Taking the NSExpress

From CoreData directly into Swift Charts

Lena Stöxen

About me

Lena Mattea Stöxen

- Student for Computer Science
- Volkswagen Commercial Vehicles
- iOS as a hobby
- Art and drawing
- Love colors and birds

My Apps

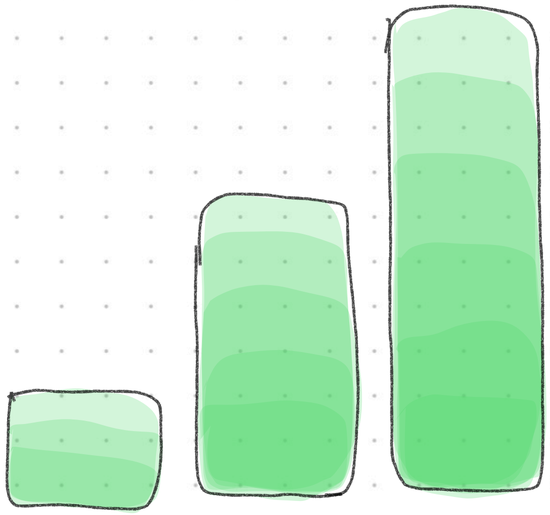
The Assembler

- Learning binary operations with pixel-art and assembly
- For iPad only

MerkMal

- Tracking symptoms of illnesses like migraines or rheumatism
- For iOS and WatchOS 10

Shiny new frameworks



Swift Charts

- Introduced in WWDC 22
- SwiftUI
- Easy to make

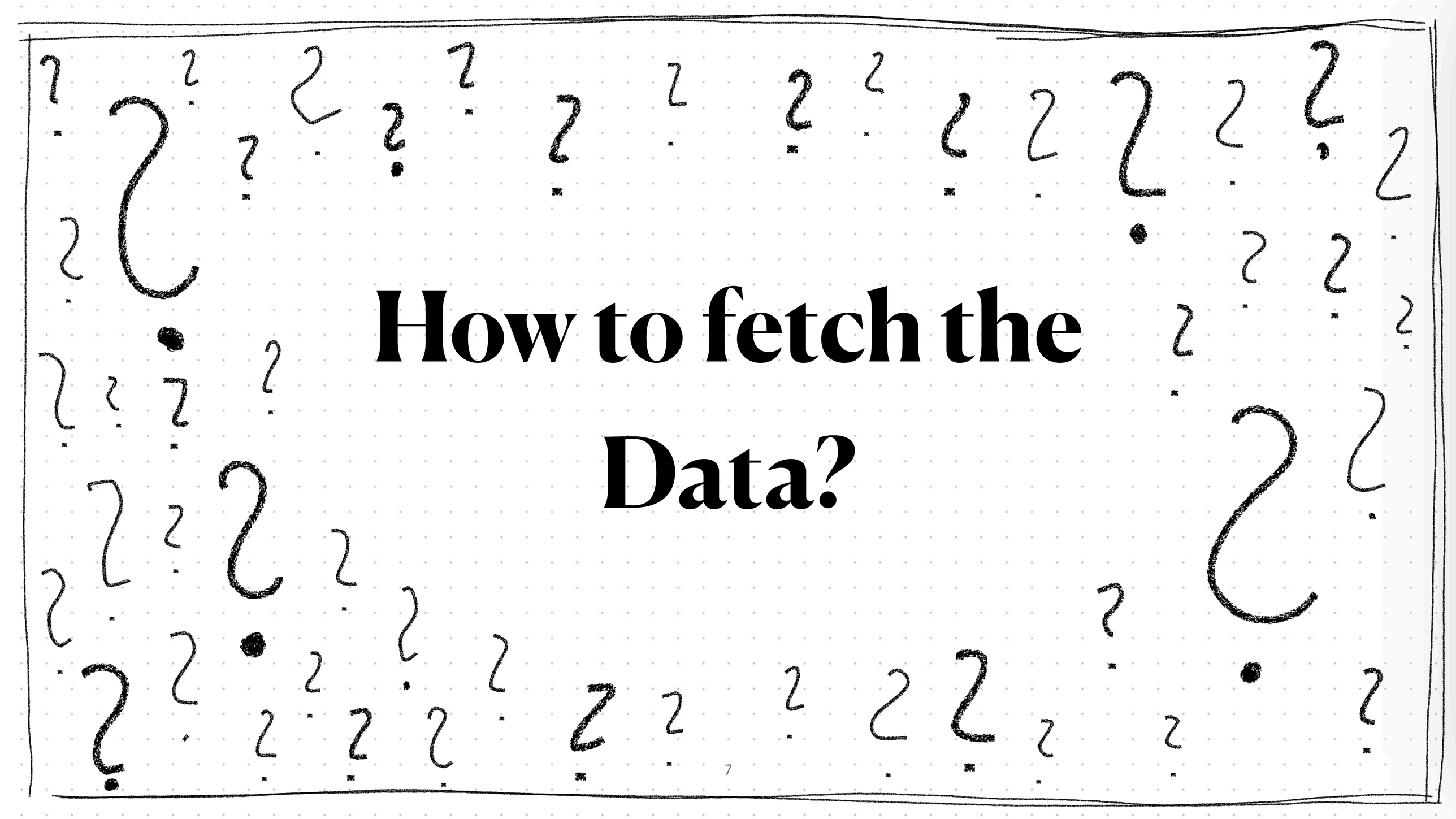
Shiny new frameworks

Swift Charts

- Introduced in W[illegible]22
- Swift UI
- E[illegible] make

Sample App

- Shows how to use Swift Charts
- Data is in nice arrays



How to fetch the Data?

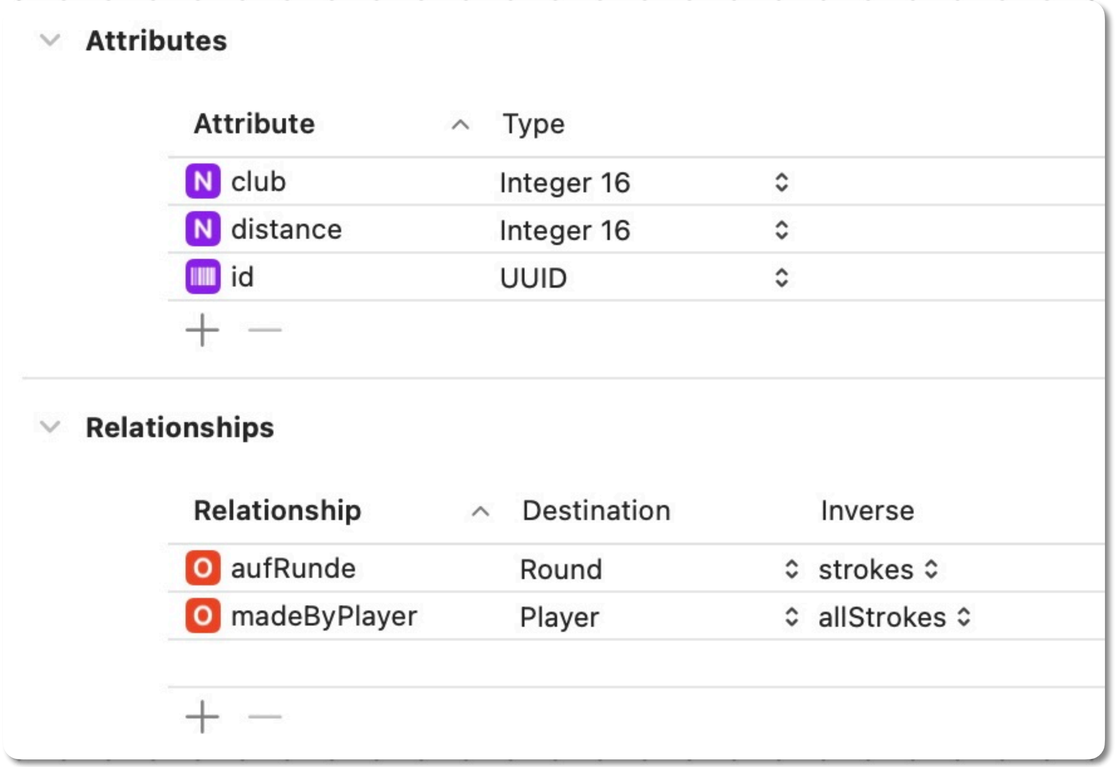
Let’s make a sample project

Sample Project

Golf

- Reasonably complex
- I play golf
- For(e) Girls!

Model: Stroke



Strokes

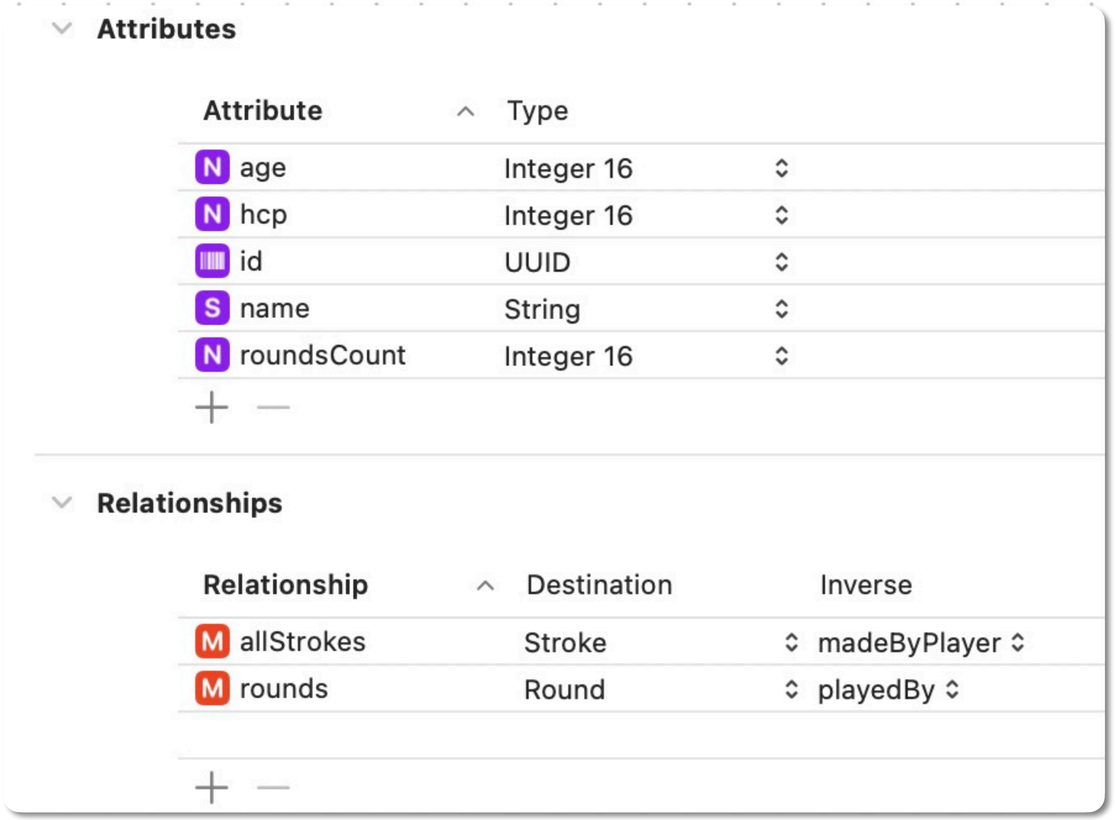
id	distance	club	madeByPlayer	madeOnRound
UUID	Integer	Integer	Relation	Relation

Model: Round

Round

id	date	par	player	score	strokes
UUID	Date	Integer	Relation	Integer, Derived	Relation to many

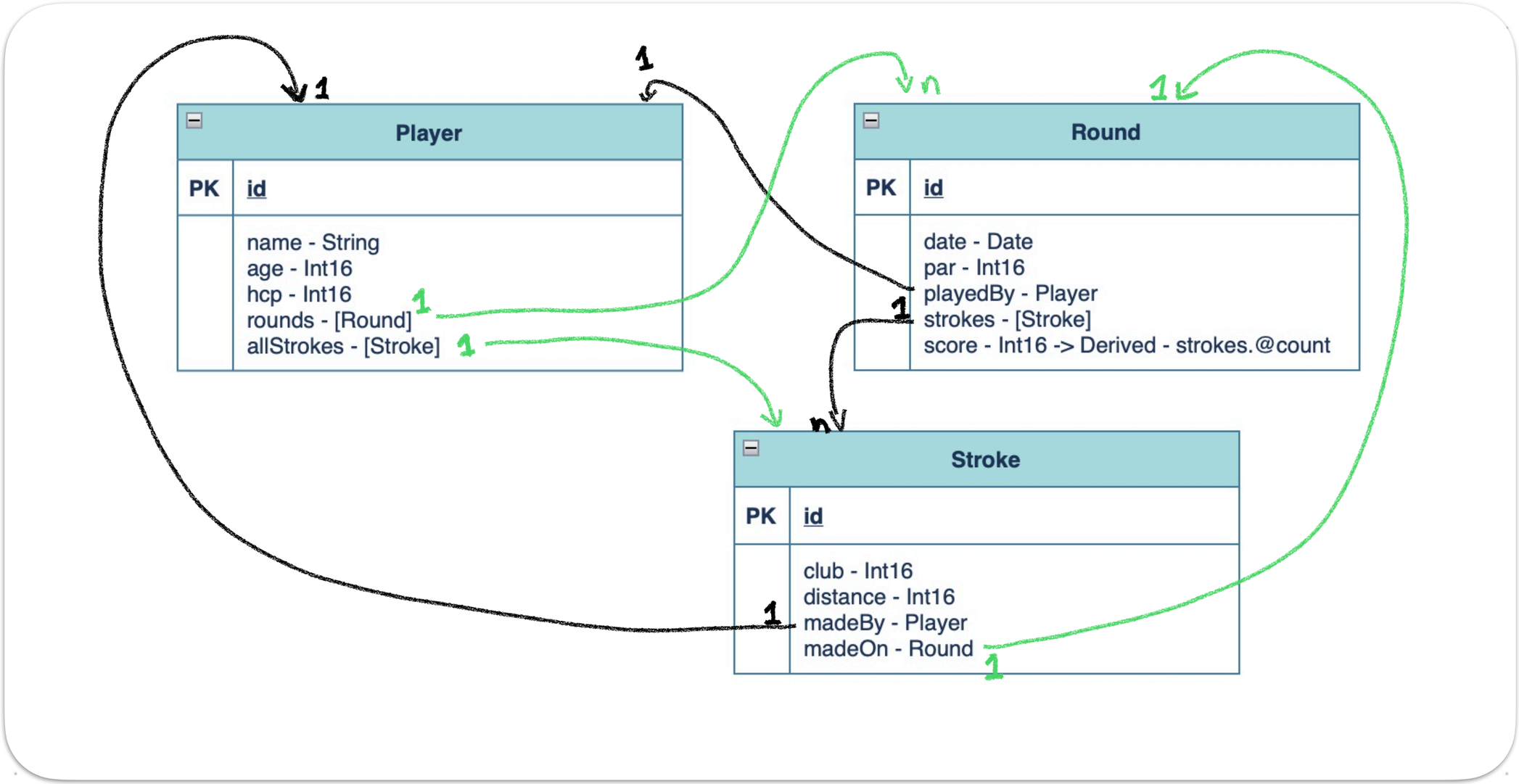
Model: Player



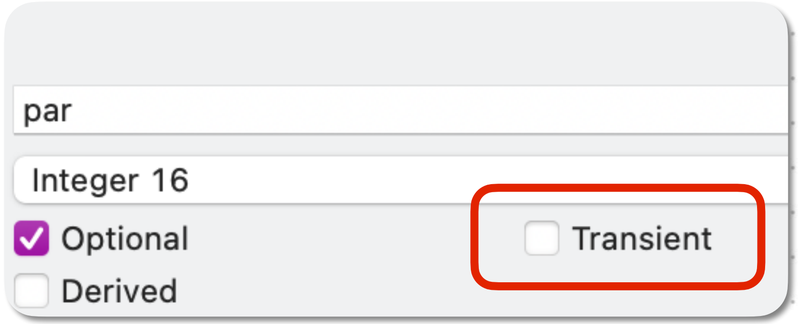
Player

id	name	age	strokes	rounds
UUID	Integer	Integer	Relation to many	Relation to many

Data model

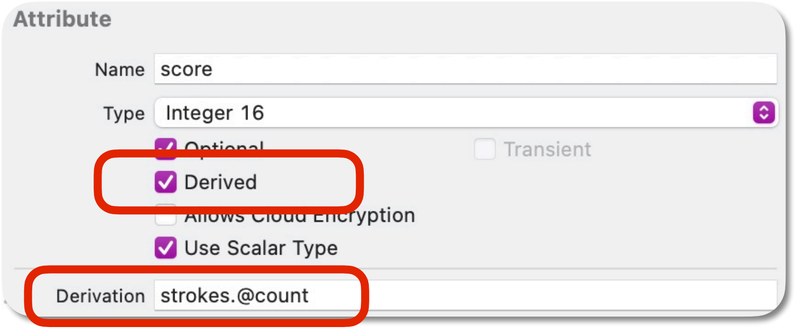

Player
PK
id
name - String
age - Int16
hcp - Int16
rounds - [Round]
allStrokes - [Stroke]
Round
PK
id
date - Date
par - Int16
playedBy - Player
strokes - [Stroke]
score - Int16 -> Derived - strokes.@count
Stroke
PK
id
club - Int16
distance - Int16
madeBy - Player
madeOn - Round
1
n

Attributes



Transient

- No keypath that leads to column
- Not in SQLite table (persistent store)



Derived

- Has keypath that leads to column
- In SQLite table (persistent store)
- @Count counts one-to-many relationships

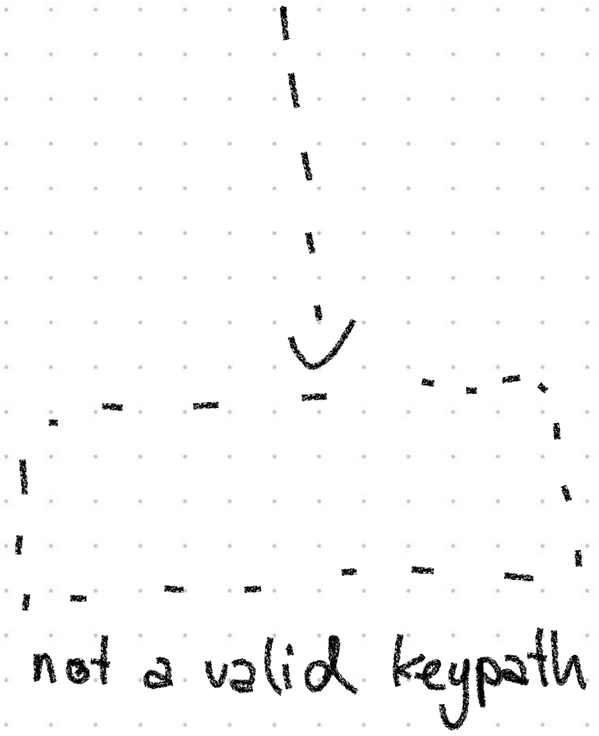
Valid Keypaths

Class in Core Data

name	score	color
Attribute	Derived Attribute	Transient Attribute

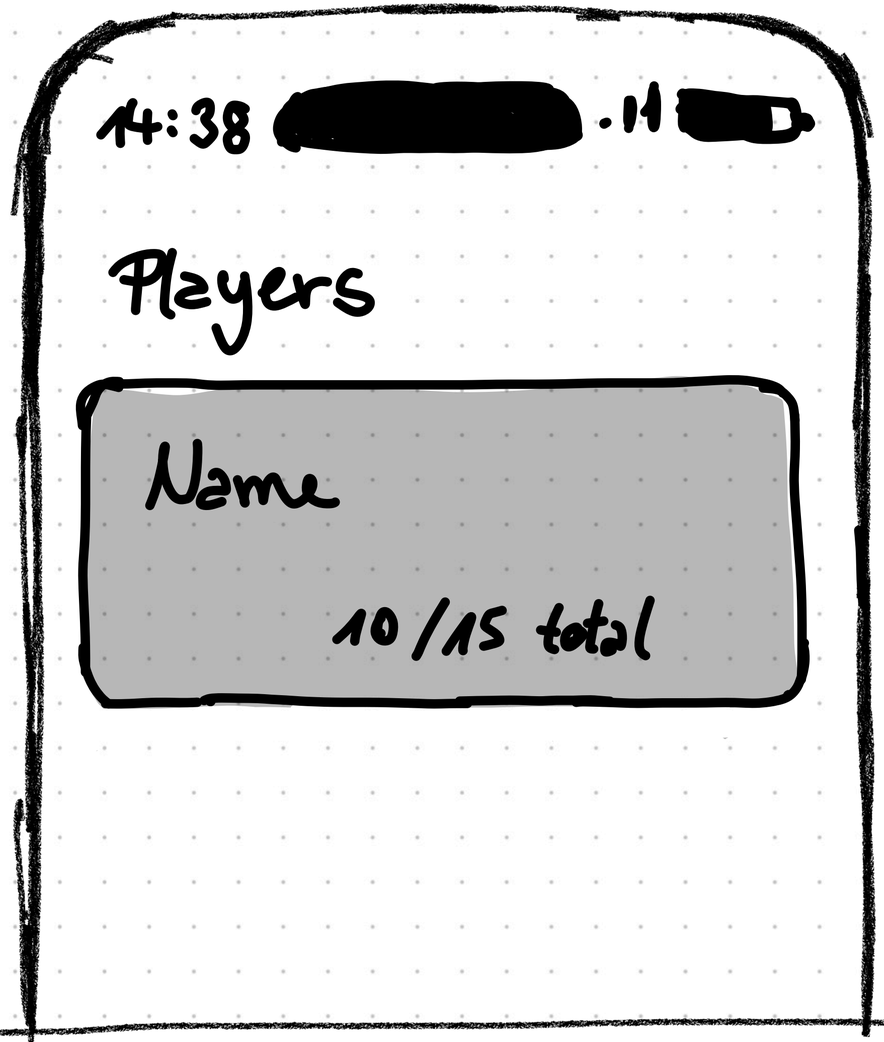
SQL

name	score



Counting

Counting



Goal

- Number of rounds
- Rounds this year

Context.count(request)

```
let request: NSFetchRequest<Round> = Round.fetchRequest()
request.predicate = NSPredicate(format: "playedBy == %@", player)

do{
    _ = try viewContext.count(for: request)
}catch{
    print(error)
}
```

- Returns a number
- SQLs count() function

- Customise what gets counted with predicates

.countRecordType

```
let request = NSFetchRequest<NSFetchRequestResult>(entityName: "Round")
request.predicate = NSPredicate(format: "playedBy == %@", player)

request.resultType = .countResultType

do{
    _ = try viewContext.fetch(request)
}catch{
    print(error)
}
```

- Returns an array with a number
- SQLs count() function

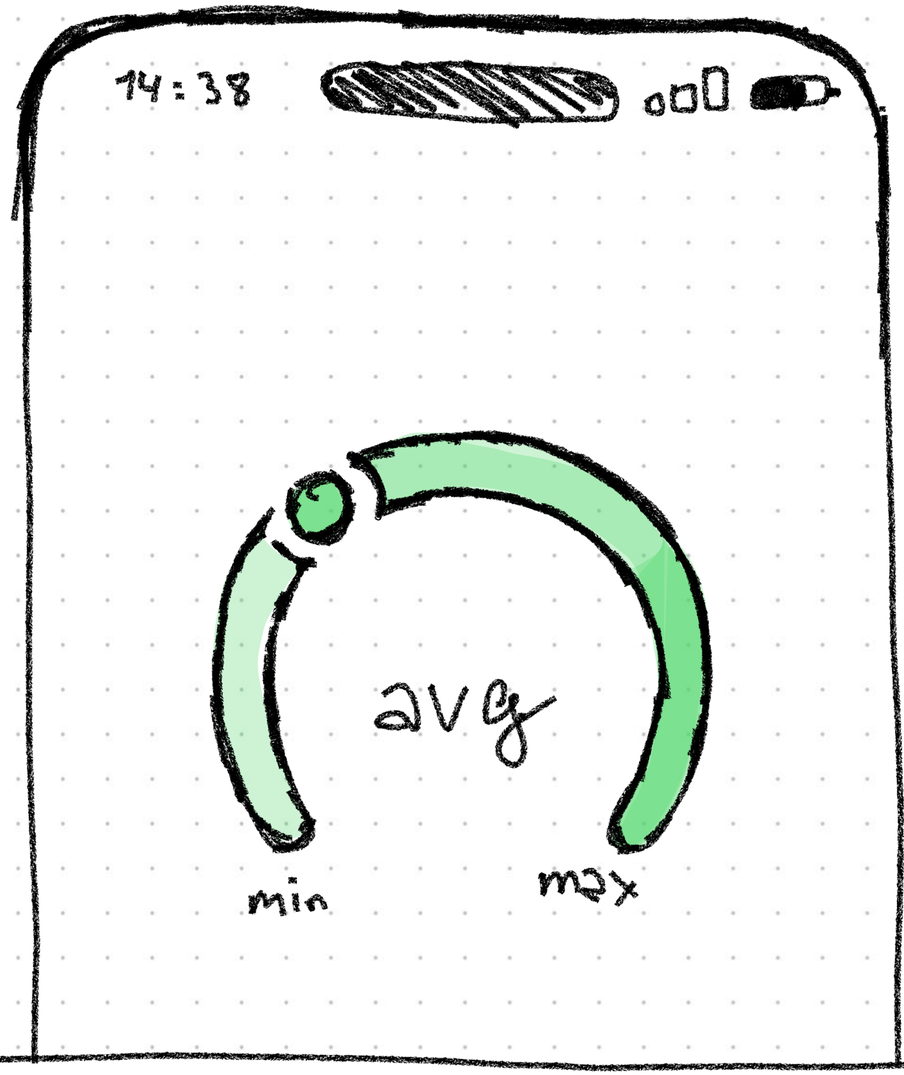
- Customise what gets counted with predicates

Statistics

Statistics

Goal

- Average score
- Maximum score
- Minimum score



Setting up NSExpressions

```
let keypathExpMin = NSExpression(forKeyPath: #keyPath(Round.score))
let expressionMin = NSExpression(forFunction: "min:", arguments: [keypathExpMin])
```

- #keyPath(Class.attribute) = „attribute“
- Keypath can be any valid keypath on the object
- Function is the name of a function that NSExpression supports
- For ‚min:‘ keypath has to lead to a column that’s filled with numbers or dates

Setting up NSExpressionDescription

```
let minDesc = NSExpressionDescription()
minDesc.expression = expressionMin
minDesc.name = "min"
minDesc.expressionResultType = .integer16AttributeType
```

- Every expression needs to be in a expression description
- Name will be in the resulting dictionary
- ResultType can be all datatypes supported by CoreData

Setting up Request

```
let request = NSFetchRequest<NSFetchRequestResult>(entityName: "Round")
request.propertiesToFetch = [minDesc]
request.resultType = .dictionaryResultType
```

- RequestType has to be dictionary
- Properties to fetch is an array of keypaths and/or NSExpressionDescriptions
- The request has to be set up with NSFetchRequestResult and not the object

Full Code

```
        let keypathExpMin = NSExpression(forKeyPath: #keyPath(Round.score))
        let expressionMin = NSExpression(forFunction: "min:", arguments: [keypathExpMin])

        let minDesc = NSExpressionDescription()
        minDesc.expression = expressionMin
        minDesc.name = "min"
        minDesc.expressionResultType = .integer16AttributeType

        let request = NSFetchRequest<NSFetchRequestResult>(entityName: "Round")
        request.propertiesToFetch = [minDesc]
        request.resultType = .dictionaryResultType

        request.predicate = NSPredicate(format: "playedBy == %@", player)

        do{
            let result = try viewContext.fetch(request)
            print(result)
//              [{
//                  min = 62;
//              }]
        }catch{
            print(error)
        }
```

Supported Functions

```
let keypathExpMin = NSExpression(forKeyPath: #keyPath(Round.score))
let expressionMin = NSExpression(forFunction: "min:", arguments: [keypathExpMin])
```

- Minimum ‚min:‘
- Maximum ‚max:‘
- Average ‚average:‘
- Count ‚count:‘
- Sum ‚sum:‘
- Addition ‚add:to:‘
- Division ‚divide:by:‘
- Subtraction ‚from:substract:‘
- Multiplication ‚multiply:by:‘
- Modulus ‚modulus:by:‘
- Absolute value ‚abs:‘

Don’t trust the documentation

On a Random Side Note

Exceptions

- NSInvalidArgumentException
- NSInternalInconsistencyException

On a “random” Side Note

random: 

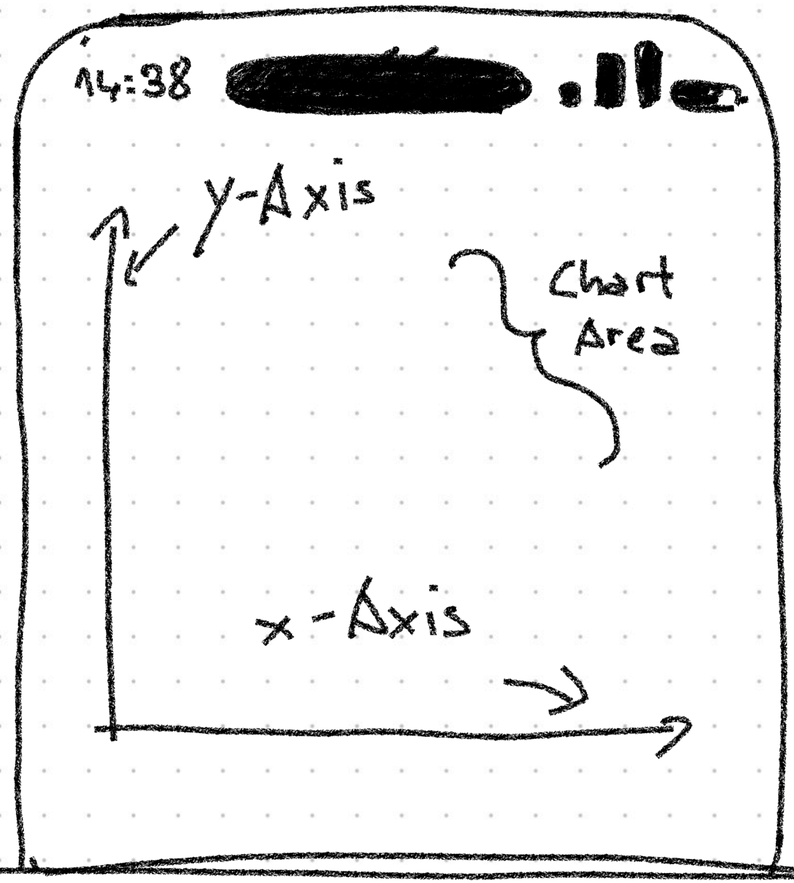
NSInternalInconsistencyException

randomn: 

NSInvalidArgumentException

Where is Swift Charts?

Charts



Real charts

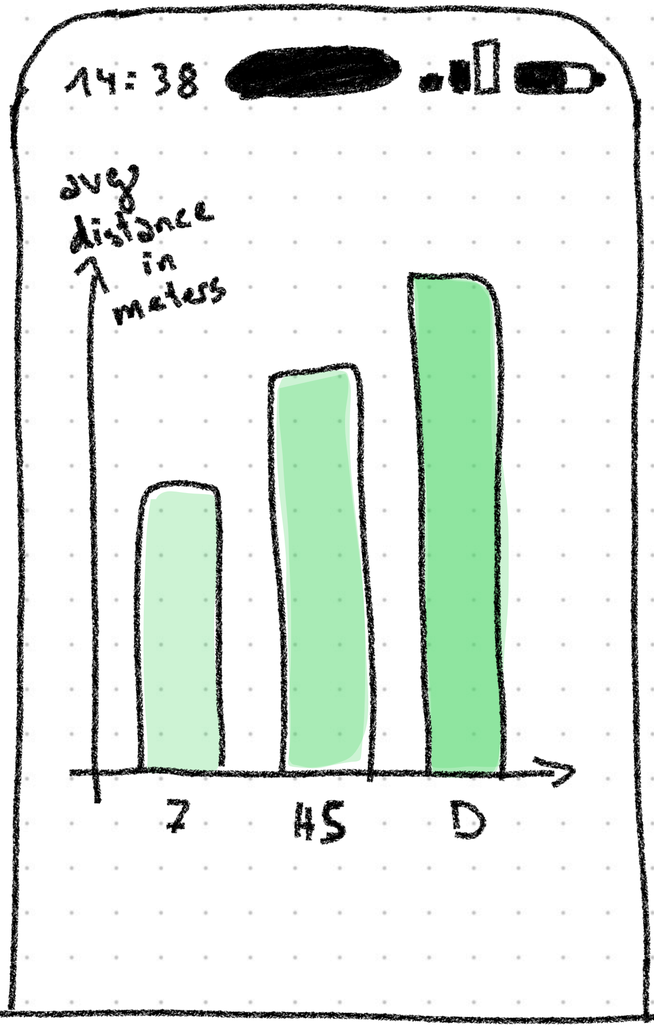
- Distance **per** club
- Average score **per** month
- Under/over Par **per** round

Grouping

Grouping

Goal

- Average distance per club



Setting up Grouping

```
let request = NSFetchRequest<NSFetchRequestResult>(entityName: "Stroke")
request.propertiesToFetch = [#keyPath(Stroke.club), avgDesc]
request.propertiesToGroupBy = [#keyPath(Stroke.club)]
request.resultType = .dictionaryResultType
```

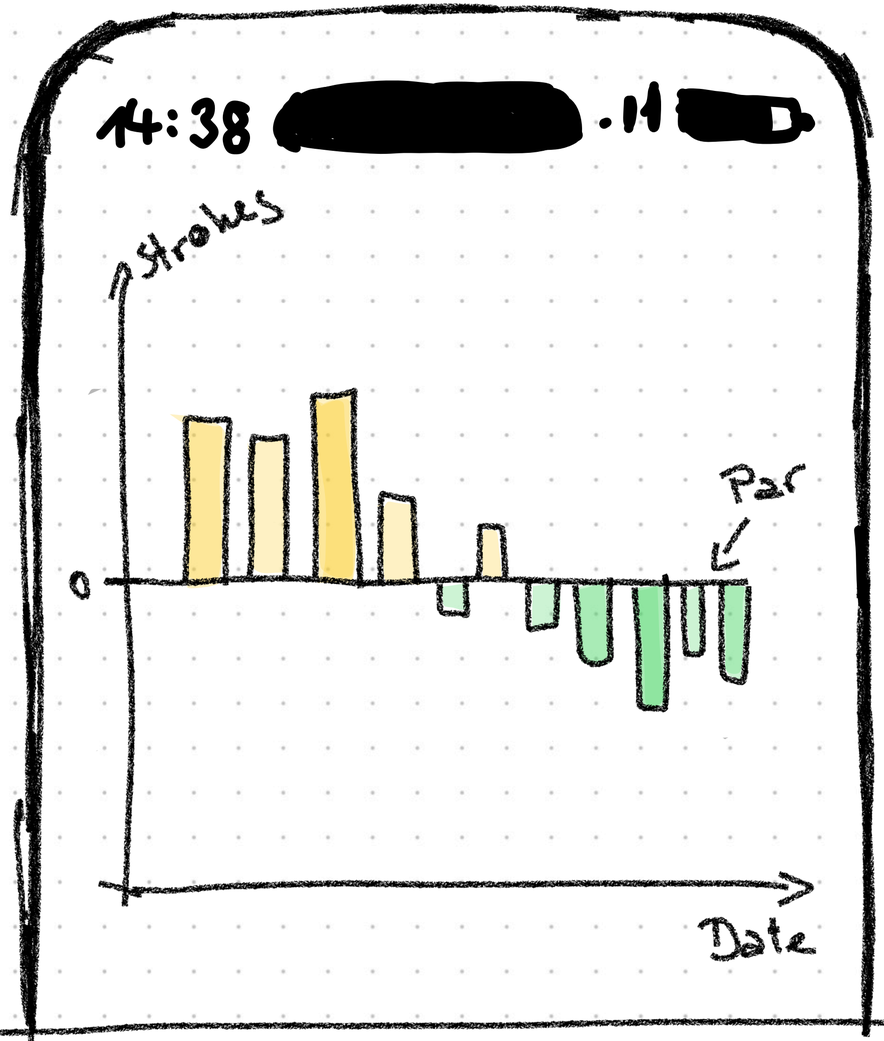
- PropertiesToFetch: keypath or NSExpressionDescription
- PropertiesToFetch: keypaths only
- All keypaths in the fetch need to be in the group
- Grouping by NSExpressions does not work

Don’t trust the documentation

Full Code

```
let keypath = NSExpression(forKeyPath: #keyPath(Stroke.distance))
let expressionAvg = NSExpression(forFunction: "average:", arguments: [keypath])

let avgDesc = NSExpressionDescription()
avgDesc.expression = expressionAvg
avgDesc.name = "average"
avgDesc.expressionResultType = .doubleAttributeType

let request = NSFetchRequest<NSFetchRequestResult>(entityName: "Stroke")
request.propertiesToFetch = [#keyPath(Stroke.club), avgDesc]
request.propertiesToGroupBy = [#keyPath(Stroke.club)]
request.resultType = .dictionaryResultType

request.sortDescriptors = [NSSortDescriptor(key: #keyPath(Stroke.club), ascending: true)]
request.predicate = NSPredicate(format: "madeByPlayer == %@", player)

var result: [NSFetchRequestResult] = []
do{
    result = try viewContext.fetch(request)
    guard let array = result as? [NSDictionary] else { return }
    ///'chartData' = [(club: String, avg: Double)] & can be used directly in Swift Charts
    chartData = array.map({
        return (club: Club(rawValue: $0["club"] as! Int16)!.description, avg: $0["average"] as! Double )
    })
}catch{
    print(error)
}
```

Group by Date



Goal

- Strokes over/under Par
- NSExpression function with two arguments

Setting up Expression

```
let score = NSExpression(forKeyPath: #keyPath(Round.score))
let par = NSExpression(forKeyPath: #keyPath(Round.par))
let overParExp = NSExpression(forFunction: "from:subtract:", arguments: [score, par])
```

- Arguments need to be two Expressions here

```
let request = NSFetchRequest<NSFetchRequestResult>(entityName: "Round")
request.propertiesToFetch = [#keyPath(Round.date), overParDesc]
request.propertiesToGroupBy = [#keyPath(Round.date)]
```

- Dates are often unique, because of the time
- Grouping by date makes (mostly) individual items

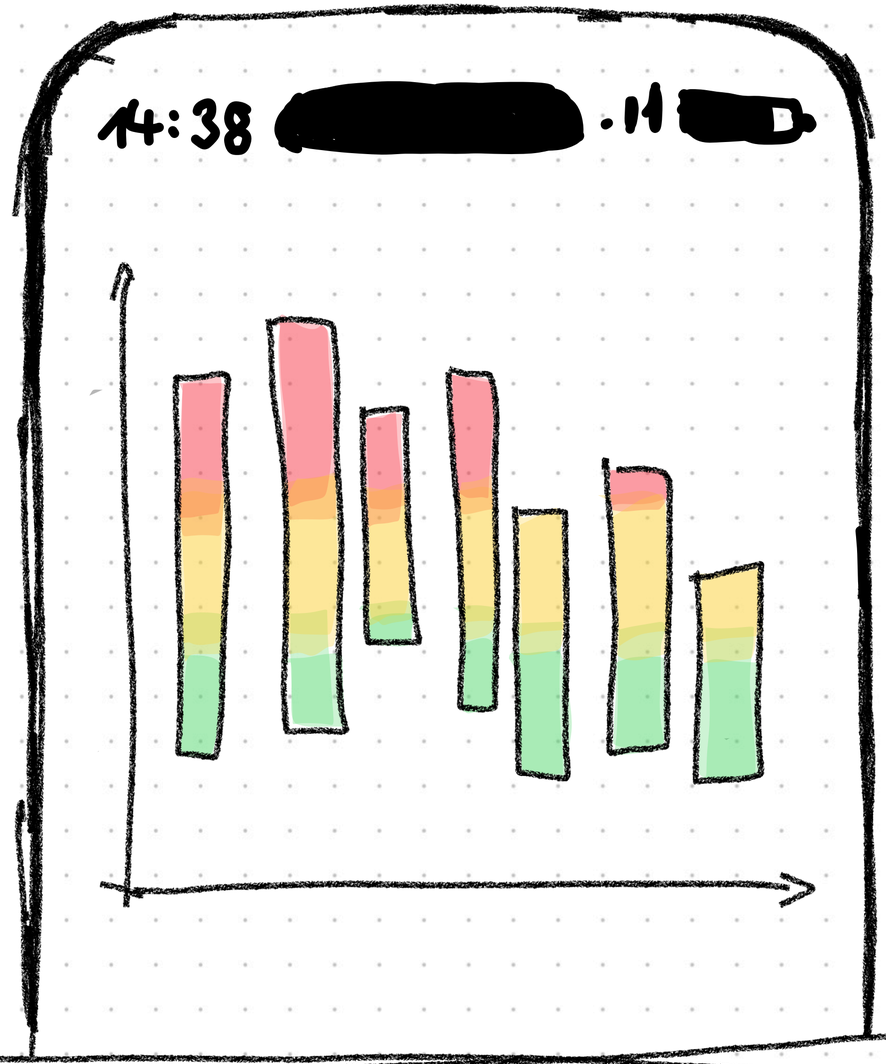
Full Code

```
let score = NSExpression(forKeyPath: #keyPath(Round.score))
let par = NSExpression(forKeyPath: #keyPath(Round.par))
let overParExp = NSExpression(forFunction: "from:subtract:", arguments: [score, par])

let overParDesc = NSExpressionDescription()
overParDesc.expression = overParExp
overParDesc.name = "strokesOverPar"
overParDesc.expressionResultType = .integer16AttributeType

let request = NSFetchRequest<NSFetchRequestResult>(entityName: "Round")
request.propertiesToFetch = [#keyPath(Round.date), overParDesc]
request.propertiesToGroupBy = [#keyPath(Round.date)]

request.resultType = .dictionaryResultType

request.sortDescriptors = [NSSortDescriptor(key: "date", ascending: true)]
request.predicate = NSPredicate(format: "playedBy == %@", player)

do{
    let result = try viewContext.fetch(request)

    guard let array = result as? [NSDictionary] else{ return }
    print(array)
    chartData = array.map({
        return (date: $0["date"] as! Date, overPar: $0["strokesOverPar"] as! Int )
    })
}catch{
    print(error)
}
```

Group by month

Goal

- Minimum score per month
- Average score per month
- Maximum score per month



What's a perfect date?

Dates

- SQL offers date separation
- CoreData doesn't offer something like that
- Dates in CoreData aren't ‚Dates' they are large integers, instances of time

This is where things get a bit complicated

Option A

Dates are Integers anyway

Dates as Integers

Idea

- Format dates as „yMMdd“
- 18th August 2023 - 20230818
- Divide by 100 to get month and year
- 20230818 - 202308 - 2023

Dates as Integers

Con

- Can’t make it a Date again - Date needed for Chart x-Axis
    - String to Date only with y.MM.dd, separator need to be added
- Can’t group by expression (can’t group by the result from division)
- Can’t fetch what’s not in group (can’t fetch a Date)
- Redundancy

Option B

More Dates

More Dates

Idea

- Add two more attributes to the object (Round)
- Format date with components
- Timestamp would be 18th August 2023 14:38
    - Attribute month -> 12th August 2023 12:00 (set time and day)
    - Attribute day -> 18th August 2023 12:00 (set time)

More Dates

Con

- Redundancy
- Time zones

Option C

Date Components

Date Components

```
self.dayComponent = Int16(Calendar.current.component(Calendar.Component.day, from: date))
self.monthComponent = Int16(Calendar.current.component(Calendar.Component.month, from: date))
self.yearComponent = Int16(Calendar.current.component(Calendar.Component.year, from: date))
```

- Split Date into it's Components
- Save Components in CoreData
- Components as Int16

Date Components

```
let request = NSFetchRequest<NSFetchRequestResult>(entityName: "Round")
request.returnsObjectsAsFaults = false
request.propertiesToFetch = [#keyPath(Round.monthComponent), #keyPath(Round.yearComponent), minDesc, maxDesc, avgDesc]
request.propertiesToGroupBy = [#keyPath(Round.monthComponent), #keyPath(Round.yearComponent)]
request.resultType = .dictionaryResultType
```

- Group by two attributes

Date Components

```
result = try viewContext.fetch(request)
guard let array = result as? [NSDictionary] else { return }

///connvert components into Date again
///chartData: [(date: Date, min: Double, max: Double, avg: Double)]
chartData = array.map({
    let components = DateComponents(year: $0["yearComponent"] as? Int,
                                    month: $0["monthComponent"] as? Int,
                                    day: 12, hour: 12, minute: 0, second: 0)
    let date = Calendar.current.date(from: components) ?? Date()
    return (date: date,
            min: $0["min"] as! Double,
            max: $0["max"] as! Double,
            avg: $0["average"] as! Double )
})
```

- Build Date from Components

Date Components

Pro

- No real redundancy
- Build a Date for Chart
- Grouping by Integers

Conclusion


Average distance per Club
200m
150m
100m
50m
0m
7 iron
5 hybrid
Driver

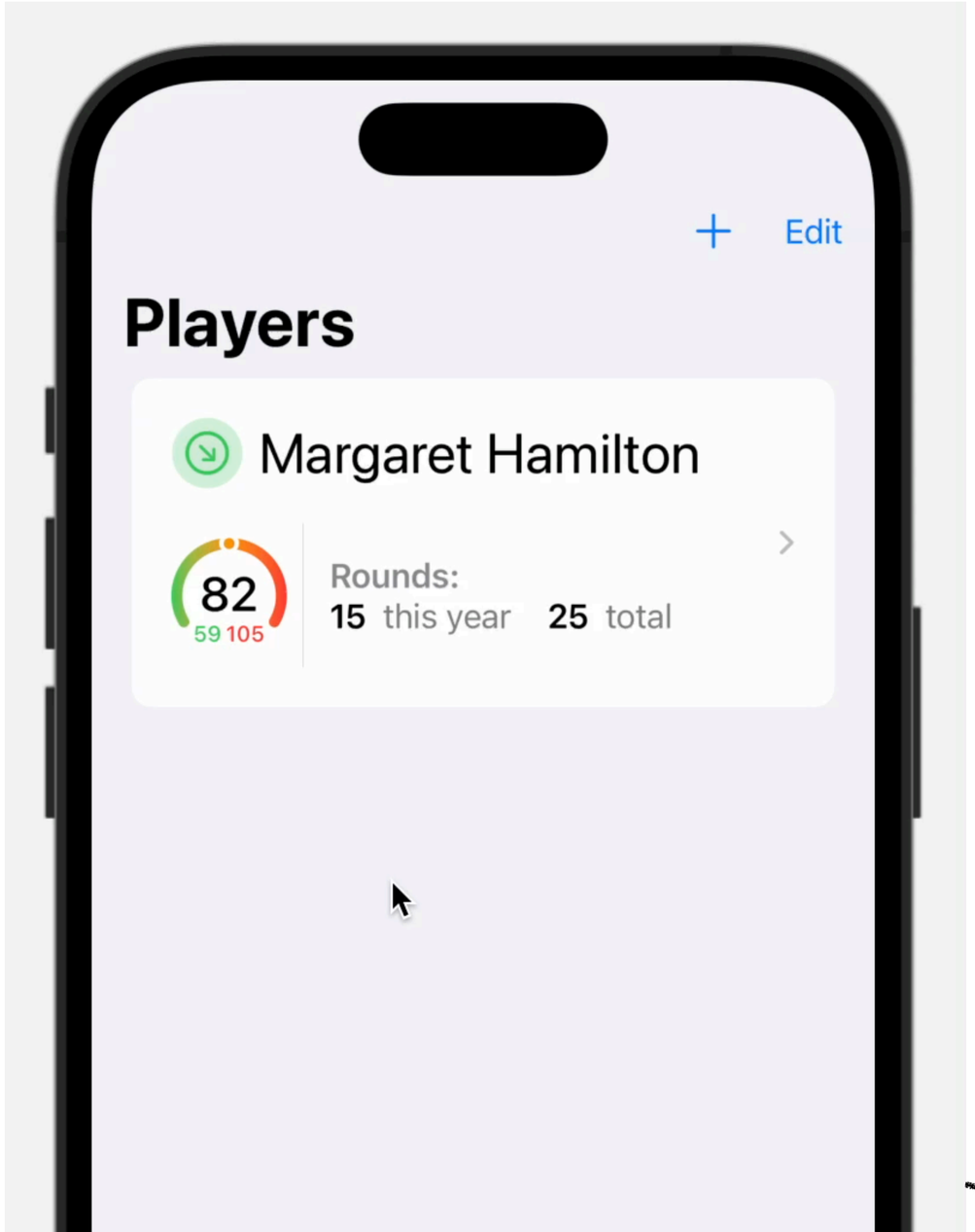

Edit
Players
Margaret Hamilton
85
59 109
Rounds:
18 this year
25 total


Monthly Scores
120
110
100
90
80
70
60
50
N
D
J
F
M
A
M
J
J
A

Conclusion

Findings

- NSExpression offers some functionality out of the box
- Map result for easy use in Charts
- Missing Date separation makes working with dates complicated
- Documentation may not help you
- Swift Data doesn’t seem to offer NSExpression functionality


Edit
Players
Margaret Hamilton
82
59 105
Rounds:
15 this year
25 total

Questions?

Thank you for listening!