МИКРОСЕРВИСЫ В БАНКЕ: НА ЧЕМ ИХ ЛУЧШЕ ПИСАТЬ?

- Java/Kotlin, а может Go?

КОЧЕРГИН ИВАН

РСХБ-Интех

Руководитель Центра Собственной Разработки

Свой путь в разработке начинал с Java.
Последние 3 года в качестве основного языка для разработки использую Kotlin.
JPoint-спикер в 2022г.[4][5]

Вызовы современной разработки

Вызовы современной разработки

- Time to Market

Вызовы современной разработки

- Time to Market
- Когнитивная сложность применяемых решений и технологий

Вызовы современной разработки

- Time to Market
- Когнитивная сложность применяемых решений и технологий
- Ресурсоэффективные приложения



Микросервисная архитектура

Микросервисы

Микросервисы

- Выполняет одну бизнес-функцию

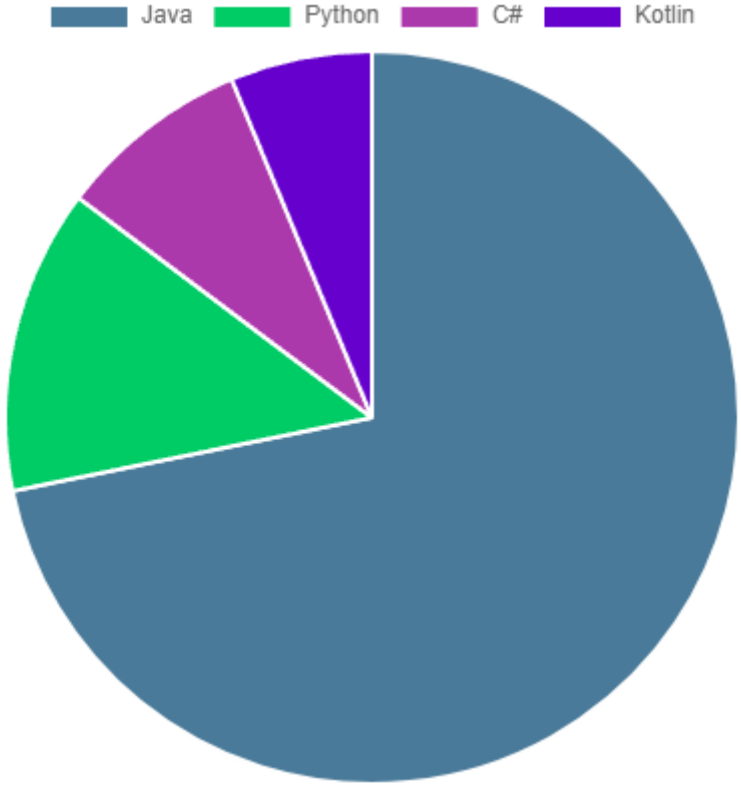
Микросервисы

- Выполняет одну бизнес-функцию
- Слабая связанность с другими сервисами

Микросервисы

- Выполняет одну бизнес-функцию
- Слабая связанность с другими сервисами
- Возможность деплоя и масштабирования независимо от других сервисов

Языки программирования РСХБ



Kotlin vs Go

Критерии для сравнения

Критерии для сравнения

- Null Safety

Критерии для сравнения

- Null Safety
- Расширение функциональности существующих классов/структур без изменения исходного кода

Критерии для сравнения

- Null Safety
- Расширение функциональности существующих классов/структур без изменения исходного кода
- Возможность удобного создания контейнеров для хранения данных

Критерии для сравнения

- Null Safety
- Расширение функциональности существующих классов/структур без изменения исходного кода
- Возможность удобного создания контейнеров для хранения данных
- Неблокирующая многопоточность и IO

Критерии для сравнения

- Null Safety
- Расширение функциональности существующих классов/структур без изменения исходного кода
- Возможность удобного создания контейнеров для хранения данных
- Неблокирующая многопоточность и IO
- Работа с функциями

Критерии для сравнения

- Null Safety
- Расширение функциональности существующих классов/структур без изменения исходного кода
- Возможность удобного создания контейнеров для хранения данных
- Неблокирующая многопоточность и IO
- Работа с функциями
- Синтаксический сахар

Kotlin: Null Safety

```
1 fun main() {
2   var a: String? = "abc" // can be set to null
3   a = null // ok
4   print(a)
5
6   var b: String = "abc" // Regular initialization means no
7   b = null // compilation error
8 }
```

Kotlin: Null Safety

```
1 fun main() {
2   var a: String? = "abc" // can be set to null
3   a = null // ok
4   print(a)
5
6   var b: String = "abc" // Regular initialization means no
7   b = null // compilation error
8 }
```

Kotlin: Null Safety

```
1 fun main() {
2   var a: String? = "abc" // can be set to null
3   a = null // ok
4   print(a)
5
6   var b: String = "abc" // Regular initialization means no
7   b = null // compilation error
8 }
```

Kotlin: Null Safety

```kotlin
fun main() {
  fun strLength(s: String): Int {
    return s.length
  }

  val notNullString = "hello"
  println(strLength(notNullString)) // 5

  var nullable: String? = "hello"
  println(strLength(nullable)) // compilation error

  println(strLength(!!nullable)) // 5
}
```

Kotlin: Null Safety

```kotlin
fun main() {
  fun strLength(s: String): Int {
    return s.length
  }

  val notNullString = "hello"
  println(strLength(notNullString)) // 5

  var nullable: String? = "hello"
  println(strLength(nullable)) // compilation error

  println(strLength(!!nullable)) // 5
}
```

Kotlin: Null Safety

```kotlin
fun main() {
  fun strLength(s: String): Int {
    return s.length
  }

  val notNullString = "hello"
  println(strLength(notNullString)) // 5

  var nullable: String? = "hello"
  println(strLength(nullable)) // compilation error

  println(strLength(!!nullable)) // 5
}
```

Kotlin: Null Safety

```kotlin
fun main() {
  fun strLength(s: String): Int {
    return s.length
  }

  val notNullString = "hello"
  println(strLength(notNullString)) // 5

  var nullable: String? = "hello"
  println(strLength(nullable)) // compilation error

  println(strLength(!!nullable)) // 5
}
```

Kotlin: Null Safety

```kotlin
fun main() {
  fun strLength(s: String): Int {
    return s.length
  }

  val notNullString = "hello"
  println(strLength(notNullString)) // 5

  var nullable: String? = "hello"
  println(strLength(nullable)) // compilation error

  println(strLength(!!nullable)) // 5
}
```

Go: Null Safety

Отсутствует! [1][2]

Go: Null Safety

Отсутствует! [1][2]

```
func main() {
  result, err := doSomething()
  if err != nil {
    os.Exit(1)
  }
}
```

Контейнеры данных: Data classes

```
1 data class UserPassword(val username: String,
2                         val password: String)
3
4 fun main() {
5   val up = UserPassword(username = "John", password = "***"
6   println(up) // UserPassword(username = John, password =
7 }
```

Контейнеры данных: Data classes

```
1 data class UserPassword(val username: String,
2                         val password: String)
3
4 fun main() {
5   val up = UserPassword(username = "John", password = "***"
6   println(up) // UserPassword(username = John, password =
7 }
```

Контейнеры данных: Data classes

```
1 data class UserPassword(val username: String,
2                         val password: String)
3
4 fun main() {
5   val up = UserPassword(username = "John", password = "***"
6   println(up) // UserPassword(username = John, password =
7 }
```

Контейнеры данных: Structs

```go
type UserPassword struct {
  Username string `json:"username"`
  Password string `json:"password"`
}

func main() {
  up := UserPassword {
    Username: "John",
    Password: "***",
  }
  fmt.Println(up) // {John ***}
}
```

Контейнеры данных: Structs

```go
type UserPassword struct {
  Username string `json:"username"`
  Password string `json:"password"`
}

func main() {
  up := UserPassword {
    Username: "John",
    Password: "***",
  }
  fmt.Println(up) // {John ***}
}
```

Контейнеры данных: Structs

```go
type UserPassword struct {
  Username string `json:"username"`
  Password string `json:"password"`
}

func main() {
  up := UserPassword {
    Username: "John",
    Password: "***",
  }
  fmt.Println(up) // {John ***}
}
```

Функции-расширения

```
1  import java.time.LocalDate
2  import java.time.format.DateTimeFormatter
3
4  fun UserPassword.isSecure(): Boolean {
5    /* validate password against some horrible regexp*/
6    return if(this.password.length < 3) {
7      false
8    } else {
9      true
10   }
11 }
12
13 fun LocalDate.toddMMyyyyFormat(): String {
14   return this.format(DateTimeFormatter.ofPattern("dd.MM.yy
15 }
```

Функции-расширения

```
 3
 4 fun UserPassword.isSecure(): Boolean {
 5   /* validate password against some horrible regexp*/
 6   return if(this.password.length < 3) {
 7     false
 8   } else {
 9     true
10   }
11 }
12
13 fun LocalDate.toddMMyyyyFormat(): String {
14   return this.format(DateTimeFormatter.ofPattern("dd.MM.yy
15 }
16
17 fun main() {
18   val up = UserPassword(username = "John", password = "**
```

Функции-расширения

```
 8    } else {
 9      true
10    }
11  }
12
13  fun LocalDate.toddMMyyyyFormat(): String {
14    return this.format(DateTimeFormatter.ofPattern("dd.MM.yy
15  }
16
17  fun main() {
18    val up = UserPassword(username = "John", password = "***
19    println(up.isSecure()) // true
20
21    println(LocalDate.now().toddMMyyyyFormat()) // 06.07.202
22  }
```

Функции-расширения

```
type UserPassword struct {
  Username string `json:"username"`
  Password string `json:"password"`
}

func (up *UserPassword) isSecure() bool {
  /* validate password against some horrible regexp*/
  var isSecure bool
  if len(up.Password) < 3 {
    isSecure = false
  } else {
    isSecure = true
  }
  return isSecure
}
```

Функции-расширения

```
 5
 6 func (up *UserPassword) isSecure() bool {
 7   /* validate password against some horrible regexp*/
 8   var isSecure bool
 9   if len(up.Password) < 3 {
10     isSecure = false
11   } else {
12     isSecure = true
13   }
14   return isSecure
15 }
16
17 func main() {
18   up := UserPassword {
19     Username: "John",
20     Password: "***",
```

Coroutines[4][5]

```
fun main() = runBlocking {
  val ch = Channel<String>()
  launch {
    getFromInternet(ch)
  }
  val resBody = ch.receive()
  println(resBody) // response body
}

suspend fun getFromInternet(ch: Channel<String>) {
  val request = HttpRequest.newBuilder()
    .GET()
    .uri(URI.create("https://example.org/"))
    .build()
```

Goroutines

```
func main() {
  ch := make(chan string)
  go getFromInternet(ch)
  resBody := <- ch
  fmt.Println(resBody) // response body
}

func getFromInternet(ch chan string) {
  res, err := http.Get("https://example.org/")
  if err != nil {
    os.Exit(1)
  }
  resBody, err := ioutil.ReadAll(res.Body)
  if err != nil {
    os.Exit(1)
```

Kotlin: First-Class Functions

```kotlin
fun main() {
  val john = UserPassword(username = "John", password = "
  val ivan = UserPassword(username = "Иван", password = "

  auth(john, ::greetingEng) // Hello, John
  auth(ivan, {name -> "Привет, $name"}) // Привет, Иван
}

fun auth(up: UserPassword, greeter: (name: String) -> Str
  println(greeter(up.username))
}

fun greetingEng(name: String): String {
  return "Hello, $name"
}
```

Kotlin: First-Class Functions

```kotlin
fun main() {
  val john = UserPassword(username = "John", password = "
  val ivan = UserPassword(username = "Иван", password = "

  auth(john, ::greetingEng) // Hello, John
  auth(ivan, {name -> "Привет, $name"}) // Привет, Иван
}

fun auth(up: UserPassword, greeter: (name: String) -> Str
  println(greeter(up.username))
}

fun greetingEng(name: String): String {
  return "Hello, $name"
}
```

Kotlin: First-Class Functions

```kotlin
fun main() {
  val john = UserPassword(username = "John", password = "
  val ivan = UserPassword(username = "Иван", password = "

  auth(john, ::greetingEng) // Hello, John
  auth(ivan, {name -> "Привет, $name"}) // Привет, Иван
}

fun auth(up: UserPassword, greeter: (name: String) -> Str
  println(greeter(up.username))
}

fun greetingEng(name: String): String {
  return "Hello, $name"
}
```

Go: First-Class Functions

```
1  func main() {
2    john := UserPassword{"John", "***"}
3    ivan := UserPassword{"Иван", "***"}
4
5    auth(john, greetingEng) // Hello, John
6    auth(ivan, func(name string) string) {
7      return "Привет, " + name
8    }) // Привет, Иван
9  }
10
11 fun auth(up *UserPassword, greeter func(name string) strin
12   fmt.Println(greeter(up.Username))
13 }
14
15 fun greetingEng(name string) string {
```

Go: First-Class Functions

```
  ivan := UserPassword{"Иван", "***"}

  auth(john, greetingEng) // Hello, John
  auth(ivan, func(name string) string) {
    return "Привет, " + name
  }) // Привет, Иван
}

fun auth(up *UserPassword, greeter func(name string) strin
  fmt.Println(greeter(up.Username))
}

fun greetingEng(name string) string {
  return "Hello, " + name
}
```

Go: First-Class Functions

```
1  func main() {
2    john := UserPassword{"John", "***"}
3    ivan := UserPassword{"Иван", "***"}
4
5    auth(john, greetingEng) // Hello, John
6    auth(ivan, func(name string) string) {
7      return "Привет, " + name
8    }) // Привет, Иван
9  }
10
11 fun auth(up *UserPassword, greeter func(name string) strin
12   fmt.Println(greeter(up.Username))
13 }
14
15 fun greetingEng(name string) string {
```

Результаты сравнения

Функциональность	Kotlin	Go
Null safety	✓	✗
Extension Functions	✓	~
Data Containers	✓	~
Syntax Sugar	✓ [10]	~
First-Class Functions	✓	✓
Non-blocking concurrency	✓	✓

Микросервисы на JVM

Микросервисы на JVM

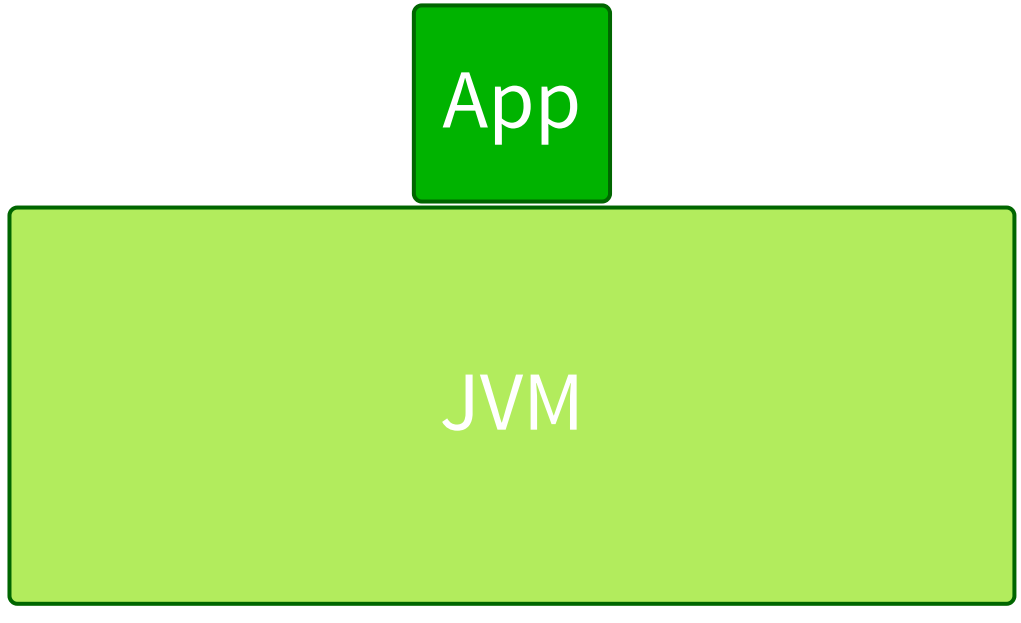
- JVM при запуске использует минимум 70МБ

Микросервисы на JVM

- JVM при запуске использует минимум 70МБ
- Для задач обслуживания приложения JVM необходимо от 10 до 25% памяти приложения[8]

Микросервисы на JVM

- JVM при запуске использует минимум 70МБ
- Для задач обслуживания приложения JVM необходимо от 10 до 25% памяти приложения[8]



Настолько ли Go эффективен?

Приложение, которое удовлетворяет требованиям App.Farm к наличию OpenAPI и healthcheck-endpoint

	JVM	Go
Версия языка	Java 17 Kotlin 1.7	1.20
Версия фреймворка	Spring 3.1	Echo 4

Сравнение объема потребляемых ресурсов

Критерий	JVM	Go
Размер артефакта	28МБ	23МБ
Размер потребляемой памяти	168МБ	4МБ
Время запуска	3.5сек	<1сек

Что может предложить Java?

Что может предложить Java?

GraalVM

Да, но нет!

Да, но нет!

- Требуются новейшие версии фреймворков

Да, но нет!

- Требуются новейшие версии фреймворков
- Время сборки значительно увеличится

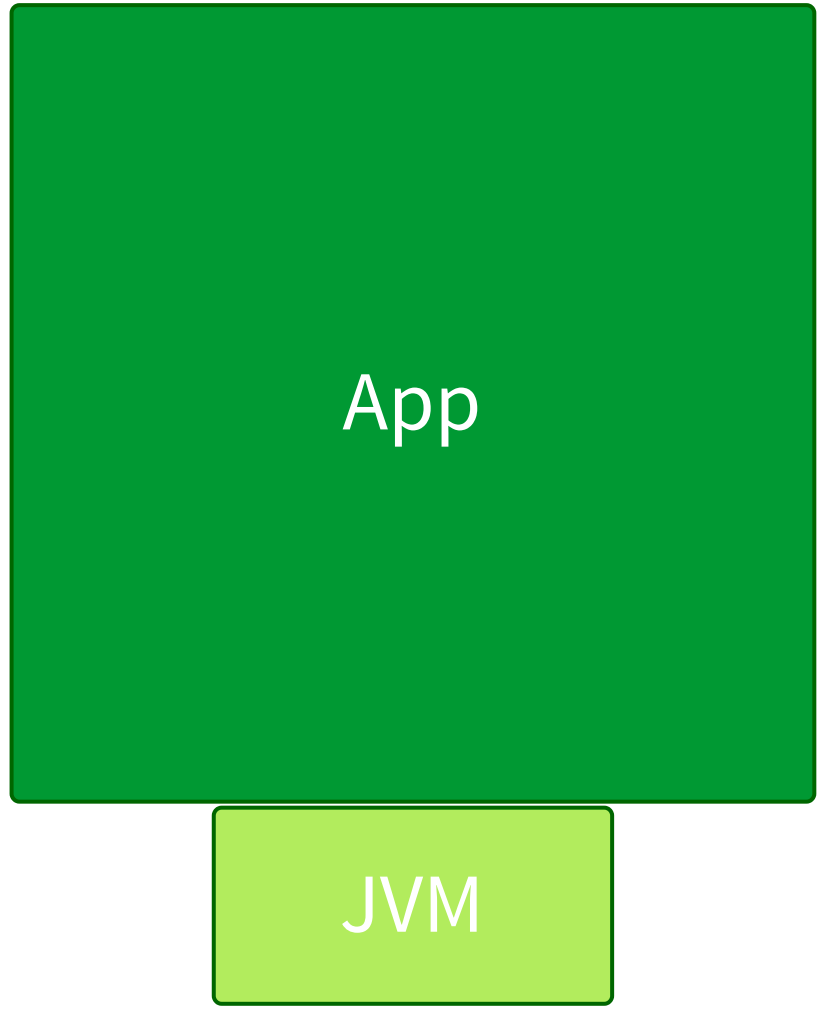
Да, но нет!

- Требуются новейшие версии фреймворков
- Время сборки значительно увеличится
- Увеличится размер артефакта

Да, но нет!

- Требуются новейшие версии фреймворков
- Время сборки значительно увеличится
- Увеличится размер артефакта
- Проблемы со сторонними библиотеками

Паттерн Strongman [7]



Паттерн Swiss Knife

Kotlin: Итоги

Kotlin: Итоги

- Полная совместимость с Java позволяет использовать существующие библиотеки и собственные наработки

Kotlin: Итоги

- Полная совместимость с Java позволяет использовать существующие библиотеки и собственные наработки
- Лаконичность синтаксиса позволяет разработчикам писать код быстрее

Kotlin: Итоги

- Полная совместимость с Java позволяет использовать существующие библиотеки и собственные наработки
- Лаконичность синтаксиса позволяет разработчикам писать код быстрее
- Неблокирующая многопоточность: разработка в привычной императивной парадигме

Kotlin: Итоги

- Полная совместимость с Java позволяет использовать существующие библиотеки и собственные наработки
- Лаконичность синтаксиса позволяет разработчикам писать код быстрее
- Неблокирующая многопоточность: разработка в привычной императивной парадигме
- Количество кода меньше на 20% по сравнению с Java[9]

Kotlin: Итоги

- Полная совместимость с Java позволяет использовать существующие библиотеки и собственные наработки
- Лаконичность синтаксиса позволяет разработчикам писать код быстрее
- Неблокирующая многопоточность: разработка в привычной императивной парадигме
- Количество кода меньше на 20% по сравнению с Java[9]
- Возможность использования со старыми версиями Java

Go: **Итоги**

Go: **Итоги**

- Маленький Application Footprint

Go: **Итоги**

- Маленький Application Footprint
- Встроенная поддержка многопоточности

Go: **Итоги**

- Маленький Application Footprint
- Встроенная поддержка многопоточности
- Неблокирующая многопоточность: разработка в привычной императивной парадигме

Go: **Итоги**

- Маленький Application Footprint
- Встроенная поддержка многопоточности
- Неблокирующая многопоточность: разработка в привычной императивной парадигме
- Небольшая экосистема по сравнению с Java

Go: Итоги

- Маленький Application Footprint
- Встроенная поддержка многопоточности
- Неблокирующая многопоточность: разработка в привычной императивной парадигме
- Небольшая экосистема по сравнению с Java
- Отсутствие поддержки старых технологий и протоколов

Go: **Итоги**

- Маленький Application Footprint
- Встроенная поддержка многопоточности
- Неблокирующая многопоточность: разработка в привычной императивной парадигме
- Небольшая экосистема по сравнению с Java
- Отсутствие поддержки старых технологий и протоколов
- Меньше возможностей для управления GC

И что дальше??

И что дальше??

Начинать использовать Kotlin

А если хочется в Go??

А если хочется в Go??

- Начинать использовать Kotlin

А если хочется в Go??

- Начинать использовать Kotlin
- Дождаться релиза сервиса-пионера на Go

А если хочется в Go??

- Начинать использовать Kotlin
- Дождаться релиза сервиса-пионера на Go
- Начинать использовать шаблон Go-приложения для нового функционала

Ссылки

1. Go null-safety↑
2. Safe Go↑
3. Reactive or Coroutines↑
4. JPoint 2022: Kotlin, Coroutines, Micronaut p.I↑
5. JPoint 2022: Kotlin, Coroutines, Micronaut p.II↑
6. Coroutines and Goroutines↑
7. Java as fast as Go↑
8. JVM resource consumption↑
9. Reducing boilerplate with Kotlin↑
10. Kotlin Syntax Constructions↑

СПАСИБО ЗА ВНИМАНИЕ!

Кочергин Иван
ivan.s.kochergin@gmail.com