ППОИС

Часть 1

Модульность

Модульность

Способ скрыть реализацию/детали

Модель черного ящика

В C++ для этого есть инструментарий:

1. Разделение на .h и .cpp
2. Статические и динамические библиотеки
3. Пространство имен

Модульность

Ключевое слово **extern**:

- `extern int i;`
- `extern "C" int printf(const char *fmt, ...);`

Ключевые слова **__declspec(dllexport)**

```
class __declspec(dllexport) C {
   int i;
   virtual int func( void ) { return 1; }
};
```

Пространство имен

Устранение конфликтов в именах

```
namespace Boo {
    namespace Doo {
        const int g_x = 7;
    }
}

namespace Foo = Boo::Doo; // Foo теперь считается как Boo::Doo

int main() {
    std::cout << Foo::g_x; // это, на самом деле, Boo::Doo::g_x
    return 0;
}
```

Директива using

```cpp
int main()
{
    using namespace std; // "using-директива" сообщает компилятору, что мы используем все объекты из пространства имен std!
    cout << "Hello, world!"; // так что никакого префикса std:: здесь уже не нужно!
    return 0;
}
```

```cpp
int main()
{
    {
        using namespace Boo;
        // Здесь всё относится к пространству имен Boo::
    } // действие using namespace Boo заканчивается здесь

    {
        using namespace Foo;
        // Здесь всё относится к пространству имен Foo::
    } // действие using namespace Foo заканчивается здесь

    return 0;
}
```

Введение в понятие архитектуры

Архитектура программной системы

Архитектура программной системы — это форма, которая придается системе ее создателями . Эта форма образуется делением системы на компоненты, их организацией и определением способов взаимодействий между ними .

Цель формы — упростить разработку, развертывание и сопровождение программной системы, содержащейся в ней .

Главная стратегия такого упрощения в том, чтобы как можно дольше иметь как можно больше вариантов.

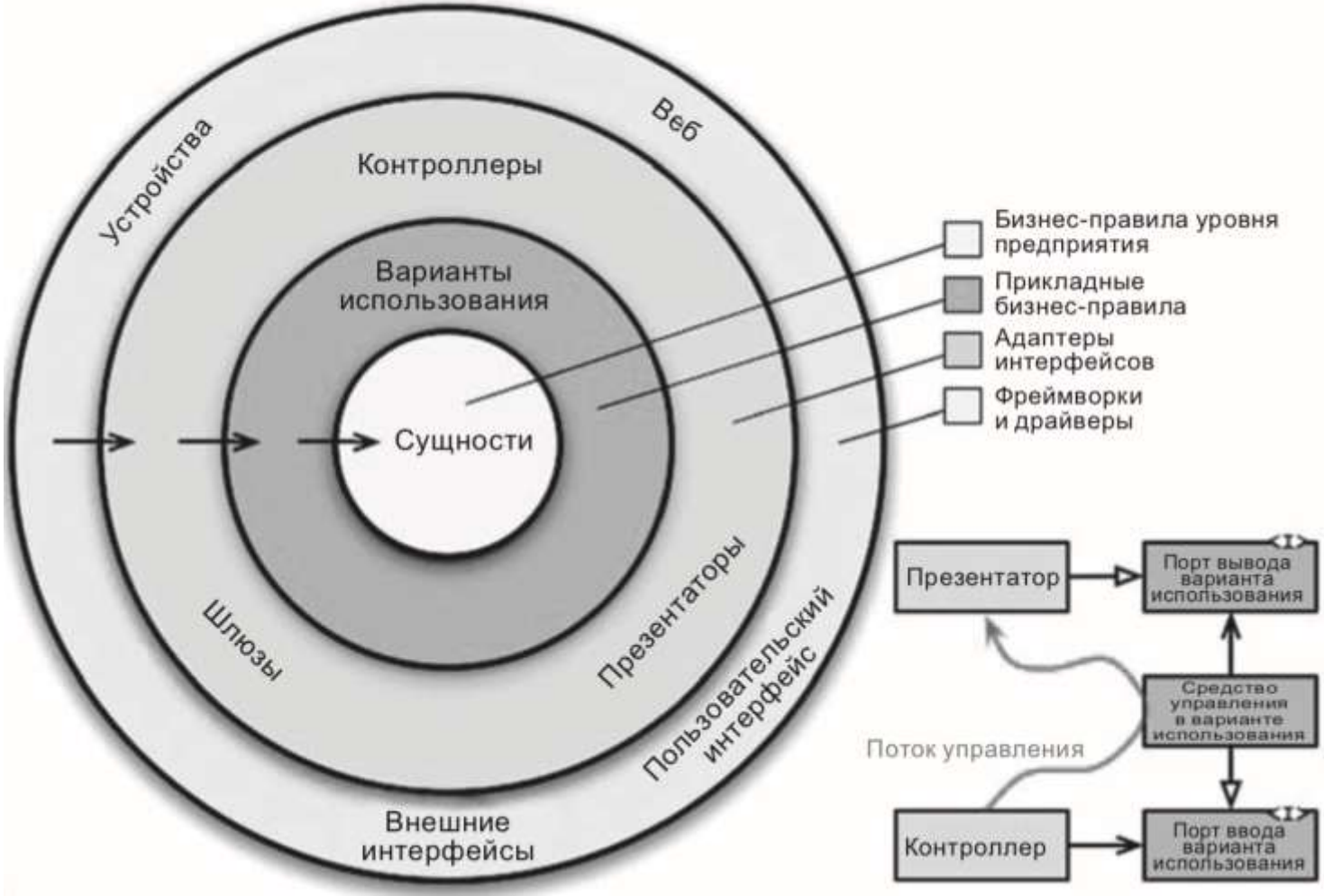
Архитектура программной системы

Главное предназначение архитектуры — поддержка жизненного цикла системы.

Хорошая архитектура делает систему легкой в освоении, простой в разработке, сопровождении и развертывании.

Конечная ее цель — минимизировать затраты на протяжении срока службы системы и максимизировать продуктивность программиста.

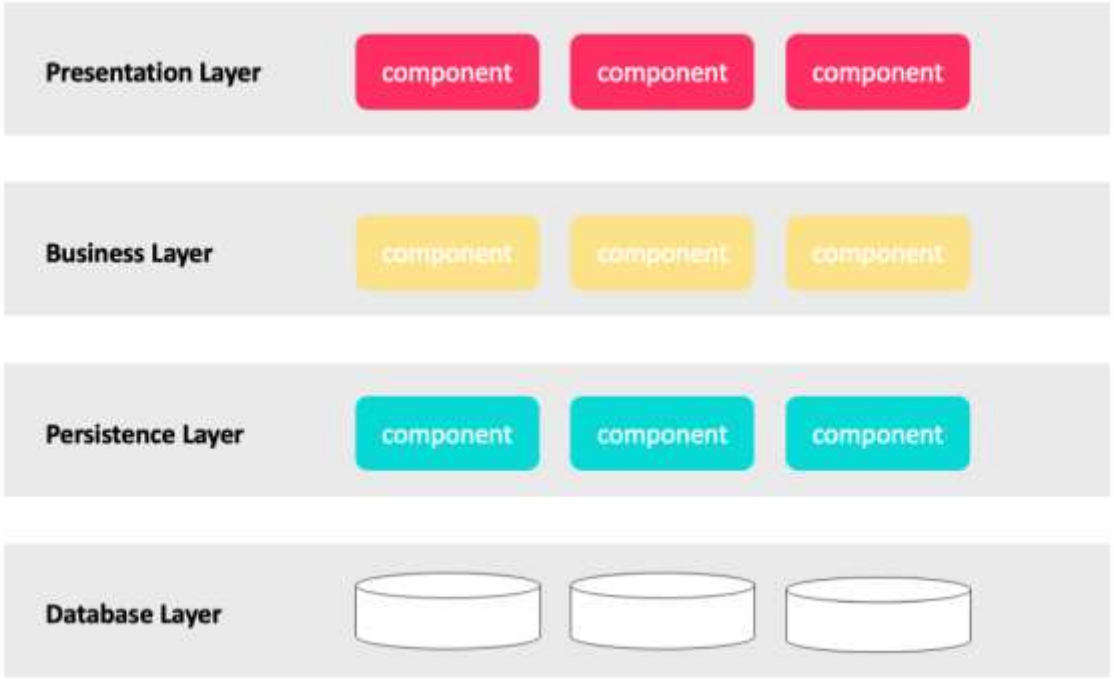
Чистая архитектура



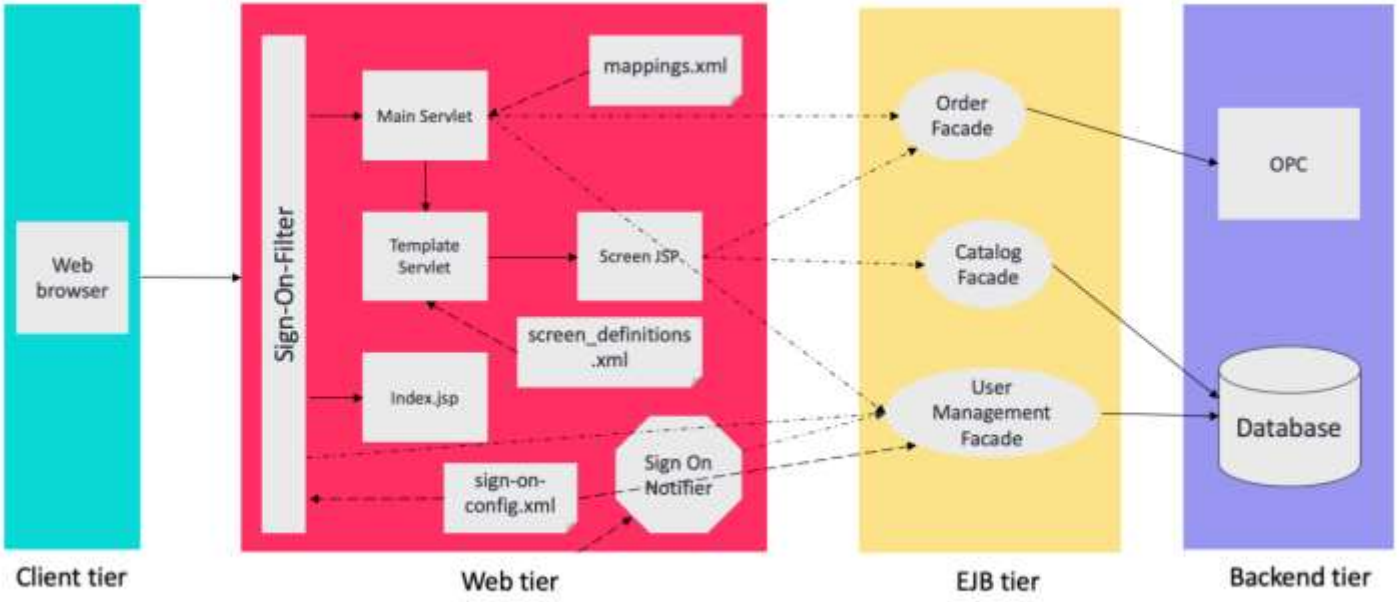
Часто используемые архитектурные шаблоны

1. Layered Architecture
2. Multi-Tier Pattern
3. Pipe and Filter
4. Client Server
5. Model View Controller
6. Event Driven Architecture
7. Microservices Architecture

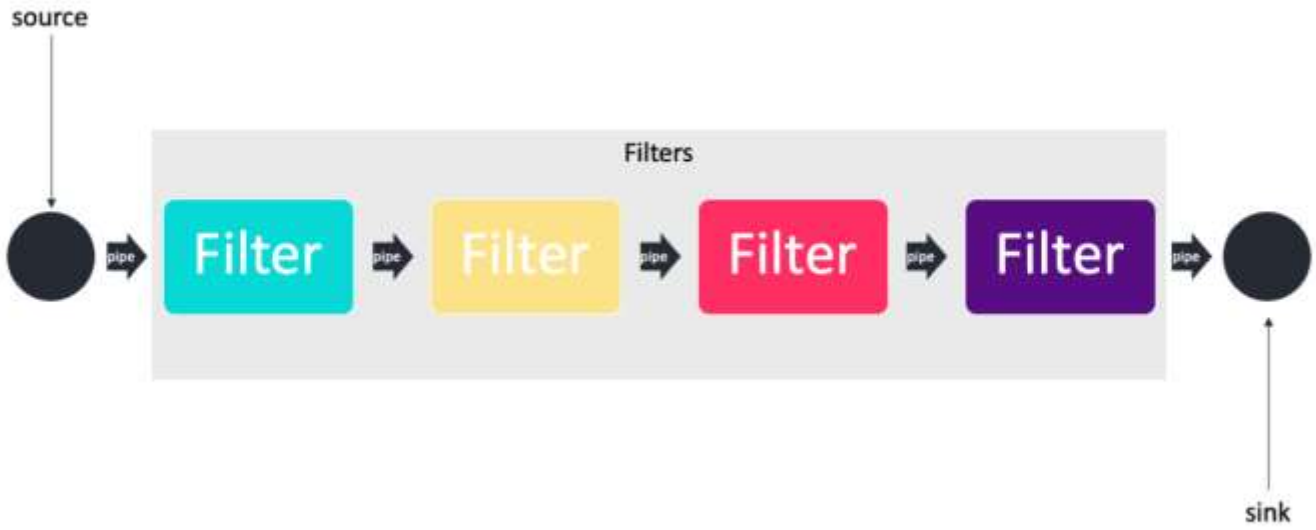
Layered Architecture



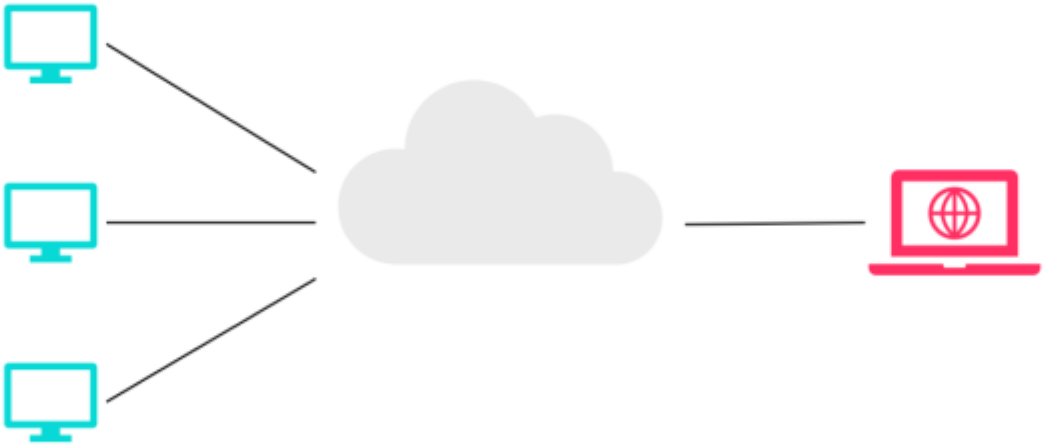
Multi-Tier Pattern



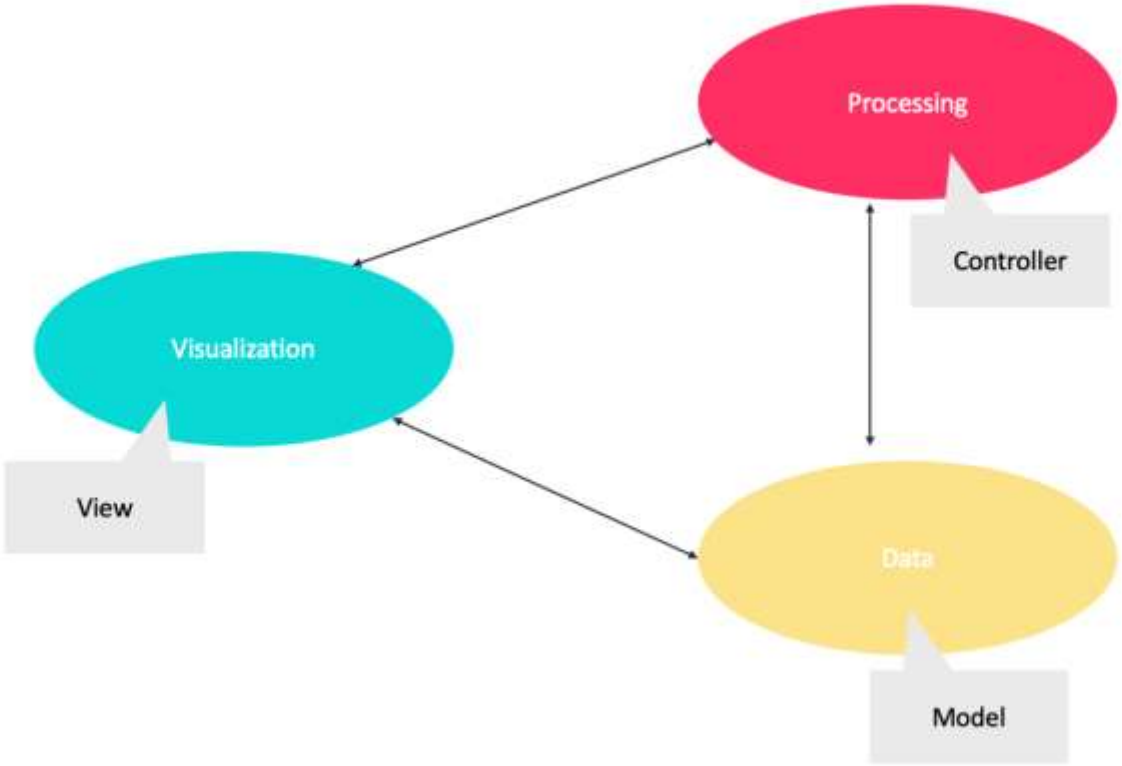
Pipe and Filter



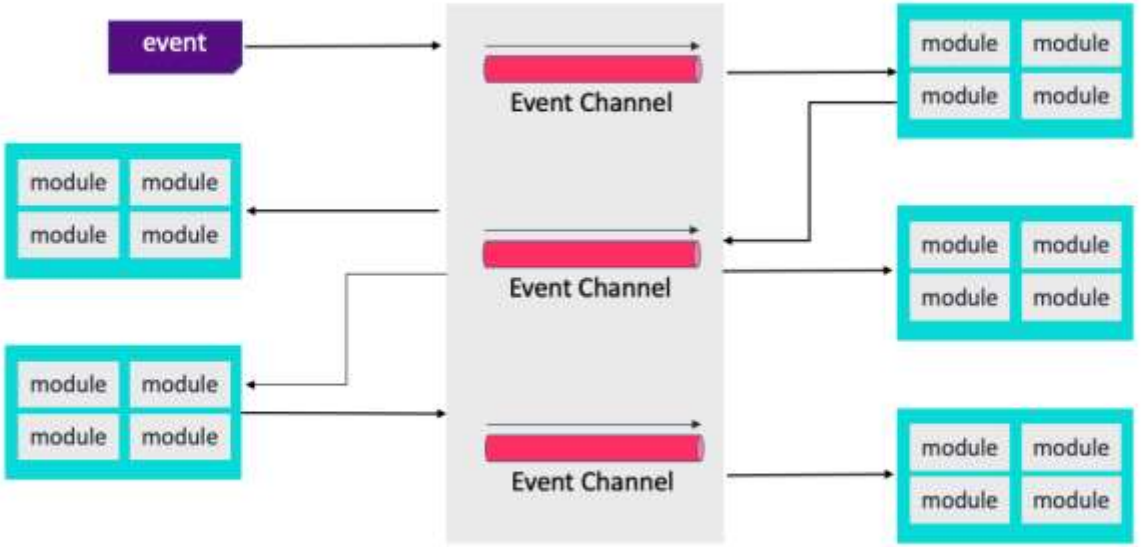
Client Server



Model View Controller



Event Driven Architecture



Microservices Architecture