**Tehtävä 24 – Oma ohjelma**Tein tämän ohjelman, jotta suunnittelu ja aikatauluttaminen olisi helpompaa ja tehdä se digitaalisessa muodossa. Jokapäiväisessä elämässäni käytän päivittäin paperisia kalentereita ja muistikirjoja, ja minusta on tärkeää kirjoittaa päivän tehtävät tarkasti. Se helpottaa ja auttaa pitämään asiat järjestyksessä. Tällä ohjelmalla haluan tuoda saman hyödyn digitaaliseen muotoon, jotta sitä on kätevä käyttää suoraan tietokoneelta. Tämä ohjelma sopii ihmisille, jotka haluavat suunnitella päivänsä sähköisessä muodossa, kuten minä, ja joille suunnittelu on tärkeää. Se sopii myös pienille yrityksille tai muille käyttäjille, jotka haluavat yksinkertaisen ja toimivan työkalun aikataulutusta varten.

Tämä ohjelma on yksinkertainen Windows Forms -pohjainen sovellus, joka käyttää **MySQL**-tietokantaa tietojen tallentamiseen. Tässä on ohjelman päätoiminnot ja kuvaus:

**Ohjelman kuvaus**

Ohjelmalla voi hallita ja näyttää viikkokohtaista aikataulua, tallentaa ja ladata tietoja MySQL-tietokannasta sekä tallentaa ja avata tietoja tekstitiedostosta.

**Komponentit:**

DataGridView: Näyttää ajan ja tapahtumat viikon jokaiselle päivälle.

MySQL-tietokanta: Lataa ja tallenna tietoja tietokantaan.

**Ohjelman päätoiminnot**

Alustus: Kun ohjelma käynnistyy, se luo rivit kullekin vuorokauden tunnille (kello 6:00–5:00) ja määrittää taulukon ulkoasun.

**Tietojen lataus:**

* MySQL-tietokannasta SQL-kyselyllä.
* Tekstitiedostosta, jos tiedosto on olemassa.

**Tietojen tallennus:** Ohjelmalla voi tallentaa muutokset tietokantaan ja viedä tiedot tekstitiedostoon, jossa sarakkeet on erotettu tabulaattoreilla.

**Tietojen tyhjennys:** Ohjelmassa voi tyhjentää tapahtumat säilyttäen kuitenkin ajan.

**Ajan päivitys:** Ohjelma näyttää nykyisen ajan, joka päivittyy säännöllisesti.

**Sulkeminen:** Kun ohjelma suljetaan, se katkaisee yhteyden tietokantaan ja lopettaa suorituksen.

**Tässä on kuvia:**

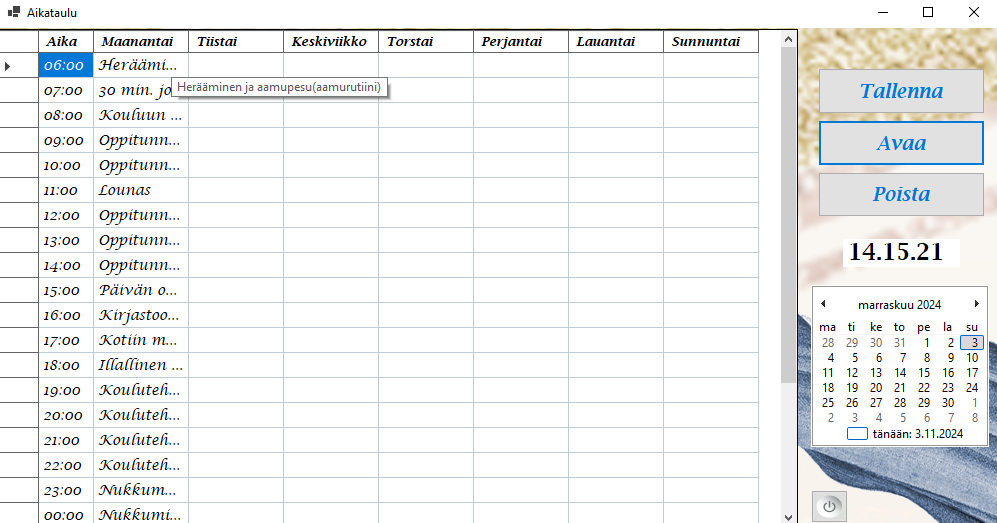
A white rectangular frame with blue leaves and black text

Description automatically generated

A blank lined paper with writing

Description automatically generated

**Tapahtumatietojen syöttäminen ja niin edelleen:**



**Tallentaminen:**

A screenshot of a computer

Description automatically generated

**Syötettyjen tietojen tallentaminen SQL-tietokantaan:**

A screenshot of a computer

Description automatically generated

**Syötettyjen tietojen tallentaminen tekstitiedostoon:**

A screenshot of a computer

Description automatically generated

A screenshot of a calendar

Description automatically generatedA screen shot of a computer

Description automatically generated **Poistumispainike ohjelmasta.**

**Kalenteri ja aika helpottavat työskentelyä.**

**Tässä on koodi:**

using System;

using System.Data;

using System.Drawing;

using System.IO;

using System.Windows.Forms;

using MySql.Data.MySqlClient;

namespace MinunOhjelmani

{

public partial class MainForm : Form

{

private MySqlConnection connection; // Yhteys MySQL-tietokantaan

public MainForm()

{

InitializeComponent(); // Alustaa komponentit

}

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

{

int aloitusaika = 6; // Aloitusaika (6:00)

int lopetusaika = 5; // Lopetusaika (5:00 seuraavana päivänä)

int rivienMaara = 24 - aloitusaika + lopetusaika + 1; // Rivien määrä

for (int i = 0; i < rivienMaara; i++)

{

int indeksi = dataGridView1.Rows.Add(); // Lisää uusi rivi

int tunti = (aloitusaika + i) % 24; // Laskee tunnin

dataGridView1.Rows[indeksi].Cells[0].Value = $"{tunti:00}:00"; // Asettaa kellonajan

}

// Asettaa fontin kaikille DataGridView:n soluilla

dataGridView1.DefaultCellStyle.Font = new Font("Lucida Calligraphy", 10F);

// Alustaa yhteyden tietokantaan

string connectionString = "datasource=localhost;port=3306;username=root;password=;database=aikataulu\_db;SSL Mode=None;";

connection = new MySqlConnection(connectionString);

try

{

connection.Open(); // Avaa yhteys

LoadDataFromDatabase(); // Lataa tiedot tietokannasta

}

catch (Exception ex)

{

MessageBox.Show($"Virhe tietokantayhteydessä: {ex.Message}"); // Virheviesti

}

// Lataa tiedot tiedostosta, jos se on olemassa

string filePath = "tieto.txt"; // Tiedoston polku

if (File.Exists(filePath))

{

LoadDataFromFile(filePath); // Lataa tiedot tiedostosta

}

}

private void LoadDataFromDatabase()

{

// SQL-kysely tietojen hakemiseksi

string query = "SELECT event\_time, maanantai, tiistai, keskiviikko, torstai, perjantai, lauantai, sunnuntai FROM aikataulu";

using (MySqlCommand command = new MySqlCommand(query, connection))

{

using (MySqlDataReader reader = command.ExecuteReader())

{

while (reader.Read()) // Lue rivit tietokannasta

{

int newRowIndex = dataGridView1.Rows.Add(); // Lisää uusi rivi DataGridView:iin

dataGridView1.Rows[newRowIndex].Cells[0].Value = reader["event\_time"].ToString(); // Aseta aika

dataGridView1.Rows[newRowIndex].Cells[1].Value = reader["maanantai"].ToString(); // Aseta maanantai

dataGridView1.Rows[newRowIndex].Cells[2].Value = reader["tiistai"].ToString(); // Aseta tiistai

dataGridView1.Rows[newRowIndex].Cells[3].Value = reader["keskiviikko"].ToString(); // Aseta keskiviikko

dataGridView1.Rows[newRowIndex].Cells[4].Value = reader["torstai"].ToString(); // Aseta torstai

dataGridView1.Rows[newRowIndex].Cells[5].Value = reader["perjantai"].ToString(); // Aseta perjantai

dataGridView1.Rows[newRowIndex].Cells[6].Value = reader["lauantai"].ToString(); // Aseta lauantai

dataGridView1.Rows[newRowIndex].Cells[7].Value = reader["sunnuntai"].ToString(); // Aseta sunnuntai

}

}

}

}

private void SaveDataToDatabase()

{

try

{

// Poista vanhat tiedot taulusta

string deleteQuery = "DELETE FROM aikataulu";

using (MySqlCommand deleteCommand = new MySqlCommand(deleteQuery, connection))

{

deleteCommand.ExecuteNonQuery(); // Suorita poistaminen

}

// Lisää uudet tiedot

string insertQuery = "INSERT INTO aikataulu (event\_time, maanantai, tiistai, keskiviikko, torstai, perjantai, lauantai, sunnuntai) VALUES (@event\_time, @maanantai, @tiistai, @keskiviikko, @torstai, @perjantai, @lauantai, @sunnuntai)";

using (MySqlCommand command = new MySqlCommand(insertQuery, connection))

{

foreach (DataGridViewRow row in dataGridView1.Rows)

{

if (row.IsNewRow) continue; // Ohita tyhjät rivit

// Tyhjennä vanhat parametrit ennen uusien lisäämistä

command.Parameters.Clear();

// Tarkista jokaisen solun arvo

command.Parameters.AddWithValue("@event\_time", row.Cells[0].Value == null ? "" : row.Cells[0].Value); // Aika

command.Parameters.AddWithValue("@maanantai", row.Cells[1].Value == null ? "" : row.Cells[1].Value); // Maanantai

command.Parameters.AddWithValue("@tiistai", row.Cells[2].Value == null ? "" : row.Cells[2].Value); // Tiistai

command.Parameters.AddWithValue("@keskiviikko", row.Cells[3].Value == null ? "" : row.Cells[3].Value);// Keskiviikko

command.Parameters.AddWithValue("@torstai", row.Cells[4].Value == null ? "" : row.Cells[4].Value); // Torstai

command.Parameters.AddWithValue("@perjantai", row.Cells[5].Value == null ? "" : row.Cells[5].Value); // Perjantai

command.Parameters.AddWithValue("@lauantai", row.Cells[6].Value == null ? "" : row.Cells[6].Value); // Lauantai

command.Parameters.AddWithValue("@sunnuntai", row.Cells[7].Value == null ? "" : row.Cells[7].Value); // Sunnuntai

command.ExecuteNonQuery(); // Suorita lisääminen

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Virhe tietojen tallentamisessa tietokantaan: {ex.Message}"); // Virheilmoitus

}

}

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

{

SaveDataToDatabase(); // Tallenna tiedot tietokantaan

using (SaveFileDialog saveFileDialog = new SaveFileDialog())

{

saveFileDialog.Filter = "Text files (\*.txt)|\*.txt|All files (\*.\*)|\*.\*"; // Tiedostotyypit

saveFileDialog.Title = "Tallenna tiedosto"; // Tallenna tiedoston otsikko

if (saveFileDialog.ShowDialog() == DialogResult.OK) // Tarkista, onko käyttäjä valinnut tiedoston

{

using (StreamWriter writer = new StreamWriter(saveFileDialog.FileName)) // Kirjoita tiedostoon

{

// Tallenna sarakkeiden otsikot

for (int i = 0; i < dataGridView1.Columns.Count; i++)

{

writer.Write(dataGridView1.Columns[i].HeaderText); // Kirjoita sarakkeen otsikko

if (i < dataGridView1.Columns.Count - 1)

writer.Write("\t"); // Erotin

}

writer.WriteLine(); // Uusi rivi

// Tallenna rivit

foreach (DataGridViewRow row in dataGridView1.Rows)

{

if (row.IsNewRow) continue; // Ohita tyhjät rivit

for (int i = 0; i < dataGridView1.Columns.Count; i++)

{

writer.Write(row.Cells[i].Value?.ToString() ?? ""); // Arvo tai tyhjää

if (i < dataGridView1.Columns.Count - 1)

writer.Write("\t"); // Erotin

}

writer.WriteLine(); // Uusi rivi

}

}

}

}

}

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

{

using (OpenFileDialog openFileDialog = new OpenFileDialog())

{

openFileDialog.InitialDirectory = "C:\\Users\\aalto\\source\\repos\\WinFormsAppMy"; // Tiedostojen kansio

openFileDialog.Filter = "Text files (\*.txt)|\*.txt|All files (\*.\*)|\*.\*"; // Tiedostotyypit

openFileDialog.Title = "Valitse tallennettu tiedosto"; // Valitse tiedoston otsikko

// Avaa dialogi tiedoston valitsemiseksi

if (openFileDialog.ShowDialog() == DialogResult.OK)

{

// Tyhjennä DataGridView ennen tietojen lataamista

dataGridView1.Rows.Clear(); // Tyhjennä rivit

LoadDataFromFile(openFileDialog.FileName); // Lataa valittu tiedosto

}

}

}

private void LoadDataFromFile(string filePath)

{

using (StreamReader reader = new StreamReader(filePath)) // Lue tiedostoa

{

string rivi; // Rivi muuttuja

bool firstLine = true; // Ensimmäinen rivi

while ((rivi = reader.ReadLine()) != null) // Lue rivit

{

if (firstLine)

{

firstLine = false; // Ohita ensimmäinen rivi otsikoilla

continue;

}

string[] values = rivi.Split('\t'); // Oletetaan, että käytetään tabulaatiota

int newRowIndex = dataGridView1.Rows.Add(); // Lisää uusi rivi

for (int i = 0; i < values.Length && i < dataGridView1.Columns.Count; i++)

{

dataGridView1.Rows[newRowIndex].Cells[i].Value = values[i]; // Aseta arvo

}

}

}

}

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

{

// Käy läpi kaikki rivit DataGridView:ssä

foreach (DataGridViewRow row in dataGridView1.Rows)

{

// Ohita tyhjät rivit

if (row.IsNewRow) continue;

// Tyhjennä solut, paitsi ensimmäinen (aika)

for (int i = 1; i < dataGridView1.Columns.Count; i++)

{

row.Cells[i].Value = string.Empty; // Tyhjennä solun arvo

}

}

}

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

{

// Päivitä nykyinen aika

AikaLB.Text = DateTime.Now.ToString("HH:mm:ss"); // Aseta aika labeliin

}

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

{

// Sulje yhteys tietokantaan

if (connection != null)

{

connection.Close(); // Sulje yhteys

}

Application.Exit(); // Sulje sovellus

}

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

{

Application.Exit(); // Sulje sovellus

}

private void dataGridView1\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

}

}