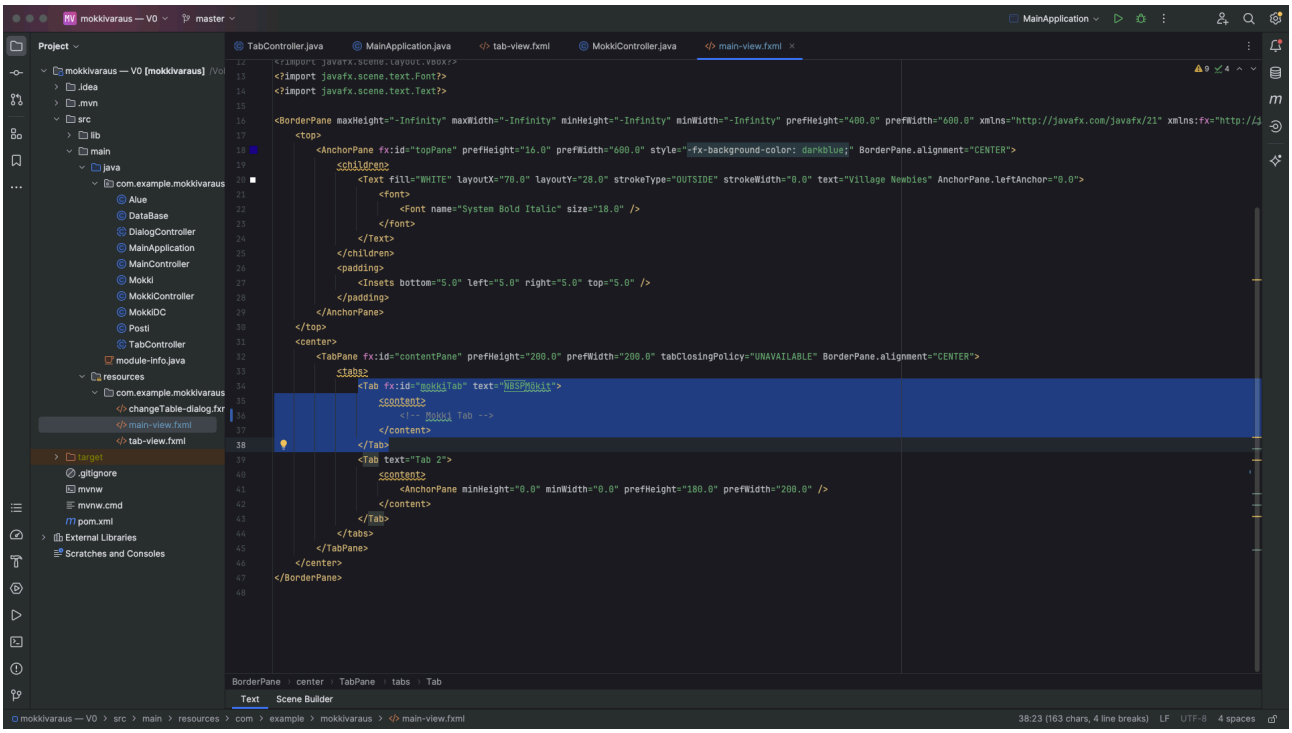
Инструкция для добавления индивидуальной вкладки со своим fxml и Controller

---

1. Создать новую вкладку Tab в main-view.fxml с fx:id и комментарием



---

2. Добавить свои файлы

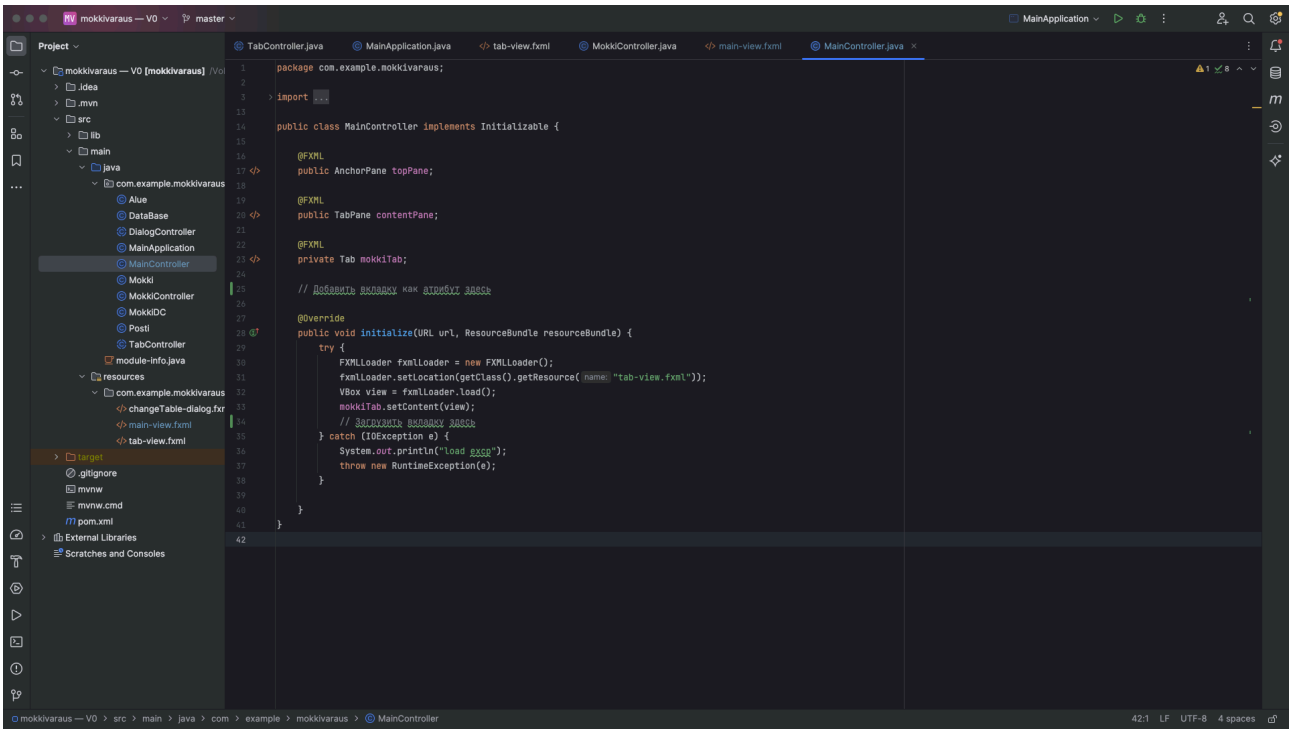
fxml - src/main/resources/com/example/mokkivaraus
java - src/main/java/com/example/mokkivaraus

---

3. Загрузить вкладку в MainController

- добавить вашу вкладку Tab в список атрибутов

@FXML
private Tab <fx:id вашего Tab>



- Загрузить вкладку в методе initialize в блоке try-catch

```
FXMLLoader fxmlLoader = new FXMLLoader();
fxmlLoader.setLocation(getClass().getResource(«имя вашего fxml файла»));
<главный view в fxml> view = fxmlLoader.load();
* Напиример в tab-view.fxml - VBox
<ваш Tab>.setContent(view);
```

---

4. Работа с базой данных, класс DataBase

Для простой работы с базой данный в классе DataBase созданы два метода

- **getData(sql)** : параметр SQL запрос в виде строки. Возвращает ResultSet объект, хранящий в себе результат запроса.
- **setData(sql)** : параметр SQL запрос в виде строки. Выполняет запрос, подходит для запросов. Не возвращающих значение, как INSERT, UPDATE, DELETE

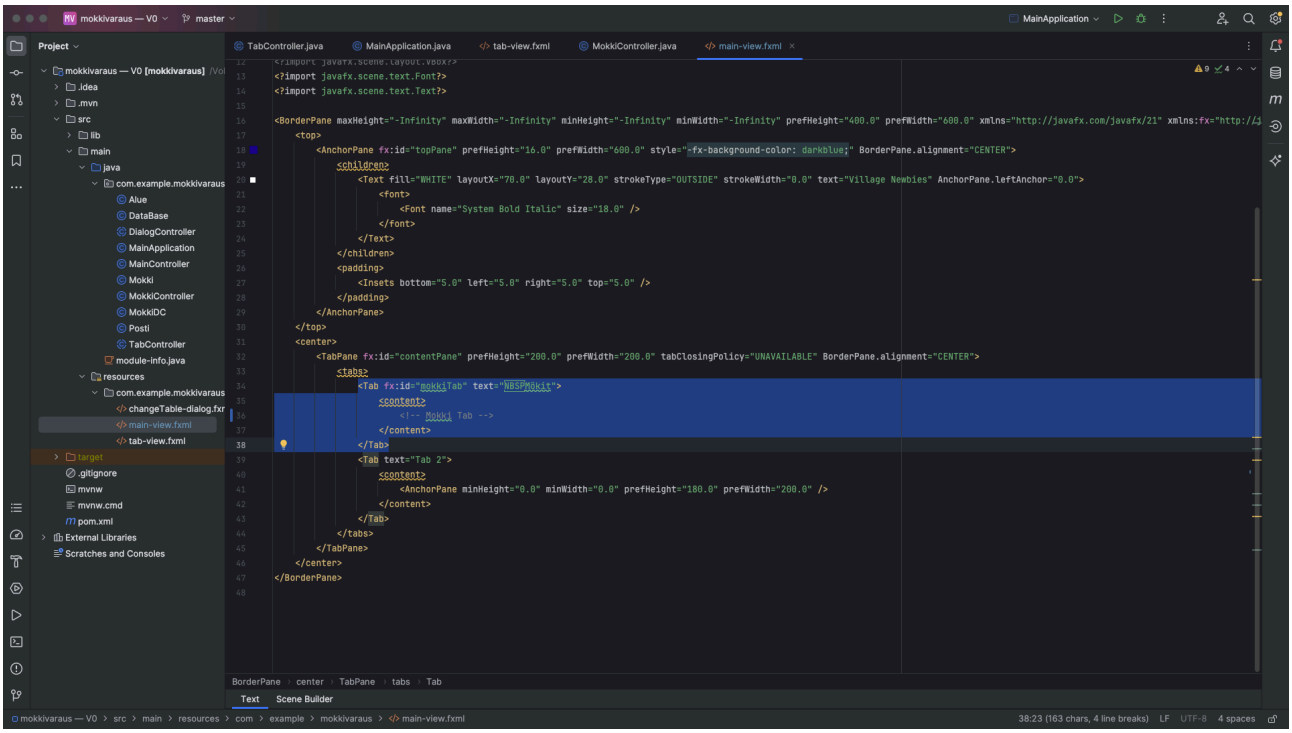
Остальные методы описаны с помощью комментариев в коде

---

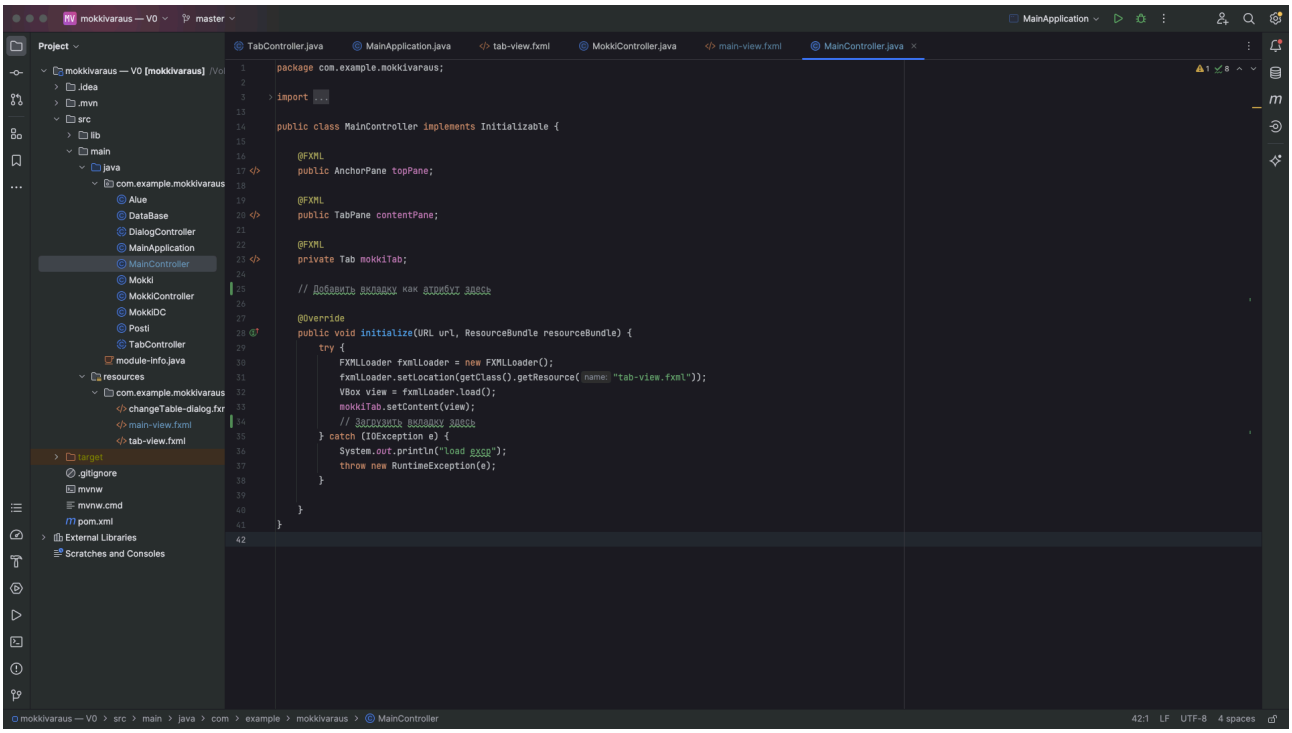
5. Диалоговое окно (changeTable-dialog.fxml и DialogController) при необходимости можно использовать отдельно, смотрите ниже

Инструкция для добавления вкладки похожей на Mökki

1. Создать новую вкладку Tab в main-view.fxml с fx:id и комментарием



2. Загрузить вкладку в MainController



- добавить вашу вкладку Tab в список атрибутов

```
@FXML
private Tab <fx:id вашего Tab>
```

- Загрузить вкладку в методе initialize в блоке try-catch

```
FXMLLoader fxmlLoader = new FXMLLoader();
fxmlLoader.setLocation(getClass().getResource(«tab-view.fxml»));
VBox view = fxmlLoader.load();
<ваш Tab>.setContent(view);
```

3. Создать необходимые классы

1. Класс Таблицы (пример Mokki)

Содержит в себе атрибуты с именами идентичными колонкам таблицы.
Содержит два обязательных конструктора

1. Пустой.
2. Принимает в качестве параметра ResultSet объект - строка результата, полученного с SQL запроса. На основании данных строки присваивает значения атрибутам.

Содержит методы get и set для всех атрибутов
Содержит переписанный метод toString() - не обязательно

2. Класс контроллера для tab-view.fxml, наследуется от класса TabController<T> (T - класс таблицы из первого пункта) (пример MokkiController)

Содержит пустой конструктор, в котором вызывается конструктор родительского класса. В конструктор (родителя) передается:

- название таблицы из базы данных (в моем случае «mokki»)
- название ключа таблицы (в моем случае «mokki_id»))
- Класс таблицы (в моем случае Mokki.class)

Содержит методы
getColToAttr() - возвращает ArrayList<String[]>, содержащий два строковых массива. Первый - названия столбов в TableView. Второй - название соответствующих атрибутов Класса Таблицы (пункт 1)
getController() - возвращает дочерний класс от DialogController (описан ниже в пункте 3). В качестве параметров контроллеру всегда передаются атрибуты: **tableName** и **identifierKey**
getSearchConditions(T object, String newValue) - возвращает true или false, в зависимости от выполнения условий поиска. В качестве параметров принимает объект Класса Таблицы и строку - то что сейчас введено в строку поиска.
getSearchFilters() - возвращает строковой массив, содержащий список фильтров для поискового бара.

Все выше перечисленные методы, являются абстрактными в классе TabController, поэтому IDE должна автоматически предложить их переписать. При необходимости вы можете переписывать методы класса TabController, используя @Override в своем классе. Сам класс TabController менять нельзя!

3. **Класс контроллер диалогового окна для changeTable-dialog.fxml, наследуется от класса DialogController (пример MokkiDC)**

Содержит атрибуты

- Инициализированный объект Класса Таблицы (пункт 1), для инициализации используется пустой конструктор
- Атрибуты - поля данных. Данные, требующиеся от пользователя.

Содержит конструктор, повторяющий родительский
Содержит методы
setObject(Object object) - принимает в качестве параметра объект Класса Таблицы (пункт 1), устанавливает значения атрибута, в котором содержится объект Класса Таблицы, равным полученному параметру. Устанавливает значение параметров: **editMode** - true, **identifierValue** - значение ключа/идентификатора полученного объекта.
setDialogContent() - заполнение центральной сетки GridPane элементами. Доступ к сетке можно получить через атрибут **formsGridPane**.
setEditContent() - устанавливает значения из полей объекта Класса Таблицы, установленного в качестве атрибута данного класса, в атрибуты полей данных.
listOfAttributes() - возвращает HashMap<String, Object>, где ключ - название столбца в таблице в базе данных, значение - соответствующее значение атрибута объекта Класса Таблицы (пункт 1). Используется для SQL запросов
checkData() - возвращает true - если введенные пользователем данные прошли проверку, и их можно добавлять в таблицу, и false - если нет. Внутри себя изменяет значения атрибутов: **alertTitle** и **alertMessage,** используются для выведения предупреждения об ошибке в заполнении данных.

Для изменений размеров окна используйте атрибут dialogPane.

Все выше перечисленные методы, являются абстрактными в классе DialogController, поэтому IDE должна автоматически предложить их переписать. При необходимости вы можете переписывать методы класса DialogController, используя @Override в своем классе. Сам класс DialogController менять нельзя!

3. Если все необходимые классы и методы написаны, то вкладка должна работать