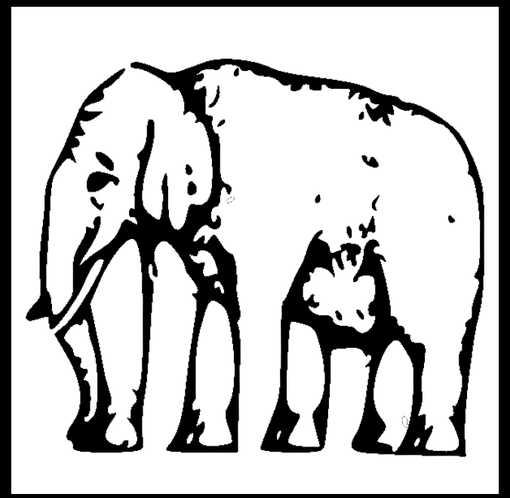
Clean Architecture in production

Lessons learned



@m_mlr

Clean Architecture in production

- Clean Architecture?
- Demo anhand einer veröffentlichten App
- Do's
- Don'ts

Rückschau

Clean Architecture by Uncle Bob

Clean architecture cocoaheads_le talk von 2014

Screaming intent

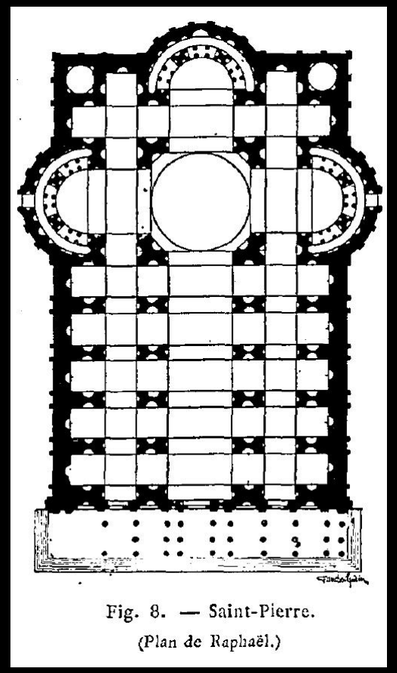
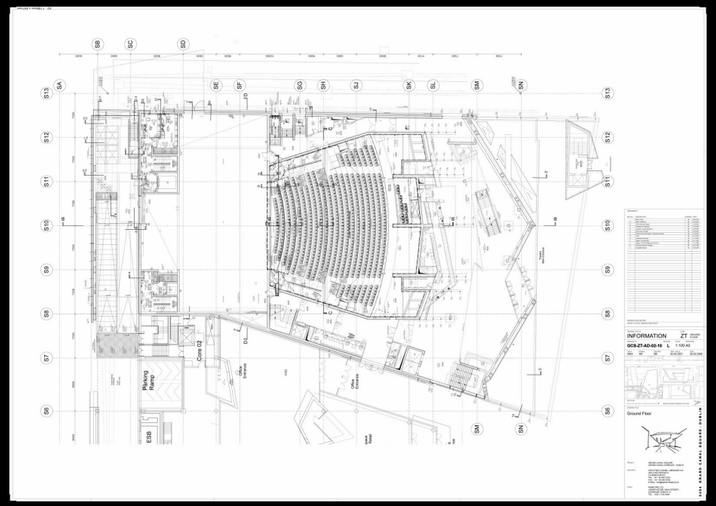
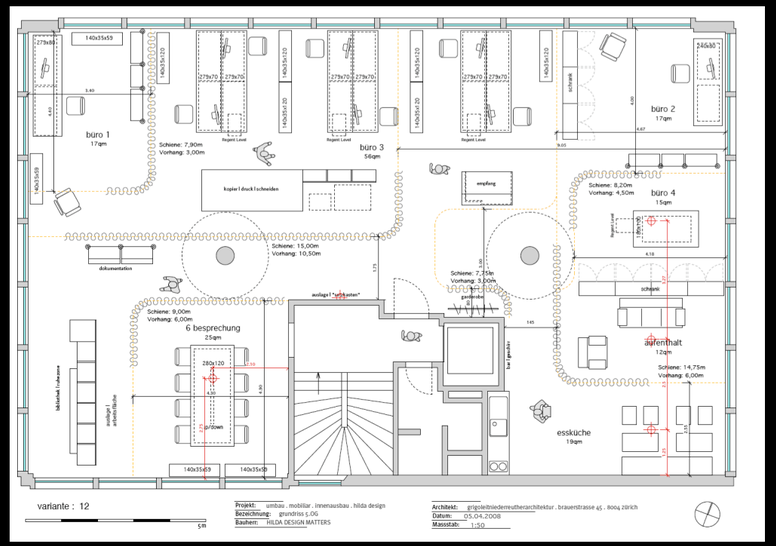
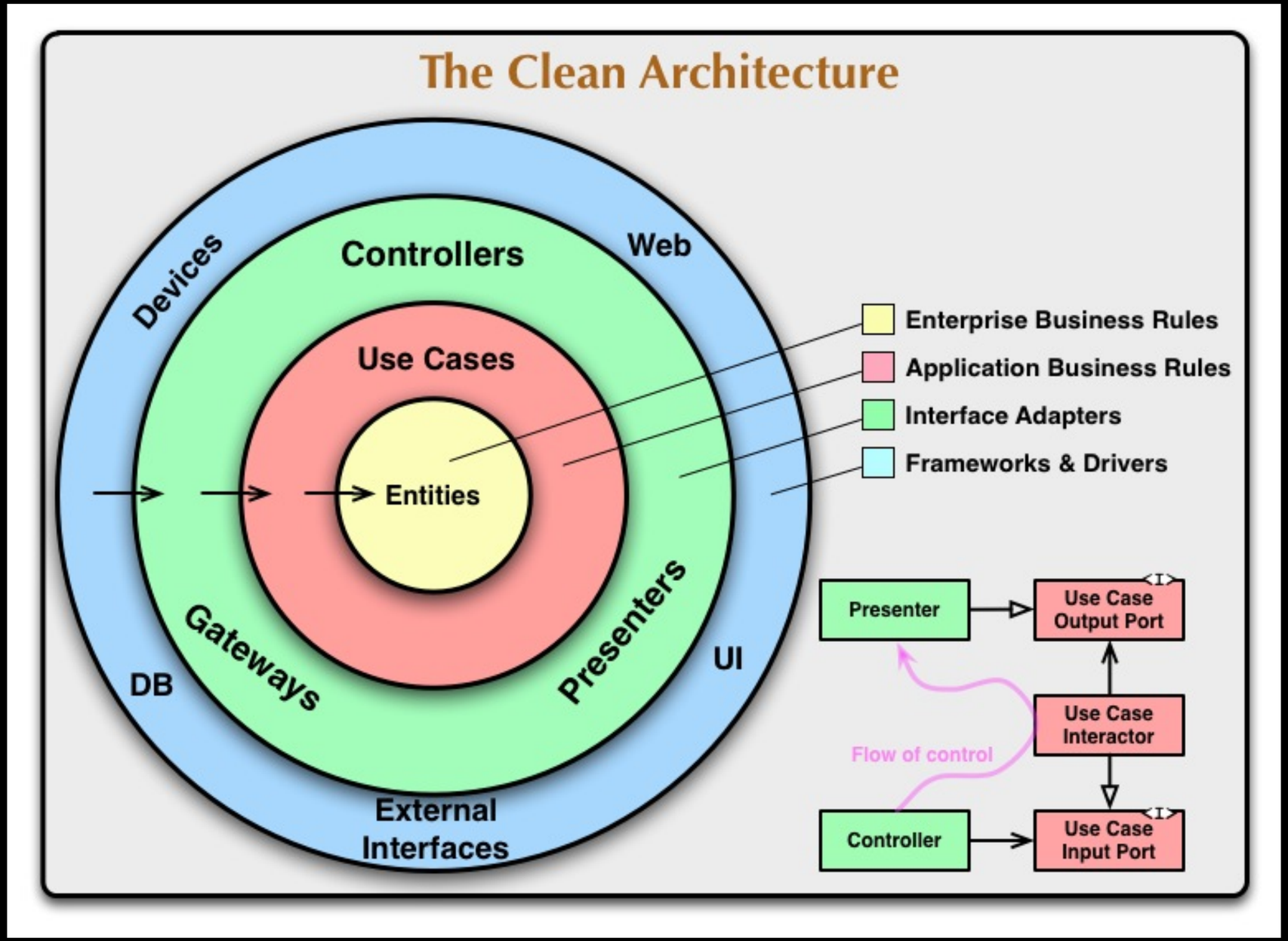


Fig. 8. — Saint-Pierre.
(Plan de Raphaël.)

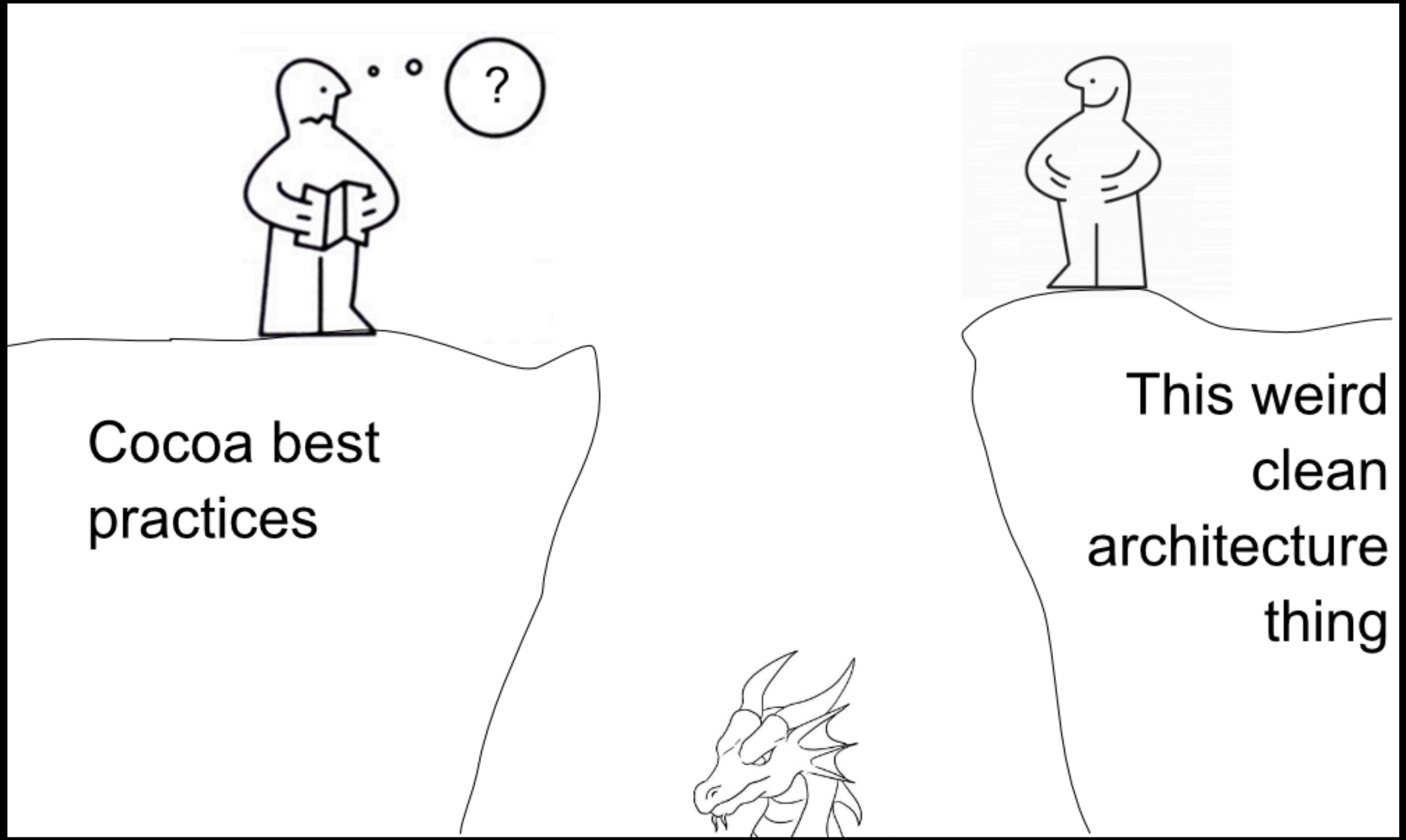





The Clean Architecture
Devices
Controllers
Web
Use Cases
Entities
Gateways
Presenters
DB
UI
External
Interfaces
Enterprise Business Rules
Application Business Rules
Interface Adapters
Frameworks & Drivers
Presenter
Use Case
Output Port
<I>
Use Case
Interactor
Flow of control
Controller
Use Case
Input Port
<I>

Wozu überhaupt Clean Architecture?

(Hint: Unabhängigkeit der Geschäftslogik von Frameworks, Datenbanken & UIs, klare Zuständigkeiten, Testing, "saubere" View Controller...)


?
Cocoa best practices
This weird clean architecture thing

Ins Netz geschaut...

Unzählige architecture talks:

Massive View Controller™

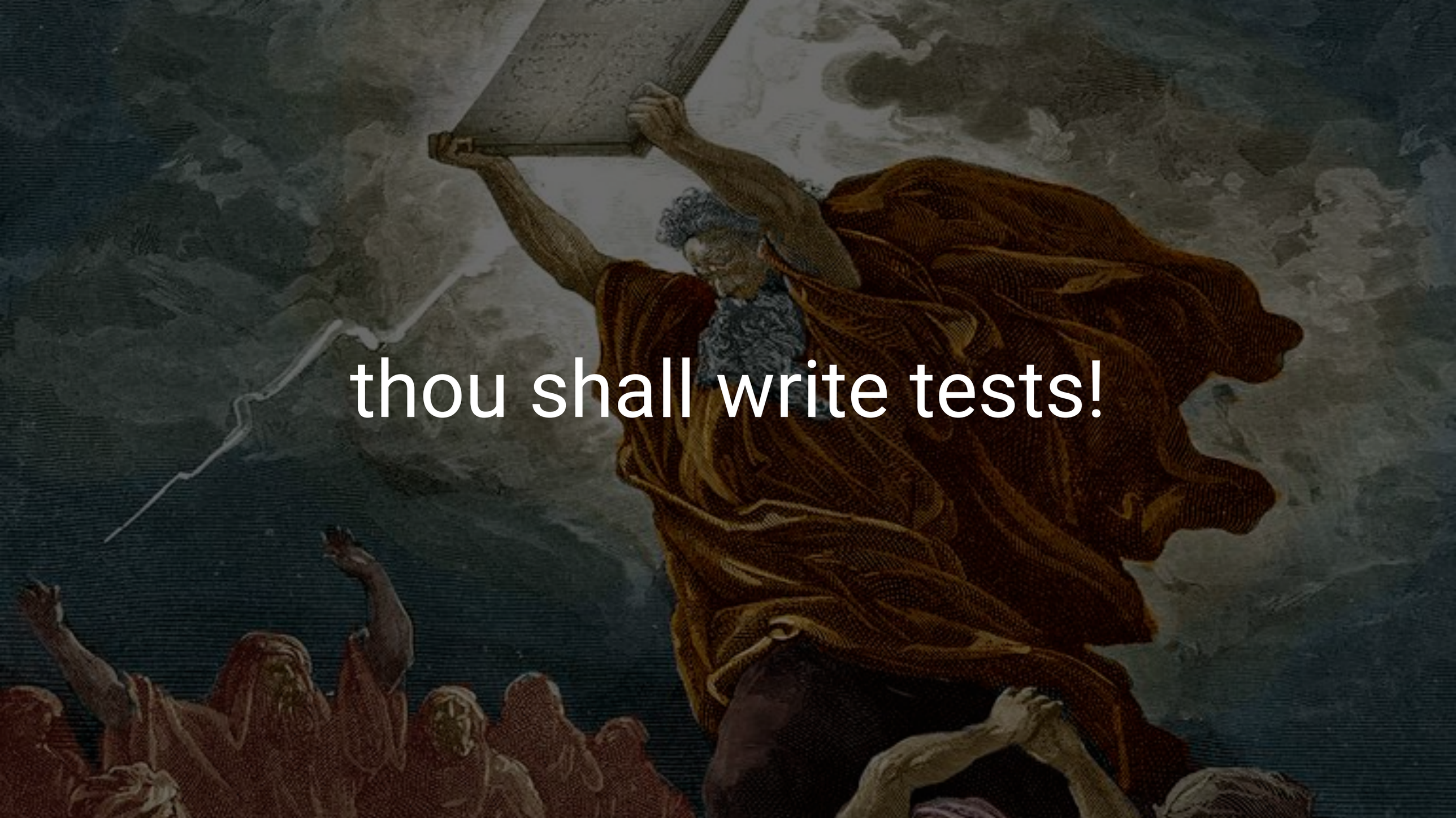
MVP

MVVM

VIPER

REACT-

-WHATEVER


thou shall write tests!

Fake Twitter Clients & To-Do-List Examples schön und gut, aber...

"Der Unterschied zwischen Theorie und Praxis ist in der Praxis weit höher als in der Theorie."

— Ernst Ferstl

Johann Sebastian Bach (1685-1750)

Demo

Bach Kantaten Referenz App

- Clean Architecture (VIPER inspiriert)
- Test driven entwickelt (> 2.500 unit tests)
- Cocoa MVC in der View-Schicht
- isolierter Anwendungskern (iOS/Mac agnostisch)

Bach Kantaten Referenz App

- native iOS Komponenten (AutoLayout, CoreData, Spotlight, Today-Widget, On-Demand-Resources, Dynamic Type, UICollection-/UITableView basiertes UI)
- schmerzfreie Releases (mittlerweile 7 major feature updates)
- Objective-C

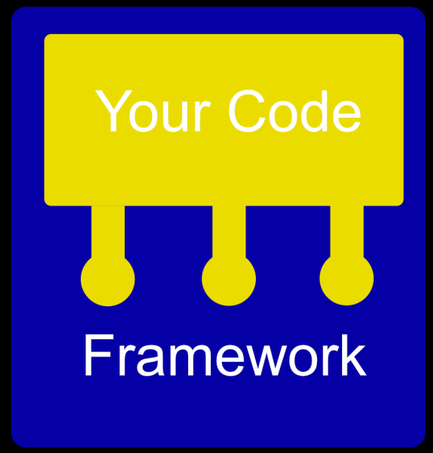
Don`t take my word...

Excellent reference app for Bach cantatas ★★★★★
This is an invaluable resource for anyone interested in Bach’s cantatas. Beautifully designed, the app is easy to use and comprehensive in its detail.

Sehr gut gemacht! ★★★★★
Bachs Kantatenwerk erschließt sich durch diese App auf vielfältige Weise. Eine fantastische Quelle! Ich bin immer wieder aufs Neue begeistert! Ich kenne nichts Vergleichbares.

Very helpful. The "Duerr" for on-the-road ★★★★★
Great App for Bach lovers and professional musicians. Worth the investment.

Frameworks & Libraries



When using a framework, the framework is in charge of running the system. It defines some extensibility points (interfaces) where you need to put your implementation.[1]

— Tomas Petricek

---

[1] http://tomasp.net/blog/2015/library-frameworks/


Medusa Lemieux
www.killerbrush.com

Frameworks are tools, not ways of life

— Uncle Bob

When using a library, you are in charge of running the system. The library defines some points through which you can access it (functions and types) and your code can call it as it needs.[1]

— Tomas Petricek

---

[1] http://tomasp.net/blog/2015/library-frameworks/

Yes, but what about...

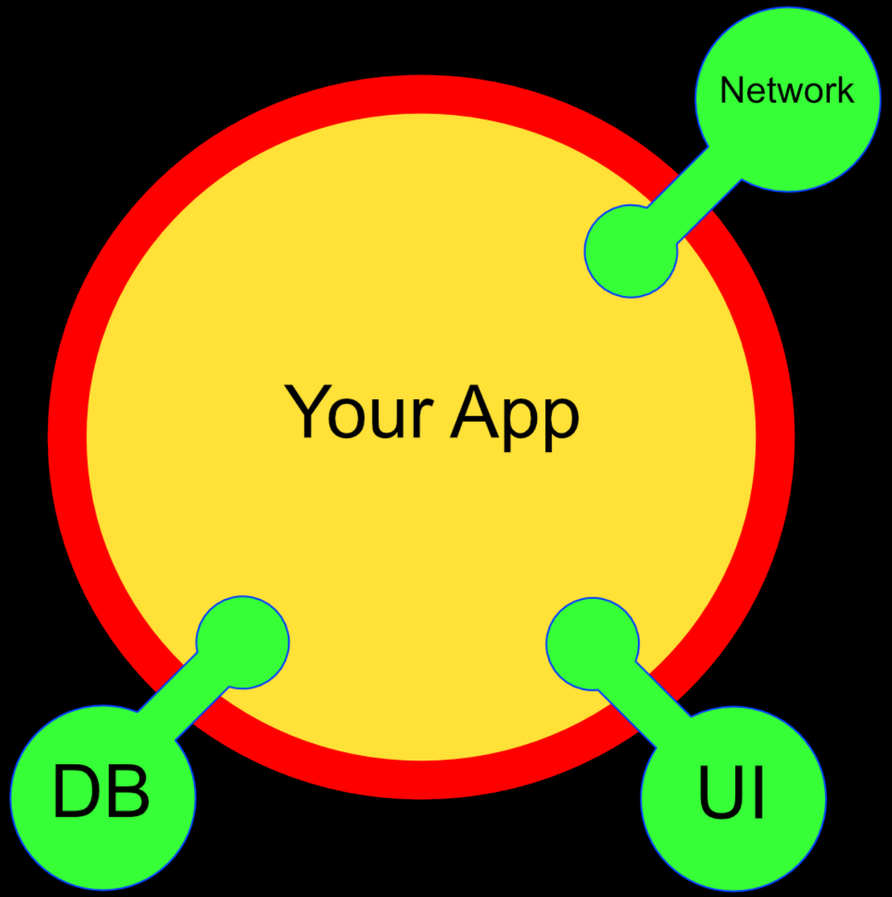
...do not fight the framework!

— meine innere Stimme

Make use of the framework but don't marry it!

— Uncle Bob

Frameworks as libraries



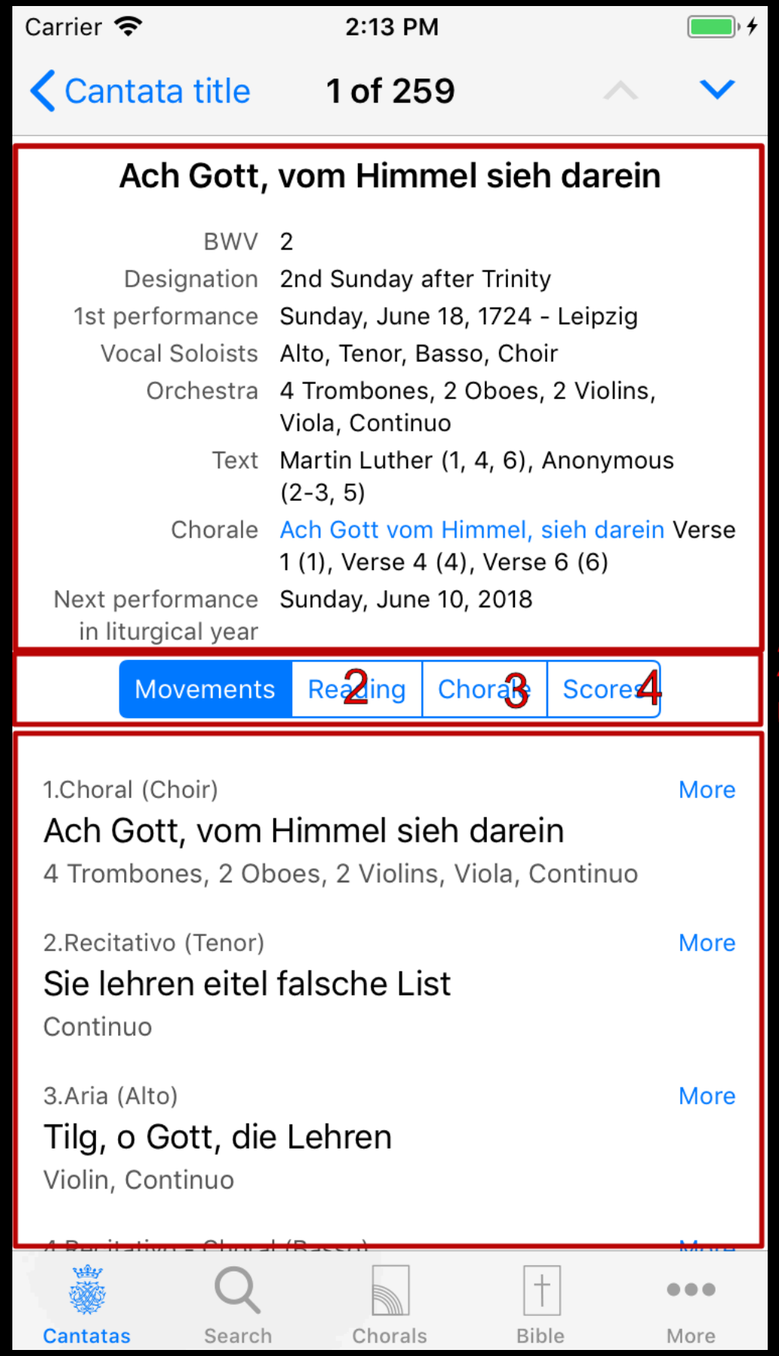

VIPER
WENGER

There are two hard things in computer science: cache invalidation, naming things, and off-by-one errors.

— Phil Karlton

- Clean architecture ist letztlich bloß ein Name für bewährte Softwareentwicklungs-Methodik (Composition, Delegation, SRP, DIP, DRY etc)
- keine den Cocoa best practices zuwider laufende Technik

Use cases

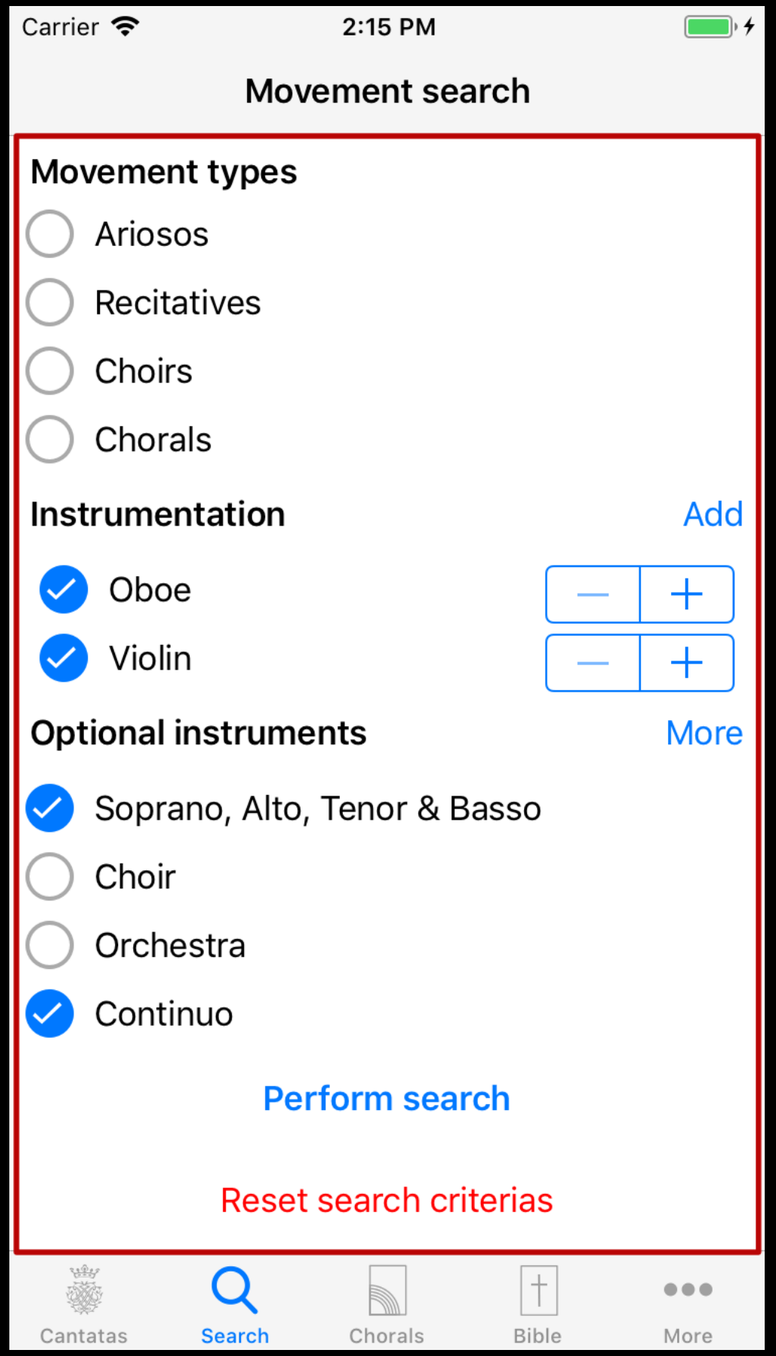


use case

2, 3 & 4 use cases

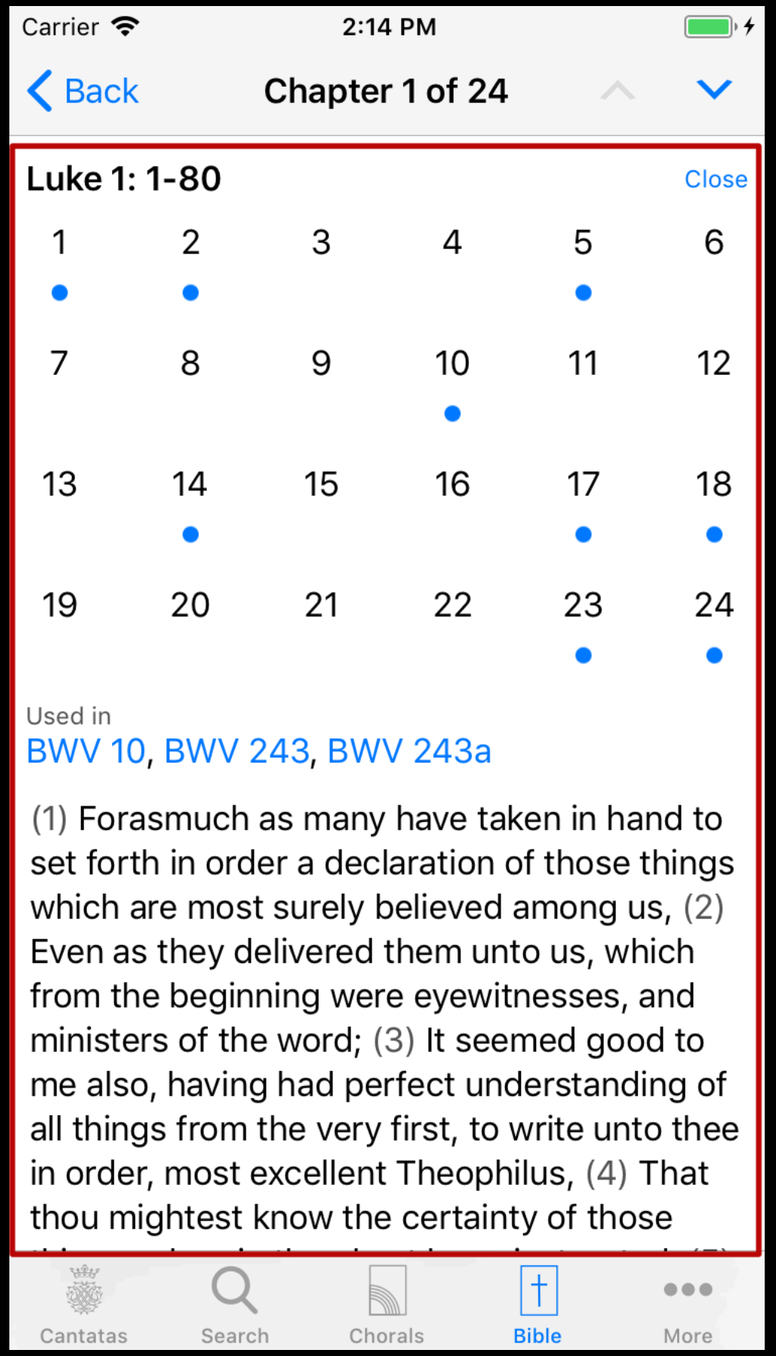
use case

```
@protocol InspectCantataInterface <NSObject>

- (void)inspectCantata:(NSString*)identifier;
- (void)inspectNext;
- (void)inspectPrevious;
- (void)noSelection;
- (void)handleURL:(NSURL *)url fromRect:(CGRect)rect;

@end
```



Use case

```
@protocol EditMovementCatalogItemInterface <NSObject>

- (void)selectMovementTypeAtIndex:(NSUInteger)theIndex;
- (void)toggleRequiredInstrumentations;
- (void)selectRequiredInstrumentationAtIndex:(NSUInteger)theIndex;
- (void)increaseRequiredInstrumentCountAtIndex:(NSUInteger)theIndex;
- (void)decreaseRequiredInstrumentCountAtIndex:(NSUInteger)theIndex;
- (void)toggleAvailableInstruments;
- (void)selectOptionalInstrumentAtIndex:(NSUInteger)theIndex;
- (void)performSearch;
- (void)reset;

@end
```



use case

```
@protocol InspectBiblicalBookInterface <NSObject, NavigatableList>

- (void)inspectBiblicalBook:(BiblicalBook)book;
- (void)toggleAvailableChapters;
- (void)selectChapterAtIndex:(NSUInteger)selectedIndex;
- (void)handleURL:(NSURL *)url fromRect:(CGRect)rect;

@end
```

VIPER

View

- UI/NSViewController
- klassisches MVC mit "dummen" View-Models
- Cocoa Views
- Humble Object

VIPER

Interactor

- eigentlicher use case, welcher die Geschäftlogik umsetzt
- isolierter Kern der App
- arbeitet mit den Entities

VI*P*ER

Presenter

- Kümmert sich um alle mit View-Models verbundene Aufgaben (Erzeugung, Umrechnung von NSIndexPath etc, Lokalization)
- Formatierung von Listen, Text und Datum etc.
- NSFormatter/NSValueTransformer

VIP***E***R

Entities

- anwendungsübergreifende Geschäftslogik
- wird einzig von Interaktoren verwendet
- nicht zu verwechseln mit CoreData- oder ORM-Entities

VIPE***R***

Router

- implementiert das jeweilige Modulinterface
- dirigiert den control flow eines use cases zw. Interactor, View, Presenter
- löst auch UI Änderungen aus wie etwa Navigation zum nächsten Screen

...one more thing!

Wireframe

```
@protocol Wireframe <NSObject>

- (UIViewController*)instantiateViewController;

@end
```

- fasst alle für einen use case benötigten Objekte zusammen
- instanziert und verwaltet die Viewcontroller

Zusammenfassung

- eigener App code ohne Abhängigkeit zu Framework-Details
- Tell, don't ask!
- klare Zuständigkeiten
- Kommunikation über definitierte Boundaries mittels value objects

"Protocol oriented programming"

— hip phrase of the week

a.k.a. Dependency inversion principle:

„Abstraktionen sollten nicht von Details abhängen. Details sollten von Abstraktionen abhängen.“

*— SOLI**D** principles*

Show me the code!

Boundaries

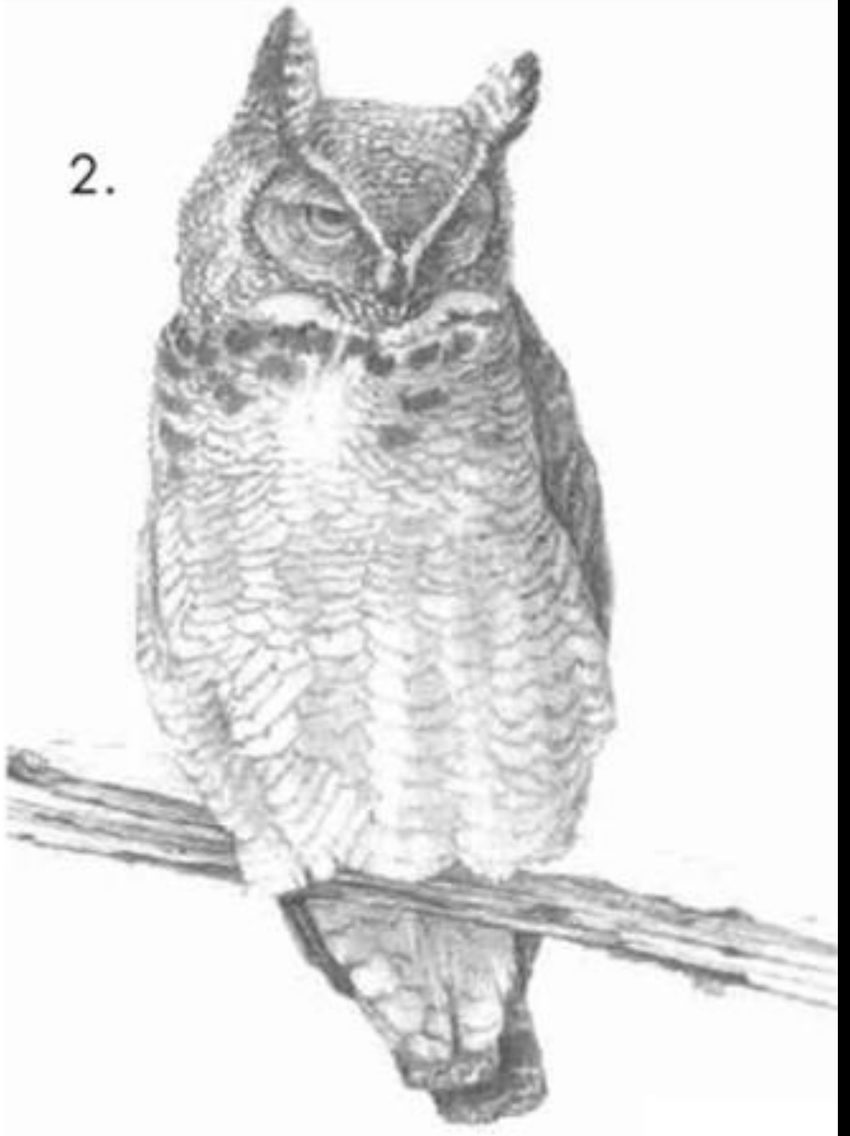
The UI is a delivery mechanism and the database is a detail.

— Uncle Bob Martin

How to draw an owl

1.

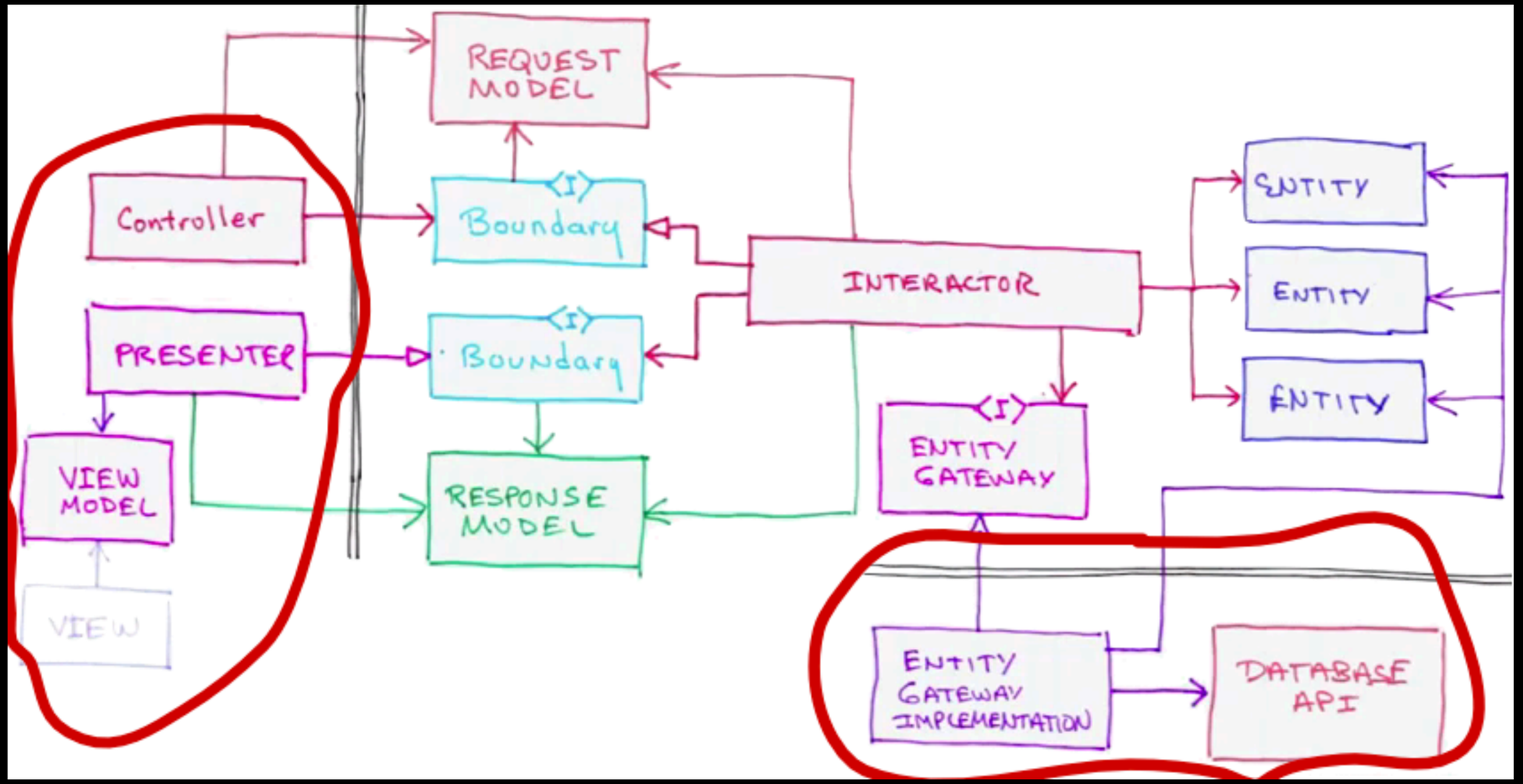
2.



1. Draw some circles

2. Draw the rest of the fucking owl

The rest of the fucking owl



Viewmodels

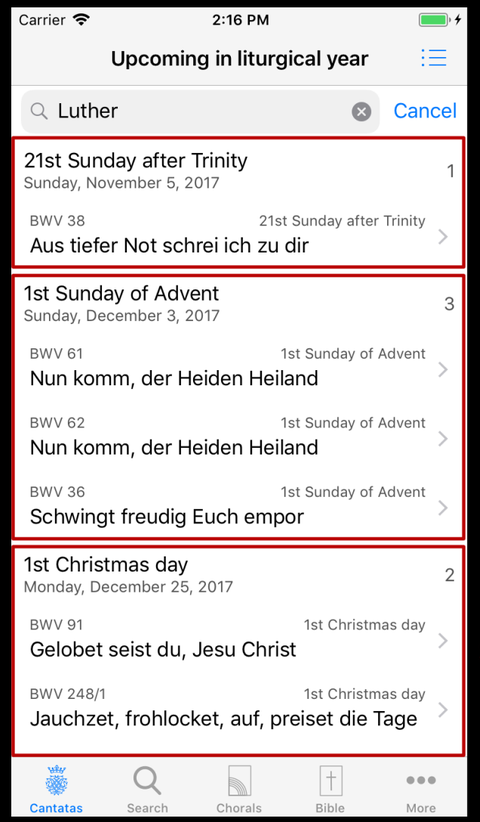
- "Endprodukt" eines use case
- Anfangspunkt für einen ViewController
- simple data structures ohne Verhalten
- M in ***M***VC
- **value semantics**

```
@protocol MKMTextualContentItem <NSObject>

@property (nonatomic, copy, readonly) NSString *primaryContent;
@property (nonatomic, copy, readonly) NSString *secondaryContent;
@property (nonatomic, copy, readonly) NSString *primarySupplementalContent;
@property (nonatomic, copy, readonly) NSString *secondarySupplementalContent;

@end
```

```objc
@protocol MKMSectionInfo <NSObject>

@property (nonatomic, readonly, copy) NSString *identifier;
@property (nonatomic, readonly, copy) NSArray *items;

@end

@protocol MKMSectionedListViewModel <NSObject>

@property (nonatomic, readonly, copy) NSString *identifier;
@property (nonatomic, readonly, copy) NSArray<id<MKMSectionInfo>> *sections;

@end
```

```
@interface MKMItemViewModel : NSObject <MKMTextualContentItem>
@end

@interface MovementViewModel : MKMItemViewModel

@property(nonatomic, copy, readonly) NSAttributedString *originalParodyContent;

@end
```

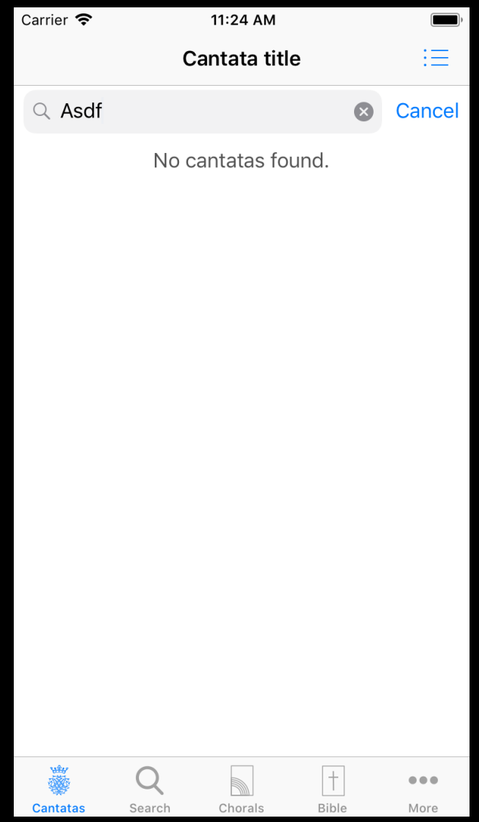
Klassischer UITableView/UICollectionView Aufbau



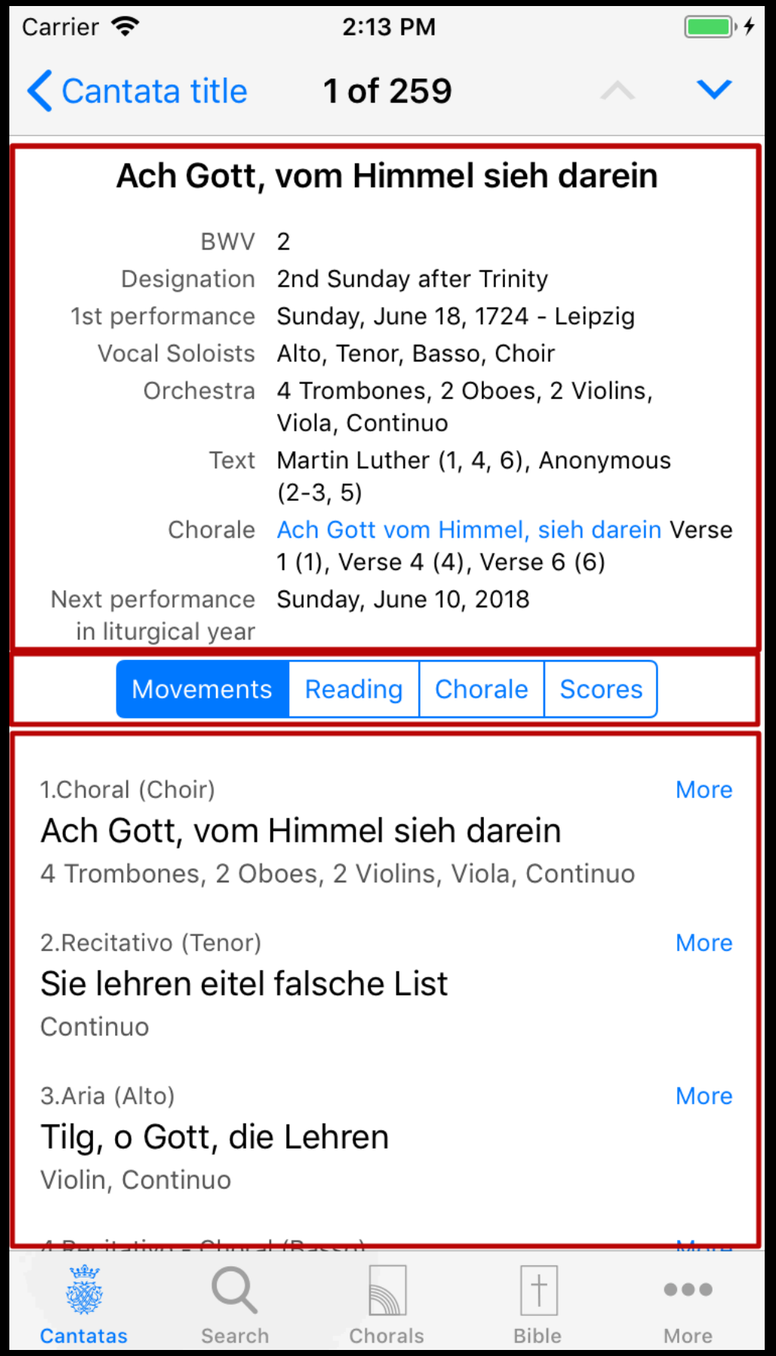
Section

Section

Section




```
@protocol CantatasListView <NSObject>
- (void)updateCantatasList:(id<MKMSectionedListViewModel>)viewModel;
- (void)showNoContentPlaceholder:(id<MKMTextualContentItem>)viewModel;
@end
```

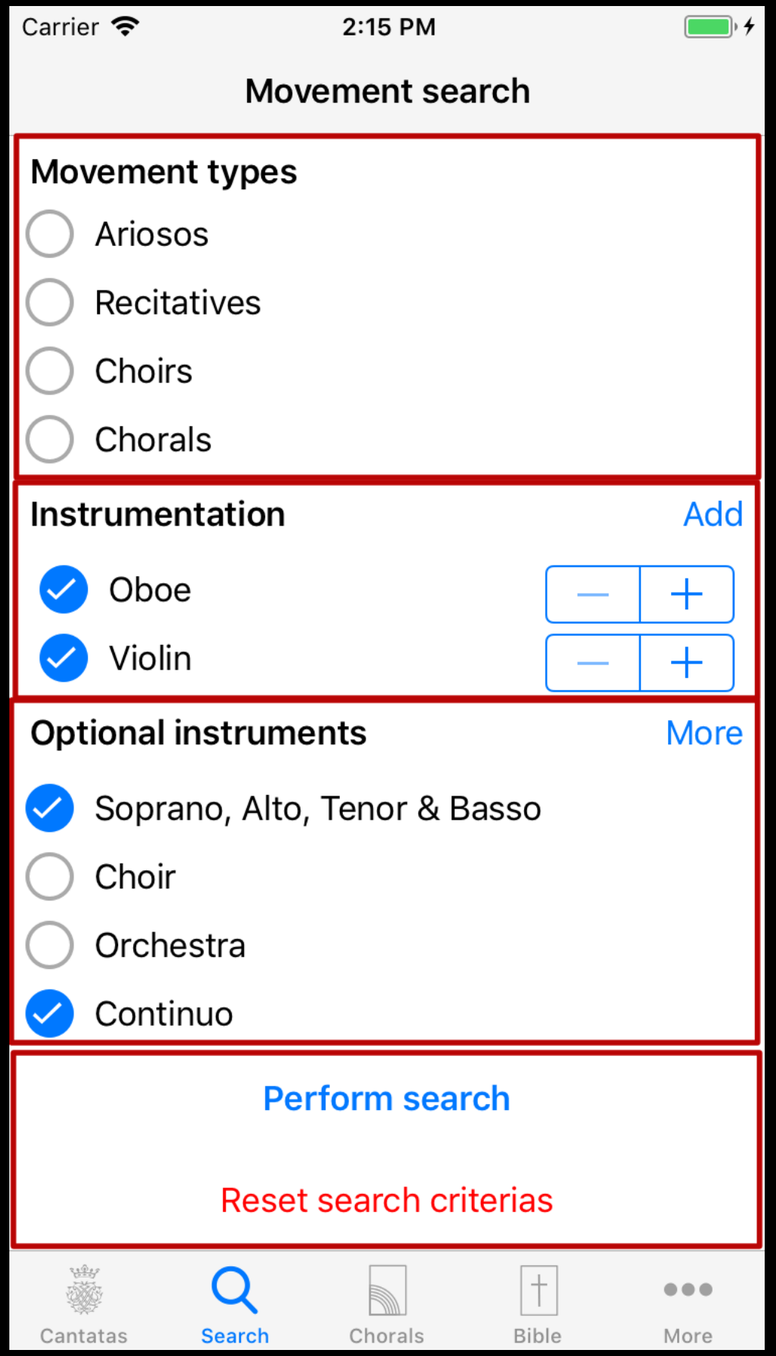


Section

Section

Section

```
@protocol InspectCantataView <NSObject>
- (void)updateCantataDetails:(MKMListViewModel*)viewModel;
- (void)showNoSelectionPlaceholder:(id<MKMTextualContentItem>)viewModel;
@end

@protocol CantataMovementsView <NSObject>
- (void)updateMovements:(id<MKMSectionedListViewModel>)viewModel;
- (void)updateMovementsViewWithNoContentViewModel:(id <MKMTextualContentItem>)
noContentViewModel;
@end
```



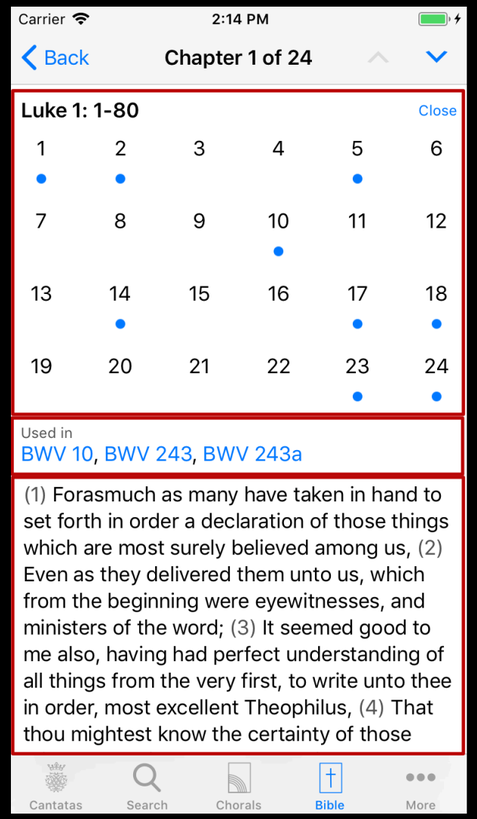
Section

Section

Section

Section

```
@protocol EditMovementCatalogItemView <NSObject>

- (void)updateMovementTypesView:(MKMListViewModel *)viewModel;
- (void)updateRequiredInstrumentsView:(MKMListViewModel*)viewModel;
- (void)updateOptionalInstrumentsView:(MKMListViewModel *)viewModel;
- (void)showEmptyInstrumentsPlaceholder:(MKMItemViewModel*)viewModel;

@end
```



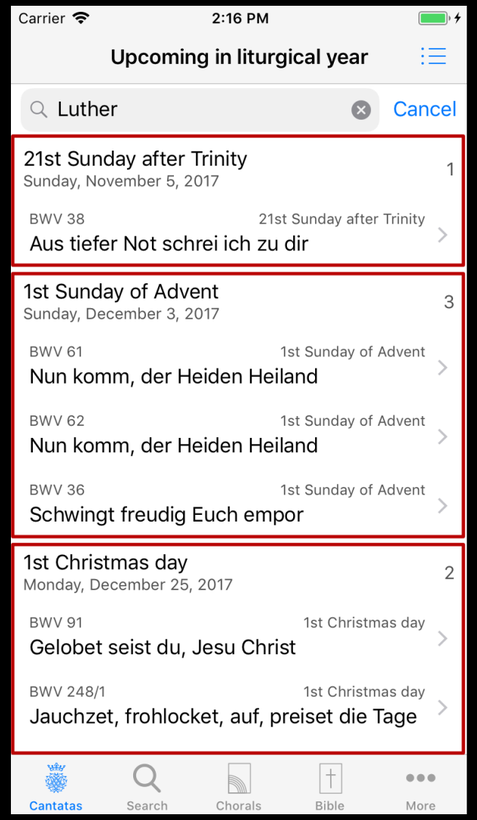
Section

Section

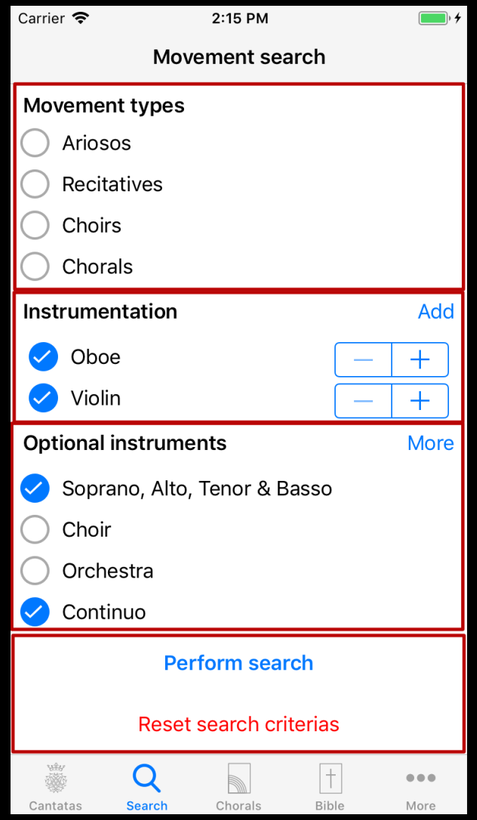
Section

```
@protocol InspectBiblicalBookView <NSObject>
- (void)updateBiblicalBookAvailableChapters:(id <MKMSectionInfo, MKMTextualContentItem>)chapterViewModel;
- (void)updateBiblicalBookDetailViewModel:(id<MKMSectionInfo, MKMTextualContentItem>)detailViewModel;
- (void)updateBiblicalBookVerses:(NSAttributedString *)viewModel;
- (void)showNoChaptersAvailablePlaceholder:(id<MKMTextualContentItem>)viewModel;
@end
```

Liste von vielen gleichförmigen Cells über mehrere Sektionen



Sektionen mit unterschiedlichen Cells

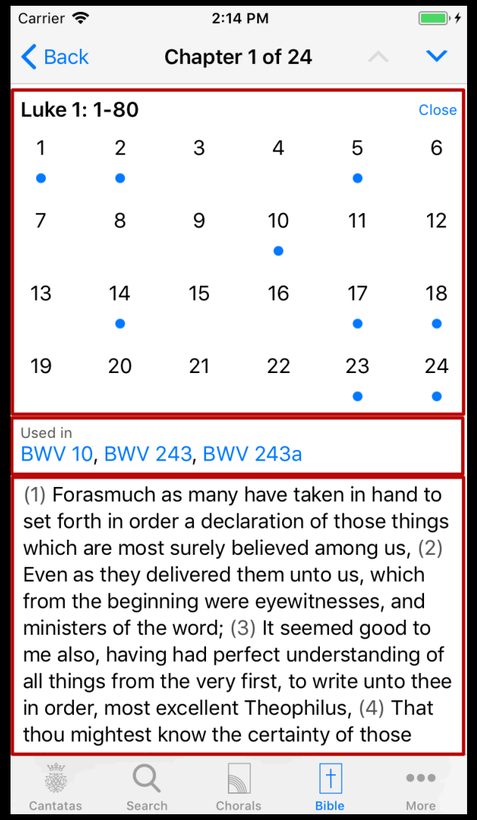


Array of items

Array of items

Array of items

Array of items



Array of items

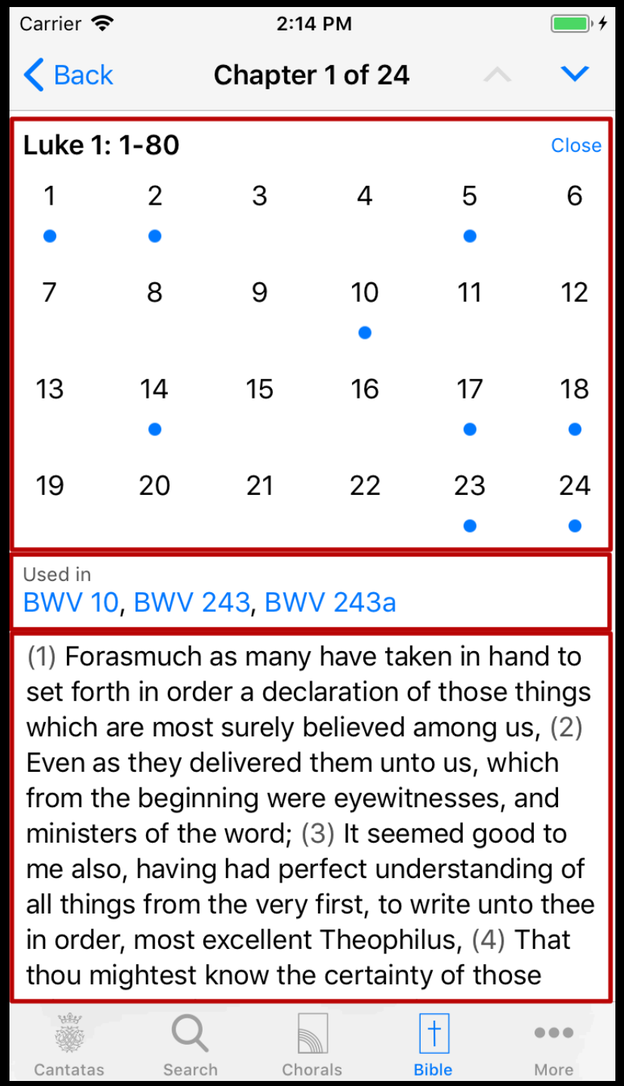
Item

Item

Datasources

MVC as your grandparents knew it

```
@protocol MKMDataSource <NSObject>

@property (nonatomic, readonly) NSUInteger numberOfSections;

- (NSUInteger)numberOfItemsInSection:(NSUInteger)section;

- (id)objectAtIndexPath:(NSIndexPath*)indexPath;

- (id<MKMSectionInfo>)infoForSection:(NSUInteger)section;

- (void)selectItemAtIndexPath:(NSIndexPath*)indexPath;

@end
```



MKMArrayDataSource

MKMItemDataSource

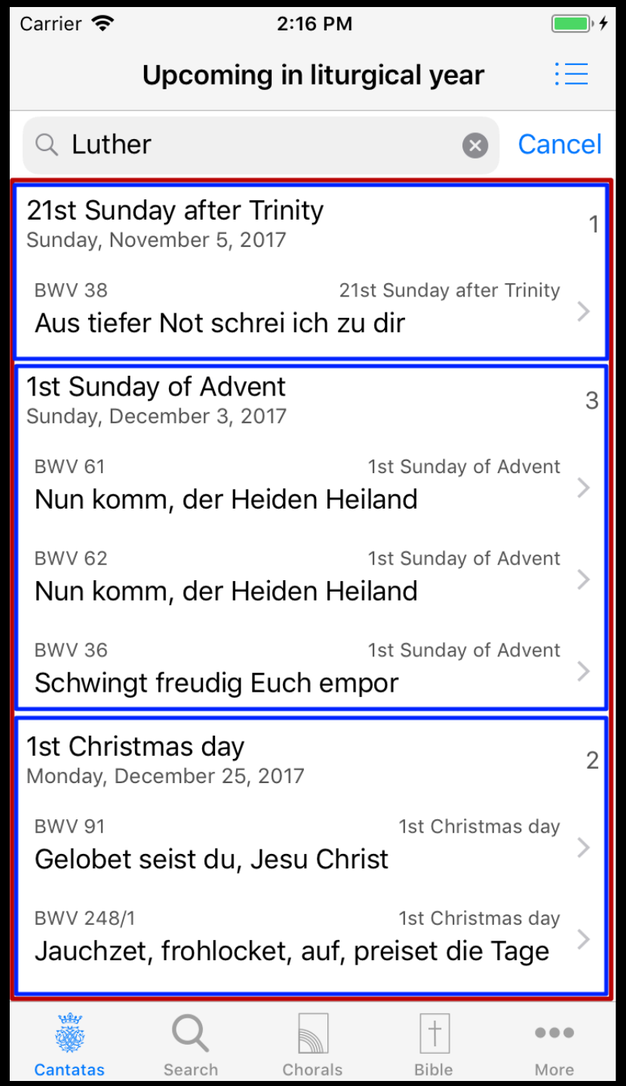
MKMItemDataSource

MKMItemDataSouce

- single item

MKMArrayDataSource

- array of items

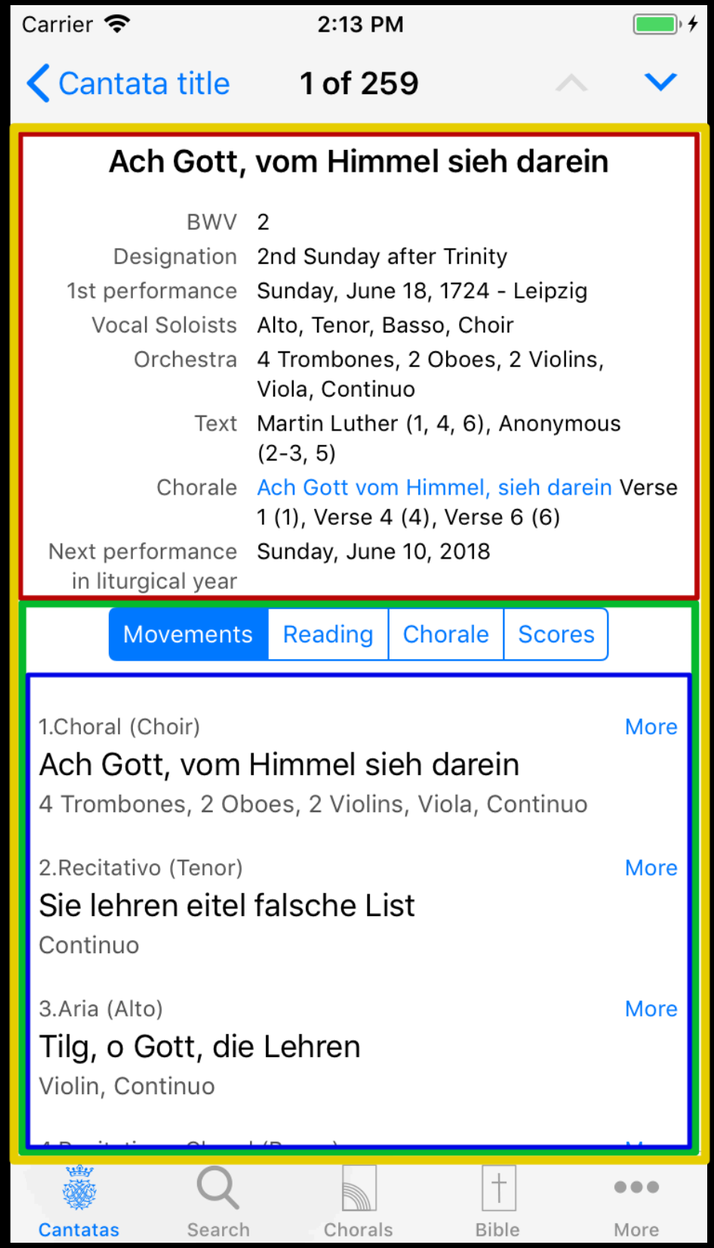


MKMSectionedDataSource

MKMSectionedDataSource

- array of array of similar items

Datasource Composition



MKMCombinedDataSource

MKMArrayDataSource

MKMSegmentedDataSource

MKMSectionedDataSource

MKMSegmentedDataSource

- mehrere child datasources
- jeweils selektierte wird angezeigt

MKMCombinedDataSource

- mehrere child datasources
- alle werden gleichzeitig untereinander angezeigt

Datasources & View-Models

- Datasources verwalten die view models
- View models sind das ***M*** in ***M***VC
- berechnen diffs für dynamische UITableView/UICollectionView updates (view models haben value sematic)
- Eigene, reguläre NSObjects ohne Abhängigkeit zu UIKit/AppKit -> Unabhängigkeit vom main thread

Simple view controller

```
// InspectCantataTableViewController

- (void)updateCantataDetails:(id<MKMSectionInfo>)viewModel {
    self.detailDataSource.content = viewModel;
}

- (void)updateMovements:(id<MKMSectionedListViewModel>)viewModel {
    self.movementsDataSource.sections = viewModel.sections;
}

- (void)showNoSelectionPlaceholder:(id<MKMTextualContentItem>)viewModel {
    [self.combinedDataSource setEmptyContentItem:viewModel];
}
```

Layout

Layout

- Beschreibung des Layouts pro Sektion
- Cell & Header (class oder nib)

```objc
@interface MKMSectionMetrics : NSObject

@property (nonatomic) CGFloat rowHeight;
@property (nonatomic) CGFloat headerHeight;
@property (nonatomic, readonly) Class cellClass;
@property (nonatomic, readonly) Class headerClass;
@property (nonatomic, strong) UIColor *backgroundColor;
@property (nonatomic, strong) UIColor *headerBackgroundColor;
@property (nonatomic, copy) MKMHeaderConfigurationBlock headerConfigurationHandler;
@property (nonatomic, copy) MKMCellConfigurationBlock cellConfigurationHandler;
@property (nonatomic, readonly) UINib *cellNib;
@property (nonatomic, readonly) UINib *headerNib;
@property(nonatomic) CGSize itemSize;

@end
```

```
@protocol MKMSectionLayoutProvider <NSObject>

- (MKMSectionMetrics *)metricsForSection:(NSUInteger)section
contentState:(MKMDataSourceState)contentState;

@end
```

Jetzt wächst zusammen, was zusammen gehört

— Ein Altkanzler

MKMDataSourceUITableViewAdapter

```
@interface MKMDataSourceUITableViewAdapter : NSObject

@property (nonatomic, weak) UITableView *tableView;

- (instancetype)initWithDataSource:(id<MKMDataSource>)datasource
layoutMetricsProvider:(id<MKMSectionLayoutProvider>)metricsProvider;

@end
```

- implementiert UITableViewDataSource & UITableViewDelegate

MKMDataSourceUICollectionViewAdapter

```
@interface MKMDataSourceUICollectionViewAdapter : NSObject

@property (nonatomic, weak) UICollectionView *collectionView;

- (instancetype)initWithDataSource:(id<MKMDataSource>)datasource
layoutMetricsProvider:(id<MKMSectionLayoutProvider>)metricsProvider;

@end
```

- implementiert UICollectionViewDataSource, UICollectionViewDelegate & UICollectionViewDelegateFlowLayout

Zusammenfassung Viewschicht

- Composition von datasources ermöglicht komplexe, dynamische UIs
- Trennung von Daten und Layout
- Diffing der view models durch die datasources
- Datasources können mit jeglichen List-Views verwendet werden
- Benutzeraktionen werden an das Modulinterface (Router) weitergeleitet

Repositories

The database is a detail

Tell, don't ask!

```
@protocol InspectCantataRepository <NSObject>

- (void)cantataWithOpus:(NSString*)opus completionBlock:(void(^)(Cantata*))completionHandler;

@end
```

```
@protocol CantatasListRepository <NSObject>

- (void)executeCantatasListRequest:(CantatasListRepositoryRequest*)request
completionBlock:(void(^)(NSArray<Cantata*>*))completionHandler;

@end
```

Humble object pattern

Code?

Do's

Know your tools

GUIDE TO YOUR TOOLBOX

Flat Screwdriver

For opening tins of paint.

Phillips Head Screwdriver

Use in conjunction with the Hammer to puncture holes in wood.

Chisel

Use to scrape dried paint off windows, work tops and sinks.

Spanner

Backup Hammer.

Pinhead Nail

Use liberally in all projects. Remember DIY Rule #1 - if in doubt stick some, or more, nails in it.

Tough Nail

Use when you run out of Pinhead Nails

Hammer

Use for driving nails, tapping glass to check for cracks and levelling putty around windows.

Pliers

Use to twist nails that have hit a “Bastard Knot” into decorative shapes that disguise the problem.

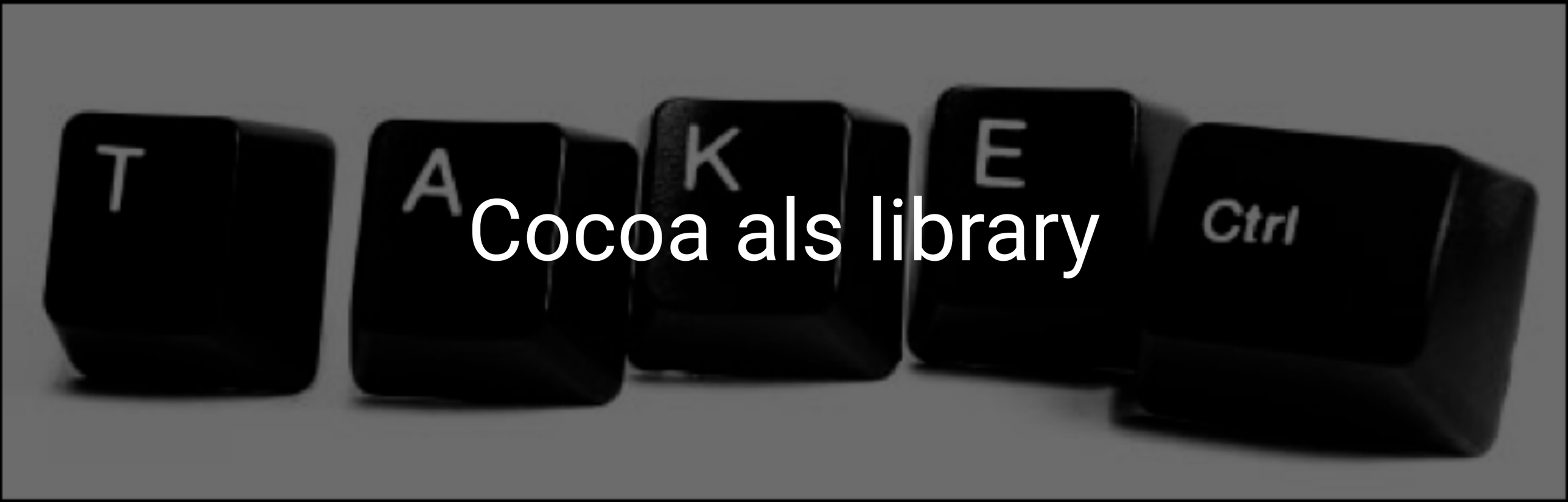
Long Nose Pliers

Use to pick up creepy crawlies that have made a home in your toolbox.

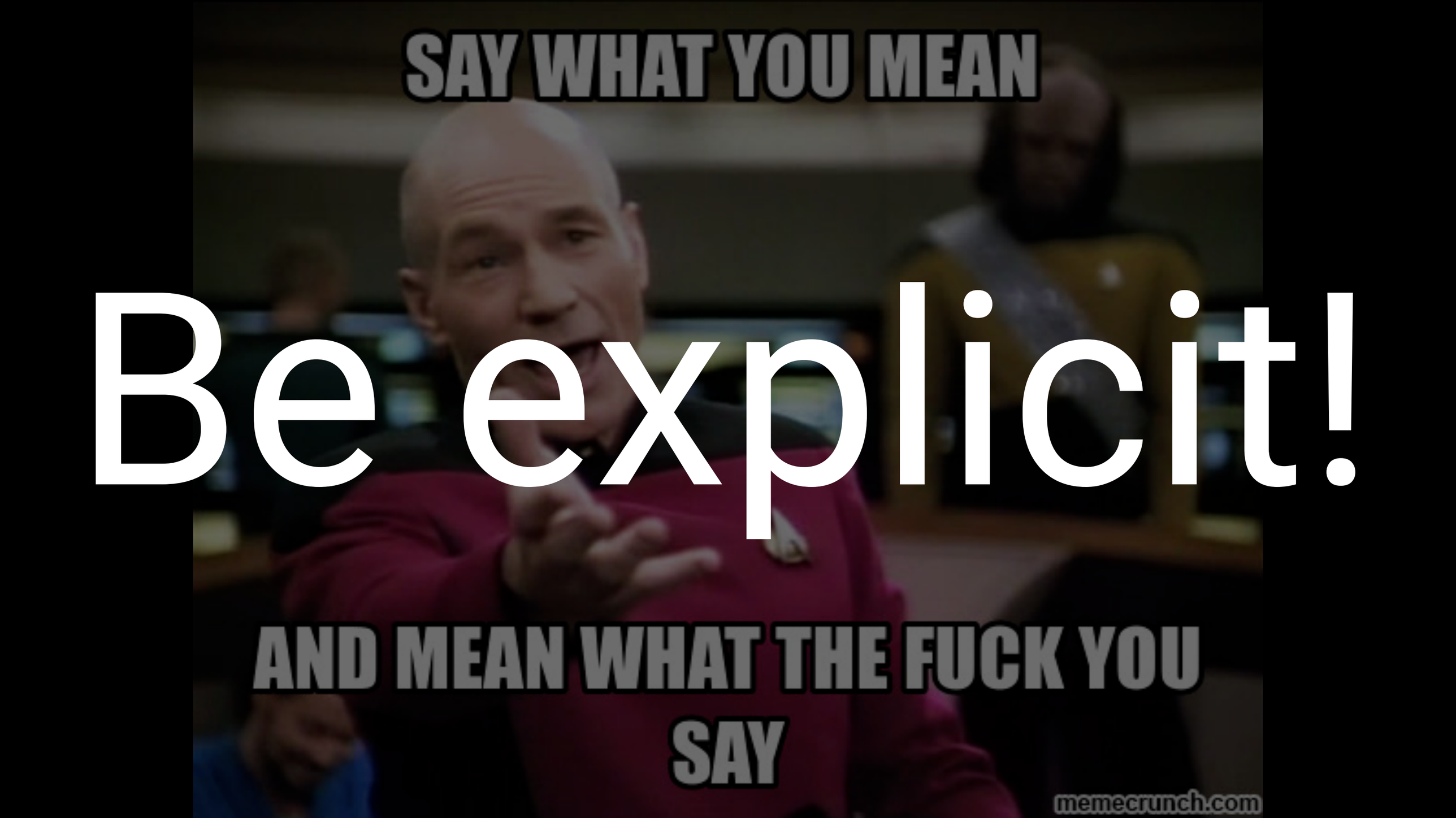
Auf Bewährtes zurückgreifen

Composition, Delegation, Cocoa Design patterns

durch die Architektur gewonnene Freiräume nutzen



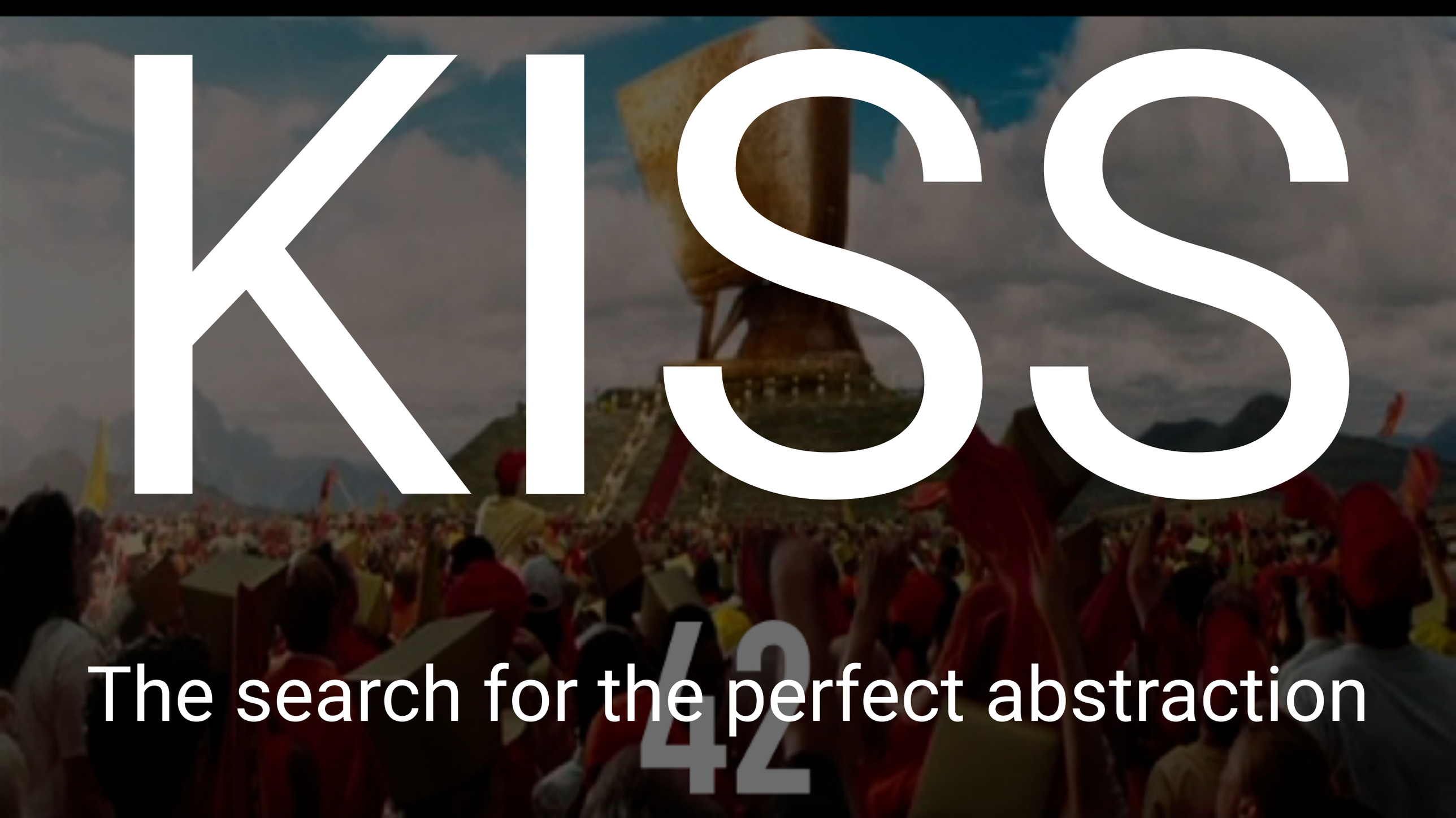
Cocoa als library


SAY WHAT YOU MEAN
AND MEAN WHAT THE FUCK YOU SAY
memecrunch.com

Be explicit!

Don'ts

Do not marry the framework!


KISS
The search for the perfect abstraction
42

Boundaries everywhere

know their costs!

ALLES wrappen

Foundation is your friend

Links

- Philippe Herreweghe & Collegium Vocale Ghent
- Advanced User Interfaces with Collection Views
- Framework Bound
- Library patterns - Why frameworks are evil
- Bach Kantaten App

Danke