Елистратова Анна

Познякевич Александр

Ким Сергей

Фельдман Марк

Группа ББИ 144

Командный проект по программированию

iPod Emulator

Текстовый редактор с удобным пользовательским интерфейсом для iPod Nano 6 поколения. Электронное устройство даёт возможность сохранить текстовую информацию в виде текста песни. Приложение будет связано с MP3 API, который обеспечивает ввод текста непосредственно в песне. Текст песни импортируется в iPod посредством iTunes, что внешне может выглядеть неаккуратно, так как iTunes форматирует его на свое усмотрение.

**Идея проекта:** создать редактор, который позволяет введенному тексту отображаться на экране iPod наиболее стильно и аккуратно.

**Главный адрес репозитории**, в которой хранятся все файлы нашего проекта:

<https://github.com/Anna94i/iPodEmulator.git>

**Распределение по ролям:**

* Ким Сергей 144: Программный разработчик - Developer (application logic);
* Елистратова Анна 144: Архитектор программного обеспечения (Software architect), MP3 API;
* Познякевич Александр 144: Разработчик пользовательского интерфейса (UI developer);
* Mark Feldman 144: Тестирование программного обеспечения (Tester).

**Список классов с кратким описанием каждого из них*.***

MP3Api.cs - статичный класс, содержащий тэги (библиотека TagLib **https://github.com/mono/taglib-sharp**), посредством которого класс MP3Editor может обращаться и вносить изменения в текст MP3 файла. В классе реализовано объявление перечисления с помощью enum, отдельного типа, состоящего из именованных констант, каждая из которых передает свое определенное значение.

MP3Editor.cs - класс типа public - редактор, содержащий методы (функции) для внесения различных изменений в файл. При открывании существующего файла, внесения/не внесения изменений и совершении сохранения, приложение сохраняет все изменения в данный файл, не создавая новый.

MP3File.cs - класс типа public, содержащий подключенную библиотеку TagLib, определение/создание пути сохранения файла и проверка его корректности, привязка к тэгам.

TextEditorForm.cs - класс, содержащий реализацию основных функций элементов формы редактирования файла TextEditorForm.resx : создание нового файла (File->Create), открытие файла (File->Open), сохранение данных в файл(File->Save), обновление экрана - очистка (File ->Close), а также предупреждение о не сохранении файла перед закрытием файла/программы с предложением сохранить. Конструктор класса содержит экземпляр классов MP3Editor.cs и TextEditorConfiguration.cs.

TextEditorConfiguration.cs - класс, который участвует в обеспечении удобства в конфигурации на форме. Например, где, что нужно вписывать. В какой конкретно строке название песни, название автора, текст песни и т.д.

TextEditorForm.Designer.cs - класс, содержащий компоненты дизайна формы редактирования файла.

AboutForm.cs - класс, содержащий текст формы, где кратко написано о самом приложении и возможностях его использования.

Program.cs - класс, осуществляющий запуск формы редактирования файла со всем ее функционалом.

[MP3ApiTests.cs](https://github.com/Anna94i/iPodEmulator/blob/master/iPodEmulatorTests/MP3ApiTests.cs), [MP3EditorTests.cs](https://github.com/Anna94i/iPodEmulator/blob/master/iPodEmulatorTests/MP3EditorTests.cs), [MP3FileTests.cs](https://github.com/Anna94i/iPodEmulator/blob/master/iPodEmulatorTests/MP3FileTests.cs), [TextEditorConfigurationTests.cs](https://github.com/Anna94i/iPodEmulator/blob/master/iPodEmulatorTests/TextEditorConfigurationTests.cs) - классы ветки iPodEmulatorTests, отвечающие за тестирование приложения.

**Интерфейс приложения (скриншоты формы и описания к ним).**



Рис. 1. Основная форма приложения

При нажатии на кнопку “About” появляется форма с описанием возможности использования приложения (рис.2).

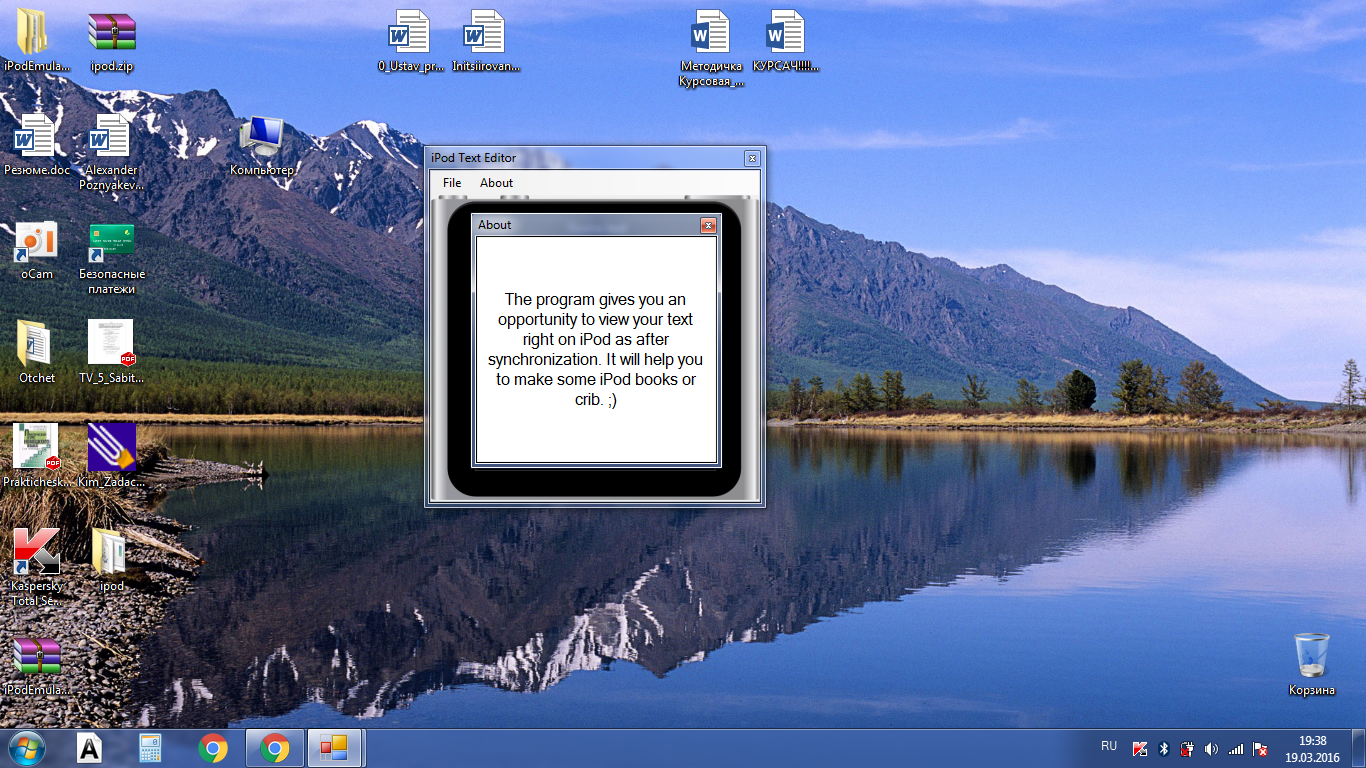


Рис.2. О приложении.

Далее при нажатии на File -> Create (рис.3) открывается форма для создания нового файла (рис.4).

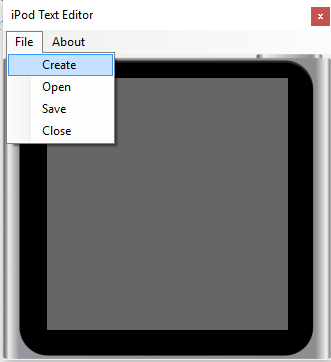
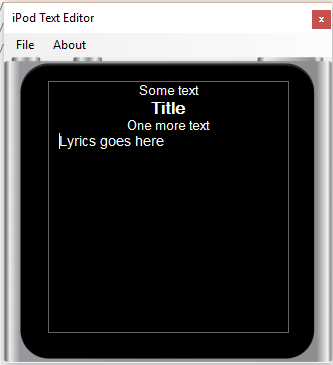
 

Рис.3. Рис 4. Создание файла

Далее можно заполнить файл текстом произвольного типа, вписав в соответствующие поля текст песни, или же любую другую информацию (рис.5).

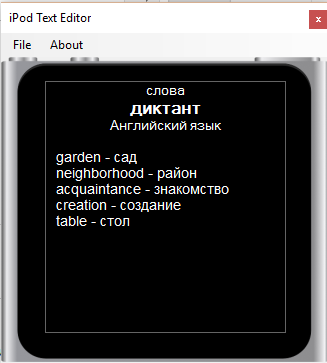


Рис. 5. Заполнение формы.

Далее можно сохранить данные в файл. При нажатии File -> Save (рис. 6) открывается системное окно сохранения данных в файл (рис. 7).

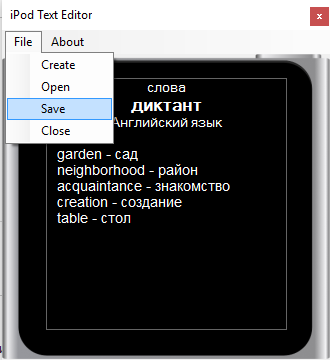


Рис. 6. Сохранение данных в файл.

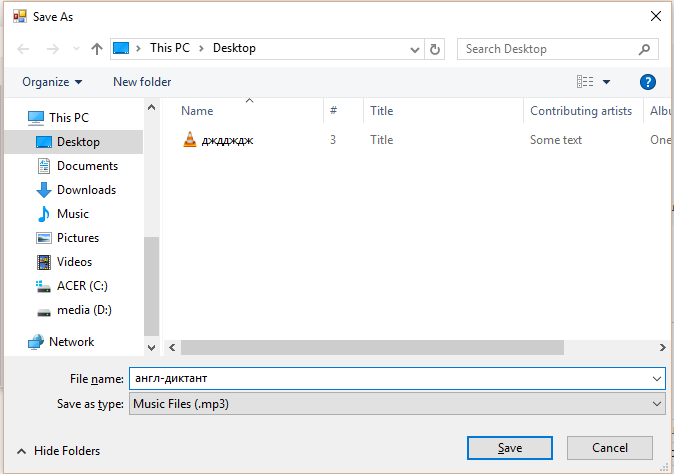


Рис. 7. Системное окно сохранения данных в файл.

Далее при нажатии на File -> Close (рис. 8) или кнопки в правом верхнем углу экрана (инициирование блокировки экрана у интерфейса самих часов), экран очистится от всей информации, вернувшись в стандартное основное состояние (рис. 1).

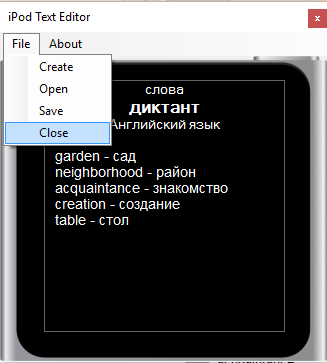


Рис. 8. Возвращение в первоначальное состояние (очистка формы).

Далее можно открыть созданный или другие существующие файлы с помощью комбинации File -> Open (рис. 9). Открывается системное окно (рис. 10), где можно выбрать интересующий файл и открыть его.

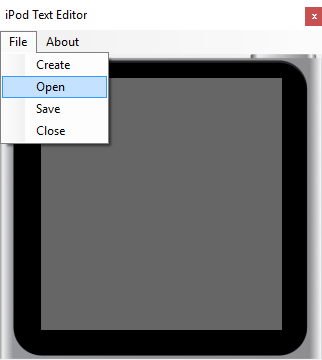


Рис. 9. Открытие файла.

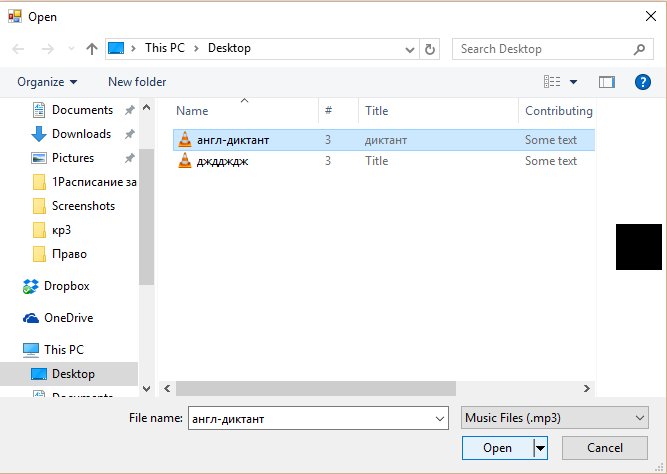


Рис. 10. Системное окно открытия файла.

Далее при открытии файла “англ-диктант”, который мы сохранили ранее, можно внести изменения, например добавив новое слово в список (рис. 11) и т.д. и далее сохранив ее при помощи комбинации File -> Save.



Рис. 11. Редактирование файла.

**Мы также ссылались на год из другого источника в .gitignore файле:**

## Ignore Visual Studio temporary files, build results, and

## files generated by popular Visual Studio add-ons.

<https://github.com/github/gitignore/blob/master/VisualStudio.gitignore#L242>

Этот код обеспечивает защиту от добавления в github лишних файлов из Visual Studio при импорте новых коммитов.