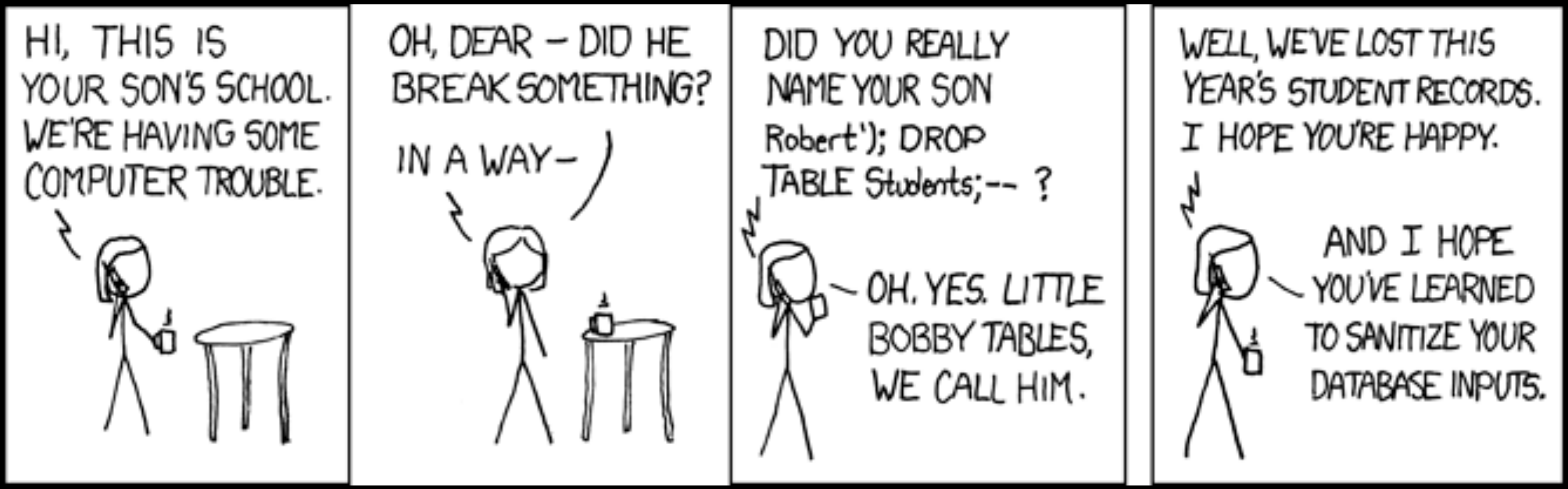
Protocol-Oriented UI

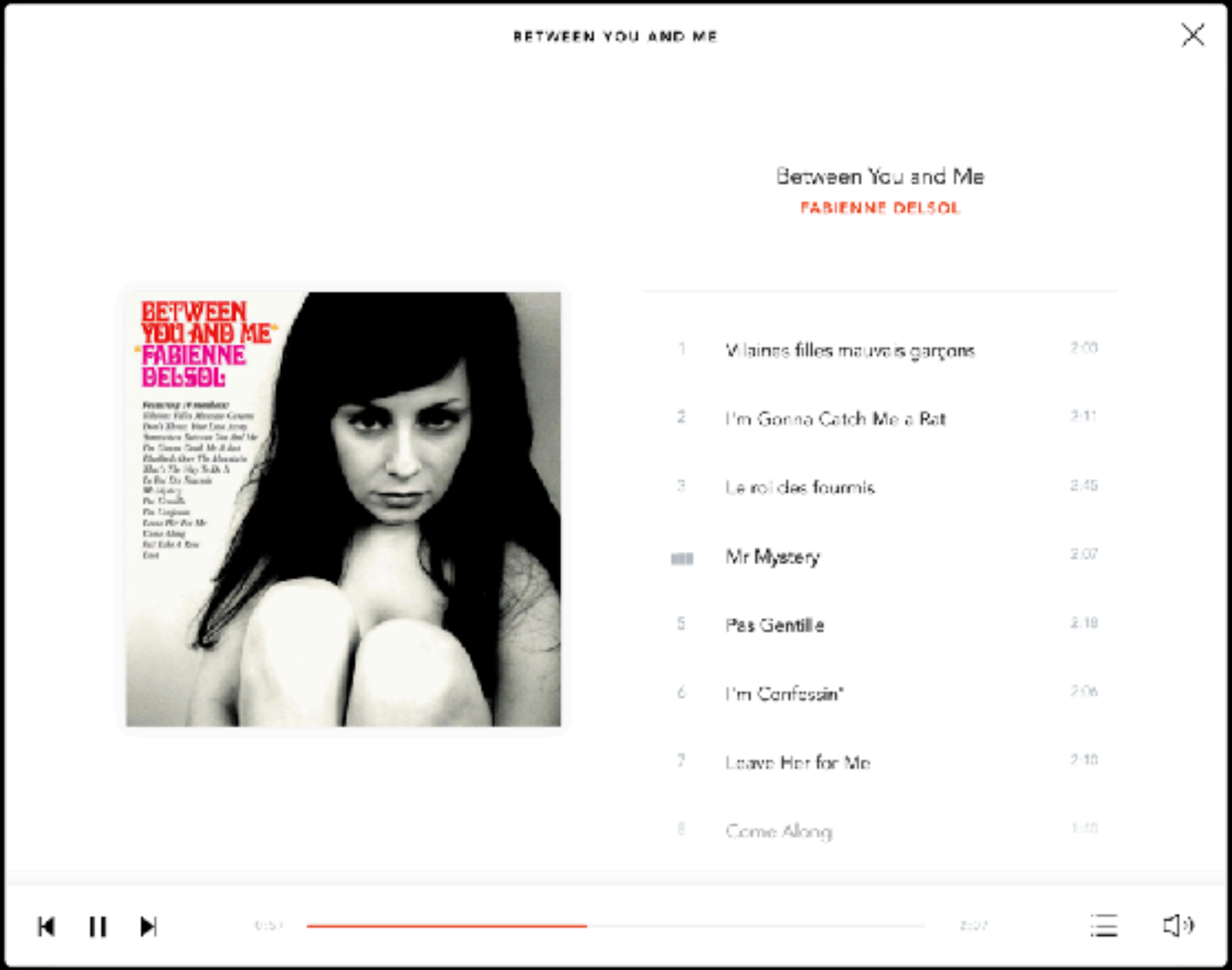
Bobby Bobak

@bobek_balinek

Bobby ~~Bobak~~ Tables

@bobek_balinek


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.



Between You and Me
FABIENNE DELSOL

1	Vilaines filles mauvais garçons	2:00
2	I'm Gonna Catch Me a Rat	2:11
3	Le roi des fourmis	2:45
▮▮▮	Mr Mystery	2:07
5	Pas Gentille	2:18
6	I'm Confessin'	2:06
7	Leave Her for Me	2:10
8	Come Along	1:40

Voltra

Music player for music collectors

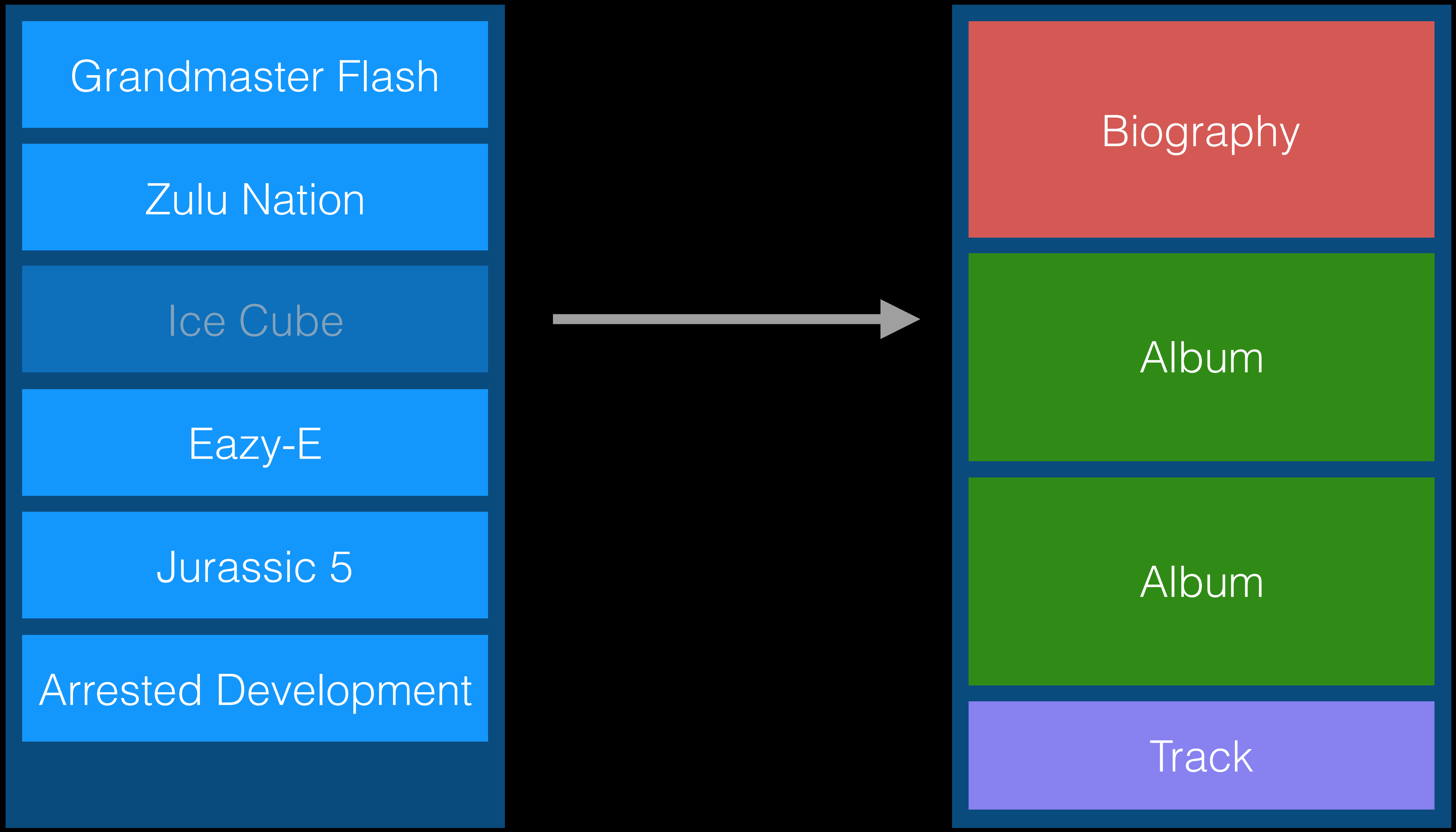
Fresh start in Swift

- MVVM > MVC
- Value Types & Generics
- ~~Massive View Controllers~~
- Make testing a delight

“Code to the interface you wish you had, not the interface you were given”

– Stephen Celis

TableViewController



TableViewController

- `numberOfSections`
- `numberOfRowsInSection section: Int`
- `cellForRowAt indexPath: IndexPath`
- `heightForRowAt indexPath: IndexPath`
- `didSelectRowAt indexPath: IndexPath`

One UITableViewCell

```
let cell = dequeueReusableCell(withIdentifier:
“ArtistRow”, for: indexPath) as! ArtistRowCell

// Cell config…

cell.artistLabel.text = artist.name

return cell
```

Multiple UITableViewCells

```
if viewModel is TrackViewModel {

let cell = dequeueReusableCell(withIdentifier: "TrackRowCell",
for: indexPath) as! TrackRowCell

// Cell Config…

}

if viewModel is BioViewModel {

let cell = dequeueReusableCell(withIdentifier: "BioRowCell",
for: indexPath) as! BioRowCell

// Cell Config…

}
```

More UITableViewCells

```
if viewModel is TrackViewModel {

let cell = dequeueReusableCell(withIdentifier: "TrackRowCell", for: indexPath) as! TrackRowCell

// Cell Config…

}

if viewModel is BioViewModel {

let cell = dequeueReusableCell(withIdentifier: "BioRowCell", for: indexPath) as! BioRowCell

// Cell Config…

}

if viewModel is AlbumViewModel {

let cell = dequeueReusableCell(withIdentifier: "AlbumRowCell", for: indexPath) as! AlbumRowCell

// Cell Config…

}
```

Even More UITableViewCells

```
if viewModel is TrackViewModel {

let cell = dequeueReusableCell(withIdentifier: "TrackRowCell", for: indexPath) as! TrackRowCell

// Cell Config…

}

if viewModel is BioViewModel {

let cell = dequeueReusableCell(withIdentifier: "BioRowCell", for: indexPath) as! BioRowCell

// Cell Config…

}

if viewModel is AlbumViewModel {

let cell = dequeueReusableCell(withIdentifier: "AlbumRowCell", for: indexPath) as! AlbumRowCell

// Cell Config…

}

if viewModel is ArtistViewModel {

let cell = dequeueReusableCell(withIdentifier: "ArtistRowCell", for: indexPath) as! ArtistRowCell

// Cell Config…

}
```

Resize UITableViewCell

```
if viewModel is TrackViewModel {
  return 120
}
if viewModel is BioViewModel {
  return tableView.frame.width / 2
}
```

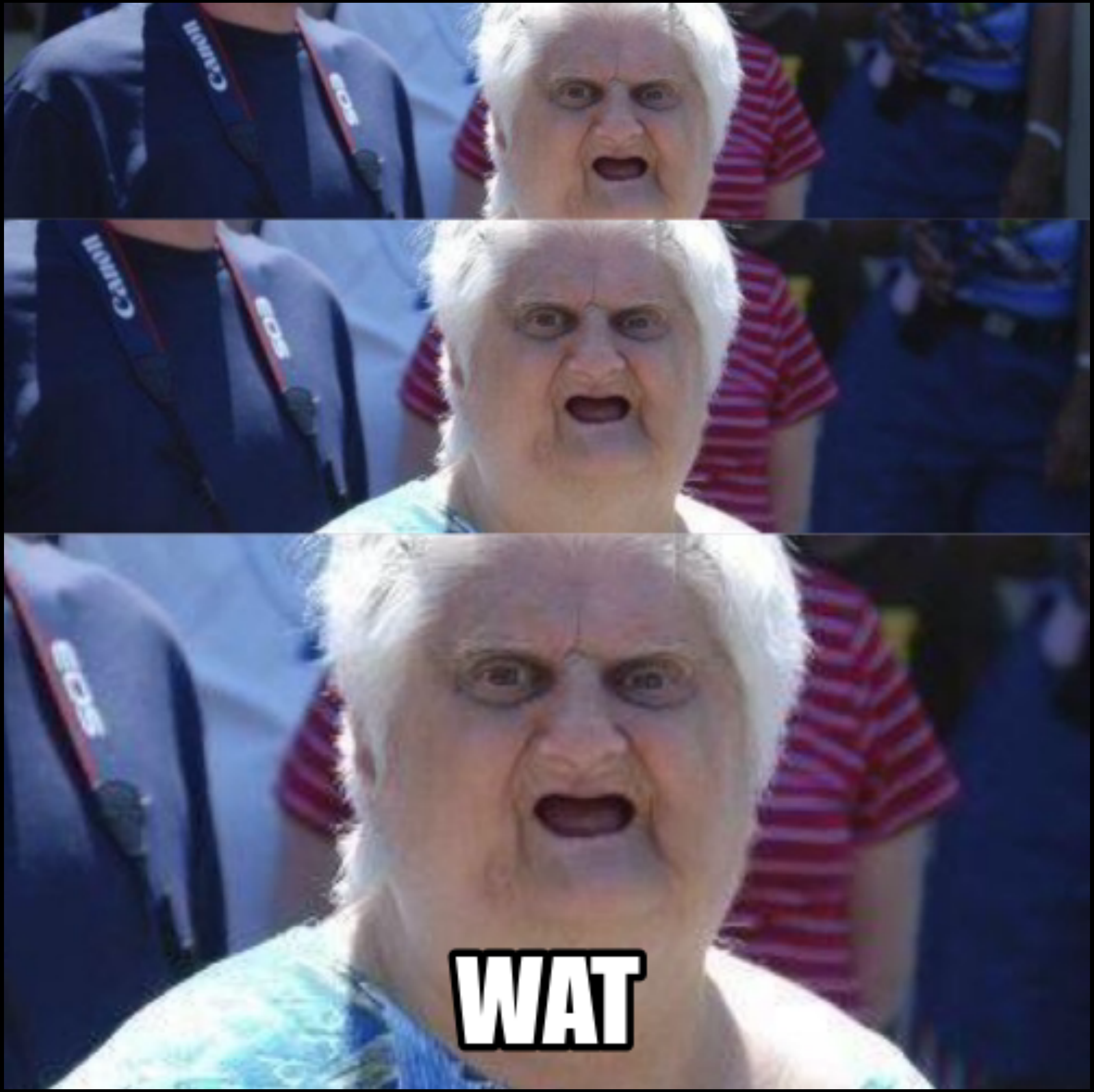
Resize UITableViewCell

```
if viewModel is TrackViewModel {
  return 120
}
if viewModel is BioViewModel {
  return tableView.frame.width / 2
}
if viewModel is AlbumViewModel {
  return 320
}
```

Increasing Complexity

```
if viewModel is TrackViewModel {

let cell = dequeueReusableCell(withIdentifier: "TrackRowCell", for: indexPath) as! TrackRowCell

// Cell Config…

}

if viewModel is BioViewModel {

let cell = dequeueReusableCell(withIdentifier: "BioRowCell", for: indexPath) as! BioRowCell

// Cell Config…

}

if viewModel is AlbumViewModel {

let cell = dequeueReusableCell(withIdentifier: "AlbumRowCell", for: indexPath) as! AlbumRowCell

// Cell Config…

}

if viewModel is TrackViewModel {

  return 120

}

if viewModel is BioViewModel {

  return tableView.frame.width / 2

}

if viewModel is AlbumViewModel {

  return 320

}
```

```
if viewModel is TrackViewModel {

let cell = dequeueReusableCell(withIdentifier: "TrackRowCell", for: indexPath) as! TrackRowCell

// Cell Config…

}

if viewModel is BioViewModel {

let cell = dequeueReusableCell(withIdentifier: "BioRowCell", for: indexPath) as! BioRowCell

// Cell Config…

}

if viewModel is AlbumViewModel {

let cell = dequeueReusableCell(withIdentifier: "AlbumRowCell", for: indexPath) as! AlbumRowCell

// Cell Config…

}

if viewModel is TrackViewModel {

  return 120

}

if viewModel is BioViewModel {

  return tableView.frame.width / 2

}

if viewModel is AlbumViewModel {

  return 320

}
```


Canon
EOS
WAT

Recipe for MassiveViewController

- Dequeue Bio, Album and Track cells
- Configure with view models
- Specify different sizes
- Load from NIBs

Protocols

to the rescue!

Quick overview

```swift
protocol Drawable {
  func draw(renderer: Renderer)
}
struct Circle: Drawable {}

extension Drawable {
  func size(in rect: CGRect) -> CGSize {
  }
}
```

Composition > Inheritance

More on protocols

Protocol-Oriented Programming in Swift

WWDC 2015 - Session #408

Applying protocols to TableViews

- Dequeue Bio, Album and Track cells
- Configure with view models
- Specify different sizes
- Load from NIBs

ViewModel Protocol

```swift
protocol ViewModel {
  let reuseIdentifier: String { get set }
}

struct ArtistViewModel: ViewModel {
  let reuseIdentifier: String = CustomCell.reuseIdentifier
  let name: String
}
```

Configurable Protocol

```
protocol Configurable {
  associatedtype DataModel
  func configure(with: DataModel)
}

extension CustomCell: Configurable {
  typealias DataModel = ArtistViewModel
  func configure(with: ArtistViewModel) {
    // Configure labels with data…
  }
}
```

Extend other custom cells

```
extension TrackRowCell: Configurable {
  typealias DataModel = TrackViewModel
  func configured(with: TrackViewModel) -> Self {
    // Configure labels with data and return itself…
  }
}

extension BioRowCell: Configurable {
  // …
}
extension AlbumRowCell: Configurable {
  // …
}
```

ArtistViewController

```
func cellForRowAt indexPath: IndexPath -> UITableViewCell

if let cell = tableView.dequeueReusableCell(
    withIdentifier: viewModel.reuseIdentifier) as? Configurable
{
  return cell.configured(with: viewModel) as! UITableViewCell
}
```

Sized Protocol

```
protocol Sized {
  func preferredSize(in frame: CGRect) -> CGSize
}

struct ArtistViewModel: ViewModel, Sized {
  // …

  func preferredSize(in frame: CGRect) -> CGSize {
    return CGSize(width: frame.width, height: 150)
  }
}
```

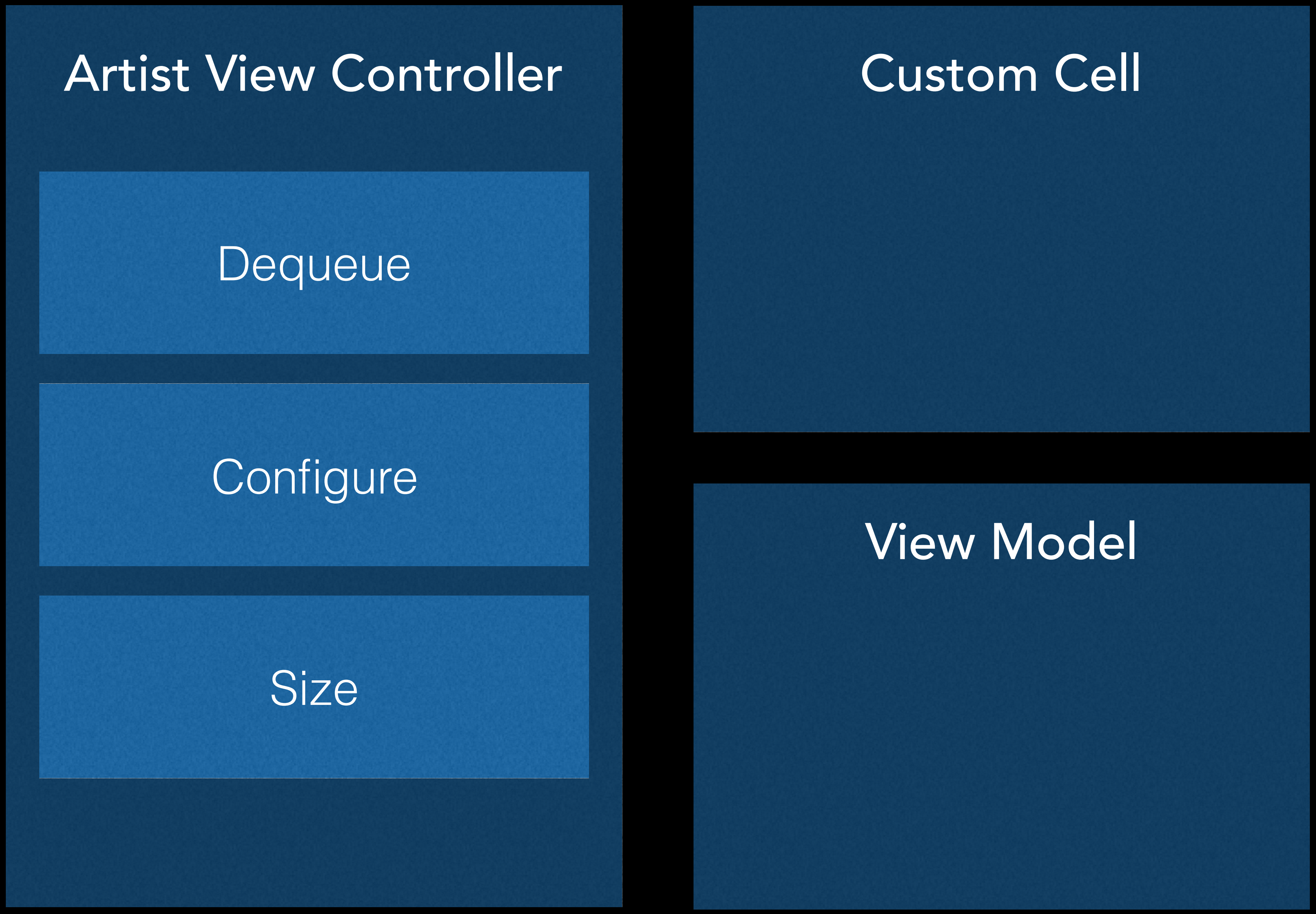
ArtistViewController

```
func heightForRowAt indexPath: IndexPath -> CGFloat

if let viewModel = item(at: indexPath) as? Sizeable {
  return viewModel.preferredSize(in: tableView.frame).height
}
return tableView.rowHeight
```

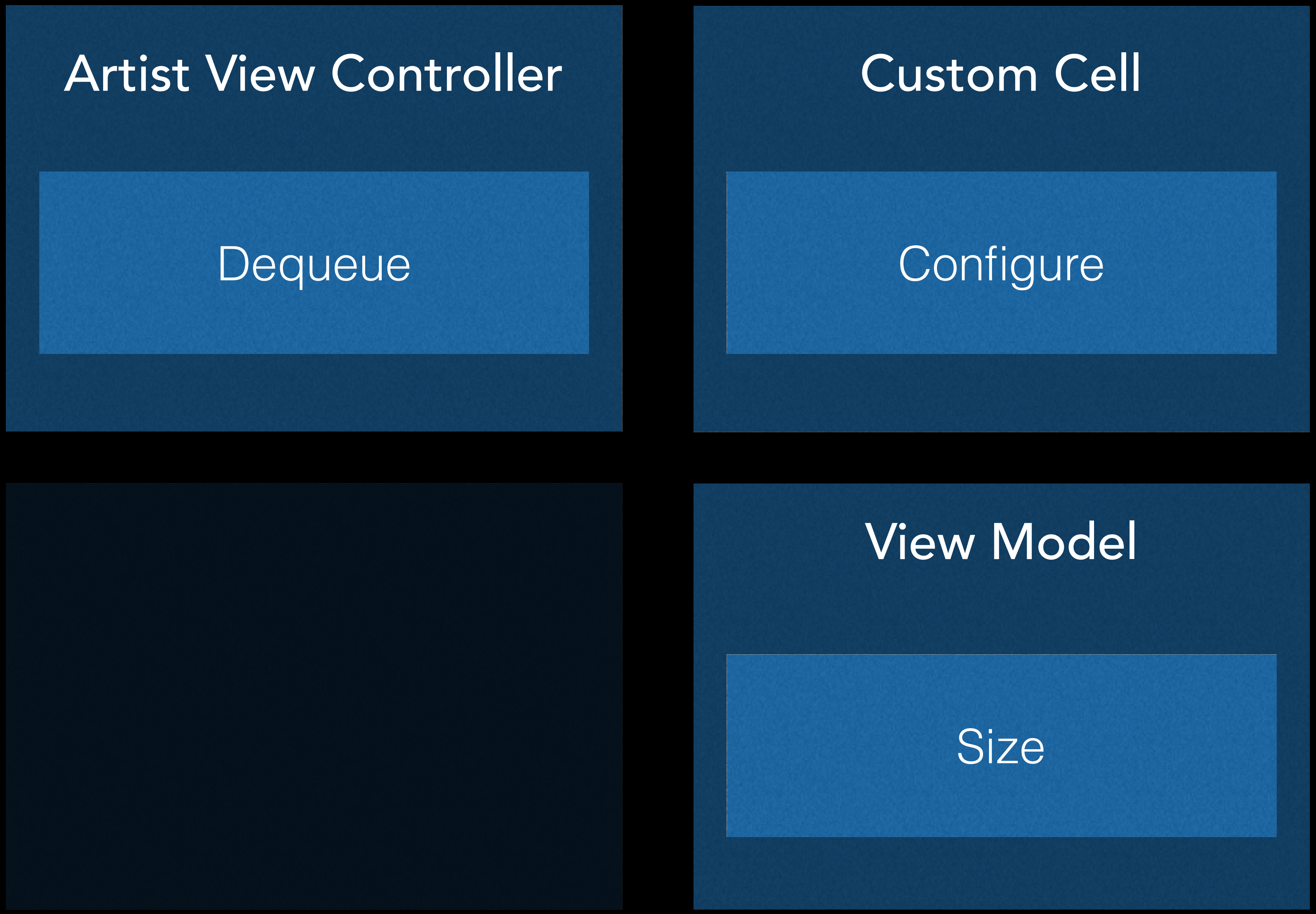
Abstracting more

```
protocol Selectable {}

extension ViewModel: Selectable {}



func shouldHighlightRowAt indexPath: IndexPath -> Bool {
    return viewModel is Selectable
}
```

Shifting responsibilities



Shifting responsibilities



Abstracting even more

```
tableView.register(BioCellView.self)
tableView.register(AlbumCellView.self)
tableView.register(TrackCellView.self)
```

viewDidLoad

```
if let cell = dequeueCell(in: tableView with:
viewModel) as? Configurable {
  return cell.configured(with: viewModel)
}
```

cellForRowAt

```
viewModel.size(in: tableView.frame).height
```

heightForRowAt

“Code to the interface you wish you had, not the interface you were given”

– Stephen Celis

Thank you!

github.com/bobek-balinek/nsmanchester-slides