

TG-Springboot 스터디 × iOS 스터디

캘린더 앱 만들기



기획, 총괄 : 송승호
Spring-Boot 스터디 : 김선진, 이민호, 정지민, 조의연
iOS 스터디 : 공태윤, 신현수, 정윤철, 한동휘, 허원(mentor)

Spring-boot Study Group + iOS Study Group

Back-End +

필터 정렬 ⌂ ... 새로 만들기 ▾

Spring-Boot Study ...

- 8/8 스터디 스터디
- 8/15 스터디 스터디 데이터베이스
- 8/22 스터디 스터디
- 8/30 스터디 백엔드

+ 새로 만들기

필터 정렬 ⌂ ... 새로 만들기 ▾

iOS Study ...

- Swift 기본 문법 (7/10) 스터디
- 8/6 스터디 스터디
- 8/14 스터디 스터디
- 8/21 스터디 스터디
- Git 관련



기획 의도

배운 지식들을 활용해 실생활에서 사용할 수 있는 애플리케이션을 구현해보자!



iOS의 기본적인 앱들을 똑같이 만들어보는 Clone Coding을 통해 만들어보자!



TG Calendar

목표

iOS 스터디 팀

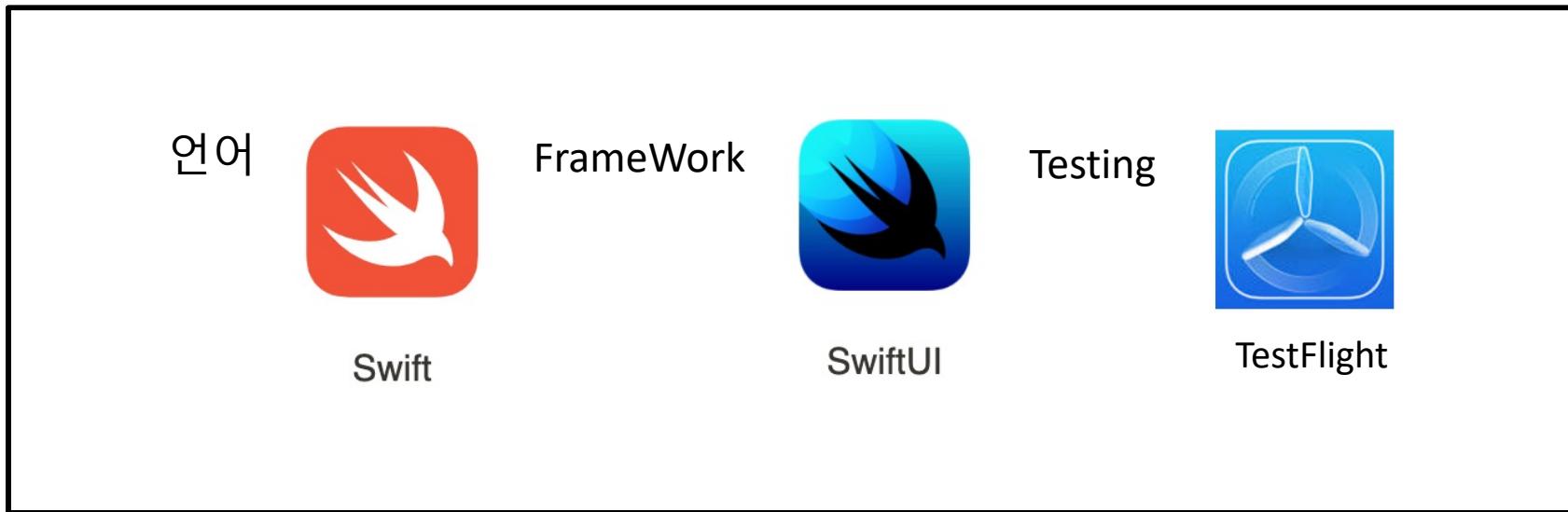
1. SwiftUI를 활용해 아이폰 기본 일정 앱의 전반적인 UI/UX 구현
2. 기본 일정 앱의 “월간 캘린더” 기능을 유사하게 구현
3. Server에 원하는 값을 요청하고, 상응하는 값을 서버로부터 받는 API 구축
4. 각종 시나리오 설계를 통한 지속적인 테스팅

Spring-Boot 스터디 팀

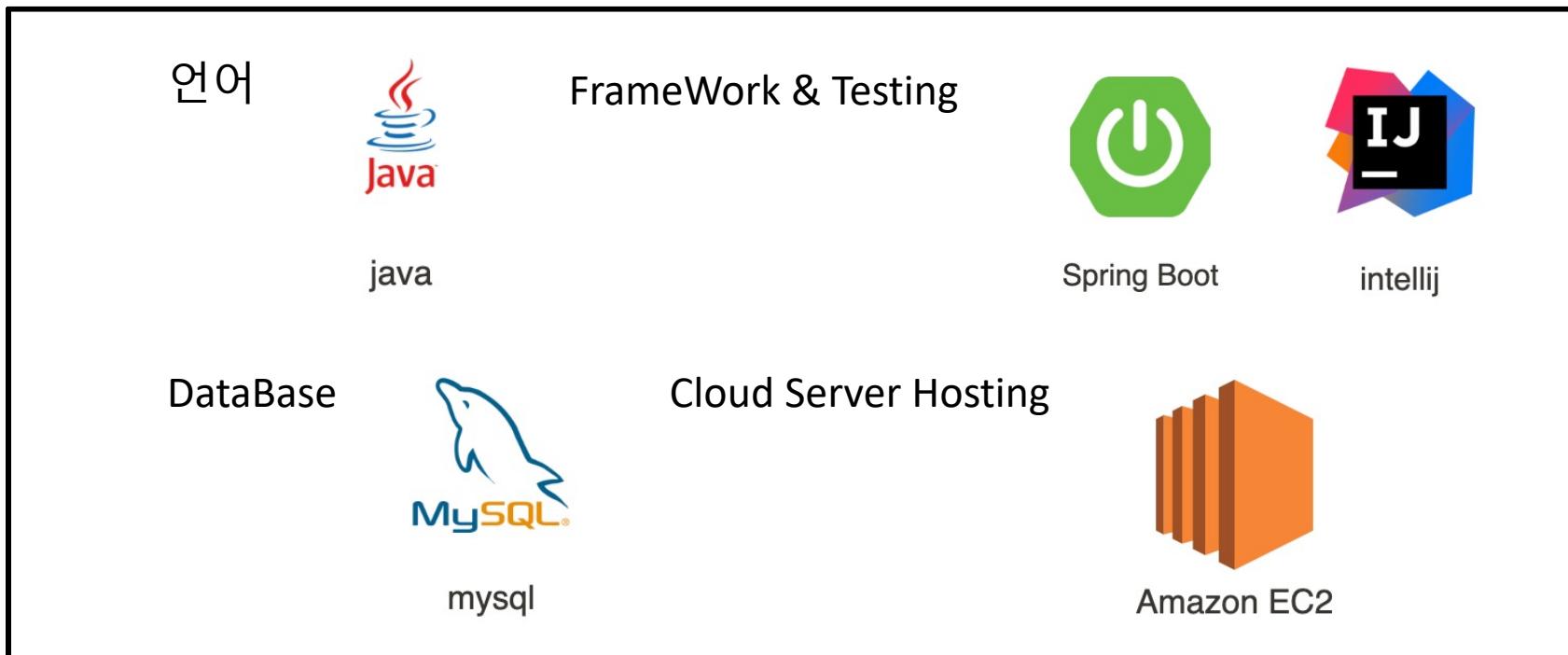
1. Spring-Boot를 통한 서버 구축
2. Client로부터 요청받는 값들을 처리, 응답
3. 앱의 전반적인 CRUD 기능 처리
4. MySQL을 활용한 DataBase 설계 및 구축
5. EC2 인스턴스를 생성하여 원격 클라우드 서버 구축

Stacks

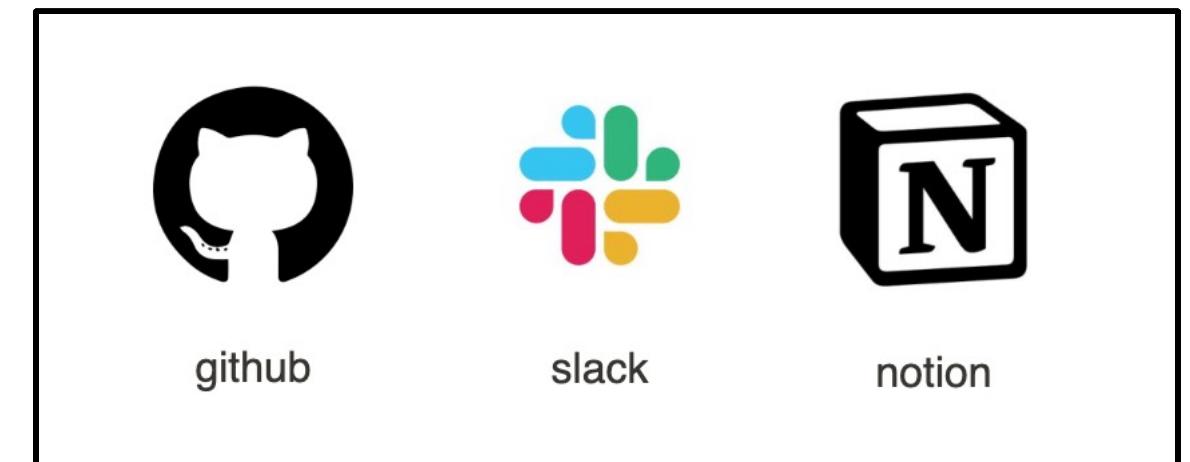
Client



Server



협업



Client – SwiftUI

TG iOS Study Group

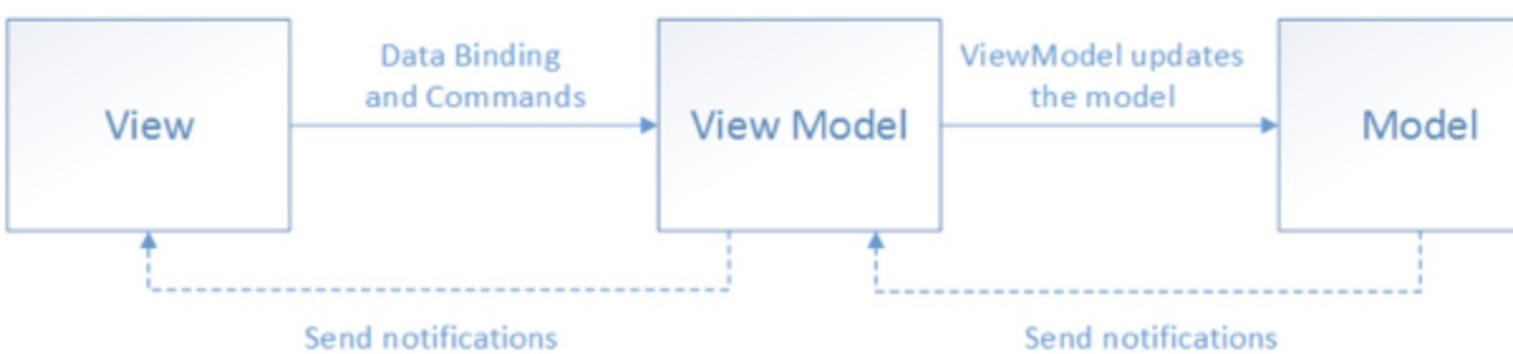
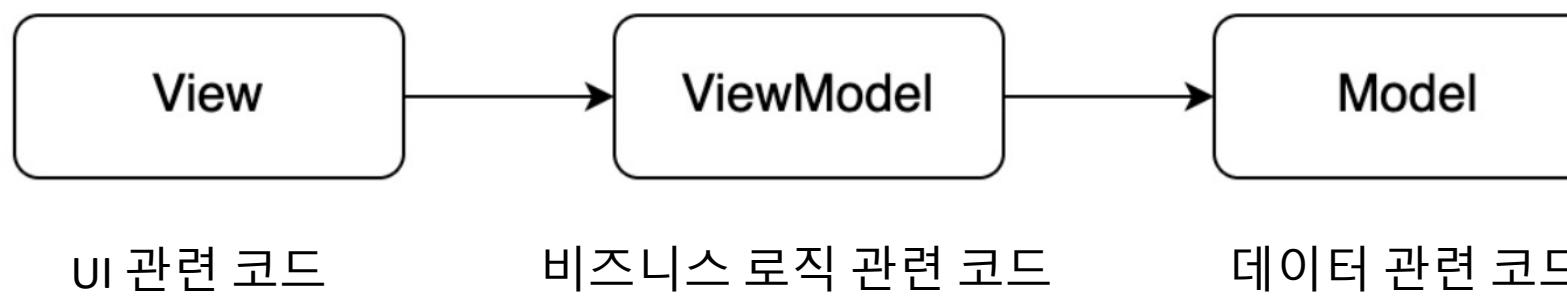
기능

iOS

- 다크 모드 활성화/비활성화 기능 ✗
- 전반적인 UI 구현
 - navigation bar
 - 뒤로 가기 버튼 → 추후 연간으로 넘어갈 수 있게끔 ✗
 - 일정 검색 버튼 ✓
 - 일정 추가 버튼 ✓
 - body - calender part
 - 요일 표시 ✓
 - 날짜 각각을 객체화 ✓
 - 월이 바뀌면 자동으로 해당 요일에 맞춰 날짜 표시
 - 날짜 개개인을 객체화하여 추후 해당 날짜 클릭 시 일정 볼 수 있게끔
 - 토→파랑 / 일,공휴일→빨강 / 평일→검정(다크 모드: 흰색) ✓
 - 오늘 날짜에 빨간 동그라미 + 흰색 글씨 ✓
 - 일정이 있는 날짜엔 아래 점으로 표시 ✓
 - body - schedule part
 - 날짜가 아무것도 눌리지 않은 초기 상태는 "이벤트 없음" 이라고 표시 ✓
 - 날짜가 클릭이 되었을 때 ✓
 - 해당 날에 이벤트 있으면 이벤트를 표시 (리스트)
 - 해당 날에 이벤트 없으면 초기 상태 유지

MVVM Pattern

Model – View – ViewModel

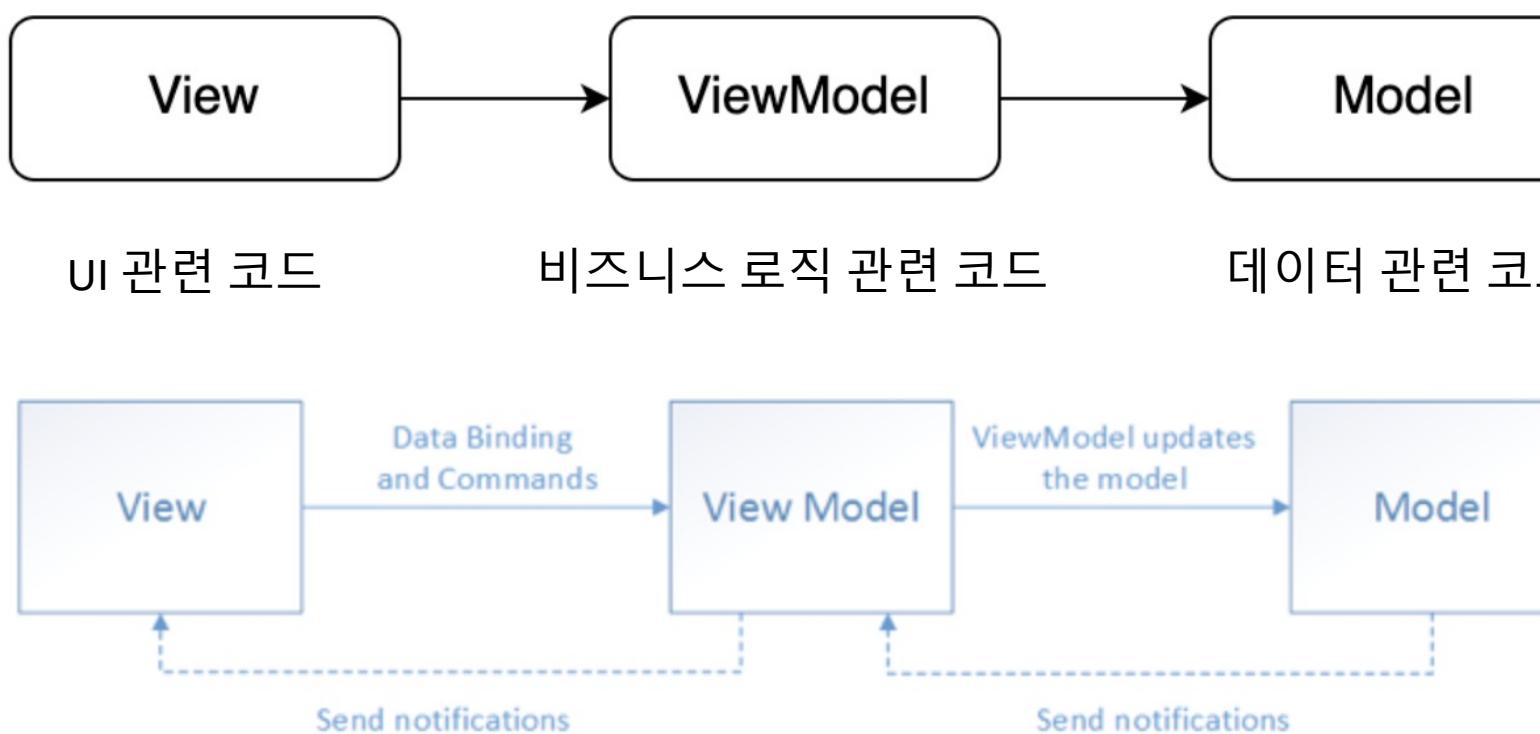


Features

1. UI를 구성하는 View와, Data를 구성하는 Model 사이의 의존성 X
-> 독립적으로 움직일 수 있다.
2. 각 계층들이 모듈화가 되어 있으므로, 코드 가독성 UP
3. ViewModel 코드 구현 난이도 HARD

MVVM Pattern

Model – View – ViewModel



Ex. Sign In 관련 코드

```

29 enum CalendarAPI {
30   case login(id: String, password: String)
31   case join(id: String, password: String, name: String, email: String)
32   case registerSchedule(title: String, content: String, startDate: Date, endDate: Date)
33   case getSchedule
34   case updateSchedule(id: Int, title: String, content: String, startDate: Date, endDate: Date)
35   case getMe
36   case updateMe(email: String?, nickName: String?, password: String?)
37
38 var baseURL: String { "http://3.39.197.209:8080" }
39
40 var path: String {
41   switch self {
42   case .login: return "/login"
43   case .join: return "/join"
44   case .registerSchedule: return "/schedule"
  
```

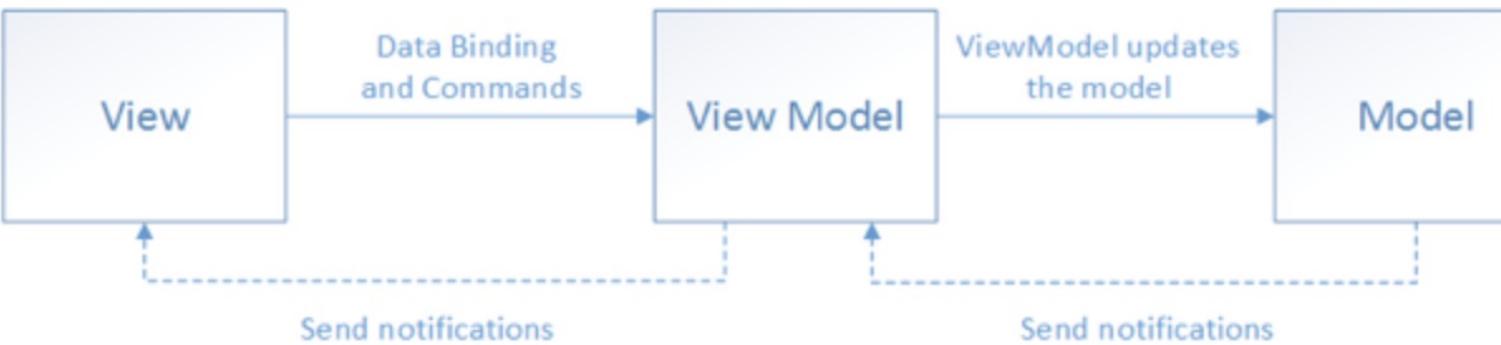
<Model>

MVVM Pattern

Model – View – ViewModel



UI 관련 코드



비즈니스 로직 관련 코드

데이터 관련 코드

Ex. Sign In 관련 코드

```

1 // 
2 //  SignInView.swift
3 //  Calendar1
4 //
5 //  Created by Coldot on 2022/09/21.
6 //
7
8 import SwiftUI
9
10
11 struct SignInView: View {
12     @ObservedObject var viewModel: SignInViewModel
13
14     init(_ viewModel: SignInViewModel) {
15         self.viewModel = viewModel
16     }
17
18     var body: some View {
19         ZStack {
20             // background layer
21             Color.gray
22                 .opacity(0.075)
23                 .ignoresSafeArea()
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
  
```

<View>

```

func signInButtonTapped() {
    guard id.isEmpty == false && password.isEmpty == false else {
        self.alertMessage = "아이디 및 패스워드를 입력해주세요"
        return
    }

    Task {
        do {
            let response: LoginResponse = try await Promise.shared.request(.login(id:
                self.id, password: self.password))

            if response.status == "OK" {
                self.jwt = response.jwt
                self.globalRouter.screen = .calendar
            } else {
                self.alertMessage = response.message
            }
        } catch let error {
            self.alertMessage = error.localizedDescription
        }
    }
}
  
```

<ViewModel>

Data Parsing

서버 API로부터 데이터 받아 오기

Routing

```
40  var path: String {  
41      switch self {  
42          case .login: return "/login"  
43          case .join: return "/join"  
44          case .registerSchedule: return "/schedule"  
45          case .getSchedule: return "/schedule"  
46          case .updateSchedule: return "/update-schedule"  
47          case .getMe: return "/user/me"  
48          case .updateMe: return "/user/me"  
49      }  
50 }
```

HTTP Method 지정

```
var method: HTTPMethod {  
    switch self {  
        case .login: return .post  
        case .join: return .post  
        case .registerSchedule: return .put  
        case .getSchedule: return .get  
        case .updateSchedule: return .put  
        case .getMe: return .get  
        case .updateMe: return .put  
    }  
}
```

Data Parsing

서버 API로부터 데이터 받아 오기

각 Case 에 맞게
Status Code 및 Message 응답

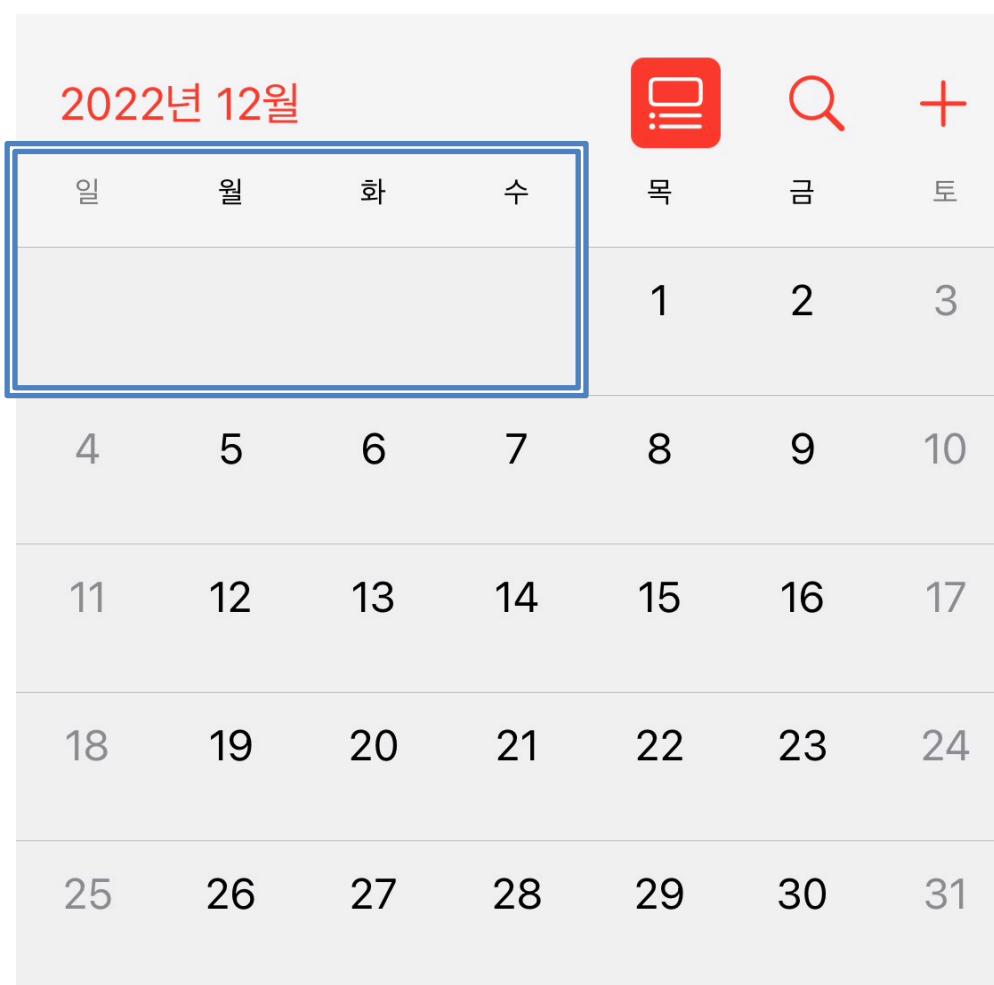
```
149 extension CalendarAPI {  
150     var mock: Data {  
151         switch self {  
152             case .login:  
153                 return Data(  
154                     """  
155                     {  
156                         "status" : 200,  
157                         "message" : "로그인 성공"  
158                     }  
159                     """".utf8  
160                 )  
161             case .join:  
162                 return Data(  
163                     """  
164                     {  
165                         "data" : "test",  
166                         "status" : 200,  
167                         "message" : "회원가입 성공"  
168                     }  
169                     """".utf8  
170                 )  
171             case .getMe:  
172                 return Data(  
173                     """  
174                     {  
175                         "data": {  
176                             "id": "test",  
177                             "email": "test@example.com",  
178                             "nickName": "Test"  
179                         },  
180                         "status": 200,  
181                     }  
182                 )  
183             case .getEvents:  
184                 return Data(  
185                     """  
186                     {  
187                         "events": [  
188                             {  
189                                 "id": "test",  
190                                 "title": "Meeting with team",  
191                                 "start": "2023-10-01T10:00:00",  
192                                 "end": "2023-10-01T11:00:00",  
193                                 "location": "Conference Room A"  
194                             }  
195                         ]  
196                     }  
197                     """".utf8  
198                 )  
199             case .updateEvent:  
200                 return Data(  
201                     """  
202                     {  
203                         "status": 200,  
204                         "message": "Event updated successfully"  
205                     }  
206                     """".utf8  
207                 )  
208             case .deleteEvent:  
209                 return Data(  
210                     """  
211                     {  
212                         "status": 200,  
213                         "message": "Event deleted successfully"  
214                     }  
215                     """".utf8  
216                 )  
217         }  
218     }  
219 }
```

Calendar 구현

CalendarHelper.swift

Feature 1

매월 1일에 해당하는 요일을 그리드에 배치 시,
시작일 앞에 몇 개의 빈칸을 넣어줘야 할지 구현



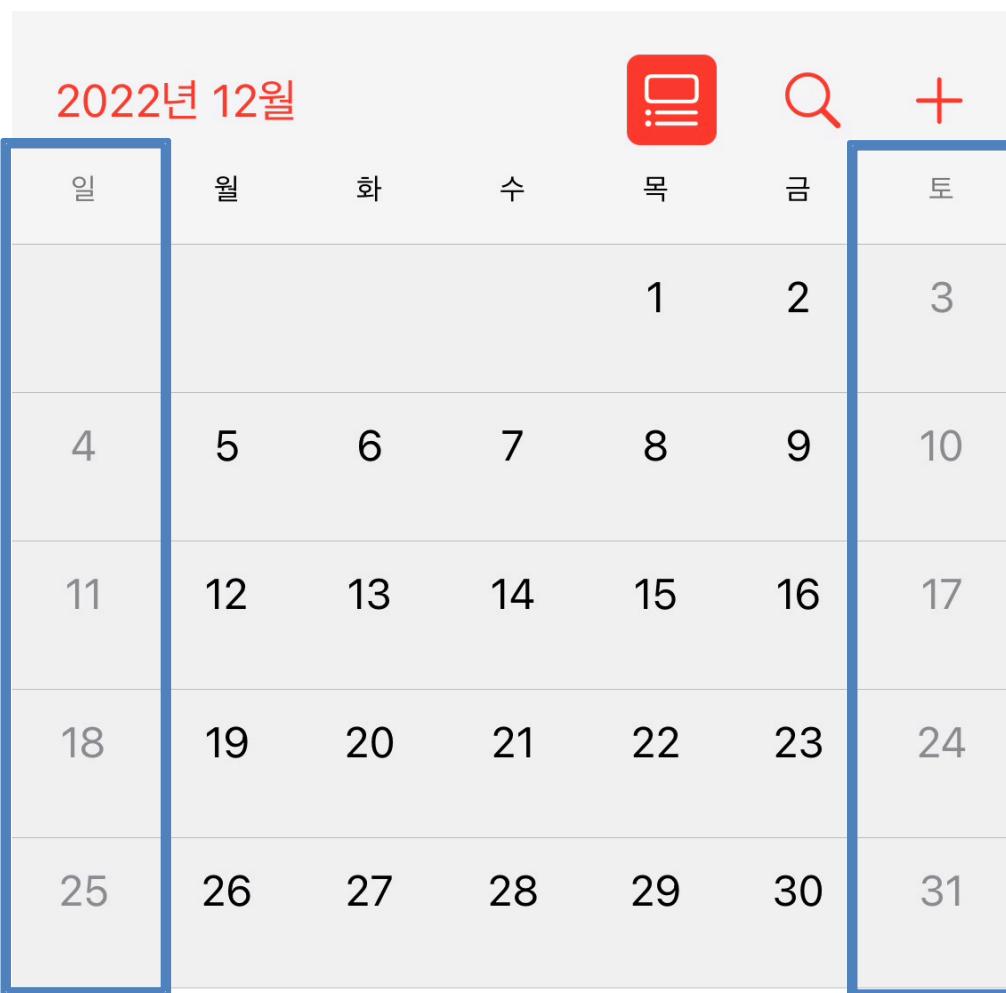
```
43 static func getFirstOfMonth(_ date: Date) -> Date {
44     let components = calendar.dateComponents([.year, .month], from: date)
45     return calendar.date(from: components)!
46 }
47
48 static func getWeekday(_ date: Date) -> Int {
49     let components = calendar.dateComponents([.weekday], from: date)
50     return components.weekday!
51 }
52
53 static func getFirstWeekdayOfMonth(_ date: Date) -> Int {
54     let firstOfMonth = getFirstOfMonth(date)
55     return getWeekday(firstOfMonth)
56 }
57
58 static func getWeekendDaysInMonth(_ date: Date) -> [Int] {
59     let firstWeekday = getFirstWeekdayOfMonth(date)
60     let daysInMonth = getNumOfDaysInMonth(date)
61     let days = Array(1...daysInMonth).filter {
62         ($0 + (firstWeekday - 1)) % 7 == 0 // saturday
63         || ($0 + (firstWeekday - 2)) % 7 == 0 // sunday
64     }
65
66     return days
67 }
```

Calendar 구현

CalendarHelper.swift

Feature 2

주말(토, 일)에 해당하는 날에 대해
회색으로 표시하는 기능 구현



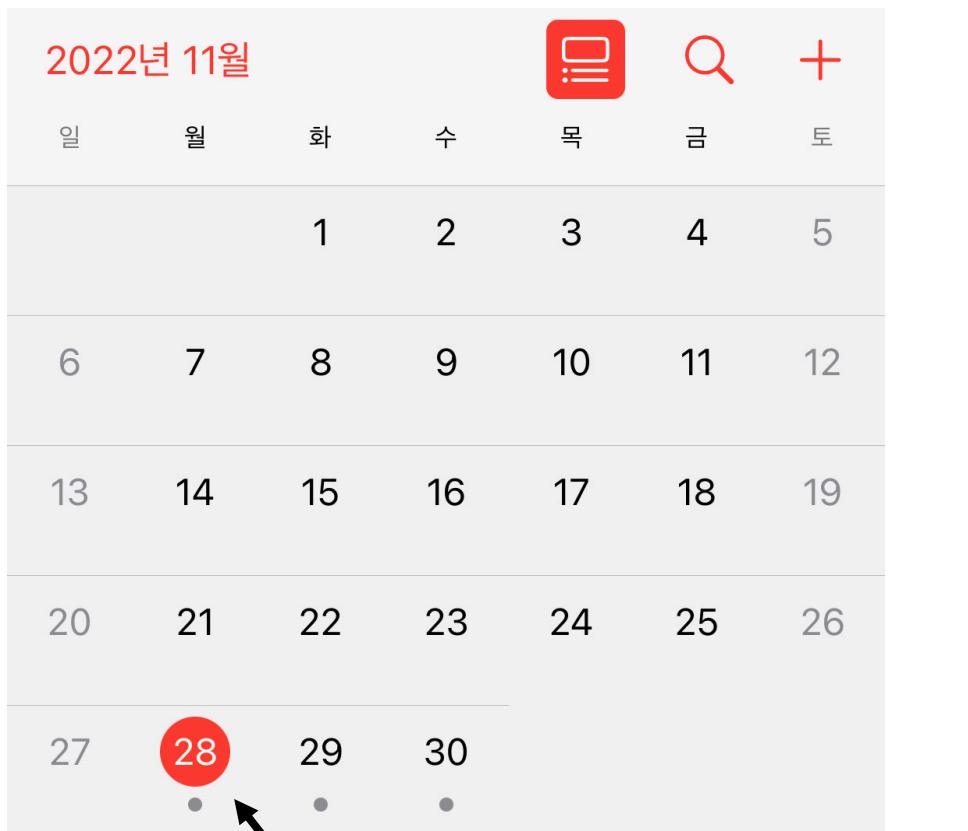
```
43     static func getFirstOfMonth(_ date: Date) -> Date {
44         let components = calendar.dateComponents([.year, .month], from: date)
45         return calendar.date(from: components)!
46     }
47
48     static func getWeekday(_ date: Date) -> Int {
49         let components = calendar.dateComponents([.weekday], from: date)
50         return components.weekday!
51     }
52
53     static func getFirstWeekdayOfMonth(_ date: Date) -> Int {
54         let firstOfMonth = getFirstOfMonth(date)
55         return getWeekday(firstOfMonth)
56     }
57
58     static func getWeekendDaysInMonth(_ date: Date) -> [Int] {
59         let firstWeekday = getFirstWeekdayOfMonth(date)
60         let daysInMonth = getNumOfDaysInMonth(date)
61         let days = Array(1...daysInMonth).filter {
62             ($0 + (firstWeekday - 1)) % 7 == 0 // saturday
63             || ($0 + (firstWeekday - 2)) % 7 == 0 // sunday
64         }
65
66         return days
67     }
```

Calendar 구현

MonthlyCalendarItem.swift

Feature 3

해당 날에 이벤트 추가 / 수정 시,
날짜 밑에 점이 찍히도록 구현



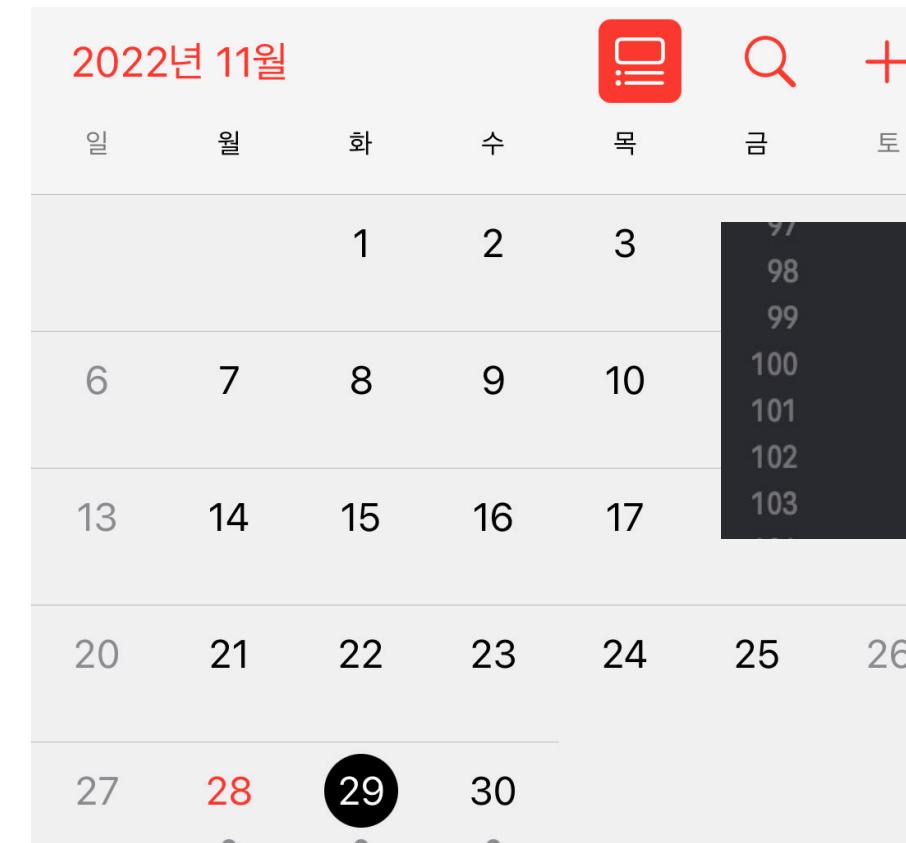
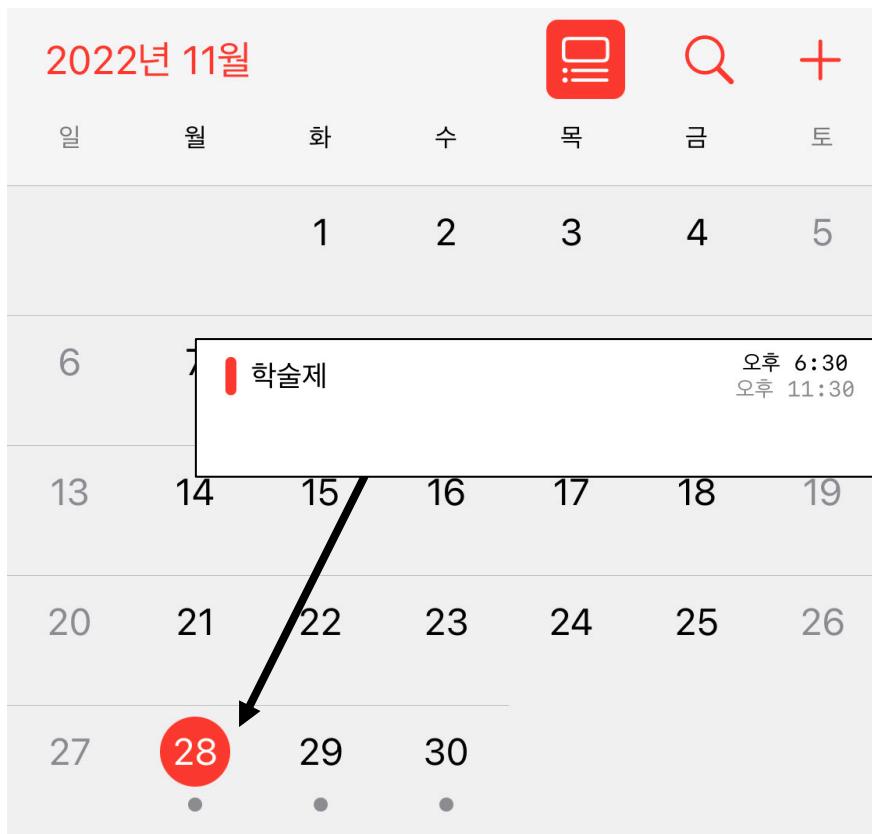
```
105 HStack {  
106     Circle()  
107     .foregroundColor(self.global.shouldShowCircle(date) ? Color.gray  
108     : Color.clear)  
109     .frame(width: 6, height: 6)  
110 }  
111 Spacer().frame(height: 10)  
112 }  
113 }  
114 }
```

Calendar 구현

CalendarHelper.swift

Feature 4

오늘 날짜에 해당하는 날은 빨간색 표시,
터치한 날에 해당하는 날은 검은색 원 표시 (오늘 날에 터치 시, 빨간색 원)

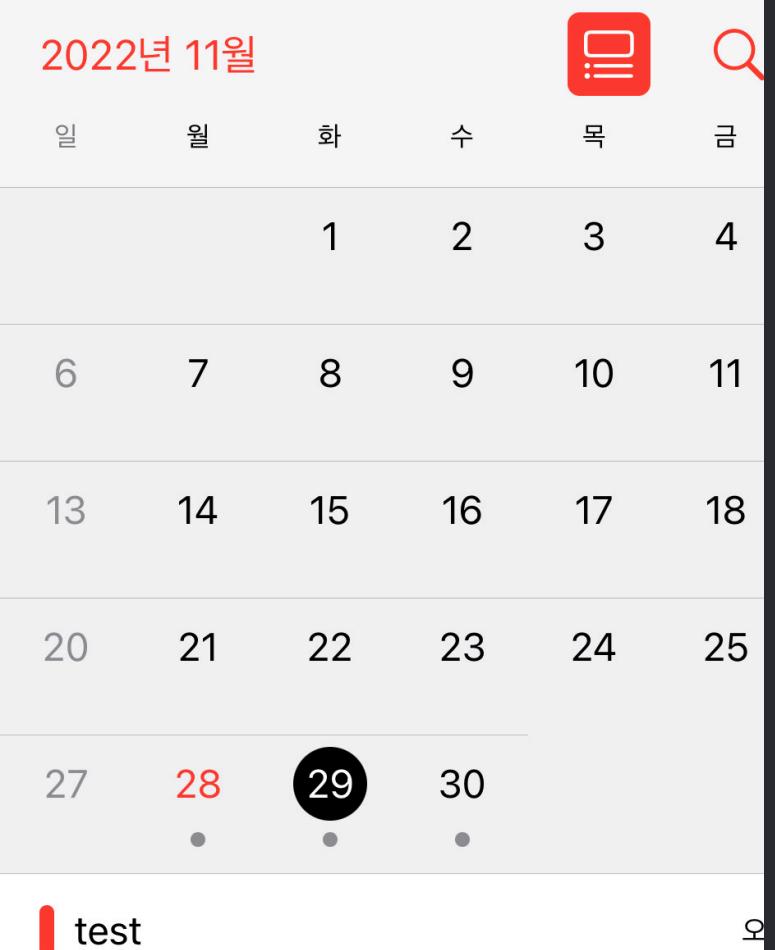
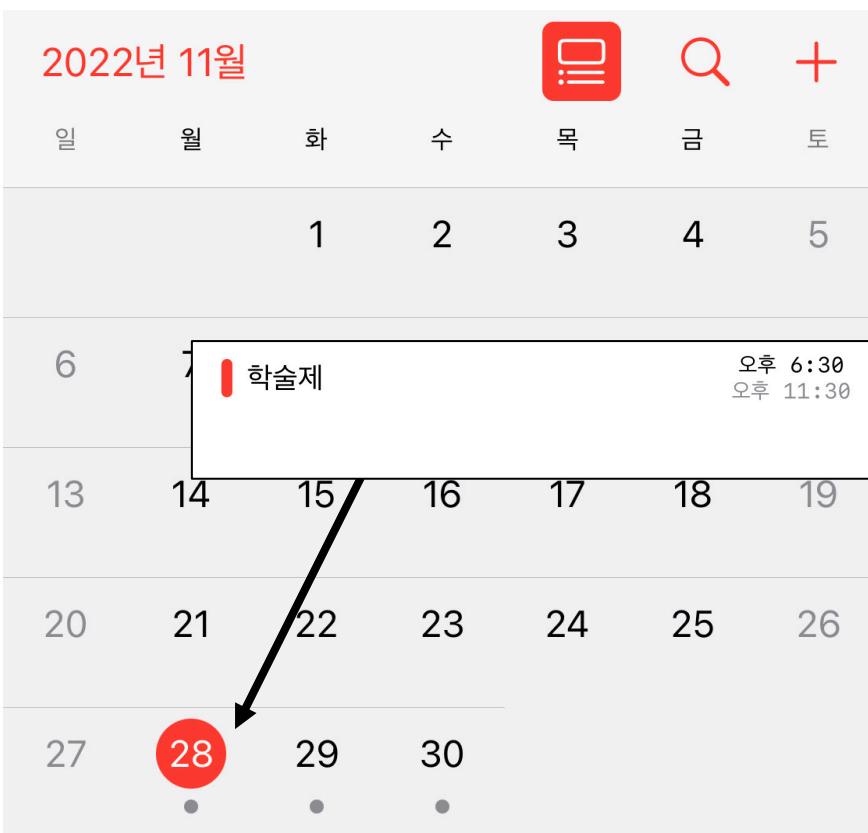


Calendar 구현

CalendarHelper.swift

Feature 4

오늘 날짜에 해당하는 날은 빨간색 표시,
터치한 날에 해당하는 날은 검은색 원 표시 (오늘 날에 터치 시, 빨간색 원)



```

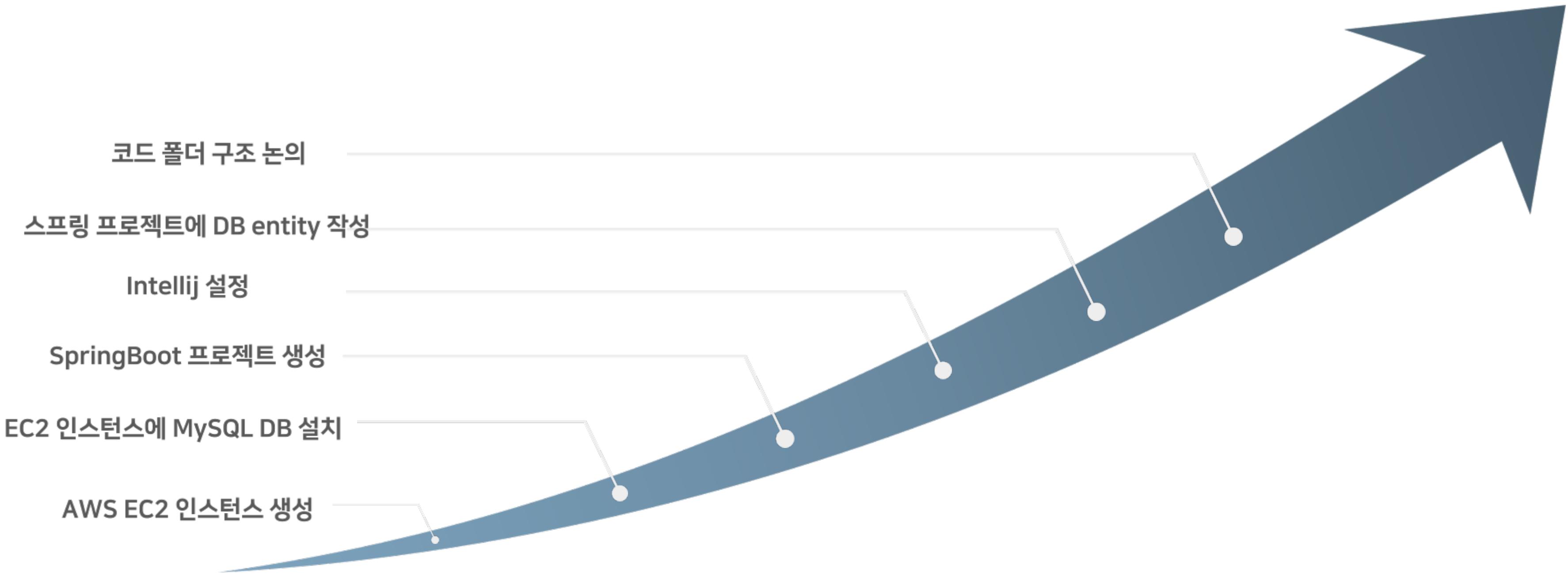
73         // real-value
74     ForEach(datesAndDaysOfMonth, id: \.1) { date, day in
75         VStack {
76             Divider()
77
78             // TODO: 이쪽 처리도 ViewModel과 함께 재구조화할 필요 있음
79             Text(day)
80                 .foregroundColor(
81                     // TODO: Date를 가지고 선택/오늘 여부 확인하도록 로직 변경 (VM 이동 후)
82                     (isSelectedMonth && (day == String(selectedDay)))
83                     ? (isCurrentMonth && (day == String(todayDay)))
84                         ? Color.white
85                         : Color.backgroundColor
86                     : (isCurrentMonth && (day == String(todayDay)))
87                         ? Color.accentColor
88                         : weekends.contains(day)
89                             ? Color.gray
90                             : nil
91
92                     .background(
93                         Circle()
94                             .foregroundColor(
95                                 (isSelectedMonth && (day == String(selectedDay)))
96                                 ? (isCurrentMonth && (day == String(todayDay)))
97                                     ? Color.accentColor
98                                     : nil
99                                     : Color.clear
100
101
102
103
104
    )
    .frame(width: 30, height: 30)
)
.padding(2)

```

Server – SpringBoot

TG Spring-Boot Study Group

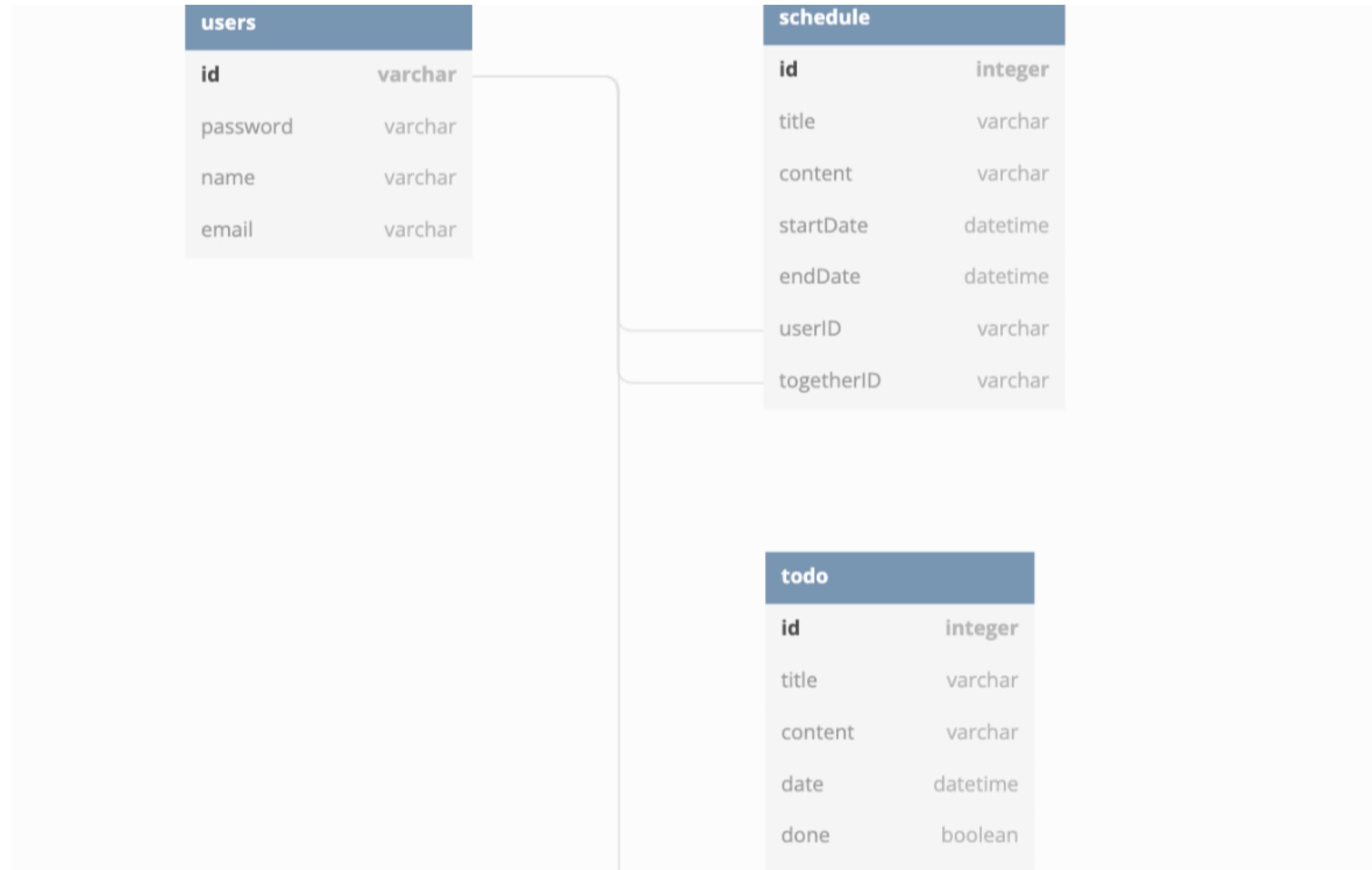
Server 서버 -초기 세팅



DB 설계

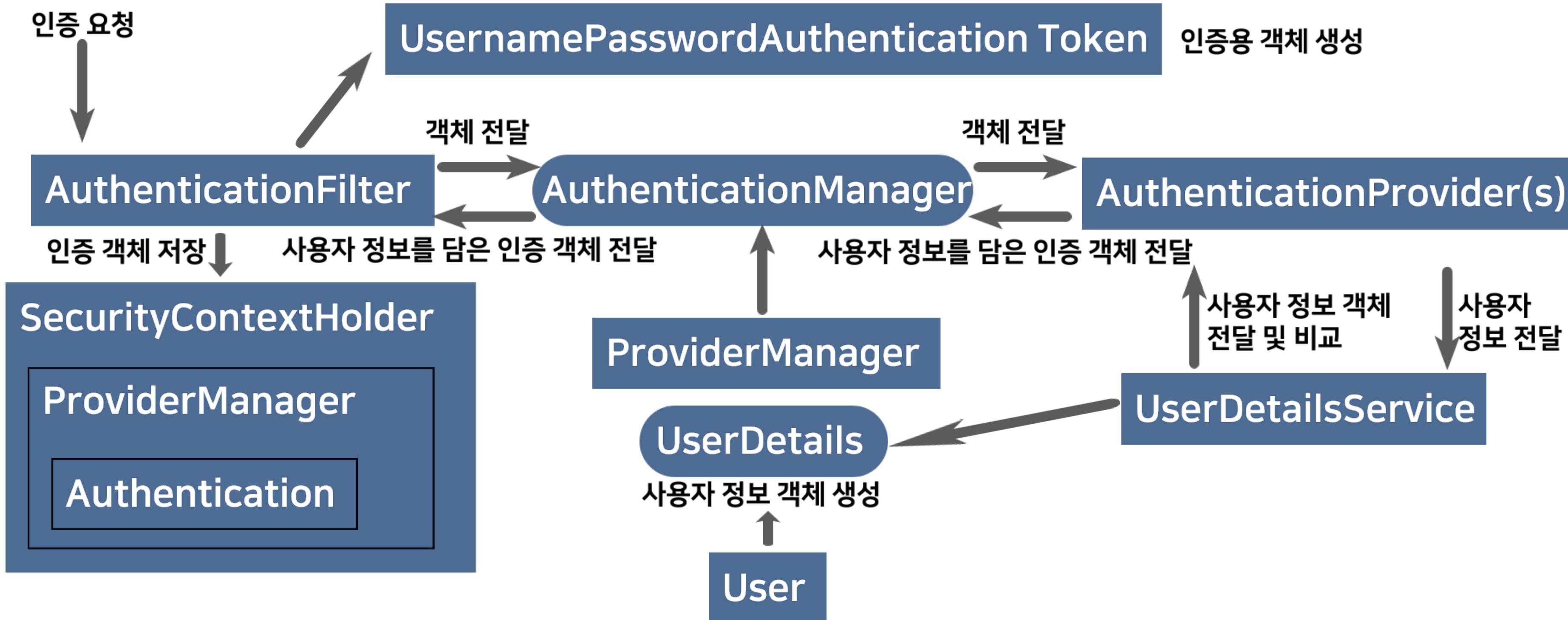
USER	SCHEDULE	TODO
<p>id 아이디 password 비밀번호 name 이름 email 이메일</p>	<p>id 일정 고유 번호 title 일정 제목 content 일정 관련 세부 내용 startDate 일정 시작일 endDate 일정 종료일 userID 일정 등록 사용자 togetherID 함께하는 사용자</p>	<p>id 고유번호 title 할일 제목 content 할일 관련 세부 내용 date 날짜 done 완료 여부 userID 등록한 사용자</p>

DB 설계



로그인 시 스프링 시큐리티를 통한 인증

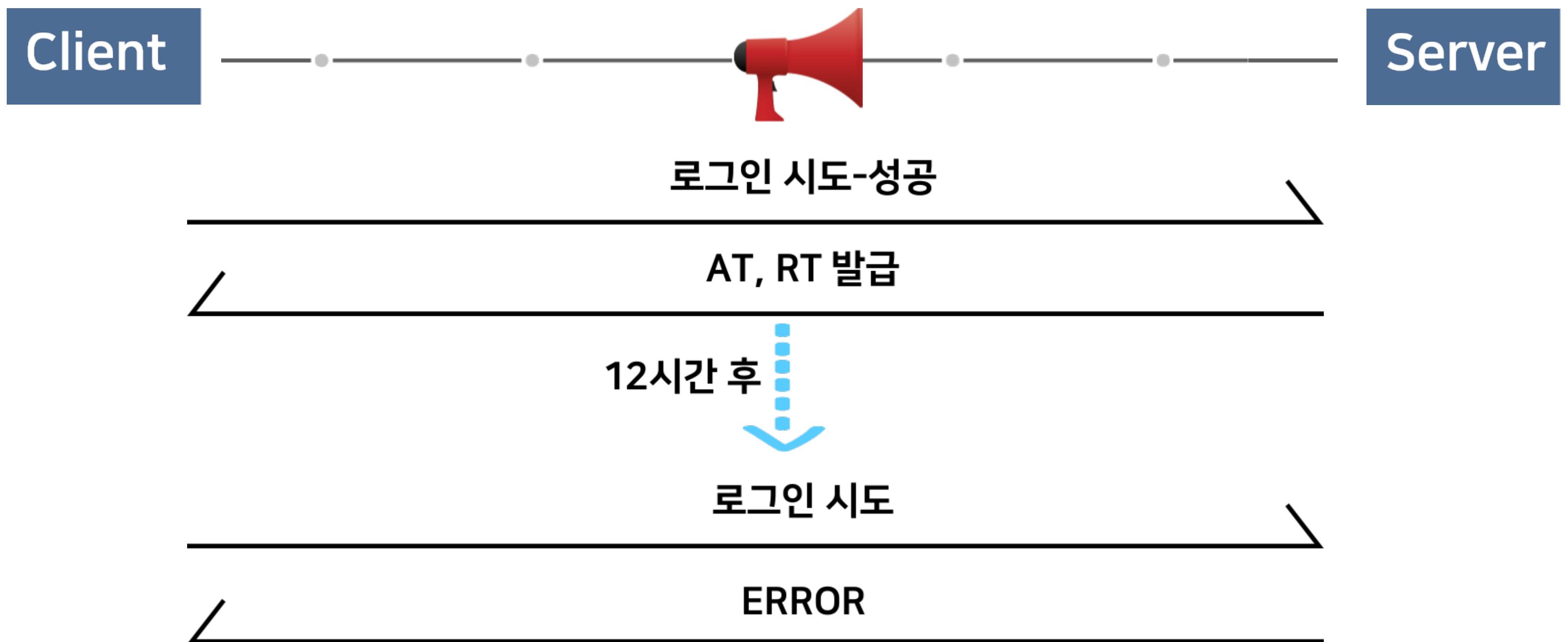
: 인터페이스



토큰(JWT) 기반 로그인-1

만료기간

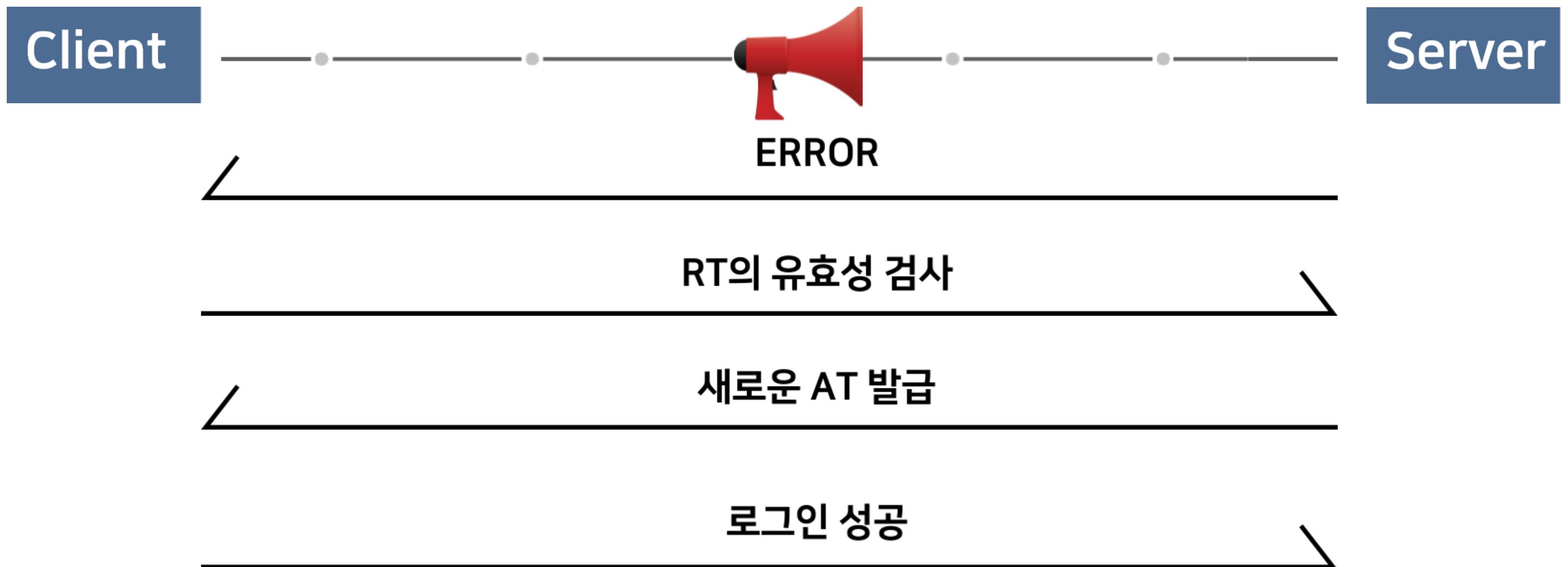
AT(Access Token) : 12시간
RT(Refresh Token) : 3개월



토큰(JWT) 기반 로그인-2

만료기간

AT(Access Token) : 12시간
RT(Refresh Token) : 3개월



AUTHENTICATION

API 설계 - 회원가입

url

POST /join

body

```
{  
  "id" : "string",  
  "password" : "string",  
  "name" : "string",  
  "email" : "string"  
}
```



response

```
{  
  "data" : {  
    등록한 유저 id (string)  
    ->JWT 토큰을 담고 있는 Mapping key type에 대해 상술  
  }  
  "status" : 200,  
  "message" : "회원가입 성공"  
}
```

AUTHENTICATION

API 설계 - 회원가입 예시 in POSTMAN

TGcalendar / 회원가입

The screenshot shows the POSTMAN interface for a POST request to `http://localhost:8080/join`. The request body is set to `form-data` and contains the following fields:

Key	Value
<code>id</code>	<code>test10</code>
<code>password</code>	<code>test10</code>
<code>nickName</code>	<code>test10</code>
<code>email</code>	<code>test10</code>

The response body is displayed in JSON format:

```
1 "status": "OK",
2 "message": "회원가입 성공",
3 "data": "test10"
```

A large blue arrow points from the 'body' section to the 'response' section.

USER

API 설계 - 내 정보 조회

url

GET /user/me

body

X

response

```
{  
  "data" : {  
    유저 정보  
  },  
  "status" : 200,  
  "message" : "내 정보 조회 성공"  
}
```

USER

API 설계 - 내 정보 조회 예시 in POSTMAN

The screenshot shows the POSTMAN interface with the following details:

- Request URL:** GET http://localhost:8080/user/me
- Headers (8):**
 - X-AUTH-TOKEN: eyJhbGciOiJIUzI1NiJ9.eyJzdWliOiJ0ZXN0M... (with a long token)
 - Content-Type: application/json
- Body:** The Body tab is not active.
- Response:** The response is displayed under the "Pretty" tab of the Body section.

```
1 "status": "OK",
2 "message": "내 정보 조회 성공",
3 "data": {
4   "id": "test1",
5   "nickName": "test1-fix",
6   "email": "test1"
7 }
```

A large blue arrow points from the "body" box on the left to the "response" box on the right, indicating the flow of data.

SCHEDULE

API 설계 - 전체 스케줄 반환

url

GET /schedule

body

X

response

```
{  
  "data" : {  
    "해당 유저의 모든 월간 스케줄 리스트 (array)  
  },  
  "status": 200,  
  "message" : "v"  
}
```

USER

API 설계 - 내 정보 조회 예시
in POSTMAN

TGcalendar / 내 정보 조회

Save

GET http://localhost:8080/user/me

Params Authorization Headers (8) Body Pre-request Script Tests Settings

Headers (6 hidden)

KEY	VALUE	DESCRIPTION	...	Bu
<input checked="" type="checkbox"/> X-AUTH-TOKEN	eyJhbGciOiJIUzI1NiJ9.eyJzdWlIiJ0ZXN0M... eyJhbGciOiJIUzI1NiJ9.eyJzdWlIiJ0ZXN0M...			
<input checked="" type="checkbox"/> Content-Type	application/json			
Key	Value	Description		

Body Cookies Headers (11) Test Results

Pretty Raw Preview Visualize JSON

```
1
2   "status": "OK",
3   "message": "유저의 모든 스케줄 리스트 반환 성공",
4   "data": [
5     {
6       "scheduleId": 1,
7       "id": 0,
8       "title": "test1",
9       "content": "test11",
10      "startDate": "2022-11-11 11:11",
11      "endDate": "2022-11-11 11:11",
12      "duration": "3",
13      "userId": "test1",
14      "togetherId": null
15    },
16    {
17      "scheduleId": 2,
18      "id": 0,
19      "title": "test2",
20      "content": "test22",
21      "startDate": "2022-11-11 11:11",
22      "endDate": "2022-11-11 11:11",
23      "duration": "3",
24      "userId": "test1",
25      "togetherId": null
26    }
]
```

body



response

SCHEDULE

API 설계 - 스케줄 수정

url

PUT /update-schedule

body

```
{  
  "schedule_id" : int,  
  "title" : string,  
  "content" : string,  
  "startDate" : Date,  
  "endDate" : Date,  
  "duration" : string,  
  "togetherId": string  
}
```

response

```
{  
  "data" : {  
    수정 성공 스케줄 id (int)  
  },  
  "status" : 200,  
  "message" : "Schedule 변경 성공"  
}
```

존재하는 스케줄일 경우

response

```
{  
  "data" : {},  
  "status" : 400,  
  "message" : "없는 스케줄입니다."  
}
```

존재하지 않는 스케줄일 경우

SCHEDULE

API 설계 - 스케줄 수정 예시 in POSTMAN

The screenshot shows the POSTMAN interface for a 'PUT' request to 'http://localhost:8080/update-schedule'. The 'Body' tab is selected, showing a JSON payload:

```
1 {  
2   ... "scheduleId": 9,  
3   ... "content": "test9-fix",  
4   ... "endDate": "2023-11-11 11:11"  
5 }
```

The response tab shows a successful update:

```
1 {  
2   "status": "OK",  
3   "message": "Schedule 변경 성공",  
4   "data": 9  
5 }
```

A large blue arrow points from the 'body' section to the 'response' section.

body



response

UI / UX

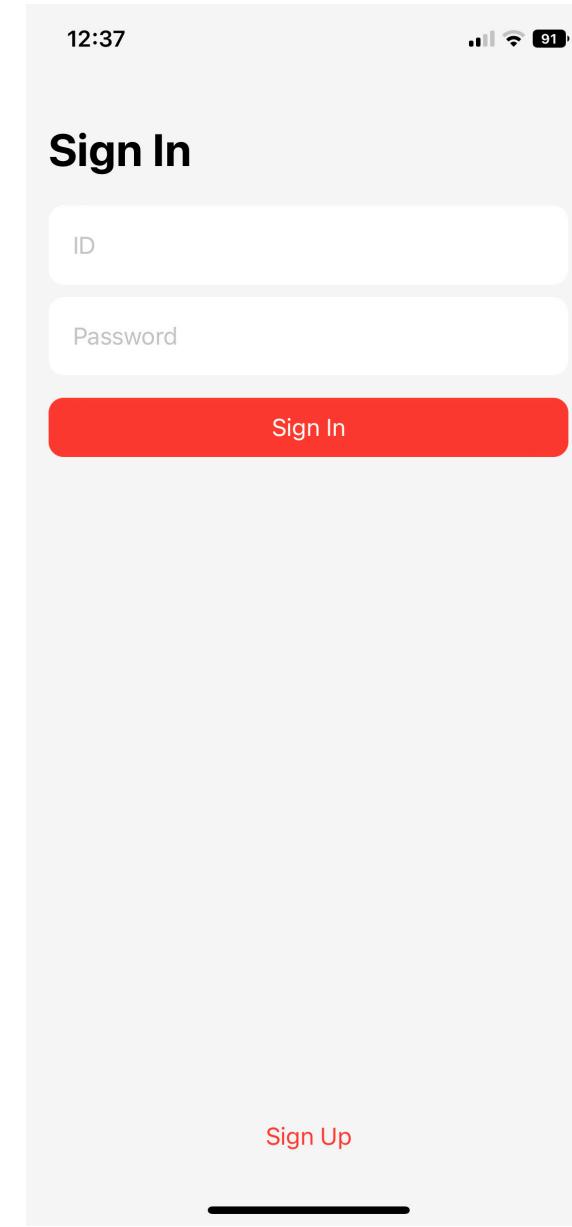
TG Spring-Boot Study Group

UI / UX 화면

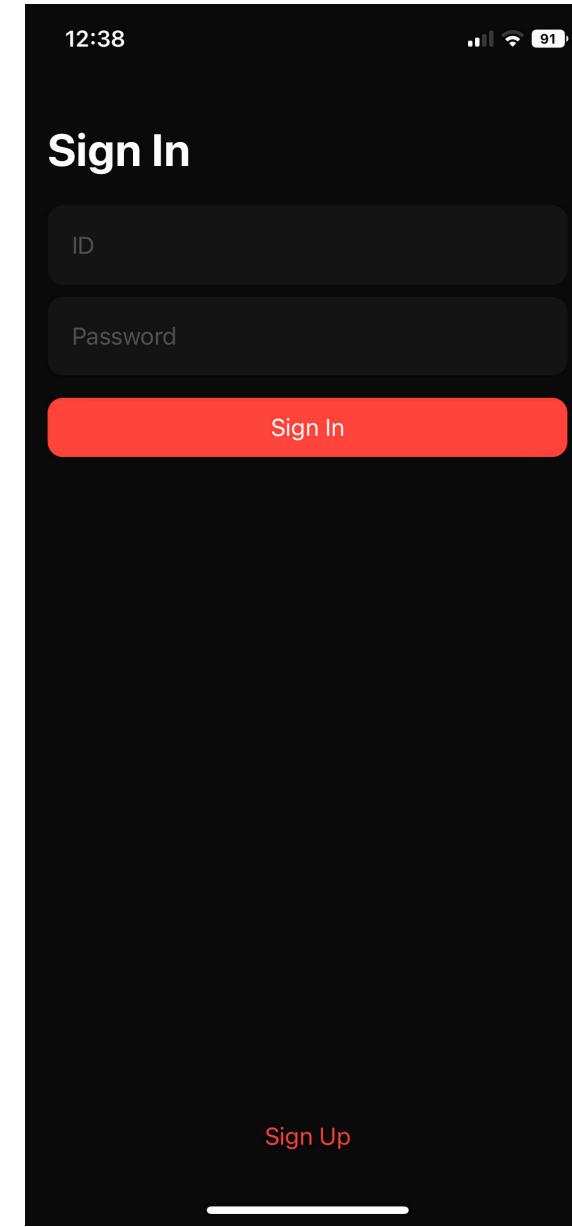
App Icon / Sign In / Sign Up



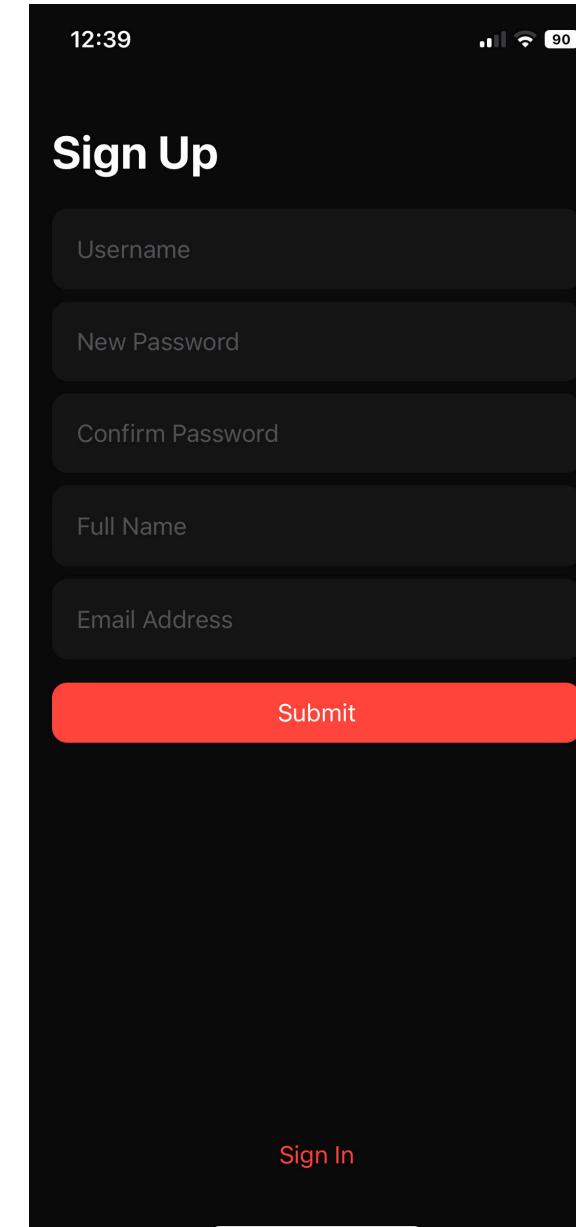
App Icon



Sign In – Light Mode



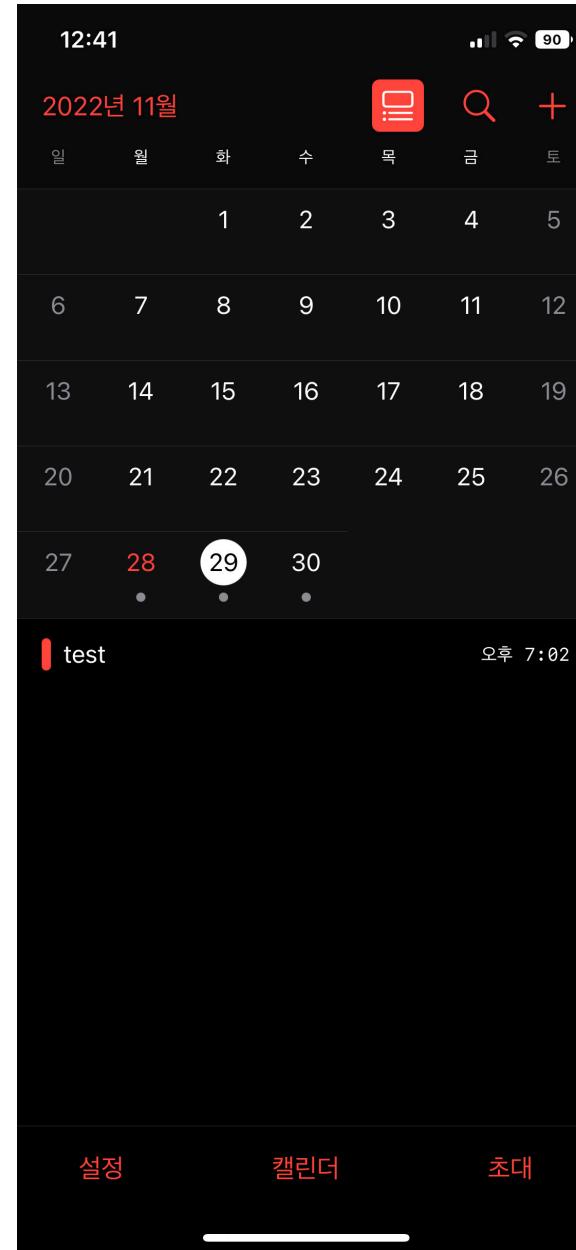
Sign In – Dark Mode



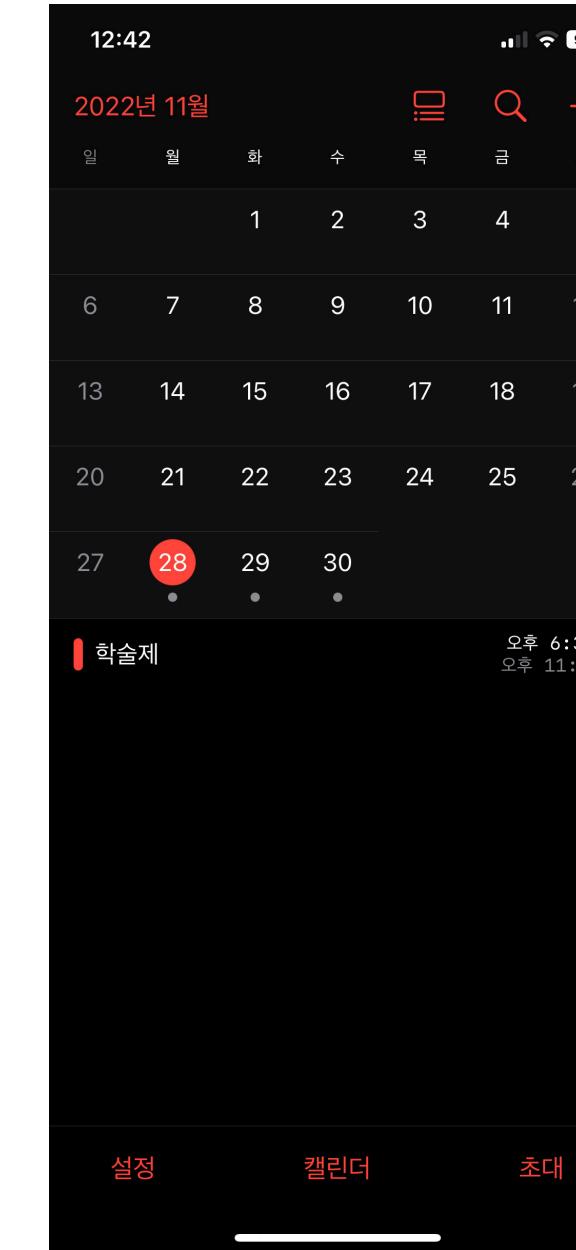
Sign Up

UI / UX 화면

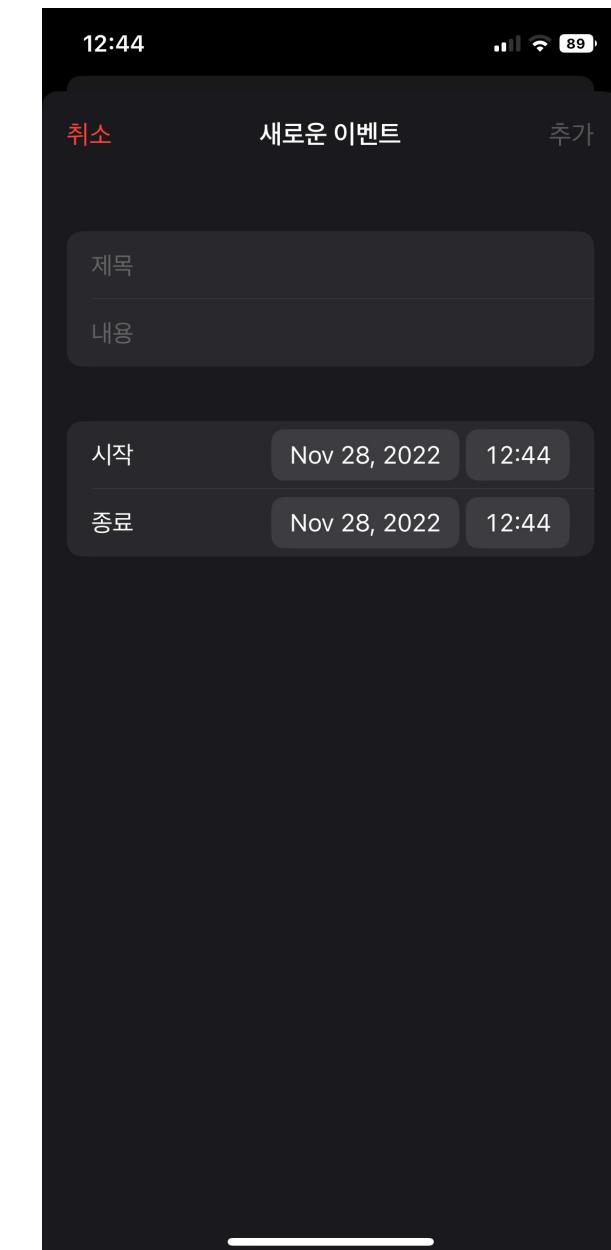
Main – Monthly View



Main – Monthly View 1



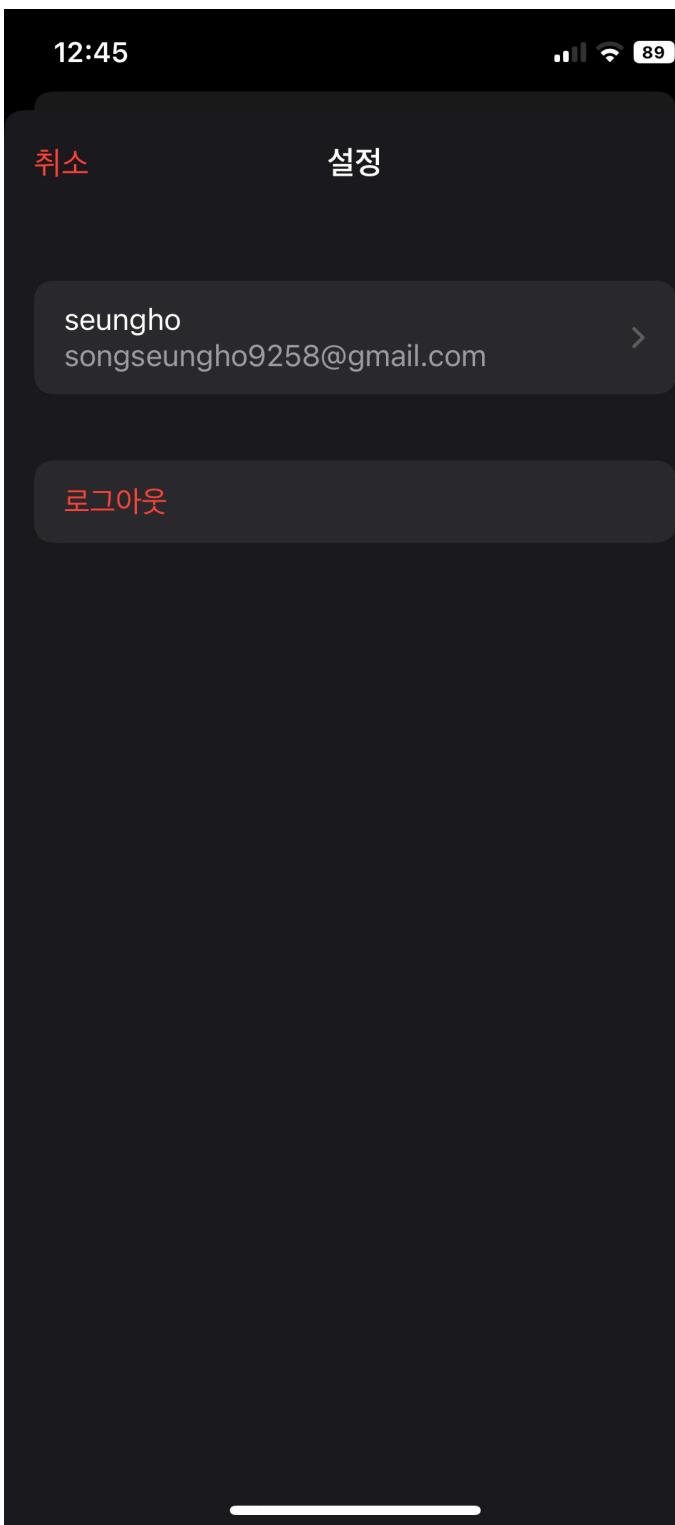
Main – Monthly View 2



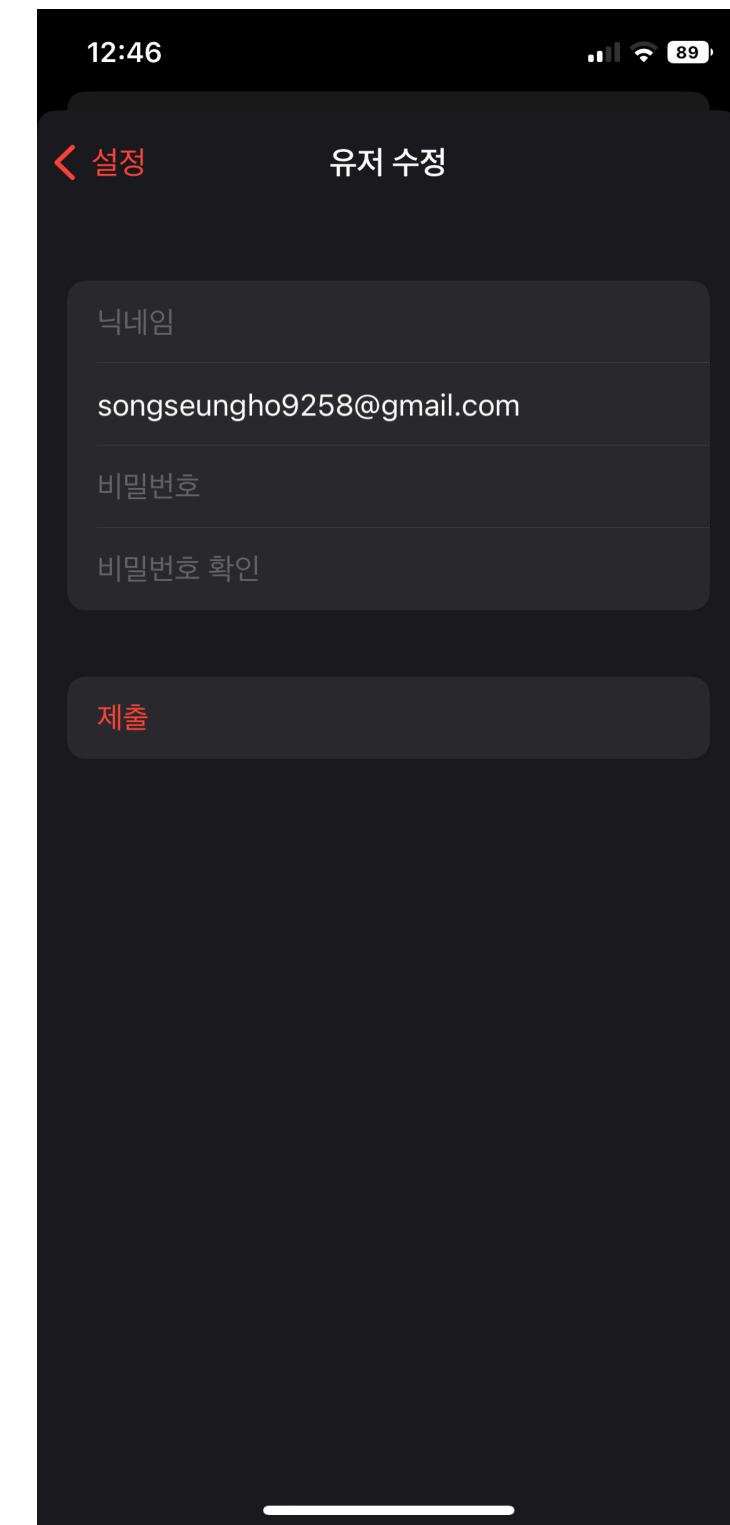
Add Schedule View

UI / UX 화면

User Setting View



Setting View - Main



Setting View – User Info

DataBase

MySQL WorkBench 내 데이터 조회

1 SELECT * FROM user;

Result Grid Filter Rows: Search Edit: Export/Import:

	id	email	nick_name	password
		test@naver.com	NULL	{bcrypt}\$2a\$10\$IGbj2rrE4Dg.W46Rjv6hluo6lIeko1Kc2R6Xb1bkAv2gmZHm.M4q
a	hh		NULL	{bcrypt}\$2a\$10\$tCeyNZ/D7XdS6epSumrbuZX1KTrAkVkwmiPTKH2bsB2wcpPwJCT.
b	q		NULL	{bcrypt}\$2a\$10\$s/vssg/991I6SsB3qH9wSuYrKZ9P7d0ksRsmUaA8idy9yp99p4DSG
cc	fff		NULL	{bcrypt}\$2a\$10\$RxY7kVX3Hwyi7j224r7XhOxgedtmaIPPoVE8l7mqZXclheZETRA6
gggggg	cdd@tyy.com		NULL	{bcrypt}\$2a\$10\$hFeKxEsna4nVYeMDyLxeGeLMOephRvz2/NQ8lh1PTwGiPvJY/6Mhm

User DB 정보 조회

1 SELECT * FROM schedule;

Result Grid Filter Rows: Search Edit: Export/Import:

	schedule_id	content	duration	end_date	id	start_date	title	together_id	user_id
▶	1	git 공부하기	5	2022-11-23 11:11:00	0	2022-11-11 11:11:00	공부	NULL	judy
	2	수영 가기	1	2022-11-21 12:00:00	0	2022-11-21 09:30:00	운동	NULL	judy
	3	친구만나기	1	2022-11-05 20:00:00	0	2022-11-05 17:30:00	약속	NULL	judy
	4	WBS 작성	30	2022-12-05 17:30:00	0	2022-11-05 17:30:00	작업	NULL	judy
	6	hihiasd	5	2022-11-25 11:11:00	0	2022-11-25 11:11:00	Y	NULL	a

Schedule DB 정보 조회

Review

5개월간 진행되었던 긴 여정의 끝.

5개월 동안 스터디 & 프로젝트 제작에 열정적으로 임해 주신

Spring-Boot : 지민, 선진, 민호, 의연 님

iOS : 현수, 태윤, 윤철 님

Special Thanks to : 허원 님

함께할 수 있어 너무 유익한 시간이였습니다. 감사합니다 :)

- 승호 -

QnA

TG Spring-Boot Study Group

Thank you.

TG Spring-Boot Study Group