Министерство образования Республики Беларусь
Учреждение образования
"БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ"

Факультет информационных технологий и управления
Кафедра интеллектуальных информационных технологий

# ЛАБОРАТОРНАЯ РАБОТА №2

по дисциплине "Проектирование программного обеспечения интеллектуальных систем"

Выполнил студент группы 121701: В. А. Пахомов

Проверил: С. В. Бутрин

Минск 2023

**Цель:** Изучить построение графического пользовательского интерфейса с использование библиотеки Kivy.

**Задание:** Разработать оконное приложение с одним главным окном и несколькими дочерними диалогами. Вызов диалогов осуществляется через соответствующие пункты меню. Команды меню должны дублироваться на панели инструментов.

**Вариант 9:**

Условия поиска и удаления:

- по ФИО и дате рождения (Может быть заполнен только один элемент ФИО, например имя);
- по позиции или составу;
- Футбольной команде или домашнему городу;

**Описание программы:** Программа состоит из трех основных модулей: main_window, read_file, write_file.

Модуль read_file содержит класс MyHandler, который парсит XML файл и складывает собранные значения в список player, каждый player складывается в players, для дальнейшего использования.

```
import xml.sax


class MyHandler(xml.sax.handler.ContentHandler):
    def __init__(self):
        self.charBuffer = []
        self.result = []

    def getCharacterData(self):
        data = ''.join(self.charBuffer).strip()
        self.charBuffer = []
        return data.strip()

    def parse(self, f):
        xml.sax.parse(f, self)
        return self.result

    def characters(self, content):
        self.charBuffer.append(content)

    def startElement(self, name, attrs):
        if name == 'player': self.result.append({})

    def endElement(self, name):
        if not name == 'player': self.result[-1][name] = self.getCharacterData()


def read_xml(path):
    info = MyHandler().parse(path)
    players = []
    for i in range(0, len(info)):
        player = []
        player.append(info[i]['name'])
        player.append(info[i]['date'])
        player.append(info[i]['team'])
        player.append(info[i]['town'])
        player.append(info[i]['line_up'])
        player.append(info[i]['position'])
        players.append(player)
    return players
```

Модуль write_file содержит функцию write(), которая создает XML файл на основе списка players.

```python
import xml.etree.ElementTree as ET

def write(lst: list, path):
    data = ET.Element('data')
    for elem in lst:
        player = ET.SubElement(data, 'player')
        name = ET.SubElement(player, 'name')
        name.text = elem[0]
        date = ET.SubElement(player, 'date')
        date.text = str(elem[1])
        team = ET.SubElement(player, 'team')
        team.text = str(elem[2])
        town = ET.SubElement(player, 'town')
        town.text = str(elem[3])
        line_up = ET.SubElement(player, 'line_up')
        line_up.text = str(elem[4])
        position = ET.SubElement(player, 'position')
        position.text = elem[5]
    ET.ElementTree(data).write(path)
```

Итоговая структура XML-файла выглядит так:

```xml
<data>
    <player>
        <name>&#1051;&#1080;&#1086;&#1085;&#1077;&#1083;&#1100; &#1052;&#1077;&#1089;&#1089;&#1080;</name>
        <date>24.06.1987</date>
        <team>&#1055;&#1057;&#1046;</team>
        <town>&#1056;&#1086;&#1089;&#1072;&#1088;&#1080;&#1086;</town>
        <line_up>&#1054;&#1089;&#1085;&#1086;&#1074;&#1085;&#1086;&#1081;</line_up>
        <position>&#1053;&#1072;&#1087;&#1072;&#1076;&#1072;&#1102;&#1097;&#1080;&#1081;</position>
    </player>
<player>
    <name>&#1050;&#1080;&#1083;&#1080;&#1072;&#1085; &#1052;&#1073;&#1072;&#1087;&#1087;&#1077;</name>
    <date>20.12.1998</date>
    <team>&#1055;&#1057;&#1046;</team>
    <town>&#1055;&#1072;&#1088;&#1080;&#1078;</town>
    <line_up>&#1054;&#1089;&#1085;&#1086;&#1074;&#1085;&#1086;&#1081;</line_up>
    <position>&#1053;&#1072;&#1087;&#1072;&#1076;&#1072;&#1102;&#1097;&#1080;&#1081;</position>
</player>
```

Модуль read_file содержит класс MyApp, в котором описаны все события, расположение кнопок, расположение различных виджетов, описаны поиск, добавление и удаление элементов.

```
class MyApp(MDApp):
    title = "Football Players"
    def build(self):
        screen = Screen()

        main_gl = MDGridLayout(rows=10, cols = 1)
        bl_labels = MDBoxLayout(orientation='horizontal', size_hint=[1, 0.2], padding = dp(20), spacing=dp(15))

        search_button = Button(text="Поиск", on_press=self.btn_press_call_search)
        delete_button = Button(text="Удаление", on_press=self.btn_press_call_delete)
        add_button = Button(text="Добавление", on_press=self.btn_press_call_add)

        bl_labels.add_widget(search_button)
        bl_labels.add_widget(delete_button)
        bl_labels.add_widget(add_button)

        self.table = MDDataTable(size_hint=[1, 1], check=False,
                                 use_pagination=True,
                                 column_data=[("ФИО", dp(70)),
                                              ("Дата", dp(70)),
                                              ("Команда", dp(70)),
                                              ("Город", dp(70)),
                                              ("Состав", dp(70)),
                                              ("Позиция", dp(70))],
                                 row_data=players
                                 )

        bl_delete = MDBoxLayout(orientation='vertical')

        self.what_delete = Spinner(
            text='Выберите условие удаления',
            values=('ФИО', 'Дата', 'Команда', 'Город', 'Состав'),
            size_hint=[1, .8]
        )

        self.what_search = Spinner(
            text='Выберите условие поиска',
            values=('Дата', 'Команда', 'Город', 'Состав'),
            size_hint=[1, .8]
        )

        self.label_delete = Label()
        self.text_input_delete = TextInput(multiline=False)

        bl_delete.add_widget(self.what_delete)
        bl_delete.add_widget(self.label_delete)
        bl_delete.add_widget(self.text_input_delete)
        bl_delete.add_widget(Button(text='Удалить', on_press=self.btn_press_delete))

        self.delete_popup = Popup(content=bl_delete, title='Окно удаления', size_hint=[.35, .5])
```

```
self.search_popup = Popup(content=bl_search, title='Окно поиска', size_hint=[0.8, 0.8])

self.table_search = MDDataTable(size_hint=[1, 1], check=False,
                                use_pagination=True,
                                column_data=[("ФИО", dp(30)),
                                             ("Дата", dp(10)),
                                             ("Команда", dp(15)),
                                             ("Город", dp(25)),
                                             ("Состав", dp(25)),
                                             ("Позиция", dp(25))],
                                )
self.label_search = Label()
self.text_input_search = TextInput(multiline=False, size_hint=[1, .7])

bl_search_menu = MDBoxLayout(orientation='vertical', size_hint=[1, 1], spacing=dp(10), padding=dp(10))

bl_search_menu.add_widget(self.what_search)
bl_search_menu.add_widget(self.label_search)
bl_search_menu.add_widget(self.text_input_search)
bl_search_menu.add_widget(Button(text='Поиск', on_press=self.btn_press_search, size_hint=[1, .7]))

bl_search.add_widget(self.table_search)
bl_search.add_widget(bl_search_menu)

bl_add = MDBoxLayout(orientation='vertical')

self.text_input_1 = TextInput(hint_text='ФИО', multiline=False)
bl_add.add_widget(self.text_input_1)
self.text_input_2 = TextInput(hint_text='Дата', multiline=False)
bl_add.add_widget(self.text_input_2)
self.text_input_3 = TextInput(hint_text='Команда', multiline=False)
bl_add.add_widget(self.text_input_3)
self.text_input_4 = TextInput(hint_text='Город', multiline=False)
bl_add.add_widget(self.text_input_4)
self.text_input_5 = TextInput(hint_text='Состав', multiline=False)
bl_add.add_widget(self.text_input_5)
self.text_input_6 = TextInput(hint_text='Позиция', multiline=False)
bl_add.add_widget(self.text_input_6)
bl_add.add_widget(Button(text='Добавить', on_press=self.btn_press_add))


self.add_popup = Popup(title='Окно добавления', size_hint=[.8, .6], content=bl_add)

self.theme_cls.theme_style = "Light"
self.theme_cls.primary_palette = "Gray"

main_gl.add_widget(self.table)
main_gl.add_widget(bl_labels)
screen.add_widget(main_gl)
return screen
```

```
def btn_press_save(self, instance):
    write_file.write(players, self.path.text)

def btn_press_call_delete(self, instance):
    self.delete_popup.open()

def btn_press_call_search(self, instance):
    self.search_popup.open()

def btn_press_call_add(self, instance):
    self.add_popup.open()

def btn_press_delete(self, instance):
    index = -1
    buffer = read_file.read_xml('players.xml')
    counter = 0
    if self.what_delete.text == 'ФИО':
        index = 0
    elif self.what_delete.text == 'Дата':
        index = 1
    elif self.what_delete.text == 'Команда':
        index = 2
    elif self.what_delete.text == 'Город':
        index = 3
    elif self.what_delete.text == 'Состав':
        index = 4

    if index == -1:
        self.label_delete.text = 'Выберите условие'
        counter = -1
    if index != -1:
        counter = 0
        for player in buffer:
            if player[index] == self.text_input_delete.text:
                players.remove(player)
                counter += 1
    write_file.write(players, 'players.xml')
    if counter == 0:
        self.label_delete.text = 'Совпадений не найдено'
    elif counter != 0 and counter != -1:
        self.label_delete.text = 'Было удалено ' + str(counter) + ' записей'
    self.table.update_row_data(self.table.row_data, players)
```