OZON tech

Есть ли ORM круче SwiftData

Сергей Балалаев, руководитель отдела разработки мобильного приложения «ПВЗ».
Компания Озон

Сергей Балалаев

Руководитель отдела разработки мобильного приложения «ПВЗ»

- **15 лет** опыта в мобильной разработке
- **10 лет** руковожу людьми
- **9 лет** преподаю на курсах iOS

Развитие темы

Затаскиваем на iOS серверный ORM под SQLite
(прошло 23 марта) о том, почему не хотим CoreData/SwiftData

Выбираем хранилку данных для iOS проекта
(прошло 12 апреля) о том как хорош Fluent

Есть ли ORM круче SwiftData
(1 июня) Сравниваем Fluent по производительности со SwiftData, Realm

AGENDA

01 **История Apple**
– CoreData
– SwiftData

02 **Есть ли альтернативы**
– из Backend?
– из прошлого?

03 **Альтернативы**
– Портирование
– Сущности и миграция
– Запросы к БД

04 **Исследование и сравнение**
– Время выполнения одного запроса
– Сравнение между собой
– Заключение

01

История одного фреймворка Apple для СУБД

CoreData

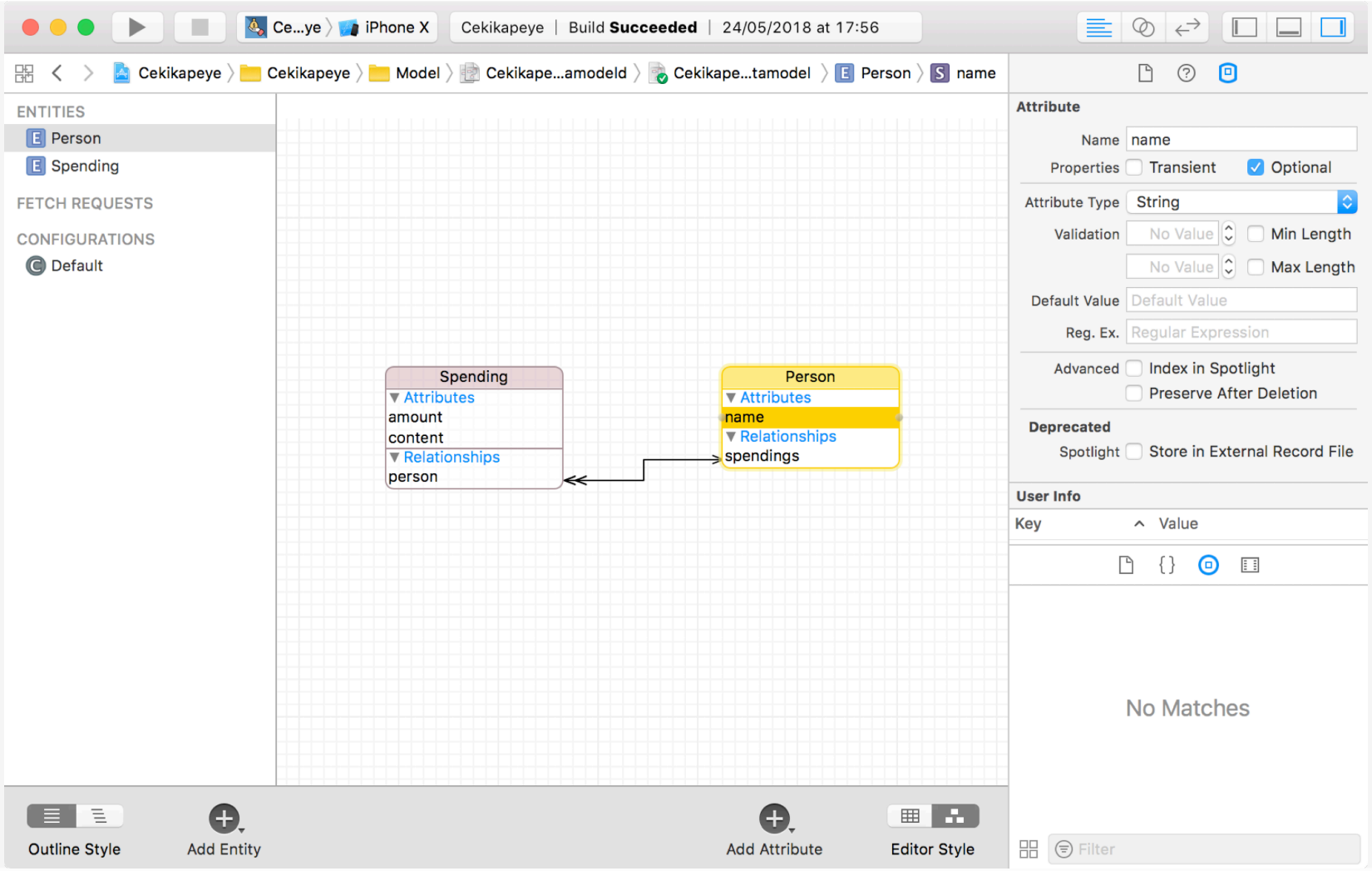
2011

SwiftData

2023

CoreData

Редактирование структуры БД



```swift
import CoreData

class DataController: NSObject {
    var managedObjectContext: NSManagedObjectContext
    init(completionClosure: @escaping () -> ()) {
        //This resource is the same name as your xcdatamodeldd contained in your project
        guard let modelURL = Bundle.main.url(forResource: "DataModel", withExtension:"momd") else {
            fatalError("Error loading model from bundle")
        }
        // The managed object model for the application.
        // It is a fatal error for the application not to be able to find and load its model.
        guard let mom = NSManagedObjectModel(contentsOf: modelURL) else {
            fatalError("Error initializing mom from: \(modelURL)")
        }
        let psc = NSPersistentStoreCoordinator(managedObjectModel: mom)
        managedObjectContext = NSManagedObjectContext(concurrencyType:
                                                      NSManagedObjectContextConcurrencyType.mainQueueConcurrencyType)
        managedObjectContext.persistentStoreCoordinator = psc
        let queue = DispatchQueue.global(qos: DispatchQoS.QoSClass.background)
        queue.async {
            guard let docURL = FileManager.default.urls(for: .documentDirectory, in: .userDomainMask).last else {
                fatalError("Unable to resolve document directory")
            }
            let storeURL = docURL.appendingPathComponent("DataModel.sqlite")
            do {
                try psc.addPersistentStore(ofType: NSSQLiteStoreType, configurationName: nil, at: storeURL, options: nil)
                DispatchQueue.main.sync(execute: completionClosure)
            } catch {
                fatalError("Error migrating store: \(error)")
            }
        }
    }
}
```

CoreData: Запросы к БД

```
NSPredicate(format: "name == %@", "Python")
```

```
struct CoreDataRequest {

    @FetchRequest(
        sortDescriptors: [SortDescriptor(\.name)],
        predicate: NSPredicate(format: "name == %@", "Python")
    )
    var languages: FetchedResults<ProgrammingLanguage>

}
```

```
@objc
class ProgrammingLanguage : NSManagedObject {
    var name: String = ""
}
```

Проблемы CoreData

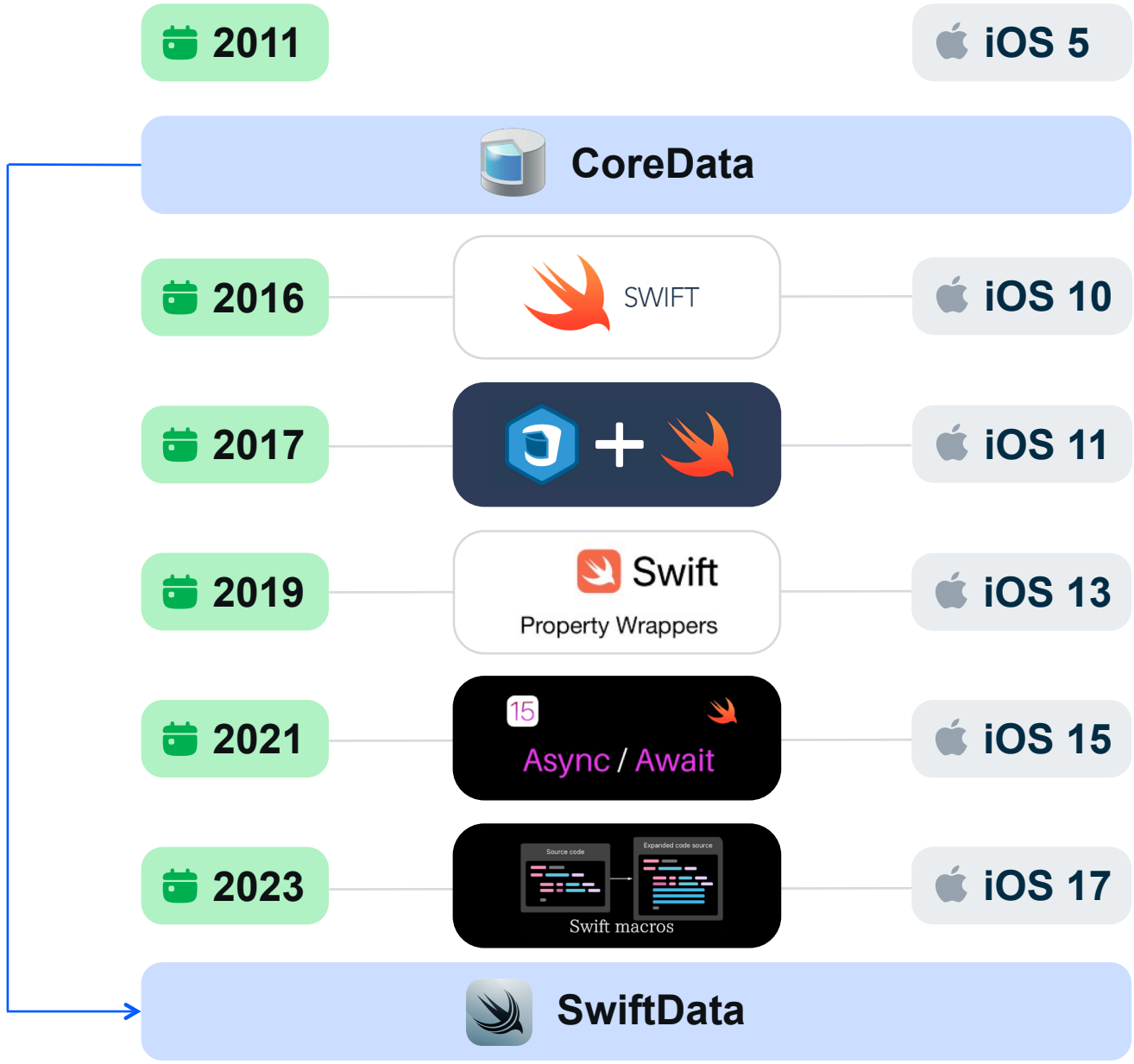
01 **Описание модели**
Отвязано от кода самой модели

02 **Сложная работа с потоками**
Через GCD

03 **Запросы через Predicate**
Текст интерпретируется в Runtime

04 **Миграция схем**
Требуется описание обеих версий

SwiftData


2011
iOS 5
CoreData
2016
SWIFT
iOS 10
2017
iOS 11
2019
Swift
Property Wrappers
iOS 13
2021
15
Async / Await
iOS 15
2023
Source code
Expanded code source
Swift macros
iOS 17
SwiftData

SwiftData решает проблемы CoreData

01 **Описание модели**
- Теперь через код

02 **Многопоточность**
- Использует Swift Concurrency

03 **Запросы через Predicate**
- Теперь через код в макросах

04 **Миграция схем**
- Автомиграция, приложения перестали падать

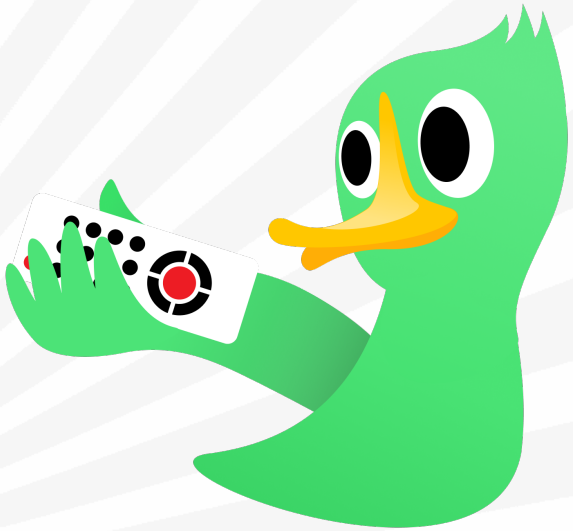
05 **Поддержка с iOS 17**
- Аммм…

Переходим на **SwiftData**?

Вопрос

02

Где искать альтернативы?



Backend на Swift

VAPOR

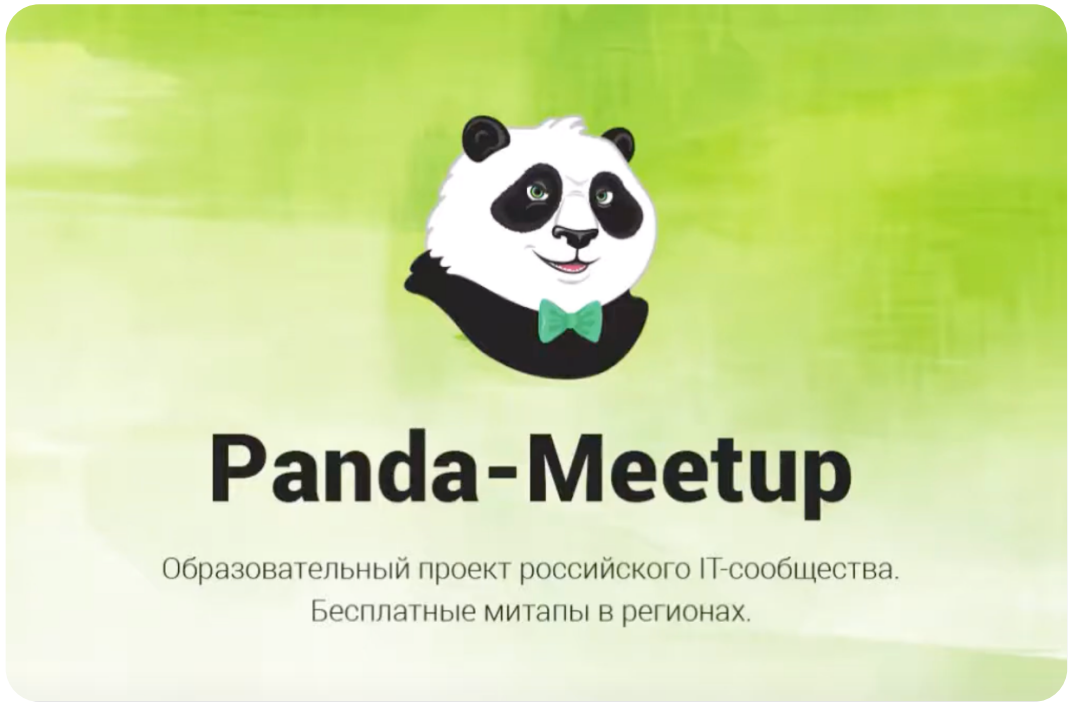
KITURA

SWIFT EXPRESS

Vapor 3 (2017)

Серверсайд на Swift и применение технологии Vapor в продакшне

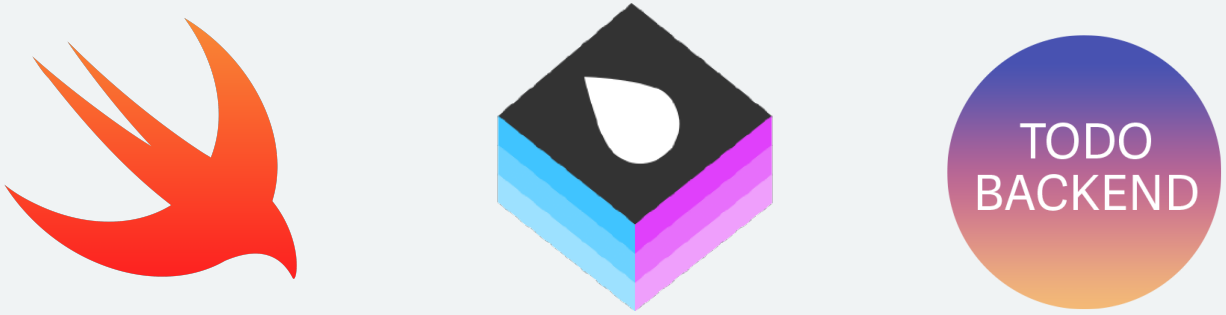
Vapor 3



https://www.youtube.com/watch?v=xD5E37pSpKI

Развитие

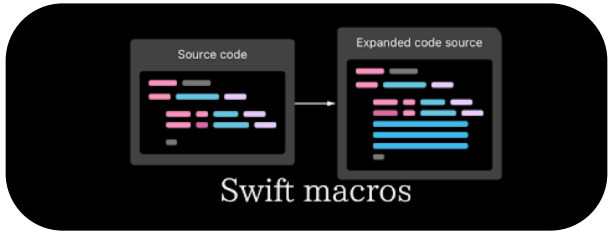
Vapor 4



Implement Todo-Backend with Vapor 4

2019

2021

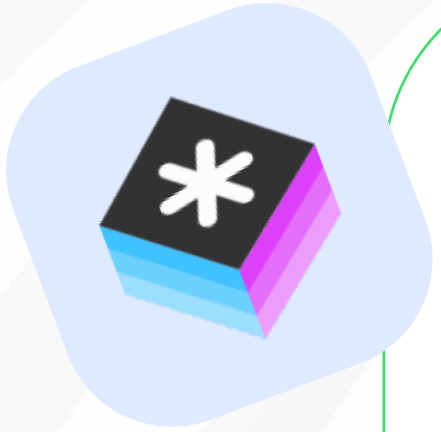


2023

Есть ли свой СУБД-фреймворк ?

Fluent

Vapor 4



https://github.com/vapor/fluent

03 Альтернативы

Портируем Fluent

FluentSQLite

Swift Package Manager

```
let package = Package(
    name: "fluent-sqlite-driver",
    platforms: [
        .macOS(.v10_15),
        .iOS(.v13),
        .watchOS(.v6),
        .tvOS(.v13),
    ],
```

FluentSQLite

https://github.com/vapor/fluent-sqlite-driver

Настраиваем Fluent на работу с Sqlite

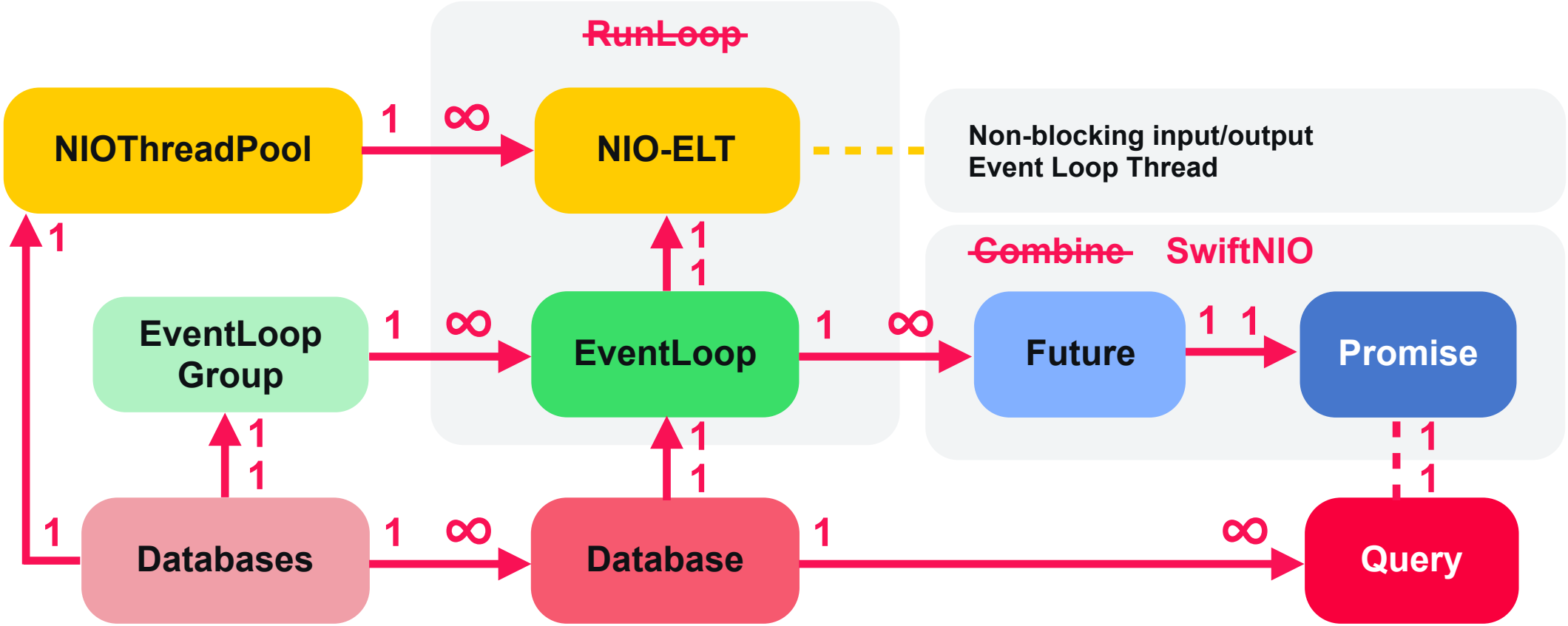
```
private static var threadsCount: Int {
    System.coreCount
}

private var group = MultiThreadedEventLoopGroup(numberOfThreads: threadsCount)
private var pool = NIOThreadPool(numberOfThreads: threadsCount)
private var log = Logger(label: "DB")
private var migrations: Migrations = Migrations()
```

```
let path = getDocumentsDirectory().appendingPathComponent("sqlite.db")

let configuration = SQLiteConfiguration(storage:.file(path: path.absoluteString))

pool.start()

dbs = Databases(threadPool: pool, on: group)
dbs.use(.sqlite(configuration), as: .sqlite)
```

```
public var db: Database {
    dbs.database(logger: log, on: dbs.eventLoopGroup.next())!
}
```

EventLoopGroup → Группа циклов для запуска Future на выбранном потоке

из **SwiftNIO** → Swift Non-blocking input/output



Уже есть альтернатива

vs

vs

Сущности

Fluent-сущность

```swift
final class TodoEntity : Model {

    static var schema: String = "Todo"

    @ID(key: .id)
    var id: UUID?

    @Field(key: "name")
    var name: String

    @OptionalField(key: "comments")
    var comments: String?

    @OptionalField(key: "date")
    var date: Date?

    @OptionalParent(key: "group_id")
    var group: TodoGroupEntity?

    @OptionalField(key: "priority")
    var priority: Int?

    var count: Int = 0
}
```

1 ∞

```swift
final class TodoGroupEntity : Model {

    static var schema: String = "TodoGroup"

    @ID(key: .id)
    var id: UUID?

    @Field(key: "name")
    var name: String

    @Children(for: \.$group)
    var todos: [TodoEntity]
}
```

Сравнение с Realm и SwiftData

```
final class TodoEntity : Object {

    @Persisted (primaryKey: true)
    var id: UUID

    @Persisted(indexed: true)
    var name: String

    @Persisted
    var comments: String?

    @Persisted
    var date: Date?

    @Persisted
    var group: TodoGroupEntity?

    @Persisted(indexed: true)
    var priority: Int?

    var count: Int = 0

}
```

```
@Model
final class TodoEntity {

    @Attribute(.unique)
    var id: UUID?

    @Attribute(.spotlight)
    var name: String

    var comments: String?

    var date: Date?

    @Attribute(.spotlight)
    var group: TodoGroupEntity?

    @Attribute(.spotlight)
    var priority: Int?

    @Transient
    var count: Int = 0

}
```

References Realm и SwiftData

```
final class TodoGroupEntity : Object {

    @Persisted(primaryKey: true)
    var id: UUID

    @Persisted
    var name: String

    @Persisted(originProperty: "group")
    var todos: LinkingObjects<TodoEntity>

}
```

```
@Model
final class TodoGroupEntity {

    @Attribute(.unique)
    var id: UUID?

    var name: String

    @Relationship(
        deleteRule: .cascade,
        inverse: \TodoEntity.group
    )
    var todos: [TodoEntity]

}
```

Миграция

Fluent: если сущности такой в БД нет

```swift
struct CreateTodoEntity: AsyncMigration {

    func prepare(on database: Database)
    async throws {
        try await database
            .schema(TodoEntity.schema)
            .ignoreExisting()
            .id()
            .field("name", .string, .required)
            .field("comments", .string)
            .create()
    }

    // Optionally reverts the changes
    // made in the prepare method.
    func revert(on database: Database)
    async throws {
        try await database
            .schema(TodoEntity.schema)
            .delete()
    }
}
```

```swift
struct DateTodoEntity: AsyncMigration {

    func prepare(on database: Database)
    async throws
    {
        try await database
            .schema(TodoEntity.schema)
            .ignoreExisting()
            .field("date", .datetime)
            .update()
    }

    func revert(on database: Database)
    async throws
    {
        try await database
            .schema(TodoEntity.schema)
            .deleteField("date")
            .update()
    }

}
```

Fluent: миграция при старте работы с БД

```
public func start() async {
    let path = getDocumentsDirectory()
        .appendingPathComponent("sqlite.db")
        .absoluteString
    let configuration: SQLiteConfiguration =
        .init(storage:.file(path: path))
    pool.start()
    dbs = Databases(threadPool: pool, on: group)
    dbs?.use(.sqlite(configuration), as: .sqlite)

    migrations.add(CreateTodoEntity())
    migrations.add(DateTodoEntity())
    migrations.add(CreateTodoGroupEntity())
    migrations.add(GroupTodoEntity())

    do {
        try await autoMigrate()
    } catch let error {
        print("Error: \(error)")
    }

}
```

```
private var migrator: Migrator? {
    guard let dbs = self.dbs else {
        return nil
    }
    return Migrator(
        databases: dbs,
        migrations: self.migrations,
        logger: self.log,
        on: self.group.any(),
        migrationLogLevel: .trace
    )
}

private func autoMigrate() async throws {
    guard let migrator = self.migrator
    else { return }

    try await
    migrator.setupIfNeeded().flatMap {
        migrator.prepareBatch()
    }.get()
}
```

Запросы на запись

Fluent: создание или редактирование Todo

```
func editOrAdd() {
    Task {
        do {
            var todo = TodoEntity(id: nil)

            If let findTodo = try await TodoEntity
                .find(editedTodo?.id, on: DatabaseManager.shared.db)
                .get()
            {
                todo = findTodo
            }
            todo.name = name
            todo.comments = comments
            todo.$group.id = group.id
            try await todo.save(on: DatabaseManager.shared.db)
            finish()
        } catch let error{
            print("Error: \(error)")
        }
    }
}
```

Сравнение с Realm и SwiftData

Fluent:

```
try await todo.save(on: DatabaseManager.shared.db)
```

Realm:

```
try databaseManager.realm.write{
    databaseManager.realm.add(todo)
}
```

SwiftData:

```
let modelContext = ModelContext(DatabaseManager.shared.container)
modelContext.insert(todo)
try modelContext.save()
```

Запросы на чтение

Fluent : получение сущностей

```
Task { @MainActor in
    var groups = try await DatabaseManager.shared.db
        .query(TodoGroupEntity.self)
        .filter(\.$name ~~ text)
        .with(\.$todos)
        .all()
        .map{ item in
            TodoGroup(name: item.name, todos: item.todos.map{$0.dao})
        }

    let todos = try await TodoEntity
        .query(on: DatabaseManager.shared.db)
        .filter(\.$name ~~ text)
        .filter(\TodoEntity.$group.$id, .equal, .none)
        .sort(\.$date, .ascending)
        .all()
        .map { $0.dao }

    groups.append(TodoGroup(id: nil, name: " ", todos: todos))
    self.groups = groups
}
```

Сравнение с Realm и SwiftData

```
var query = databaseManager.realm
    .objects(TodoEntity.self)

if !searchText.isEmpty {
    query = query
        .where {
            $0.name.contains(searchText)
        }
}

query
    .sorted(by: \.name, ascending: true)
   .map { $0.dao }
```

```
let newContext = ModelContext(DatabaseManager.shared.container)

let todosPredicate = #Predicate<TodoEntity>{ entity in
    if let priority = entity.priority {
        return priority >= startPriority
            && priority <= stopPriority
    } else {
        return false
    }
}

let todosDescriptor = FetchDescriptor<TodoEntity>(
    predicate: todosPredicate,
    sortBy: [SortDescriptor(\.priority)]
)

let todos = try newContext
    .fetch(todosDescriptor)
    .map { $0.dao }
```

Fluent решает те же проблемы с 2016

01 **Описание модели**
- Только код

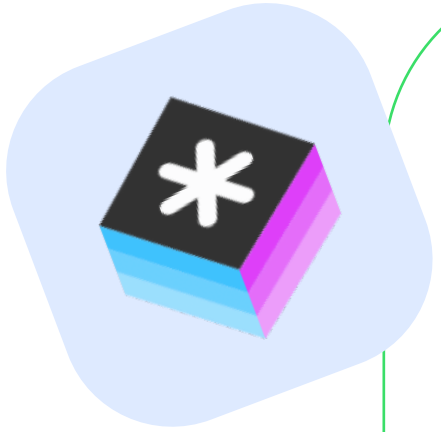
02 **Многопоточность**
- И Swift Concurrency и Future.Promise

03 **Запросы через Predicate**
- Чисто функциональный стиль

04 **Миграция схем**
- Автомиграция, тонкая настройка, чистый мапинг

05 **Поддержка с iOS 13**
- Ого! Вот это можно тащить в проект сейчас

Documentation



https://docs.vapor.codes/fluent/overview/

04 Исследование и Сравнение

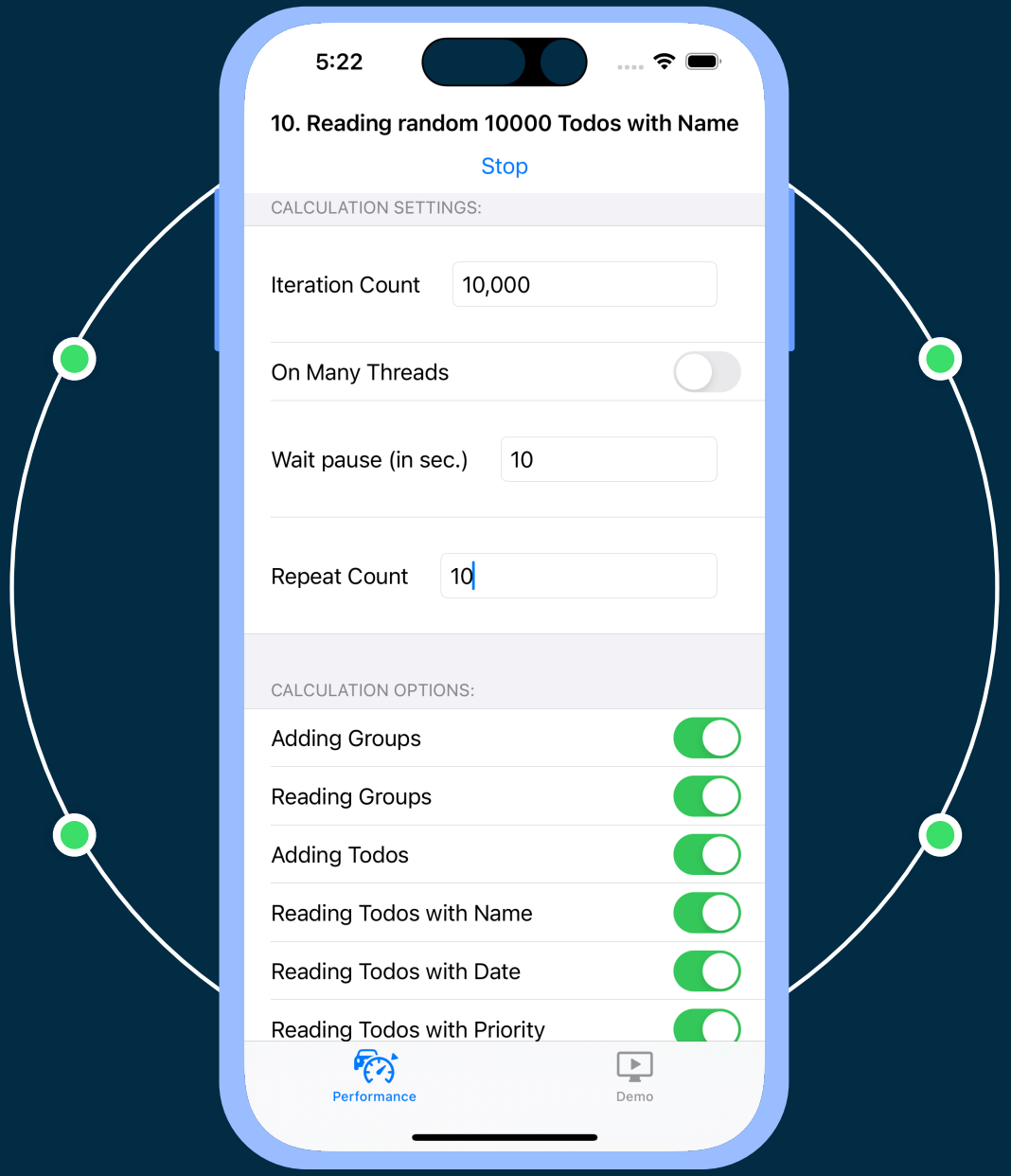
В сравнении участвуют…

vs

vs

Тест

Замеряем запись и чтения данных сущностей Group и Todo с разными запросами



10
замеров

10
секунд
перерыва

10 000
повторов
запроса

Group — тонкая,

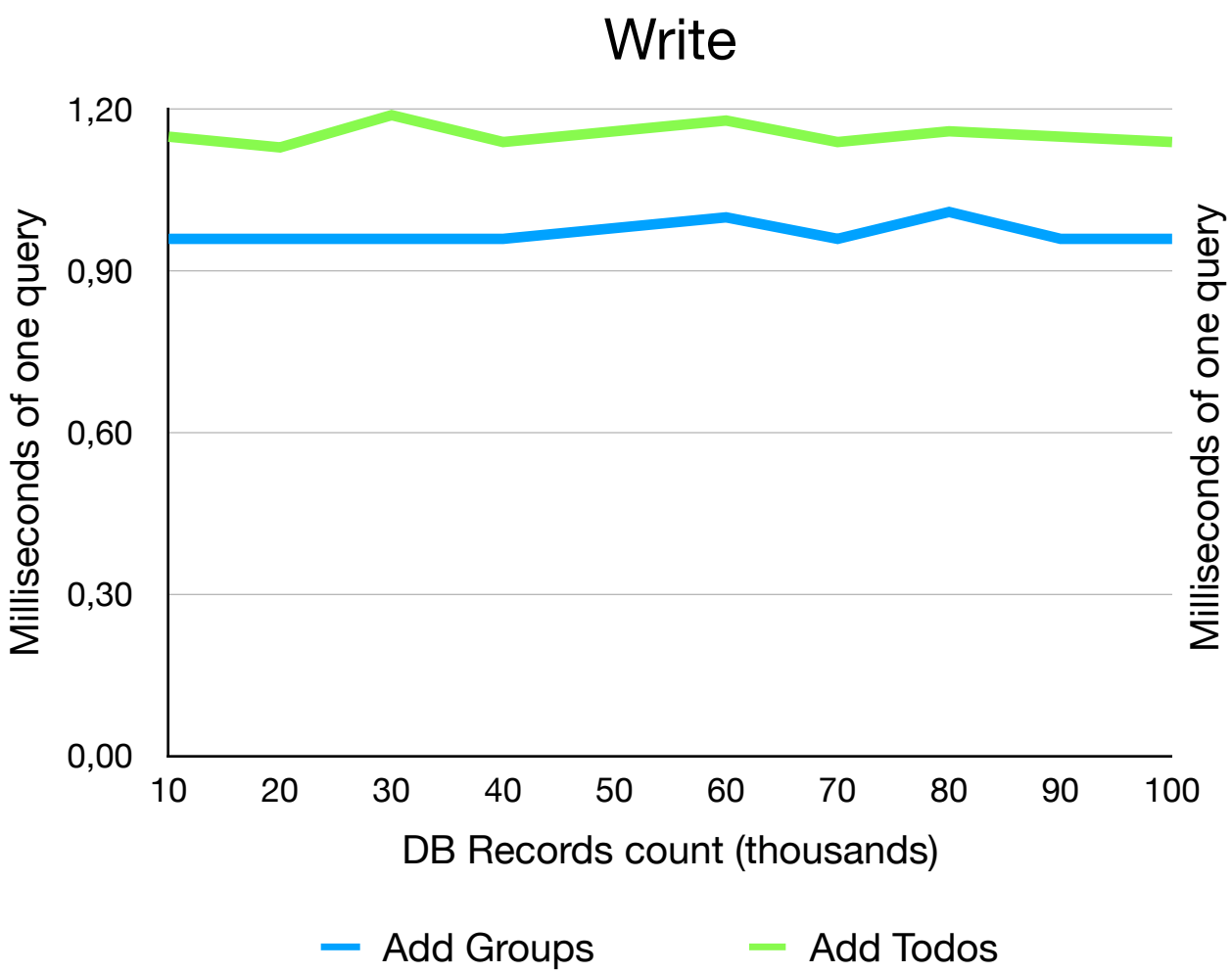
```
final class TodoGroupEntity : Model {

    static var schema: String = "TodoGroup"

    @ID(key: .id)
    var id: UUID?

    @Field(key: "name")
    var name: String

    @Children(for: \.$group)
    var todos: [TodoEntity]
}
```

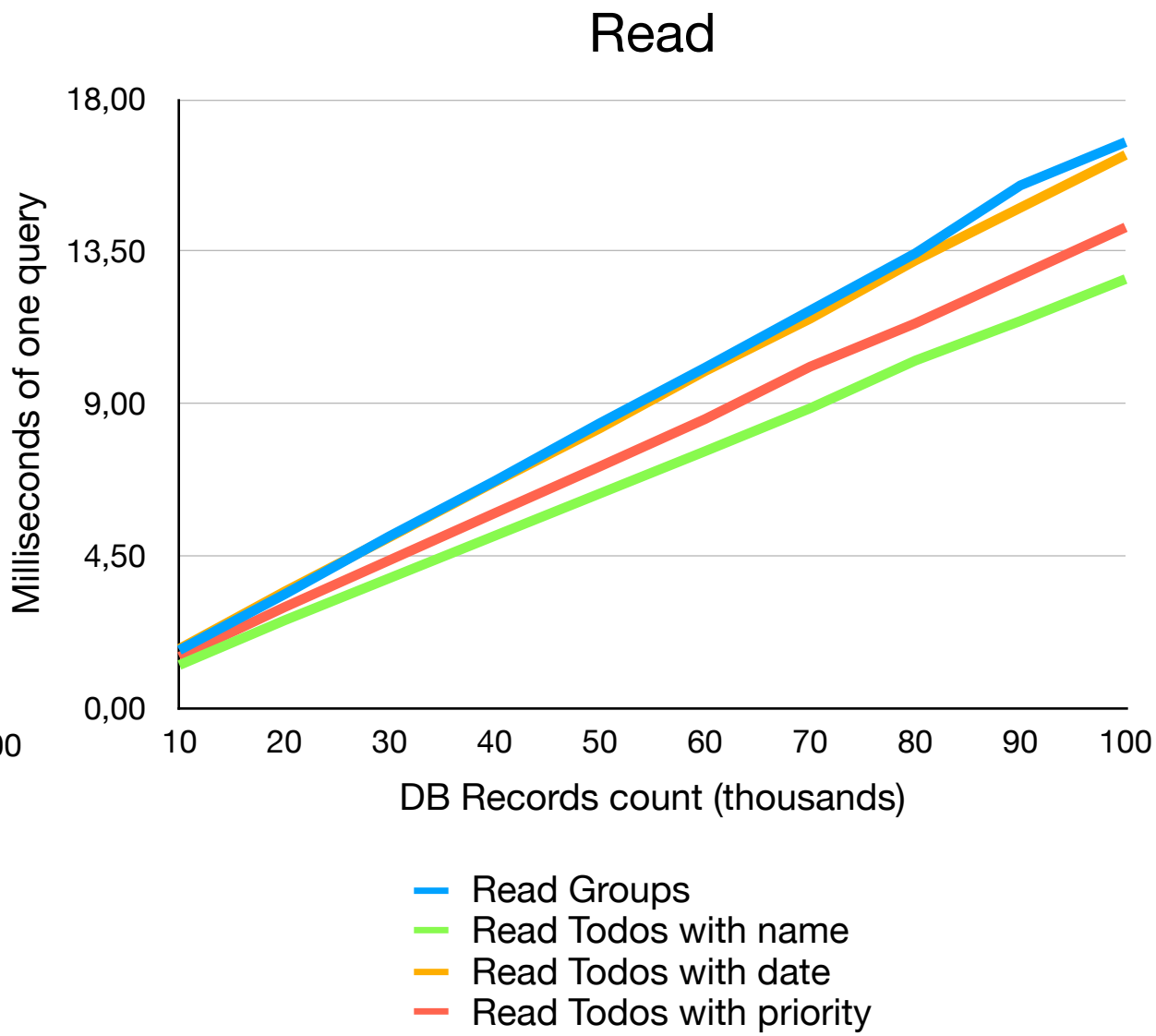
Todo — жирная

```
final class TodoEntity : Model {

    static var schema: String = "Todo"

    @ID(key: .id)
    var id: UUID?

    @Field(key: "name")
    var name: String

    @OptionalField(key: "comments")
    var comments: String?

    @OptionalField(key: "date")
    var date: Date?

    @OptionalParent(key: "group_id")
    var group: TodoGroupEntity?

    @OptionalField(key: "priority")
    var priority: Int?

    var count: Int = 0
}
```

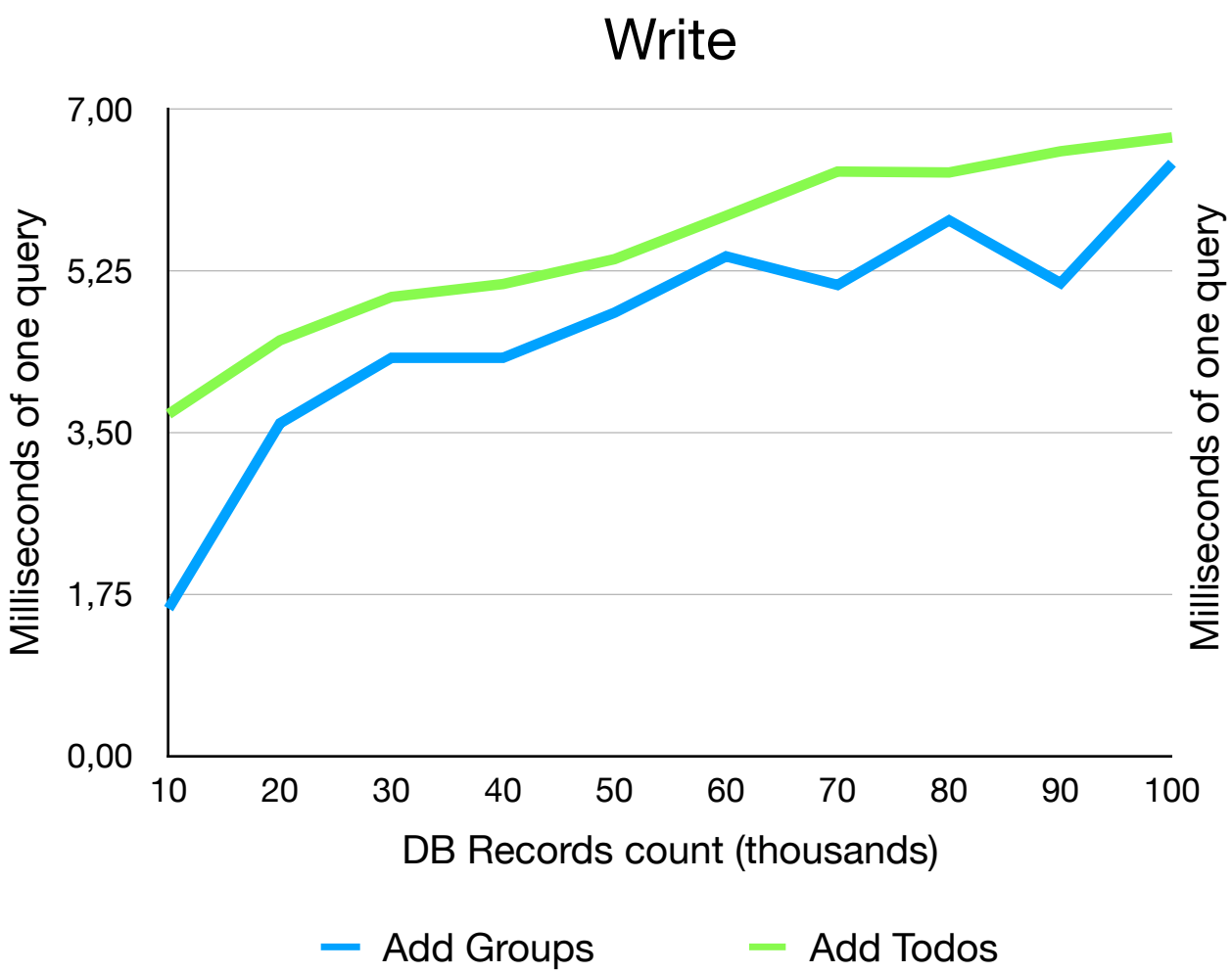
Анализ времени на один запрос

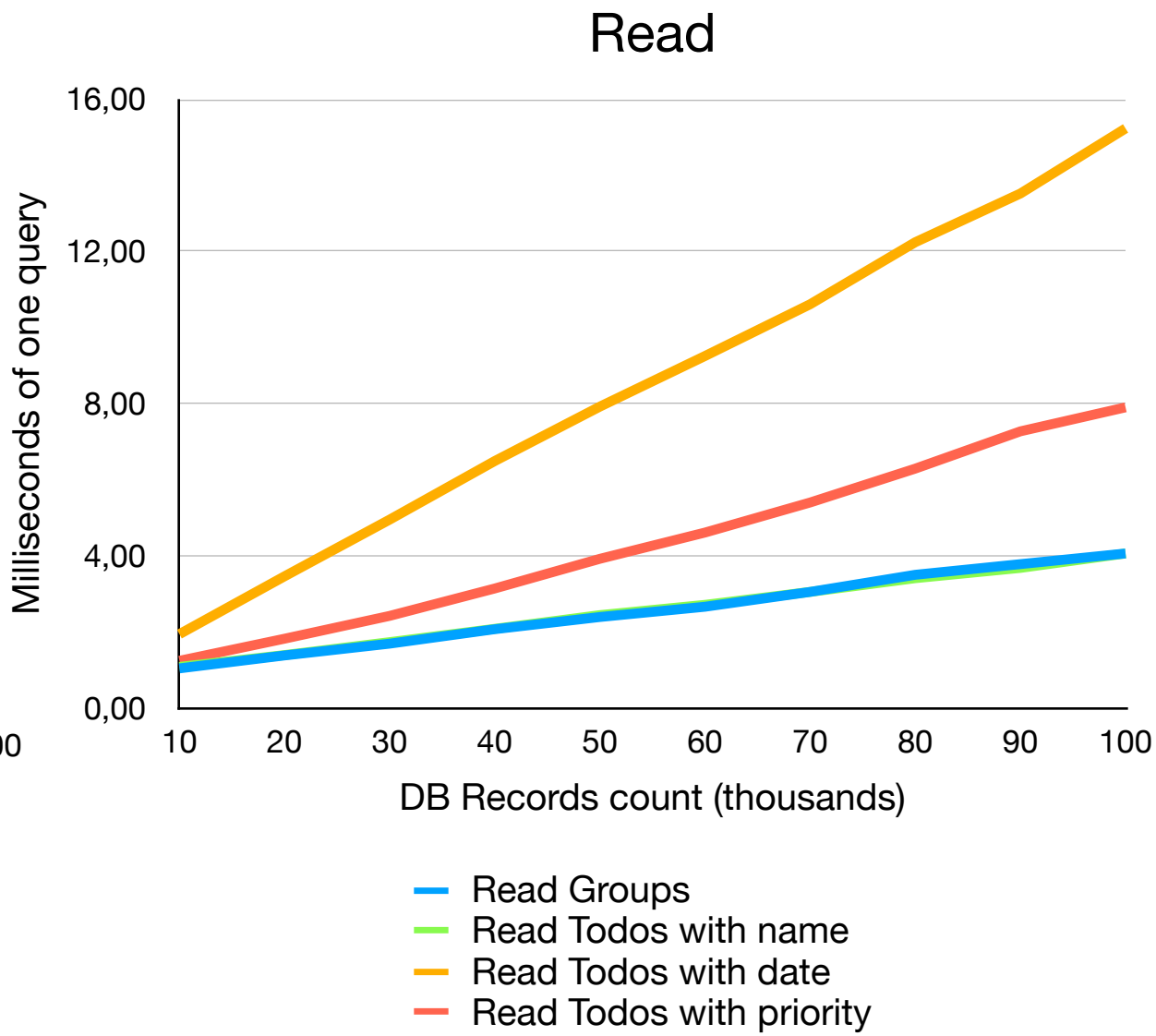
SwiftData


Write
1,20
0,90
0,60
0,30
0,00
Milliseconds of one query
10
20
30
40
50
60
70
80
90
100
DB Records count (thousands)
Add Groups
Add Todos

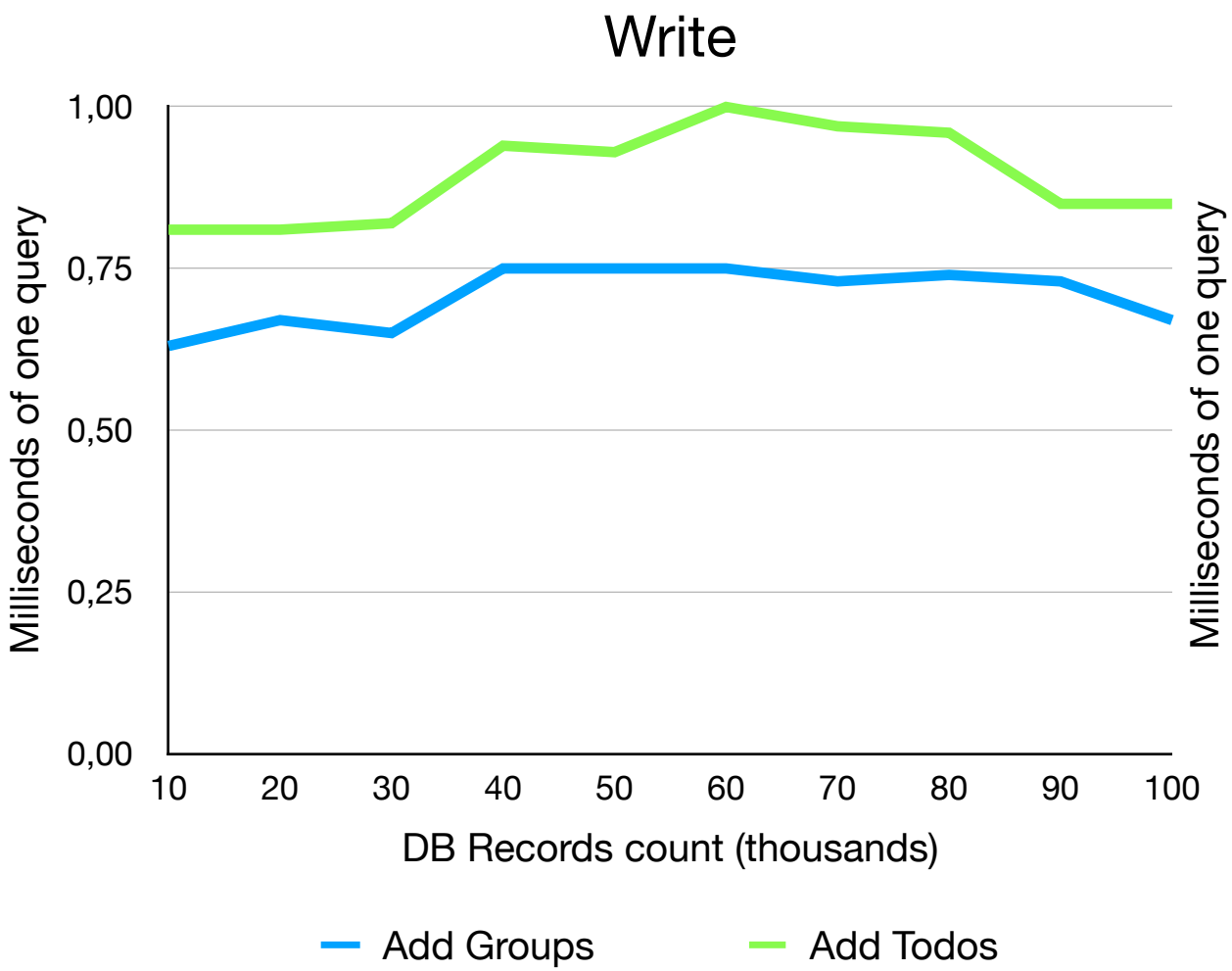

Read
18,00
13,50
9,00
4,50
0,00
Milliseconds of one query
10
20
30
40
50
60
70
80
90
100
DB Records count (thousands)
Read Groups
Read Todos with name
Read Todos with date
Read Todos with priority

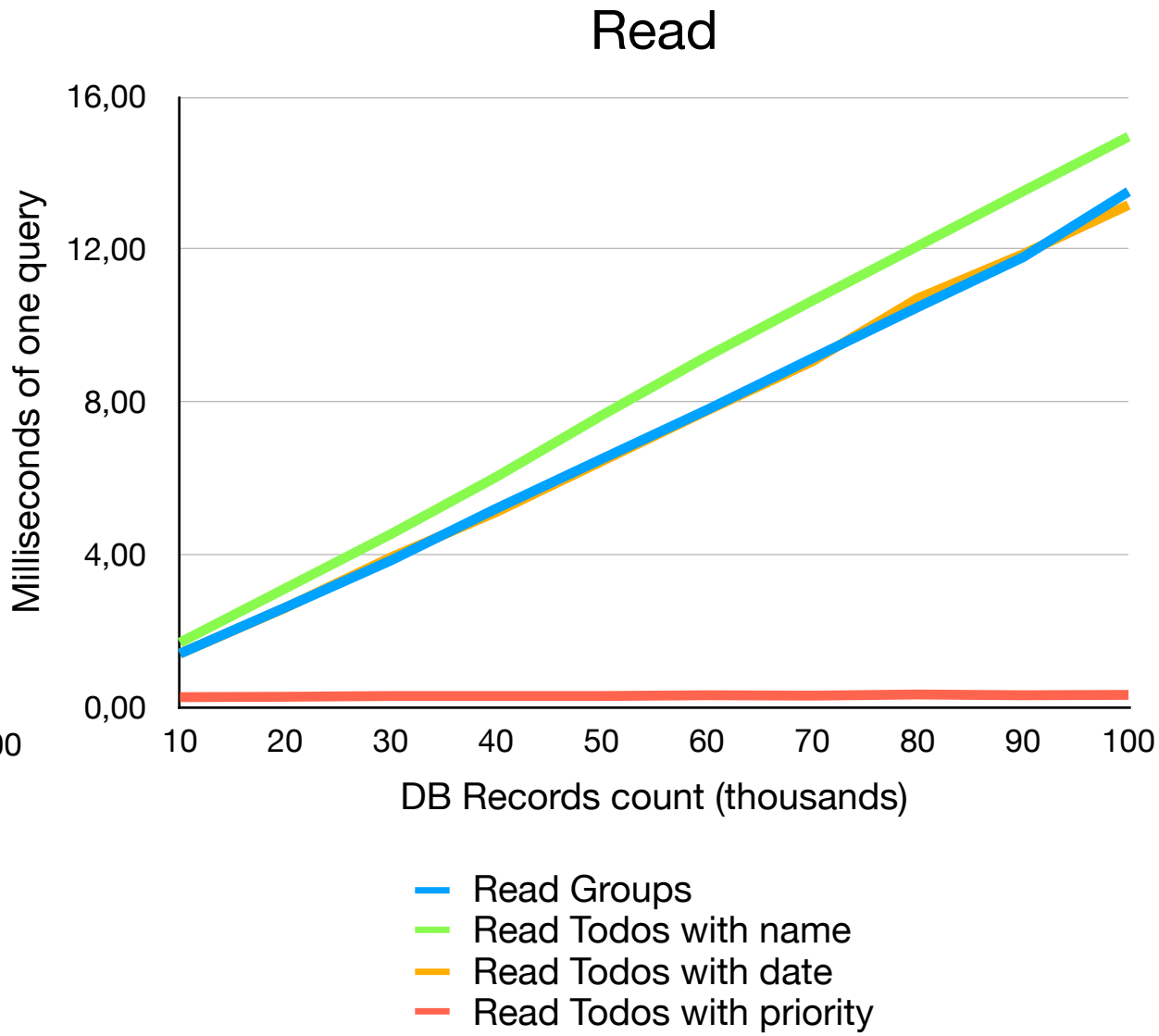
Realm


Write
7,00
5,25
3,50
1,75
0,00
Milliseconds of one query
10
20
30
40
50
60
70
80
90
100
DB Records count (thousands)
Add Groups
Add Todos

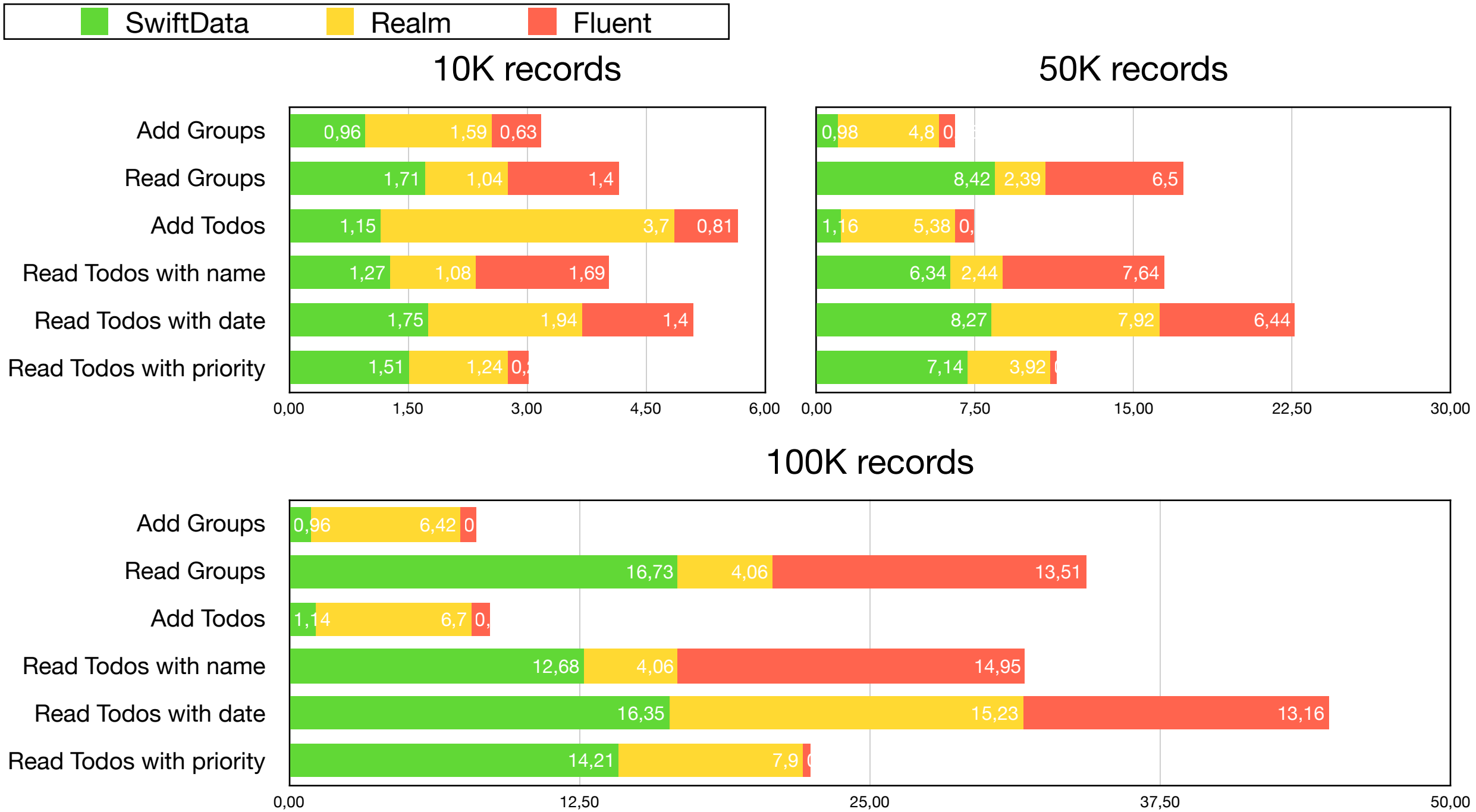

Read
16,00
12,00
8,00
4,00
0,00
Milliseconds of one query
10
20
30
40
50
60
70
80
90
100
DB Records count (thousands)
Read Groups
Read Todos with name
Read Todos with date
Read Todos with priority

Fluent

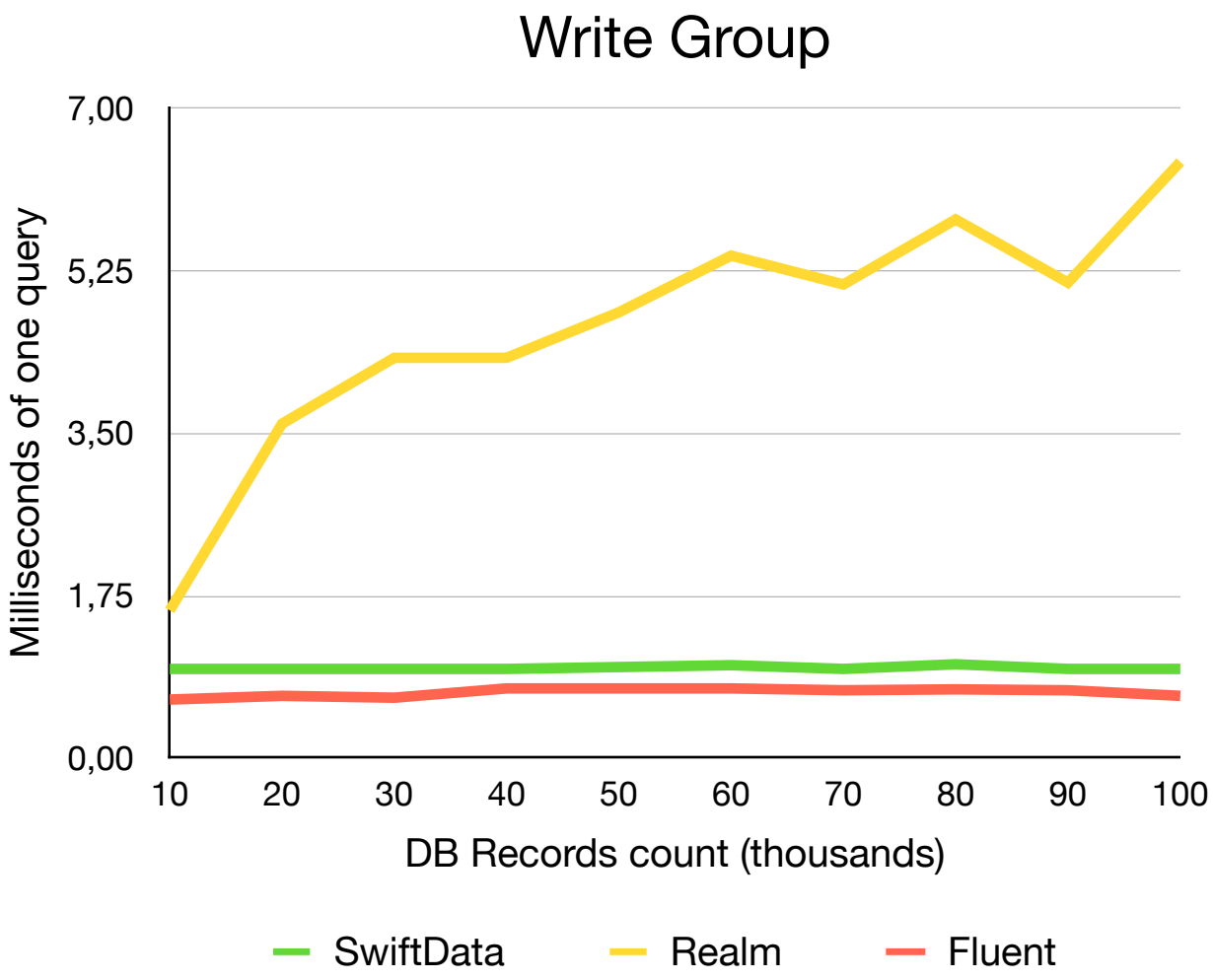

Write
1,00
0,75
0,50
0,25
0,00
Milliseconds of one query
10
20
30
40
50
60
70
80
90
100
DB Records count (thousands)
Add Groups
Add Todos

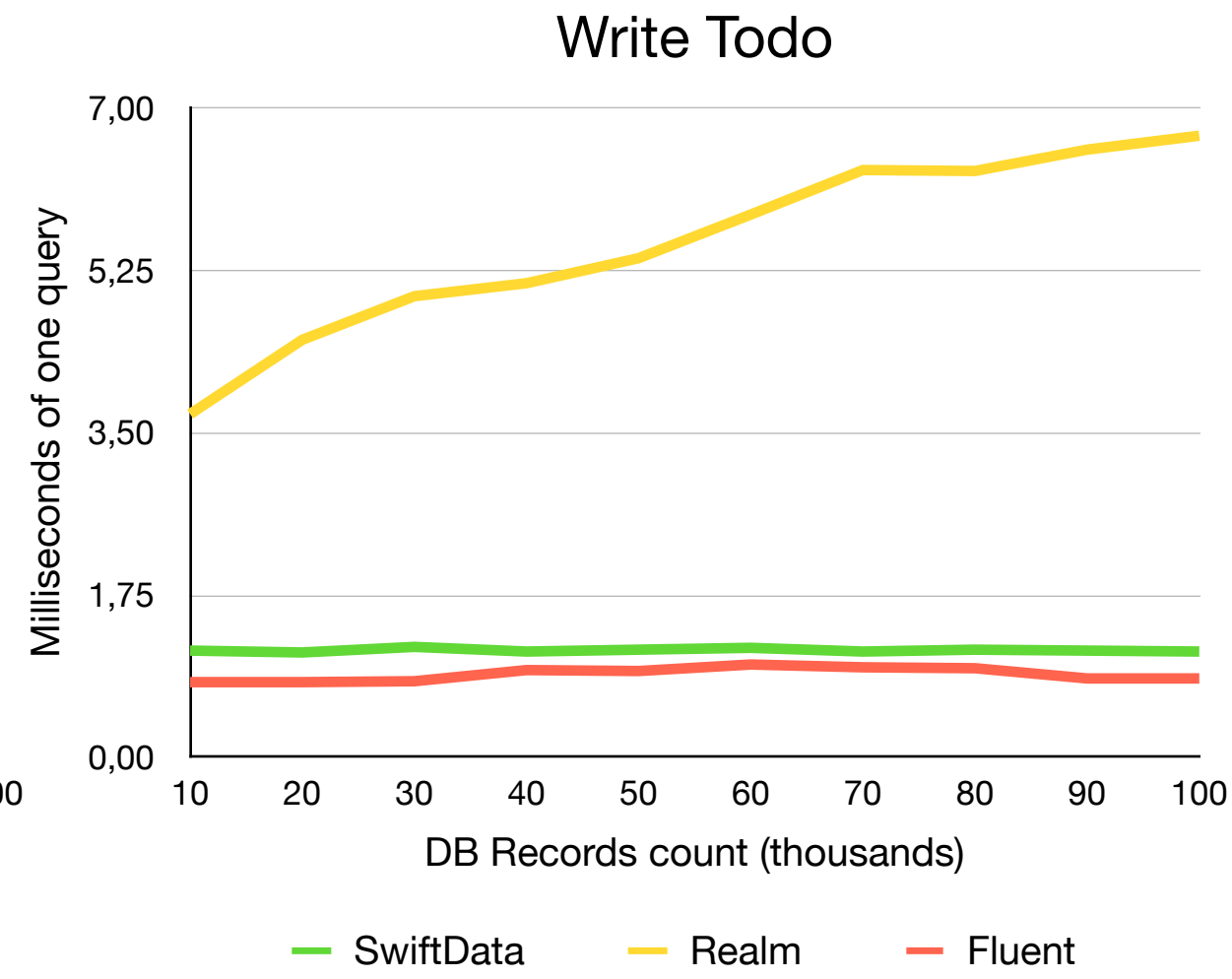

Read
16,00
12,00
8,00
4,00
0,00
Milliseconds of one query
10
20
30
40
50
60
70
80
90
100
DB Records count (thousands)
Read Groups
Read Todos with name
Read Todos with date
Read Todos with priority

Сравниваем между собой

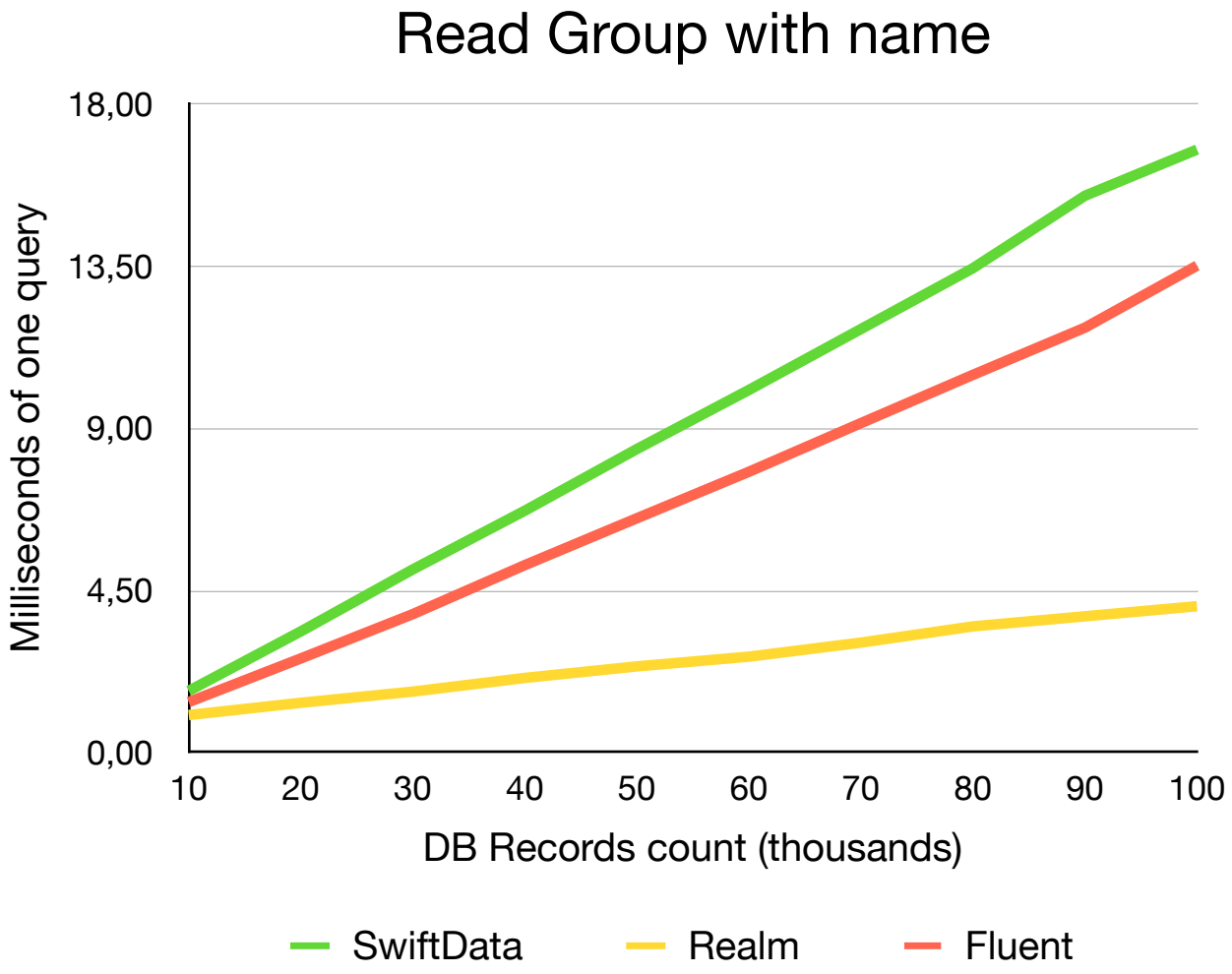

SwiftData
Realm
Fluent
10K records
Add Groups
Read Groups
Add Todos
Read Todos with name
Read Todos with date
Read Todos with priority
0,96
1,59
0,63
1,71
1,04
1,4
1,15
3,7
0,81
1,27
1,08
1,69
1,75
1,94
1,4
1,51
1,24
0,00
1,50
3,00
4,50
6,00
50K records
0,98
4,8
8,42
2,39
6,5
1,16
5,38
6,34
2,44
7,64
8,27
7,92
6,44
7,14
3,92
0,00
7,50
15,00
22,50
30,00
100K records
0,96
6,42
0
16,73
4,06
13,51
1,14
6,7
12,68
4,06
14,95
16,35
15,23
13,16
14,21
7,9
0,00
12,50
25,00
37,50
50,00

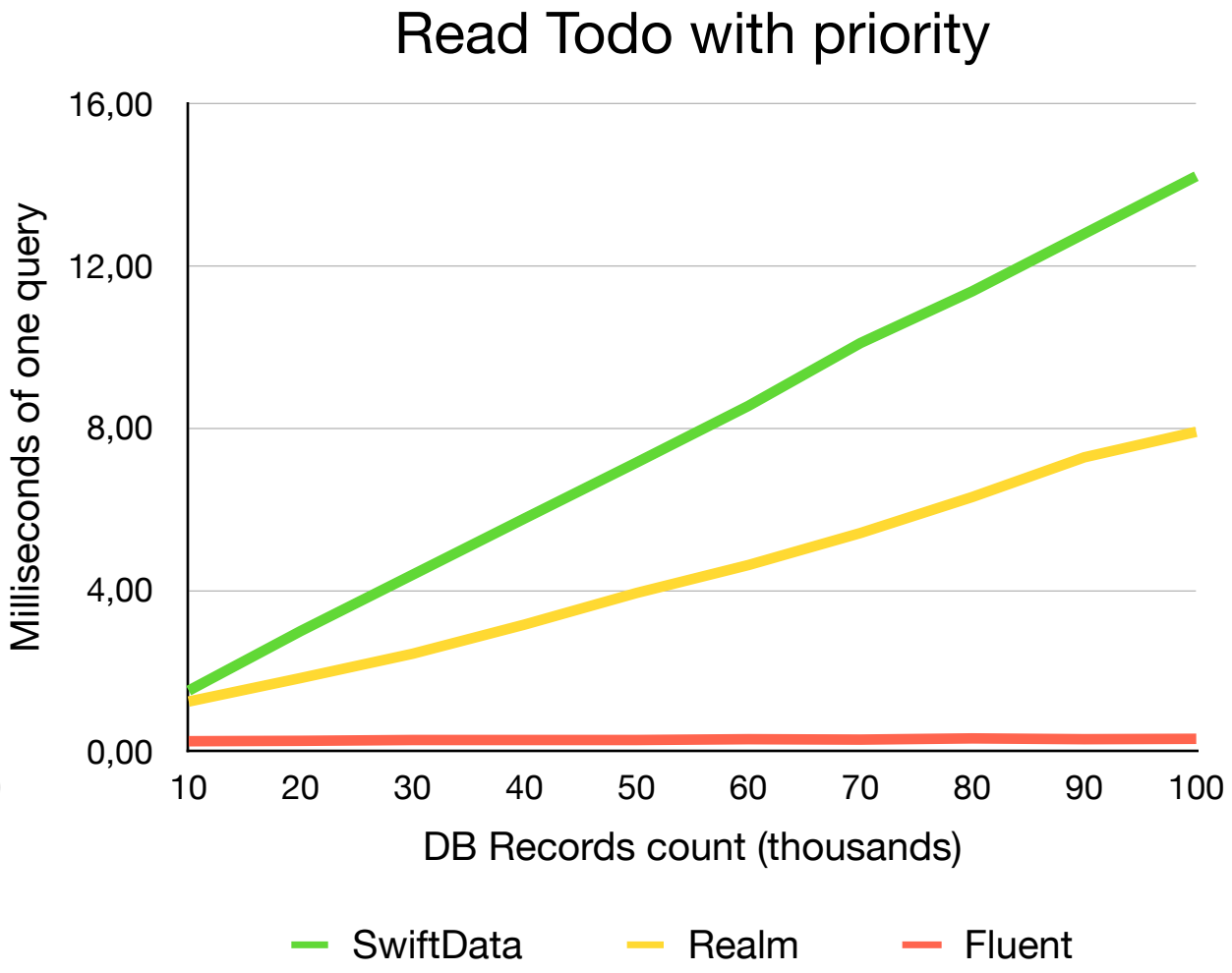
Сравнение фреймворков по записи





Сравнение по чтению индексированных полей





Финальный прирост производительности в процентах

Название фичи	SwiftData	Realm	Fluent
Запись	0	-50..-500	20..35
Чтение	0	-10..70	18..22
Фильтрация - Сортировка	0	17..44	82..97

Другие показатели в абсолютных величинах

Название фичи	SwiftData	Realm	Fluent
Поддержка iOS	17	12	13
БД на диске (100K records), Мбайт	37,9	14,9	27,4
Сборка на диске, Мбайт	0,43	14,1	8,4
Оперативная память (100K), Мбайт	39,3	44,9	63,2
Время Debug сборки, в сек.	1,8	80,7	29,4

Финальное сравнение

Название фичи - фреймворк	SwiftData	Realm	Fluent
Скорость записи			
Скорость чтения			
Фильтрация - Сортировка			
Поддержка iOS			
Память на диске			
Оперативная память			
Порядок в зависимых сущностях			
Функциональные запросы			
Время сборки			
Ускорение в конкурентных запросах			

Легенда:

 - Отлично

 - Норма

 - Такое

 - Плохо

Заключение

Что же выбрать в проде?

- Можно просто использовать UserDefaults или файлы
- **CoreData**, если сущностей мало и записей побольше
- **Realm:** плохо запись и конкурентность, отлично поиск по тексту
- **Fluent:** средневзвешенно оптимально
- **SwiftData:** может, когда-нибудь…

SwiftData/Realm/Fluent iOS example code

https://github.com/sofbix/DBiOS

OZON tech

Спасибо за внимание!

@sofbix

Сергей Балалаев, руководитель отдела разработки мобильного приложения «ПВЗ».
Компания Озон