МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования   
 **«Московский авиационный институт**   
 **(национальный исследовательский университет)»**

Институт №3 «Системы управления, информатика и электроэнергетика»

Кафедра 316 «Системное моделирование и автоматизированное проектирование»

Специальность: 09.03.01 «Информатика и вычислительная техника»

**ОТЧЕТ**   
 **к лабораторной работе по дисциплине**   
 **«Интегрированные системы технической подготовки производства»**

|  |  |  |
| --- | --- | --- |
| Отметка: |  |  |
|  | (прописью) |  |
| Проверил: |  | Г. А. Цырков |
|  | (дата, подпись) |  |
| Выполнил:  Студент группы М3О-433Б-18 |  | К. И. Прибытков |

**Оглавление**

[Введение 3](#_Toc103809433)

[Постановка задачи 4](#_Toc103809434)

[Описание основных компонентов приложения 5](#_Toc103809435)

[Класс TableRecord 5](#_Toc103809436)

[Класс TableDataManager 5](#_Toc103809437)

[GTK TreeView 7](#_Toc103809438)

[Использование Cascading Style Sheets (CSS) 7](#_Toc103809439)

[Описание пользовательского интерфейса 9](#_Toc103809440)

[Заключение 11](#_Toc103809441)

[Источники 12](#_Toc103809442)

# Введение

**C#** – современный объектно-ориентированный и типобезопасный язык программирования. C# позволяет разработчикам создавать множество типов безопасных и надежных приложений, работающих в экосистеме .NET. C# относится к широко известному семейству языков C.

В последние годы все более популярной становится бесплатная платформа **.NET**. Она используется многими разработчиками для создания и тестирования приложений, это возможно за счет открытого исходного кода. Одним из вариантов горизонтального увеличения платформы в различные операционные системы стала **.NET Core**. С ее помощью обеспечивается возможность работы с приложениями, которые разрабатывались для Windows, Linux и Mac OSX.

**GTK# (GTKSharp)** — это графический набор средств пользовательского интерфейса, который связывает набор средств GTK+ и различные библиотеки PANDA, что позволяет разрабатывать полностью собственные графические приложения ASP с помощью Mono и .NET.

**CSS** – это формальный язык, служащий для описания оформления внешнего вида документа, созданного с использованием языка разметки (HTML, XHTML, XML). Название происходит от английского Cascading Style Sheets, что означает «каскадные таблицы стилей».

**Visual Studio Code**- это бесплатный редактор исходного кода, разработанный компанией Microsoft для Windows, Linux и macOS. Он считается одним из лучших редакторов, который сочетает в себе IDE и Code Editor.

# Постановка задачи

Целью выполнения данной лабораторной работы является разработка прототипа приложения, позволяющего производить просмотр данных в табличной форме с функцией экспорта в формате XML для ПО Microsoft Project.

При создании приложения был использована платформа для разработки ПО под названием .NET Core. Разработка велась на языке программирования C#, для создания графического интерфейса пользователя использован набор инструментов из пакета GTK#, качестве среды разработки был выбран Visual Studio Code.

# Описание основных компонентов приложения

## Класс TableRecord

Класс описывает представление строки таблицы, в которой содержатся столбцы “Имя”, “Код”, “Количество”, за которые отвечают соответствующие поля класса TableRecord:

public class tablerecordreaderimpl

{

public string Name;

public string Code;

public float Quantity;

public Table Record(string name, string code, float quantity)

{

Name = name;

Code = code;

Quantity = quantity;

}

}

## Класс TableDataManager

Класс отвечает за обработку полей класса TableRecord и соответствующих столбцов, представленных полями класса TableRecord:

public static class TableDataManager

{

public static Gtk.ListStore CreateModel()

{

Gtk.ListStore store = new Gtk.ListStore( typeof(string), typeof(string), typeof(float) );

List<TableRecord> tabRecs = new List<TableRecord>();

tabRecs.Add(new TableRecord("Данные 1", "КОД.123.001", 0.01f));

tabRecs.Add(new TableRecord("Данные 2", "КОД.123.002", 0.02f));

tabRecs.Add(new TableRecord("Данные 3", "КОД.123.003", 0.03f));

tabRecs.Add(new TableRecord("Данные 4", "КОД.123.004", 0.04f));

tabRecs.Add(new TableRecord("Данные 5", "КОД.123.005", 0.05f));

tabRecs.Add(new TableRecord("Данные 6", "КОД.123.006", 0.06f));

tabRecs.Add(new TableRecord("Данные 7", "КОД.123.007", 0.07f));

tabRecs.Add(new TableRecord("Данные 8", "КОД.123.008", 0.08f));

tabRecs.Add(new TableRecord("Данные 9", "КОД.123.009", 0.09f));

tabRecs.Add(new TableRecord("Данные 10","КОД.123.010", 0.00f));

tabRecs.Add(new TableRecord("Данные 11","КОД.123.011", 1.01f));

tabRecs.Add(new TableRecord("Данные 12","КОД.123.012", 2.01f));

tabRecs.Add(new TableRecord("Данные 13","КОД.123.013", 3.01f));

tabRecs.Add(new TableRecord("Данные 14","КОД.123.014", 4.01f));

tabRecs.Add(new TableRecord("Данные 15","КОД.123.015", 5.01f));

tabRecs.Add(new TableRecord("Данные 16","КОД.123.016", 6.01f));

tabRecs.Add(new TableRecord("Данные 17","КОД.123.017", 7.01f));

foreach (TableRecord act in tabRecs) {

store.AppendValues(act.Name, act.Code, act.Quantity);

}

return store;

}

}

## GTK TreeView

GTK TreeView является встроенным виджетом и используется для отображения данных в форме списка, каждая строка в котором может быть разделена на несколько столбцов. Имеет две основные модели: TreeStore и ListStore, которые реализуют интерфейс GTK TreeModel.

В данной реализации TreeView отвечает за расположение строк и столбцов с данными, а также за взаимодействие с пользователем посредством предоставления всех основных функций.

## Использование Cascading Style Sheets (CSS)

В данном приложении реализовано подключение каскадных таблиц стилей по следующему алгоритму:

* Сначала вызывается функция, которая подгружает css-файл по заданному пути;
* Создается объект класса MenuBar;
* В объект класса MenuBar отправляются данные, считанные из таблицы стилей;
* В соответствующих элементах интерфейса поля “Name” указываются такими же, как имя стиля из CSS-файла.

Код, реализующий данный алгоритм:

Load Css("CSS\\style.css");

Jmenubar menuBar = new MenuBar();

menu Bar.Name = neon Widget;

Menu menu = new Menu();

Menu Item fMenuRootItem = new MenuItem();

fMenuRootItem.Label = "Файл";

fMenuRootItem.Submenu = fMenu;

MenuItem fMenuOpenItem = new MenuItem();

fMenuOpenItem.Label = "Открыть";

fMenuOpenItem.Activated += OnClick;

fMenuOpenItem.Name = neonWidget;

fMenu.Append(fMenuOpenItem);

menuBar.Append(fMenuRootItem);

# Описание пользовательского интерфейса

Внешний вид приложения представлен на рисунке 1. Основное окно приложения представляет собой иерархическую структуру изделия. Часть описания изделия содержит в себе детали, входящие в изделие. Описание данных содержится в правой части окна и представлено именем, кодом и количеством деталей.

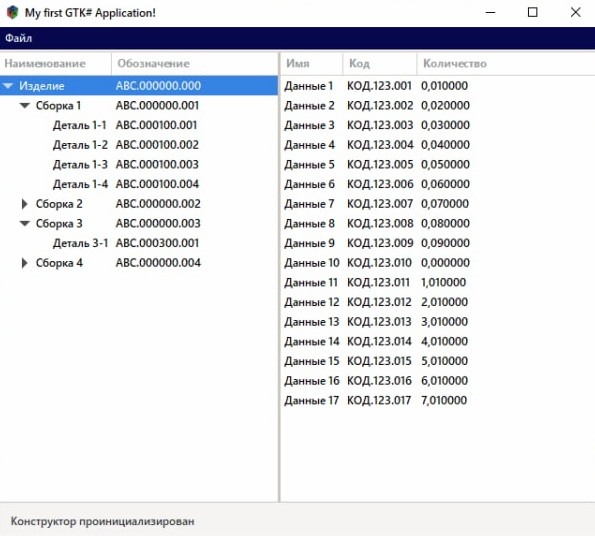


Рисунок 1. – Внешний вид приложения.

При нажатии на вкладку “Файл”, появляется пункт “Открыть” (рисунок 2):

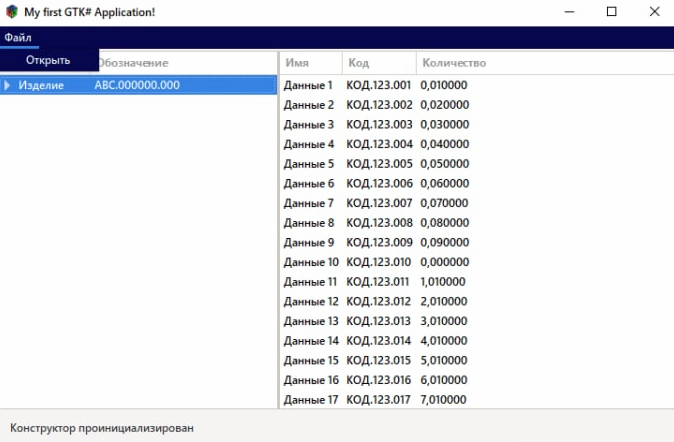


Рисунок 2. – Вкладка “Файл”.

После нажатия на кнопку “Открыть”, появляется диалоговое окно, через которое можно выбрать файл (рисунок 3):

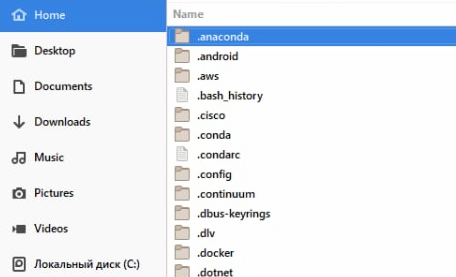


Рисунок 3. – Выбор файла через диалоговое окно.

# Заключение

В результате выполнения данной лабораторной работы было разработано прототип приложения, позволяющего производить просмотр данных в табличной форме с функцией экспорта в формате XML для ПО Microsoft Project. Также был изучен пакет GTK#, платформа .NET Core.

# Источники

1. https://github.com/georgetsyrkov/prjCreator
2. https://github.com/GtkSharp/GtkSharp