Introducing Realm

for CAU

Eunjoo Im

realm

eunjoo.im@realm.io

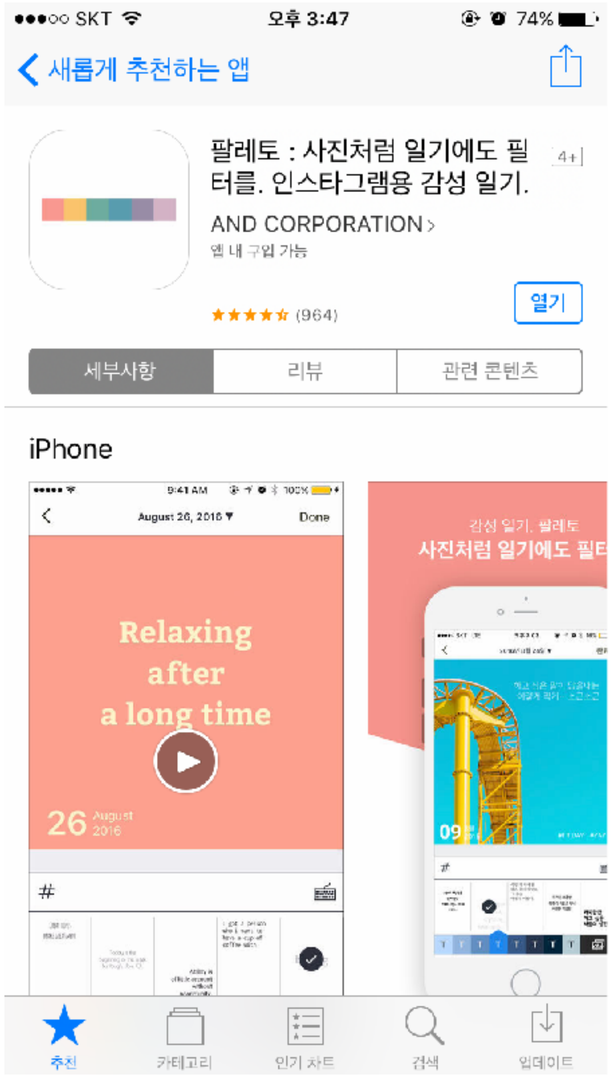
앱을 만들어 봅시다!

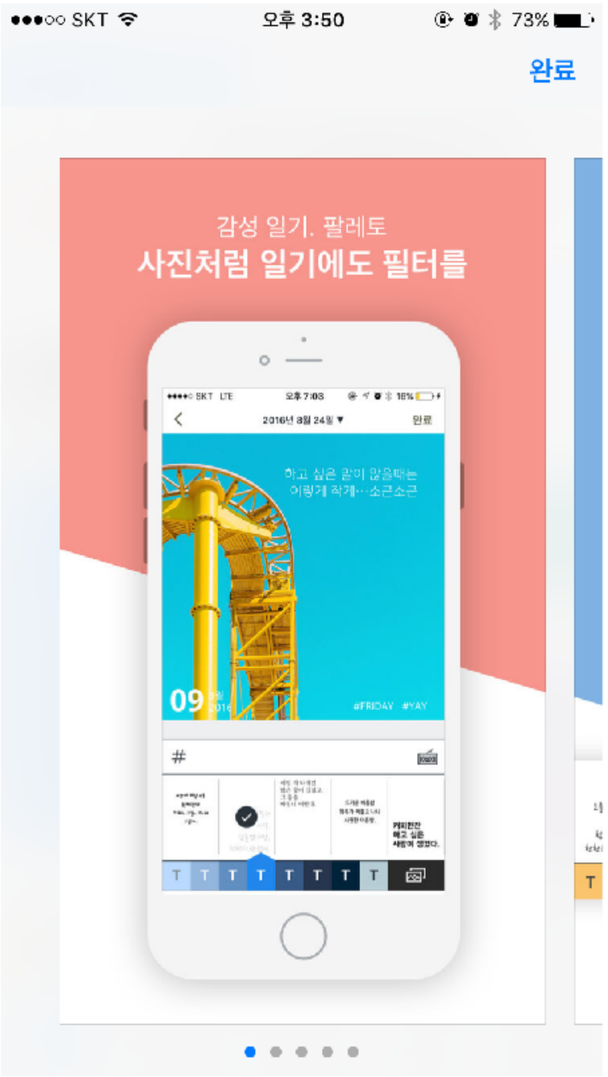
어떤 앱을 만들까요?

eunjoo.im@realm.io

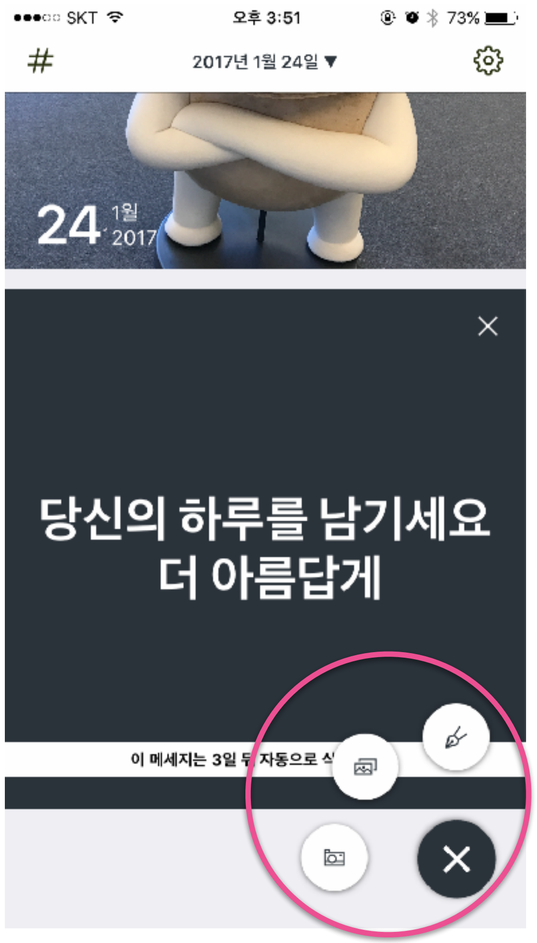
Palleto

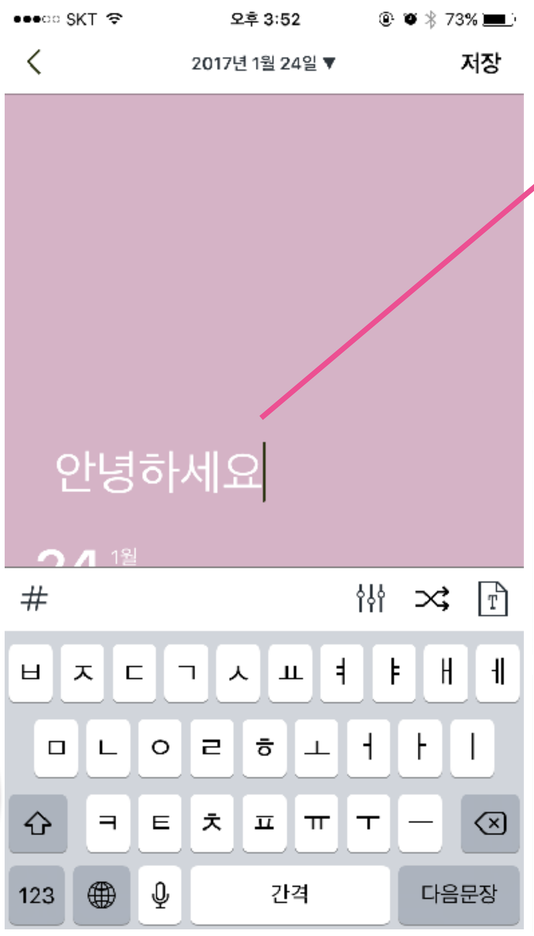
= 일기 앱

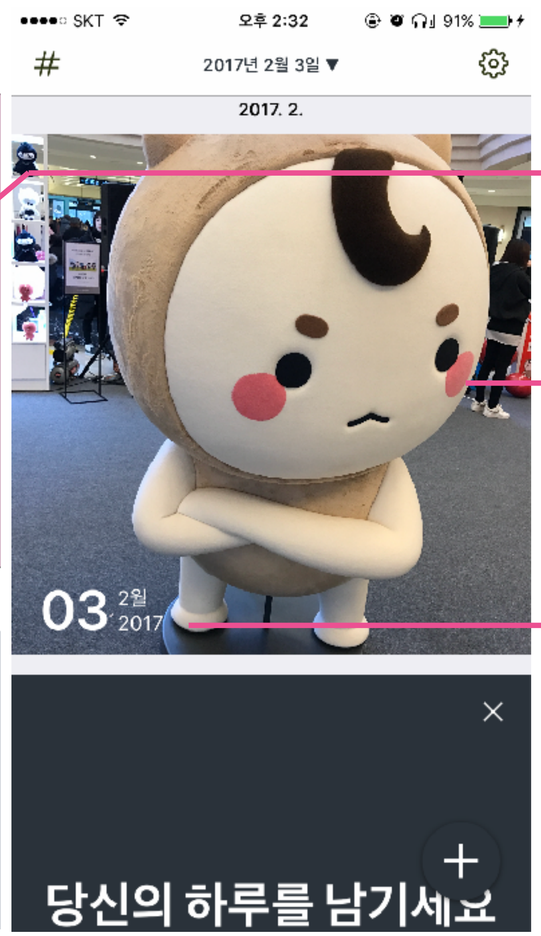




eunjoo.im@realm.io



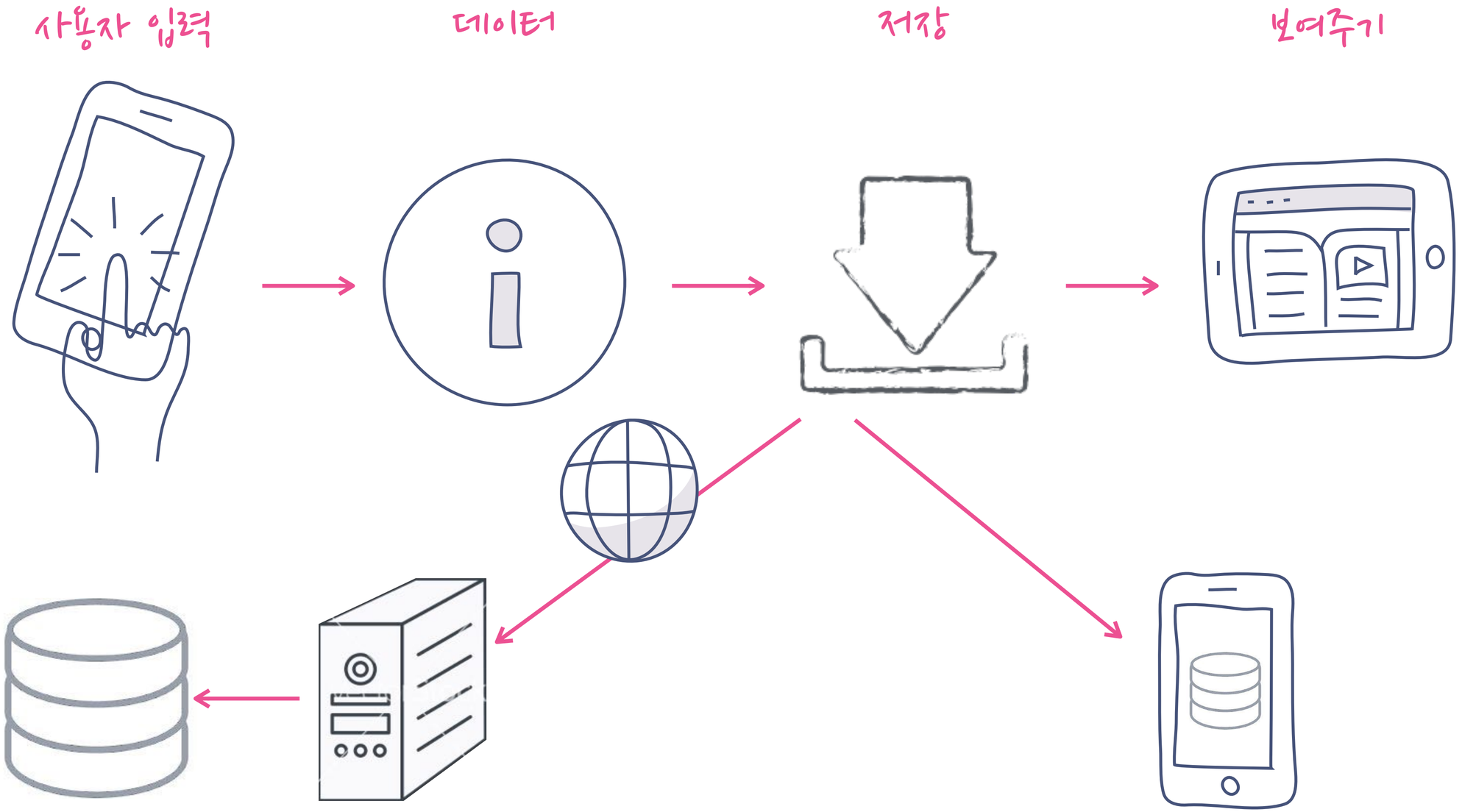




텍스트 + 사진 + 날짜

"데이터"

eunjoo.im@realm.io



realm

eunjoo.im@realm.io

자아실현의 욕구

존경의 욕구

사회적 욕구

안전의 욕구

생리적 욕구

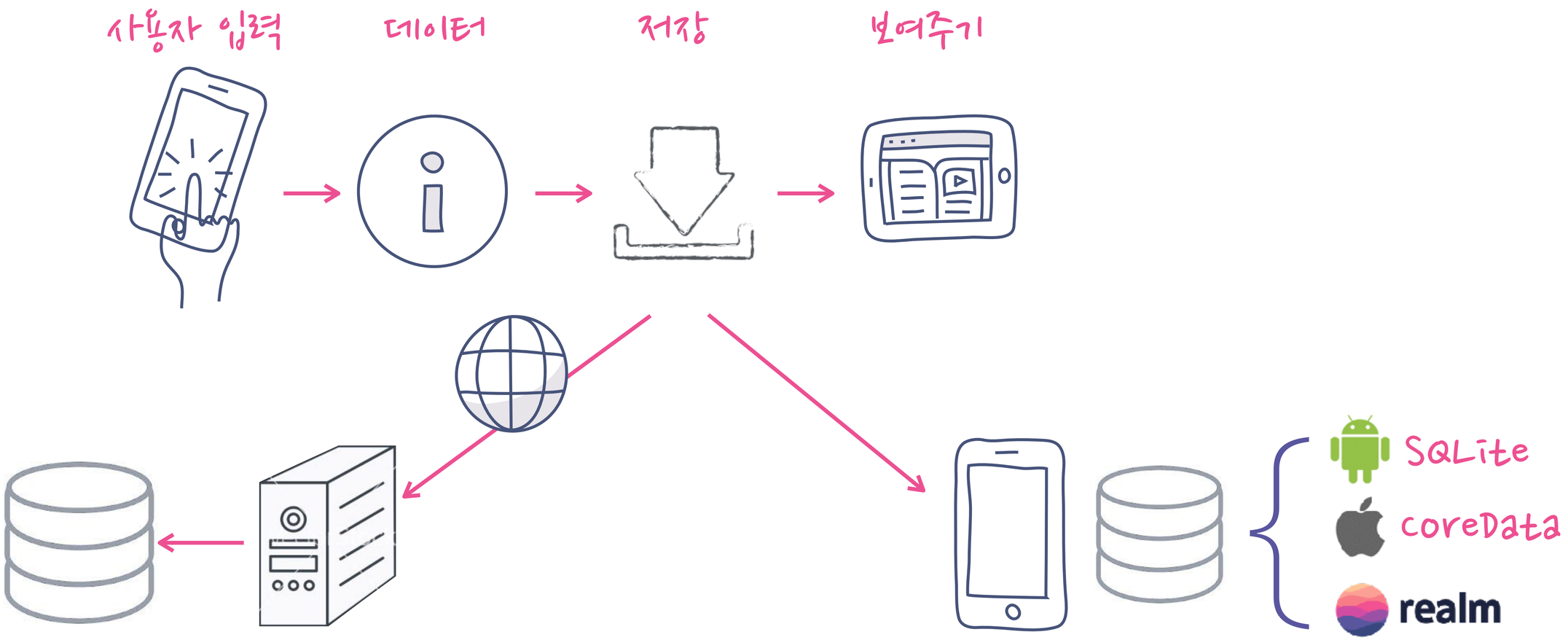
?!

eunjoo.im@realm.io

Mobile Database

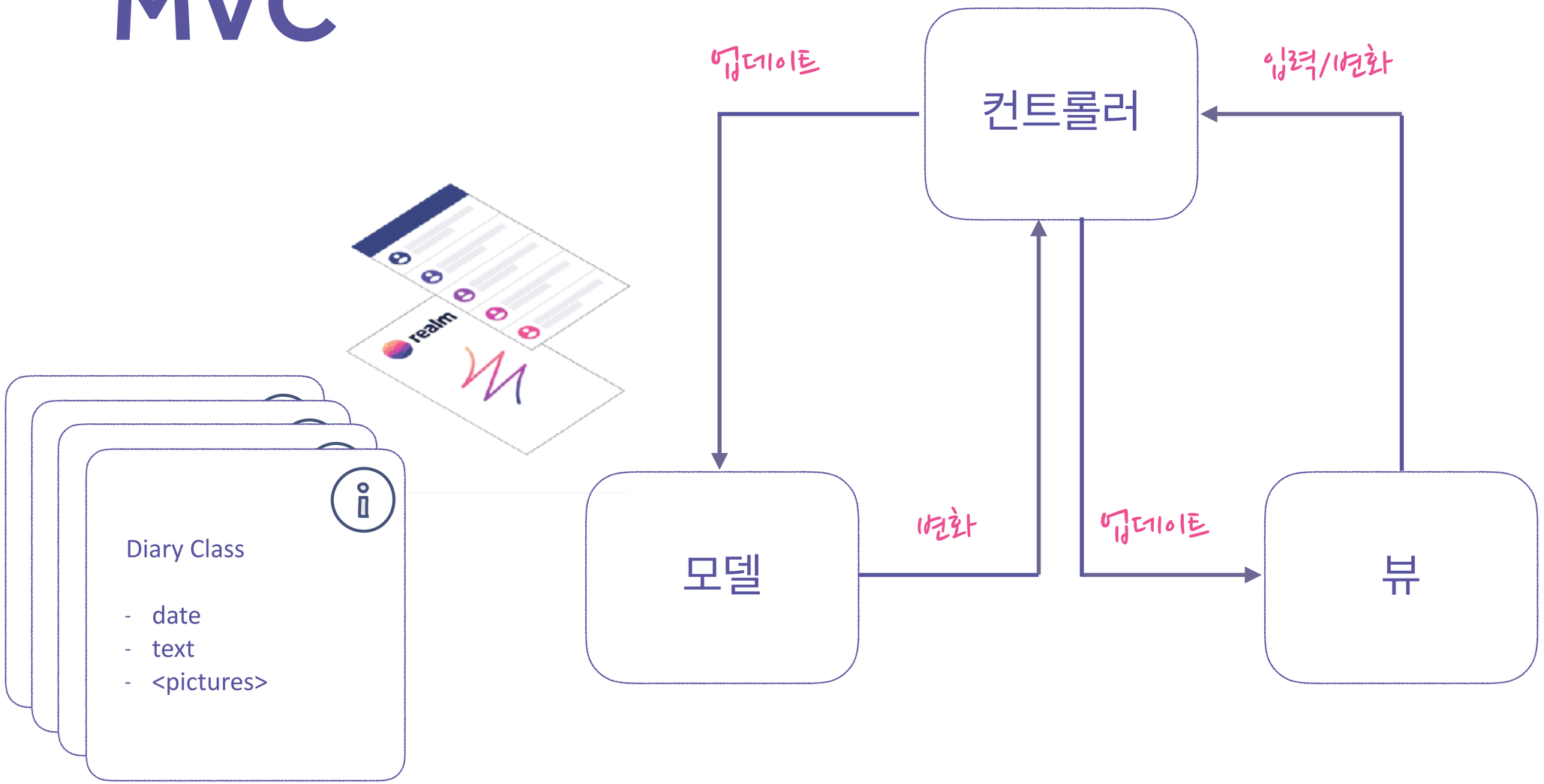
eunjoo.im@realm.io

Database



eunjoo.im@realm.io

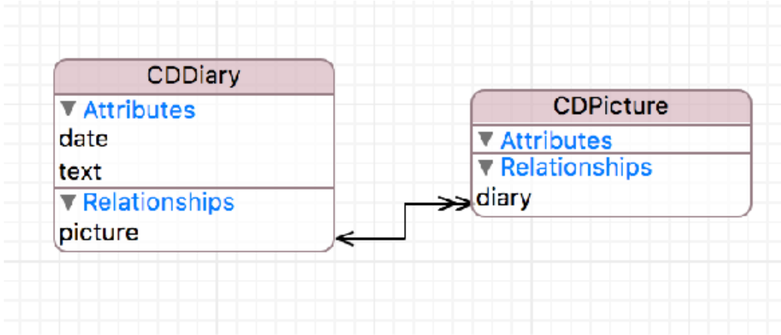
MVC

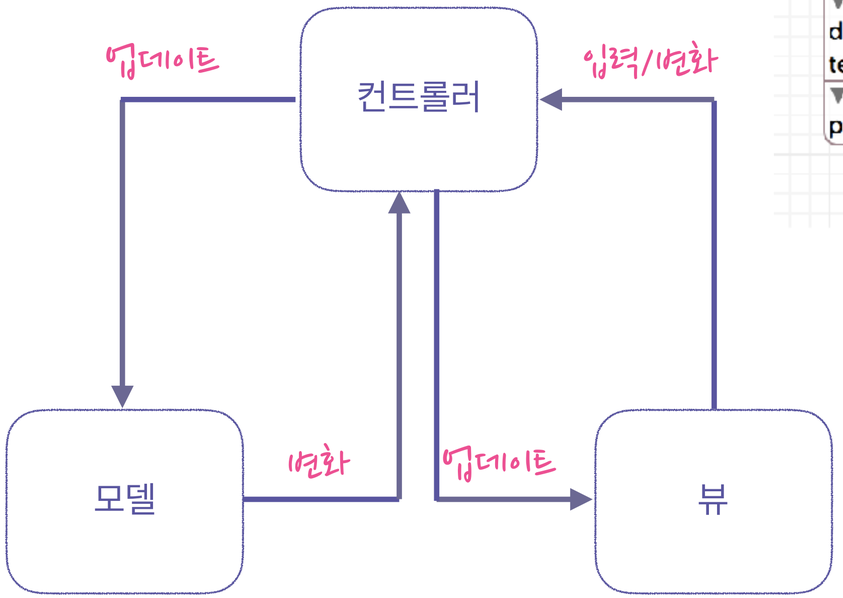


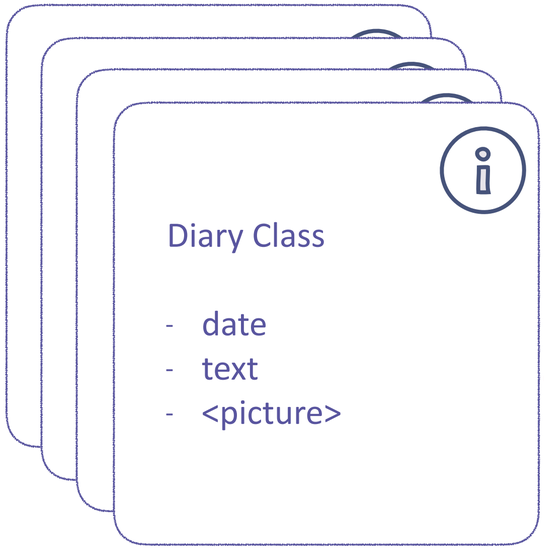
eunjoo.im@realm.io

ORM

= Object-relational mapping





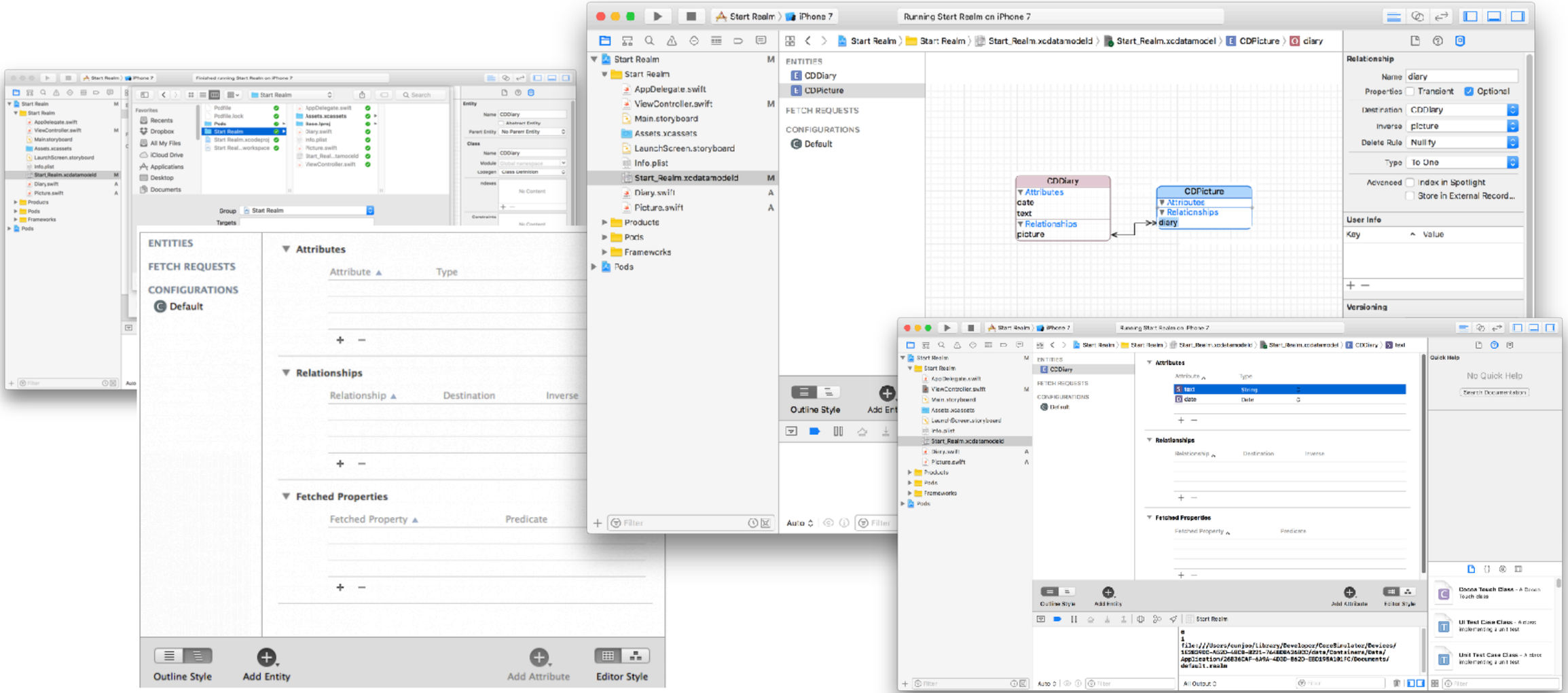


No	Date	Text	Picture ?
1	2016-01-24	OOO	?
2	2016-01-25	XXX	?
3	2016-01-26	ZZZ	?
4	2016-01-27	QQQ	?

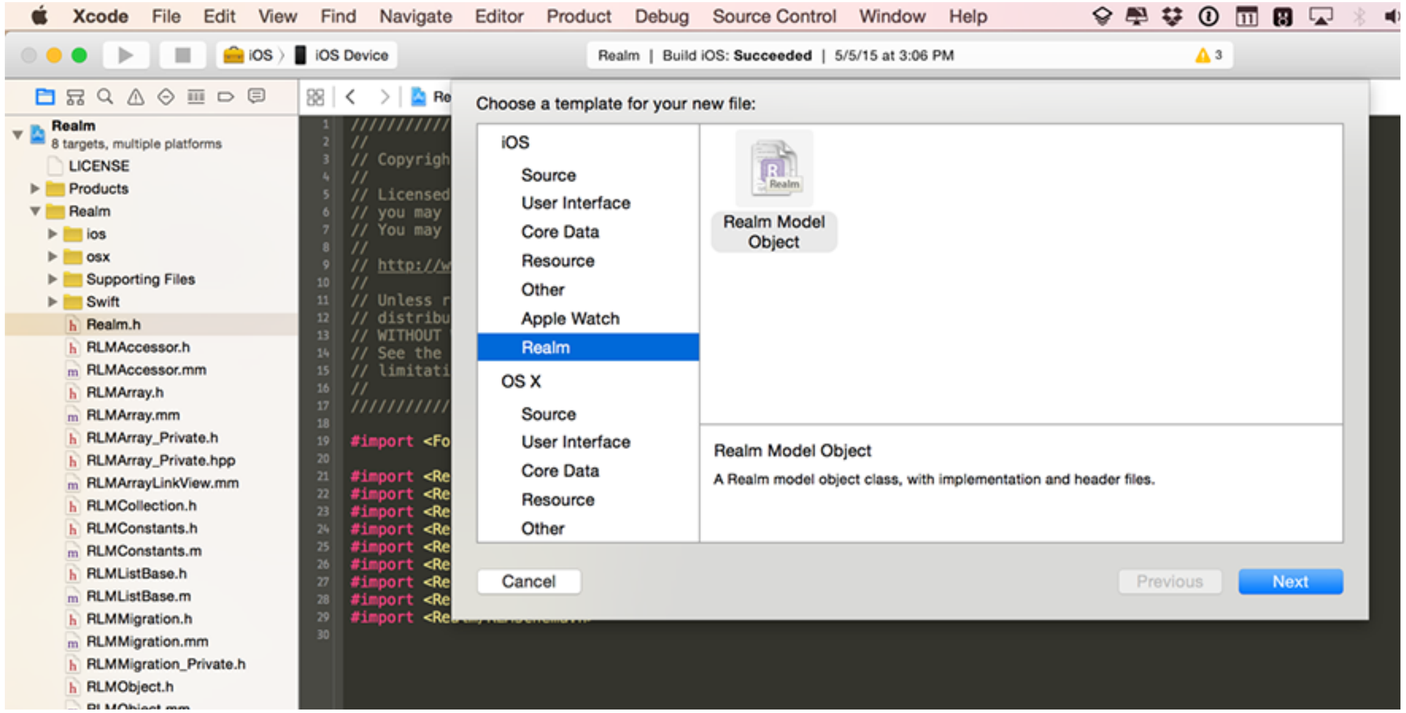
realm

eunjoo.im@realm.io

CoreData 모델 만들기



Realm 모델 만들기

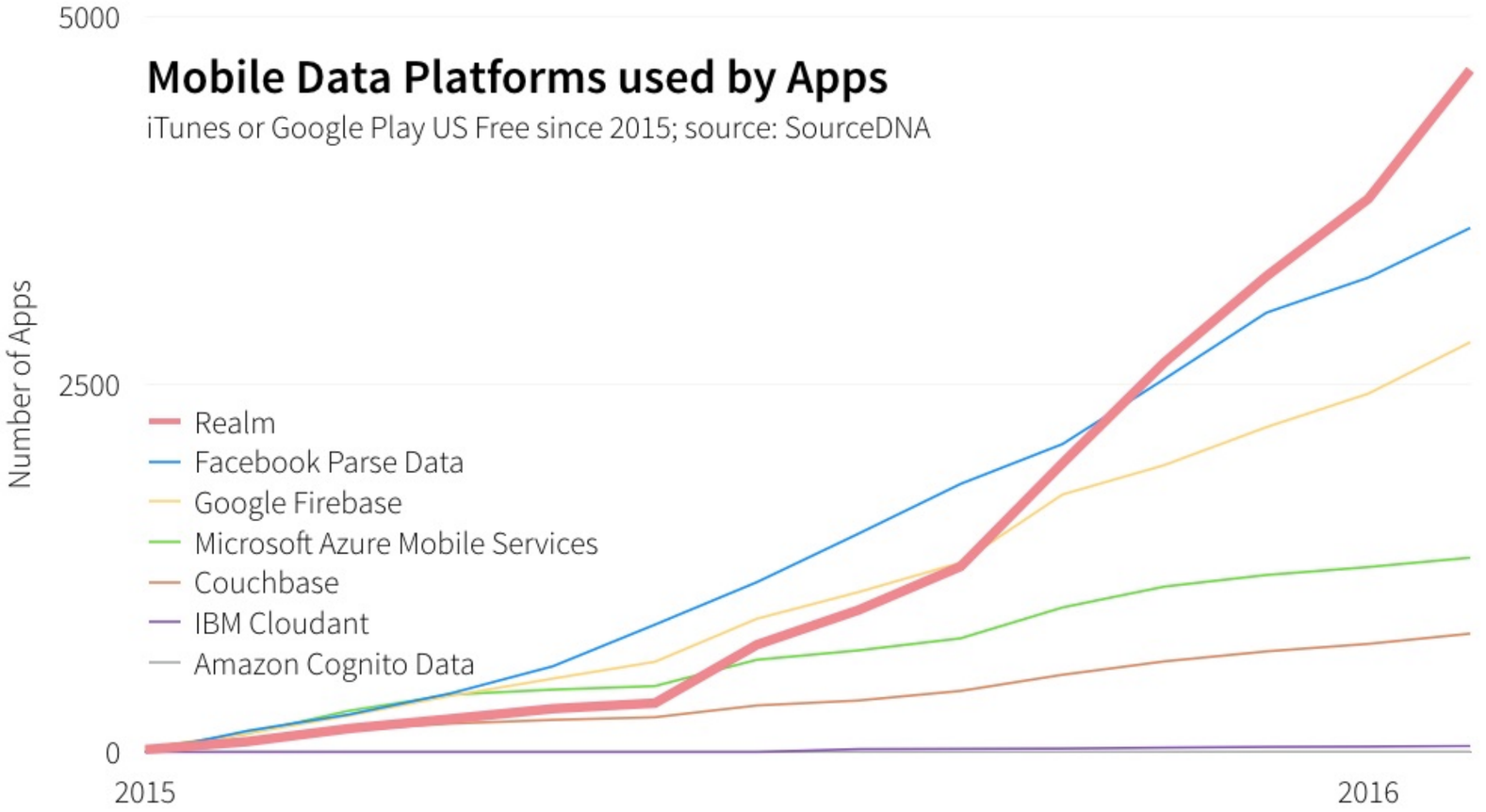



```
class Diary: Object {
   dynamic var date = Date()
   dynamic var text = ""
   let pictures = List<Picture>()
}
```


eunjoo.im@realm.io

Realm Mobile Database

eunjoo.im@realm.io



빠르게 성장하는 모바일 데이터베이스 (오픈소스)

https://realm.io/kr

SQLite와 CoreData를 대체할 수 있는 크로스 플랫폼 데이터베이스

realm

eunjoo.im@realm.io

2014년 7월 15일 공개

↓

2년 반이 지난 지금 사용 기업

realm

eunjoo.im@realm.io

C++ 기반의 크로스 플랫폼

SQLite 기반이 아니라 고유 C++ 코어를 가짐

eunjoo.im@realm.io

장점

빠른 속도

- 모바일을 위해서 만들어졌기 때문에 데이터 읽기/쓰기가 빠르고 효율적

풍부한 기능

- 마이그레이션, 그래프 쿼리, 암호화와 스레드 등의 다양한 기능을 지원

쉬운 사용

- Object를 상속하여 쉬운 사용

eunjoo.im@realm.io

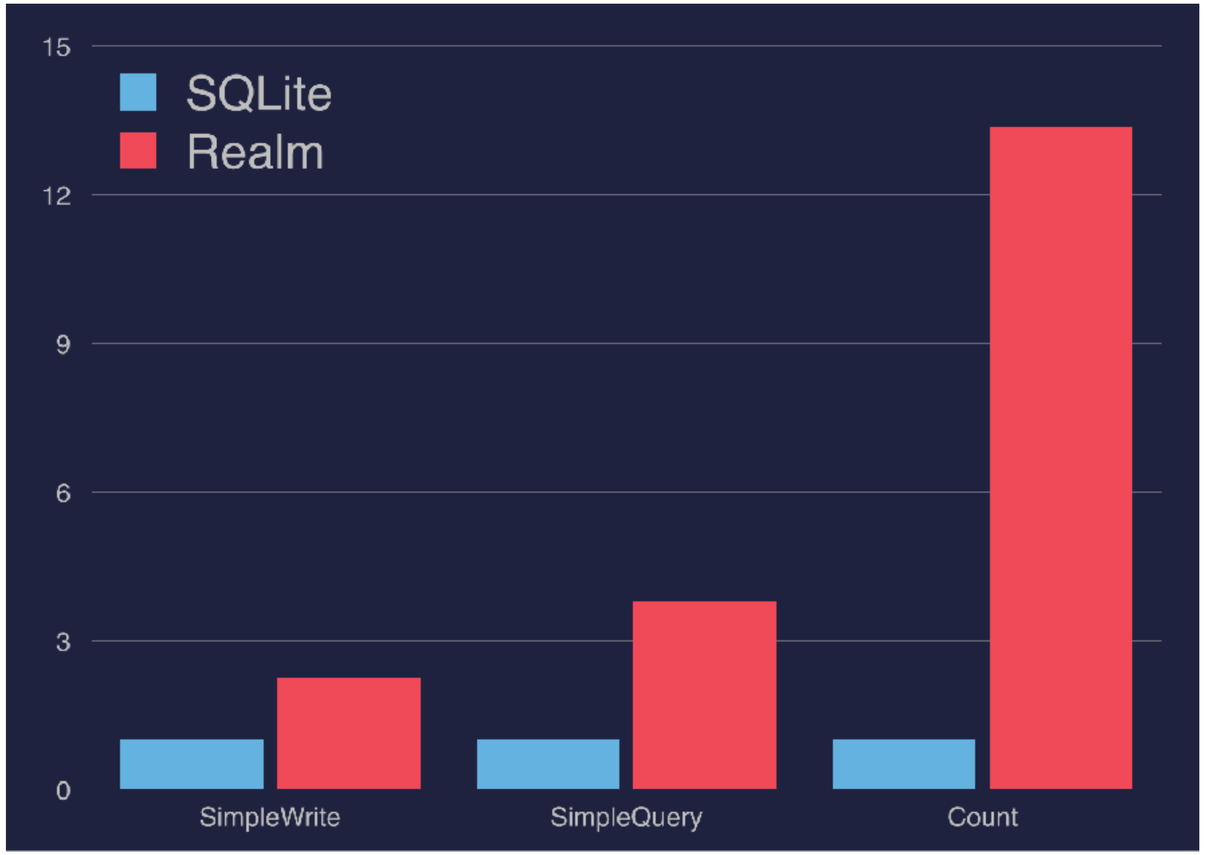
빠른 속도

- 모바일을 위해서 만들어졌기 때문에 데이터 읽기/쓰기가 빠르고 효율적

eunjoo.im@realm.io

빠른 속도

v.s. SQLite



출처: http://static.realm.io/downloads/java/android-benchmark.zip

빠른 속도

Zero-copy

기존 ORM은 Copy 필요 <-> Realm은 불필요

- memory mapped file 사용
- In-memory 처럼 disk 사용
  - 읽을 데이터의 offset 계산
  - mapped file에서 읽어옴
  - property에 access해서 원본값을 반환

Result 자동 업데이트

```
let diaries = realm.objects(Diary.self)
print(diaries.count) // => 아무 객체도 넣지 않아서 결과는 0입니다.

try! realm.write {
    realm.add(myDiary)
}

print(diaries.count) // => 1
```


eunjoo.im@realm.io

풍부한 기능

- 마이그레이션, 그래프 쿼리, 암호화와 스레드 등의 다양한 기능을 지원

eunjoo.im@realm.io

암호화

Realm 생성 시에 설정

```swift
// Generate a random encryption key
var key = Data(count: 64)
_ = key.withUnsafeMutableBytes { bytes in
  SecRandomCopyBytes(kSecRandomDefault, 64, bytes)
}

// Open the encrypted Realm file
let config = Realm.Configuration(encryptionKey: key)
do {
  let realm = try Realm(configuration: config)
  // Use the Realm as normal
  let dogs = realm.objects(Dog.self).filter("name contains 'Fido'")
} catch let error as NSError {
  // If the encryption key is wrong, `error` will say that it's an invalid database
  fatalError("Error opening realm: \(error)")
}
```


eunjoo.im@realm.io

Notification

Realm, Results, List, LinkingObjects 등이 바뀔 때마다 통지
Key-Value Observation

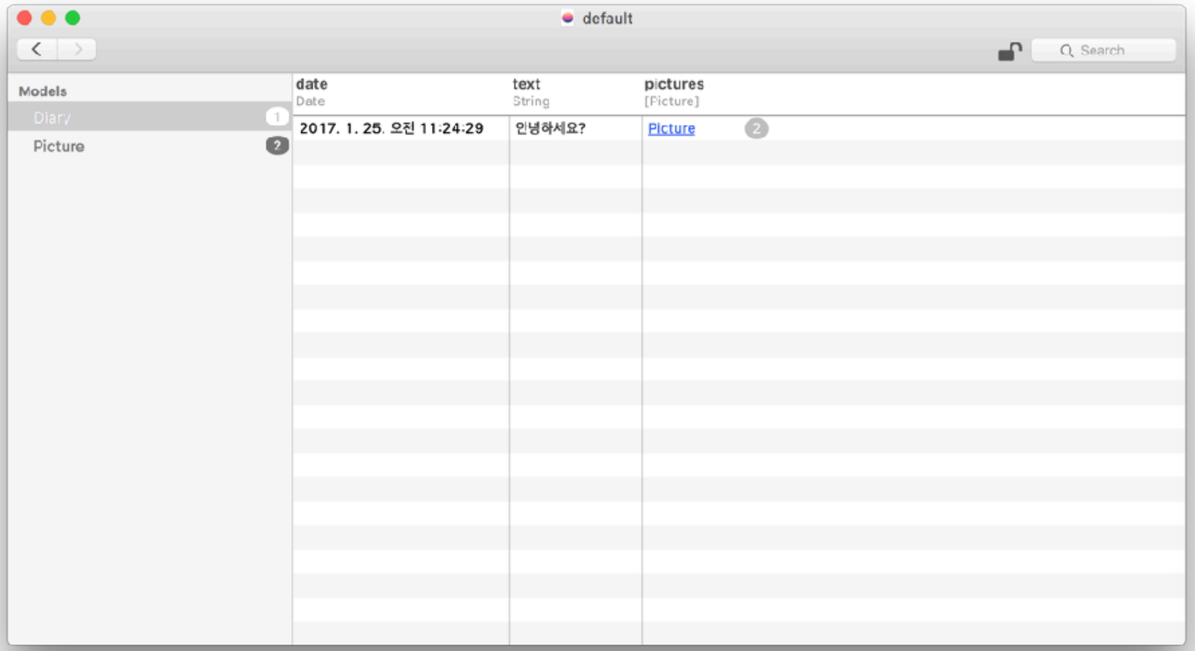
```
// Observe Realm Notifications
let token = realm.addNotificationBlock { notification, realm in
    viewController.updateUI()
}

// later
token.stop()
```

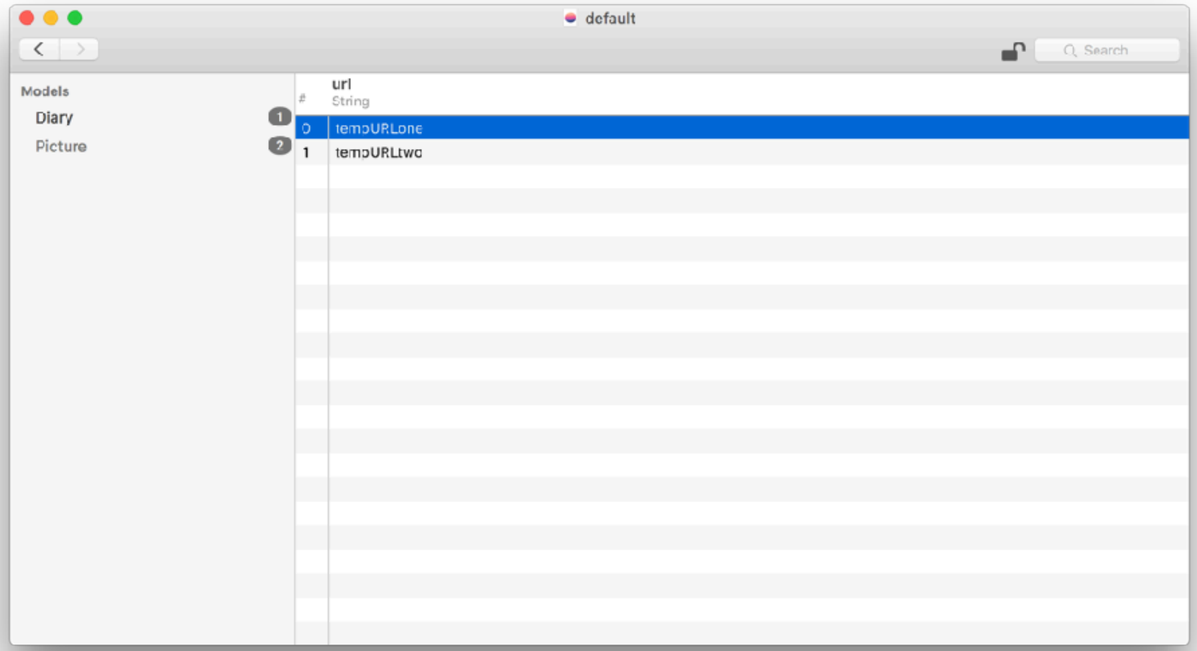

eunjoo.im@realm.io

Realm Browser

데이터베이스를 읽고 편집할 수 있는 브라우저 제공

https://goo.gl/Nnz0SC





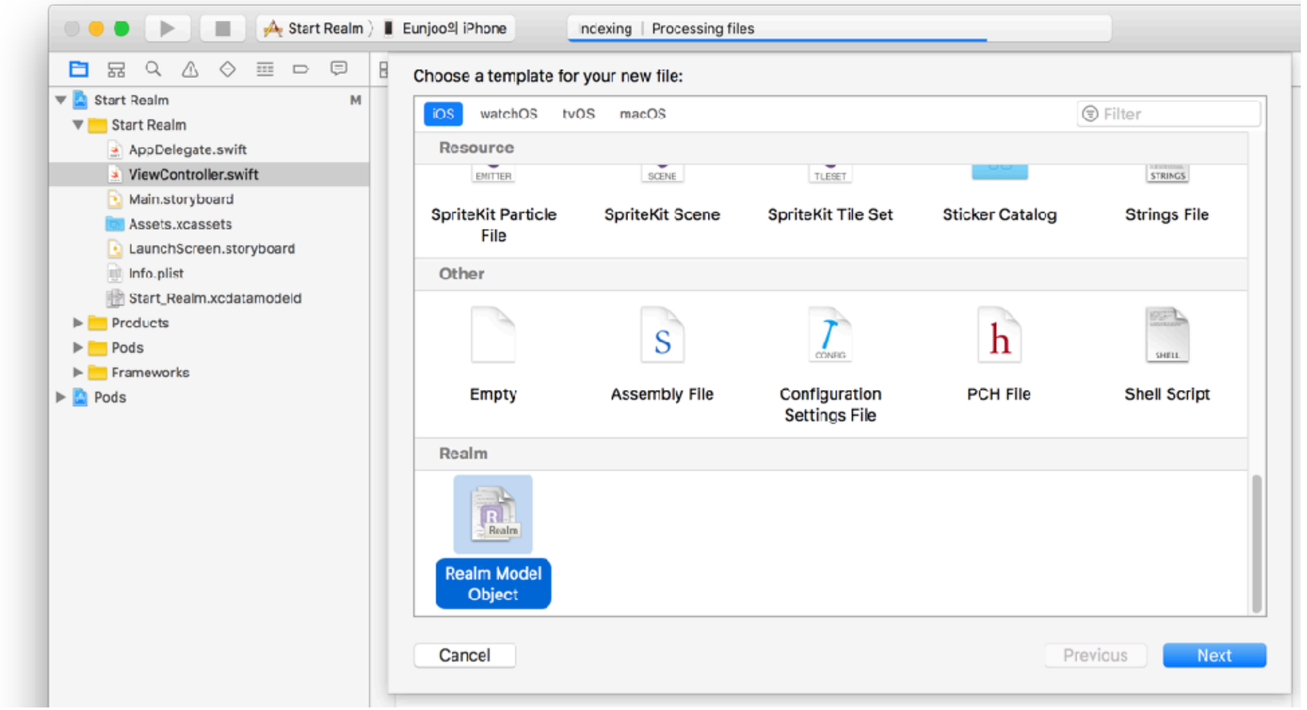
realm

eunjoo.im@realm.io

Xcode 플러그인

새 Realm 모델을 쉽게 만들 수 있는 플러그인 제공

https://goo.gl/bVHNhh



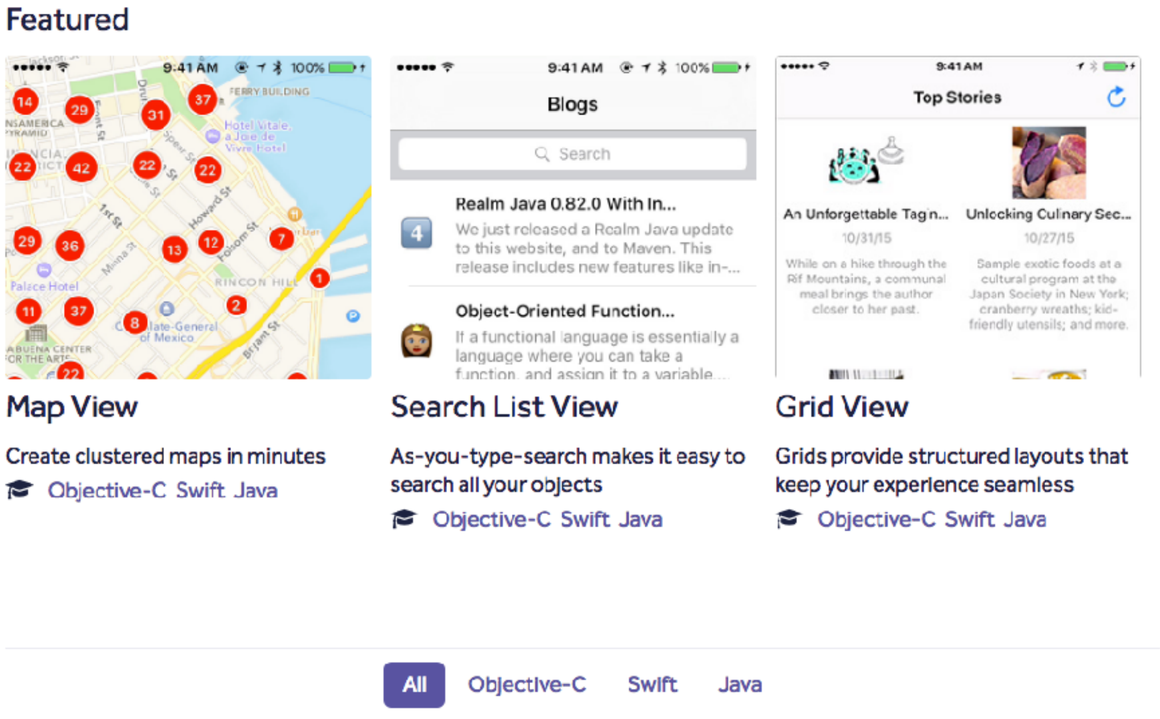
realm

eunjoo.im@realm.io

Xcode 애드온

Realm 기반의 앱 개발에 도움이 되는 커뮤니티 애드온

https://realm.io/kr/addons/



eunjoo.im@realm.io

쉬운 사용

- Object를 상속하여 쉬운 사용

eunjoo.im@realm.io

간결한 코드

모델 정의

Object를 상속하여 모델 정의
속성은 property로 사용하여 정의
관계와 자료구조는 타겟 타입의 속성이나 객체의 List를 포함하여 간단하게 정의

```
class Diary: Object {

    dynamic var date = NSDate(timeIntervalSince1970: 1)
    dynamic var text = ""
    var pictures = List<Picture>()
}


class Picture: Object {
    dynamic var url = ""
}
```

realm

eunjoo.im@realm.io

간결한 코드

Insert or update

```swift
var diary: NSManagedObject

let moc = managedObjectContext
let diaryFetch = NSFetchRequest<NSFetchRequestResult>(entityName: "CDDiary")

do {
    diary = managedObjectContext.existingObject(with: objectID)
    diary.setValue("반갑습니다", forKey: "text")
} catch {
    print("Can't find object \(error)")
    let entity =
        NSEntityDescription.entity(forEntityName: "CDDiary",
                                   in: managedObjectContext)!
    diary = NSManagedObject(entity: entity,
                            insertInto: managedObjectContext)
}

do {
    try diary.managedObjectContext?.save()
} catch {
    let saveError = error as NSError
    print(saveError)
}
```

```swift
let diary = Diary()
diary.text = "반갑습니다"
diary.id = 1

try! realm.write {
    realm.add(diary, update: true)
}
```

realm

eunjoo.im@realm.io

간결한 코드

Fetch

```
func fetchFromCoreData() {
    let moc = managedObjectContext
    let diaryFetch = NSFetchRequest<NSFetchRequestResult>(entityName: "CDDiary")

    do {
        let fetchedDiaries = try moc.fetch(diaryFetch) as! [CDPicture]
        print(fetchedDiaries)
    } catch {
        fatalError("Failed to fetch diaries: \(error)")
    }
}
```

```
let diaries = realm.objects(Diary.self)
```

realm

eunjoo.im@realm.io

다양한 쿼리

Chaining이 가능한 다양한 쿼리 지원 (ex. 필터링)

```
let tanDogs = realm.objects(Dog.self).filter("color = 'tan'")
let tanDogsWithBNames = tanDogs.filter("name BEGINSWITH 'B'")
```

하나 또는 여러 개의 속성으로 정렬 기준이나 순서 설정

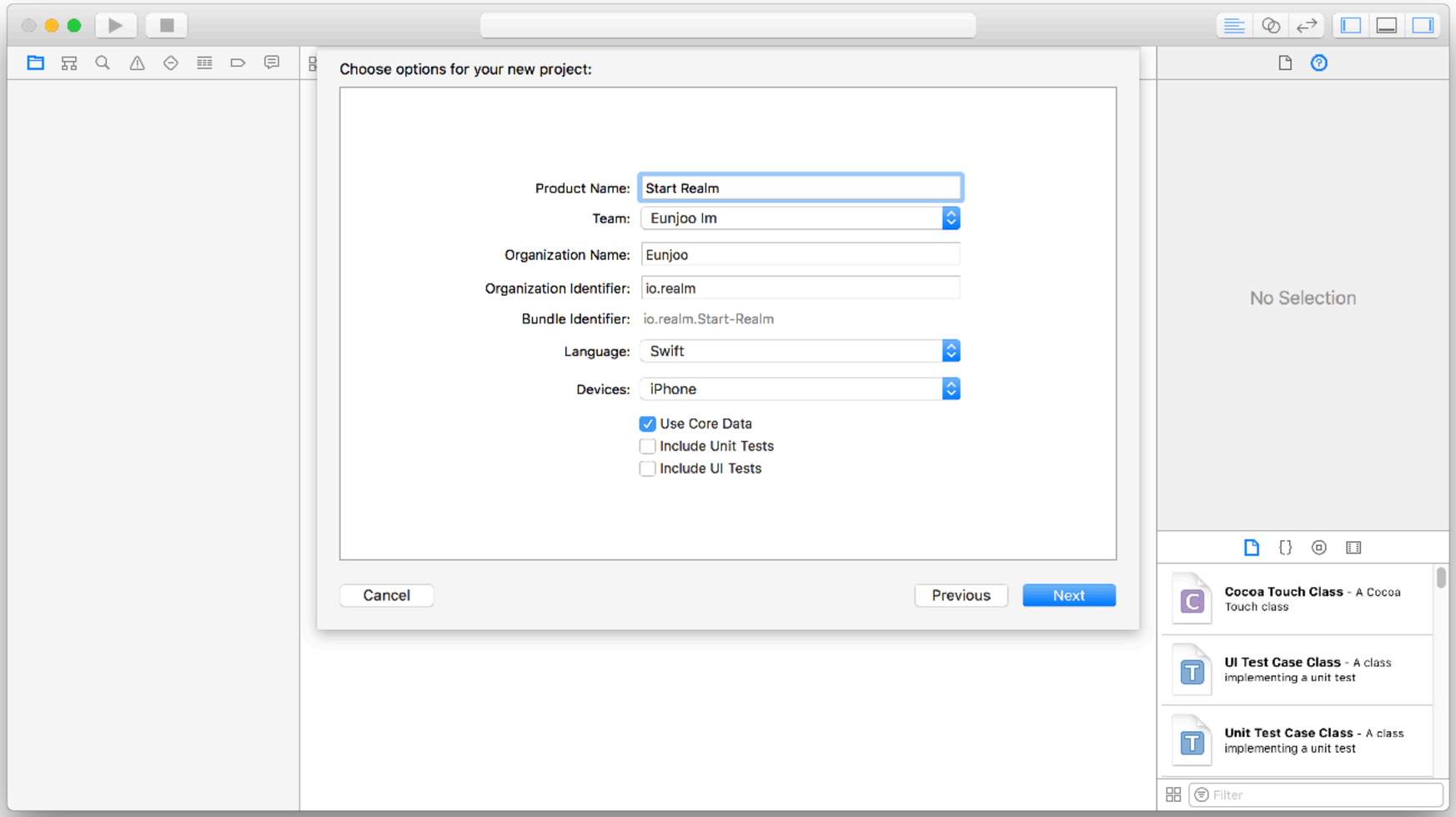
```
class Person: Object {
  dynamic var name = ""
  dynamic var dog: Dog?
}
class Dog: Object {
  dynamic var name = ""
  dynamic var age = 0
}

let dogOwners = realm.objects(Person.self)
let ownersByDogAge = dogOwners.sorted(byKeyPath: "dog.age")
```


eunjoo.im@realm.io

만들어 봅시다!

eunjoo.im@realm.io

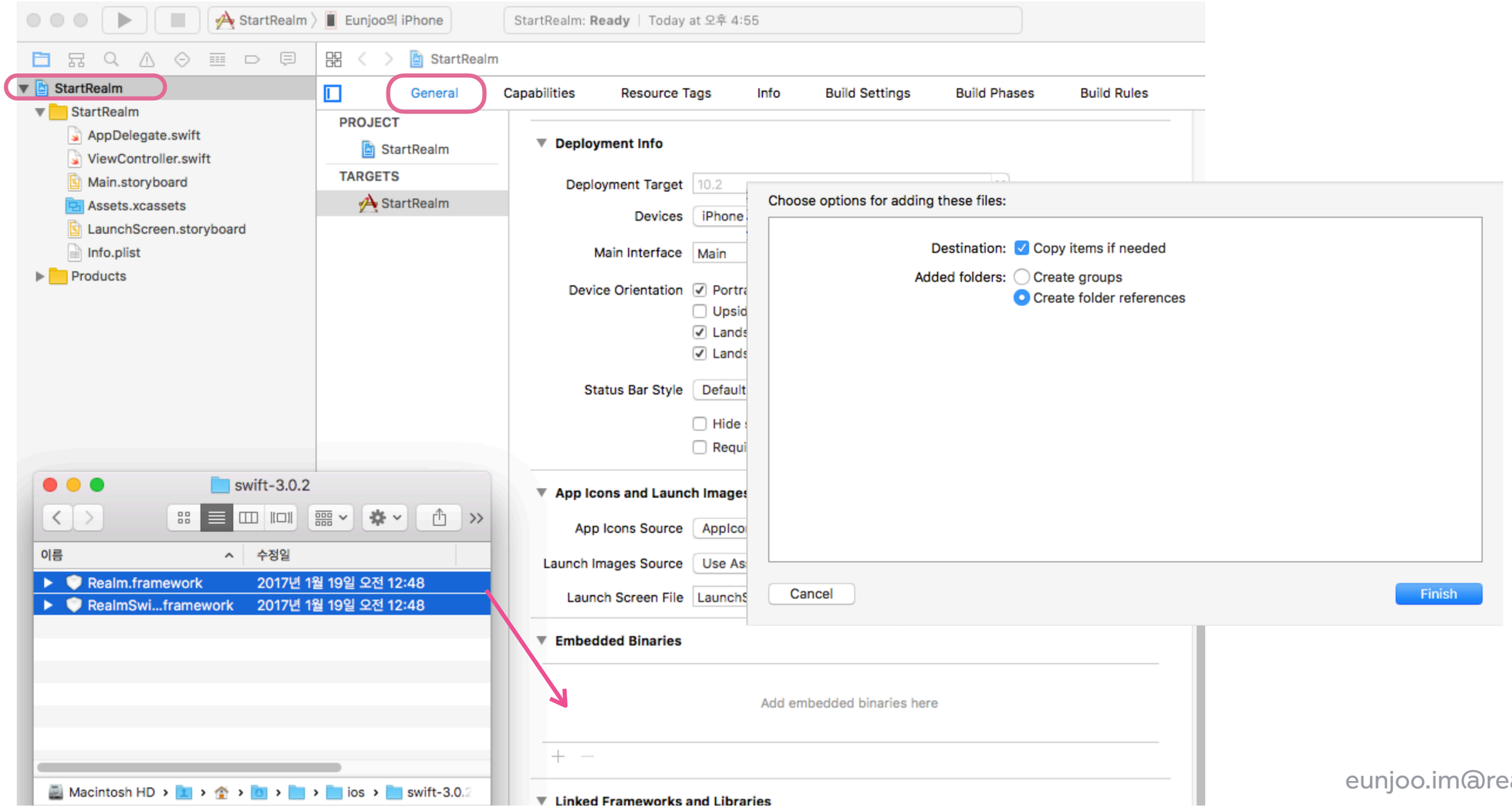
Single 프로젝트로 시작



realm

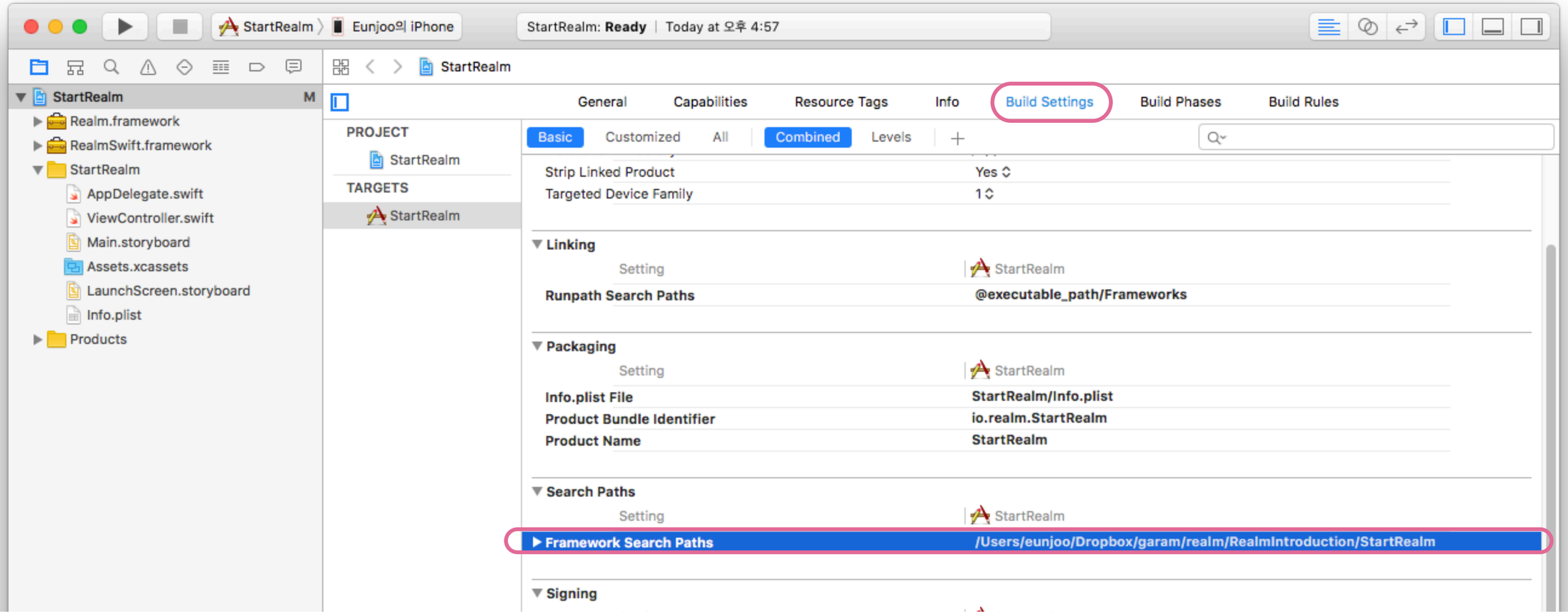
eunjoo.im@realm.io

Realm 설정



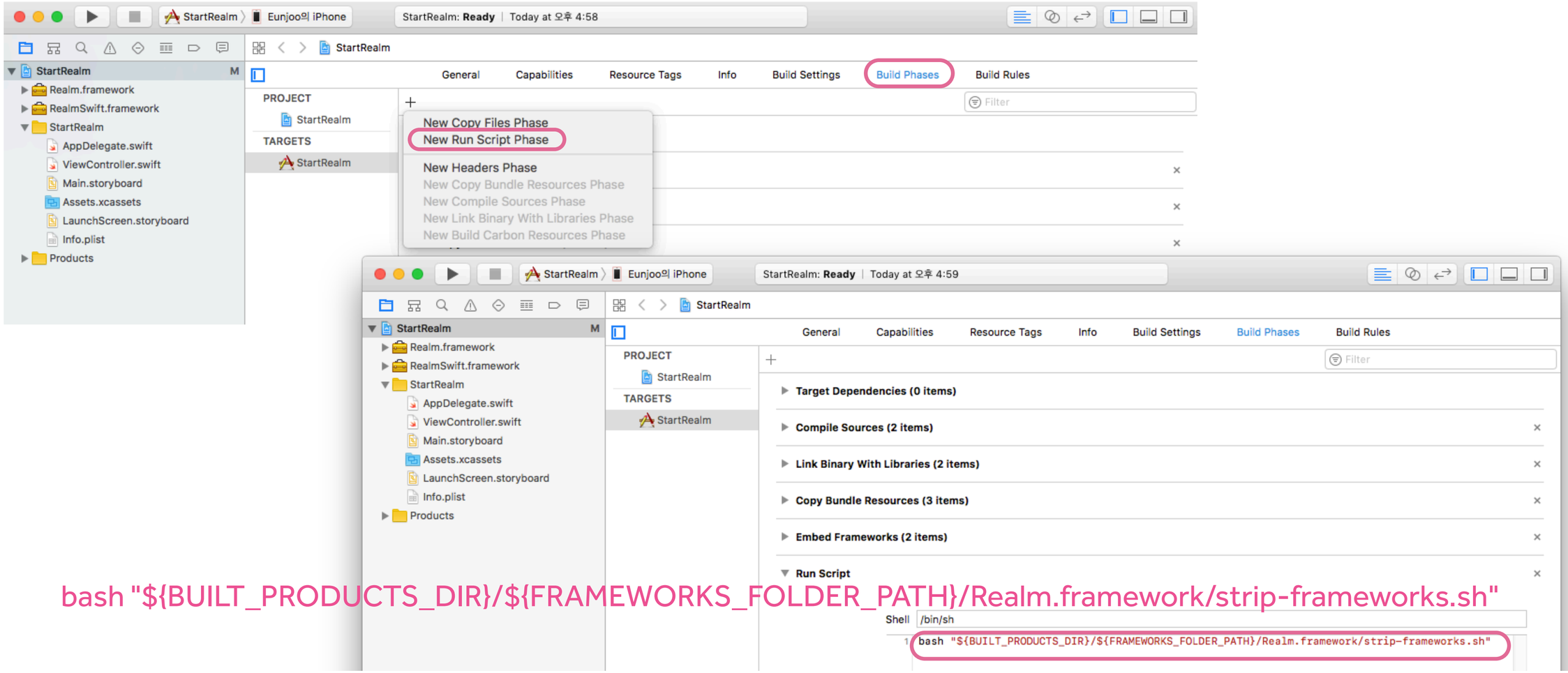
eunjoo.im@realm.io

Realm 설치



eunjoo.im@realm.io

Realm 설치

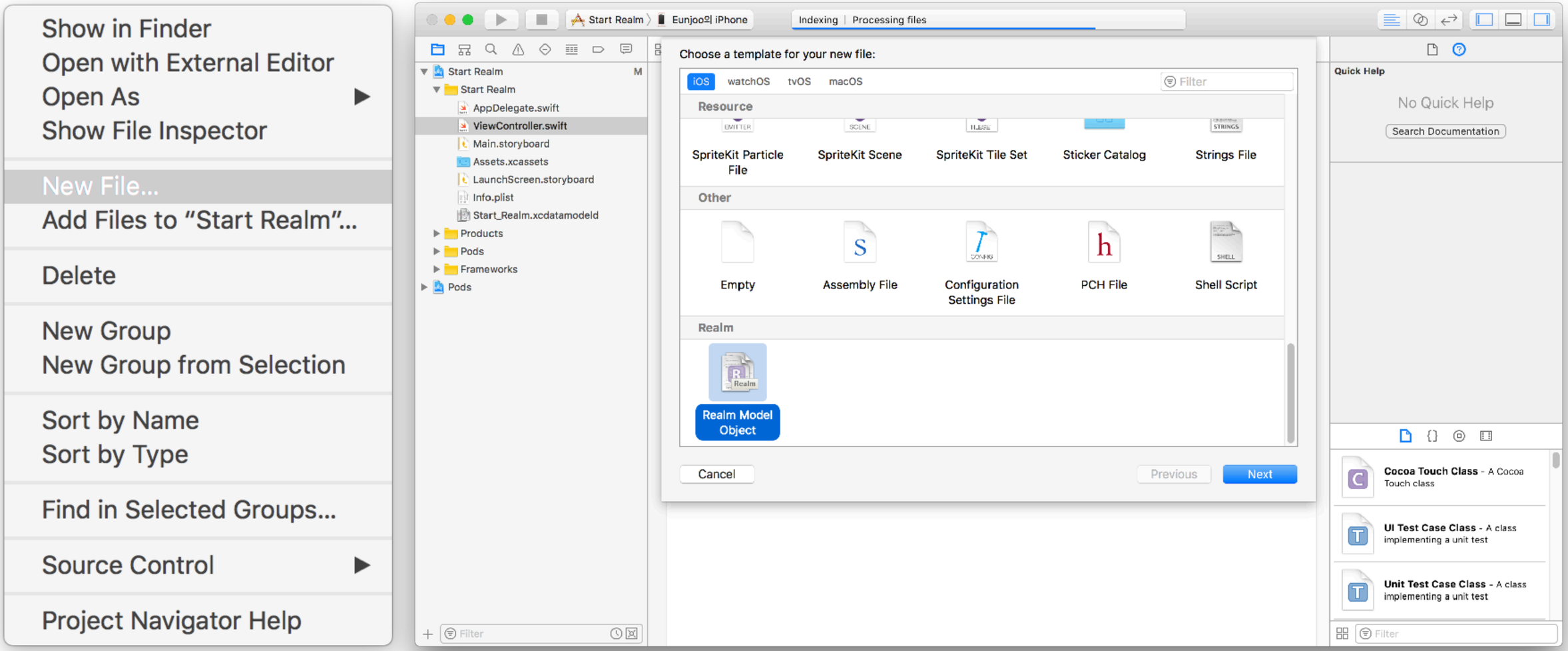


bash "${BUILT_PRODUCTS_DIR}/${FRAMEWORKS_FOLDER_PATH}/Realm.framework/strip-frameworks.sh"

realm

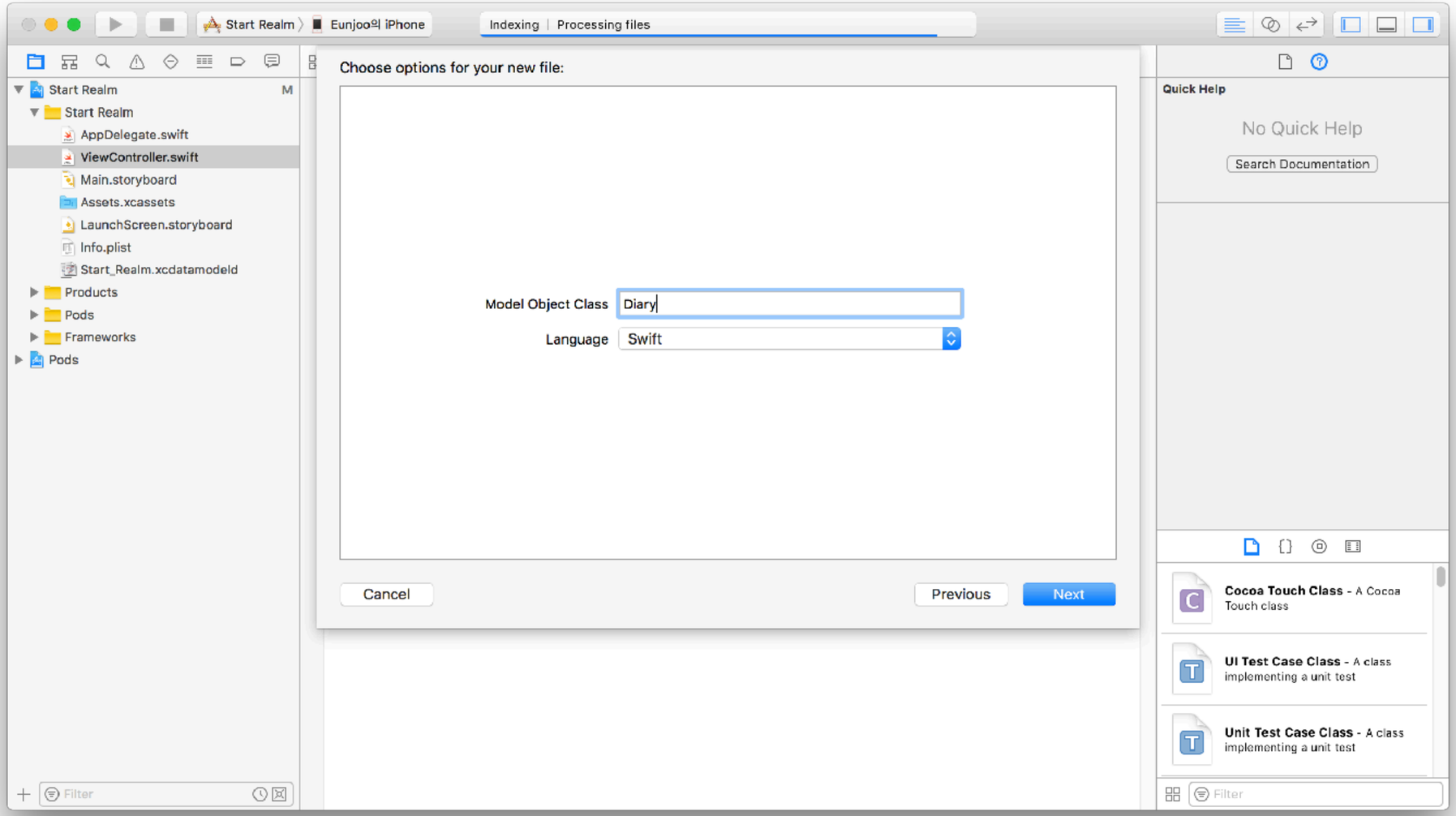
eunjoo.im@realm.io

Realm 모델 만들기



eunjoo.im@realm.io

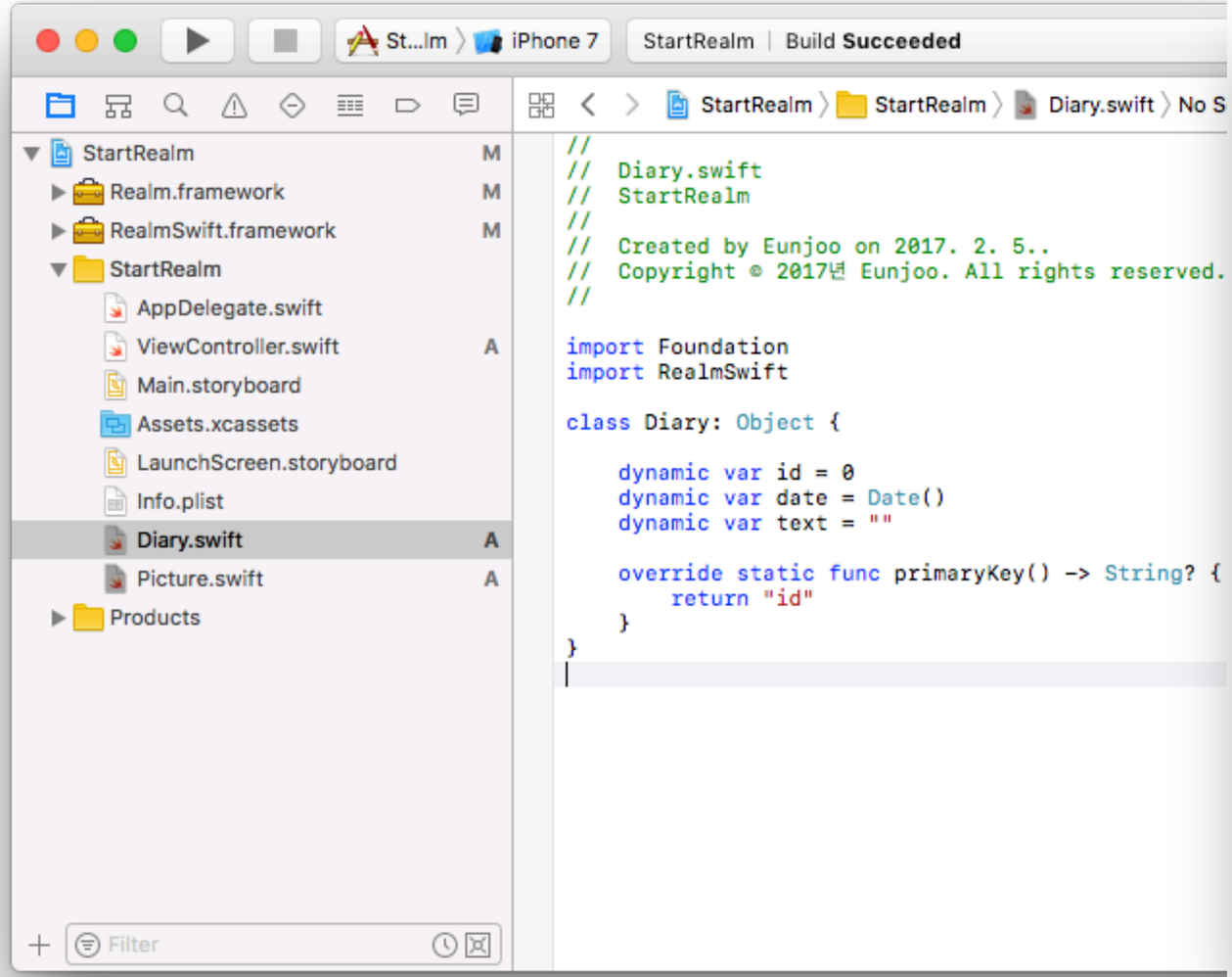
Realm 모델 만들기

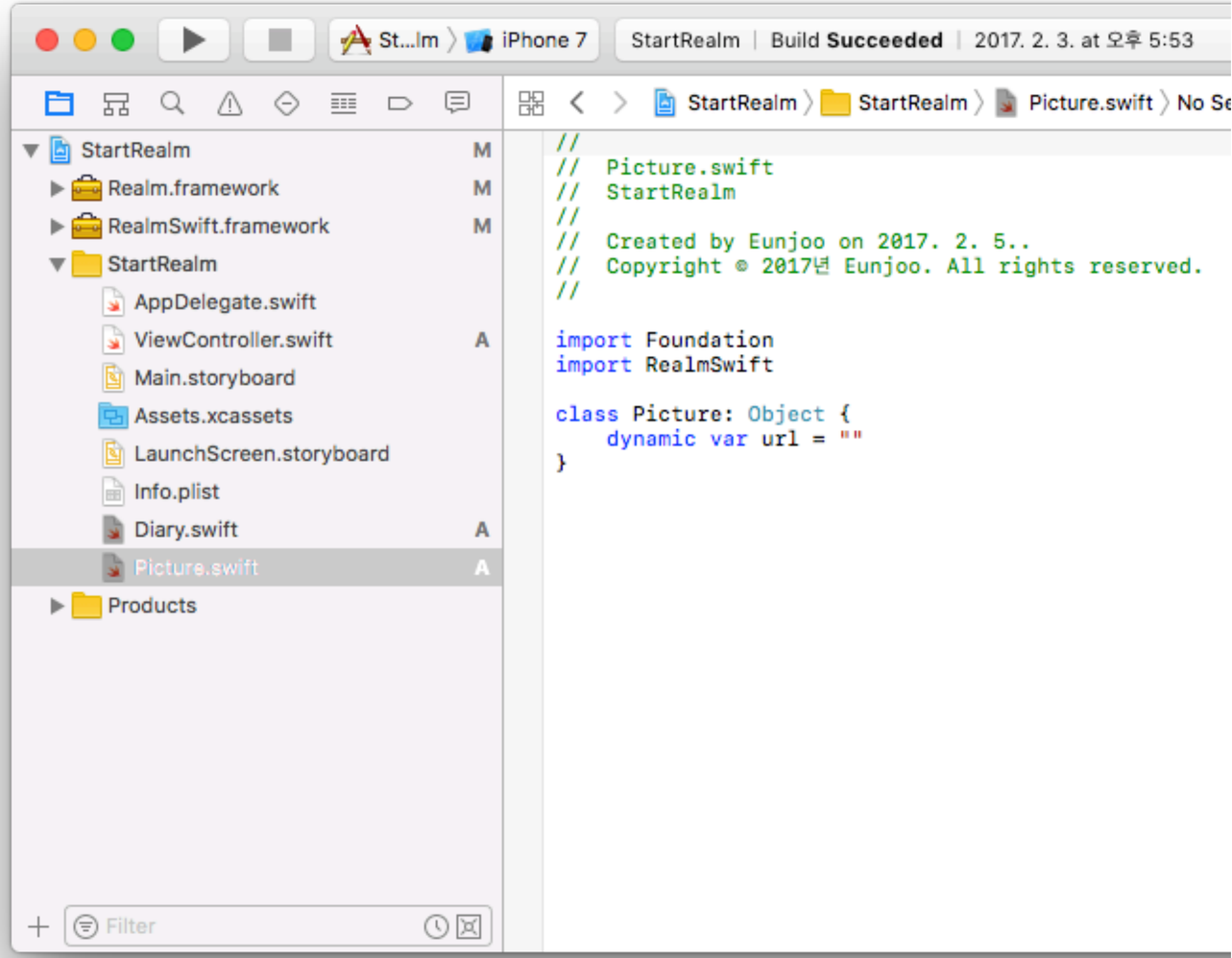


realm

eunjoo.im@realm.io

Realm 모델 만들기





realm

eunjoo.im@realm.io

데이터 쓰기(1)

```
func addDiary() {
    // 기본 Realm을 가져옵니다.
    let realm = try! Realm()

    // Picture 객체를 만들고 값을 넣습니다.
    let testPicture1 = Picture(value:["01.jpg"])
    let testPicture2 = Picture(value:["02.jpg"])

    // Diary 객체를 만들고 값을 넣습니다.
    let myDiary = Diary(value: [incrementID(), Date(), "안녕하세요?", [testPicture1, testPicture2]]);

    // 트랜잭션 안에서 Realm에 Diary 객체를 추가합니다.
    try! realm.write {
        realm.add(myDiary)
    }

    // Realm 경로를 찾습니다.
    print(Realm.Configuration.defaultConfiguration.fileURL!)
}
```


eunjoo.im@realm.io

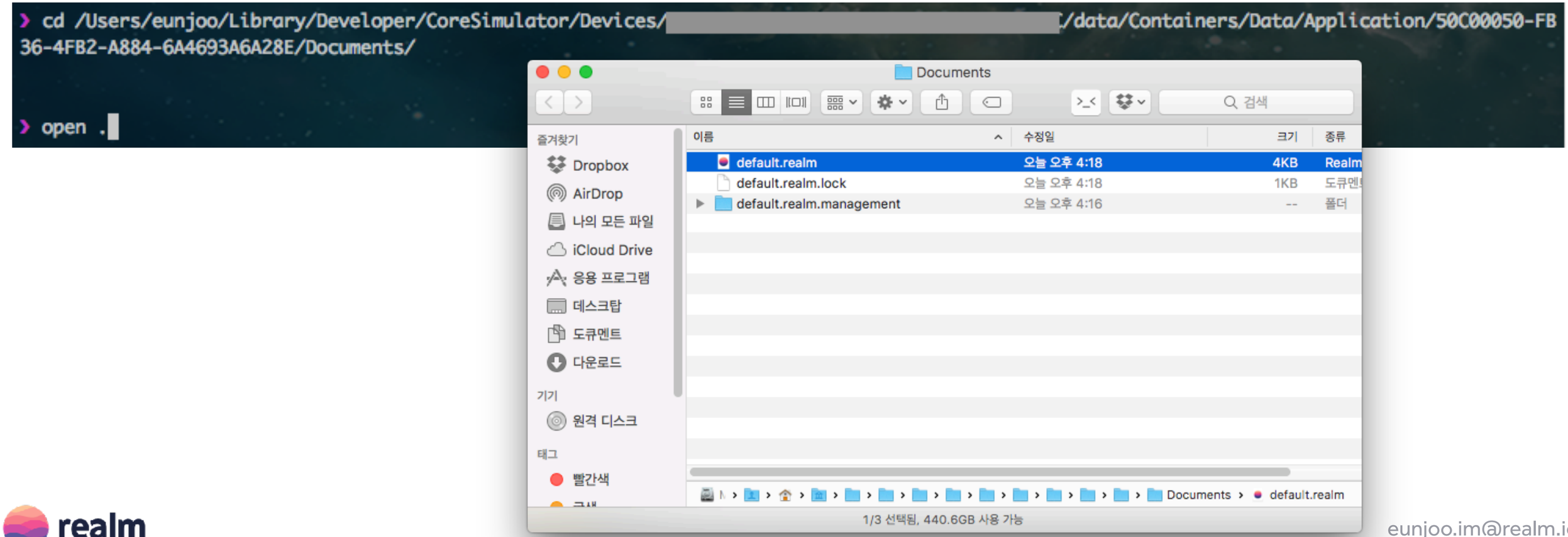
데이터 쓰기(2)

```
class ViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()
        addDiary()
    }

    func addDiary() {
        //
    }

    // ID 값을 증가시킵니다.
    func incrementID() -> Int {
        let realm = try! Realm()
        return (realm.objects(Diary.self).max(ofProperty: "id") as Int? ?? 0) + 1
    }
}
```

realm

eunjoo.im@realm.io

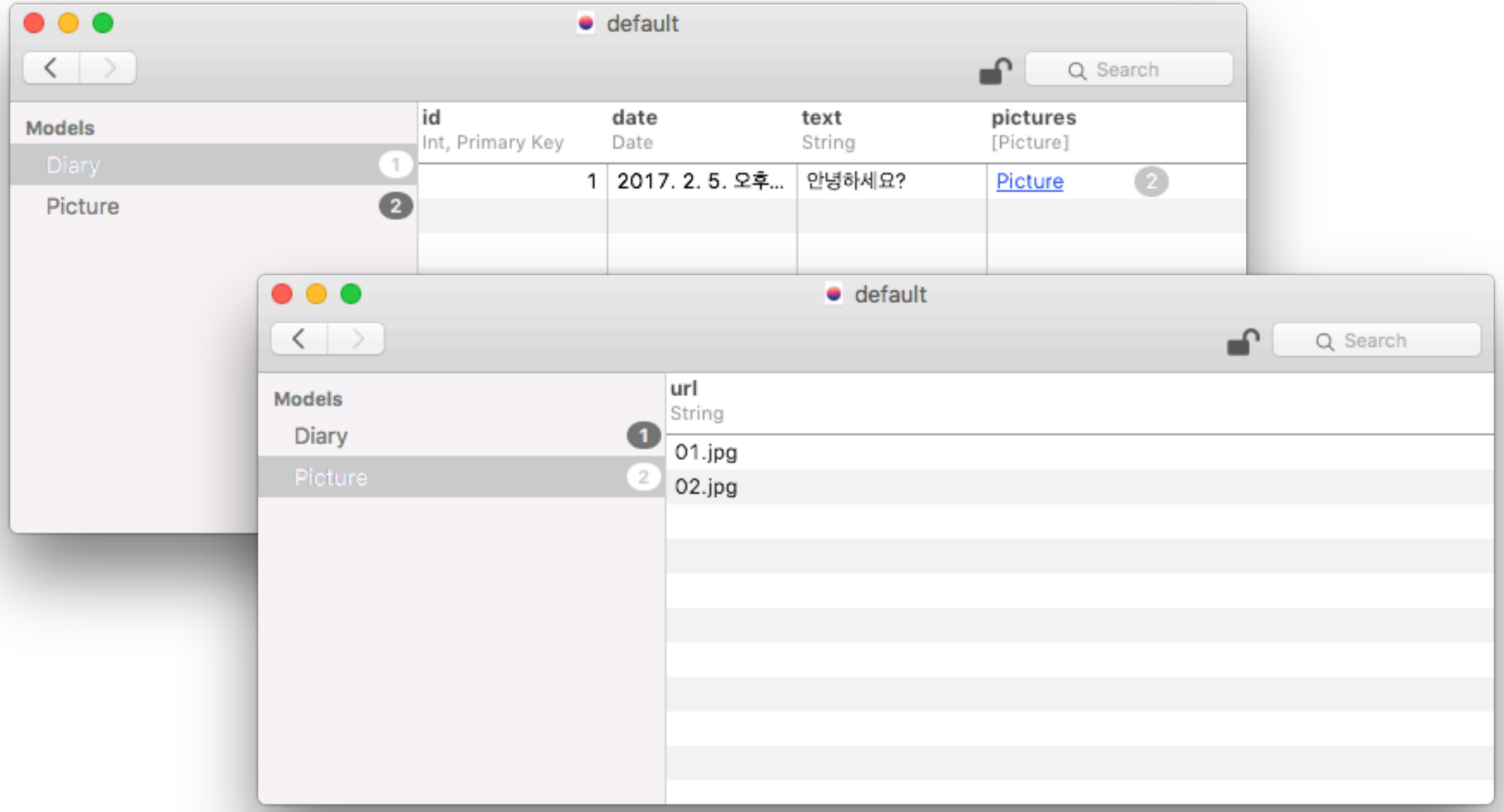
데이터 확인(1)



realm

eunjoo.im@realm.io

데이터 확인(2)



realm

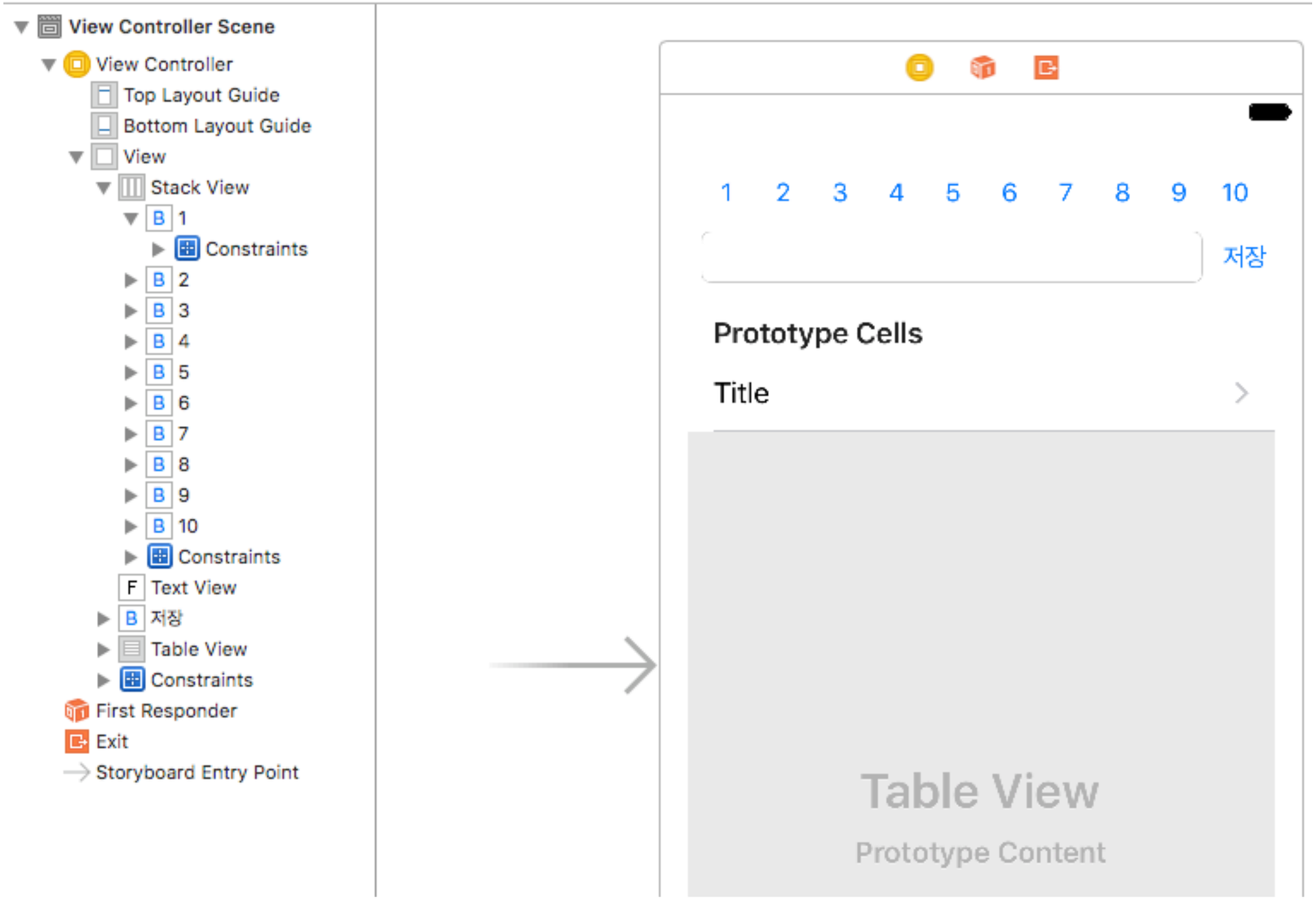
eunjoo.im@realm.io

데이터 읽기

```
func readDiary() {
    // 기본 Realm을 가져옵니다.
    let realm = try! Realm()

    // 모든 Diary 데이터를 읽습니다. 반환값: Result
    let diaries = realm.objects(Diary.self)

    //결과값에 들어있는 각 객체를 읽고 텍스트를 출력합니다.
    for diary in diaries {
        let text = diary.text
        print(text)
    }
}
```


eunjoo.im@realm.io

Diary 저장 UI 만들기(1)



eunjoo.im@realm.io

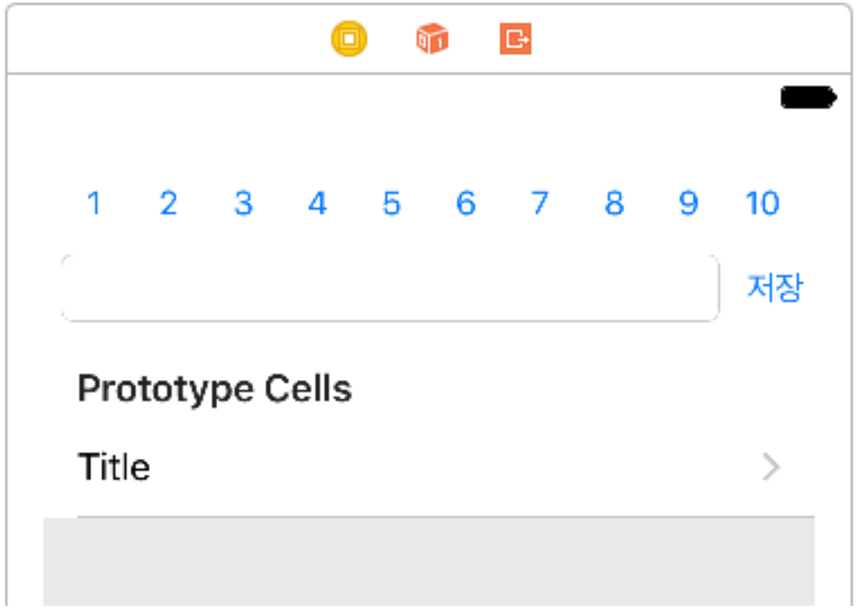
Diary 저장 UI 만들기(2)

```
class ViewController: UIViewController {
    @IBOutlet weak var textView: UITextField!
    @IBOutlet weak var tableView: UITableView!

    var diaries: Results<Diary>!
    var picturesArray = NSMutableArray()
    override func viewDidLoad() {
        super.viewDidLoad()
    }

    @IBAction func picOneButton(_ sender: Any) {
        picturesArray.add("01.jpg")
    }

    @IBAction func picTwoButton(_ sender: Any) {
        picturesArray.add("02.jpg")
    }
```

```
@IBAction func saveButton(_ sender: Any) {
    // picturesArray 대로 Picture 객체를 만들고 프로퍼티 값을 넣습니다.

    let photos = List<Picture>()
    for fileName in picturesArray {
        let picture = Picture()
        picture.url = fileName as! String
        photos.append(picture)
    }

    // Diary 객체를 만들고 사용자가 입력한 프로퍼티 값을 넣습니다.
    let myDiary = Diary(value: [incrementID(), Date(), textView.text ??
        "", photos]);

    // Realm에 Diary 객체를 저장합니다.
    addDiary(diary: myDiary)

    // 사용자 입력 정보를 초기화합니다.
    textView.text = ""
    picturesArray = NSMutableArray()
    self.tableView.reloadData()
}

func addDiary(diary: Diary) {
    // 기본 Realm을 가져옵니다.
    let realm = try! Realm()

    // 트랜잭션 안에서 Realm에 Diary 객체를 추가합니다.
    try! realm.write {
        realm.add(diary)
    }
}
```


eunjoo.im@realm.io

Diary 테이블뷰 만들기(1)




```swift
override func viewDidLoad() {
    super.viewDidLoad()
    self.tableView.delegate = self
    self.tableView.dataSource = self
}

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

func tableView(_ tableView: UITableView, cellForRowAt indexPath:
    IndexPath) -> UITableViewCell {
    let cell = UITableViewCell(style: UITableViewCellStyle.default,
        reuseIdentifier: "tableCell")
    cell.textLabel?.text = diaries[indexPath.row].text
    return cell
}

func tableView(_ tableView: UITableView, didSelectRowAt indexPath:
    IndexPath) {
    print(indexPath.row)
}
```


eunjoo.im@realm.io

Diary 테이블뷰 만들기(2)


```swift
import UIKit
import RealmSwift

class DetailViewController: UIViewController, UITableViewDelegate, UITableViewDataSource  {

    var myDiary: Diary!
    var photos = List<Picture>()

    @IBOutlet weak var diaryDate: UILabel!
    @IBOutlet weak var diaryText: UILabel!
    @IBOutlet weak var diaryTableView: UITableView!

    override func viewDidLoad() {
        super.viewDidLoad()
        self.diaryDate.text = dateAsString(date: self.myDiary.date)
        self.diaryText.text = self.myDiary.text
        self.photos = self.myDiary.pictures

        self.diaryTableView.delegate = self
        self.diaryTableView.dataSource = self
    }

    func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        print(photos.count)
        return photos.count
    }
```

realm

eunjoo.im@realm.io

Diary 테이블뷰 만들기(3)

```
func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell
    {
    let cell = UITableViewCell(style: UITableViewCellStyle.default, reuseIdentifier:
        "detailTableCell")
    cell.imageView?.image = UIImage(named: photos[indexPath.row].url)
    print(photos[indexPath.row].url)

    return cell
}

func dateAsString(date: Date) -> String {

    let date = NSDate()
    let dateFormatter = DateFormatter()
    dateFormatter.dateFormat = "yyyy-MM-dd HH:mm"
    let str = dateFormatter.string(from: date as Date)
    return str
}

@IBAction func clossButton(_ sender: Any) {
    self.dismiss(animated: true)
}
```


eunjoo.im@realm.io

간단 실습

1. 삭제 버튼을 만들고, 데이터를 업데이트하세요.
2. 수정 버튼을 만들고, 데이터를 삭제하세요.

```
func deleteDiary() {
    // 기본 Realm을 가져옵니다.
    let realm = try! Realm()

    // 트랜잭션 안에서 Realm의 Diary 객체를 지웁니다.
    try! realm.write {
        realm.delete(myDiary)
    }
}
```

```
func editDiary() {
    // 기본 Realm을 가져옵니다.
    let realm = try! Realm()

    // 트랜잭션 안에서 Realm의 Diary 객체를 업데이트합니다.
    try! realm.write {
        myDiary.text = ""
    }
}
```


eunjoo.im@realm.io

+ a

1. 임시로 만든 그림 파일을 실제로 Photo Library와 Camera에서 받아 보세요.
2. Realm에 직접 data를 넣을 수 있습니다. Photo에 url 대신 data를 넣을 수 있어요.

```
let jpgData = NSData(data: UIImageJPEGRepresentation(yourImage, 0.9))
let pngData = NSData(data: UIImagePNGRepresentation(yourImage))
```


eunjoo.im@realm.io

Questions?

Eunjoo Im
eunjoo.im@realm.io
www.realm.io/kr
https://www.facebook.com/realmkr