Министерство образования и науки Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра компьютерных систем в управлении и проектировании (КСУП)

ОТЧЕТ

К лабораторной работе №3 (Разработка пользовательского интерфейса)

по дисциплине «Системное программное обеспечение»

Студент

гр. 538 А. В. Вачиля

Принял:

доцент кафедры КСУП:

\_\_\_\_\_\_\_\_ Коцубинский В.П.

Дата: «\_\_\_»\_\_\_\_\_\_\_\_2020 г.

2020

Оглавление

[1 Цель и задачи лабораторной работы 3](#_Toc57324675)

[2 Ход лабораторной работы 4](#_Toc57324676)

[Главное окно пользовательского интерфейса MainForm 4](#_Toc57324677)

[Окно добавления и редактирования заметок EditForm 9](#_Toc57324678)

[Окно информации AboutForm 11](#_Toc57324679)

[История коммитов 11](#_Toc57324680)

[3 Заключение 12](#_Toc57324681)

1 Цель и задачи лабораторной работы

Цель работы: изучить разработку адаптивного пользовательского интерфейса десктоп-приложения.

Задачи:

1. Изучить использование фреймворка пользовательского интерфейса Windows Forms и его компоненты.

2. Освоить компоненты и свойства для создания адаптивного дизайна.

3. Освоить обработку событий компонентов Windows Forms.

4. Реализовать передачу данных между разрабатываемыми формами и пользовательскими элементами управления.

5. Реализовать защиту от некорректного ввода.

2 **Ход лабораторной работы**

В ходе лабораторной работы мы реализовали следующие типы Windows Forms:

**Главное окно пользовательского интерфейса MainForm**

Главное окно интерфейса MainForm, представлен на рисунке 2.1:

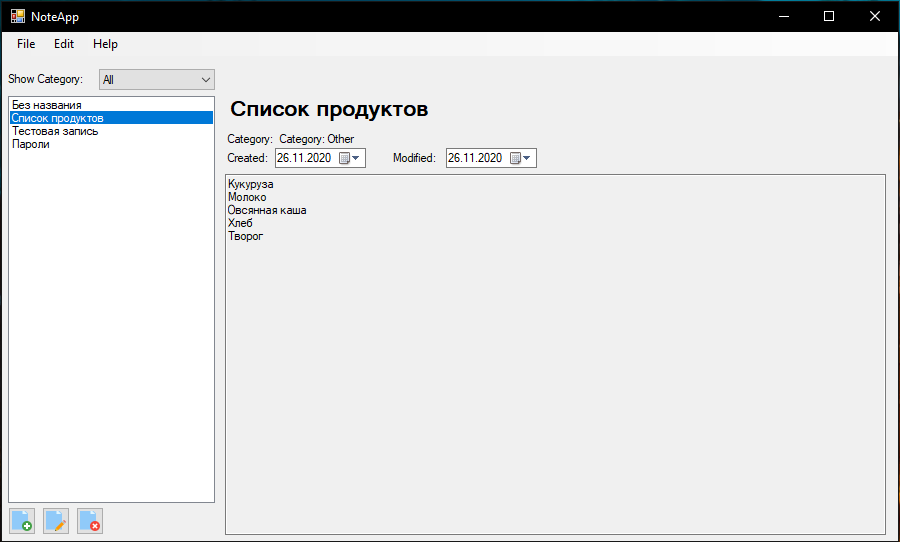


Рисунок 2.1 - Главное окно - MainForm

Также код, который осуществляет взаимодействие между представленным интерфейсом и разработанной раннее нами бизнес-логики:

Для начала заполним категориями CategorysComboBox:

CategorysComboBox.Items.Add("All");

foreach (Category element in Enum.GetValues(typeof(Category)))

{

CategorysComboBox.Items.Add(element);

}

Дальше создаем сущности allNotes и sortNotes. allNotes – это сущность отвечает за все заметки, а sortNotes – это та же allNotes, но отсортированнная по категориям. Код создания представлен ниже:

Project allNotes = new Project();

List<Note> sortNotes = new List<Note>();

После этого мы создаём функции создания новой записи -CreateNote(), функцию редактирования выбранной записи - ChangeNote() и функцию удаления выбранной записи - DeleteNote():

private void CreateNote()

{

//получаем выбранную заметку

Note newNote = new Note(string.Empty, NoteCategory.Work, string.Empty, DateTime.Now); //сама заметка

newNote.TimeLastChange = DateTime.Now;

EditForm inner = new EditForm(); //создаем форму

inner.Note = newNote; //передаем форме данные

inner.Text = ("Add Note");

//если было нажато Cancel завершаем выполнение обработчика

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

{

var updatedNote = inner.Note; //забираем измененные данные

//добавляем новую заметку в список

allNotes.Glossary.Add(updatedNote);

var changeTitle = updatedNote.NameNote;

FillTitleListBox();

TitlesListBox.SelectedItem = changeTitle;

ProjectManager.WritingToFile(allNotes);

}

}

private void ChangeNote()

{

//если заметка не выбрана завершаем выполнение обработчика(ничего не происходит при нажатии на "Изменить")

if (TitlesListBox.SelectedIndex == -1)

{

return;

}

//получаем выбранную заметку

var selectedIndex = TitlesListBox.SelectedIndex; //индекс нашей заметки в списке всех заметок allNotes

var selectedNote = sortNotes[selectedIndex]; //сама заметка

EditForm inner = new EditForm(); //создаем форму

inner.Note = selectedNote; //передаем форме данные

inner.Text = ("Edit Note");

//если было нажато Cancel завершаем выполнение обработчика

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

{

var updatedNote = inner.Note; //забираем измененные данные

//удалить и заменить старые данные

allNotes.Glossary.RemoveAt(allNotes.RealIndexes[selectedIndex]);

allNotes.Glossary.Add(updatedNote);

FillTitleListBox();

var changeTitle = updatedNote.NameNote;

TitlesListBox.SelectedItem = changeTitle;

ProjectManager.WritingToFile(allNotes);

}

}

private void DeleteNote()

{

//если заметка не выбрана завершаем выполнение обработчика(ничего не происходит при нажатии на "Удалить")

if (TitlesListBox.SelectedIndex == -1)

{

return;

}

if (MessageBox.Show("Вы уверены что хотите удалить заметку?", "Удаление", MessageBoxButtons.OKCancel, MessageBoxIcon.Question) == DialogResult.OK)

{

//получаем выбранную заметку

var selectedIndex = TitlesListBox.SelectedIndex; //индекс нашей заметки в списке всех заметок allNotes

TitlesListBox.Items.RemoveAt(selectedIndex);

allNotes.Glossary.RemoveAt(allNotes.RealIndexes[selectedIndex]);

allNotes.\_currentNote = -1;

FillTitleListBox();

NoteTextBox.Clear();

ProjectManager.WritingToFile(allNotes);

}

}

Мы создали эти функции для их вызова в соответствующих кнопках. Вот их код:

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

{

CreateNote();

}

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

{

ChangeNote();

}

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

{

DeleteNote();

}

Это отдельные кнопки, расположенные на форме. Помимо них есть ещё кнопки меню в верхней части панели. Не имеет смысла дублировать в их реализацию вызов тех же самых функций, поэтому через конструктор формы в свойствах кнопок меню мы поставили действия ранее созданных кнопок.

Так же в меню есть кнопка закрытия программы. Вот её код:

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

{

this.Close();

}

Функция FillTitleListbox(), упомянутая в CreateNote(), ChangeNote() и DeleteNote(), работает с названиями заметок и позволяет выводить их в ListBox:

private void FillTitleListBox()

{

if (allNotes != null)

{

TitlesListBox.Items.Clear();

sortNotes = allNotes.SortWithSelectionCategory(CategoriesComboBox.SelectedIndex);

{

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

{

TitlesListBox.Items.Add(sortNotes[i].NameNote);

}

}

}

}

Код для работы TitlesListBox:

private void TitlesListBox\_SelectedIndexChanged(object sender, EventArgs e)

{

//Проверяем и завершаем метод, если элемент не выбран

if (TitlesListBox.SelectedIndex == -1) return;

allNotes.\_currentNote = allNotes.RealIndexes[TitlesListBox.SelectedIndex];

ProjectManager.WritingToFile(allNotes);

NoteTextBox.Clear();

sortNotes = allNotes.SortWithSelectionCategory(CategoriesComboBox.SelectedIndex);

//Заполняем данными правую часть окна

TitleLabel.Text = sortNotes[TitlesListBox.SelectedIndex].NameNote;

CategoryLabel.Text = "Category: " + sortNotes[TitlesListBox.SelectedIndex].Category;

CreateDateTimePicker.Value = sortNotes[TitlesListBox.SelectedIndex].TimeCreate;

ChangeDateTimePicker.Value = sortNotes[TitlesListBox.SelectedIndex].TimeCreate;

NoteTextBox.Text = sortNotes[TitlesListBox.SelectedIndex].TextNote;

}

**Окно добавления и редактирования заметок EditForm**

EditForm – это окно для добавления или редактирования заметок, представлено на рисунке 2.2:

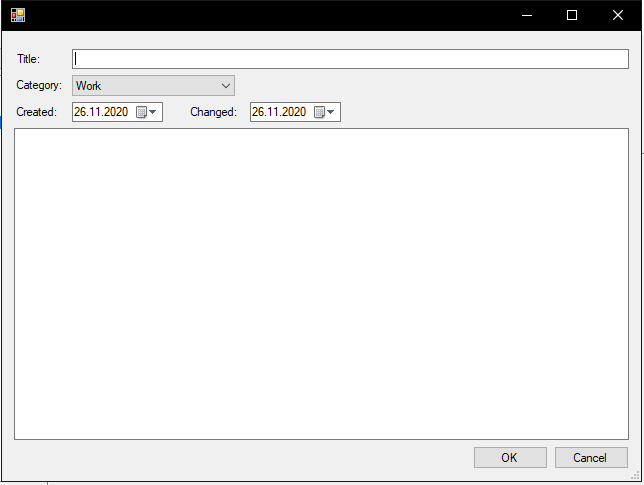


Рисунок 2.2 – Окно создания и редактирования записей EditForm.

Код, который осуществляет взаимодействие между представленным интерфейсом и разработанной раннее нами бизнес-логики:

Сначала создаем закрытую переменную типа Note и открытую функцию для передачи значения, созданной нами переменной в форму:

private Note \_note;

public Note Note

{

get { return \_note; }

set

{

\_note = value;

if (\_note != null)

{

//при отправлении заметки в это окно, автоматом заполнятся компоненты формы

TitleTextBox.Text = \_note.NameNote;

CategoriesComboBox.SelectedItem = \_note.Category;

DateTimeCreate.Value = \_note.TimeCreate;

DateTimeChange.Value = \_note.TimeLastChange;

NoteTextBox.Text = \_note.TextNote;

}

}

}

Далее передаем в CategoriesComboBox наши возможные категории:

foreach (Category element in Enum.GetValues(typeof(Category)))

{

CategoriesComboBox.Items.Add(element);

}

После этого создаём кнопку для сохранения созданной/изменённой записи OkButton. Так же вписываем код проверки заполненности полей. Если поля не заполнены, то выйдет сообщение и не позволит нам сохранить незаконченную запись.

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

{

if ((CategoriesComboBox.SelectedIndex == -1) || (NoteTextBox.Text == string.Empty) || (TitleTextBox.Text == string.Empty))

{

MessageBox.Show("Отсутствует текст/название/категория");

}

else

{

\_note.NameNote = TitleTextBox.Text;

\_note.TextNote = NoteTextBox.Text;

\_note.Category = (NoteCategory)CategoriesComboBox.Items[CategoriesComboBox.SelectedIndex];

\_note.TimeLastChange = DateTime.Now;

DialogResult = DialogResult.OK;

this.Close();

}

}

Так же нам необходимо создать кнопку отмены действия CancelButton:

private void CancelButton1\_Click(object sender, EventArgs e)

{

this.Close();

}

**Окно информации AboutForm**

Окно «About» представлена на рисунке 2.3:

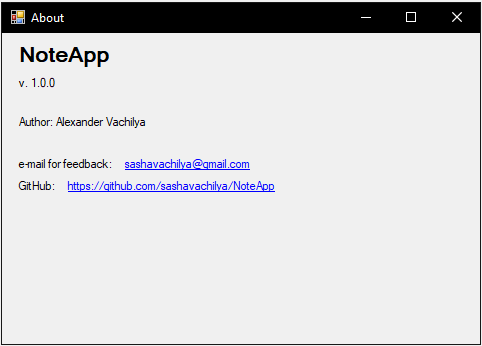


Рисунок 2.3 – Окно информации «About».

**История коммитов**

Просмотрим историю коммитов ветки develop, чтобы убедиться, что мы все сделали верно:



Рисунок 2.4 - История коммитов ветки develop.

3 Заключение

В ходе данной лабораторной работе мы изучили процесс создания пользовательского интерфейса для проекта логики из лабораторной работы №2 (процесс связывания форм и создания функций для обмена данными между формами). Этим самым мы получили первую версию нашей программы для создания и обработки записей «NoteApp». Цель лабораторной работы, а так же все поставленные задачи были успешно выполнены.