Микросервисы на C#

C# на сервере, архитектура, подводные камни

Марк Шевченко

Тимлид, backend-программист

Сайт	http://markshevchenko.pro/
Telegram	https://t.me/markshevchenko
Московский клуб программистов	http://prog.msk.ru/

C#

Почему не C#?

Почему не C#?

Почему не C#?

- Продуман
- Интеграция с C
- Простота
- Django и Flask
- Зарплаты
- Порог входа
- Менеджер проектов
- Популярность

- Старожил
- ~~Legacy~~ Много кода
- Много разработчиков
- Facebook
- Порог входа
- Простые сайты
- Развивается

- Параллельность
- Простота
- Скорость
- Сборка мусора
- Google

- Простота
- Менеджер проектов
- Библиотека
- Много разработчиков

Почему не C#?

- Язык новый, мало кода, фреймворков и разработчиков. Не интегрируется с C.
- Сложный, с высоким порогом входа, плохо продуман, плохо читаем.
- Перестал развиваться.
- Не популярен. И программисты дорогие.
- Нет менеджера пакетов.
- Не умеет параллельную разработку.
- Медленный.

Почему C#?

- Статическая типизация.
- Развитой объектно-ориентированный язык уровня C++.
- Обобщённое программирование с поддержкой на уровне виртуальной машины.
- Рефлексия.
- Вывод типов, деревья выражений, вариантность типов.
- Лямбды. Сопоставление с образцом. ФП.
- Динамическая типизация.
- Асинхронное программирование.

Деревья выражений

```
class User
{
    public Guid Id { get; set; }

    public DateTime CreatedAt { get; set; }

    public string Name { get; set; }
}
```

Деревья выражений

```
var users = new[]
{
    new User { Id = new Guid("{6217017B-F1C7-48A8-86E5-BA6DD0CE1D94}")
             , CreatedAt = new DateTime(2019, 11, 4, 13, 7, 00)
             , Name = "alexivanov"},
    new User { Id = new Guid("{139571FD-0F5E-4F03-A240-C6A7D2BB279A}")
             , CreatedAt = new DateTime(2019, 10, 14, 11, 18, 00)
             , Name = "BorisZhdanov" },
    new User { Id = new Guid("{CCE17931-74A5-47A7-82FC-011B37C75D7E}")
             , CreatedAt = new DateTime(2019, 10, 11, 1, 45, 00)
             , Name = "alexandr-zhukov" },
    new User { Id = new Guid("{ADCF3E14-70E0-4507-9A26-315E2CA377FD}")
             , CreatedAt = new DateTime(2019, 11, 2, 2, 1, 00)
             , Name = "alexey-bitov" },
    . . .
};
```

Деревья выражений

```
var names1 = from user in users
             where user.Name.StartsWith("alex")
             orderby user.CreatedAt
             select user.Name;
```

Деревья выражений

```
var names1 = from user in users
             where user.Name.StartsWith("alex")
             orderby user.CreatedAt
             select user.Name;


var names2 = users.Where(x => x.Name.StartsWith("alex"))
                  .OrderBy(x => x.CreatedAt)
                  .Select(x => x.Name);
```

Деревья выражений

```
var names2 = users.Where(x => x.Name.StartsWith("alex"))
                  .OrderBy(x => x.CreatedAt)
                  .Select(x => x.Name);


foreach (var name in names1)
{
    Console.WriteLine(name);
}
```

Деревья выражений

```
var names2 = users.Where(x => x.Name.StartsWith("alex"))
                  .OrderBy(x => x.CreatedAt)
                  .Select(x => x.Name);


foreach (var name in names1)
{
    Console.WriteLine(name);
}

alexandr-zhukov
alexey-bitov
alexivanov
```

Деревья выражений

```
var names2 = users.Where(x => x.Name.StartsWith("alex"))
                  .OrderBy(x => x.CreatedAt)
                  .Select(x => x.Name);


static IEnumerable<TSource> Where<TSource>(this IEnumerable<TSource> source,
                                           Func<TSource, bool> predicate)
{
    return new WhereEnumerableIterator<TSource>(source, predicate);
}
```

Деревья выражений

```
var names2 = users.Where(x => x.Name.StartsWith("alex"))
                  .OrderBy(x => x.CreatedAt)
                  .Select(x => x.Name);


static IEnumerable<TSource> Where<TSource>(this IEnumerable<TSource> source,
                                           Func<TSource, bool> predicate)
{
    return new WhereEnumerableIterator<TSource>(source, predicate);
}
```

Деревья выражений

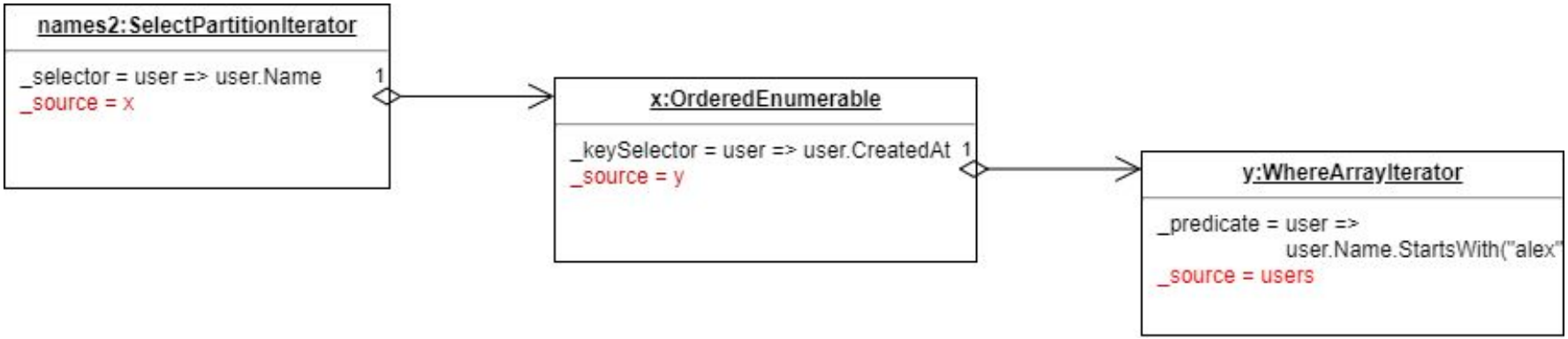
```
var names2 = users.Where(x => x.Name.StartsWith("alex"))
                  .OrderBy(x => x.CreatedAt)
                  .Select(x => x.Name);


static IEnumerable<TSource> Where<TSource>(this IEnumerable<TSource> source,
                                           Func<TSource, bool> predicate)
{
    return new WhereEnumerableIterator<TSource>(source, predicate);
}
```

Деревья выражений

```
var names2 = users.Where(x => x.Name.StartsWith("alex"))
                  .OrderBy(x => x.CreatedAt)
                  .Select(x => x.Name);


static IEnumerable<TSource> Where<TSource>(this IEnumerable<TSource> source,
                                           Func<TSource, bool> predicate)
{
    return new WhereEnumerableIterator<TSource>(source, predicate);
}
```

Деревья выражений

```
var names2 = users.Where(x => x.Name.StartsWith("alex"))
                  .OrderBy(x => x.CreatedAt)
                  .Select(x => x.Name);
```



Деревья выражений

```
var dbContext = new PresentationDbContext();
var names4 = dbContext.Users
                      .Where(x => x.Name.StartsWith("alex"))
                      .OrderBy(x => x.CreatedAt)
                      .Select(x => x.Name);
```

Деревья выражений

```
var dbContext = new PresentationDbContext();
var names4 = dbContext.Users
                      .Where(x => x.Name.StartsWith("alex"))
                      .OrderBy(x => x.CreatedAt)
                      .Select(x => x.Name);
```

```
SELECT "u"."Name"
FROM "Users" AS "u"
WHERE "u"."Name" IS NOT NULL AND ("u"."Name" LIKE 'alex%')
ORDER BY "u"."CreatedAt"
```

Деревья выражений

```
var names2 = users.Where(x => x.Name.StartsWith("alex"))
                  .OrderBy(x => x.CreatedAt)
                  .Select(x => x.Name);

var names3 = users.AsQueryable()
                  .Where(x => x.Name.StartsWith("alex"))
                  .OrderBy(x => x.CreatedAt)
                  .Select(x => x.Name);
```

Деревья выражений

```
var names3 = users.AsQueryable()
                  .Where(x => x.Name.StartsWith("alex"))
                  .OrderBy(x => x.CreatedAt)
                  .Select(x => x.Name);

static IQueryable<TSource> Where<TSource>(this IQueryable<TSource> source,
                                          Expression<Func<TSource, bool>> predicat
{
    return source.Provider.CreateQuery<TSource>(Expression.Call(null,
           GetMethodInfo(Queryable.Where, source, predicate), new Expression[]
           { source.Expression, Expression.Quote(predicate) } ));
}
```

Деревья выражений

```
var names3 = users.AsQueryable()
                  .Where(x => x.Name.StartsWith("alex"))
                  .OrderBy(x => x.CreatedAt)
                  .Select(x => x.Name);

static IQueryable<TSource> Where<TSource>(this IQueryable<TSource> source,
                                          Expression<Func<TSource, bool>> predicat
{
    return source.Provider.CreateQuery<TSource>(Expression.Call(null,
           GetMethodInfo(Queryable.Where, source, predicate), new Expression[]
           { source.Expression, Expression.Quote(predicate) } ));
}
```

Деревья выражений

```
var names3 = users.AsQueryable()
                  .Where(x => x.Name.StartsWith("alex"))
                  .OrderBy(x => x.CreatedAt)
                  .Select(x => x.Name);

static IQueryable<TSource> Where<TSource>(this IQueryable<TSource> source,
                                          Expression<Func<TSource, bool>> predicat
{
    return source.Provider.CreateQuery<TSource>(Expression.Call(null,
           GetMethodInfo(Queryable.Where, source, predicate), new Expression[]
           { source.Expression, Expression.Quote(predicate) } ));
}
```

Деревья выражений

```
var names3 = users.AsQueryable()
                  .Where(x => x.Name.StartsWith("alex"))
                  .OrderBy(x => x.CreatedAt)
                  .Select(x => x.Name);

static IQueryable<TSource> Where<TSource>(this IQueryable<TSource> source,
                                          Expression<Func<TSource, bool>> predicat
{
    return source.Provider.CreateQuery<TSource>(Expression.Call(null,
           GetMethodInfo(Queryable.Where, source, predicate), new Expression[]
           { source.Expression, Expression.Quote(predicate) } ));
}
```

Деревья выражений

```
.Call System.Linq.Queryable.Select(
  .Call System.Linq.Queryable.OrderBy(
    .Call System.Linq.Queryable.Where(
      .Constant<System.Linq.EnumerableQuery`1[Presentation.User]>(Presentation.User[]),
      '(.Lambda #Lambda1<System.Func`2[Presentation.User,System.Boolean]>)),
    '(.Lambda #Lambda2<System.Func`2[Presentation.User,System.DateTime]>)),
  '(.Lambda #Lambda3<System.Func`2[Presentation.User,System.String]>))

.Lambda #Lambda1<System.Func`2[Presentation.User,System.Boolean]>(Presentation.User $x) {
  .Call ($x.Name).StartsWith("alex")
}

.Lambda #Lambda2<System.Func`2[Presentation.User,System.DateTime]>(Presentation.User $x) {
  $x.CreatedAt
}

.Lambda #Lambda3<System.Func`2[Presentation.User,System.String]>(Presentation.User $x) {
  $x.Name
}
```

Деревья выражений

```
.Call System.Linq.Queryable.Select(
  .Call System.Linq.Queryable.OrderBy(
    .Call System.Linq.Queryable.Where(
      .Constant<System.Linq.EnumerableQuery`1[Presentation.User]>(Presentation.User[]),
      '(.Lambda #Lambda1<System.Func`2[Presentation.User,System.Boolean]>)),
    '(.Lambda #Lambda2<System.Func`2[Presentation.User,System.DateTime]>)),
  '(.Lambda #Lambda3<System.Func`2[Presentation.User,System.String]>))

.Lambda #Lambda1<System.Func`2[Presentation.User,System.Boolean]>(Presentation.User $x) {
  .Call ($x.Name).StartsWith("alex")
}

.Lambda #Lambda2<System.Func`2[Presentation.User,System.DateTime]>(Presentation.User $x) {
  $x.CreatedAt
}

.Lambda #Lambda3<System.Func`2[Presentation.User,System.String]>(Presentation.User $x) {
  $x.Name
}
```

Миграции

```
class PresentationDbContext : DbContext
{
    public DbSet<User> Users { get; set; }

    protected override void OnModelCreating(ModelBuilder builder)
    {
        builder.Entity<User>()
               .HasData(Program.Users);
    }
}
```

Миграции

```
class PresentationDbContext : DbContext
{
    public DbSet<User> Users { get; set; }

    protected override void OnModelCreating(ModelBuilder builder)
    {
        builder.Entity<User>()
               .HasData(Program.Users);
    }
}
```

```
Add-Migration InitialCreate
```

Миграции

```
class InitialCreate : Migration
{
    protected override void Up(MigrationBuilder migrationBuilder) {
        migrationBuilder.CreateTable(name: "Users", columns: table => new
        {
            Id = table.Column<Guid>(nullable: false),
            CreatedAt = table.Column<DateTime>(nullable: false),
            Name = table.Column<string>(nullable: true)
        },
        constraints: table =>
        {
            table.PrimaryKey("PK_Users", x => x.Id);
        });
    }

    protected override void Down(MigrationBuilder migrationBuilder) {
        migrationBuilder.DropTable(name: "Users");
    }
}
```

Асинхронное программирование

```
var listener = new HttpListener();
listener.Prefixes.Add("http://localhost:8080/");
listener.Start();

while (true)
{
    var context = listener.GetContext();

    ProcessRequest(context);
}
```

https://medium.com/fantageek/understanding-socket-and-port-in-tcp-2213dc2e9b0c
https://www.ibm.com/support/knowledgecenter/en/SSFKSJ_7.5.0/com.ibm.mq.con.doc/q016510_.htm

Асинхронное программирование

```
private static void SendIndexHtml(HttpListenerResponse response)
{
    response.StatusCode = 200;

    using (var writer = new StreamWriter(response.OutputStream))
    {
        writer.WriteLine("<!DOCTYPE html>");
        writer.WriteLine("<html lang='en'>");
        writer.WriteLine("  <head>");
        writer.WriteLine("  <meta charset='utf-8' />");
        writer.WriteLine("  <title>Example HTTP server</title>");
        writer.WriteLine("  </head>");
        writer.WriteLine("  <body>");
        writer.WriteLine("    <p>Example HTTP server</p>");
        writer.WriteLine("  </body>");
        writer.WriteLine("</html>");
    }
}
```

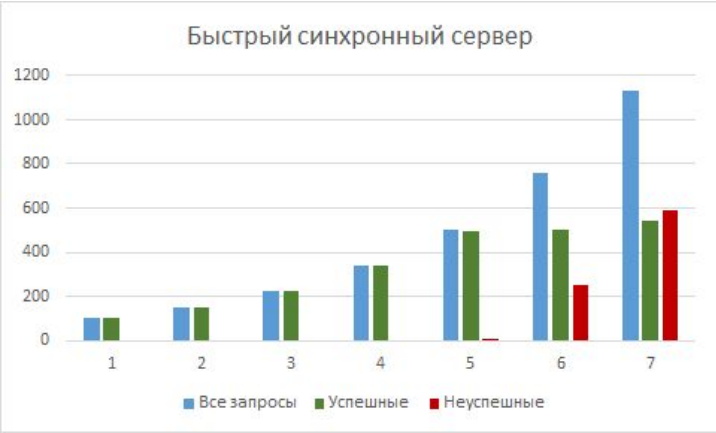
Асинхронное программирование

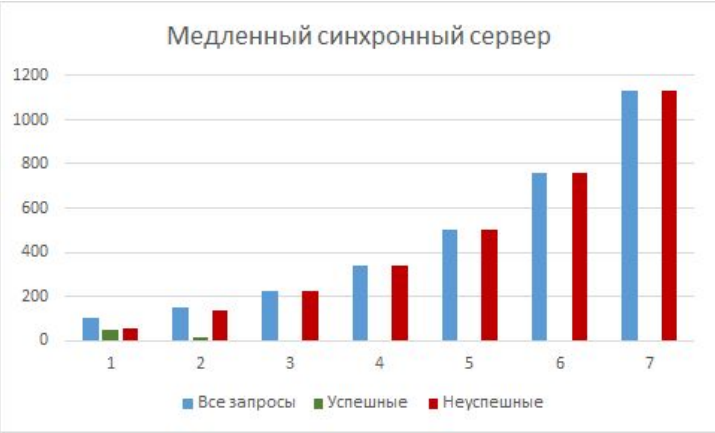
```
var listener = new HttpListener();
listener.Prefixes.Add("http://localhost:8080/");
listener.Start();

while (true)
{
    var context = listener.GetContext();

    var thread = new Thread(() => ProcessRequest(context));
    thread.Start();
}
```

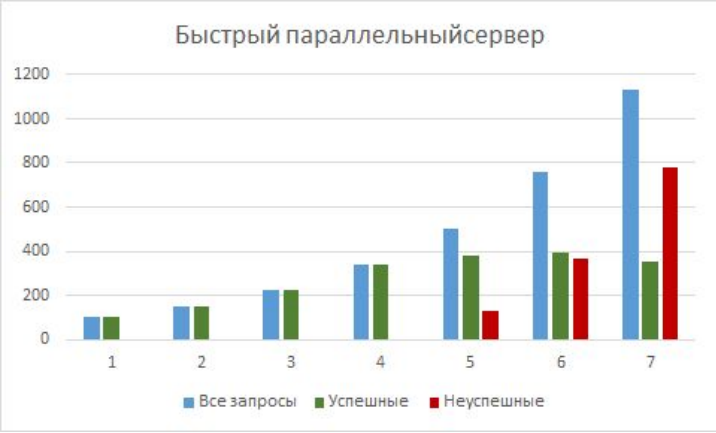
Асинхронное программирование

```
var listener = new HttpListener();
listener.Prefixes.Add("http://localhost:8080/");
listener.Start();

listener.BeginGetContext(AsyncProcessRequest, listener);
Console.ReadKey();

. . .

private static void AsyncProcessRequest(IAsyncResult ar)
{
    var listener = (HttpListener)ar.AsyncState;
    listener.BeginGetContext(AsyncProcessRequest, listener);

    var context = listener.EndGetContext(ar);
    . . .
}
```

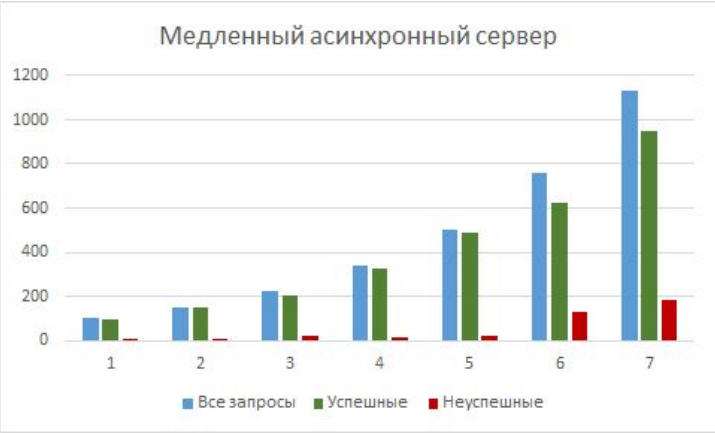
Асинхронное программирование



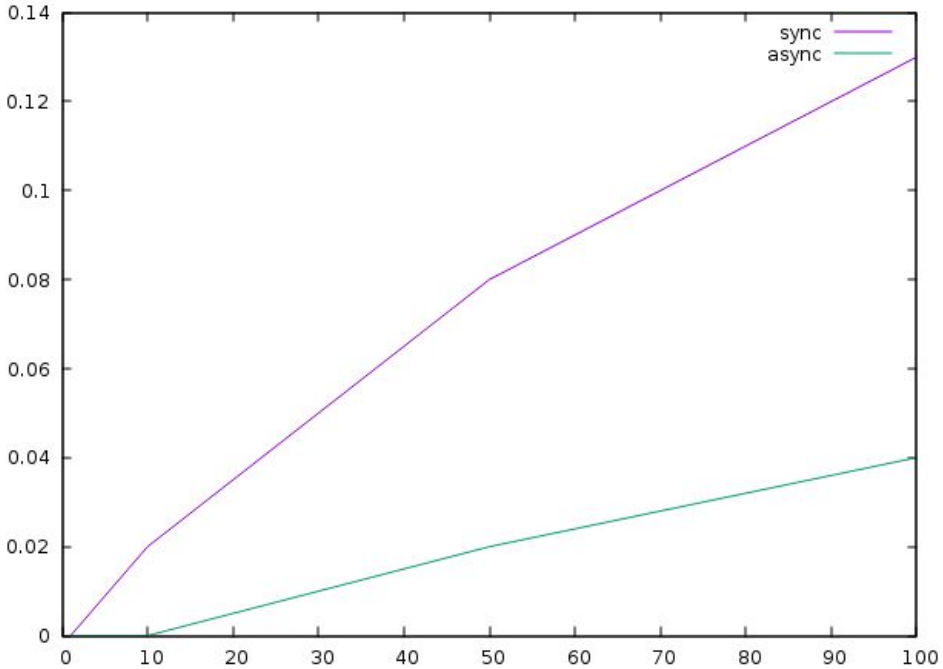




Асинхронное программирование



Асинхронное программирование



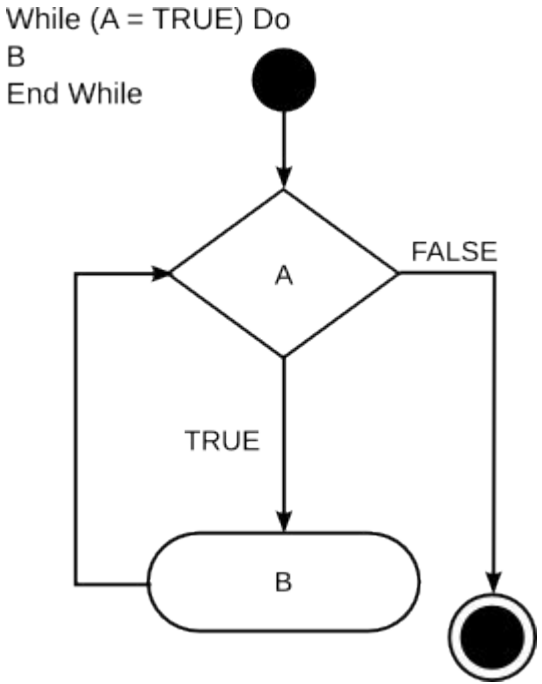
http://julien.gunnm.org/programming/linux/2017/04/15/comparison-sync-vs-async/

Асинхронное программирование

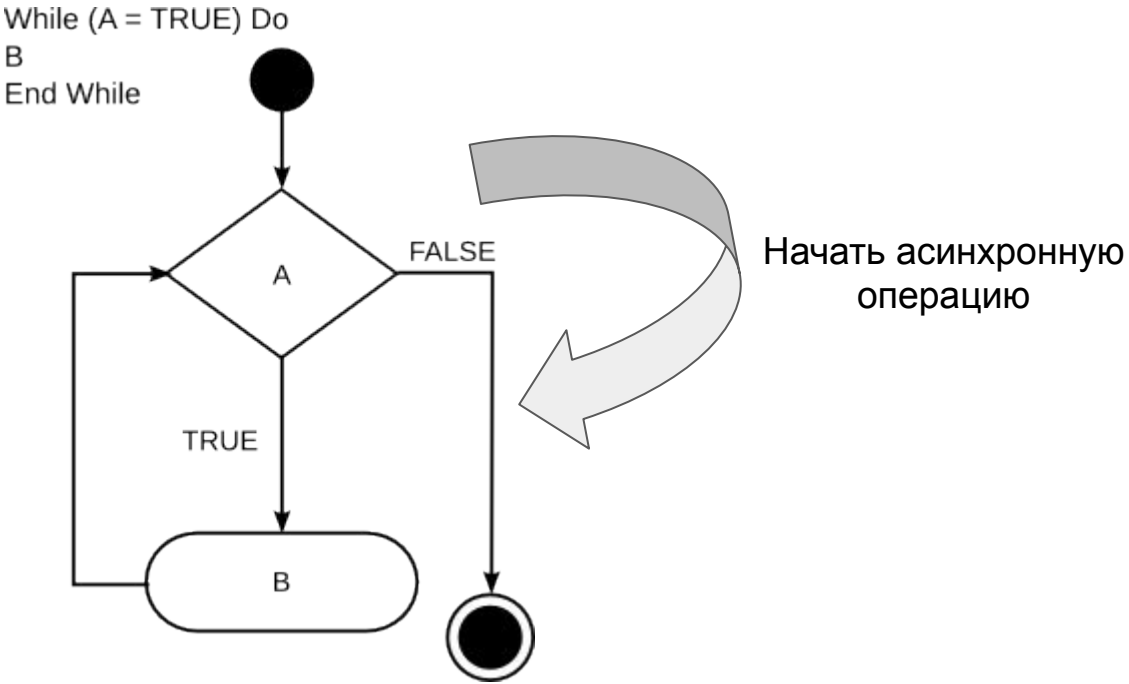
Главы 27 и 28

Асинхронное программирование

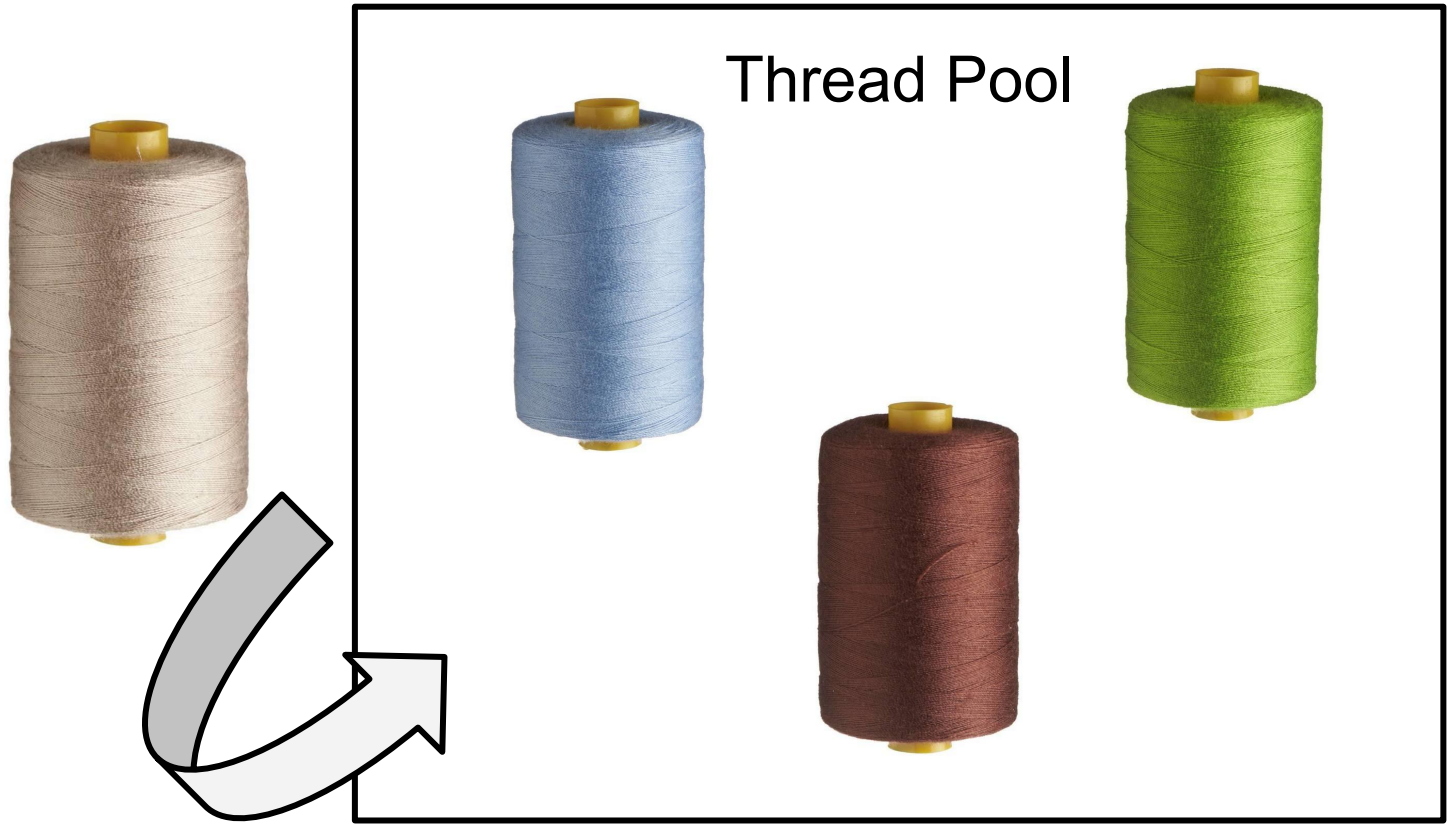
Асинхронное программирование



Асинхронное программирование



Асинхронное программирование



Асинхронное программирование

http://callbackhell.com/

```
fs.readdir(source, function (err, files) {
  if (err) {
    console.log('Error finding files: ' + err)
  } else {
    files.forEach(function (filename, fileIndex) {
      console.log(filename)
      gm(source + filename).size(function (err, values) {
        if (err) {
          console.log('Error identifying file size: ' + err)
        } else {
          console.log(filename + ' : ' + values)
          aspect = (values.width / values.height)
          widths.forEach(function (width, widthIndex) {
            height = Math.round(width / aspect)
            console.log('resizing ' + filename + 'to ' + height + 'x' + height)
            this.resize(width, height).write(dest + 'w' + width + '_' + filename, function(err) {
              if (err) console.log('Error writing file: ' + err)
            })
          }.bind(this))
        }
      })
    })
  }
})
```

Асинхронное программирование

```
var buffer = new byte[1024];
var task = tcpListener.AcceptTcpClientAsync()
                      .ContinueWith(client =>
                      {
                          var stream = client.Result.GetStream();

                          return stream.ReadAsync(buffer);
                      }).ContinueWith(read =>
                      {
                          var hash = Cryptography.MD5
                                                 .Create()
                                                 .ComputeHash(buffer, 0, read);
                          . . .
                      });
```

Асинхронное программирование

```
var listener = new HttpListener();
listener.Prefixes.Add("http://localhost:8080/");
listener.Start();

GetContextAsync(listener);
Console.ReadKey();

. . .

private static async void GetContextAsync(HttpListener listener)
{
    await Task.Yield();

    var context = await listener.GetContextAsync();
    GetContextAsync(listener);
    . . .
}
```

Асинхронное программирование

```
var listener = new HttpListener();
listener.Prefixes.Add("http://localhost:8080/");
listener.Start();

GetContextAsync(listener);
Console.ReadKey();

. . .

private static async void GetContextAsync(HttpListener listener)
{
    await Task.Yield();

    var context = await listener.GetContextAsync();
    GetContextAsync(listener);
    . . .
}
```

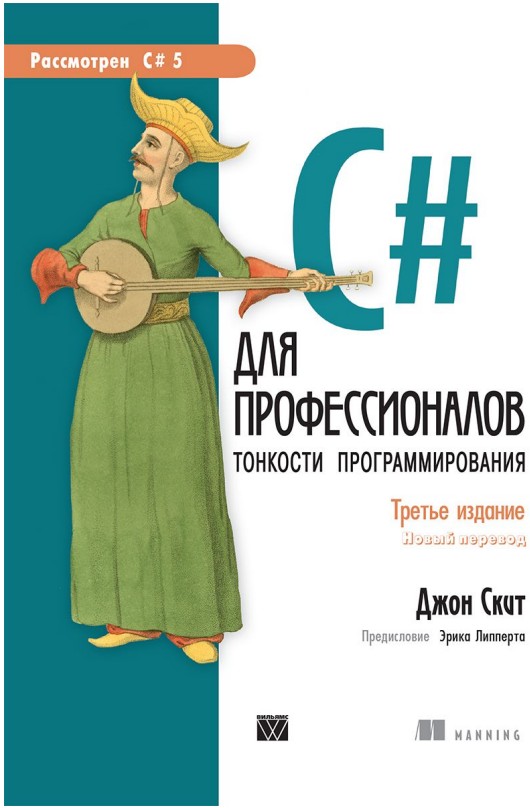
Асинхронное программирование

```
var listener = new HttpListener();
listener.Prefixes.Add("http://localhost:8080/");
listener.Start();

ProcessContextAsync(listener);
Console.ReadKey();

. . .

private static async void ProcessContextAsync(HttpListener listener)
{
    await Task.Yield();

    var context = await listener.GetContextAsync();
    ProcessContextAsync(listener);
    . . .
}
```

Асинхронное программирование

```
var listener = new HttpListener();
listener.Prefixes.Add("http://localhost:8080/");
listener.Start();

GetContextAsync(listener);
Console.ReadKey();

. . .

private static async void GetContextAsync(HttpListener listener)
{
    await Task.Yield();

    var context = await listener.GetContextAsync();
    GetContextAsync(listener);
    . . .
}
```

Асинхронное программирование

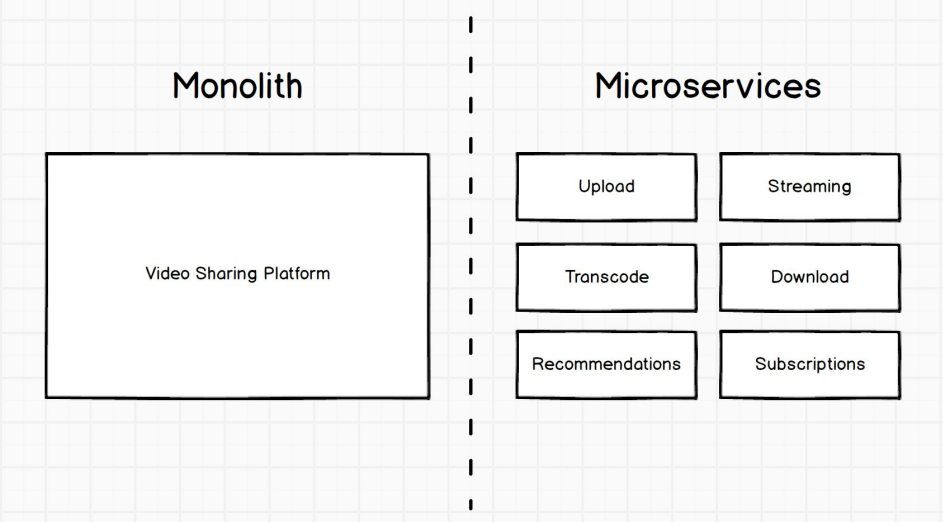


Глава 15

https://habr.com/ru/post/137317/

Микросервисы

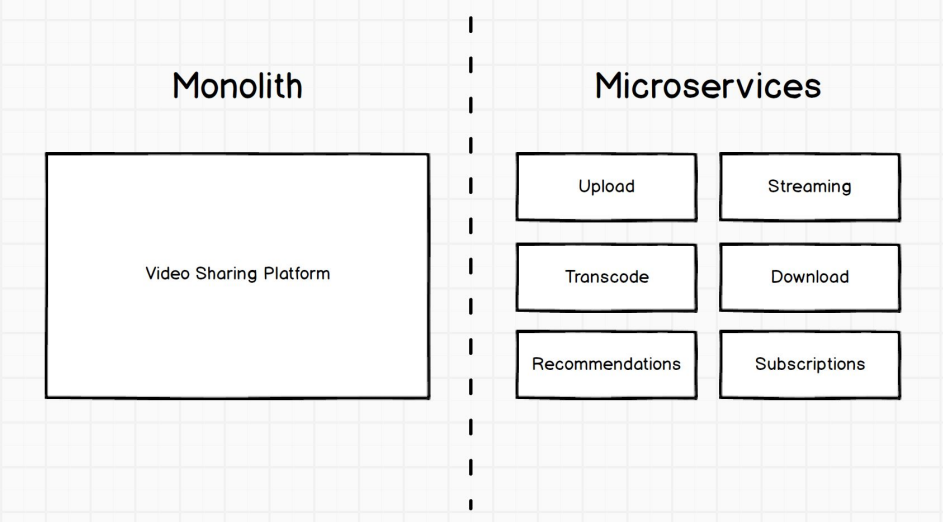
Микросервисы



- Независимое развёртывание
- Независимая разработка
- Независимое масштабирование
- Повторное использование

https://www.dwmkerr.com/the-death-of-microservice-madness-in-2018/

Микросервисы

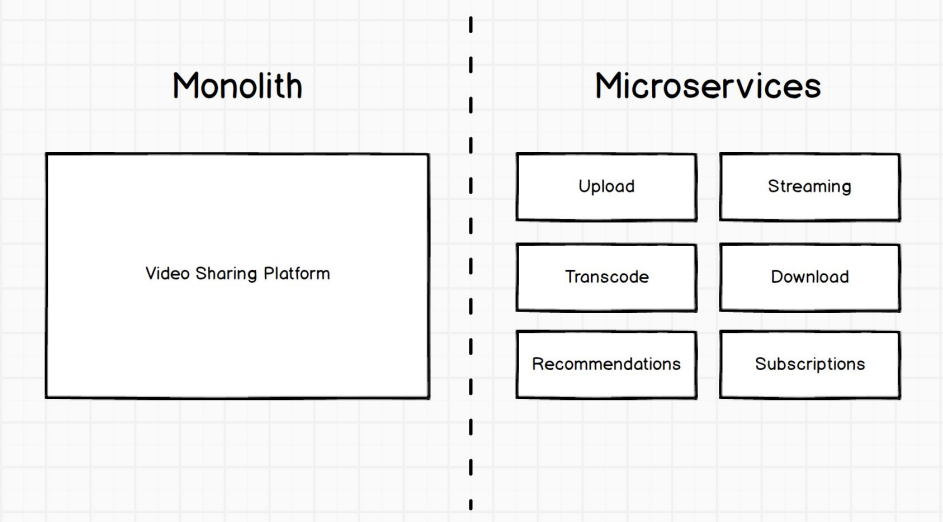


- Независимое развёртывание
- Независимая разработка
- Независимое масштабирование
- Повторное использование

https://www.dwmkerr.com/the-death-of-microservice-madness-in-2018/

CORBA, DCOM

Микросервисы

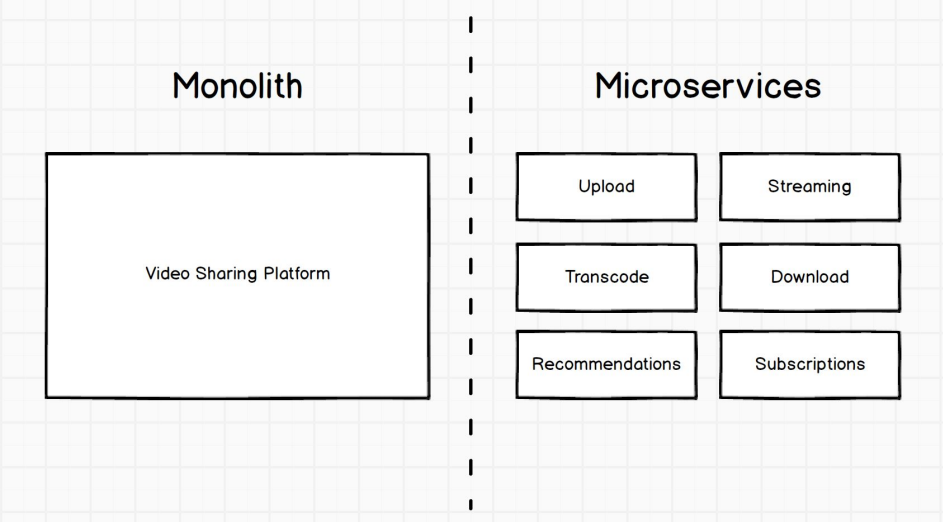


- Независимое развёртывание
- Независимая разработка
- Независимое масштабирование
- Повторное использование

https://www.dwmkerr.com/the-death-of-microservice-madness-in-2018/

~~CORBA~~, ~~DCOM~~, SOAP, REST, gRPC — веб-сервисы

Микросервисы

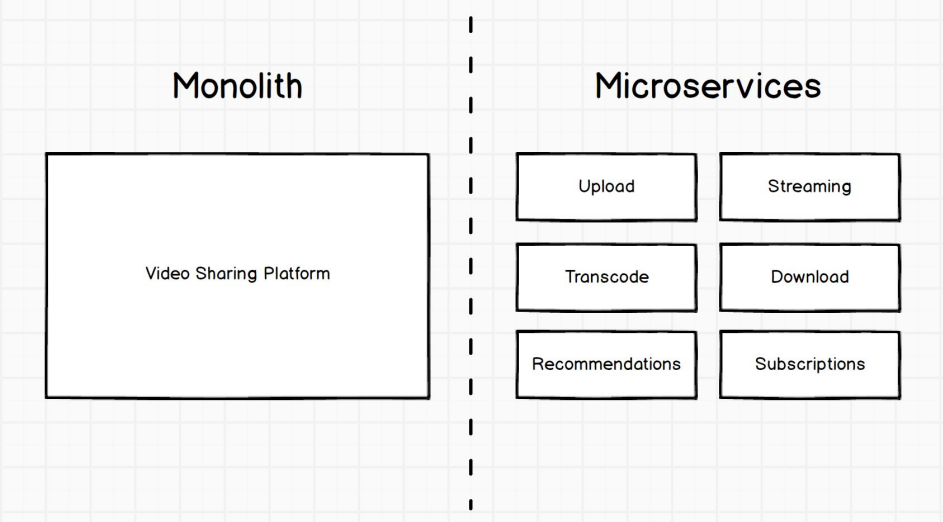


- Независимое развёртывание
- Независимая разработка
- Независимое масштабирование
- Повторное использование

https://www.dwmkerr.com/the-death-of-microservice-madness-in-2018/

~~CORBA~~, ~~DCOM~~, SOAP, REST, gRPC — веб-сервисы в контейнерах

Микросервисы



- Независимое развёртывание
- Независимая разработка
- Независимое масштабирование
- Повторное использование

https://www.dwmkerr.com/the-death-of-microservice-madness-in-2018/

~~CORBA~~, ~~DCOM~~, SOAP, REST, gRPC — веб-сервисы в контейнерах и CI/CD

Микросервисы

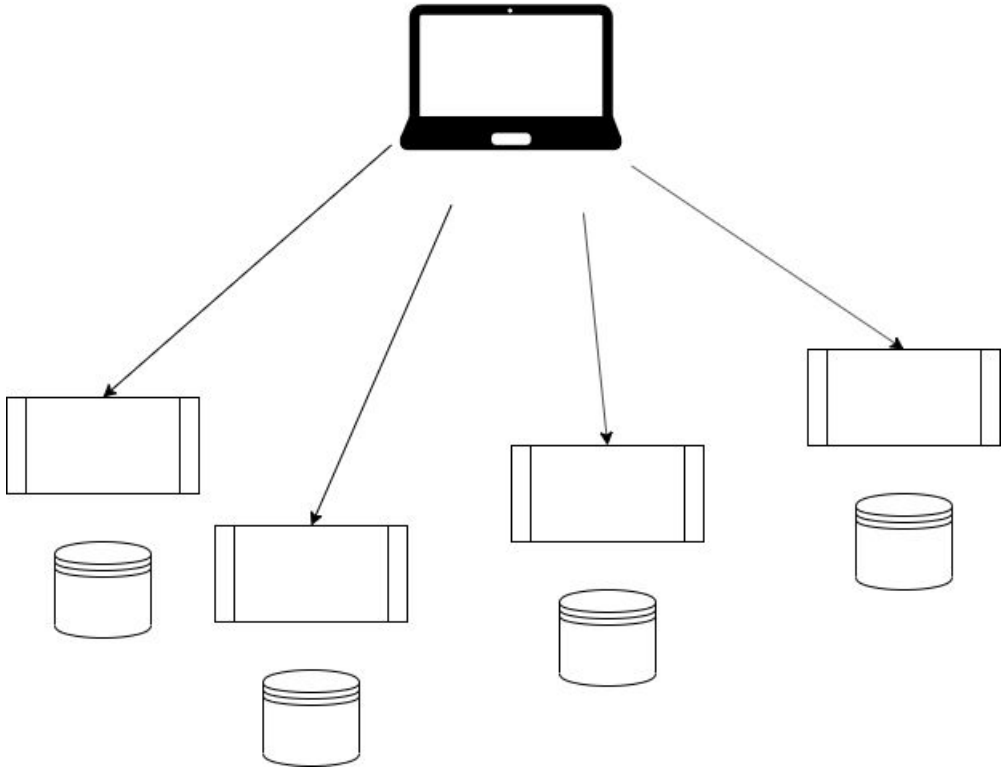
Введение

https://auth0.com/blog/an-introduction-to-microservices-part-1/
https://auth0.com/blog/an-introduction-to-microservices-part-2-API-gateway/
https://auth0.com/blog/an-introduction-to-microservices-part-3-the-service-registry/
https://auth0.com/blog/introduction-to-microservices-part-4-dependencies/

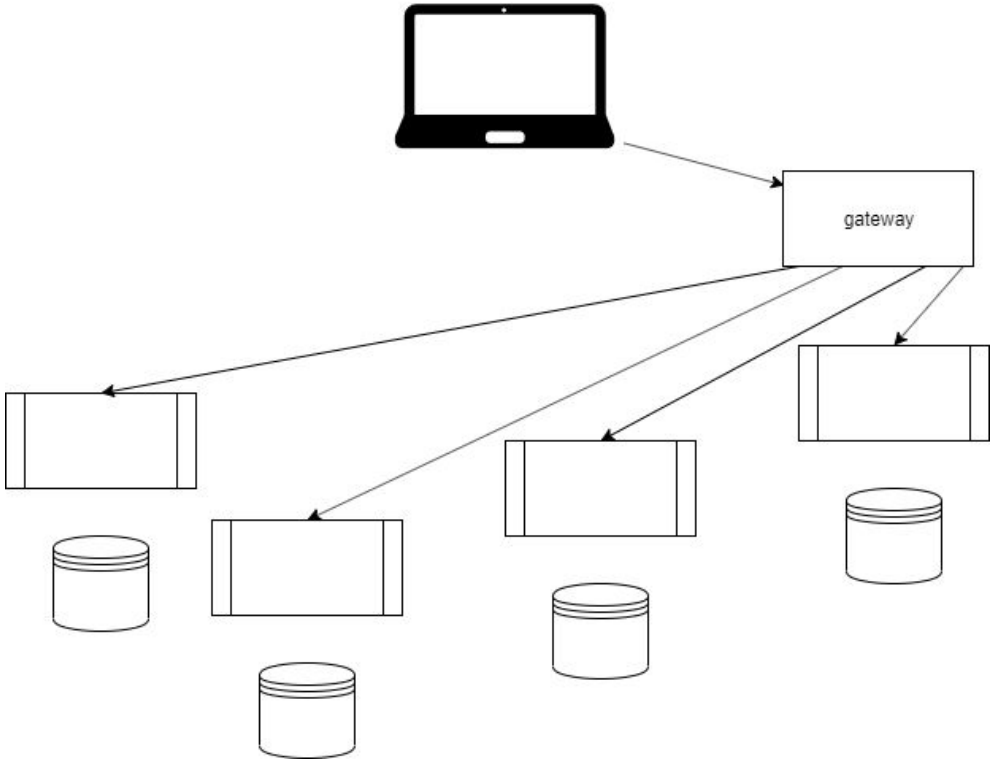
Двенадцатифакторные приложения

https://12factor.net/ru/

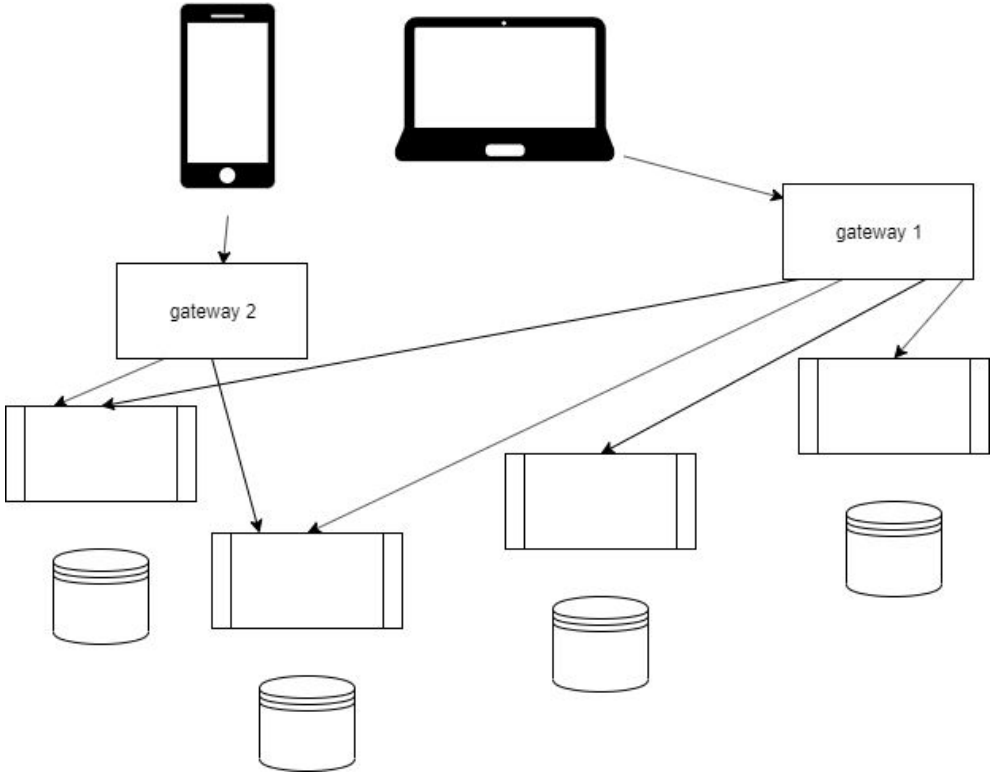
Микросервисы



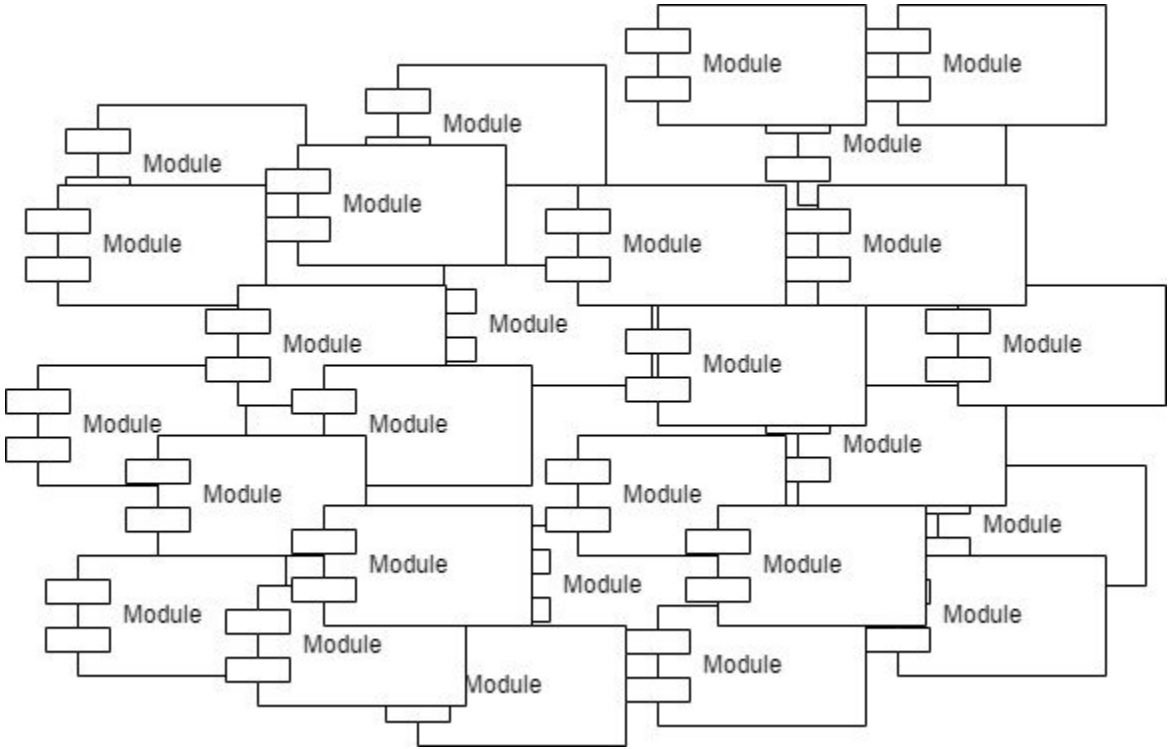
Микросервисы



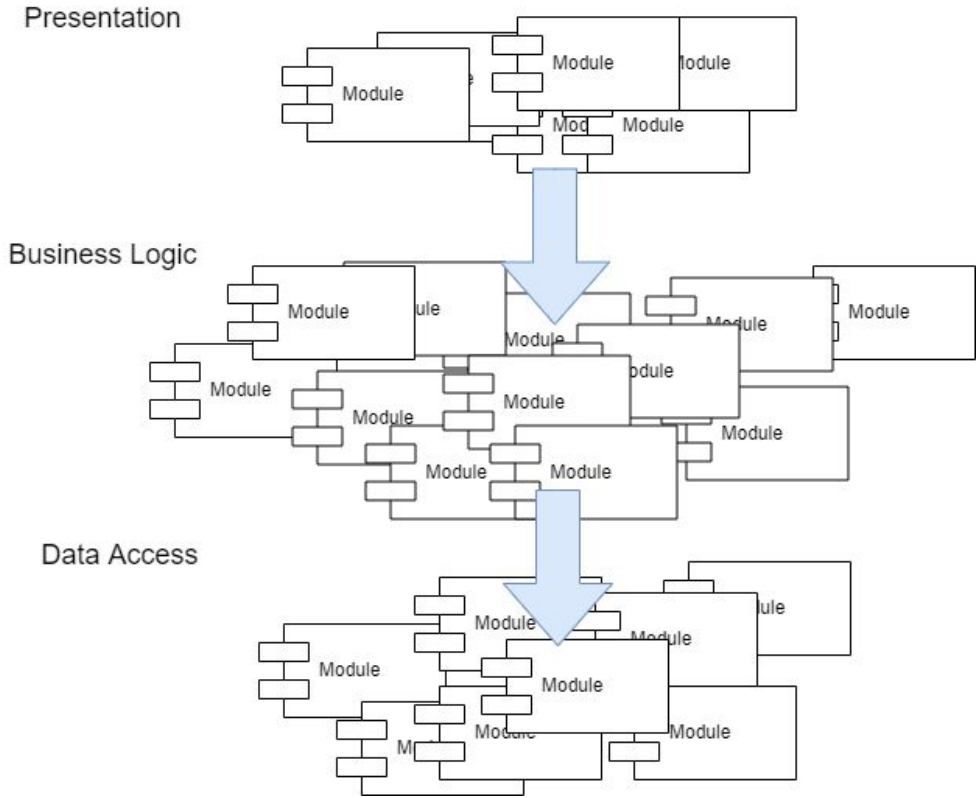
Микросервисы



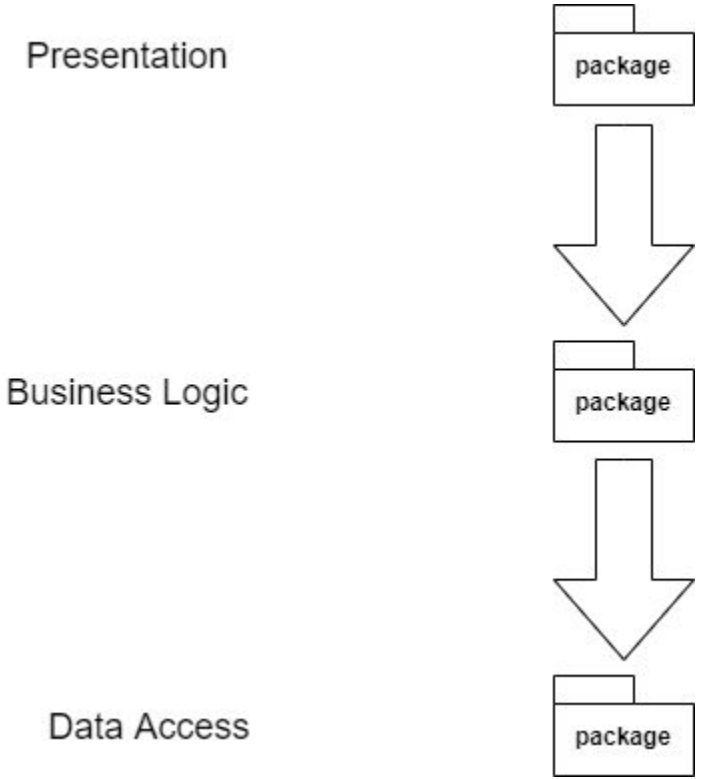
Микросервисы


Module
Module
Module

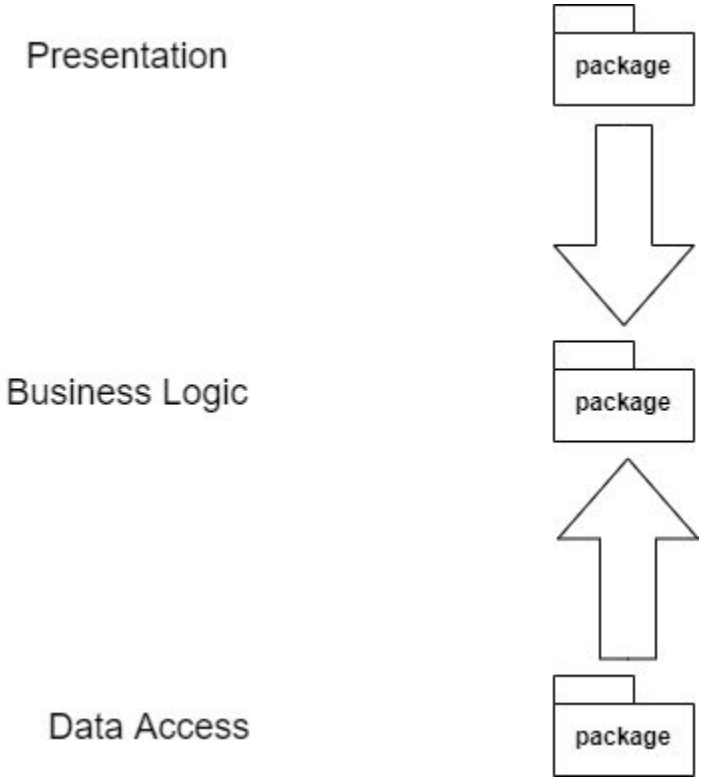
Микросервисы



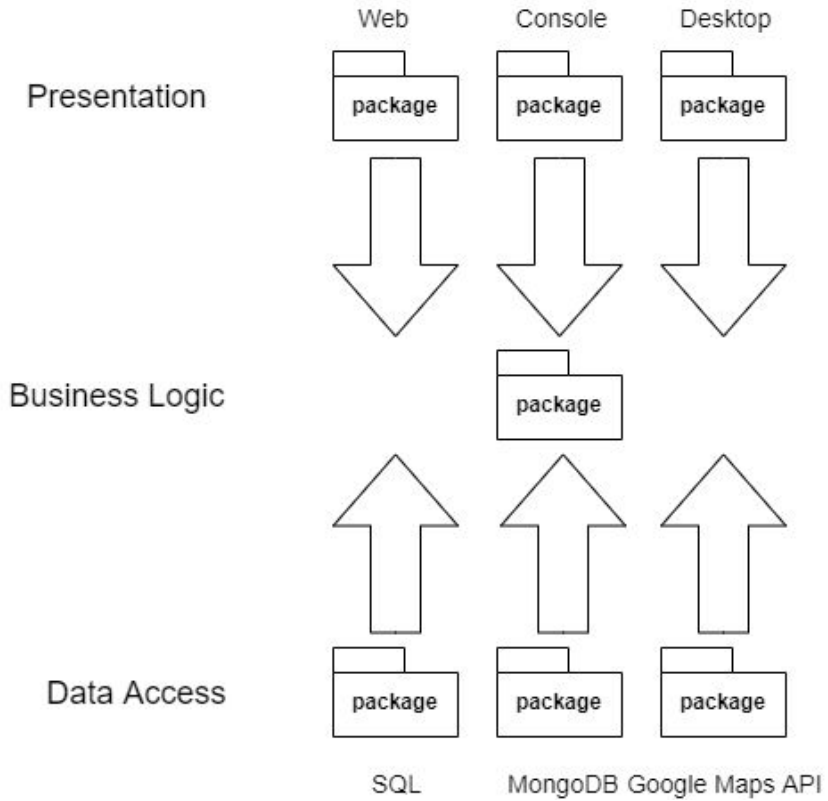
Микросервисы



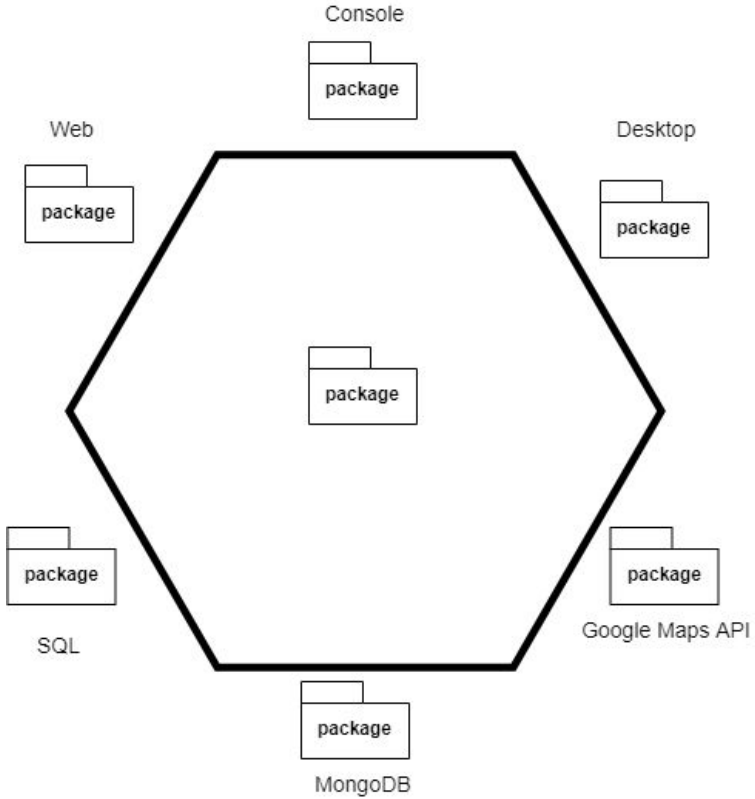
Микросервисы



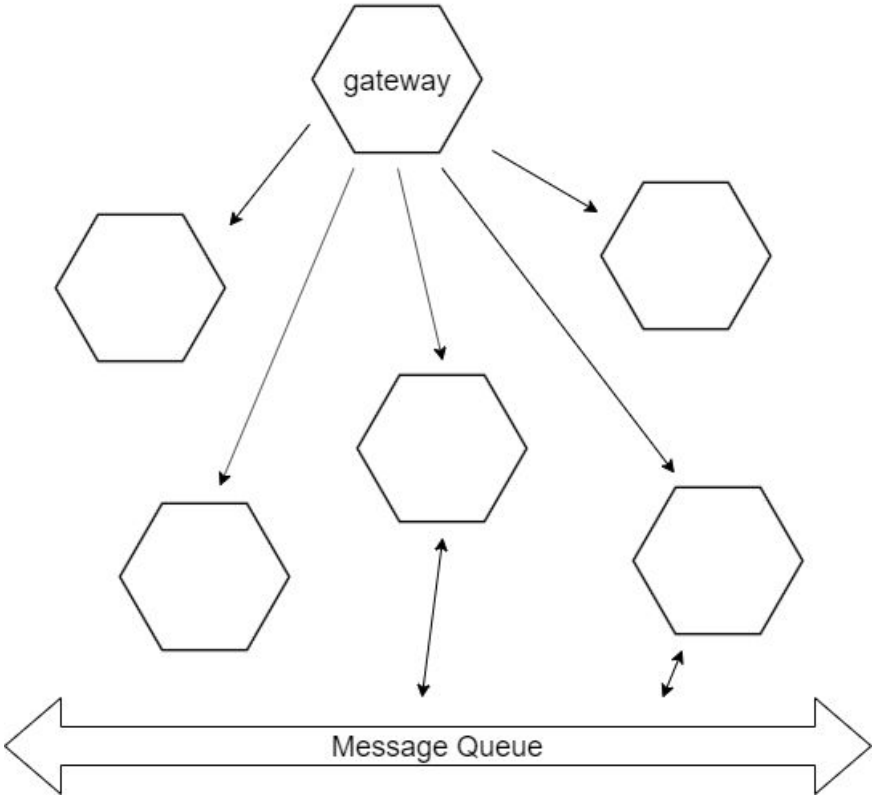
Микросервисы



Микросервисы



Микросервисы



Микросервисы

- ASP.NET Web Forms (2003)

Микросервисы

- ASP.NET Web Forms (2003)
- ASP.NET MVC (2009) — Ruby on Rails

Микросервисы

- ASP.NET Web Forms (2003)
- ASP.NET MVC (2009) — Ruby on Rails
- ASP.NET Kestrel (2014) — libuv

Микросервисы

- ASP.NET Web Forms (2003)
- ASP.NET MVC (2009) — Ruby on Rails
- ASP.NET Kestrel (2014) — libuv
- .NET Core (2016)

https://github.com/libuv/libuv
https://dotnetcore.show/episode-1-a-brief-history-of-net-core/
https://docs.microsoft.com/ru-ru/aspnet/core/tutorials/first-web-api

Микросервисы

```
[Route("api/v1/todo-items")]
[ApiController]
public class TodoItemsController : ControllerBase
{
    private readonly TodoDbContext dbContext;

    [HttpGet(“{id}”)]
    public async Task<TodoItem> GetByIdAsync(int id)
    {
        return await dbContext.TodoItems
                                .SingleAsync(x => x.Id == id);
    }
}
```

Микросервисы

/api/v1/todo-items/123

```
[Route("api/v1/todo-items")]
[ApiController]
public class TodoItemsController : ControllerBase
{
    private readonly TodoDbContext dbContext;

    [HttpGet(“{id}”)]
    public async Task<TodoItem> GetByIdAsync(int id)
    {
        return await dbContext.TodoItems
                              .SingleAsync(x => x.Id == id);
    }
}
```

Микросервисы

```
[Route("api/v1/todo-items")]
[ApiController]
public class TodoItemsController : ControllerBase
{
    private readonly TodoDbContext dbContext;

    [HttpGet(“{id}”)]
    public async Task<TodoItem> GetByIdAsync(int id)
    {
        return await dbContext.TodoItems
                                .SingleAsync(x => x.Id == id);
    }
}
```

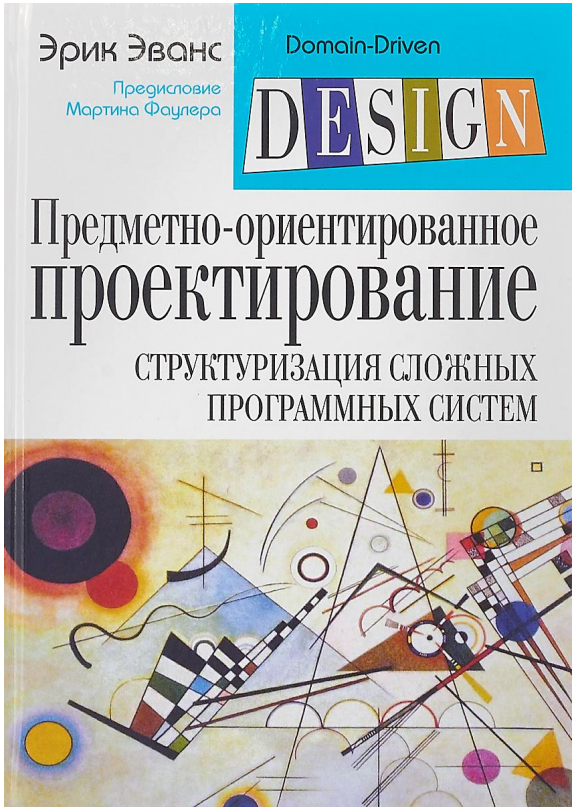
Микросервисы

```
[Route("api/v1/todo-items")]
[ApiController]
public class TodoItemsController : ControllerBase
{
    private readonly TodoDbContext dbContext;

    [HttpGet(“{id}”)]
    public async Task<TodoItem> GetByIdAsync(int id)
    {
        return await dbContext.TodoItems
                                .SingleAsync(x => x.Id == id);
    }
}
```

Микросервисы

- Как делать версионирование API? А если Agile?
- У каждого микросервиса своя база данных?
- Что с транзакциями?

Микросервисы



Микросервисы

Микросервисы

Глава 27

Не нужно на ранних этапах разработки приспосабливать интерфейс REST, потому что высокоуровневая политика не должна зависеть от интерфейса с внешним миром. Также не нужно приспосабливаться под архитектуру микрослужб или SOA. Высокоуровневая политика не должна зависеть от подобных деталей.

Заключение

- Откладываем решение на потом.
- Вначале ограничиваем контексты.
- Продумываем интерфейсы между ними.
- И объекты переноса данных.
- Чтобы превратить контексты в микросервисы:
  - Реализуем интерфейсы как контроллеры на сервере.
  - Добавляем пакет Swashbuckle.
  - Генерируем код доступа на клиенте (паттерн *Заместитель*).
- Во время разработки:
  - Не боимся асинхронного кода.
  - Применяем ORM. Они дают нам LINQ и миграции.

Марк Шевченко http://markshevchenko.pro/ https://t.me/markshevchenko