mongoDB.

Combine-ing MongoDB Realm with SwiftUI v2 ✨

Ben Chatelain | Chief iOS Engineer | Kaiser Permanente | @phatblat

Task Tracker App

Remix of the Classic Tutorial

To try out this new version of MongoDB Realm I followed the updated Task Tracker tutorial, but with a twist: SwiftUI instead of UIKit.

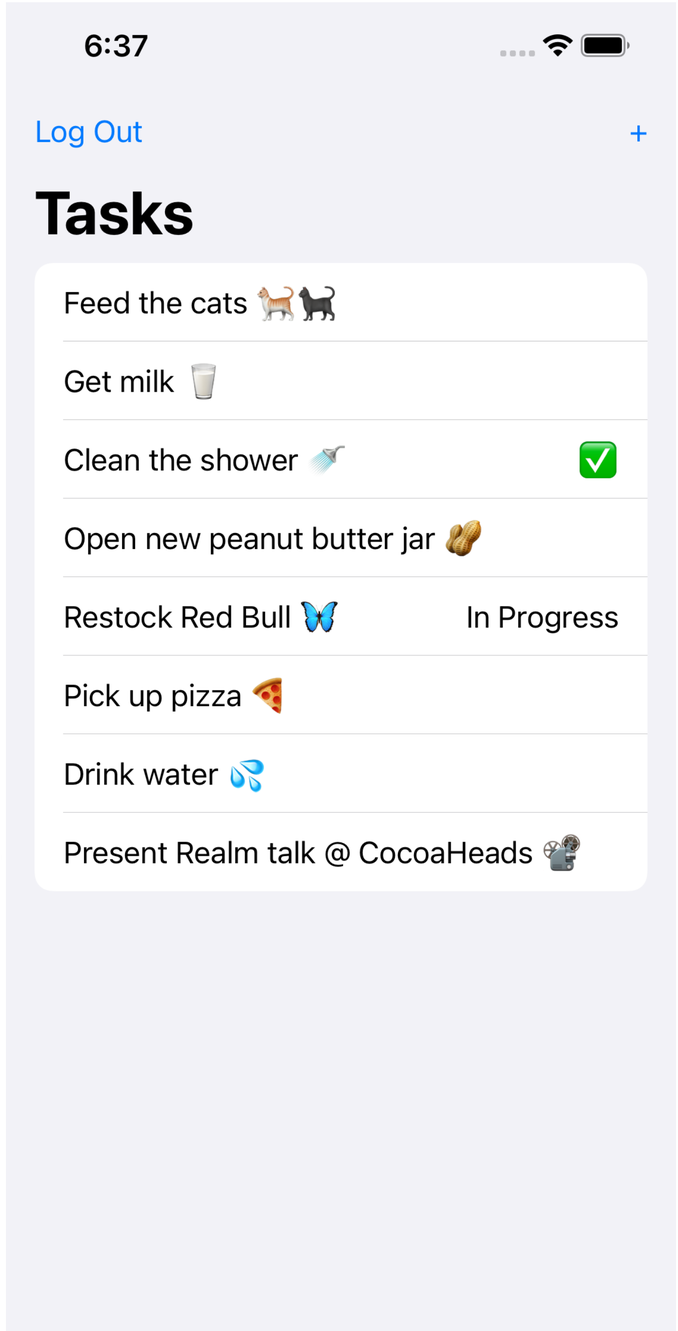
Early Success, then Headaches

Back end configured in Atlas, I had app working within a few hours. However, deletes on the client or server caused the app to crash. Eventually, I found workarounds to issues.

RealmSwift Improvements

Over the last year, the Realm SDK added Combine support and property wrappers to simplify working with SwiftUI.

✅ Tasks



Add Task

Emoji highly encouraged 🎉

Task Status

Tap row to change

github.com/phatblat/RealmTaskTracker

Agenda

MongoDB & Realm

RealmSwift v10 SDK

SwiftUI & Property Wrappers

Companion App

MongoDB

Mango

A fruit

MongoDB

A database

MongoDB

JSON Database

NoSQL document database using BSON, a binary JSON format. Tools use JSON and JavaScript-like syntax.

Open Source

Community and enterprise options built on top of an open source core. Many open source tools and SDKs. Since 2009.

Atlas Portal

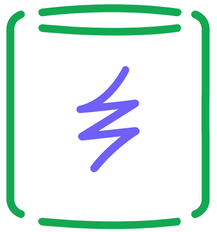
Manage clusters without servers using the MongoDB Atlas Portal. Free tier and many levels of paid tiers.

Local Realms

Object database for iOS, macOS and more.

Free & open source.

Schema defined by your code.



Realm SDK

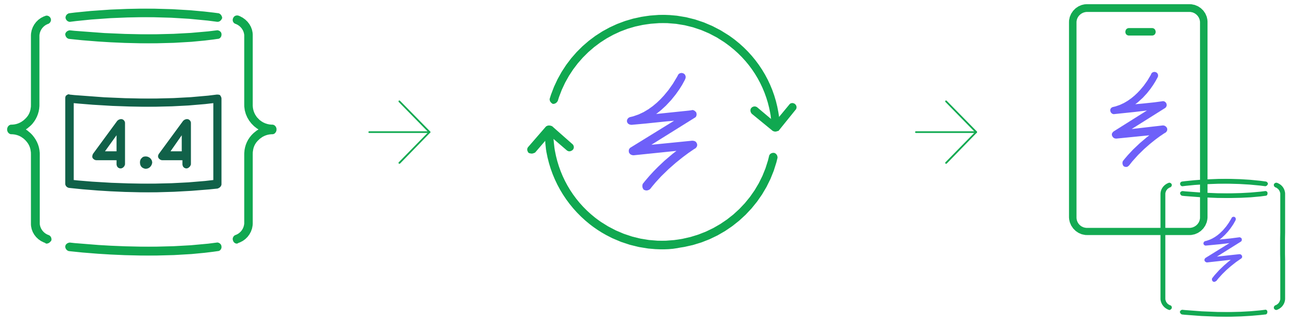
mongoDB®
Realm

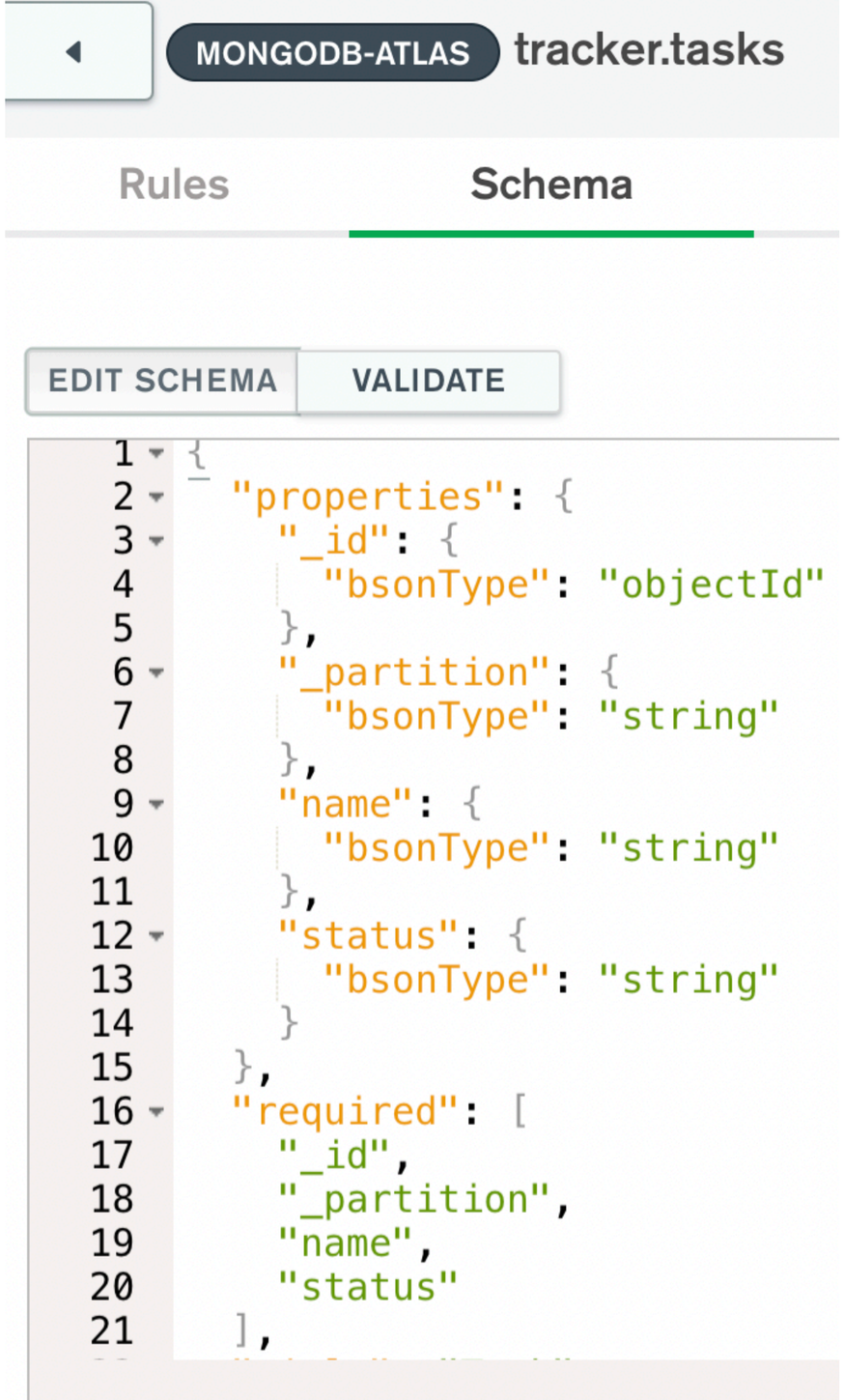
Realm Sync

Middleware between your data and mobile app.

Schema defined on server.

Dev mode allows client to change schema.





Sync Schema

Collection

"tasks" is the name of the collection (table)

Properties

Fields to be included in sync must be specified on the server.

Primary Key

All Realm objects need to have an "_id" property. Can be any supported type.

Realm + Swift

2014

Swift 1.0 released at WWDC

2015

RealmSwift SDK. 360iDev presentation was an apology

2019

MongoDB buys Realm
Apple releases SwiftUI 🎉

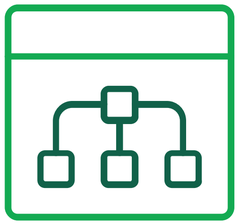
2020

MongoDB 4.4 + Realm 10.0

2021

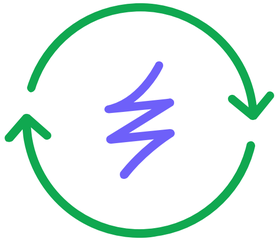
RealmSwift 10.12

SDK



Model Syntax

More Swifty

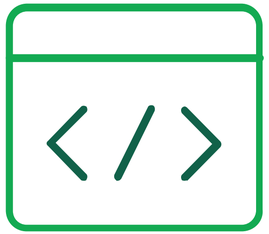


Sync

REST-less networking

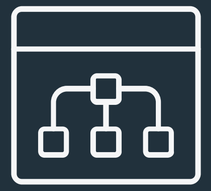
SwiftUI

Gotchas



Property Wrappers

@Hotness



New Model Syntax

Since RealmSwift SDK v10.10

Model Syntax Changes

```
@objc dynamic 👉 @Persisted
```

Properties

```
class Task: Object {

    @objc dynamic var name: String = ""

    @Persisted var name: String

}
```

Primary Key

```
class Task: Object {

    @objc dynamic var _id: ObjectId = ObjectId.generate()

    override static func primaryKey() -> String? {

        return “_id"

    }

    @Persisted(primaryKey: true) var _id: ObjectId

}
```

Relationships

```
class Task: Object {

    let tasks = List<Task>()

    @Persisted var tasks: List<Task>

}
```

Ignored Properties

```
class Task: Object {

    var ignored = true

    override static func ignoredProperties() -> [String] {

        return [“ignored"]

    }

}
```

Enum Properties

```
enum TaskStatus: String, PersistableEnum {

    case Open, InProgress, Complete

}

class Task: Object {

    @Persisted var status: TaskStatus

}
```

Sync Changes

RealmSwift SDK v10

Sync Changes: Realm App

App ID

Login

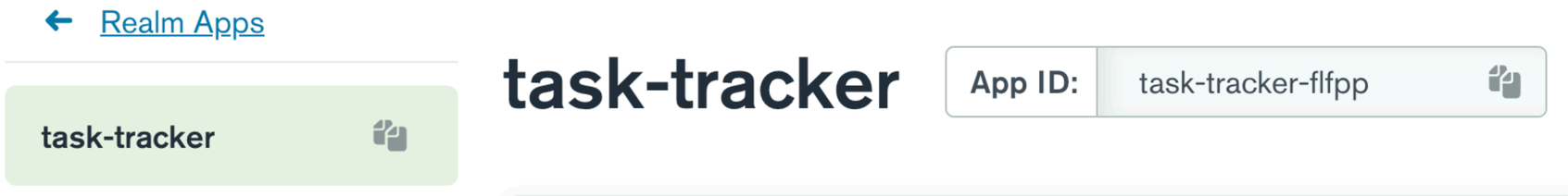
Partition

Open Realm

App ID

Realm App ID
Used instead of a Realm server URL

Find in MongoDB Atlas portal



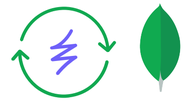
Example

```
let app = RealmSwift.App(id: “task-tracker-flfpp”)
```

Login

Authentication
Required to connect to a synced Realm. Delivers a User to the subscriber.

```
app.login(credentials: Credentials.anonymous)
  .sink(receiveCompletion: {
    if case .failure(let error) = $0 { print("Login err: (error)") }
  }, receiveValue: { user in
    print "\(user) logged in")
  })
```

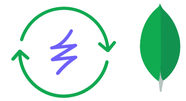


Configure Partition

Partition Key
Name of property on all your Realm objects that specifies the partition (e.g. _partition).
This is configured on your Realm App in MongoDB Atlas.

Partition Value
Determines what data to sync to the device.

```
var config = user.configuration(partitionValue: "Europe")
```



Open Realm

```
config.objectTypes = [User.self, Task.self]


Realm.asyncOpen(configuration: config)
  .sink(receiveCompletion: {
    if case .failure(let error) = $0 { print("Error: (error)") }
  }, receiveValue: { realm in
    print "\(realm) opened")
  })
```

Realm Combine Publishers

Since RealmSwift SDK v10.1

Combine Publishers

`Future<User, Error>`

`AsyncOpenPublisher`

`RealmWillChange`

Value

See `RealmPublishers` enum

App Login

```
App.login(credentials: Credentials) -> Future<User, Error>

App.login(credentials: Credentials,
   _ completion: (Result<User, Error>) -> Void)
```

SwiftUI Gotchas

Live Realm objects vs. structs

Conundrum

SwiftUI
Designed to work with immutable structs. Views need to be informed about data changes, property wrappers: `@State`, `@StateObject` and `@EnvironmentObject`.

Realm
Provides "live" (auto-updating) objects and collections.

Crashes
SwiftUI caches views, so a `List` with `ForEach` will crash when Realm collection has an object removed. Stale data displayed in other cases because SwiftUI doesn't know it changed.

SwiftUI Workarounds

freeze()
Freeze Realm objects before they are passed into a SwiftUI `List` view.

thaw()
Thaw Realm objects before they are modified.

Caveat
Stale views could still be displayed.

Property Wrappers

RealmSwift SDK v10.6

StateRealmObject

```
@StateRealmObject var task: Task
```

Use instead of `@StateObject` with your Realm objects, Lists and EmbeddedObjects.

ObservedResults ✨

```
@ObservedResults(Task.self) var tasks
```

1. Set default configuration

```
Realm.Configuration.defaultConfiguration = config
```

2. Pass config to view

```
ListView().environment(config)
```

`ObservedResults` Helper Methods 😍

```
ForEach(tasks) { task in ... }
  .onDelete(perform: $tasks.remove)
  .onMove(perform: $tasks.move)
  .onInsert(perform: $tasks.append)
```

ObservedRealmObject

```
@ObservedRealmObject(Task.self) var task
```

Use when passing an item from `@ObservedResults` into a detail view.

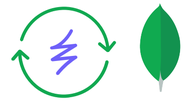
New *Open Property Wrappers

@AsyncOpen
Asynchronously opens Realm notifying SwiftUI view on state change.

@AutoOpen
Similar to `@AsyncOpen`, but will open realm even without network access.

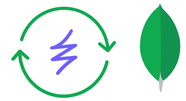
AsyncOpen ✨

```
@AsyncOpen(appId: "app-id",
     partitionValue: "PUBLIC",
     timeout: 1000)
var asyncOpenState
```



AutoOpen ✨

```
@AutoOpen(appId: "app-id",
     partitionValue: "PUBLIC",
     timeout: 1000)
var asyncOpenState
```

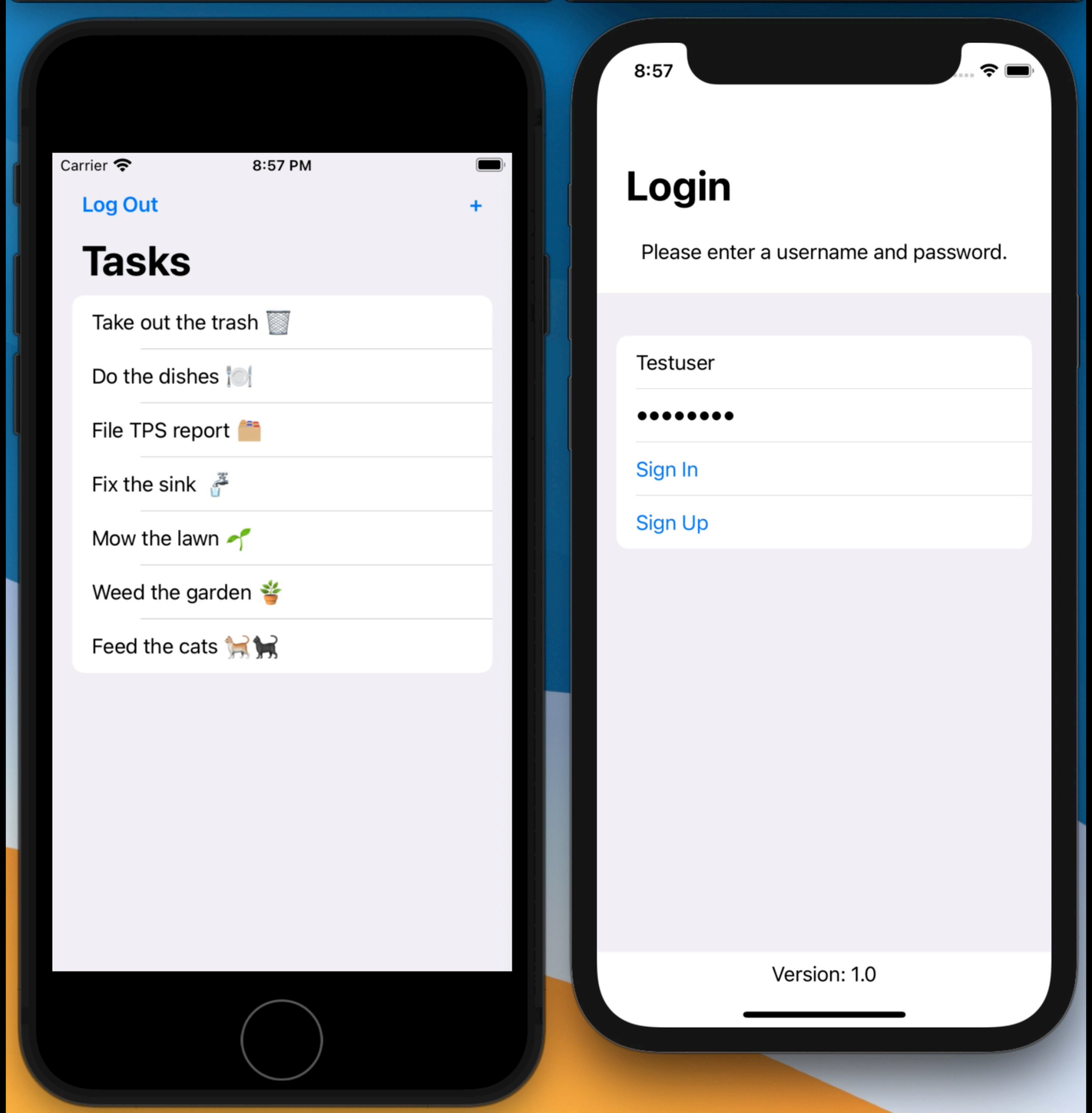


```
switch asyncOpen {
case .waitingForUser:
    ProgressView("Waiting for user to be logged in...")
case .connecting:
    ProgressView()
case .progress(_):
    ProgressView()
case .open(let realm):
    ListView(editTask: Task())
        .environment(\.realm, realm)
case .error(let error):
    ErrorView(error: error)
}
```

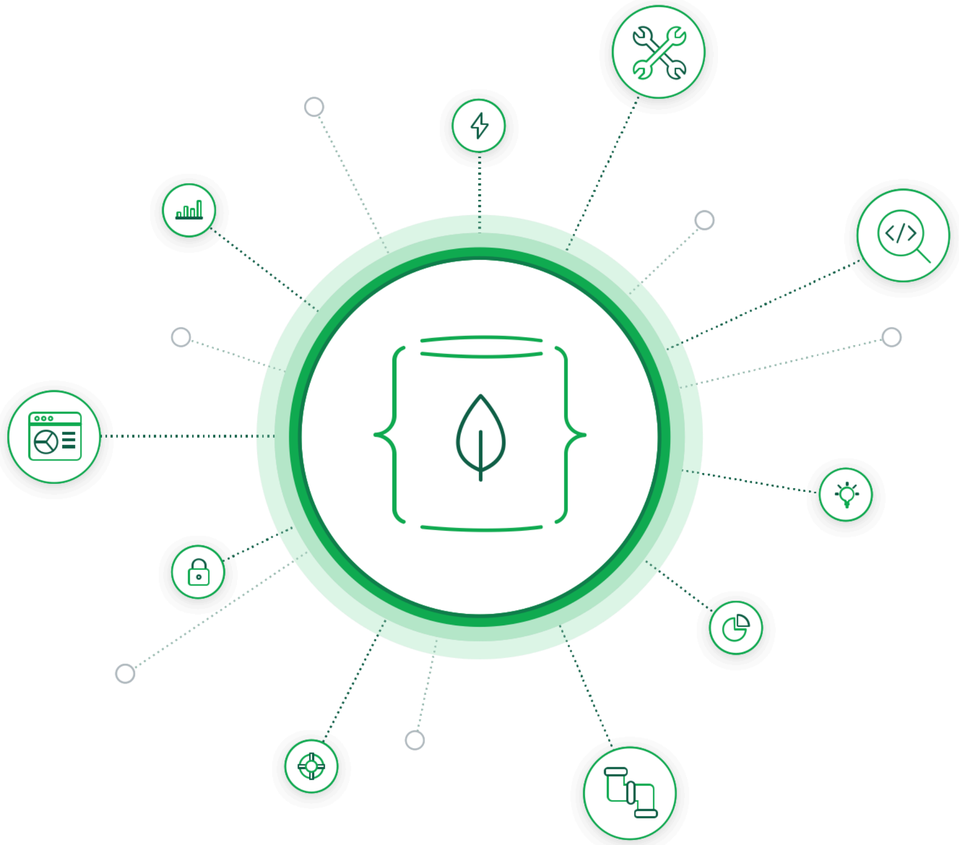
Realm Task Tracker App

github.com/phatblat/RealmTaskTracker


Demo? 🤞


Carrier
8:57 PM
Log Out
+
Tasks
Take out the trash
Do the dishes
File TPS report
Fix the sink
Mow the lawn
Weed the garden
Feed the cats
8:57
Login
Please enter a username and password.
Testuser
Sign In
Sign Up
Version: 1.0

Questions



Thank you

github.com/phatblat/RealmTaskTracker

Ben Chatelain | Chief iOS Engineer | Kaiser Permanente | @phatblat