**1 ОБЗОР ЛИТЕРАТУРЫ**

Перед началом разработки дипломного проекта, нужно определиться с основными понятиями в теории звука.

Звуковая волна – [физическое явление](https://ru.wikipedia.org/wiki/%D0%A4%D0%B8%D0%B7%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5_%D1%8F%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5), представляющее собой распространение в виде [упругих волн](https://ru.wikipedia.org/wiki/%D0%A3%D0%BF%D1%80%D1%83%D0%B3%D0%B8%D0%B5_%D0%B2%D0%BE%D0%BB%D0%BD%D1%8B) механических колебаний в твёрдой, жидкой или газообразной среде. В узком смысле под звуком имеют в виду эти колебания, рассматриваемые в связи с тем, как они воспринимаются органами животных и человека. Природа звуковых волн колебательная, вызываемая и производимая вибрацией любых тел. Возникновение и распространение классической звуковой волны возможно в трёх упругих средах: газообразных, жидких и твёрдых.

Так-как звуковая волна имеет колебательную природу, то она характеризуется такими величинами как амплитуда, частота и фаза. Частота измеряется в герцах и обозначает количество колебаний за период времени, равный одной секунде. От частоты звука зависит и субъективное понятие его высоты. Чем больше звуковых колебаний совершается за секунду, тем «выше» кажется звучание. У звуковой волны так же имеется ещё одна важнейшая характеристика, имеющая название - длина волны. Длина волны – это расстояние между двумя ее точками, движущимися одинаково (в одной фазе). Поскольку колеблющиеся точки оказываются в одинаковом положении (в одной фазе) через время, равное периоду колебаний, то длина волны, это расстояние, на которое она распространяется за время, равное одному периоду.

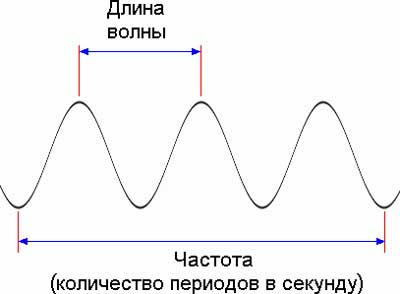


Рисунок 1.1 – Звуковая волна

На рисунке 1.1 проиллюстрированы длина и частота волны.

В музыкальной терминологии звука существуют такие важные обозначения, как октава, тон и обертон звука.

Октава – [музыкальный интервал](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%B2%D0%B0%D0%BB_(%D0%BC%D1%83%D0%B7%D1%8B%D0%BA%D0%B0)), в котором соотношение [частот](https://ru.wikipedia.org/wiki/%D0%A7%D0%B0%D1%81%D1%82%D0%BE%D1%82%D0%B0) между звуками составляет 1 к 2 (то есть частота высокого звука в 2 раза больше низкого). Субъективно на слух октава воспринимается как устойчивый, базисный музыкальный интервал. Два звука, отстоящие на октаву, воспринимаются очень похожими друг на друга, хотя явно различаются по высоте.

Октава существует в трёх разновидностях:

1. Чистая октава — интервал в восемь ступеней и шесть тонов, простейшее соотношение двух звуков различной высоты, у которых частота колебаний относится как два к одному, то есть верхний звук имеет вдвое большую частоту колебаний, чем нижний звук.
2. Уменьшённая октава — интервал в восемь ступеней и пять с половиной тонов.
3. Увеличенная октава — интервал в восемь ступеней и шесть с половиной тонов.

Октава в свою очередь состоит из тонов и обертонов. Переменные колебания в гармонической звуковой волне одной частоты воспринимаются человеческим ухом как музыкальныйтон. Колебания высокой частоты можно интерпретировать как звуки высокого тона, колебания низкой частоты – как звуки низкого тона. Человеческое ухо способно чётко отличать звуки с разницей в один тон (в диапазоне до 4000 Гц). Несмотря на это, в музыке используется крайне малое число тонов. Объясняется это из соображений принципа гармонической созвучности, всё основано на принципе октав.

Рассмотрим теорию музыкальных тонов на примере струны, натянутой определённым образом. Такая струна, в зависимости от силы натяжения, будет иметь «настройку» на какую-то одну конкретную частоту. При воздействии на эту струну чем-либо с одной определённой силой, что вызовет её колебания, стабильно будет наблюдаться какой-то один определенный тон звука, мы услышим искомую частоту настройки. Этот звук называется основным тоном. Однако, большинство музыкальных инструментов никогда не воспроизводят одни чистые основные тона, их неизбежно сопровождают призвуки, именуемые обертонами. Обертоны бывают гармоническими и негармоническими. Частоты гармонических обертонов кратны частоте основного тона (гармонические обертоны вместе с основным тоном также называются гармониками). В музыке практически исключается оперирование некратными обертонами, поэтому термин сводится к понятию "обертон", подразумевая под собой гармонический. У некоторых инструментов, например, фортепиано, основной тон даже не успевает сформироваться, за короткий промежуток происходит нарастание звуковой энергии обертонов, а затем так же стремительно происходит спад. Многие инструменты создают так называемый эффект "переходного тона", когда энергия определённых обертонов максимальна в определённый момент времени, обычно в самом начале, но потом резко меняется и переходит к другим обертонам. Частотный диапазон каждого инструмента можно рассмотреть отдельно, и он обычно ограничивается частотами основных тонов, который способен воспроизводить данный конкретный инструмент. В теории звука также присутствует такое понятие как шум. Шум - это нерегулярные колебания без закономерной зависимости.

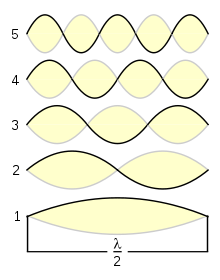


Рисунок 1.2 – Колебания идеальной струны

На рисунке 1.2 изображены составляющие колебания идеальной струны: 1 – основной тон, 2-5 – гармоники, соответствующие первому — четвёртому обертонам.

Тут уместно вспомнить важное определение музыкальной акустики, понятие тембра звука. Тембр - это особенность музыкальных звуков, которые придают музыкальным инструментам и голосам их неповторимую узнаваемую специфику звучания, даже если сравнивать звуки одинаковой высоты и громкости. Тембр каждого музыкального инструмента зависит от распределения звуковой энергии по обертонам в момент появления звука.

Поскольку на практике практически не встречаются волны одной частоты, то возникает необходимость разложения всего звукового спектра слышимого диапазона на обертоны или гармоники, такой метод называется Фурье-анализом. Для этих целей существуют графики, которые отображают зависимость относительной энергии звуковых колебаний от частоты. Такой график называется графиком частотного спектра звука (см. рисунок 1.3). Частотный спектр звука бывает двух типов: дискретный и непрерывный. Дискретный график спектра отображает частоты по отдельности, разделённые пустыми промежутками. В непрерывном спектре присутствуют сразу все звуковые частоты.

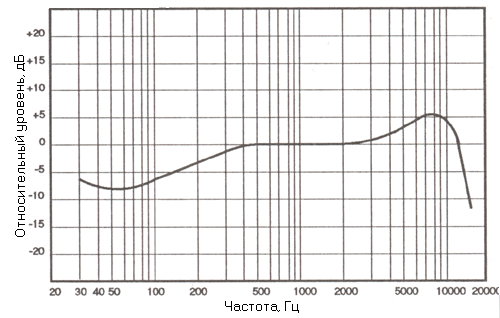


Рисунок 1.3 – Пример графика частотного спектра звука

В случае с музыкой или акустикой чаще всего используется обычный график Амплитудно-Частотой характеристики (см. рисунок 1.4). На таком графике представлена зависимость амплитуды звуковых колебаний от частоты на протяжении всего спектра частот (20 Гц - 20 кГц). Глядя на такой график легко понять, например, сильные или слабые стороны конкретного динамика или акустической системы в целом, наиболее сильные участки энергетической отдачи, частотные спады и подъёмы, затухания, а также проследить крутизну спада.

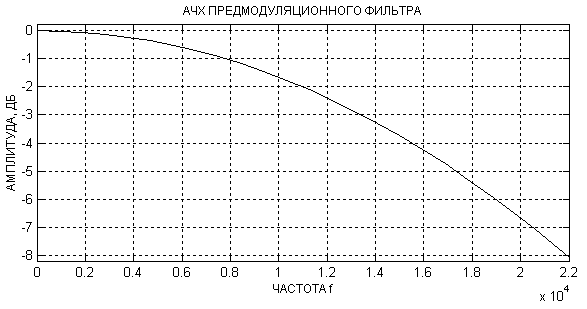


Рисунок 1.4 – Пример графика АЧХ

Теперь, когда мы разобрались с основными понятиями теории звука, необходимо понять, как устройства ввода – вывода могут воспроизводить мелодии. Рассмотрим данный вопрос на примере дисководов гибких дисков.

Дисководы гибких дисков — уже устаревшее устройство: сегодня практически никто не использует дискеты как носитель информации, поскольку существуют куда более удобные и быстрые способы распространения информации. Работающие экземпляры дисковода гибких дисков могли бы для чего-нибудь сгодиться, например, для извлечения из них музыки. Дисководы гибких дисков достаточно просты: они получают команду, когда нужно вращать диск, когда нужно читать или писать и как далеко сдвинуть магнитную головку. Используя какое-либо устройство управления и записанный музыкальный алгоритм возможно получить столь голосную полифонию, сколько используется дисководов.

Мелодии извлекаются из флоппи-дисководов через управление скоростью читающей головки, как правило, используется MIDI-запись нужной мелодии. Для управления головкой используется любой микроконтроллер, который может посылать управляющие сигналы на дисковод в нужной последовательности, например, Arduino. Управление может осуществляться, как и «hardware», когда код мелодии загружается непосредственно в контроллер, так и «software» — когда мелодия обрабатывается специальной программой на компьютере, а контроллер выступает в роли драйвера, распределяя и передавая полученный сигнал на дисководы.

MIDI – [стандарт](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82) [цифровой звукозаписи](https://ru.wikipedia.org/wiki/%D0%A6%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%BE%D0%B9_%D0%B7%D0%B2%D1%83%D0%BA) на формат обмена данными между [электронными музыкальными инструментами](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D1%8B%D0%B5_%D0%BC%D1%83%D0%B7%D1%8B%D0%BA%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5_%D0%B8%D0%BD%D1%81%D1%82%D1%80%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D1%8B). Интерфейс позволяет единообразно кодировать в цифровой форме такие данные как нажатие клавиш, настройку [громкости](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%BE%D0%BC%D0%BA%D0%BE%D1%81%D1%82%D1%8C_%D0%B7%D0%B2%D1%83%D0%BA%D0%B0) и других акустических параметров, выбор [тембра](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BC%D0%B1%D1%80), [темпа](https://ru.wikipedia.org/wiki/%D0%9C%D1%83%D0%B7%D1%8B%D0%BA%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D1%82%D0%B5%D0%BC%D0%BF), [тональности](https://ru.wikipedia.org/wiki/%D0%A2%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D1%8C) и др., с точной привязкой во времени. В системе кодировок присутствует множество свободных команд, которые производители, программисты и пользователи могут использовать по своему усмотрению. Стандартный midi-файл – это специально разработанный формат файлов, предназначенный для хранения данных, записываемых и/или исполняемых секвенсором, [секвенсор](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D0%BA%D0%B2%D0%B5%D0%BD%D1%81%D0%BE%D1%80) может быть, как программой для компьютера, так и аппаратным модулем. Через MIDI интерфейс передается информация о действиях, выполняемых на музыкальном устройстве - к примеру, нажатие клавиши. Это содержит информацию о двух параметрах: номере нажатой клавиши и силе удара по ней. Большинство параметров MIDI могут принимать значения от 0 до 127 (1 байт). Поэтому размер полной MIDI-клавиатуры составляет 128 клавиш. Так как музыкантам привычнее оперировать нотами и номерами октав, в MIDI описано соответствие между номером клавиши и его интуитивным значением. Однако номера октав здесь отличаются от принятых на традиционных акустических инструментах.

Пусть MIDI-карта управляет несколькими устройствами. В этом случае управляемые устройства подключаются цепочкой через соединение MIDI THRU – MIDI IN. Поскольку на выход MIDI THRU передается вся информация, пришедшая на MIDI IN, все подключенные устройства получают одну и ту же MIDI-информацию. Однако, управление несколькими устройствами имеет смысл в том случае, если каждое из них будет исполнять свою партию. Таким образом, существует необходимость разделить поток MIDI-сообщений, чтобы каждое устройство могло принимать только свои сообщения. Эта проблема в MIDI решена с помощью каналов. Предполагается, что существует 16 MIDI-каналов, и каждое сообщение может идти только по одному из них. При этом каждое устройство настраивается на прием сообщений, приходящих только по одному из каналов. Поток MIDI-информации передается побайтно. Для контроля за состоянием линии в начале каждого байта передается стартовый бит (1), а в конце — столовый (0). Каждый байт состоит из 8 значащих битов. Любой байт, передаваемый по MIDI, является байтом либо статуса, либо значения. Статусный байт всегда первый в MIDI-сообщении, он определяет его тип и номер MIDI-канала. В каждом MIDI-сообщении содержится только один статусный байт. Байты значения содержат параметры, необходимые для данного типа MIDI-сообщения. MIDI-секвенцией называют записанную последовательность MIDI-сообщений и временных промежутков между ними. Такая последовательность при воспроизведении повторяет оригинал, как обычная аудиозапись. MIDI-секвенция, записанная в файл, занимает очень немного места по сравнению с аудиозаписью. При разработке формата предусмотрели возможность для любого секвенсора читать и записывать файл таким образом, чтобы, с одной стороны, не потерялись его данные, а с другой стороны, чтобы формат был достаточно гибким, приложения могли сохранять в файлах свою специфическую информацию, понятную только им, но не понятную другим программам-приложениям, причём при загрузке файлов MIDI непонятная другим программам-приложениям информация не приводит к недоразумениям, а просто игнорируется. В этом смысле формат файлов MIDI можно сравнить с форматами текстовой информации. Различные программы-секвенсоры способны читать MIDI-файлы, подобно тому, как различные текстовые редакторы читают [ASCII](https://ru.wikipedia.org/wiki/ASCII)-файлы, которые могут содержать вспомогательную информацию, понятную лишь данному редактору. Но в отличие от ASCII-файлов MIDI-файлы содержат цифровую информацию, и к тому же эта информация сохранена в виде записей, то есть групп байтов, которые содержат свой заголовок, состоящий из идентификатора записи и длины записи. Эти записи могут форматироваться, загружаться, игнорироваться и т. д. независимо друг от друга. Для осуществления работы с записями программы-приложения используют дополнительную информацию, записываемую в MIDI-файл. Например, возможно, программа «захочет» сохранить флаг, указывающий на то, что пользователь установил включенным звук метронома. Программа может вставить этот флаг в MIDI-файл таким образом, что другая программа-приложение сможет пропустить этот флаг без внимания. В будущем, возможно, существующий формат MIDI будет расширен и появятся новые типы записей. Новые программы для работы с MIDI-файлами будут распознавать и новые типы записей. Однако старые MIDI-файлы могут быть воспроизведены в своем исходном виде. Формат MIDI задуман таким образом, что с его расширениями будут совместимы более ранние его версии.

Cформулируем основные задачи, решаемые данным дипломным проектом, а также проведем анализ рынка на проекты схожей тематики. Выявим преимущества и недостатки сторонних проектов, а также определимся с функциональностью, которая будет индивидуальна для разрабатываемого комплекса.

Основные задачи, которые решает разрабатываемый комплекс могут быть сформулированы таким образом:

* Реализация алгоритма преобразования mp3 и wav мелодий в midi мелодию.
* Реализация расширяемого аппаратного комплекса для проигрывания мелодий на устройствах ввода-вывода.
* Предоставления удобного пользовательского интерфейса для управления проигрыванием мелодии.

Рассмотрим уже существующие сервисы, исходя из задач, решаемых разрабатываемым программным продуктом.

# 2 СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ

* 1. **Выбор целевой платформы и средств разработки**

Задачей дипломного проекта является проектирование приложения для обработки и хранения звуковых файлов, а также методов управления аппаратной частью.

Исходя из этих условий и нужно выбирать подходящую операционную систему и инструменты разработки приложения.

Так как в первую очередь приложение будет проектироваться для обычных пользователей, имеет смысл выбрать наиболее популярную операционную систему. На данный момент господствующее место на рынке операционных систем занимает компания Microsoft. Целевыми операционными система для дипломного проекта были выбраны Windows 7, Windows 8, Windows 8.1, Windows 10, т.к. вместе они занимают более 77% рынка операционных систем (см. рисунок 2.1.1).

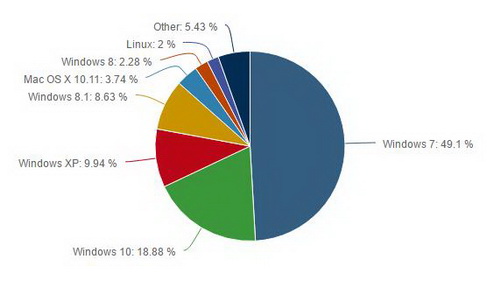


Рисунок 2.1.1 – Диаграмма разбиения рынка операционных систем

Проанализировав основные операции, которые будет выполнять проектируемое приложение, а именно: взаимодействие с файловой системой, работа с базой данных, реализация алгоритмов дискретизации, взаимодействие с контроллером Arduino, предоставление пользователям стандартного Windows интерфейса – было решено выбрать язык программирования C# и фреймворк .NET.

На сегодняшний момент язык программирования C# один из самых мощных, быстро развивающихся и востребованных языков в ИТ-отрасли. В настоящий момент на нем пишутся самые различные приложения: от небольших десктопных программ до крупных веб-порталов и веб-сервисов, обслуживающих ежедневно миллионы пользователей.

По сравнению с другими языками C# достаточно молодой, но в то же время он уже прошел большой путь. Первая версия языка вышла вместе с релизом Microsoft Visual Studio .NET в феврале 2002 года. Текущей версией языка является версия C# 6.0, которая вышла в 20 июля 2015 года вместе с Visual Studio 2015. C# поддерживает полиморфизм, наследование, перегрузку операторов, статическую типизацию. Объектно-ориентированный подход позволяет решить задачи по построению крупных, но в тоже время гибких, масштабируемых и расширяемых приложений. И C# продолжает активно развиваться, и с каждой новой версией появляется все больше интересных функции, как, например, лямбды, динамическое связывание, асинхронные методы. Когда говорят C#, нередко имеют в виду технологии платформы .NET**.** И, наоборот, когда говорят .NET, нередко имеют в виду C#. Однако, хотя эти понятия связаны, отождествлять их неверно. Язык C# был создан специально для работы с фреймворком .NET, однако само понятие .NET несколько шире. Фреймворк .NET представляет мощную платформу для создания приложений. Можно выделить следующие ее основные черты:

* Поддержка нескольких языков. Основой платформы является общеязыковая среда исполнения Common Language Runtime (CLR), благодаря чему .NET поддерживает несколько языков: наряду с C# это также VB.NET, C++, F#, а также различные диалекты других языков, привязанные к .NET, например, Delphi.NET. При компиляции код на любом из этих языков компилируется в сборку на общем языке CIL (Common Intermediate Language) - своего рода ассемблер платформы .NET. Поэтому мы можем сделать отдельные модули одного приложения на отдельных языках.
* Кроссплатформенность. .NET является переносимой платформой (с некоторыми ограничениями). Например, последняя версия платформы на данный момент .NET Framework поддерживается на большинстве современных ОС Windows (Windows 10/8.1/8/7/Vista). А благодаря проекту Mono можно создавать приложения, которые будут работать и на других ОС семейства Linux, в том числе на мобильных платформах Android и iOS.
* Мощная библиотека классов. .NET представляет единую для всех поддерживаемых языков библиотеку классов. И какое бы приложение мы не собирались писать на C# - текстовый редактор, чат или сложный веб-сайт - так или иначе мы задействуем библиотеку классов .NET.
* Разнообразие технологий. Общеязыковая среда исполнения CLR и базовая библиотека классов являются основой для целого стека технологий, которые разработчики могут задействовать при построении тех или иных приложений. Например, для работы с базами данных в этом стеке технологий предназначена технология ADO.NET. Для построения графических приложений с богатым насыщенным интерфейсом - технология UWP.

Также еще следует отметить такую особенность языка C# и фреймворка .NET, как автоматическая сборка мусора. А это значит, что нам в большинстве случаев не придется, в отличие от С++, заботиться об освобождении памяти. Вышеупомянутая общеязыковая среда CLR сама вызовет сборщик мусора и очистит память. Код на C# компилируется в приложения или сборки с расширениями exe или dll на языке CIL. Далее при запуске на выполнение подобного приложения происходит JIT-компиляция (Just-In-Time) в машинный код, который затем выполняется. При этом, поскольку наше приложение может быть большим и содержать кучу инструкций, в текущий момент времени будет компилироваться лишь та часть приложения, к которой непосредственно идет обращение. Если мы обратимся к другой части кода, то она будет скомпилирована из CIL в машинный код. При том уже скомпилированная часть приложения сохраняется до завершения работы программы. В итоге это повышает производительность.

Для построения пользовательского интерфейса в дипломном проекте используется технология UWP.

* 1. **Планирование логических модулей**

Весь разрабатываемый проект разбивается на отдельные модули. Данный подход позволяет добиться гибкой структуры программы. Хорошо продуманная модель абстракций каждого блока позволяет менять внутреннюю структуру блока, сохраняя оговоренные интерфейсы, что в итоге никоим образом не отразится на функциональности других модулей. Разбиение на модули происходит исходя из задач, решаемых программных обеспечением:

* взаимодействие с файловой системой;
* работа с базой данных;
* реализация алгоритмов дискретизации;
* взаимодействие с контроллером Arduino;
* предоставление пользователям стандартного Windows интерфейса;

Каждый логический модуль решает одну или несколько вышеперечисленных задач. Кроме того, каждый модуль программного обеспечения взаимодействует с остальными модулями, что обеспечивает работоспособность всего приложения в целом. Связь реализуется при помощи интерфейсов и IoC-контейнера.  IoC-контейнер – это [компоновщик](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%BE%D0%BD%D0%BE%D0%B2%D1%89%D0%B8%D0%BA), который собирает не [объектные файлы](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D1%8B%D0%B9_%D0%BC%D0%BE%D0%B4%D1%83%D0%BB%D1%8C), а объекты [ООП](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) ([экземпляры класса](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5))) во [время исполнения](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%BD%D1%82%D0%B0%D0%B9%D0%BC) программы. Очевидно, для реализации подобной идеи было необходимо создать не только сам компоновщик, но и [фабрику](https://ru.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B1%D1%80%D0%B8%D1%87%D0%BD%D1%8B%D0%B9_%D0%BC%D0%B5%D1%82%D0%BE%D0%B4_(%D1%88%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)), производящую объекты. Для разрабатываемой программного средства были выделены модули, которые представлены в обобщенной структурной схеме на рисунке 2.2.1

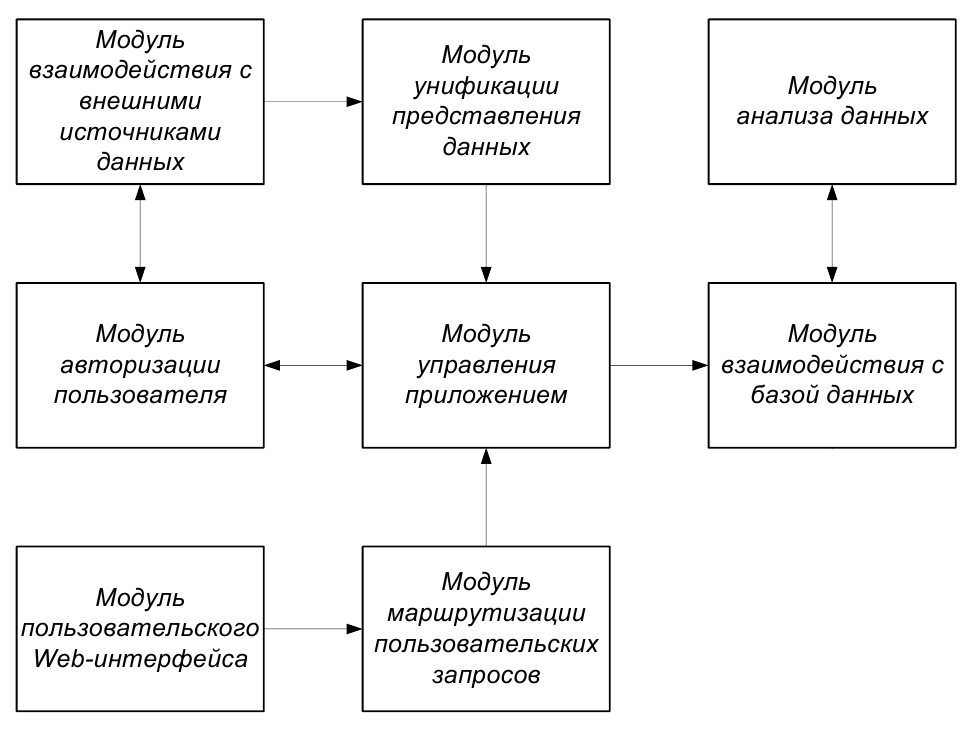


Рисунок. 2.2.1 – Обобщенная структурная схема