Министерство науки и высшего образования 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)

**РАЗРАБОТКА БИЗНЕС-ЛОГИКИ ПРИЛОЖЕНИЯ**

Отчет по лабораторной работе №2

по дисциплине «Новые технологии в программировании»

Вариант 1

Студент гр. 588-2

\_\_\_\_\_\_\_\_\_\_\_\_ Робканов.К.Д

(подпись)

« 9 » марта 2021г.

Руководитель старший научный сотрудник, доцент каф. КСУП

\_\_\_\_\_\_\_\_\_\_\_\_ Горяинов А.Е.

(подпись)

«\_\_\_» \_\_\_\_\_\_\_\_2021 г.

Томск 2021

**Содержание**

1 Введение……………………………………………………………………………3

2 Основная часть……………………………………………………………………..4

2.1 Требования к бизнес-логике……………………………………………...4

2.2 Классы проекта……………………………………………………………5

Заключение………………………………………………………………………….11

**1 Введение**

**Цель работы:** изучить типовые требования, предъявляемые к бизнес-логике приложения, получить умения разработки логики приложения с обеспечением данных требований.

**Задачи:**

1. Изучить требования и процесс разработки логики приложения.
2. Повторить синтаксис языка C# для разработки объектно-ориентированных программ.
3. Разработать классы, необходимые для работы логики приложения.
4. Обеспечить целостность данных классов с помощью свойств и генерации исключений.

**2 Основная часть**

**2.1 Требования к бизнес-логике**

Необходимо реализовать следующие типы данных:

— Перечисление «Категория заметки», содержащее значения «Работа», «Дом», «Здоровье и Спорт», «Люди», «Документы», «Финансы», «Разное».

— Класс «Заметка» с полями «Название», «Категория заметки», «Текст заметки», «Время создания», «Время последнего изменения». Название, категория и текст заметки доступны для изменений. Время создания инициализируется один раз при создании объекта «Заметка» и больше не модифицируется, доступна для чтения. Время последнего изменения меняется автоматически при изменении названия, категории или текста заметки. Название ограничено 50 символами. Название по умолчанию «Без названия». Допустимы заметки с одинаковыми названиями. Реализует интерфейс ICloneable.

— Класс «Проект». Содержит список(или словарь) всех заметок, созданных в приложении.

— Класс «Менеджер проекта». Реализует метод для сохранения объекта «Проект» в файл и метод загрузки проекта из файла. Сохранение и загрузка осуществляются в один и тот же файл «…\My Documents\NoteApp.notes», имя которого задано закрытой константой внутри класса. Формат данных – json, библиотека сериализации (преобразования данных в json-формат) – Newtonsoft JSON.NET.

**2.2 Классы проекта**

**Перечисление «Категория заметок» («NoteCategory»)**

«Категория заметок» содержит значения «Работа», «Дом», «Здоровье и Спорт», «Люди», «Документы», «Финансы», «Разное». Это перечисление является частью класса «Заметка».

**Класс «Заметка» («Note»).**

Класс «Заметка» содержит поля «Название», «Категория заметки», «Текст заметки», «Время создания», «Время последнего изменения». Название, категория и текст заметки доступны для изменений. Время создания инициализируется один раз при создании объекта «Заметка» и больше не модифицируется, доступна для чтения. Время последнего изменения меняется автоматически при изменении названия, категории или текста заметки. Название ограничено 50 символами. Название по умолчанию «Без названия». Допустимы заметки с одинаковыми названиями.

Класс использует в качестве своего поля перечисление «Категория заметок». Связь: композиция. Также класс связан с интерфейсом – IClonable связью реализация.

**Класс «Проект» («Project»).**

Содержит список всех заметок, созданных в приложении. Связь: ассоциация.

**Класс «Менеджер проекта» («ProjectManager»).**

Реализует метод для сохранения объекта «Проект» в файл и метод загрузки проекта из файла. Сохранение и загрузка осуществляются в один и тот же файл «…\My Documents\NoteApp.notes», имя которого задано закрытой константой внутри класса. Связь с классом «Проект» — использование.

UML-диаграмма классов представлена на рисунке 2.1.

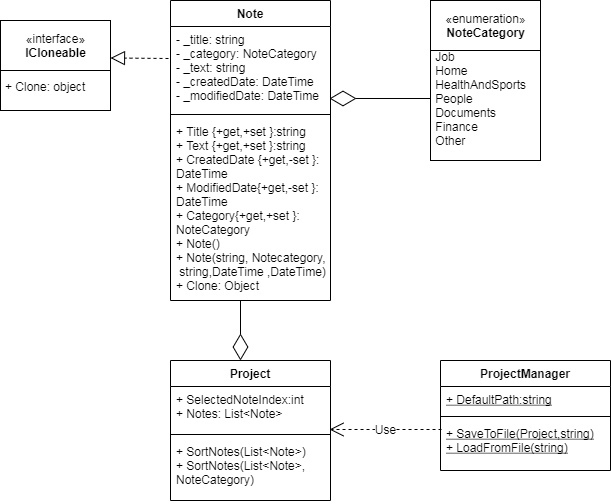


Рисунок 2.1 – UML-диаграмма классов проекта логики

В таблице 2.1 описан класс Project.

Таблица 2.1 – описание класса Project.

|  |  |
| --- | --- |
| **Поля** | **Описание** |
| Notes | Хранит в себе список классов Note. |
| SelectedNoteIndex | Хранит индекс последней выбранной заметки. |
| **Методы** |  |
| SortNotes(List<Note>) | Возвращает список, отсортированный по дате последнего изменения по убыванию. |
| SortNotes (List<Note>, NoteCategory) | Возвращает список, отсортированный по указанной категории. |

Пример создания и инициализации класса Project

Project project = new Project();

Project.Notes.Insert(0, note);

В таблице 2.2 описан класс ProjectManager.

Таблица 2.2 – описание класса ProjectManager.

|  |  |
| --- | --- |
| **Поля** | **Описание** |
| DefaultPath | Хранит в виде строки путь до папки с файлом. По умолчанию сохраняет по пути \Robkanov\NoteApp\NoteApp.notes |
| **Методы** |  |
| SaveFile(Project, string) | Создает или перезаписывает файл, данными которого является переведённый в формат json класс Project, по указанному пути. |
| LoadFromFile(string) | Считывает файл по указанному пути, возвращает список заметок. |

Так как данный класс статический,то невозможно привести пример создания и инициализации класса.

В таблице 1.3 описан класс Note.

Таблица 1.3 – описание класса Note.

|  |  |
| --- | --- |
| **Поля** | **Описание** |
| \_title | Хранит название заметки |
| \_category | Хранит категорию заметки |
| \_text | Хранит текст заметки |
| \_createdDate | Хранит дату создания заметки |
| \_modifiedDate | Хранит время последнего изменения |
| **Свойства** |  |
| Name | Позволяет получить или задать имя заметки |
| Category | Позволяет получить или задать категорию заметки |
| NoteText | Позволяет получить или задать текст заметки |
| CreatedDate | Позволяет получить дату создания |
| ModifiedDate | Позволяет получить или задать время последнего изменения |

Пример создания и инициализации класса Note(отрывок код из Unit-теста).

var expectedTitle = "TestTitle";

var expectedText = "TestText";

var expectedCategory = NoteCategory.Job;

var expectedCreated = DateTime.Parse("2021/01/01"); var expectedModified = DateTime.Parse("2022/02/01"); var note = new Note(expectedTitle, expectedCategory, expectedText,

expectedCreated, expectedModified);

В таблице 1.4 описан интерфейс ICloneable.

Таблица 1.4 – описание интерфейс ICloneable.

|  |  |
| --- | --- |
| **Метод** | **Описание** |
| Clone() | Позволяет создать копию класса. |

Пример использования метода Clone().

// Создаем две заметки.

Note note1 = new Note()

{  
 Name = “Первая заметка”,

Category = NoteCategory.Work,

NoteText = “Текст первой заметки”

};

Note note2 = new Note()

{  
 Name = “Вторая заметка”,

Category = NoteCategory.Documents,

NoteText = “Текст второй заметки”

};

// Создаем третью заметку путем клонирования второй.

Note note3 = (Note)note2.Clone();

По итогу выполнения лабораторной работы история в репозитории на сервисе GitHub стала выглядеть следующим образом (рисунок 2.2).

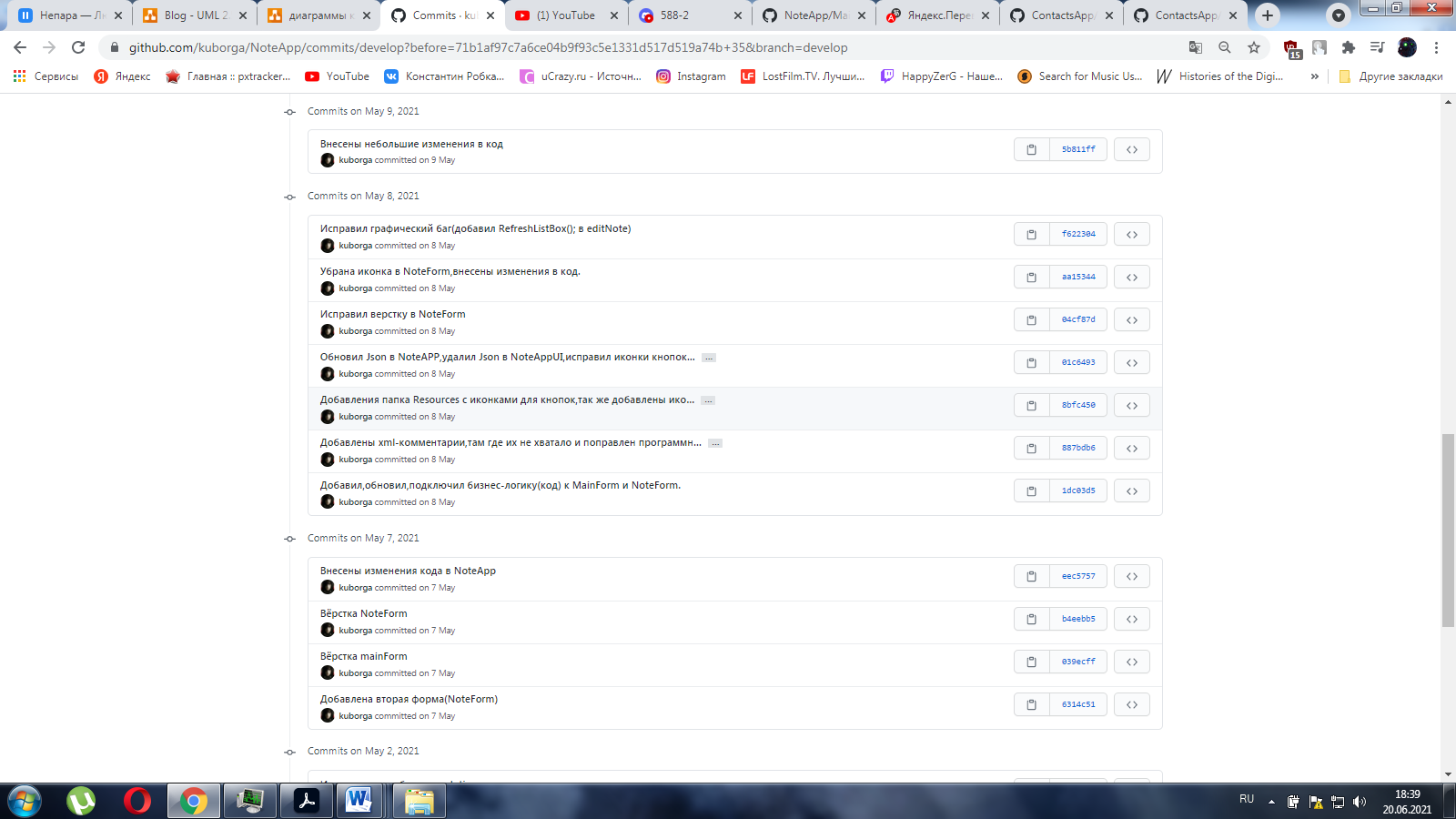


Рисунок 2.2 – История фиксаций изменений в репозитории

**Заключение**

В ходе лабораторной работы изучены типовые требования, предъявляемые к бизнес-логике приложения, получены умения разработки логики приложения с обеспечением данных требований.