Swifty Tables

1. Register cell
2. Dequeue cell
3. Configure cell
4. Profit! err… Use cell

Reusable Views

```
class TalksViewController: UITableViewController {
    let reuseIdentifier = "reuseIdentifier"

    // ...

   override func viewDidLoad() {
        super.viewDidLoad()

        tableView.register(TalkCell.self, forCellReuseIdentifier: reuseIdentifier)
   }

    override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
        let cell = tableView.dequeueReusableCell(withIdentifier: reuseIdentifier) as! TalkCell

        // configure cell

        return cell
    }

    // …
}
```

```swift
let nib = UINib(nibName: "MeetupCell", bundle: nil)

tableView.register(nib, forCellReuseIdentifier: reuseIdentifier)
```

```swift
protocol Reusable: AnyObject {
    static var reuseIdentifier: String { get }
}
```

```swift
protocol Reusable: AnyObject {
    static var reuseIdentifier: String { get }
}

extension Reusable {
    static var reuseIdentifier: String {
        return String(describing: self)
    }
}
```

```swift
protocol NibLoadable: AnyObject {
    static var nib: UINib { get }
}

extension NibLoadable {
    static var nib: UINib {
        return UINib(nibName: String(describing: self),
                     bundle: Bundle(for: self))
    }
}

typealias NibReusable = Reusable & NibLoadable
```

```swift
protocol NibLoadable: AnyObject {
    static var nib: UINib { get }
}

extension NibLoadable {
    static var nib: UINib {
        return UINib(nibName: String(describing: self),
                     bundle: Bundle(for: self))
    }
}

typealias NibReusable = Reusable & NibLoadable
```

```swift
extension UITableView {
   func register<T: UITableViewCell>(_: T.Type) where T: Reusable {
            self.register(T.self, forCellReuseIdentifier: T.reuseIdentifier)
    }
}
```

```swift
extension UITableView {
    func register<T: UITableViewCell>(_: T.Type) where T: Reusable {
            self.register(T.self, forCellReuseIdentifier: T.reuseIdentifier)
     }
}
```

```
extension UITableView {
   func register<T: UITableViewCell>(_: T.Type) where T: Reusable {
            self.register(T.self, forCellReuseIdentifier: T.reuseIdentifier)
    }
}
```

```
func register<T: UITableViewCell>(_: T.Type)
        where T: Reusable & NibLoadable {

     self.register(T.nib, forCellReuseIdentifier: T.reuseIdentifier)
}
```

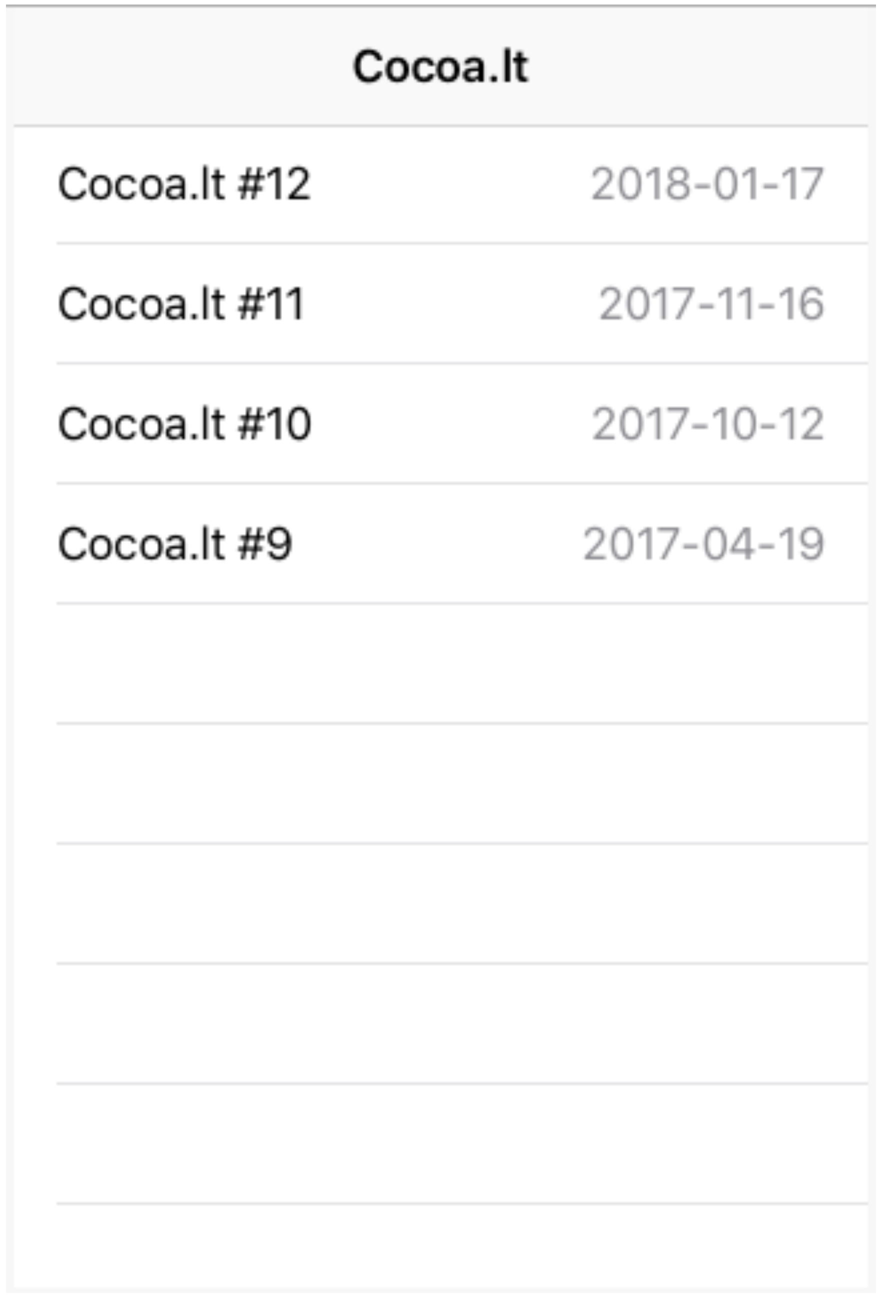
```swift
func dequeueReusableCell<T: UITableViewCell>(for indexPath: IndexPath) -> T
    where T: Reusable {
        guard let cell = self.dequeueReusableCell(
            withIdentifier: T.reuseIdentifier,
            for: indexPath) as? T
        else {
            fatalError("Could not dequeue a cell for \(T.reuseIdentifier)")
        }
    return cell
}
```

```swift
final class MeetupCell: UITableViewCell, Reusable {}
final class MeetupCell: UITableViewCell, Reusable {}

// MeetupsViewController
tableView.register(MeetupCell.self)
```

```
override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {

    let cell = tableView.dequeueReusableCell(for: indexPath) as MeetupCell
```

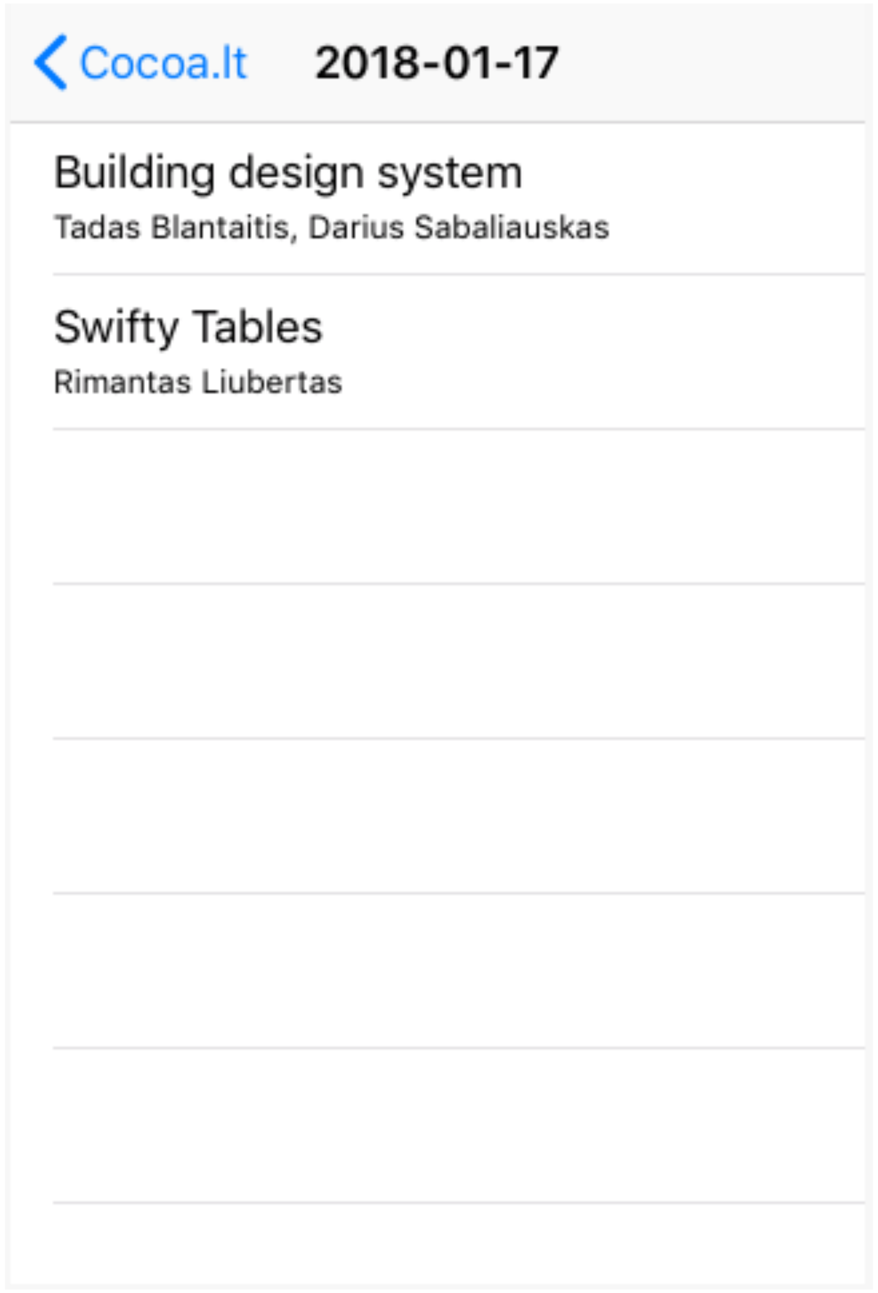
```swift
override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {

    let cell: MeetupCell = tableView.dequeueReusableCell(for: indexPath)
```

Generic Tables



Cocoa.lt

Cocoa.lt #12	2018-01-17
Cocoa.lt #11	2017-11-16
Cocoa.lt #10	2017-10-12
Cocoa.lt #9	2017-04-19


Cocoa.lt 2018-01-17

Building design system
Tadas Blantaitis, Darius Sabaliauskas

Swifty Tables
Rimantas Liubertas

```swift
final class MeetupsViewController: UITableViewController {
    var meetups: [Meetup] = []
    var didSelect: ((Meetup) -> Void)?

    init(meetups: [Meetup]) {
        super.init(style: .plain)
        self.meetups = meetups
    }

    required init?(coder aDecoder: NSCoder) {
        fatalError("init(coder:) has not been implemented")
    }

    override func viewDidLoad() {
        super.viewDidLoad()
        tableView.register(MeetupCell.self, forCellReuseIdentifier: "MeetupCell")
    }

    override func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {
        let meetup = meetups[indexPath.row]
        didSelect?(meetup)
    }

    override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        return meetups.count
    }

    override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) ->
UITableViewCell {
        let meetup = meetups[indexPath.row]
        let cell = tableView.dequeueReusableCell(withIdentifier: "MeetupCell", for: indexPath) as!
MeetupCell
        cell.configure(with: meetup)
        return cell
    }
}
```

```
final class TalksViewController: UITableViewController {
    var talks: [Talk] = []
    var didSelect: ((Talk) -> Void)?

    init(talks: [Talk]) {
        super.init(style: .plain)
        self.talks = talks
    }

    required init?(coder aDecoder: NSCoder) {
        fatalError("init(coder:) has not been implemented")
    }

    override func viewDidLoad() {
        super.viewDidLoad()
        tableView.register(TalkCell.self, forCellReuseIdentifier: "MeetupCell")
    }

    override func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {
        let talk = talks[indexPath.row]
        didSelect?(talk)
    }

    override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        return talks.count
    }

    override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) ->
UITableViewCell {
        let talk = talks[indexPath.row]
        let cell = tableView.dequeueReusableCell(withIdentifier: "MeetupCell", for: indexPath) as!
TalkCell
        cell.configure(with: talk)
        return cell
    }
}
```

```
final class TalksViewController: UITableViewController {
    var talks: [Talk] = []
    var didSelect: ((Talk) -> Void)?

    init(talks: [Talk]) {
        super.init(style: .plain)
        self.talks = talks
    }

    required init?(coder aDecoder: NSCoder) {
        fatalError("init(coder:) has not been implemented")
    }

    override func viewDidLoad() {
        super.viewDidLoad()
        tableView.register(TalkCell.self, forCellReuseIdentifier: "MeetupCell")
    }

    override func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {
        let talk = talks[indexPath.row]
        didSelect?(talk)
    }

    override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        return talks.count
    }

    override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) ->
UITableViewCell {
        let talk = talks[indexPath.row]
        let cell = tableView.dequeueReusableCell(withIdentifier: "MeetupCell", for: indexPath) as!
TalkCell
        cell.configure(with: talk)
        return cell
    }
}
```

```
final class TalksViewController: UITableViewController {
    var items: [Talk] = []
    var didSelect: ((Talk) -> Void)?

    init(items: [Talk]) {
        super.init(style: .plain)
        self.items = items
    }

    required init?(coder aDecoder: NSCoder) {
        fatalError("init(coder:) has not been implemented")
    }

    override func viewDidLoad() {
        super.viewDidLoad()
        tableView.register(TalkCell.self, forCellReuseIdentifier: "Cell")
    }

    override func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {
        let item = items[indexPath.row]
        didSelect?(item)
    }

    override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        return items.count
    }

    override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) ->
UITableViewCell {
        let item = items[indexPath.row]
        let cell = tableView.dequeueReusableCell(withIdentifier: "Cell", for: indexPath) as! TalkCell
        cell.configure(with: item)
        return cell
    }
}
```

```swift
final class ItemsViewController<Item, Cell: UITableViewCell>: UITableViewController
  where Cell: Reusable {

    var items: [Item] = []
    var didSelect: (Item) -> () = { _ in }
    var configure: (Cell, Item) -> () = { _, _ in }

    init(items: [Item]) {
        super.init(style: .plain)
        self.items = items
    }

    required init?(coder aDecoder: NSCoder) {
        fatalError("init(coder:) has not been implemented")
    }

    override func viewDidLoad() {
        super.viewDidLoad()
        tableView.register(Cell.self)
    }

    override func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {
        let item = items[indexPath.row]
        didSelect(item)
    }

    override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        return items.count
    }

    override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) ->
UITableViewCell {
        let item = items[indexPath.row]
        let cell: Cell = tableView.dequeueReusableCell(for: indexPath)
        configure(cell, item)
        return cell
    }
}
```

```
final class ItemsViewController<Item, Cell: UITableViewCell>: UITableViewController
  where Cell: Reusable {

    var items: [Item] = []
    var didSelect: (Item) -> () = { _ in }
    var configure: (Cell, Item) -> () = { _, _ in }

    init(items: [Item]) {
        super.init(style: .plain)
        self.items = items
    }

    required init?(coder aDecoder: NSCoder) {
        fatalError("init(coder:) has not been implemented")
    }

    override func viewDidLoad() {
        super.viewDidLoad()
        tableView.register(Cell.self)
    }

    override func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {
        let item = items[indexPath.row]
        didSelect(item)
    }

    override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        return items.count
    }

    override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) ->
UITableViewCell {
        let item = items[indexPath.row]
        let cell: Cell = tableView.dequeueReusableCell(for: indexPath)
        configure(cell, item)
        return cell
    }
}
```

```swift
final class ItemsViewController<Item, Cell: UITableViewCell>: UITableViewController
  where Cell: Reusable {

    var items: [Item] = []
    var didSelect: (Item) -> () = { _ in }
    var configure: (Cell, Item) -> () = { _, _ in }

    init(items: [Item]) {
        super.init(style: .plain)
        self.items = items
    }

    required init?(coder aDecoder: NSCoder) {
        fatalError("init(coder:) has not been implemented")
    }

    override func viewDidLoad() {
        super.viewDidLoad()
        tableView.register(Cell.self)
    }

    override func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {
        let item = items[indexPath.row]
        didSelect(item)
    }

    override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        return items.count
    }

    override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) ->
UITableViewCell {
        let item = items[indexPath.row]
        let cell: Cell = tableView.dequeueReusableCell(for: indexPath)
        configure(cell, item)
        return cell
    }
}
```

```swift
final class ItemsViewController<Item, Cell: UITableViewCell>: UITableViewController
  where Cell: Reusable {

    var items: [Item] = []
    var didSelect: (Item) -> () = { _ in }
    var configure: (Cell, Item) -> () = { _, _ in }

    init(items: [Item]) {
        super.init(style: .plain)
        self.items = items
    }

    required init?(coder aDecoder: NSCoder) {
        fatalError("init(coder:) has not been implemented")
    }

    override func viewDidLoad() {
        super.viewDidLoad()
        tableView.register(Cell.self)
    }

    override func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {
        let item = items[indexPath.row]
        didSelect(item)
    }

    override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        return items.count
    }

    override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) ->
UITableViewCell {
        let item = items[indexPath.row]
        let cell: Cell = tableView.dequeueReusableCell(for: indexPath)
        configure(cell, item)
        return cell
    }
}
```

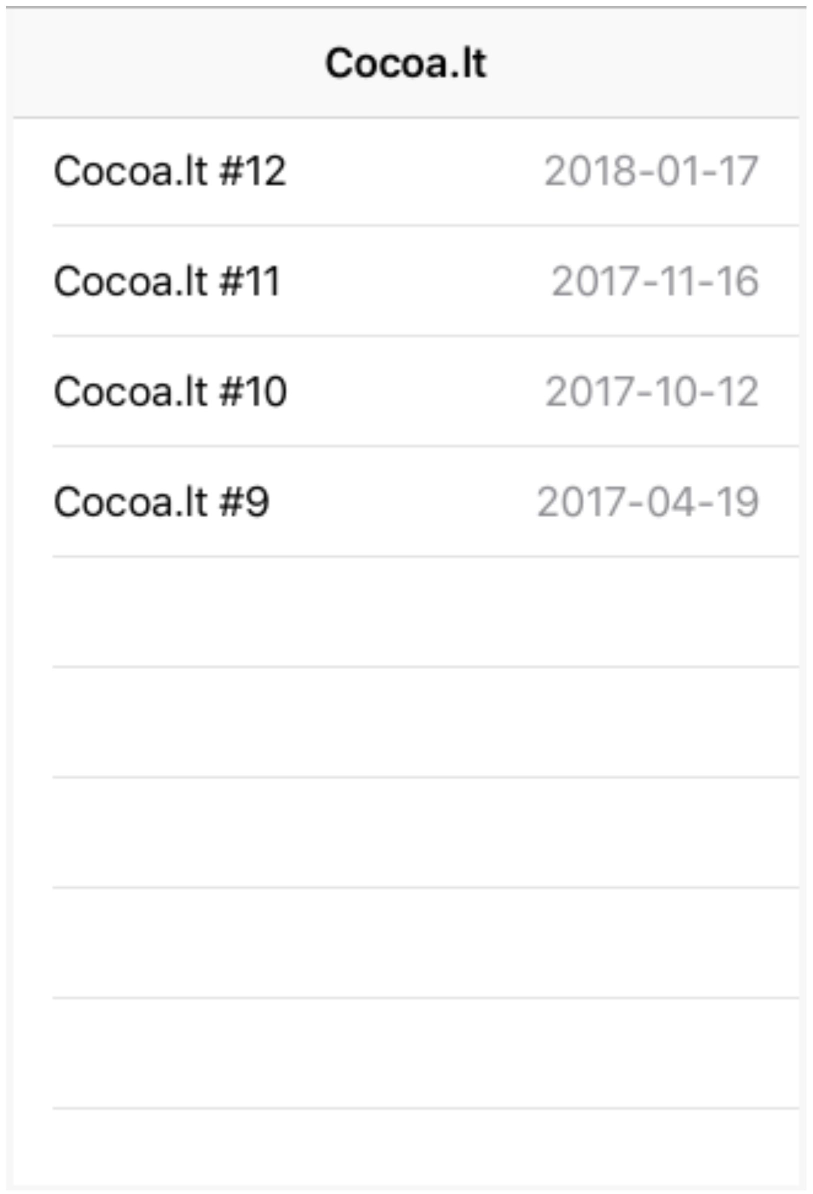
```swift
final class ItemsViewController<Item, Cell: UITableViewCell>: UITableViewController
  where Cell: Reusable {

    var items: [Item] = []
    var didSelect: (Item) -> () = { _ in }
    var configure: (Cell, Item) -> () = { _, _ in }

    init(items: [Item]) {
        super.init(style: .plain)
        self.items = items
    }

    required init?(coder aDecoder: NSCoder) {
        fatalError("init(coder:) has not been implemented")
    }

    override func viewDidLoad() {
        super.viewDidLoad()
        tableView.register(Cell.self)
    }

    override func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {
        let item = items[indexPath.row]
        didSelect(item)
    }

    override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        return items.count
    }

    override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) ->
UITableViewCell {
        let item = items[indexPath.row]
        let cell: Cell = tableView.dequeueReusableCell(for: indexPath)
        configure(cell, item)
        return cell
    }
}
```

```swift
final class ItemsViewController<Item, Cell: UITableViewCell>: UITableViewController
  where Cell: Reusable {

    var items: [Item] = []
    var didSelect: (Item) -> () = { _ in }
    var configure: (Cell, Item) -> () = { _, _ in }

    init(items: [Item]) {
        super.init(style: .plain)
        self.items = items
    }

    required init?(coder aDecoder: NSCoder) {
        fatalError("init(coder:) has not been implemented")
    }

    override func viewDidLoad() {
        super.viewDidLoad()
        tableView.register(Cell.self)
    }

    override func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {
        let item = items[indexPath.row]
        didSelect(item)
    }

    override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        return items.count
    }

    override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) ->
UITableViewCell {
        let item = items[indexPath.row]
        let cell: Cell = tableView.dequeueReusableCell(for: indexPath)
        configure(cell, item)
        return cell
    }
}
```

```swift
final class ItemsViewController<Item, Cell: UITableViewCell>: UITableViewController
  where Cell: Reusable {

    var items: [Item] = []
    var didSelect: (Item) -> () = { _ in }
    var configure: (Cell, Item) -> () = { _, _ in }

    init(items: [Item]) {
        super.init(style: .plain)
        self.items = items
    }

    required init?(coder aDecoder: NSCoder) {
        fatalError("init(coder:) has not been implemented")
    }

    override func viewDidLoad() {
        super.viewDidLoad()
        tableView.register(Cell.self)
    }

    override func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {
        let item = items[indexPath.row]
        didSelect(item)
    }

    override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        return items.count
    }

    override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) ->
UITableViewCell {
        let item = items[indexPath.row]
        let cell: Cell = tableView.dequeueReusableCell(for: indexPath)
        configure(cell, item)
        return cell
    }
}
```

```swift
let meetups = [
    Meetup(number: 12, date: "2018-01-17"),
    Meetup(number: 11, date: "2017-11-16"),
    Meetup(number: 10, date: "2017-10-12"),
    Meetup(number: 9, date: "2017-04-19")
]

let meetupsVC = ItemsViewController<Meetup, MeetupCell>(items: meetups)

meetupsVC.configure =  { cell, item in
    cell.textLabel?.text = "Cocoa.lt #\(item.number)"
    cell.detailTextLabel?.text = item.date
}
```

```
let meetups = [
    Meetup(number: 12, date: "2018-01-17"),
    Meetup(number: 11, date: "2017-11-16"),
    Meetup(number: 10, date: "2017-10-12"),
    Meetup(number: 9, date: "2017-04-19")
]

let meetupsVC = ItemsViewController<Meetup, MeetupCell>(items: meetups)

meetupsVC.configure =  { cell, item in
    cell.textLabel?.text = "Cocoa.lt #\(item.number)"
    cell.detailTextLabel?.text = item.date
}
```

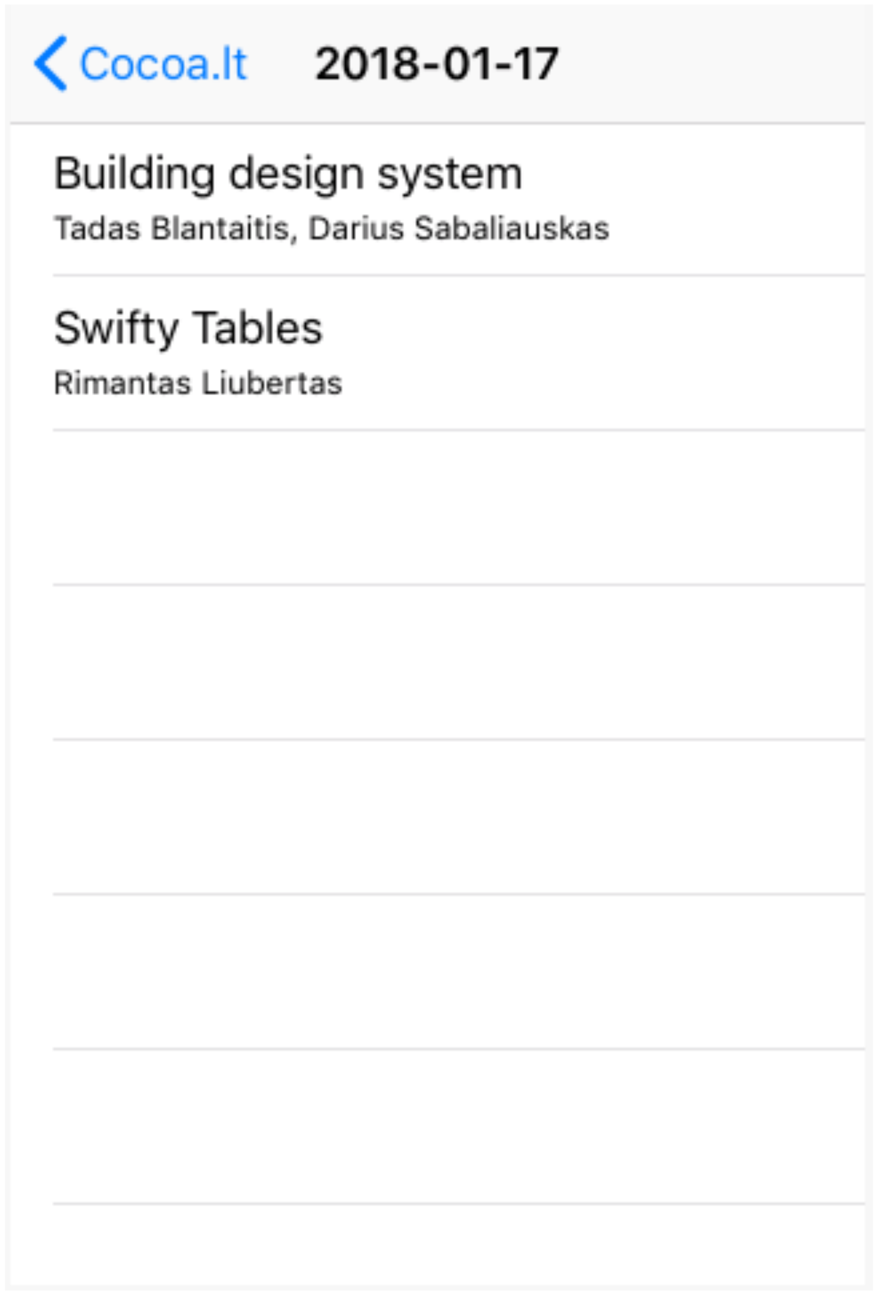


Cocoa.lt

Cocoa.lt #12	2018-01-17
Cocoa.lt #11	2017-11-16
Cocoa.lt #10	2017-10-12
Cocoa.lt #9	2017-04-19

```swift
meetupsVC.didSelect = { meetup in
    let talks = getTalks(at: meetup)
    let talksVC = ItemsViewController<Talk, TalkCell>(items: talks)
    talksVC.title = meetup.date
    talksVC.configure = { cell, item in
        cell.textLabel?.text = item.title
        cell.detailTextLabel?.text = item.speaker
    }
    navigationController?.pushViewController(talksVC, animated: true)
}
```

```
meetupsVC.didSelect = { meetup in
    let talks = getTalks(at: meetup)
    let talksVC = ItemsViewController<Talk, TalkCell>(items: talks)
    talksVC.title = meetup.date
    talksVC.configure = { cell, item in
        cell.textLabel?.text = item.title
        cell.detailTextLabel?.text = item.speaker
    }
    navigationController?.pushViewController(talksVC, animated: true)
}
```



Building design system
Tadas Blantaitis, Darius Sabaliauskas

Swifty Tables
Rimantas Liubertas

```swift
struct Meetup {
    var number: Int
    //    …
}

struct MeetupViewModel {
    let name: String
    //    …
}

extension MeetupViewModel {
    init(_ meetup: Meetup) {
        self.name = "Cocoa.lt #\(meetup.number)"
    }

    func configureCell(_ cell: UITableViewCell) {
        cell.textLabel?.text = name
    }
}
```

```swift
struct Meetup {
    var number: Int
    //    …
}

struct MeetupViewModel {
    let name: String
    //    …
}

extension MeetupViewModel {
    init(_ meetup: Meetup) {
        self.name = "Cocoa.lt #\(meetup.number)"
    }

    func configureCell(_ cell: UITableViewCell) {
        cell.textLabel?.text = name
    }
}

// cellForRowAtIndexPath
item.configureCell(cell)
```

Loading…

```swift
enum Optional<Wrapped> {
    case none
    case some(Wrapped)
}
```

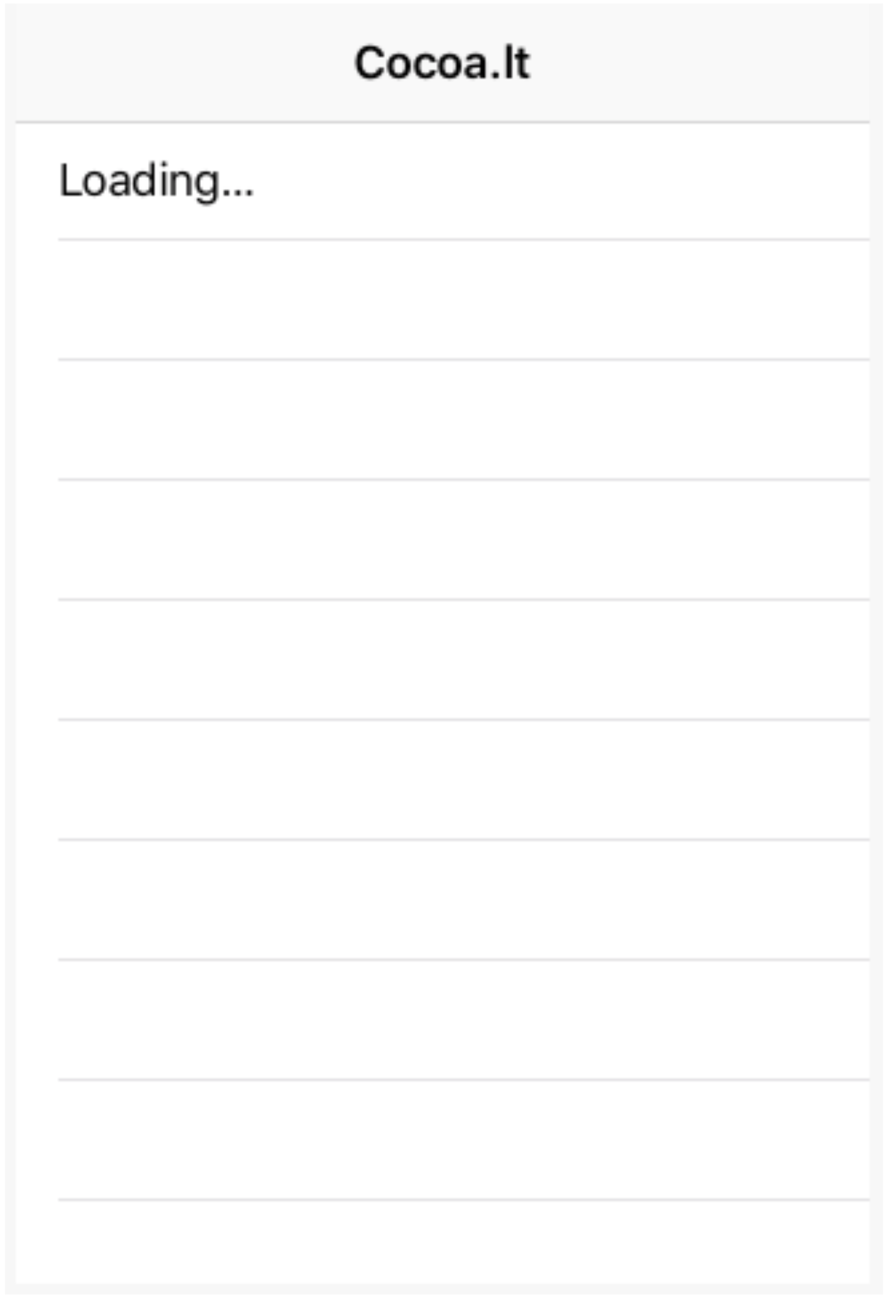
```swift
enum Result<T> {
    case success(T)
    case error(Error)
}
```

```swift
protocol TableAbleData {
    associatedtype TableItem
}

enum TableDataState<T: TableAbleData> {
    case loading
    case failure
    case success([T])
}
```

```swift
extension TableDataState {
    var count: Int {
        switch self {
        case .success(let items):
            return items.count
        default:
            return 1
        }
    }
}
```

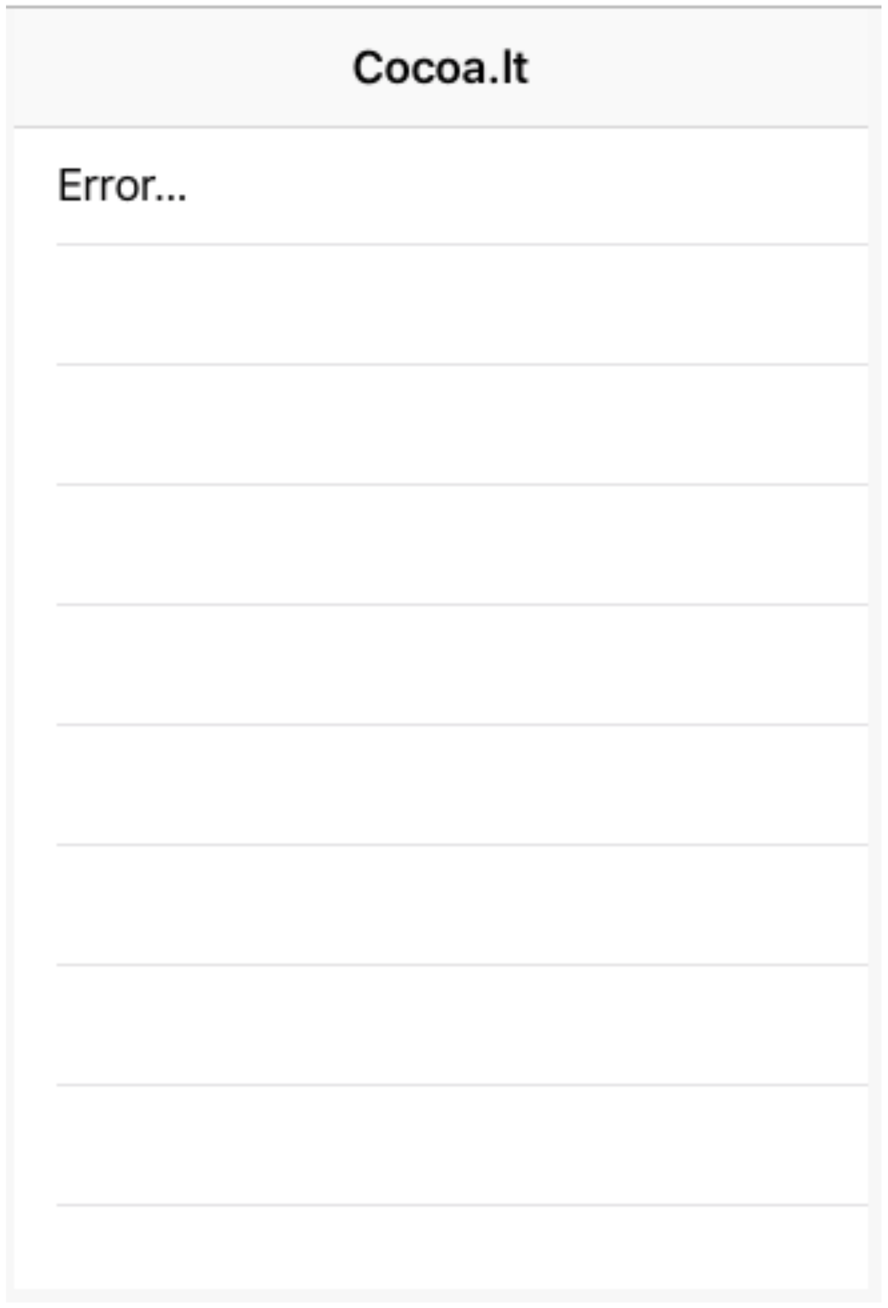
```swift
final class MeetupsViewController: UITableViewController {
    var data: TableDataState<Meetup> {
        didSet {
            tableView.reloadData()
        }
    }


// cellForRowAtindexPath

switch data {
case .success(let meetups):
    let cell = tableView.dequeueReusableCell(for: indexPath) as MeetupCell
    let item = meetups[indexPath.row]
    configure(cell, item)
    return cell

case .loading:
    let cell = tableView.dequeueReusableCell(for: indexPath) as LoadingCell
    return cell

case .failure:
    let cell = tableView.dequeueReusableCell(for: indexPath) as ErrorCell
    return cell
}
```

```
meetupsVC.data = TableDataState.loading
```



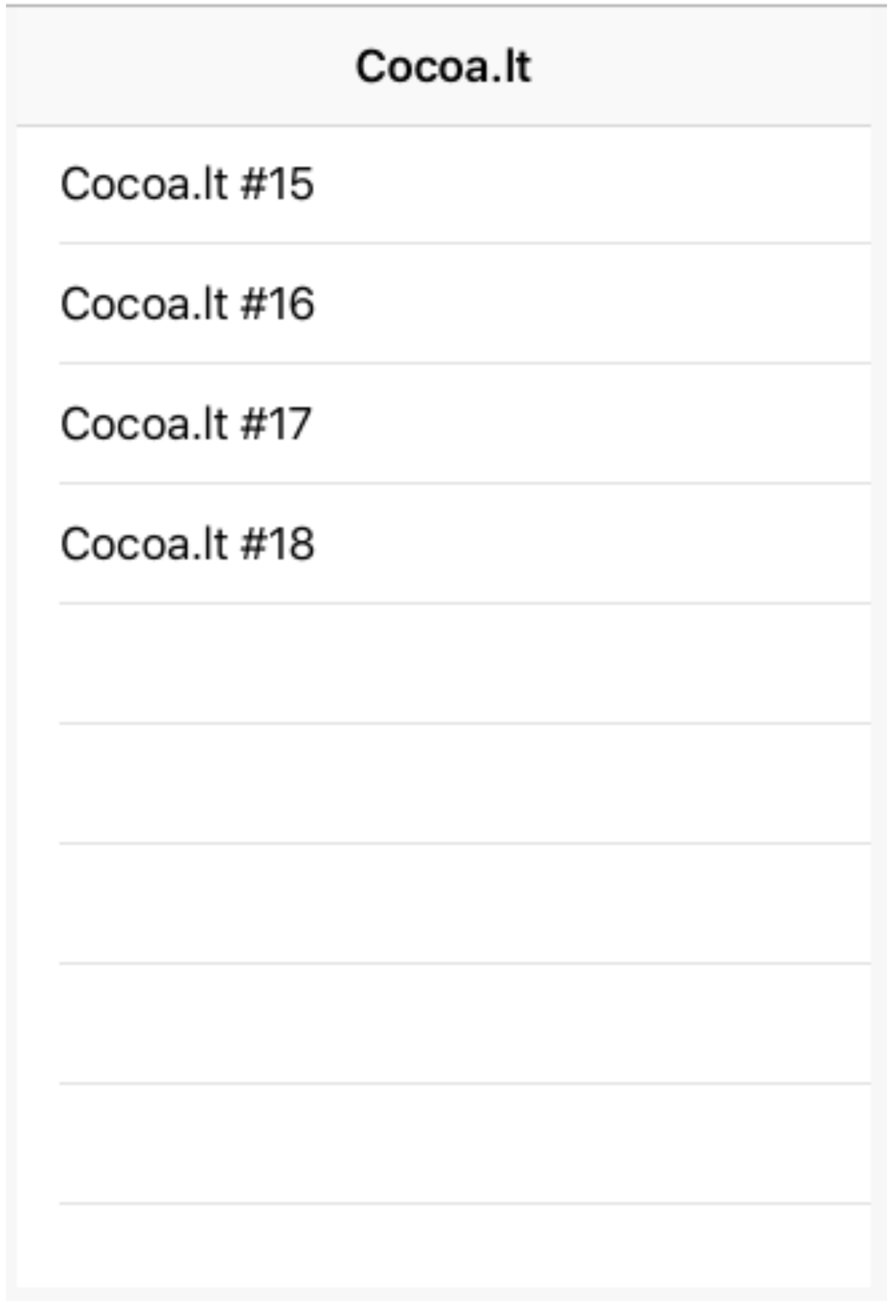
Loading...

```
meetupsVC.data = TableDataState.failure
```


Cocoa.It

Error...

```
meetupsVC.data = TableDataState.success(meetups)
```



Cocoa.lt

Cocoa.lt #15

Cocoa.lt #16

Cocoa.lt #17

Cocoa.lt #18

Swifty Tables

Swifty Tables

- **Reusable views**

Swifty Tables

- **Reusable views**
- **Generic UITableViewController**

Swifty Tables

- Reusable views
- Generic UITableViewController
- **Loading state**

Swifty Tables

- Reusable views
- Generic UITableViewController
- Loading state
- **Ačiū**