Views: complexity and reusability

Experiments comparing SwiftUI and UIKit

$ whomami

- Felipe Espinoza
- iOS Dev @ FINN.no
- @fespinoza on github
- @fespinozacast on twitter

What do I mean by complexity?

Creating views in `UIKit`

- initialize
- position / size[1]
- styling
- passing data
- update to changes

---

[1] size may depend on content

```swift
class SampleViewController: UIViewController {
    override func viewDidLoad() {
        super.viewDidLoad()
        view.backgroundColor = .red

        let label = UILabel()
        label.translatesAutoresizingMaskIntoConstraints = false
        label.text = "Hello"

        let button = UIButton()
        button.setTitle("Hello Who?", for: .normal)
        button.translatesAutoresizingMaskIntoConstraints = false
        button.addTarget(self, action: #selector(tap), for: .touchUpInside)

        self.view.addSubview(label)
        self.view.addSubview(button)

        NSLayoutConstraint.activate([
            label.topAnchor.constraint(equalTo: self.view.topAnchor, constant: 30),
            label.centerXAnchor.constraint(equalTo: self.view.centerXAnchor),

            button.topAnchor.constraint(equalTo: label.bottomAnchor, constant: 30),
            button.centerXAnchor.constraint(equalTo: self.view.centerXAnchor),
        ])
    }
}
```

FinniversKit

- FINN's UI component library
- it's open source and it's awesome :D
- FinniversKit on Github

iPhone 11 Pro — 13.3

iPhone 11 Pro — 13.3

BetaFeatureView	**SettingDetailView**
BetaFeatureViewModel	SettingDetailViewModel
BetaFeatureViewDelegate	SettingDetailViewDelegate

- similar layout
  - they differ in data requirements and order of elements
- delegates to bubble up touches on buttons
- tons of similar non-shareable boilerplate

Case 1: Object page & KeyValueGridView

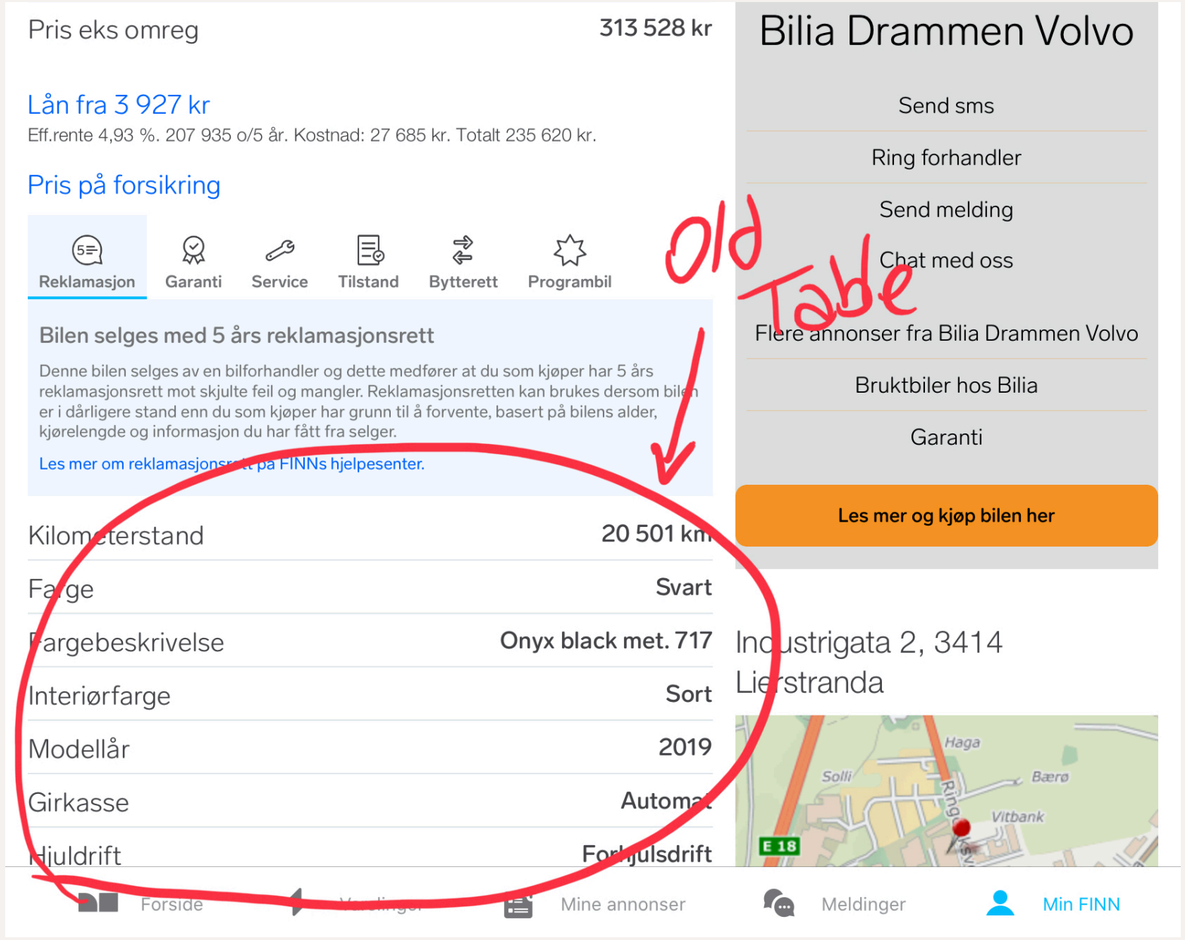
The object page

bilia | Drammen

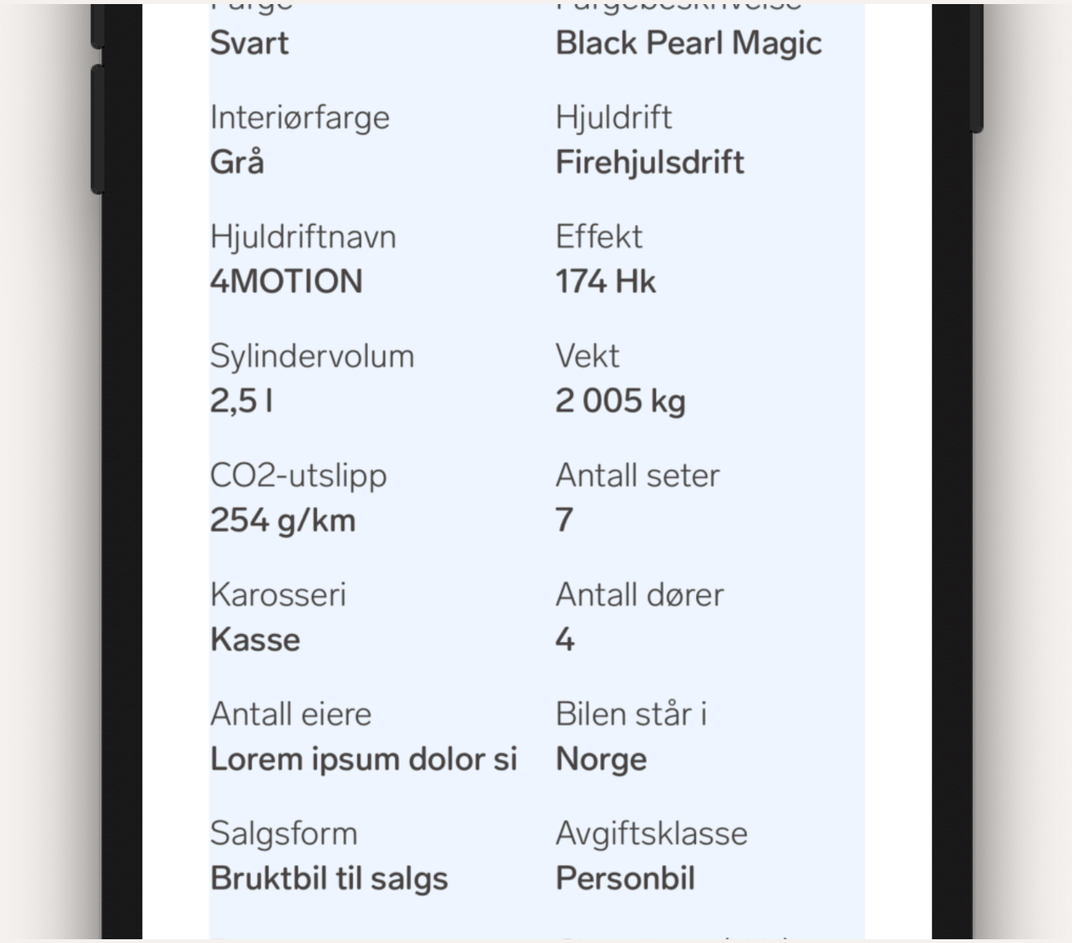
Bilde 1 (1 / 29)

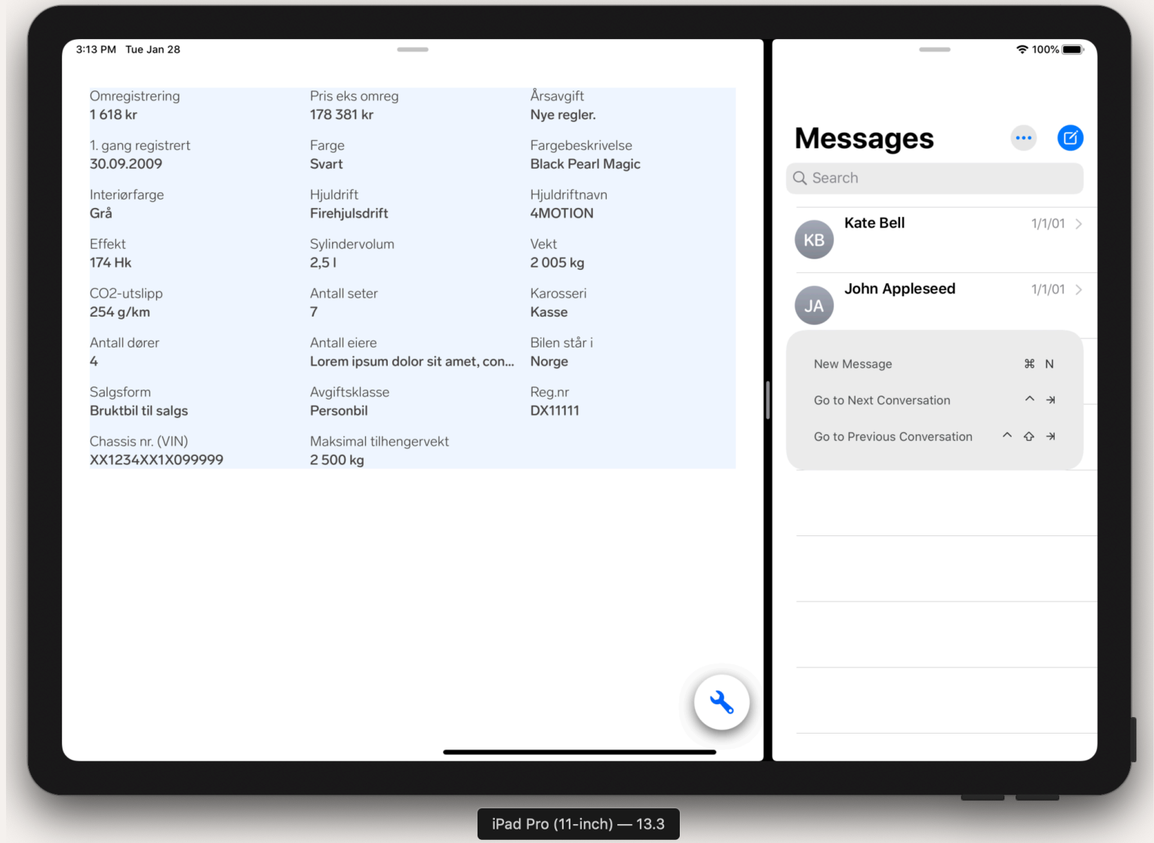
Volvo V40 Cross Country

Example: Object page & `KeyValueGridView`



Example: Object page & `KeyValueGridView`





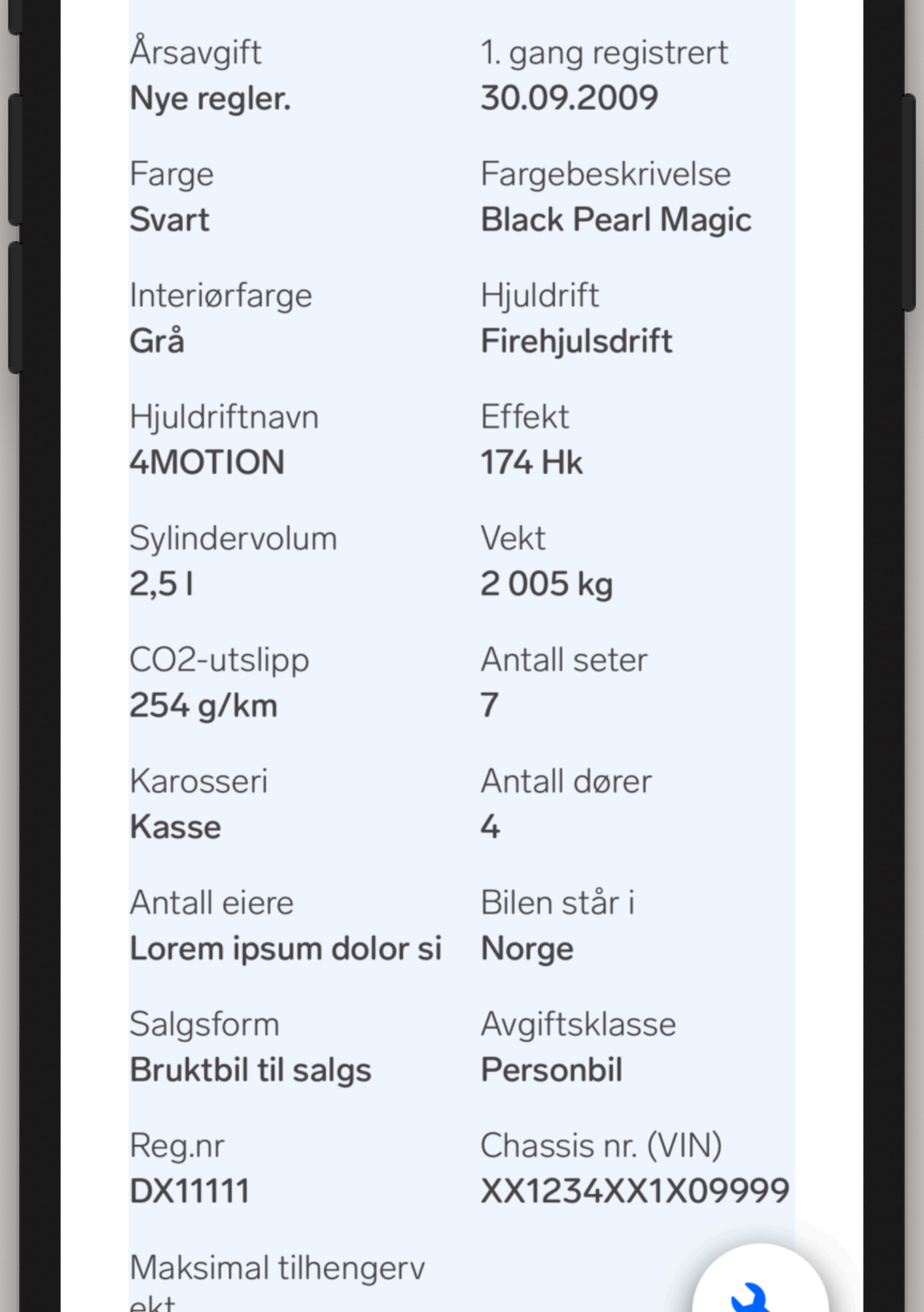
UICollectionView

Understanding
`UICollectionViewLayout`

To achieve the desired layout:

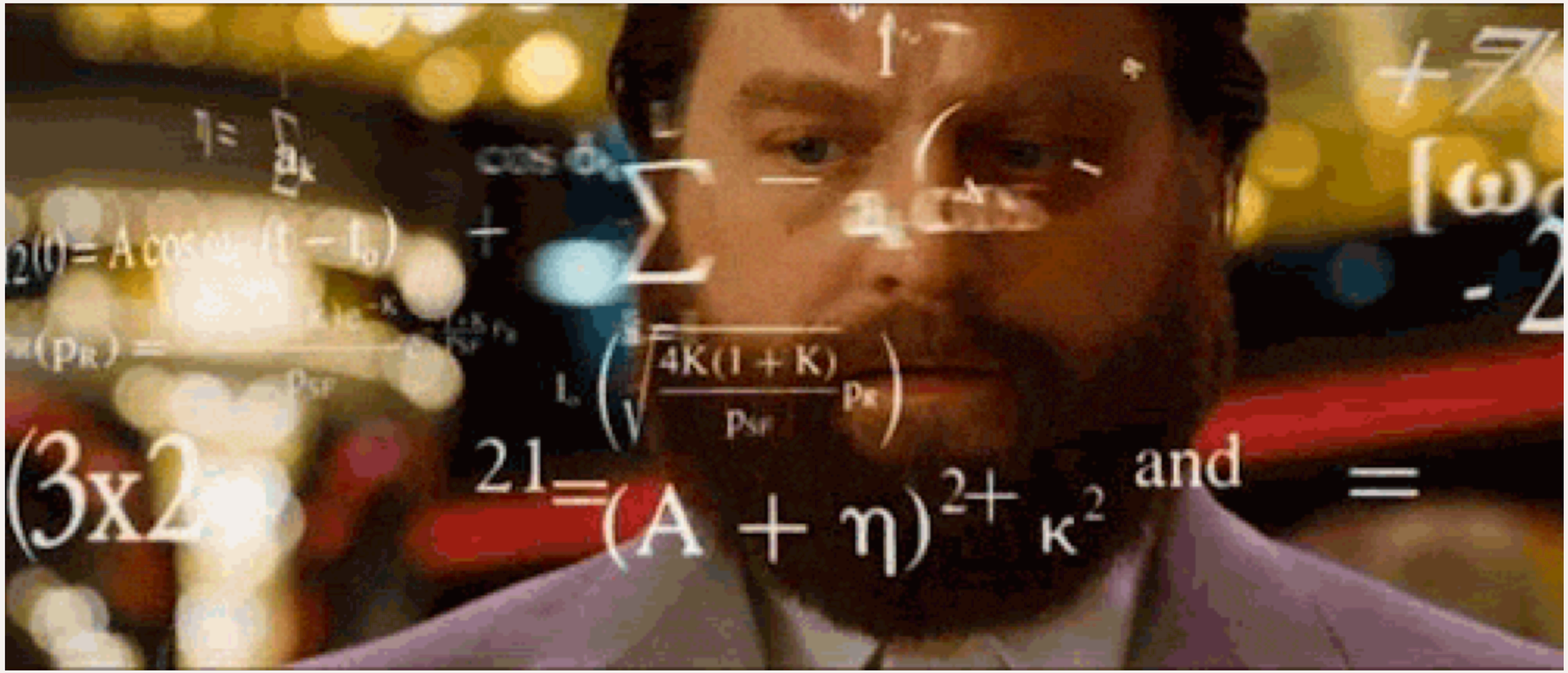
- Do i use flow layout?
- Do i subclass it? or the normal view layout?
- Do i use the delegate methods?

I need to make the cells's height to adjust to the content



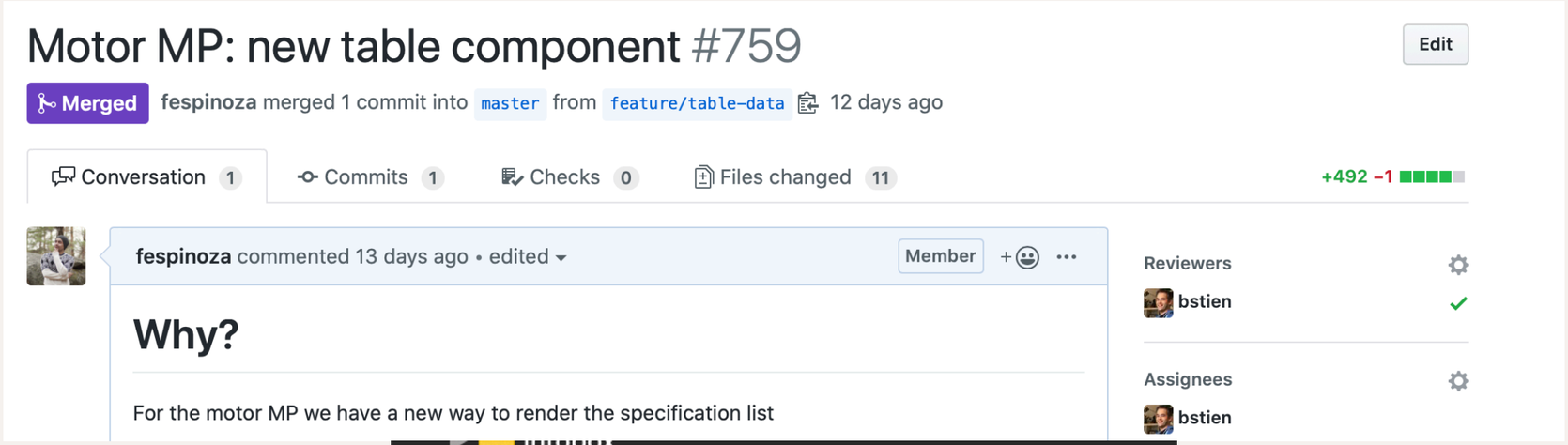
Understanding UICollectionViewLayout

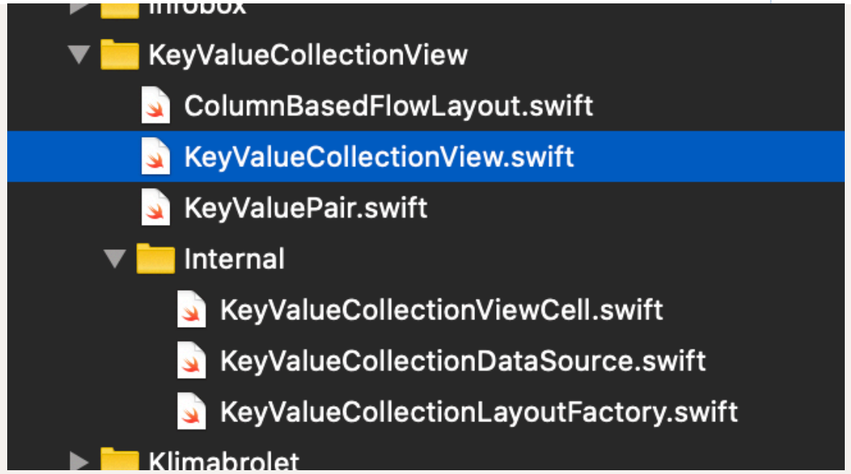
```
override func prepare() {
 super.prepare()

 itemAttributes = [UICollectionViewLayoutAttributes]()

 guard let collectionView = collectionView else {
     return
 }

 let columnsRange = 0 ..< configuration.numberOfColumns

 var columns = columnsRange.map { _ in 0 }
 var attributesCollection = [UICollectionViewLayoutAttributes]()
 var yOffset = configuration.topOffset

 if let height = delegate.adsGridViewLayout(self, heightForHeaderViewInCollectionView: collectionView) {
     let attributes = UICollectionViewLayoutAttributes(forSupplementaryViewOfKind: UICollectionView.elementKindSectionHeader, with: IndexPath(item: 0, section: 0))
     attributes.frame = CGRect(x: 0, y: 0, width: collectionView.frame.size.width, height: height)
     attributesCollection.append(attributes)

     yOffset += height
 }

 for index in 0 ..< numberOfItems {
     let columnIndex = indexOfLowestValue(in: columns)

     let xOffset = xOffsetForItemInColumn(itemWidth: itemWidth, columnIndex: columnIndex)
     let topPadding = configuration.numberOfColumns > index ? yOffset : 0.0
     let verticalOffset = CGFloat(columns[columnIndex]) + topPadding

     let indexPath = IndexPath(item: index, section: 0)
     let itemHeight = delegate.adsGridViewLayout(self, heightForItemWithWidth: itemWidth, at: indexPath)

     columns[columnIndex] = Int(verticalOffset + itemHeight + configuration.columnSpacing)

     let attributes = UICollectionViewLayoutAttributes(forCellWith: indexPath)
     attributes.frame = CGRect(x: xOffset, y: verticalOffset, width: itemWidth, height: itemHeight)
     attributesCollection.append(attributes)
 }

 itemAttributes.append(contentsOf: attributesCollection)
}
```


21=(A + η)²+ κ² and =

It shows clearly the "*How*", not the "*What*"

First attempt: `UICollectionView`





But then...

- The object page was a collection view
- The cells need to calculate their height for the given width
- My new component is supposed to be inside one of this cells
- The height of the new table component was calculated asynchronously

Challenges when using `UICollectionView`

- self-sizing content
- asynchronous calculation of height

$$collectionView(collectionView) = complexity^2$$

Challenges when using `UICollectionView`

- it's hacking time!

Or...

Second Attempt: `UIStackView`

Refactor: key value grid view #765

Edit

Merged **fespinoza** merged 6 commits into `master` from `refactor/key-value-collection-view` 7 days ago

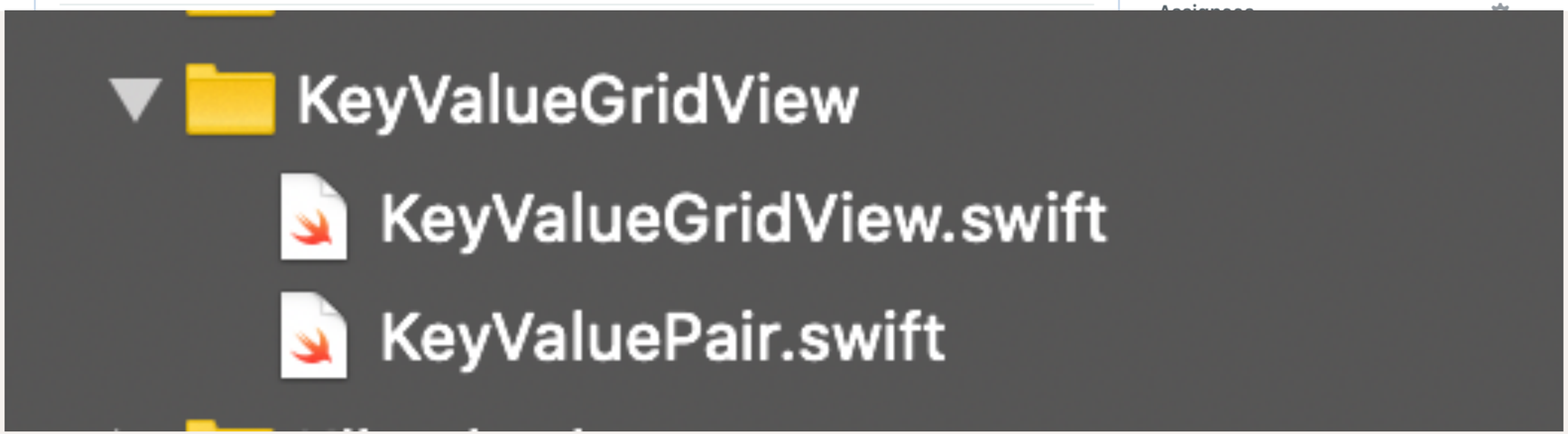
Conversation 3 Commits 6 Checks 0 Files changed 15 +157 −410

fespinoza commented 10 days ago Member

Why?

Reviewers

bstien



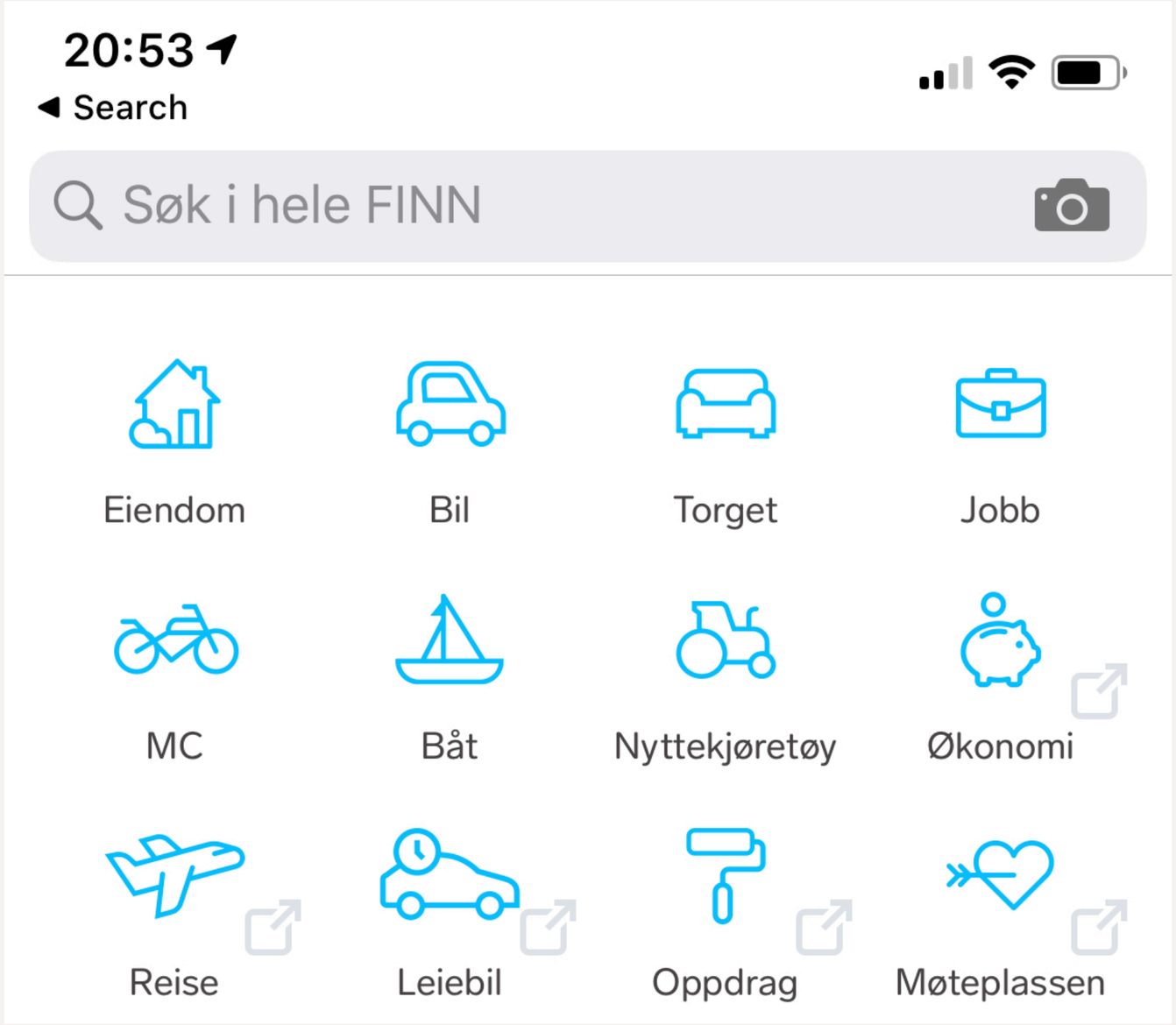
Second Attempt: `UIStackView`

- The height of the component itself will be available synchronously when passing the data.
- The integration to the object page doesn't require any hacks and it's easier.

When not to use `UICollectionView`

- No cell reuse
- No dynamic data

Case 2: SwiftUI


20:53
Search
Søk i hele FINN
Eiendom
Bil
Torget
Jobb
MC
Båt
Nyttekjøretøy
Økonomi
Reise
Leiebil
Oppdrag
Møteplassen

Tilbake

Solgt

Linksys EA7500 trådløs router AC1900

500 kr

Send melding

Svarer vanligvis innen ti minutter

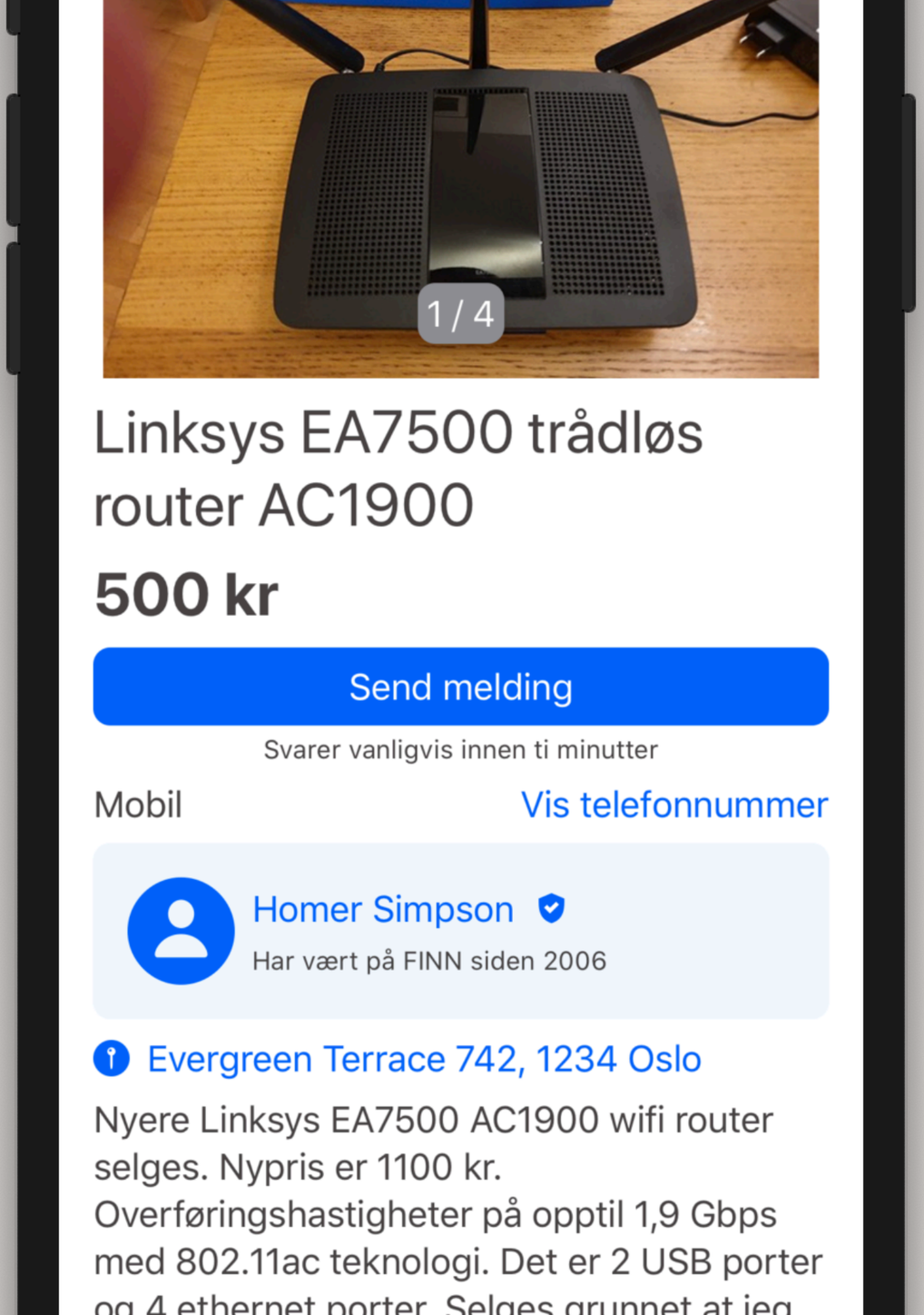
Mobil Vis telefonnummer

Har vært på FINN siden 2006

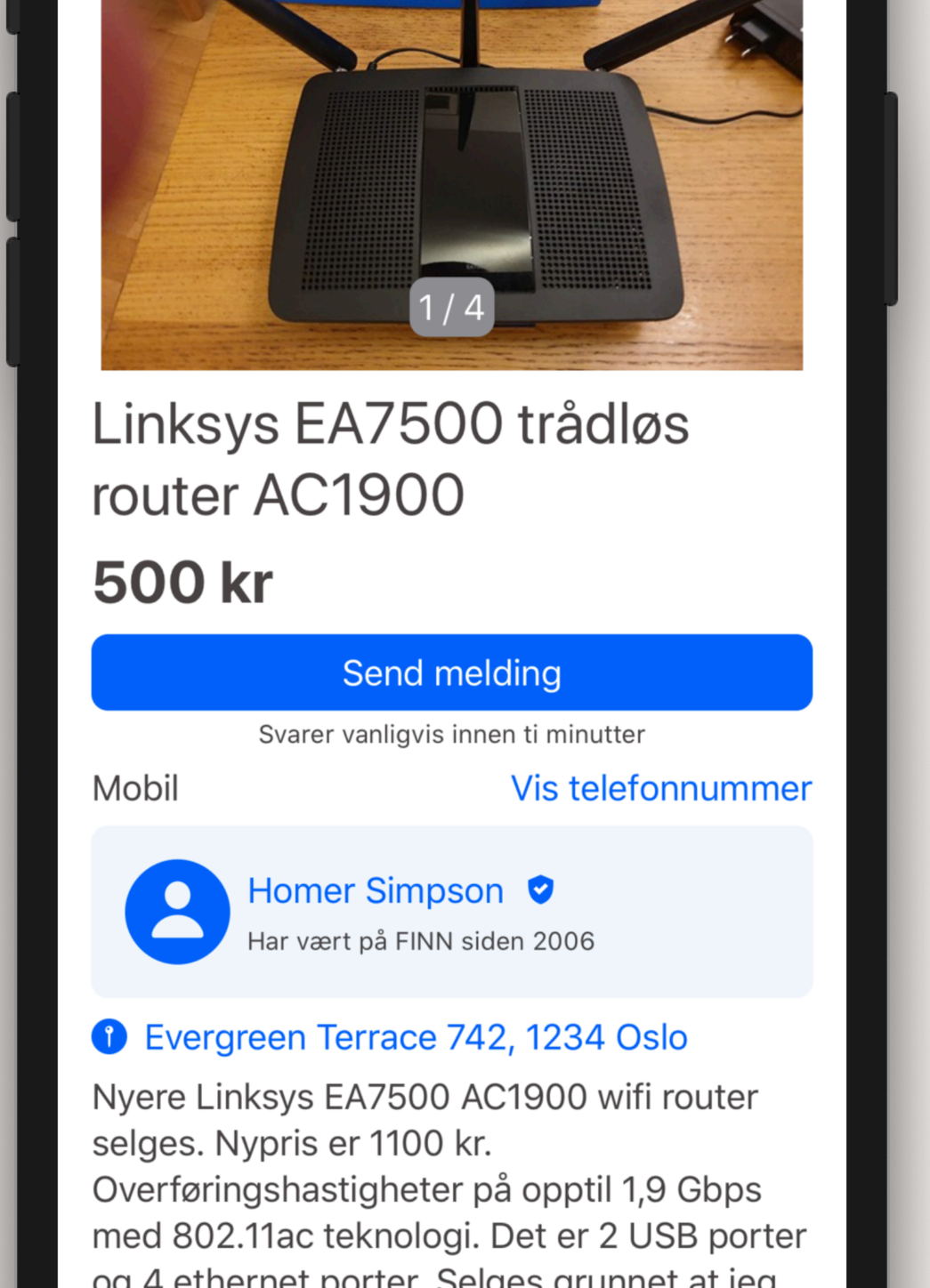
Sjekk reiserute med ENTUR

Nyere Linksys EA7500 AC1900 wifi router selges. Nypris er 1100 kr. Overføringshastigheter på opptil 1,9 Gbps med 802.11ac teknologi. Det er 2 USB porter og 4 ethernet porter. Selges grunnet at jeg

```swift
import SwiftUI

struct AdView: View {
    let adViewModel: AdViewModel
    let imageDownloader: CollectionImageDownloader

    var body: some View {
        ScrollView {
            VStack(alignment: .leading, spacing: .spacingM) {
                imageGalleryView

                Group {
                    titleView
                    priceView
                    sendMessageButton
                    phoneNumber
                    profileView
                    addressView
                    descriptionView
                    conditionView
                }.padding(.horizontal)

                similarAdsView
            }
        }
    }
}
```



```swift
extension AdView {
    var titleView: some View {
        Text(adViewModel.title).title()
    }

    var priceView: some View {
        Text(adViewModel.price).titleStrong()
    }

    var sendMessageButton: some View {
        VStack(alignment: .center, spacing: .spacingS) {
            FINCTAButton {
                Text("Send melding")
            }
            Text(adViewModel.onwerAverageResponseTime).caption()
        }
    }

    var imageGalleryView: some View {
        GalleryView(imageDownloader: imageDownloader)
    }
}
```



8:56 PM Tue Feb 11

100%

Linksys EA7500 trådløs router AC1900

500 kr

LUSETJERNVEIEN 96, 1253 Oslo

Nyere Linksys EA7500 AC1900 wifi router selges. Nypris er 1100 kr. Overføringshastigheter på opptil 1,9 Gbps med 802.11ac teknologi. Det er 2 USB porter og 4 ethernet porter. Selges grunnet at jeg fikk ny trådløs router gjennom Get. Selges med original emballasje. Anmeldelse av router kan leses her: https://www.dinside.no/data/linksys-ea7500-med-mu-mimo-teknologi/60966544

Tilstand Brukt

Elektronikk og hvitevarer > Data > Datatilbehør >

Pris på lån

Rapporter svindel/regelbrudd

FINN-kode 163137299

Sist endret 20. jan. 2020, 13:36

Homer Simpson

Har vært på FINN siden 2006

Send melding

Svarer vanligvis innen ti minutter

Mobil Vis telefonnummer

Lignende annonser

Diverse Philips Hue utstyr selges (E27 pærer, brytere)

Oslo

Google Nest Hello

Oslo

Philips hue startsett med 2 ekstra GU10 spotter og 1 stk E27 pære

Oslo

iPad Pro (11-inch) — 13.3

Conditional Layout

```
struct AdView: View {
    let adViewModel: AdViewModel
    var imageDownloader: CollectionImageDownloader {
        CollectionImageDownloader(adViewModel: adViewModel)
    }

    @Environment(\EnvironmentValues.horizontalSizeClass) var horizontalSizeClass

    var body: some View {
        ScrollView {
            if horizontalSizeClass == UserInterfaceSizeClass.compact {
                compactLayout
            } else {
                regularLayout
            }
        }
    }
}
```

Conditional Layout

```
struct AdView: View {
    let adViewModel: AdViewModel
    var imageDownloader: CollectionImageDownloader {
        CollectionImageDownloader(adViewModel: adViewModel)
    }

    @Environment(\EnvironmentValues.horizontalSizeClass) var horizontalSizeClass

    var body: some View {
        ScrollView {
            if horizontalSizeClass == UserInterfaceSizeClass.compact {
                compactLayout
            } else {
                regularLayout
            }
        }
    }
}
```

Conditional Layout

```
struct AdView: View {
    let adViewModel: AdViewModel
    var imageDownloader: CollectionImageDownloader {
        CollectionImageDownloader(adViewModel: adViewModel)
    }

    @Environment(\EnvironmentValues.horizontalSizeClass) var horizontalSizeClass

    var body: some View {
        ScrollView {
            if horizontalSizeClass == UserInterfaceSizeClass.compact {
                compactLayout
            } else {
                regularLayout
            }
        }
    }
}
```

Conditional Layout

```
extension AdView {
    var compactLayout: some View {
        VStack(alignment: .leading, spacing: .spacingM) {
            imageGalleryView

            Group {
                titleView
                priceView
                sendMessageButton
                phoneNumber
                // same as before..
            }.padding(.horizontal)

            similarAdsView
        }
    }
}
```

Conditional Layout

```
extension AdView {
    var regularLayout: some View {
        VStack(alignment: .leading, spacing: .spacingM) {
            imageGalleryView

            HStack(alignment: .top, spacing: .spacingM) {
                VStack(alignment: .leading, spacing: .spacingM) {
                    Group {
                        titleView
                        priceView
                        addressView
                        descriptionView
                        // ...
                    }.padding(.horizontal)
                }

                VStack(alignment: .leading, spacing: .spacingM) {
                    profileView
                    sendMessageButton
                    phoneNumber
                }
            }

            similarAdsView
        }
    }
}
```

Recap

Recap (or just stating the obvious)

- use the right tool for the job (*I'm looking at you* **`UICollectionView`**)
- SwiftUI declarative syntax makes creating small/reusable components way better than UIKit
- multiple simple components, over single "universal" component
- ☝️ together with top level decisions, make implementing different layouts/scenarios easy.

Views: complexity and reusability

Experiments comparing SwiftUI and UIKit

Bonus

Bonus

- Attemp to use iOS 13 UICollectionView features while supporting iOS 11+: https://github.com/finn-no/FinniversKit/pull/759