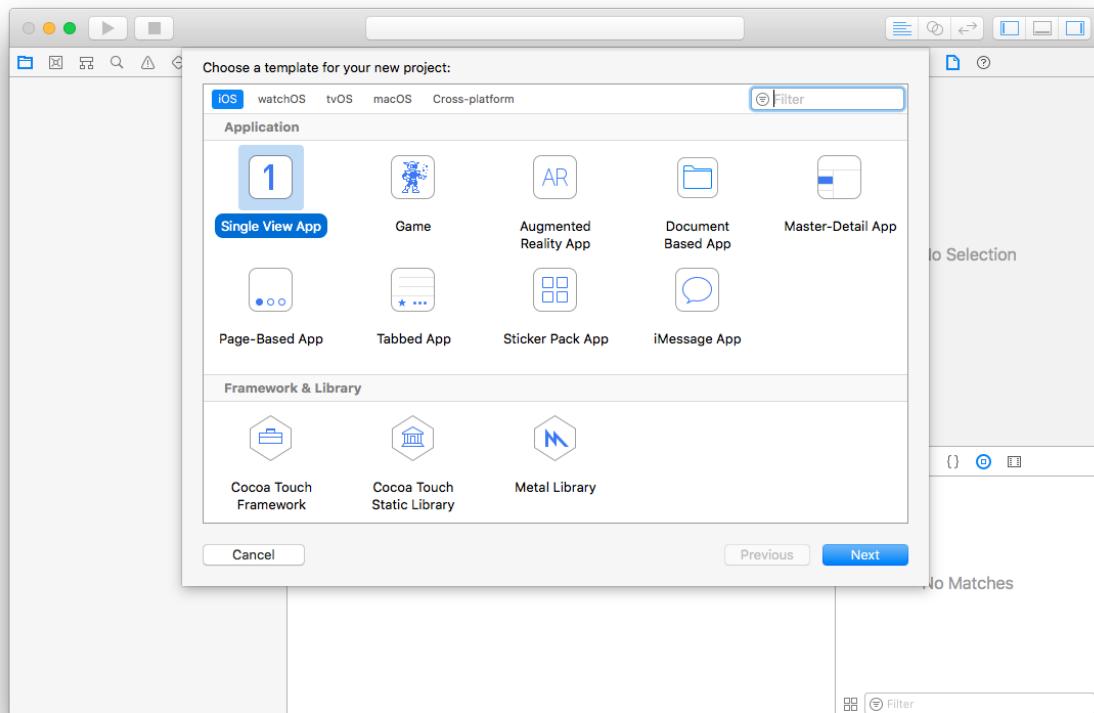


Wetter App iOS

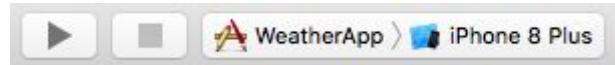
1. Neues Xcode-Projekt erstellen

- a. Create new Xcode project
- b. Single View App
 - i. ohne CoreData, mit Unit und UI-Tests
- c. Create Git repository on my Mac



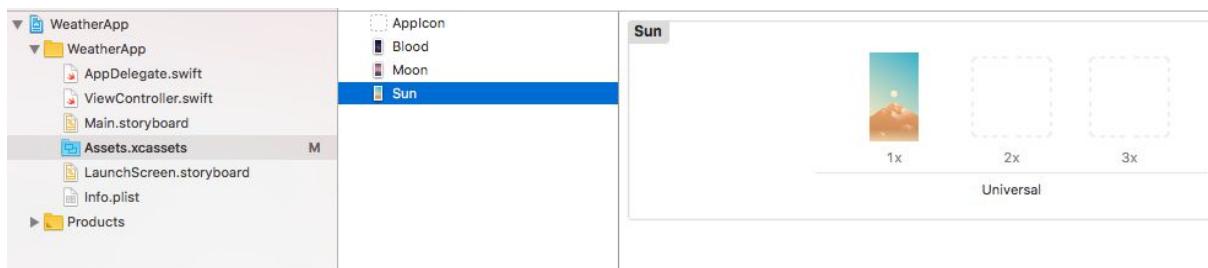
2. Auf Simulator laufen lassen

- a. diverse Simulatoren ausprobieren



3. Hintergrundbild ergänzen

- a. Bilder aus /ressources in Xcode-Ordner Assers.xcassets importieren (in Xcode)

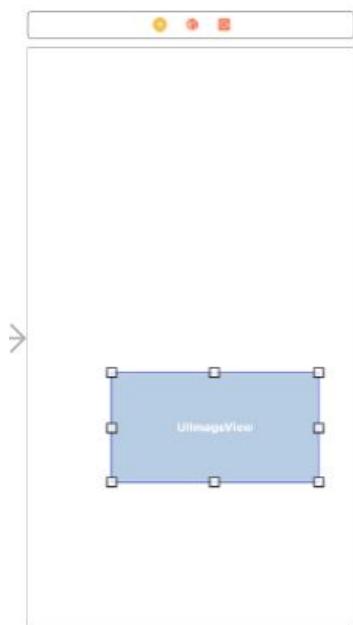


4. In Main.storyboard eine ImageView einfügen

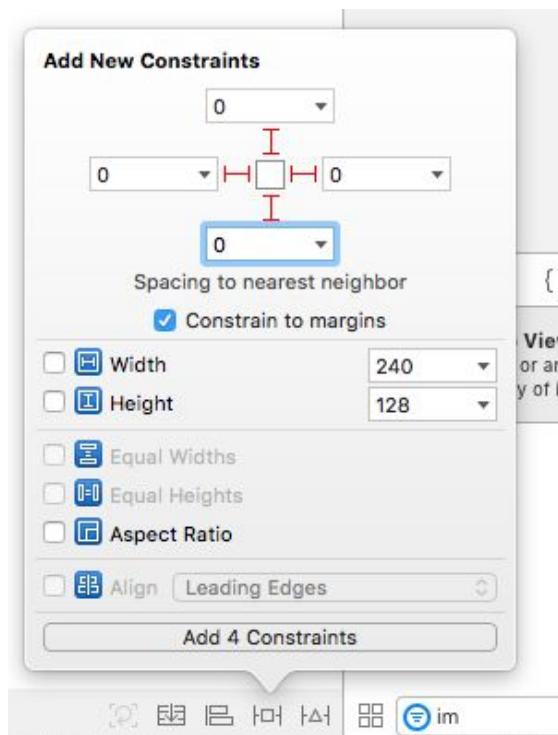
- im Katalog (object library) nach Image View suchen (Filterfunktion)



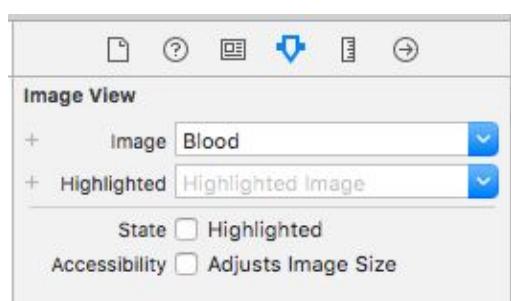
- ImageView auf View Controller ziehen



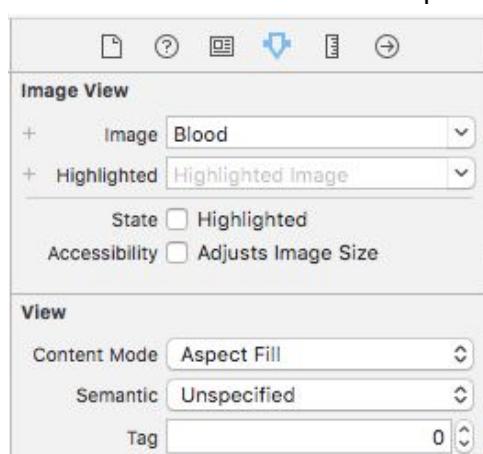
c. Autolayout Constraints setzen



d. Bild auswählen



e. Content Mode des Bildes auf Aspect Fill setzen



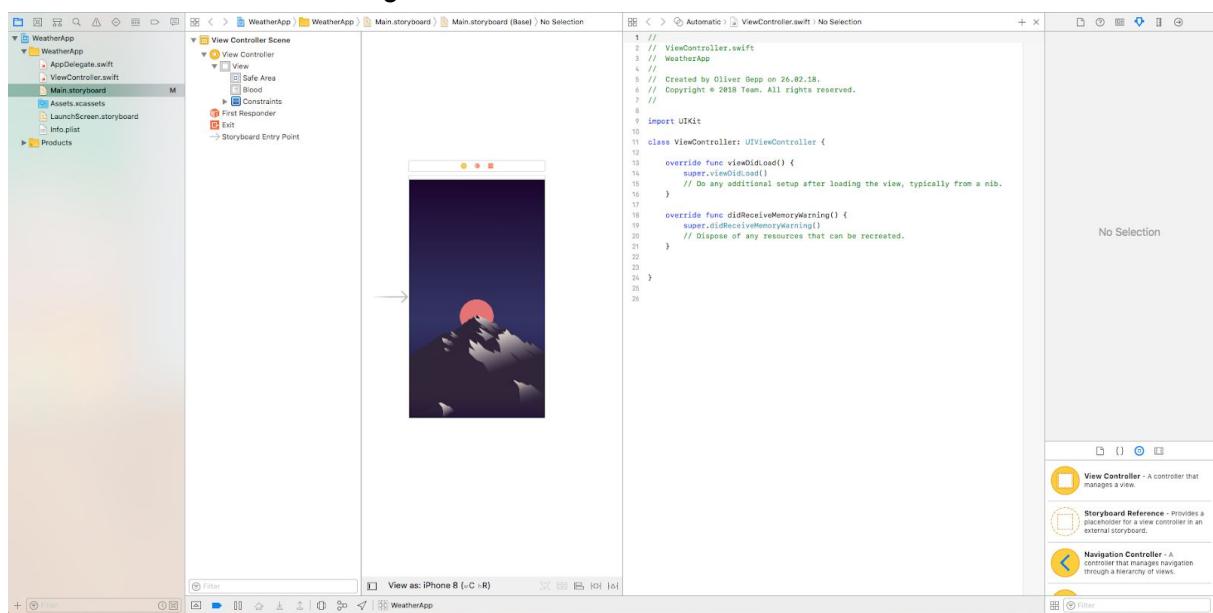
- f. Ergebnis im Simulator anschauen 
- a. auch in verschiedenen Simulatoren
 - b. Simulatoren rotieren (command + Pfeil rechts)



iPhone 8 Plus - 11.2

5. UI mit Code verbinden

a. Assistant Editor anzeigen lassen

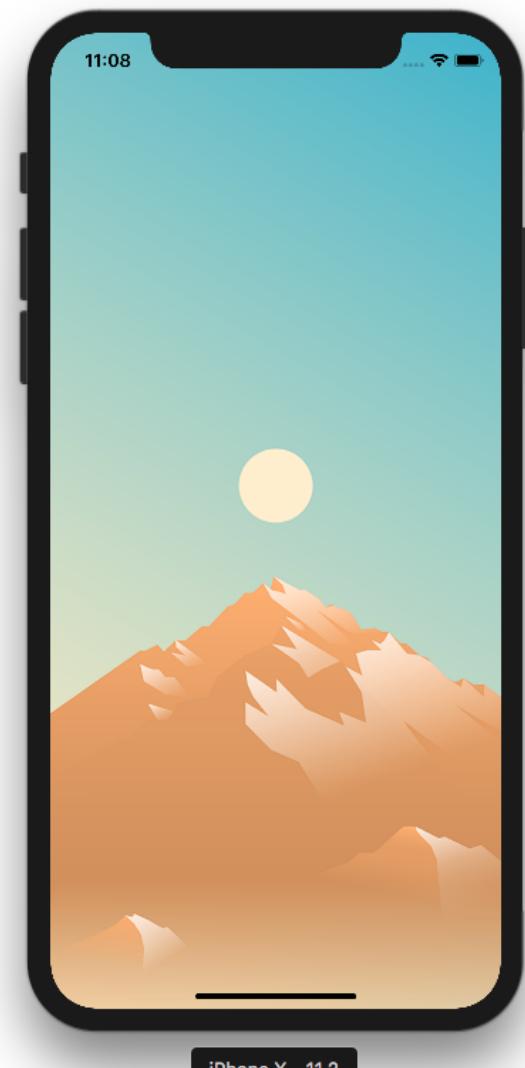


7. In Abhängigkeit zur aktuellen Tageszeit Bild anzeigen

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

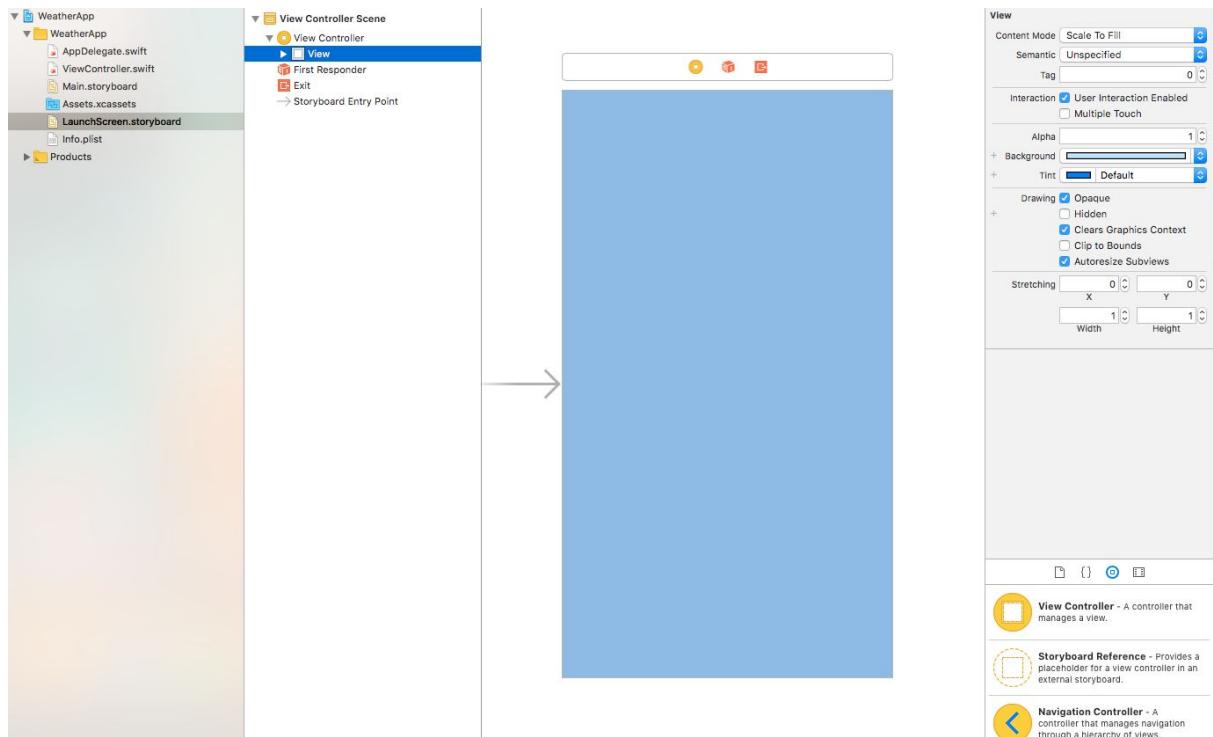
    let hour = Calendar.current.component(.hour, from: Date())

    switch hour {
    case 0...6:
        dayTimeImageView.image = UIImage(named: "Blood")
    case 7...19:
        dayTimeImageView.image = UIImage(named: "Sun")
    case 19...22:
        dayTimeImageView.image = UIImage(named: "Moon")
    default:
        dayTimeImageView.image = UIImage(named: "Blood")
    }
}
```



8. Splashscreen einfärben

- a. LaunchScreen.storyboard auswählen
- b. ViewController > View auswählen
- c. Im Attributes Inspector eine Farbe bei Background wählen



9. App-Icon ergänzen

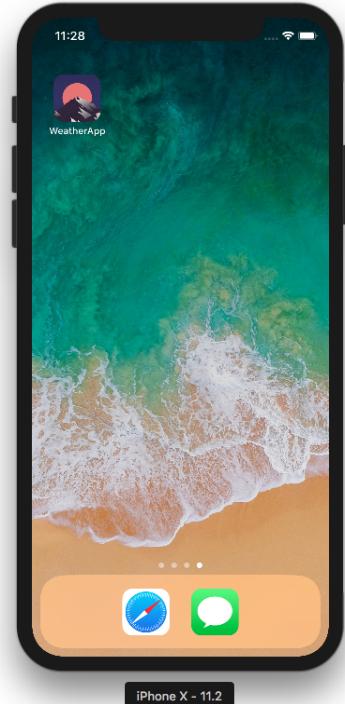
- a. In Assets.xcassets die diversen Größen von Apolcon anschauen
- b. 18 Bilder erstellen und zuweisen oder
- c. Bilder generieren lassen
- d. Seite <https://makeappicon.com/> aufrufen
- e. Bild app_icon.png aus /resources "in den Toaster schieben"
- f. Zip-Datei über E-Mail laden
- g. Ordner ApplIcon.appiconset aus Zip-Datei kopieren

▼	app_icon	Heute, 11:19
►	android	Heute, 11:19
►	imessenger	Heute, 11:19
▼	ios	Heute, 11:19
►	AppIcon.appiconset	Heute, 11:19
	iTunesArtwork@1x.png	Heute, 10:18
	iTunesArtwork@2x.png	Heute, 10:18
	iTunesArtwork@3x.png	Heute, 10:18
	README.md	29.04.16, 13:18
►	watchkit	Heute, 11:19

- h. In Ordner WeatherApp/Assets.xcassets einfügen und bestehenden Ordner AppIcon.appiconset ersetzen (Im Filesystem)
- i. In XCode AppIcon in Assets.xcassets überprüfen



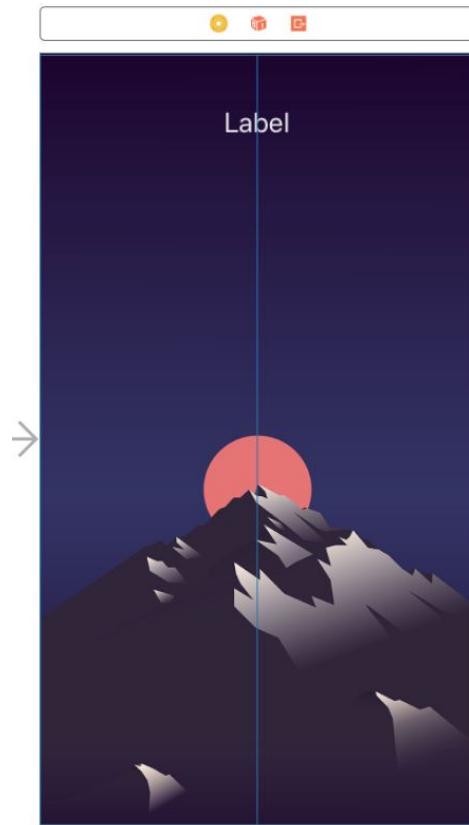
- g. App in Simulator laufen lassen und wieder beenden. Prüfen, dass App-Icon gesetzt ist.



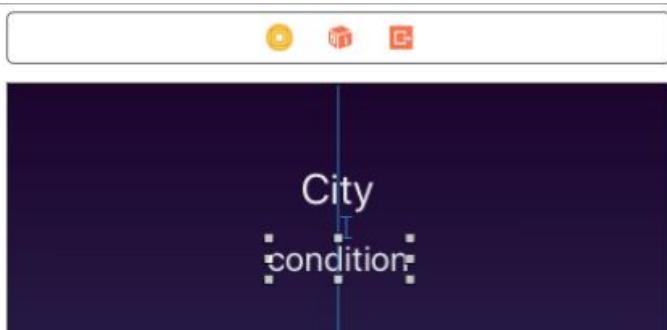
iPhone X - 11.2

10. Labels im UI ergänzen

- a. Main.storyboard öffnen, ViewController fokussieren
- b. Label aus Object library in View Controller ziehen
- c. Color: Gray, Font: System 24 im Attributes inspector setzen
- d. Autolayout-Constraints setzen
 - i. Center Horizontally
 - ii. Top: 25



- e. Label in City im Attributes Inspector umbenennen
- f. Mit gedrückter Alt-Taste Label mit der Maus nach unten ziehen: Ein Duplikat entsteht
- g. Autolayout-Constraints setzen
 - a. Vertical spacing zu City label
 - b. Center Horizontally
- h. Font: System 20
- i. Label City in condition umbenennen



- j. Schritte f-h entsprechend wiederholen. Ziel:



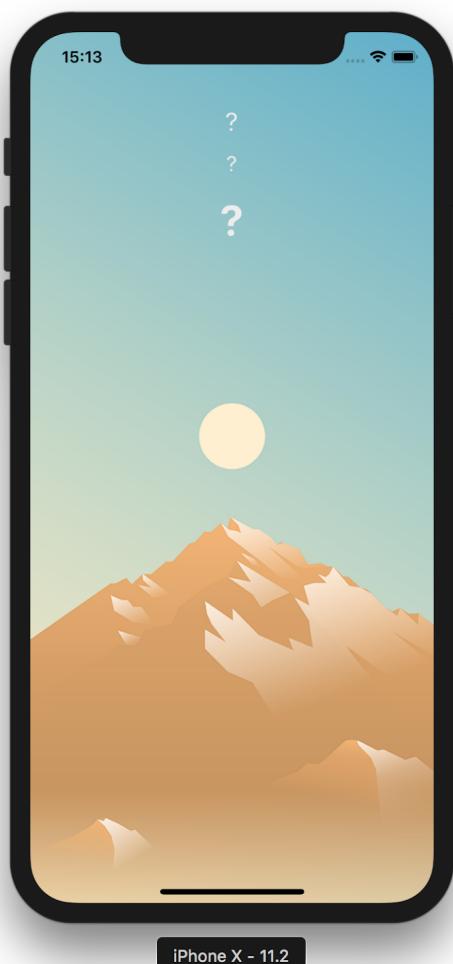
11. Labels mit Code verknüpfen

- a. analog Schritt 5
- b. Vorschlag Namen: cityLabel, conditionLabel, tempLabel

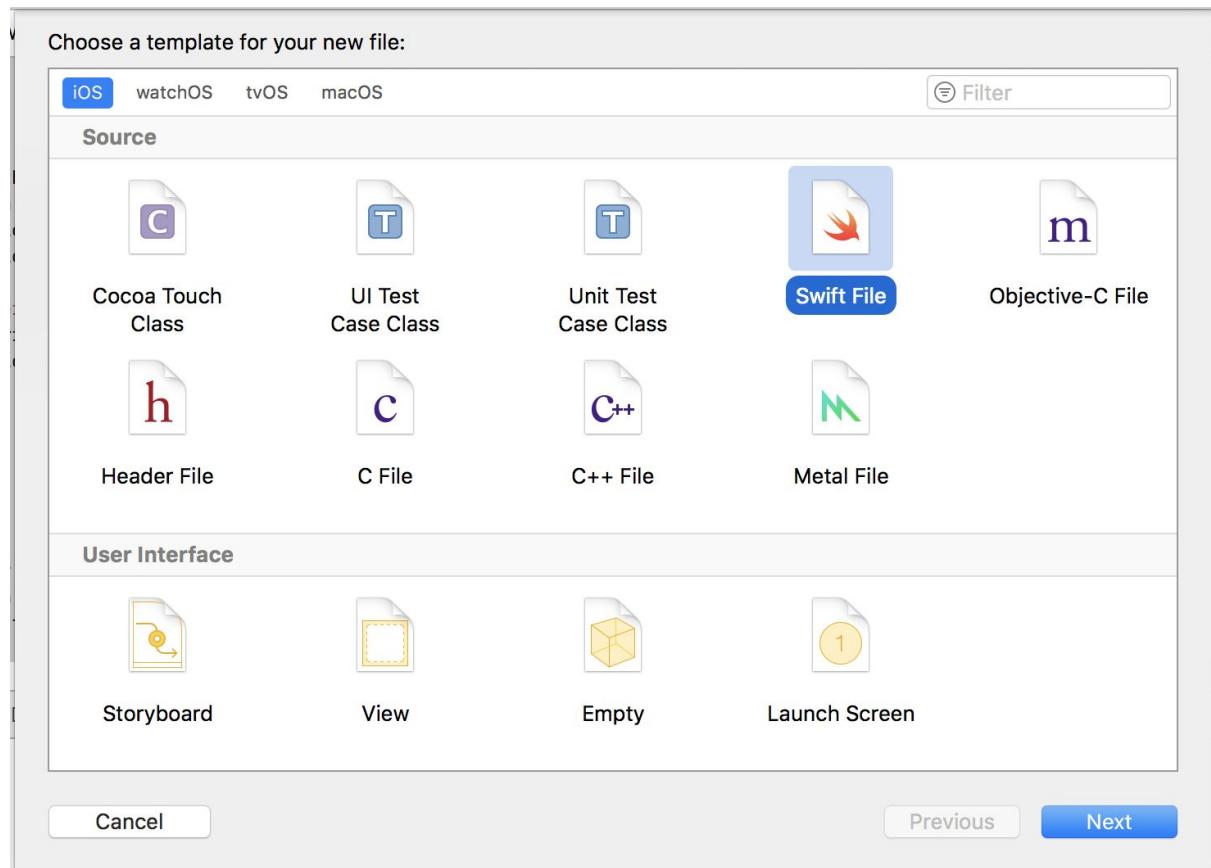
12. Verknüpfungen im Code nutzen um Werte zu resetten

```
fileprivate func resetLabels() {  
    cityLabel.text = "?"  
    conditionLabel.text = "?"  
    tempLabel.text = "?"  
}
```

- a. in viewDidLoad aufrufen.
- b. Code zum Setzen des Hintergrundbildes analog Methode `resetLabels()` refakturieren



13. Klasse WeatherModel.swift erzeugen



14. WeatherModel.swift implementieren

```
import Foundation

class WeatherModel{

    let apiKey = "TODO"
    let long = "7.44744"
    let lat = "46.94809"
    let endpoint =
"https://api.openweathermap.org/data/2.5/forecast?lat={LAT}&lon={LON}&lang=de
&units=metric&APPID={KEY}"

    func loadWeather(completionHandler: @escaping ([String: AnyObject]?, Error?) -> Void) {

        let urlString = endpoint
            .replacingOccurrences(of: "{LAT}", with: lat)
            .replacingOccurrences(of: "{LON}", with: long)
    }
}
```

```

        .replacingOccurrences(of: "{KEY}", with: apiKey)

    guard let url = URL(string: urlString) else {
        print("Error: cannot create URL")
        let error = BackendError.urlError(reason: "Could not construct
URL")
        completionHandler(nil, error)
        return
    }
    let urlRequest = URLRequest(url: url)

    let session = URLSession.shared
    let task = session.dataTask(with: urlRequest, completionHandler: {
        (data, response, error) in

        guard error == nil else {
            completionHandler(nil, error!)
            return
        }

        guard let responseData = data else {
            print("Error: did not receive data")
            let error = BackendError.objectSerialization(reason: "No data
in response")
            completionHandler(nil, error)
            return
        }

        do {
            let weatherDic = try JSONSerialization.jsonObject(
                with: responseData,
                options: .mutableContainers) as? [String: AnyObject]
            completionHandler(weatherDic, nil)
        } catch {
            print("error trying to convert ")
            print(error)
            completionHandler(nil, error)
        }
    })
    task.resume()
}

enum BackendError: Error {
    case urlError(reason: String)
    case objectSerialization(reason: String)
}

```

(Falls noch kein Key vorhanden ist, unter https://home.openweathermap.org/users/sign_up erstellen)

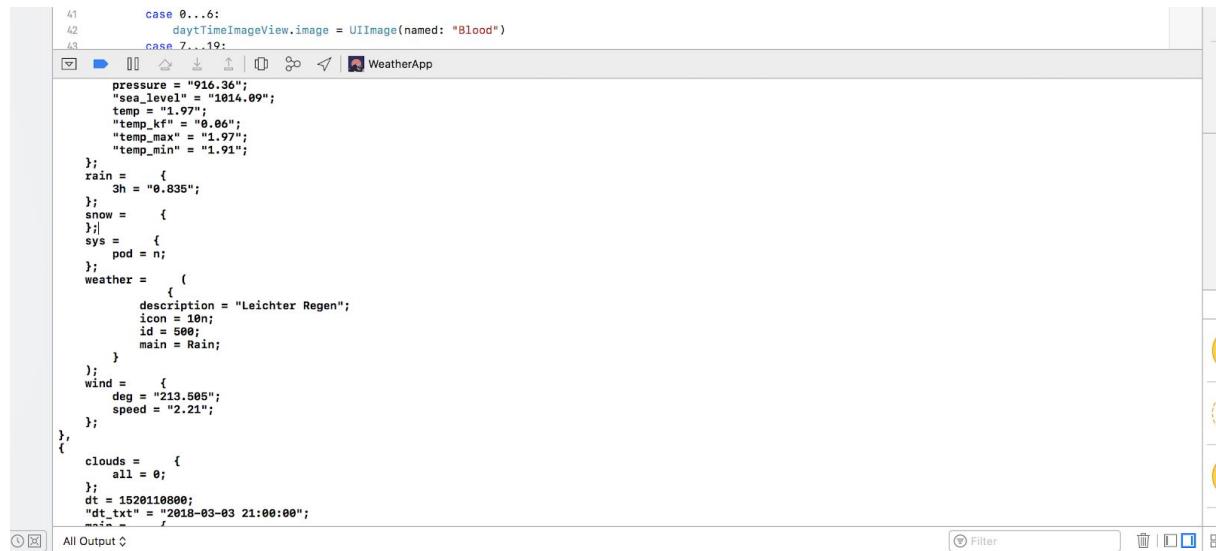
15. WeatherModel aus ViewController aufrufen

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

    setBackgroundImage()
    resetLabels()
    weatherModel.loadWeather { [String: AnyObject]?, error: Error? } in
        if let weatherDic = weatherDic{
            print(weatherDic)
        }

        if let error = error{
            print(error)
        }
    }
}
```

16. Ergebnis in Konsole anschauen



The screenshot shows the Xcode interface with the "Output" tab selected. The console output displays a JSON dictionary representing weather data. The dictionary includes fields like pressure, sea_level, temp, temp_k, temp_max, temp_min, rain, snow, sys, pod, weather, wind, and clouds. The weather object contains a description ("Leichter Regen"), icon (10n), id (500), and main (Rain). The wind object contains deg (213.505) and speed (2.21). The clouds object contains all (0). A timestamp (2018-03-03 21:00:00) and a date string (1520110800) are also present.

```
41     case 0...6:
42         dayTimeImageView.image = UIImage(named: "Blood")
43     case 7...19:
44
45         pressure = "916.36";
46         "sea_level" = "1014.09";
47         temp = "1.97";
48         "temp_k" = "0.06";
49         "temp_max" = "1.97";
50         "temp_min" = "1.91";
51
52         rain = {
53             "3h" = "0.835";
54         };
55         snow = {
56         };
57         sys = {
58             "pod" = n;
59         };
60         weather = {
61             {
62                 description = "Leichter Regen";
63                 icon = 10n;
64                 id = 500;
65                 main = Rain;
66             }
67         };
68         wind = {
69             deg = "213.505";
70             speed = "2.21";
71         };
72
73     {
74         clouds = {
75             all = 0;
76         };
77         dt = 1520110800;
78         "dt_txt" = "2018-03-03 21:00:00";
79     }
80 }
```

17. Die Stadt extrahieren und ins Label schreiben

```
weatherModel.loadWeather { (weatherDic: [String: AnyObject]?, error: Error?) in
    DispatchQueue.main.async {
        if let weatherDic = weatherDic{
            if let city = weatherDic["city"] {
                if let cityName = city["name"] as? String{
                    self.cityLabel.text = cityName
                }
            }
        }

        if let error = error{
            print(error)
        }
    }
}
```

18. Weitere Werte extrahieren

```
if let listArray = weatherDic["list"] as? [AnyObject]{
    if let firstItem = listArray.first as? [String: AnyObject]{
        if let weatherArray = firstItem["weather"] as? [AnyObject]{
            if let fristWeatherItem = weatherArray.first as? [String: AnyObject]{
                if let description =
fristWeatherItem["description"] as? String{
                    self.conditionLabel.text = description
                }
            }
        }

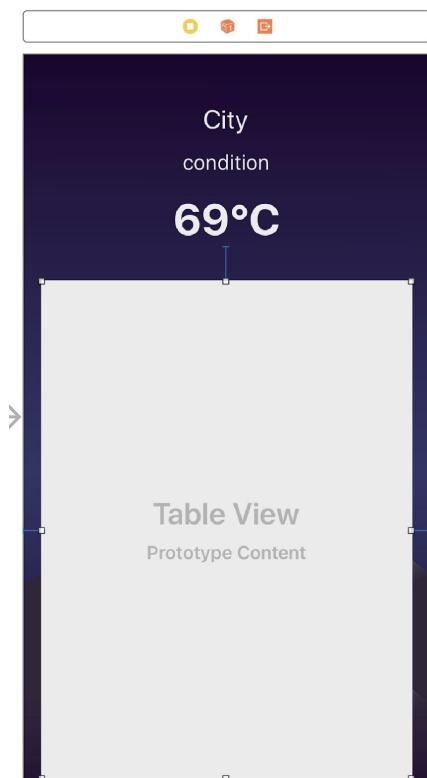
        if let mainArray = firstItem["main"] as? [String: AnyObject]{
            if let temp = mainArray["temp"] as? Double{
                self.tempLabel.text = "\\" + (temp) + "°C"
            }
        }
    }
}
```



iPhone 8 Plus - 11.2

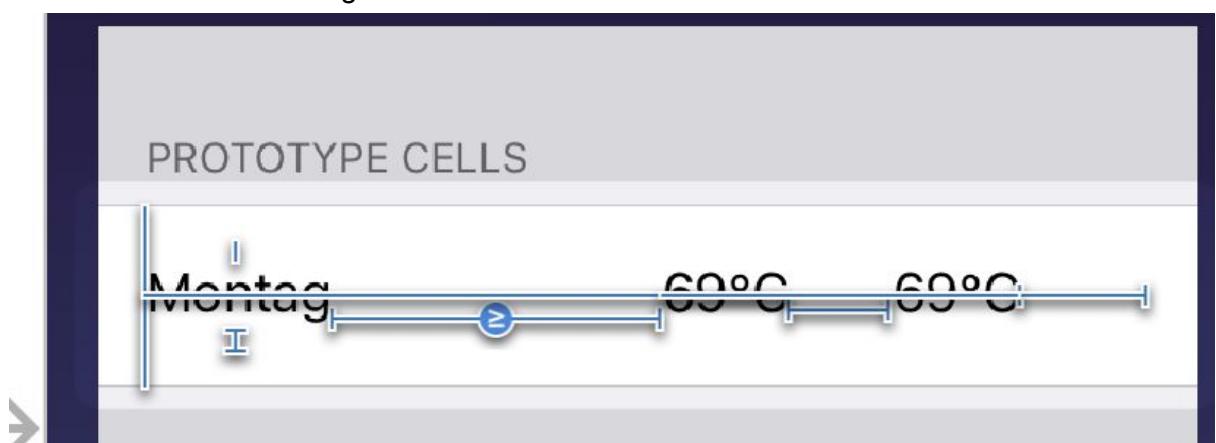
19. UITableView ergänzen

- Table View aus Object library auf ViewController ziehen
- Constraints ergänzen (16,32,16,0)
- Background: Clear Color
- Style: Grouped



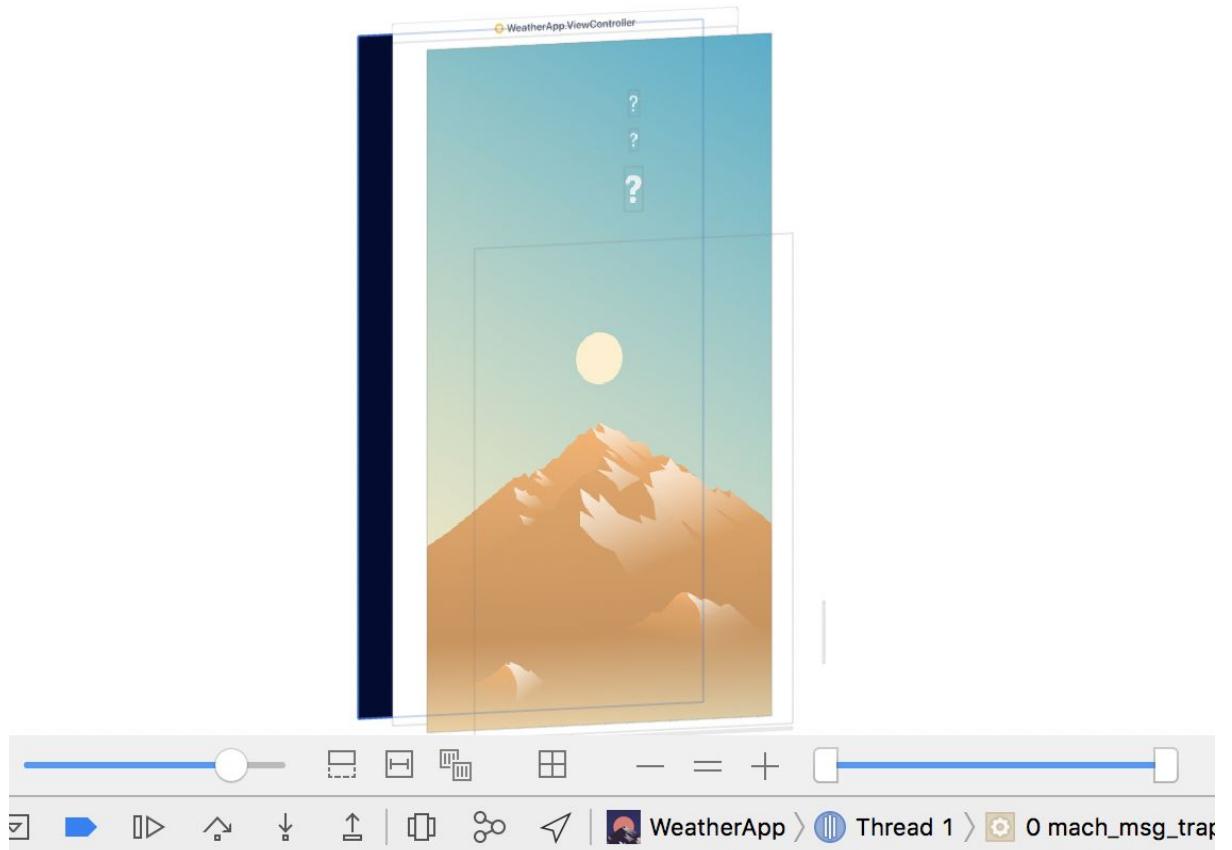
20. Tabellenzelle ergänzen

- PrototypeCells: 1
- 3 Labels in die Zelle ziehen und mit Constraints ergänzen
- ContentView BackgroundColor: Clear Color



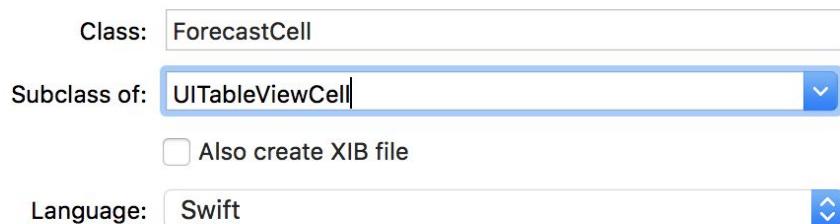
21. Tabelle mit ViewController verknüpfen

- a. Im Assistent Editor analog der Labels in Schritt 5 das IBOutlet tableView erstellen
- b. App ausführen
- c. Debug View Hierarchy starten

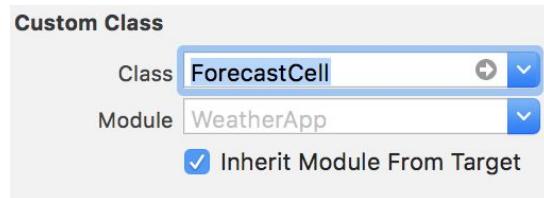


22. TableViewCell Code ergänzen

- d. Cocoa Touch Class vom Typ UITableViewCell hinzufügen
- e. Name: ForecastCell



- f. Im Storyboard der TableViewCell die Klasse ForecastCell zuweisen



- g. Im Attributes inspector der Zelle den Identifier ForecastCell geben

23. UITableViewDataSource implementieren

- h. In ViewController.swift unterhalb der Klasse eine extension ergänzen

```
extension ViewController: UITableViewDataSource{

    func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int)
-> Int {
        return 5
    }

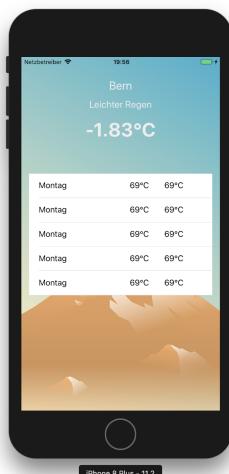
    func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath)
-> UITableViewCell {

        return tableView.dequeueReusableCell(withIdentifier: "ForecastCell", for:
indexPath)
    }
}
```

- b. TableView-Delegate innerhalb von viewDidLoad() ergänzen

```
tableView.dataSource = self
```

- c. App ausführen → 5 Zellen



24. Model umstrukturieren

- a) WeatherItem struct erstellen

```
struct WeatherItem{  
  
    var city: String  
    var date: Date  
    var min: Double  
    var max: Double  
    var description: String  
}
```

- b) WeatherModel so umbauen, dass dieses ein Array von WeatherItems zurück gibt

```
func loadWeather(completionHandler: @escaping ([WeatherItem]?, Error?) -> Void){...}
```

- c) Gegebenfalls Hilfsmethode im Model zur Parsen verwenden

```
private func parseWeatherDictionary(weatherDic: [String: AnyObject])->[WeatherItem]{  
    ...  
    //parsing date  
    if let interval = item["dt"] as? Double{  
        date = Date(timeIntervalSince1970: interval)  
    }  
    ...  
}
```

- d) ViewController anpassen

```
weatherModel.loadWeather { (weatherItems: [WeatherItem]?, error: Error?) in  
  
    DispatchQueue.main.async {  
  
        guard let items = weatherItems else{return}  
  
        if let item = items.first{  
            self.conditionLabel.text = item.description  
            self.tempLabel.text = "\(item.max) °C"  
            self.cityLabel.text = item.city  
        }  
  
        self.weatherItems.removeAll()  
        self.weatherItems.append(contentsOf: items)  
        self.tableView.reloadData()  
    }  
}
```

(weatherItems ist eine Klassenvariable (Array von WeatherItems))

25. ForecastCell mit UI verknüpfen

- Im Main.Storyboard das Label für den Wochentag selektieren
- In den Assistant-Editor wechseln und im rechten Feld die ForecastCell auswählen
- 3 Outlets für die Labels in der Zelle erstellen

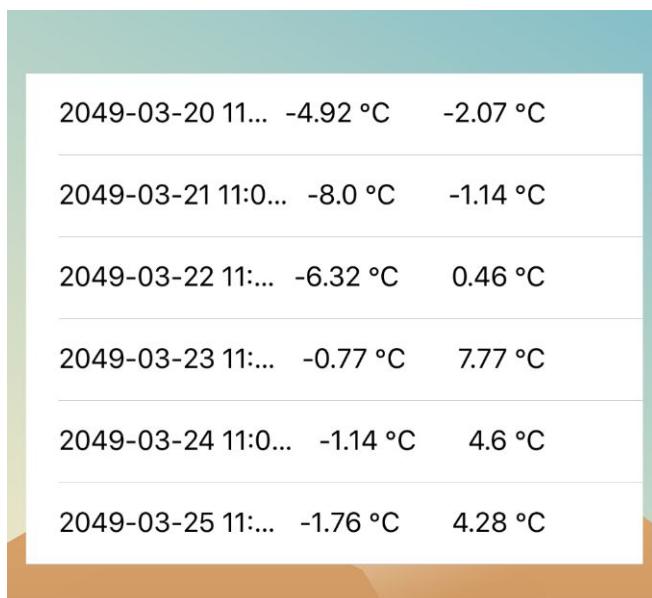
```
@IBOutlet weak var weekdayLabel: UILabel!
@IBOutlet weak var minTempLabel: UILabel!
@IBOutlet weak var maxTempLabel: UILabel!
```

26. Zellen mit Inhalt befüllen

```
extension ViewController: UITableViewDataSource{

    func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        return weatherItems.count - 1
    }

    func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
        if let cell = tableView.dequeueReusableCell(withIdentifier: "ForecastCell", for: indexPath) as? ForecastCell{
            let weatherItem = weatherItems[indexPath.row+1] //+1 because first item already displayed in view
            cell.weekdayLabel.text = "\(weatherItem.date)"
            cell.minTempLabel.text = "\(weatherItem.min) °C"
            cell.maxTempLabel.text = "\(weatherItem.max) °C"
            return cell
        }
        return UITableViewCell()
    }
}
```



27. DateFormatter verwenden

- Einen DateFormatter in den ViewController ergänzen als Klassenvariable ergänzen

```
private let dateFormatter = DateFormatter()
```

- DateFormatter zum Anzeigen des Wochentags formatieren

```
dateFormatter.dateFormat = "EEEE" //weekday
```

- Datum mit Dateformatter verwenden

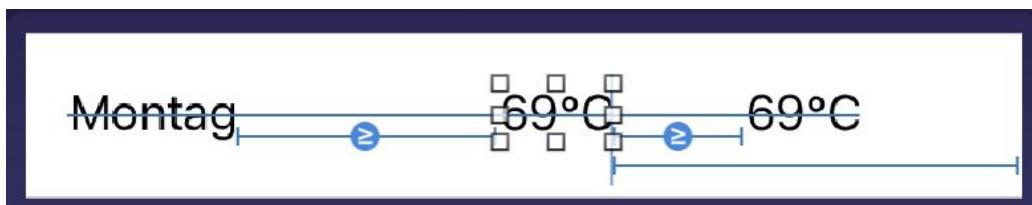
```
cell.weekdayLabel.text = dateFormatter.string(from: weatherItem.date)
```

Samstag	-4.92 °C	-2.07 °C
Sonntag	-8.0 °C	-1.14 °C
Montag	-6.32 °C	0.46 °C
Dienstag	-0.77 °C	7.77 °C
Mittwoch	-1.14 °C	4.6 °C
Donnerstag	-1.76 °C	4.28 °C

(Es wird die Gerätesprache verwendet)

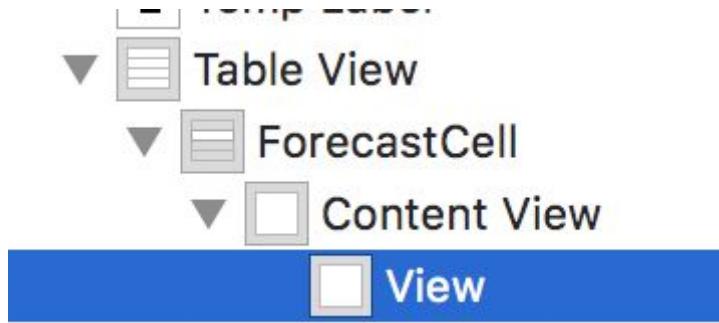
28. UI optimieren

- das Label in der Mitte springt durch die Abhängigkeit zur Label rechts. Dies durch Anpassen des Constraints nach rechts anpassen

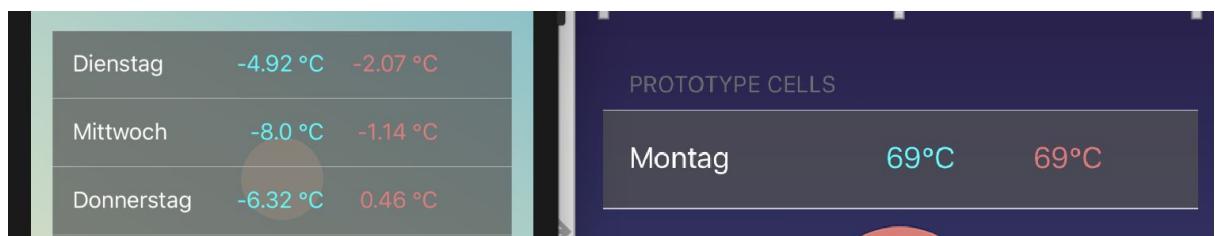


- Min und Max farblich anpassen
- Transparenz für die Zelle umsetzen
- Eigene View in die Hierarchie ganz oben einfügen

- b. Diese View stylen (dunkle Farbe, alpha 0.7)



- c. alle Views oberhalb in der ViewHierarchy transparent machen (Background auf Clear Color setzen)



29. Temperatur runden

- a. Eine Extension für Double erstellen: Neue SwiftKlasse DoubleExt.swift erzeugen

```

import Foundation

extension Double{

    func rounded(places: Int) -> Double{
        let divisor = pow(10.0, Double(places))
        return (self * divisor).rounded() / divisor
    }
}
  
```

- b. Extension anwenden

```
weatherItem.max.rounded(places: 1)
```

30. App ausführen und auf verschiedenen Geräten testen



Disclaimer: Das API gibt nicht die aktuelle Temperatur nur das Max und Minimum für den aktuellen Tag. Wir zeigen in der App das Maximum für den heutigen Tag an. Für den akademischen Zweck sollte das aber ausreichend sein.

18:13



Bern

Mäßiger Schnee

1.3 °C

Dienstag -4.9 °C -2.1 °C

Mittwoch -8.0 °C -1.1 °C

Donnerstag -6.3 °C 0.5 °C

Freitag -0.8 °C 7.8 °C

Samstag -1.1 °C 4.6 °C

Sonntag -1.8 °C 4.3 °C

iPhone X - 11.2