



# Tips & Tricks com Realm no iOS

**TDC São Paulo**  
20 de julho de 2017

**Rafael Kellermann Streit**  
@rafaelks

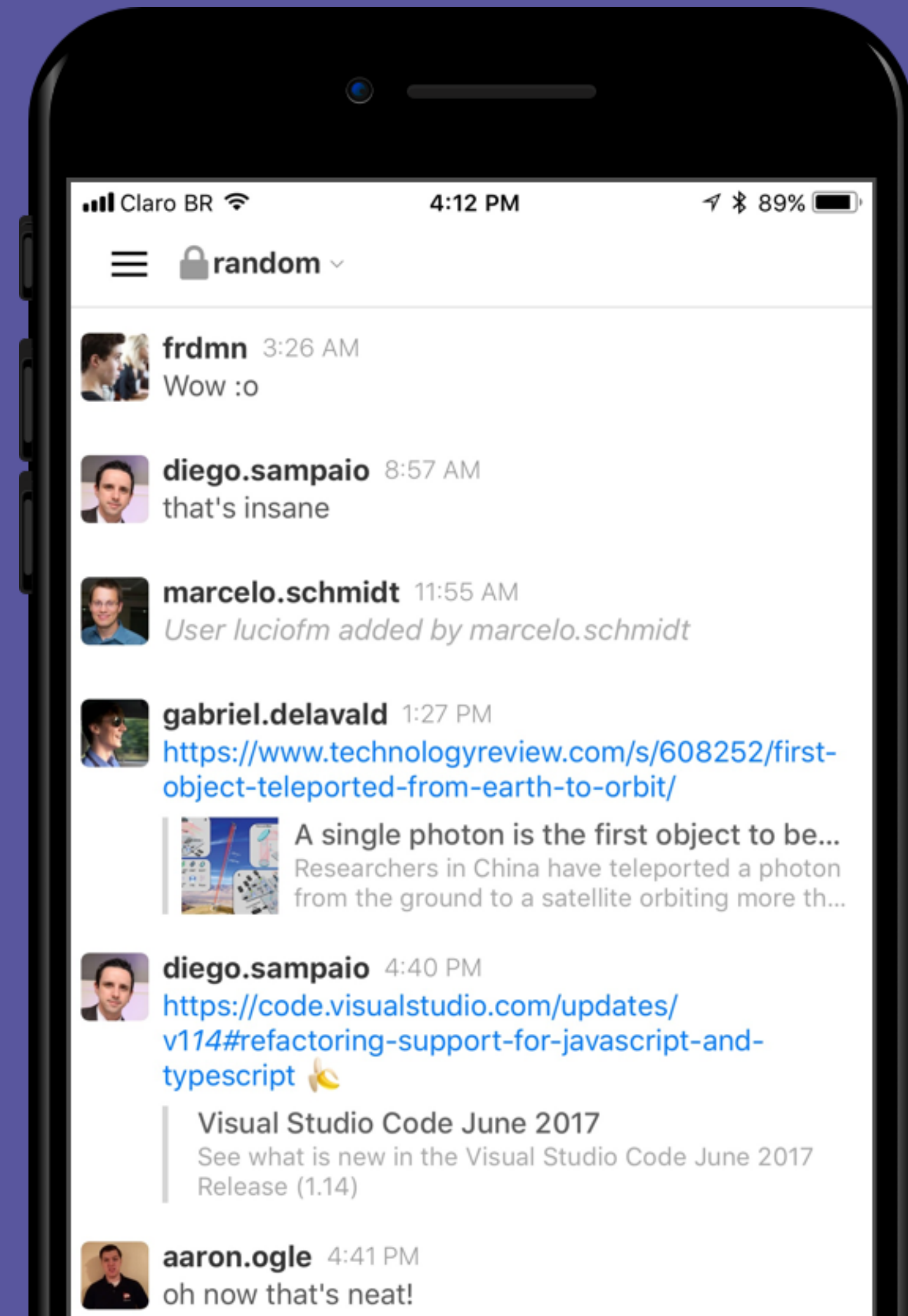
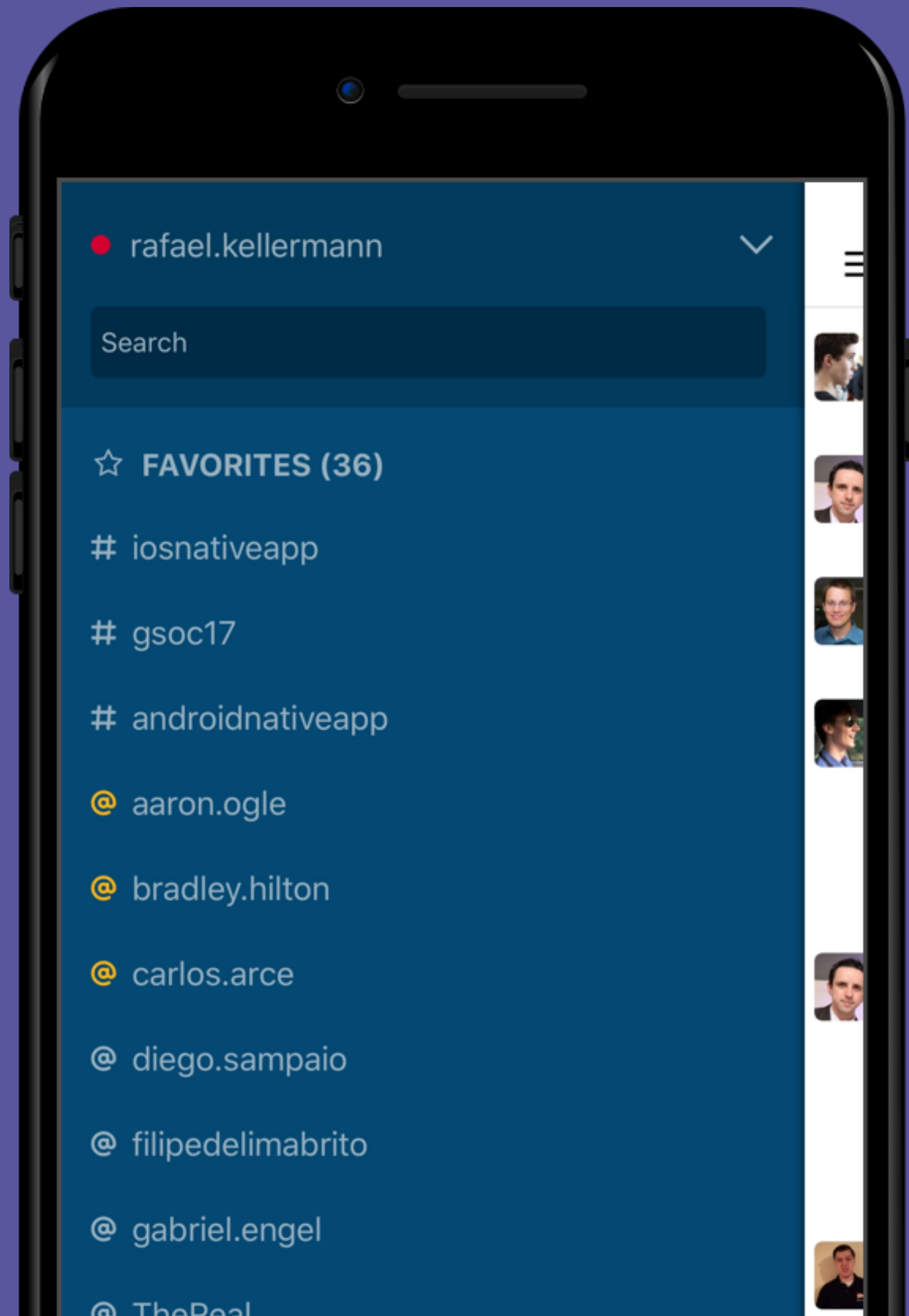


# Rocket.Chat

Open source chat platform

- Open source
- 10MM+ users
- 100K+ servers deployed
- Native apps for iOS and Android

# Rocket.Chat



# Realm?

- Open source database framework
- Complete replacement of Core Data / SQLite
- Based on ORM model
- Fast performance, simple API, thread safe
- Cross-platform

# Realm()

```
let realm = try? Realm()
```

# Object

```
final class Dog: Object {  
    dynamic var name: String?  
    dynamic var age: Int = 0  
    dynamic var createdAt: Date?  
}
```

# Start

```
// Create a Dog instance
let dog = Dog()
dog.name = "Foobar"
dog.age = 5
dog.createdAt = Date()

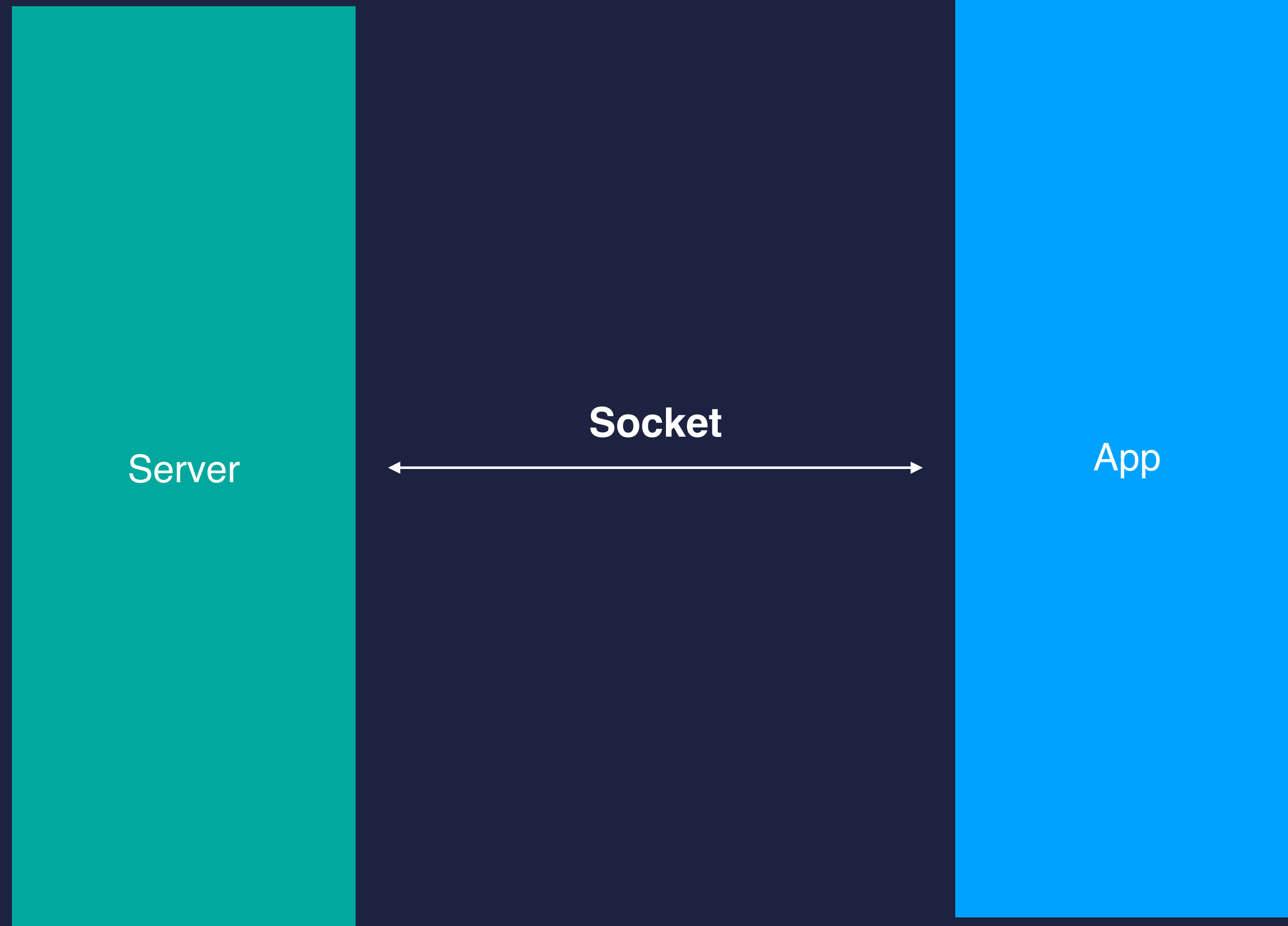
// Persist data in Realm
realm.write {
    realm.add(dog)
}
```

# Problem #1

Update UI with tons of  
single changes via socket

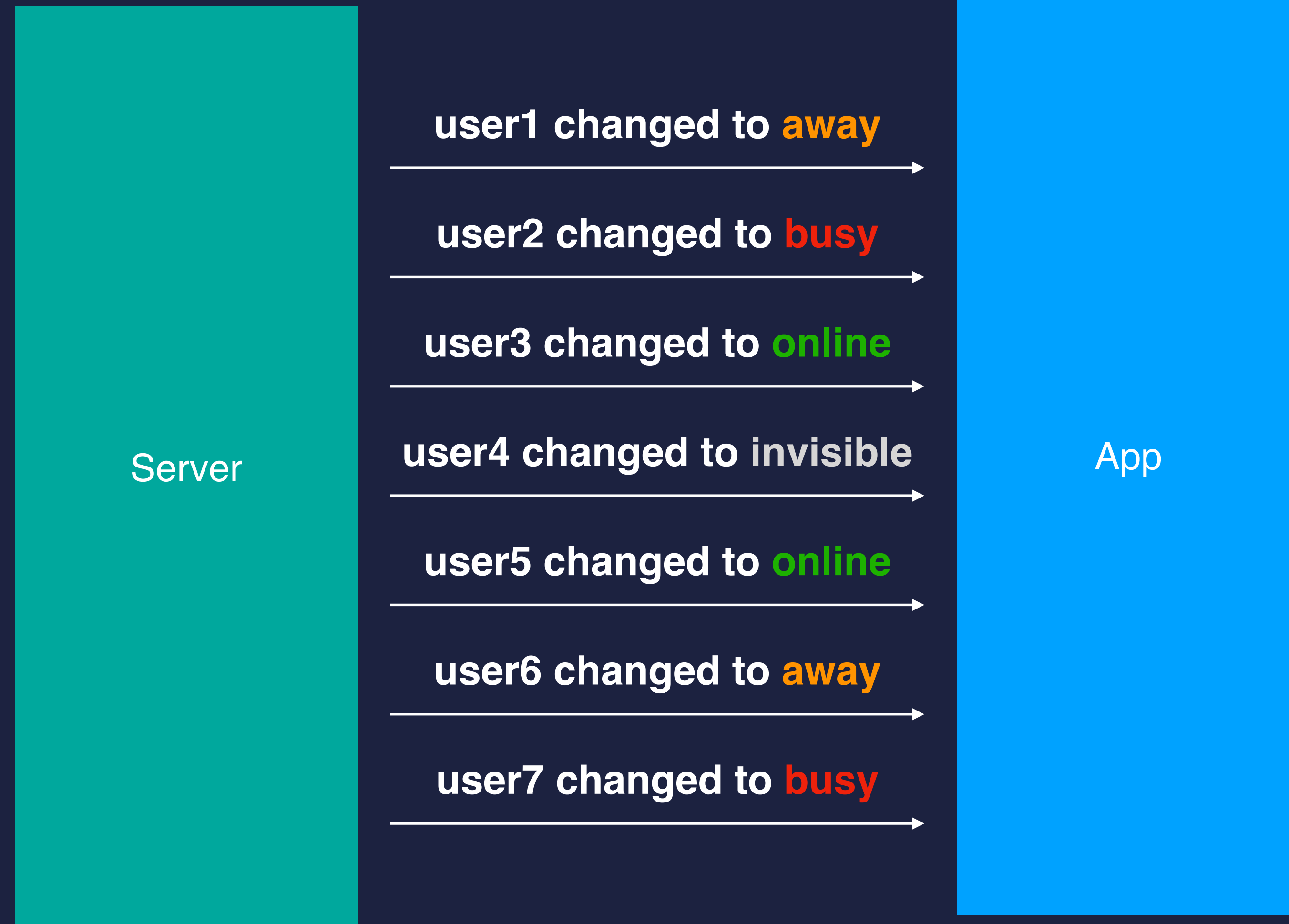


# App connection



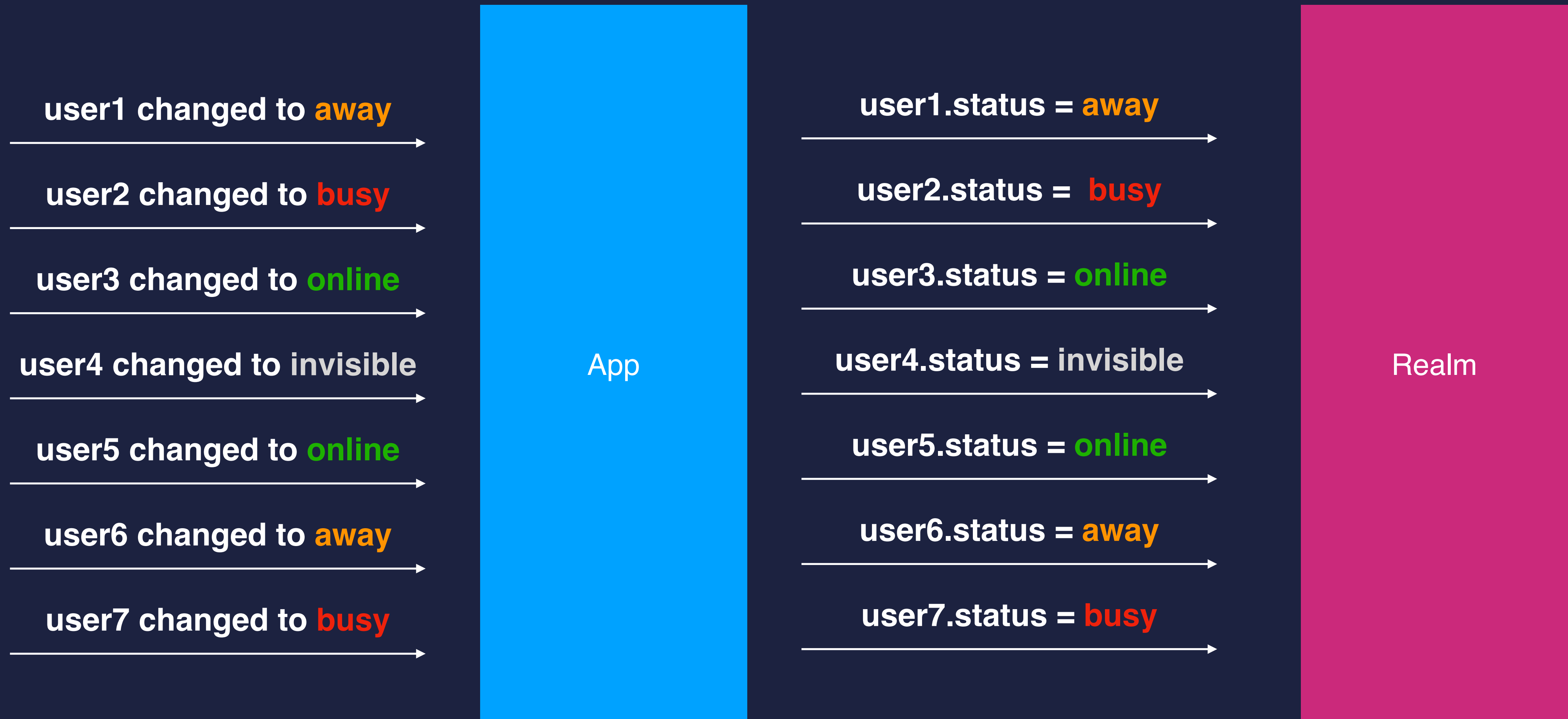
# Status update of users

## Socket messages



# Status update of users

Database update



# Status update of users

## UI update



# Notification block

```
token = query.addNotificationBlock { [weak self] changes in
    switch changes {
        case .initial(let results): break
        case .update(let results, let deletions, let insertions, let modifications): break
        case .error: break
    }
}
```

# Problem #2

Reuse model mapping &  
model handling code

# Base model

```
class BaseModel: Object {  
    dynamic var identifier: String?  
  
    override static func primaryKey() -> String? {  
        return "identifier"  
    }  
}
```

Handle model  
changes



# Protocols

## Handle updates

```
protocol ModelHandler {  
    func add(_ values: JSON, realm: Realm)  
    func update(_ values: JSON, realm: Realm)  
    func remove(_ values: JSON, realm: Realm)  
}
```


# Extensions

## Handle updates

```
guard let msg = result.msg else { return }
guard let identifier = result.result["id"].string else { return }
let fields = result.result["fields"]

switch collection {
  case "users":
    User.handle(msg: msg, primaryKey: identifier, values: fields)
    break
  case "subscriptions":
    Subscription.handle(msg: msg, primaryKey: identifier, values: fields)
    break
  default: break
}
```

**add, change, remove**



# Extensions

## Handle updates

```
extension ModelHandler where Self: BaseModel {
    static func handle(msg: ResponseMessage, primaryKey: String, values: JSON) {
        Realm.execute({ (realm) in
            var object: Self!

            if let existentObject = realm.object(ofType: Self.self, forPrimaryKey: primaryKey as AnyObject) {
                object = existentObject
            }

            if object == nil {
                object = Self()
                object.setValue(primaryKey, forKey: Self.primaryKey() ?? "")
            }

            switch msg {
            case .added:
                object.add(values, realm: realm)
                break
            case .changed:
                object.update(values, realm: realm)
                break
            case .removed:
                object.remove(values, realm: realm)
                break
            default:
                object.update(values, realm: realm)
                break
            }
        })
    }
}
```

# Handle model mapping

JSON > Realm

# Protocols

## Model mapping

```
protocol ModelMappable {  
    func map(_ values: JSON, realm: Realm?)  
}
```

# Protocols

## Model mapping

```
extension User: ModelMappable {
  func map(_ values: JSON, realm: Realm?) {
    if self.identifier == nil {
      self.identifier = values["_id"].string
    }

    if let username = values["username"].string {
      self.username = username
    }

    if let status = values["status"].string {
      self.status = UserStatus(rawValue: status) ?? .offline
    }
  }
}
```

# Final tip

**Avoid using main-thread  
to call Realm**

# Wrapper

## Background execution

```
extension Realm {
    static func execute(_ execution: @escaping (Realm) -> Void, completion: VoidCompletion? = nil) {
        var backgroundTaskId: UIBackgroundTaskIdentifier?

        let name = "chat.rocket.realm.background"
        backgroundTaskId = UIApplication.shared.beginBackgroundTask(withName: name, expirationHandler: { _ in
            backgroundTaskId = UIBackgroundTaskInvalid
        })

        if let backgroundTaskId = backgroundTaskId {
            DispatchQueue.global(qos: .background).async { _ in
                guard let realm = try? Realm() else { return }
                try? realm.write {
                    execution(realm)
                }

                DispatchQueue.main.async {
                    completion?()
                }

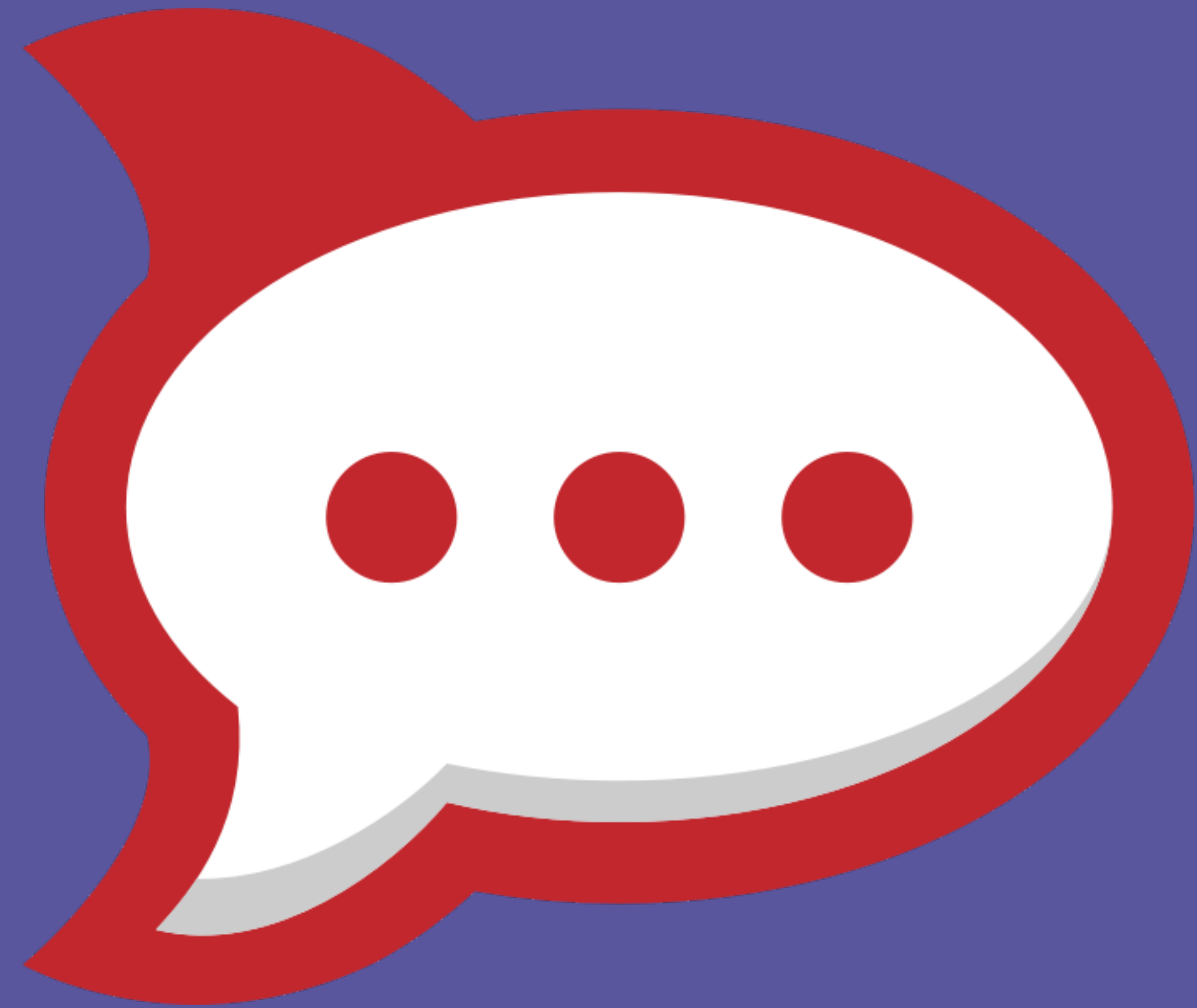
                UIApplication.shared.endBackgroundTask(backgroundTaskId)
            }
        }
    }
}
```



# Wrapper

## Example

```
Realm.execute({ (realm) in
    let users = realm.objects(User.self)
    users.setValue("offline", forKey: "status")
}, completion: {
    done()
})
```



# Rocket.Chat.iOS

[github.com/RocketChat/Rocket.Chat.iOS](https://github.com/RocketChat/Rocket.Chat.iOS)

# Thanks!

- @rafaelks
- [github.com/RocketChat](https://github.com/RocketChat)