

### **3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ**

Функциональная структура проекта во многом аналогична структурной. Взаимосвязь между основными компонентами представлена на диаграмме классов ГУИР.400201.304 РР.1.

#### **3.1. Блок игровой логики**

В данном блоке находятся описания базовых логических законов и правил игры. Определение этих базовых элементов осуществляется в элементе `GameBase_BP`, который является наследуемым классом от базового класса `GameBase`. Ниже будут рассмотрены отдельные элементы и переменные определённые в файле.

#### **3.2. Блок управления**

Блок управления представлен в виде класса `PlayerController_BP`. Данный класс наследуется от базового класса `PlayerController`, и отвечает за управление объектом `playerPawn`, посредством использования следующих системных асинхронных вызовов: `InputAxis MoveForwardBack`, `InputAxis MoveRightLeft`.

#### **3.3. Блок игрового интерфейса**

Данный блок реализован с использованием базовых инструментов Unreal Engine 4, для создания графических интерфейсов. Логика, описывающая функционал интерфейса описана в файлах: `MenuUI_BP`, `PlayerUI_BP`, `OptionsUI_BP`.

#### **3.4. Алгоритм чтения аудиофайла**

Данный алгоритм является частью подключаемого плагина для работы и анализа аудио файла. К данному алгоритму относятся следующие заголовочные файлы: `WAV_Header.h`, `BPM_Detector.h`. Функционал данных заголовочных файлов описан в исходных файлах `WAV_Header.cpp` и `BPM_Detector.cpp` соответственно.

В заголовочном файле `WAV_Header.h` задана структура `WAVHEADER`, предназначенная для хранения полей заголовка аудио файла. В исходном файле `WAV_Header.cpp` описаны методы `GetHeader()`, отвечающий за чтение заголовка файла, а так же `PrintHeader()`, предназначенный для вывода значений заголовка аудио файла в формате `string`.

В заголовочном файле `BPM_Detector.h` инициализированы методы `ValidateFile()` и `ReadFromFile()`. В исходном файле `BPM_Detector.cpp` описаны реализации данных методов, для валидации входящего файла и чтения данных из входящего файла.

### 3.5. Алгоритм анализа аудиофайла

Данный алгоритм также является частью подключаемого плагина для работы и анализа аудио файла. К данному алгоритму относится заголовочный файл `FFT_Functions.h`. Функционал данного заголовочного файла описан в исходном файле `FFT_Functions.cpp` соответственно.

В заголовочном файле `FFT_Functions.h` инициализированы следующие методы:

- `ConvertToComplex()`;
- `FFT()`;
- `CalculateBassIntensity()`;
- `CalculateBPM()`.

В исходном файле `BPM_Detector.cpp` описаны реализации вышеописанных методов.

Метод `ConvertToComplex()` реализует преобразование массива значений типа `int16_t`, в массив комплексных чисел типа `double`. Данный метод необходим для последующей реализации метода быстрого преобразования Фурье.

Метод `FFT()` производит рекурсивную операцию преобразования Фурье.

Метод `CalculateBassIntensity()` осуществляет анализ массива, полученного из метода `FFT()`, и определяет существенные изменения нижних частотных диапазонов в конкретный интервал времени.

Метод `CalculateBPM()` возвращает значение ударов в минуту, вычисленных из корреляции последних нескольких результатов метода `CalculateBassIntensity()`.