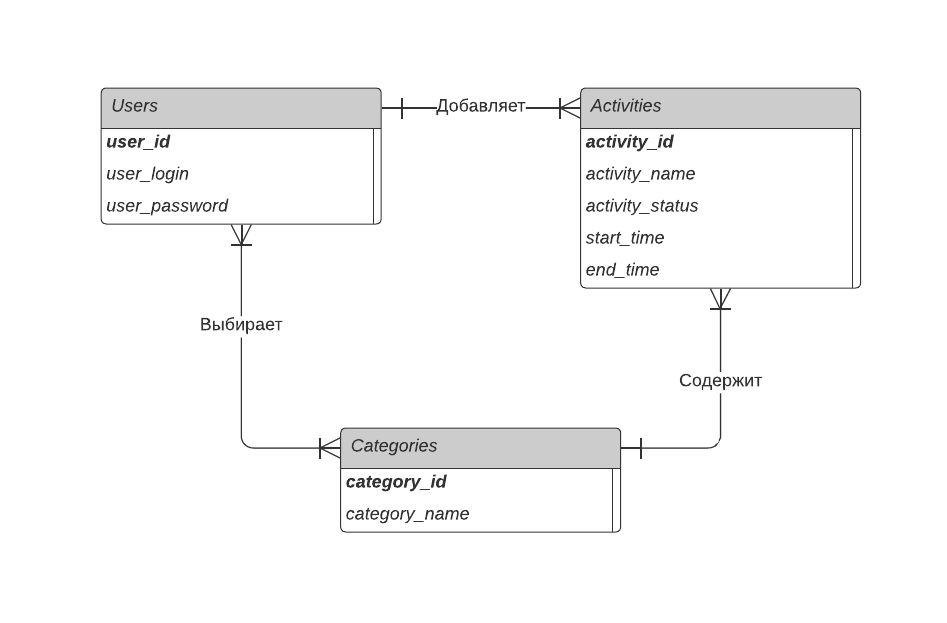
Для упрощения разработки базы данных была создана ее концептуальная модель (рис. 5), которая демонстрирует логические взаимосвязи между объектами и их характеристики. Из нее видно, что пользователь может добавлять активности в базу данных, а также выбирать нужную ему категорию для каждой активности.

Рисунок 5 – Концептуальная модель БД

# РАЗРАБОТКА

В соответствии с системным ПО для реализации продукта был выбран язык C#, так как:

1. На нем можно реализовать структурированный и модульный код (C# – это объектно-ориентированный язык программирования).
2. Данный язык обладает огромным комьюнити, благодаря чему при возникновении затруднений можно найти полезную информацию.
3. Было решено, что приложение будет разрабатываться на платформе .NET, работающей с несколькими языками программирования. Из них был выбран C#, как наиболее знакомый.

Для реализации проекта потребовалось подключить две библиотеки. Одной из них является Npgsql. Она позволяет программе взаимодействовать с сервером баз данных PostgreSQL. Второй библиотекой является Aspose.Cells, которая, в свою очередь, позволяет разработчикам программного обеспечения создавать, читать, преобразовывать, обновлять и печатать файлы электронных таблиц из своих приложений.

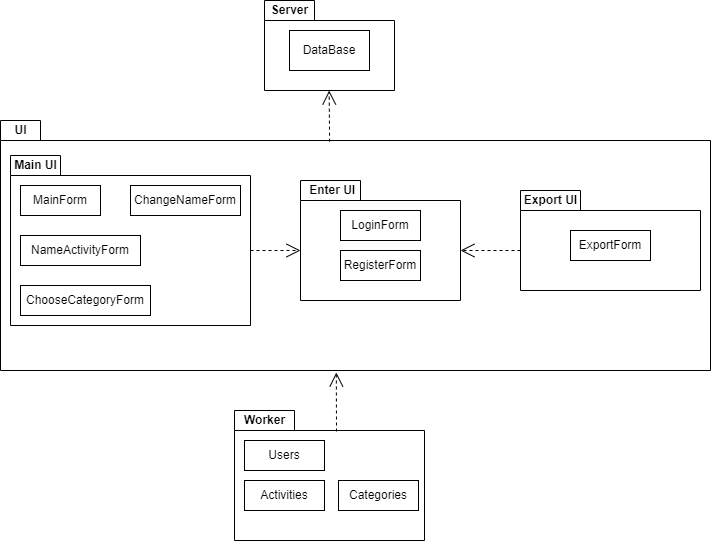
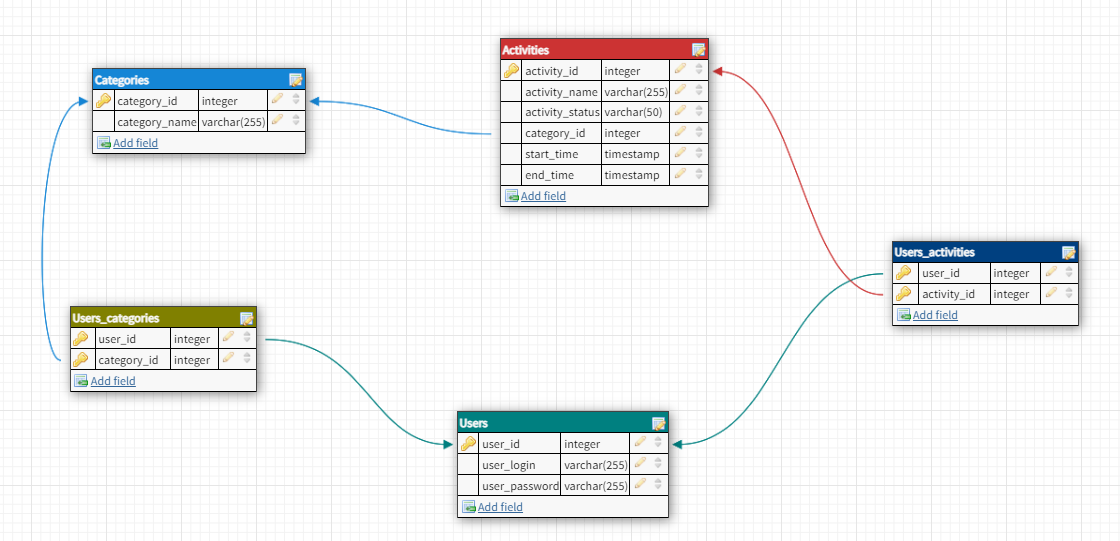
Для того, чтобы продемонстрировать взаимодействие элементов системы, была создана диаграмма пакетов, представленная на рисунке 6.

Рисунок 6 – Диаграмма пакетов

Далее на основе концептуальной модели была реализована физическая модель базы данных, представленная на рисунке 7.

Рисунок 7 – Физическая модель БД

По сравнению с концептуальной моделью в физической есть еще две таблицы (users\_activities, users\_categories). Они были созданы для реализации в БД концепции наличия у пользователя нескольких задач и категорий.

В результате работы был получен конечный вариант кода программы (приложение А).

# АЛГОРИТМ РАБОТЫ

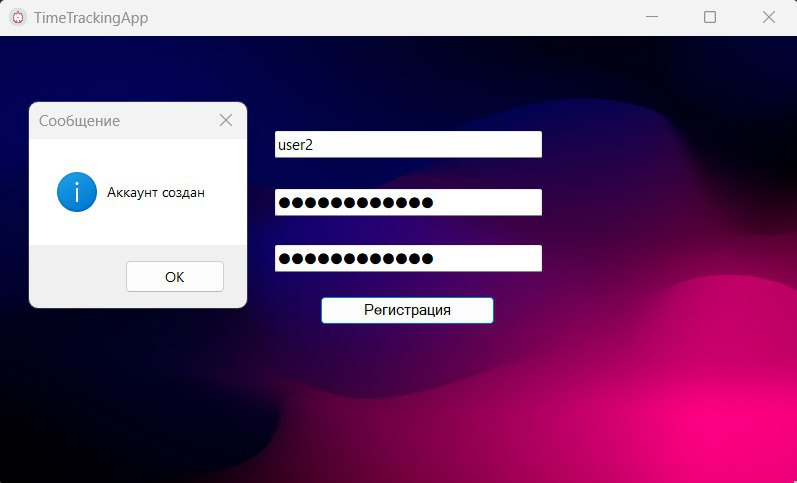
В начале у пользователя есть выбор, пройти аутентификацию или регистрацию (рис. 8, 9). Чтобы продемонстрировать всю работу программы, создадим нового пользователя с произвольным паролем и логином, после чего зайдем под ним в систему.

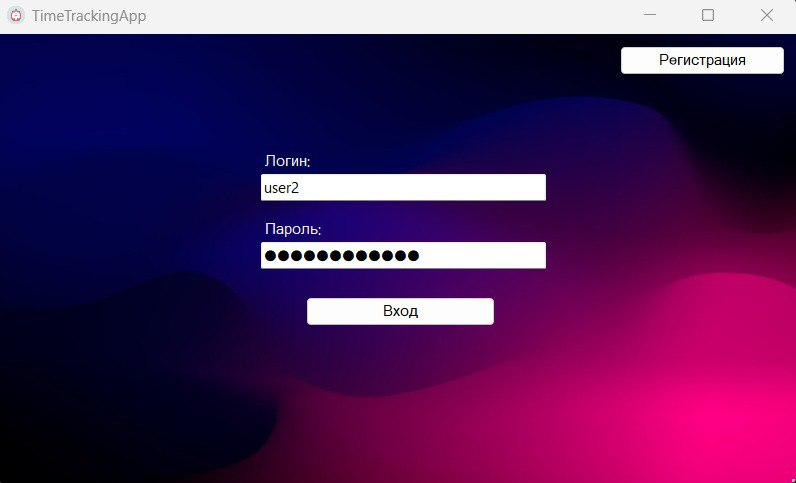
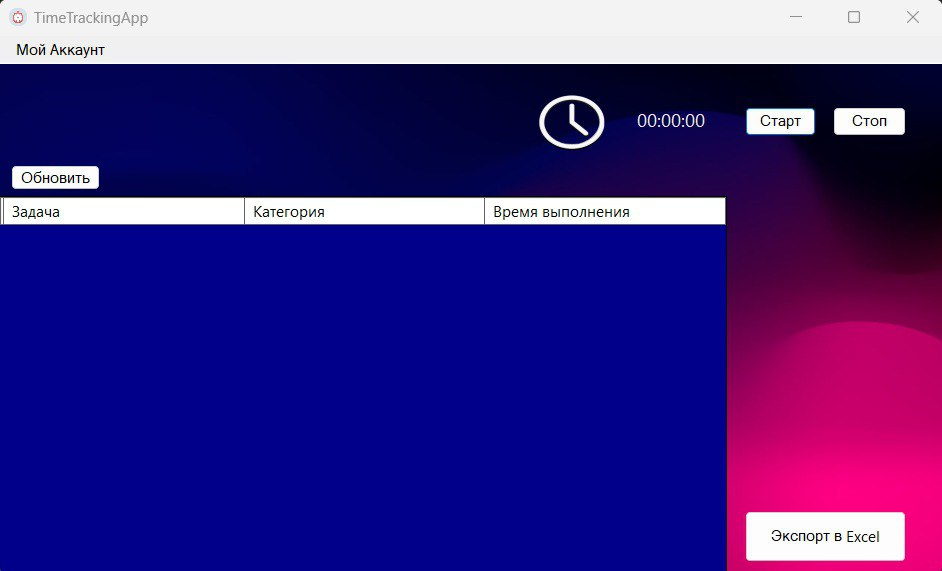
Рисунок 8 – Регистрация

Рисунок 9 – Аутентификация

Пройдя аутентификацию, пользователь переносится на главное окно приложения, представленное на рисунке 10.

Рисунок 10 – Главное окно программы

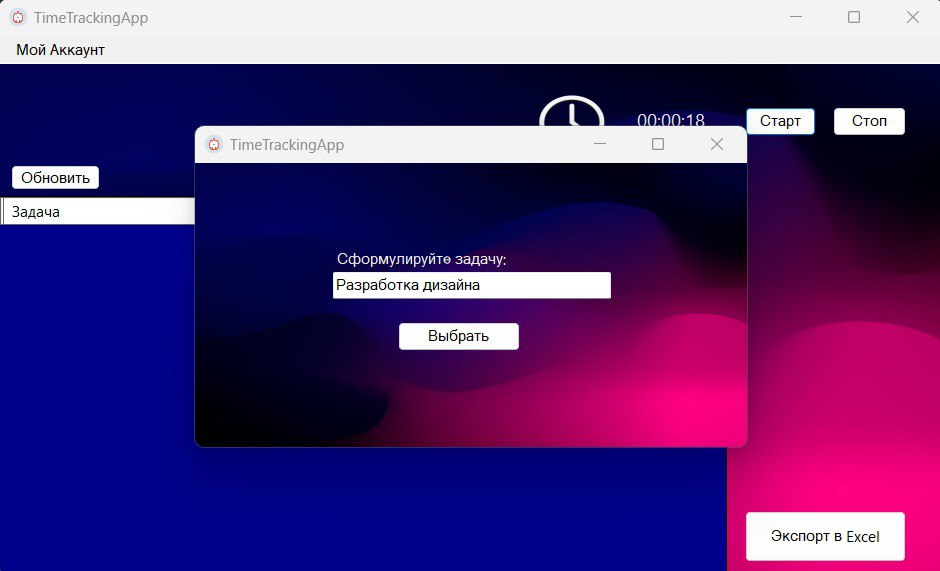
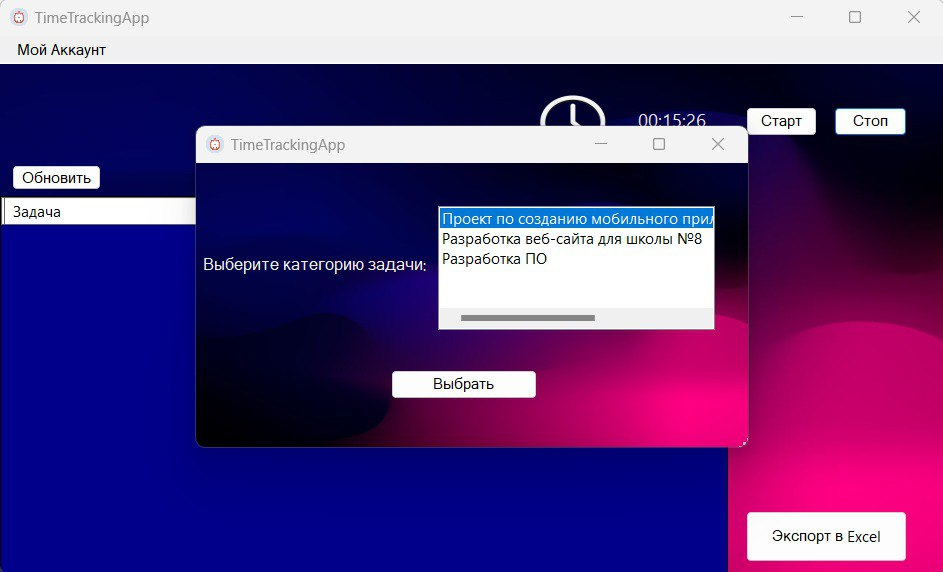
Для начала работы пользователю нужно запустить таймер, нажав на «Старт». Далее ему нужно сформулировать задачу в отведенном окне (рис. 11). По окончании работы пользователю нужно нажать на «Стоп». После нажатия ему будет предложено выбрать категорию, к которой относится его задача (рис. 12).

Рисунок 11 – Окно формулировки задачи

Рисунок 12 – Окно выбора категории

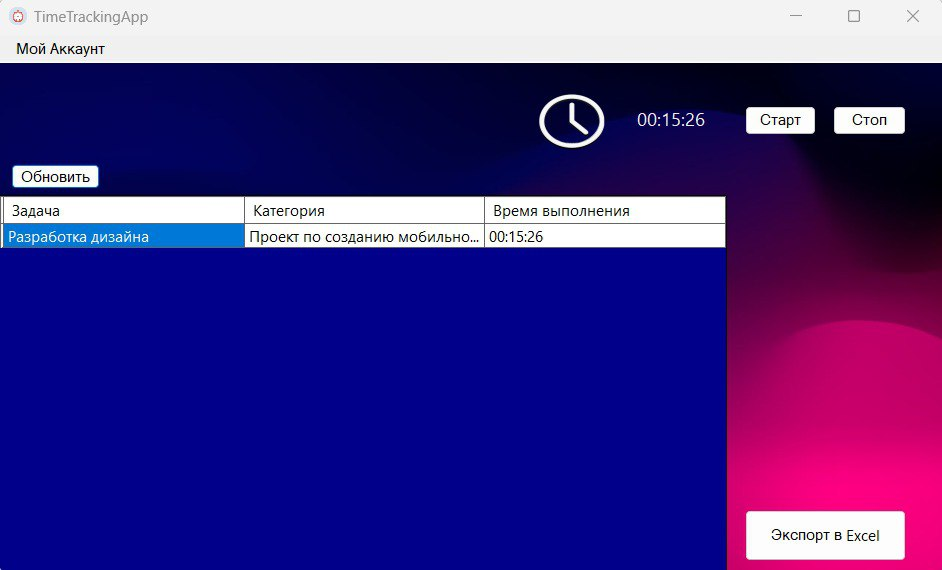
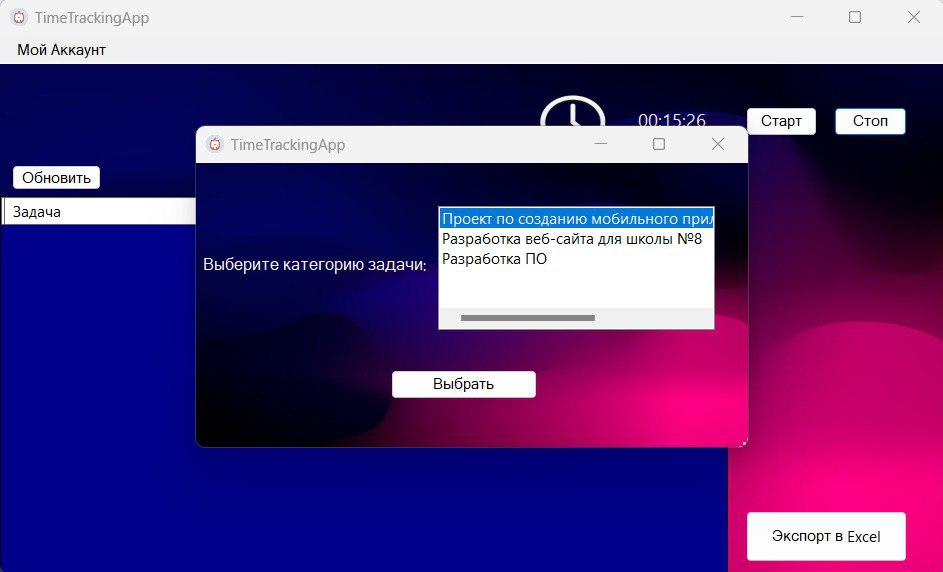
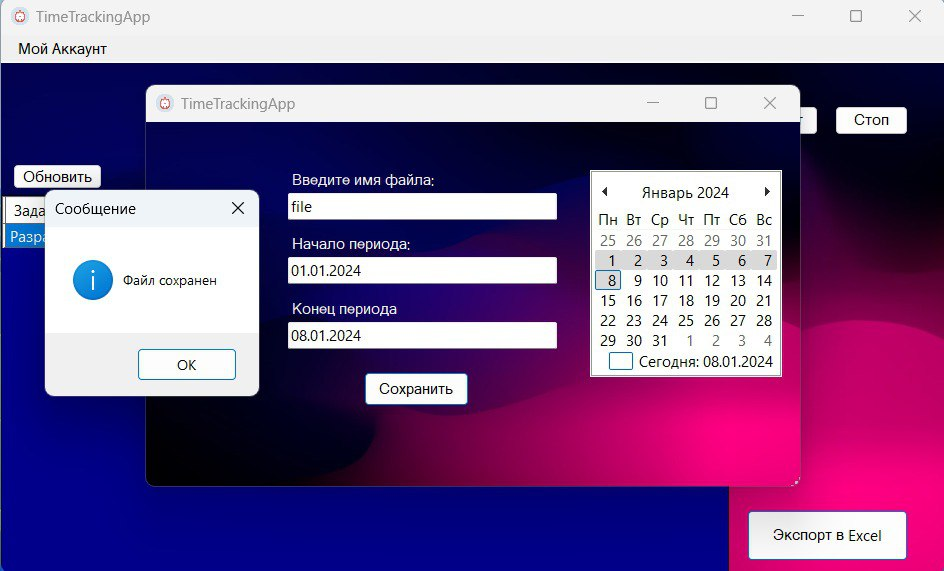
Для того, чтобы таблица с задачами на главном окне обновилась, нужно нажать на «Обновить» (рис. 13).

Рисунок 13 – Таблица с задачами

Далее пользователь может экспортировать данные таблицы, взятые из базы данных, в Excel-файл. Для этого нужно в отведенном окне (рис. 14) ввести имя файла и выбрать необходимый временной промежуток.

Рисунок 14 – Окно экспорта в Excel

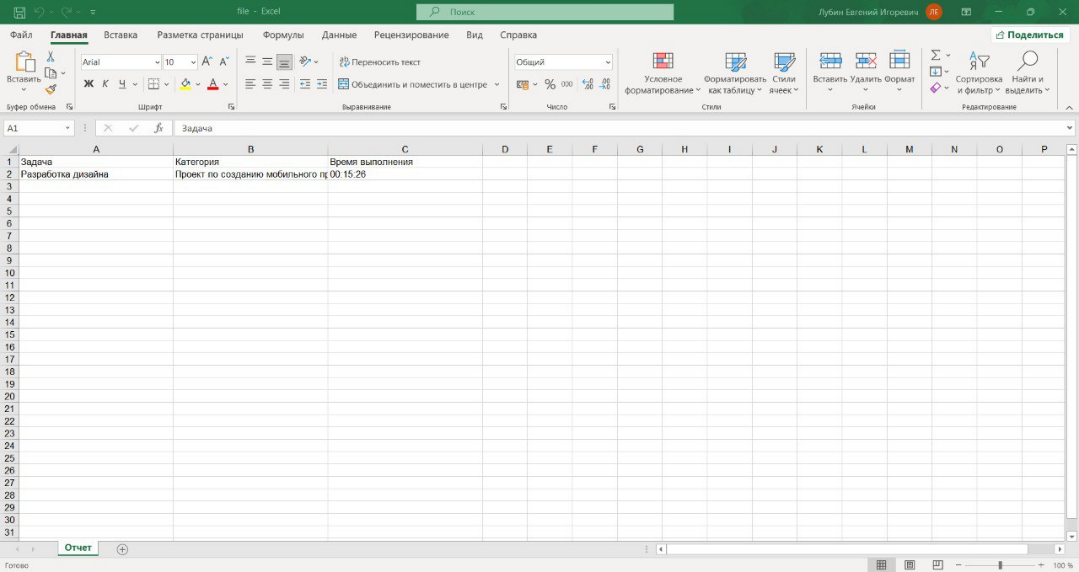
Откроем скачанный файл и убедимся в корректности полученного отчета, выгруженного из базы данных (рис. 15).

Рисунок 15 – Excel-файл с данными из БД

Для обеспечения безопасности паролей пользователей было решено хранить их хеш с добавлением соли (через хеш-функцию проходит пароль с приписанной справа солью). Структура столбца «user\_password»:

«хеш строки пароль+соль»$«соль»

Таблицы «users», «activities» и «users\_activities» представлены на рисунках 16, 17, 18 соответственно.

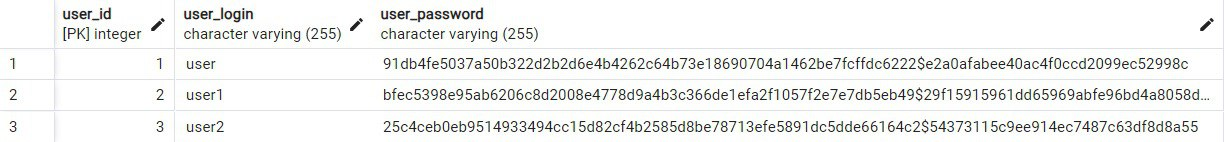
Рисунок 16 – Таблица «users»

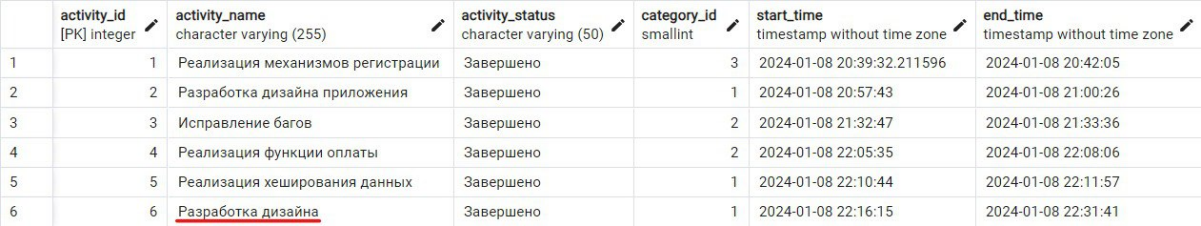
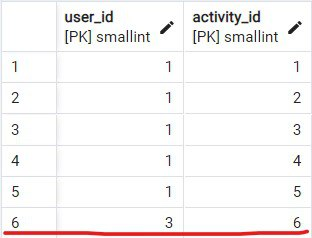
Рисунок 17 – Таблица «activities»

Рисунок 18 – Таблица «users\_activities»

По рисункам 17, 18 можно заметить, что задача, созданная ранее пользователем, зафиксировалась в базе.

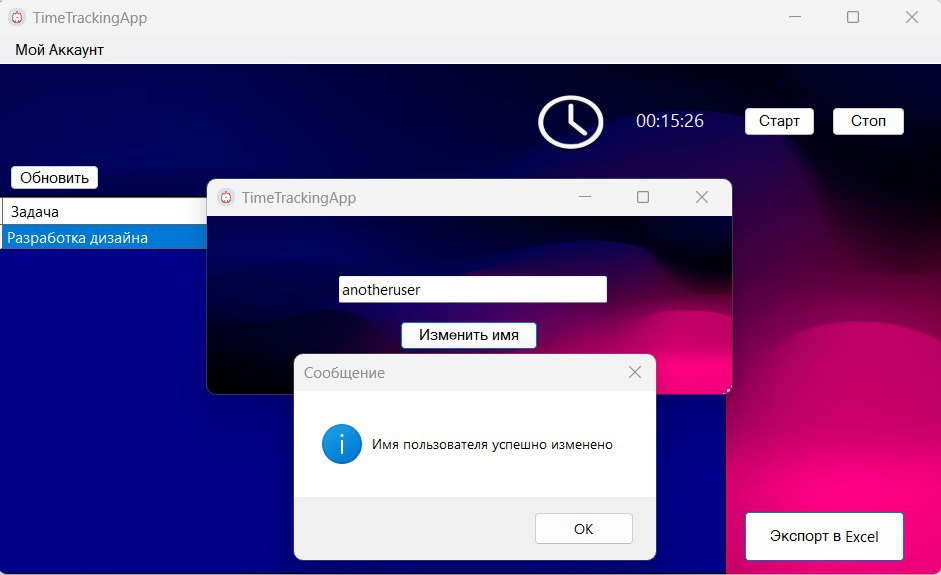
Вдобавок была реализована функция смены логина. Его можно сменить, нажав на «Мой Аккаунт»/«Сменить логин», заполнив поле в отведенном окне и подтвердив все изменения (рис. 19).

Рисунок 19 – Окно смены логина

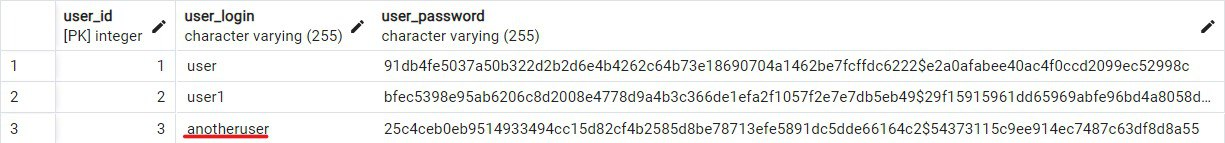
Данные изменения сразу же отразились в базе данных. В таблице «users» в столбце user\_login значение поменялось на новое (рис. 20).

Рисунок 20 – Измененная таблица «users»

# ЗАКЛЮЧЕНИЕ

Таким образом, наша программа полностью соответствует первоначальной задумке. Наш продукт можно использовать для учета рабочего времени, например, при удаленной работе. Учет времени поможет рабочим грамотно распределить нагрузку и упростит руководителю контроль за работой подопечных.

У нас получилось создать приложение с удобным интерфейсом, которое связано с базой данной. Оно берет данные из базы (например, логин и пароль при аутентификации), а также вносит их в нее (например, заполняет таблицу «activities» и «users\_activities» при создании задачи). Также мы смогли реализовать выгрузку информации о задачах за любой произвольный временной промежуток в Excel-файл. Данная функция может быть полезна для анализа затраченного на работу времени.

# ЛИТЕРАТУРА

1. Документация по C# [Электронный ресурс] -https://learn.microsoft.com/ru-ru/dotnet/csharp/
2. Документация по Windows Forms для .NET 7 [Электронный ресурс] - https://learn.microsoft.com/ru-ru/dotnet/desktop/winforms/?view=netdesktop-8.0
3. Документация по Npgsql [Электронный ресурс] - https://www.npgsql.org/doc/index.html
4. Что такое построение архитектурных диаграмм? [Электронный ресурс]

https://aws.amazon.com/ru/what-is/architecture-diagramming/

1. Функциональные диаграммы - Технология разработки программного обеспечения [Электронный ресурс] -

https://studref.com/311808/informatika/funktsionalnye\_diagrammy

# ПРИЛОЖЕНИЕ А

**Файл «Users.cs»**

using Npgsql;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace TimeTrackingApp

{

public class Activities

{

public int Id { get; set; }

public string? Name { get; set; }

public DateTime Start { get; set; }

public DateTime End { get; set; }

public void SetId()

{

var DB = new DataBase();

int id = 0;

var command = new NpgsqlCommand("SELECT activity\_id FROM activities" +

" WHERE activity\_name = @activityname ORDER BY activity\_id DESC LIMIT 1", DB.Connection);

command.Parameters.Add("@activityname", NpgsqlTypes.NpgsqlDbType.Varchar).Value = Name;

DB.OpenConnection();

NpgsqlDataReader reader = command.ExecuteReader();

if (reader.Read())

id = reader.GetInt32(0);

DB.CloseConnection();

Id = id;

}

}

}

**Файл «Activities.cs»**

using Npgsql;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace TimeTrackingApp

{

public class Activities

{

public int Id { get; set; }

public string? Name { get; set; }

public DateTime Start { get; set; }

public DateTime End { get; set; }

public void SetId()

{

var DB = new DataBase();

int id = 0;

var command = new NpgsqlCommand("SELECT activity\_id FROM activities" +

" WHERE activity\_name = @activityname ORDER BY activity\_id DESC LIMIT 1", DB.Connection);

command.Parameters.Add("@activityname", NpgsqlTypes.NpgsqlDbType.Varchar).Value = Name;

DB.OpenConnection();

NpgsqlDataReader reader = command.ExecuteReader();

if (reader.Read())

id = reader.GetInt32(0);

DB.CloseConnection();

Id = id;

}

}

}

**Файл «Categories.cs»**

using Npgsql;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace TimeTrackingApp

{

public class Categories

{

public int Id { get; set; }

public string? Name { get; set; }

public void SetId()

{

var DB = new DataBase();

int id = 0;

var command = new NpgsqlCommand("SELECT category\_id FROM categories" +

" WHERE category\_name = @categoryname", DB.Connection);

command.Parameters.Add("@categoryname", NpgsqlTypes.NpgsqlDbType.Varchar).Value = Name;

DB.OpenConnection();

NpgsqlDataReader reader = command.ExecuteReader();

if (reader.Read())

id = reader.GetInt32(0);

DB.CloseConnection();

Id = id;

}

public List<string> GetAllNames()

{

var categories = new List<string>();

var DB = new DataBase();

var command = new NpgsqlCommand("SELECT category\_name FROM categories", DB.Connection);

DB.OpenConnection();

NpgsqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

categories.Add(reader.GetString(0));

}

DB.CloseConnection();

return categories;

}

}

}

**Файл «DataBase.cs»**

using Npgsql;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace TimeTrackingApp

{

public class DataBase

{

NpgsqlConnection \_connection = new NpgsqlConnection("Host=78.153.5.230;Port=8595;Username=dbekbulatov;Password=XNSQTXcGHk;Database=timetrackingapp");

public NpgsqlConnection Connection { get { return \_connection; } }

public bool ConnectionCheck()

{

try

{

\_connection.Open();

}

catch

{

MessageBox.Show("Произошла ошибка при соединении с сервером", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return false;

}

\_connection.Close();

return true;

}

public void OpenConnection()

{

try

{

\_connection.Open();

}

catch

{

MessageBox.Show("Произошла ошибка при соединении с сервером", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

public void CloseConnection()

{

try

{

\_connection.Close();

}

catch

{

MessageBox.Show("Произошла ошибка при соединении с сервером", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

**Файл «Program.cs»**

global using global::System;

global using global::System.Collections.Generic;

global using global::System.Drawing;

global using global::System.IO;

global using global::System.Linq;

global using global::System.Net.Http;

global using global::System.Threading;

global using global::System.Threading.Tasks;

global using global::System.Windows.Forms;

namespace TimeTrackingApp

{

internal static class Program

{

[STAThread]

static void Main()

{

ApplicationConfiguration.Initialize();

var loginForm = new LoginForm(new Users());

loginForm.ShowDialog();

if (loginForm.DialogResult == DialogResult.OK)

{

var mainForm = new MainForm(loginForm.User);

Application.Run(mainForm);

}

}

}

}

**Файл «ChangeLoginForm.cs»**

using Microsoft.VisualBasic.ApplicationServices;

using Npgsql;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace TimeTrackingApp

{

public partial class ChangeLoginForm : Form

{

string \_initialNewLoginBoxText = "Введите новое имя пользователя";

public Users User { get; set; }

public ChangeLoginForm(Users user)

{

User = user;

InitializeComponent();

}

private void ChangeNameForm\_Load(object sender, EventArgs e)

{

NewLoginBox.Text = \_initialNewLoginBoxText;

}

private void ChangeLoginButton\_Click(object sender, EventArgs e)

{

if (User.DoesExist(NewLoginBox.Text))

{

MessageBox.Show("Пользователь с данным логином существует", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

User.ChangeName(NewLoginBox.Text);

Close();

}

private void NewLoginBox\_Enter(object sender, EventArgs e)

{

if (NewLoginBox.Text == \_initialNewLoginBoxText)

NewLoginBox.Text = string.Empty;

}

private void NewLoginBox\_Leave(object sender, EventArgs e)

{

if (NewLoginBox.Text == string.Empty)

NewLoginBox.Text = \_initialNewLoginBoxText;

}

}

}

**Файл «ChooseCategoryForm.cs»**

using Npgsql;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace TimeTrackingApp

{

public partial class ChooseCategoryForm : Form

{

public Categories Category { get; set; }

public Users User { get; set; }

public Activities Activity { get; set; }

public ChooseCategoryForm(Categories category, Users user, Activities activity)

{

Category = category;

User = user;

Activity = activity;

InitializeComponent();

}

private void ChooseCategoryForm\_Load(object sender, EventArgs e)

{

var categories = Category.GetAllNames();

foreach (var category in categories)

{

categoryBox.Items.Add(category);

}

}

private void chooseButton\_Click(object sender, EventArgs e)

{

Category.Name = categoryBox.Text;

if (Category.Name != null)

{

Category.SetId();

var DB = new DataBase();

var command = new NpgsqlCommand("CALL add\_category\_id(@categoryid, @activityid); " +

"CALL insert\_users\_categories(@userid, @categoryid)", DB.Connection);

command.Parameters.Add("@categoryid", NpgsqlTypes.NpgsqlDbType.Integer).Value = Category.Id;

command.Parameters.Add("@userid", NpgsqlTypes.NpgsqlDbType.Integer).Value = User.Id;

command.Parameters.Add("@activityid", NpgsqlTypes.NpgsqlDbType.Integer).Value = Activity.Id;

DB.OpenConnection();

command.ExecuteNonQuery();

DB.CloseConnection();

Close();

}

}

private void ChooseCategory\_FormClosing(object sender, FormClosingEventArgs e)

{

if (Category.Name == null)

{

MessageBox.Show("Выберите категорию", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

e.Cancel = true;

}

}

}

}

**Файл «ExportForm.cs»**

using Aspose.Cells;

using Npgsql;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace TimeTrackingApp

{

public partial class ExportForm : Form

{

public Users User { get; set; }

public ExportForm(Users user)

{

User = user;

InitializeComponent();

}

private void monthCalendar\_DateChanged(object sender, DateRangeEventArgs e)

{

firstDateBox.Text = monthCalendar.SelectionStart.ToShortDateString();

secondDateBox.Text = monthCalendar.SelectionEnd.ToShortDateString();

}

private void saveButton\_Click(object sender, EventArgs e)

{

if (IsFileNameBoxEmpty()) return;

if (!IsFileNameValid(fileNameBox.Text)) return;

if (AreDateBoxesEmpty()) return;

var path = Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.Desktop), $"{fileNameBox.Text}.xlsx");

var startdate = DateOnly.Parse(firstDateBox.Text);

var enddate = DateOnly.Parse(secondDateBox.Text);

var datatable = GetData(startdate, enddate);

var workbook = CreateWorkbook(datatable);

if (datatable.Rows.Count > 0)

{

try

{

workbook.Save(path);

MessageBox.Show("Файл сохранен", "Сообщение", MessageBoxButtons.OK, MessageBoxIcon.Information);

Close();

}

catch

{

MessageBox.Show("Произошла ошибка", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

else

{

var dialogresult = MessageBox.Show("За данный промежуток времени не было выполнено никаких задач. " +

"Хотите ли вы создать Excel-файл?", "Сообщение", MessageBoxButtons.YesNo, MessageBoxIcon.Question);

if (dialogresult == DialogResult.Yes)

{

try

{

workbook.Save(path);

MessageBox.Show("Файл сохранен", "Сообщение", MessageBoxButtons.OK, MessageBoxIcon.Information);

Close();

}

catch

{

MessageBox.Show("Произошла ошибка", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

private Workbook CreateWorkbook(DataTable datatable)

{

var workbook = new Workbook();

var worksheet = workbook.Worksheets[0];

worksheet.Name = "Отчет";

var tableoptions = new ImportTableOptions { IsFieldNameShown = true };

worksheet.Cells.ImportData(datatable, 0, 0, tableoptions);

worksheet.Cells.Columns[0].Width = 40;

worksheet.Cells.Columns[1].Width = 40;

worksheet.Cells.Columns[2].Width = 40;

return workbook;

}

private DataTable GetData(DateOnly startdate, DateOnly enddate)

{

var table = new DataTable();

var firstColumnName = "Задача";

var secondColumnName = "Категория";

var thirdColumnName = "Время выполнения";

var DB = new DataBase();

var command = new NpgsqlCommand("SELECT \* FROM " +

"get\_activities\_info\_for\_export(@userid, @startdate, @enddate)", DB.Connection);

command.Parameters.Add("@userid", NpgsqlTypes.NpgsqlDbType.Integer).Value = User.Id;

command.Parameters.Add("@startdate", NpgsqlTypes.NpgsqlDbType.Date).Value = startdate;

command.Parameters.Add("@enddate", NpgsqlTypes.NpgsqlDbType.Date).Value = enddate;

var adapter = new NpgsqlDataAdapter();

adapter.SelectCommand = command;

adapter.Fill(table);

table.Columns[0].ColumnName = firstColumnName;

table.Columns[1].ColumnName = secondColumnName;

table.Columns[2].ColumnName = thirdColumnName;

return table;

}

private bool IsFileNameBoxEmpty()

{

if (fileNameBox.Text == string.Empty)

{

MessageBox.Show("Введите имя файла", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return true;

}

else

return false;

}

private bool AreDateBoxesEmpty()

{

if (firstDateBox.Text == string.Empty && secondDateBox.Text == string.Empty)

{

MessageBox.Show("Временной промежуток не указан", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return true;

}

else

return false;

}

private bool IsFileNameValid(string filename)

{

string[] illegalFileNames = { "CON", "PRN", "AUX", "NUL", "COM0", "COM1", "COM2", "COM3", "COM4", "COM5",

"COM5", "COM6", "COM7", "COM8", "COM9", "COMSCSI", "LPT0", "LPT1", "LPT2", "LPT3", "LPT4", "LPT5", "LPT6",

"LPT7", "LPT8", "LPT9", "LPTSCSI"};

foreach (var chr in Path.GetInvalidFileNameChars())

{

if (filename.Contains(chr))

{

MessageBox.Show($"Имя файла содержит запрещенные символы", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return false;

}

}

foreach (var fname in illegalFileNames)

{

if (filename == fname)

{

MessageBox.Show($"Недопустимое имя файла", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return false;

}

}

return true;

}

}

}

**Файл «LoginForm.cs»**

using Npgsql;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Security.Cryptography;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace TimeTrackingApp

{

public partial class LoginForm : Form

{

int \_numberOfAttempts = 5;

string \_initialLoginBoxText = "Пользователь";

string \_initialPassBoxText = "\*\*\*\*\*\*\*\*\*\*\*\*\*";

public Users User { get; set; }

public LoginForm(Users user)

{

User = user;

InitializeComponent();

}

private void LoginForm\_Load(object sender, EventArgs e)

{

LoginBox.Text = \_initialLoginBoxText;

PassBox.Text = \_initialPassBoxText;

}

private void AuthButton\_Click(object sender, EventArgs e)

{

User.Login = LoginBox.Text;

var enteredPassword = PassBox.Text;

if (!User.DoesExist(User.Login))

{

MessageBox.Show("Пользователя с таким именем не существует",

"Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

LoginFail();

return;

}

var DB = new DataBase();

var command = new NpgsqlCommand("SELECT user\_password FROM users" +

" WHERE user\_login = @userlogin", DB.Connection);

command.Parameters.Add("@userlogin", NpgsqlTypes.NpgsqlDbType.Varchar).Value = User.Login;

DB.OpenConnection();

NpgsqlDataReader reader = command.ExecuteReader();

reader.Read();

var dbData = reader.GetString(0).Split("$");

DB.CloseConnection();

var passwordHash = dbData[0];

var salt = dbData[1];

if (CheckPassword(enteredPassword, salt, passwordHash))

DialogResult = DialogResult.OK;

else

{

LoginFail();

MessageBox.Show("Неверный пароль", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

if (\_numberOfAttempts == 0)

DialogResult = DialogResult.Cancel;

}

private void LoginFail()

{

LoginBox.Text = \_initialLoginBoxText;

PassBox.Text = \_initialPassBoxText;

PassBox.UseSystemPasswordChar = false;

\_numberOfAttempts -= 1;

WarnMessage.Text = $"Осталось попыток {\_numberOfAttempts}";

}

private void RegisterButton\_Click(object sender, EventArgs e)

{

var registerForm = new RegisterForm();

registerForm.ShowDialog();

}

private void LoginBox\_Enter(object sender, EventArgs e)

{

if (LoginBox.Text == \_initialLoginBoxText)

LoginBox.Text = string.Empty;

}

private void LoginBox\_Leave(object sender, EventArgs e)

{

if (LoginBox.Text == string.Empty)

LoginBox.Text = \_initialLoginBoxText;

}

private void PassBox\_Enter(object sender, EventArgs e)

{

if (PassBox.Text == \_initialPassBoxText)

{

PassBox.Text = string.Empty;

PassBox.UseSystemPasswordChar = true;

}

}

private void PassBox\_Leave(object sender, EventArgs e)

{

if (PassBox.Text == string.Empty)

{

PassBox.Text = \_initialPassBoxText;

PassBox.UseSystemPasswordChar = false;

}

}

private void LoginForm\_FormClosed(object sender, FormClosedEventArgs e)

{

if (User.Login != null)

User.SetId();

}

private static byte[] GenerateSHA256Hash(string password, string salt)

{

var saltedPassword = string.Concat(password, salt);

var saltedPasswordBytes = Encoding.UTF8.GetBytes(saltedPassword);

var sha256 = SHA256.Create();

return sha256.ComputeHash(saltedPasswordBytes);

}

private static bool CheckPassword(string enteredPassword, string salt, string passwordHash)

{

var enteredPasswordHash = GenerateSHA256Hash(enteredPassword, salt);

var hashInString = BitConverter.ToString(enteredPasswordHash).Replace("-", String.Empty).ToLower();

if (hashInString.SequenceEqual(passwordHash))

return true;

else

return false;

}

}

}

**Файл «MainForm.cs»**

using Aspose.Cells;

using Npgsql;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Configuration;

using System.Data;

using System.Diagnostics;

using System.Drawing;

using System.Linq;

using System.Net.Http.Headers;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace TimeTrackingApp

{

public partial class MainForm : Form

{

int \_ticks;

public Users User { get; set; }

public Categories Category = new();

public Activities Activity = new();

public MainForm(Users user)

{

User = user;

InitializeComponent();

}

private void StartTimerButton\_Click(object sender, EventArgs e)

{

if (timer.Enabled == false)

{

\_ticks = 0;

TimerLabel.Text = "00:00:00";

var nameActivityForm = new NameActivityForm(Activity);

nameActivityForm.ShowDialog();

}

timer.Start();

}

private void timer1\_Tick(object sender, EventArgs e)

{

\_ticks++;

Activity.End = Activity.Start;

Activity.End = Activity.End.AddSeconds(\_ticks);

var difference = Activity.End.TimeOfDay - Activity.Start.TimeOfDay;

TimerLabel.Text = string.Format("{0:00}:{1:00}:{2:00}", difference.Hours, difference.Minutes, difference.Seconds);

}

private void StopTimerButton\_Click(object sender, EventArgs e)

{

if (timer.Enabled == true)

{

timer.Stop();

var DB = new DataBase();

if (!DB.ConnectionCheck()) return;

var command = new NpgsqlCommand("CALL update\_activities(@activityid, @datetime); " +

"CALL insert\_users\_activities(@userid, @activityid)", DB.Connection);

command.Parameters.Add("@activityid", NpgsqlTypes.NpgsqlDbType.Integer).Value = Activity.Id;

command.Parameters.Add("@userid", NpgsqlTypes.NpgsqlDbType.Integer).Value = User.Id;

command.Parameters.Add("@datetime", NpgsqlTypes.NpgsqlDbType.Timestamp).Value = Activity.End;

DB.OpenConnection();

command.ExecuteNonQuery();

DB.CloseConnection();

var chooseCategory = new ChooseCategoryForm(Category, User, Activity);

chooseCategory.ShowDialog();

}

}

private void MainForm\_Load(object sender, EventArgs e)

{

InitializeDataGridView();

}

private void InitializeDataGridView()

{

dataGridView.DataSource = GetActivitiesInfo();

dataGridView.AllowUserToAddRows = false;

}

private DataTable GetActivitiesInfo()

{

var DB = new DataBase();

var firstColumnName = "Задача";

var secondColumnName = "Категория";

var thirdColumnName = "Время выполнения";

var command = new NpgsqlCommand("SELECT \* FROM get\_activities\_info(@userid)", DB.Connection);

command.Parameters.Add("@userid", NpgsqlTypes.NpgsqlDbType.Integer).Value = User.Id;

var adapter = new NpgsqlDataAdapter();

adapter.SelectCommand = command;

var table = new DataTable();

adapter.Fill(table);

table.Columns[0].ColumnName = firstColumnName;

table.Columns[1].ColumnName = secondColumnName;

table.Columns[2].ColumnName = thirdColumnName;

return table;

}

private void refreshButton\_Click(object sender, EventArgs e)

{

InitializeDataGridView();

}

private void exportButton\_Click(object sender, EventArgs e)

{

var exportForm = new ExportForm(User);

exportForm.ShowDialog();

}

private void ChangeLogin\_Click(object sender, EventArgs e)

{

var changeNameForm = new ChangeLoginForm(User);

changeNameForm.ShowDialog();

}

private void Exit\_Click(object sender, EventArgs e)

{

var loginForm = new LoginForm(User);

loginForm.ShowDialog();

InitializeDataGridView();

TimerLabel.Text = "00:00:00";

}

}

}

Файл «NameActivityForm.cs»

using Npgsql;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace TimeTrackingApp

{

public partial class NameActivityForm : Form

{

public Activities Activity { get; set; }

public NameActivityForm(Activities activity)

{

Activity = activity;

InitializeComponent();

}

private void ChooseButton\_Click(object sender, EventArgs e)

{

Activity.Start = DateTime.Now;

Activity.Name = ActivityNameBox.Text;

if (Activity.Name != null || Activity.Name != string.Empty)

{

var DB = new DataBase();

var command = new NpgsqlCommand("CALL insert\_activities(@activityname, @datetime)", DB.Connection);

command.Parameters.Add("@activityname", NpgsqlTypes.NpgsqlDbType.Varchar).Value = Activity.Name;

command.Parameters.Add("@datetime", NpgsqlTypes.NpgsqlDbType.Timestamp).Value = Activity.Start;

DB.OpenConnection();

command.ExecuteNonQuery();

DB.CloseConnection();

Activity.SetId();

Close();

}

else

MessageBox.Show("Сформулируйте задачу", "Предупреждение", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

private void NameActivityForm\_FormClosing(object sender, FormClosingEventArgs e)

{

if (Activity.Name == null)

{

MessageBox.Show("Введите название задачи", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

e.Cancel = true;

}

}

}

}

**Файл «RegisterForm.cs»**

using Npgsql;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Security.Cryptography;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace TimeTrackingApp

{

public partial class RegisterForm : Form

{

string \_initialLoginBoxText = "Введите логин";

string \_initialPassBoxText = "Придумайте пароль";

string \_initialRepPassBoxText = "Повторите введенный пароль";

public Users User = new ();

public RegisterForm()

{

InitializeComponent();

}

private void RegisterForm\_Load(object sender, EventArgs e)

{

LoginBox.Text = \_initialLoginBoxText;

PassBox.Text = \_initialPassBoxText;

RepPassBox.Text = \_initialRepPassBoxText;

}

private void RegisterButton\_Click(object sender, EventArgs e)

{

if (IsLoginFieldEmpty()) return;

if (IsPasswordFieldEmpty()) return;

User.Login = LoginBox.Text;

var password = PassBox.Text;

var repPassword = RepPassBox.Text;

if (!PasswordValidation(PassBox.Text)) return;

if (!DoPasswordsMatch(password, repPassword)) return;

if (User.DoesExist(User.Login))

{

MessageBox.Show("Пользователь с данным логином уже существует", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

var salt = GenerateSalt();

var hashedPassword = GenerateSHA256Hash(password, salt);

var passwordInDB = $"{ConvertByteArrToHexString(hashedPassword)}${salt}";

var DB = new DataBase();

var command = new NpgsqlCommand("INSERT INTO users(user\_login, user\_password)" +

" VALUES (@userlogin, @userpass)", DB.Connection);

command.Parameters.Add("@userlogin", NpgsqlTypes.NpgsqlDbType.Varchar).Value = User.Login;

command.Parameters.Add("@userpass", NpgsqlTypes.NpgsqlDbType.Varchar).Value = passwordInDB;

DB.OpenConnection();

if (command.ExecuteNonQuery() == 1)

{

MessageBox.Show("Аккаунт создан", "Сообщение", MessageBoxButtons.OK, MessageBoxIcon.Information);

Close();

}

else

MessageBox.Show("Аккаунт не был создан", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

DB.CloseConnection();

}

private bool DoPasswordsMatch(string pass1, string pass2)

{

if (pass1 != pass2)

{

MessageBox.Show("Пароли не совпадают", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return false;

}

else

return true;

}

private bool IsLoginFieldEmpty()

{

if (LoginBox.Text == \_initialLoginBoxText)

{

MessageBox.Show("Не введен логин", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return true;

}

else

return false;

}

private bool IsPasswordFieldEmpty()

{

if (PassBox.Text == \_initialPassBoxText)

{

MessageBox.Show("Не введен пароль", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return true;

}

else

return false;

}

private bool PasswordValidation(string password)

{

if (IsPasswordStrong(password))

return true;

else if (password.Length < 10)

MessageBox.Show("Длина пароля должна быть более десяти символов", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

else

MessageBox.Show("Пароль должен являться сочетанием букв верхнего регистра, " +

"букв нижнего регистра и цифр", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return false;

}

private bool IsPasswordStrong(string password)

{

bool digits = false, uppercase = false, lowercase = false;

foreach (char c in password)

{

if (char.IsDigit(c) && digits == false) digits = true;

else if (char.IsUpper(c) && uppercase == false) uppercase = true;

else if (char.IsLower(c) && lowercase == false) lowercase = true;

}

if (password.Length >= 10 && digits && uppercase && lowercase)

return true;

else

return false;

}

private string ConvertByteArrToHexString(byte[] arr)

{

return BitConverter.ToString(arr).Replace("-", string.Empty).ToLower();

}

private string GenerateSalt()

{

var rand = new Random();

var salt = new byte[rand.Next(10, 24)];

rand.NextBytes(salt);

return string.Concat(ConvertByteArrToHexString(salt));

}

private byte[] GenerateSHA256Hash(string password, string salt)

{

var saltedPassword = string.Concat(password, salt);

var saltedPasswordBytes = Encoding.UTF8.GetBytes(saltedPassword);

var sha256 = SHA256.Create();

return sha256.ComputeHash(saltedPasswordBytes);

}

private void LoginBox\_Enter(object sender, EventArgs e)

{

if (LoginBox.Text == \_initialLoginBoxText)

LoginBox.Text = string.Empty;

}

private void LoginBox\_Leave(object sender, EventArgs e)

{

if (LoginBox.Text == string.Empty)

LoginBox.Text = \_initialLoginBoxText;

}

private void PassBox\_Enter(object sender, EventArgs e)

{

if (PassBox.Text == \_initialPassBoxText)

{

PassBox.Text = string.Empty;

PassBox.UseSystemPasswordChar = true;

}

}

private void PassBox\_Leave(object sender, EventArgs e)

{

if (PassBox.Text == string.Empty)

{

PassBox.Text = \_initialPassBoxText;

PassBox.UseSystemPasswordChar = false;

}

}

private void RepPassBox\_Enter(object sender, EventArgs e)

{

if (RepPassBox.Text == \_initialRepPassBoxText)

{

RepPassBox.Text = string.Empty;

RepPassBox.UseSystemPasswordChar = true;

}

}

private void RepPassBox\_Leave(object sender, EventArgs e)

{

if (RepPassBox.Text == string.Empty)

{

RepPassBox.Text = \_initialRepPassBoxText;

RepPassBox.UseSystemPasswordChar = false;

}

}

}

}