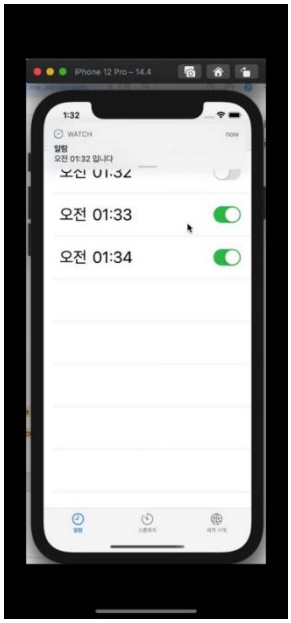
1. 학번: 1871384 이름: 유수미
2. 제목: Watch
3. 앱 개요 및 구조
   - 개요: 원하는 시간을 직접 설정하여 설정된 시간에 알람을 받을 수 있고, 스톱워치 기능과 세계시계 기능이 가능하다.
   - 구조:

1) 알람



2) 스톱워치

3) 세계시계

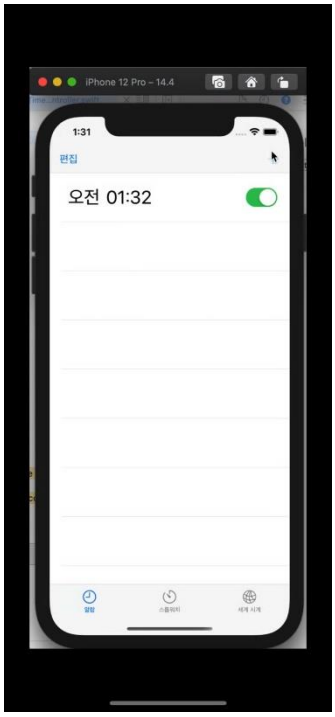
4. 앱 기능

- 알람 화면: 시간을 선택하여 추가하면, 추가된 시간에 알람을 받을 수 있다.
- 스톱워치 화면: 이 화면에서 start 버튼을 누르면 스톱워치가 시작되고, stop 버튼을 눌러 멈출 수 있으며 또, reset 버튼으로 시간을 리셋 할 수 있다. 또, 랩 버튼을 눌러 시간을 기록할 수 있다.
- 세계시계 화면: 현재 시간을 알고싶은 나라를 검색해 목록에 추가하여 그 나라의 현재 시간을 볼 수 있다.

5. 구현 내용

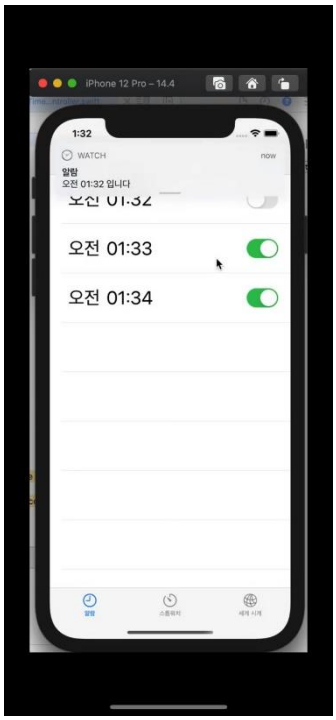
- 알람(FirstViewController -> ViewController)

사용자가 원하는 시간을 설정하면 그 시간에 알람을 받는 기능이다.

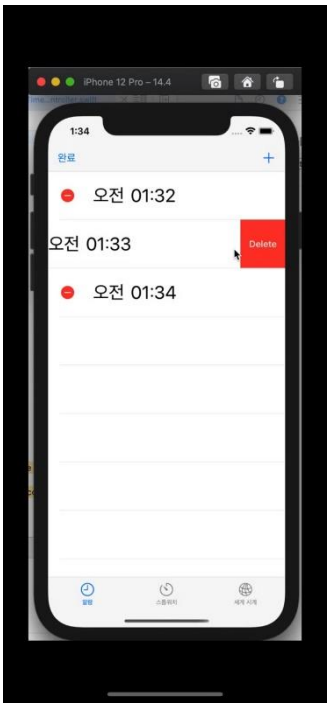


1.알람 첫 화면

2.알람 시간 설정



3.알람 받기(foreground)



4.알람 목록에서 삭제하기

5.알람 받기(background)

첫번째 화면은 FirstViewController이고, BarButtonItem를 이용해 편집과 삽입 버튼을 넣어주었다. 삽입 버튼을 누르면 ViewController로 화면이 전환되고, DatePicker로 시간을 설정할 수 있다. 원하는 시간으로 설정하고 저장버튼을 누르면 알람이 등록되고 다시 FirstViewController로 화면이 전환된다. FirstViewController의 TableView에는 설정한 시간과 알람을 on/off할 수 있는 Switch가 있는 TableViewCell로 채워진다. 편집 버튼을 누르면 TableView가 편집모드로 되어 원하는 TableViewCell을 삭제할 수 있다. Foreground, Background 두 경우 모두 알람을 받을 수 있다.

사용자가 설정한 시간으로 알람을 설정하기 위해 다음과 같이 구현하였다.

```
extension FirstViewController{

    func registerAlarm(identifier:String){

        //현재 시간 구하기(시,분,초)

        let date = Date()

        let calendar = Calendar.current

        let realHour = calendar.component(.hour, from: date)

        let realMinute = calendar.component(.minute, from: date)

        let realSecond = calendar.component(.second, from: date)

        //identifier는 사용자에게 입력받은 시간이다.

        let time = identifier.split(separator: ":").map { val -> Int in return Int(val)!}

        // 현재 시간과 사용자에게 입력받은 시간 차를 계산

        var timeDiffer = Double((time[0] - realHour) * 3600 + (time[1] - realMinute) * 60 + (time[2] - realSecond))

        if timeDiffer < 0{ //현재시간과 설정시간이 24시간 이상 차이가 나면 timeDiffer값이 음수가 나온다. 이 경우엔 위에서 한 계산과는 다르게 86400초(24시간) + timeDiffer로 해줘야 올바른 시간에 알림을 받을 수 있다.

            timeDiffer = 86400 + timeDiffer

        }
```

```
        let content = UNMutableNotificationContent() // 노티피케이션 메세지 객체

        content.sound = UNNotificationSound.default

        content.title = NSString.localizedUserNotificationString(forKey: "알람", arguments: nil)

        if time[0] < 12{ //AM

            if time[0] == 0 {

                content.body = NSString.localizedUserNotificationString(forKey: "오전 " + String(format: "%02d", time[0]+12) + ":" + String(format: "%02d", time[1]) + " 입니다", arguments: nil)

            }else{

                content.body = NSString.localizedUserNotificationString(forKey: "오전 " + String(format: "%02d", time[0]) + ":" + String(format: "%02d", time[1]) + " 입니다", arguments: nil)

            }

        }else{ //PM

            if time[0] > 12 {

                content.body = NSString.localizedUserNotificationString(forKey: "오후 " + String(format: "%02d", time[0]-12) + ":" + String(format: "%02d", time[1]) + " 입니다", arguments: nil)

            }else{

                content.body = NSString.localizedUserNotificationString(forKey: "오후 " + String(format: "%02d", time[0]) + ":" + String(format: "%02d", time[1]) + " 입니다", arguments: nil)

            }

        }

        let trigger = UNTimeIntervalNotificationTrigger(timeInterval: timeDiffer, repeats: false) //얼마 후에 실행할건지

        let request = UNNotificationRequest(

            identifier: identifier,

            content: content,

            trigger: trigger

        )

        center.delegate = self

        center.add(request) { (error:Error?) in}

    }

}
```

위에서처럼 알람을 등록하고, userNotificationCenter()로 알람 처리를 했다.

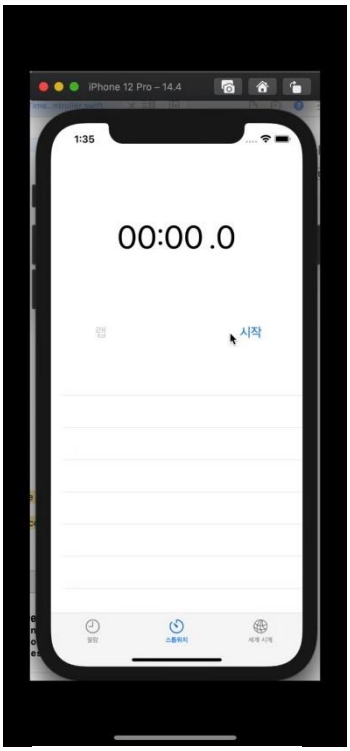
```
//알람 처리

extension FirstViewController:UNUserNotificationCenterDelegate{

    func userNotificationCenter(_ center: UNUserNotificationCenter, willPresent notification: UNNotification, withCompletionHandler completionHandler: @escaping (UNNotificationPresentationOptions) -> Void) {

        completionHandler([.badge, .alert, .sound])

        //현재 시간 구하기

        let date = Date()

        let calendar = Calendar.current

        let realHour = calendar.component(.hour, from: date)

        let realMinute = calendar.component(.minute, from: date)

        let realHourString = String(format: "%02d", realHour)

        let realMinuteString = String(format: "%02d", realMinute)


        //알람의 이름(설정한 시간으로 이름을 정해준다.)

        let ident = realHourString+":"+realMinuteString+":00"


        //알람이 울리면 해당 시간 알람의 UISwitch는 off로 바꾼다.

        var index = -1

        for i in 0..<alarmGroup.count(){

            let selectedCell = alarmTable.cellForRow(at: [0,i]) as! CustomTableViewCell

            if selectedCell.identifier == ident{ //등록했던 알림의 이름(설정한 시간)과 같은 항목의 Switch를 off로 바꾼다.

                for j in 0..<selectedCell.subviews.count{

                    if (selectedCell.subviews[j] as? UISwitch) != nil{

                        let uiSwitch = selectedCell.subviews[j] as! UISwitch

                        uiSwitch.isOn = false

                        index = i

                        break

                    }

                }

            }

        }

}
```

알람 항목의 스위치를 on/off하여 알림을 등록하거나 취소를 하는 기능은 다음과 같이 구현했다.

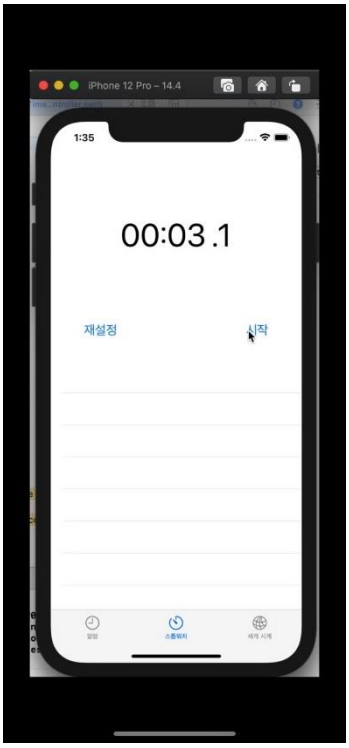
```
//UISwitch를 off하면 알람이 취소되고, on하면 알람이 등록된다.

extension FirstViewController{

    @objc func switchChanged(sender: UISwitch){

        let cell = sender.superview as! UITableViewCell

        let indexPath = alarmTable.indexPath(for: cell)

        let selectedAlarm = alarmGroup.alarms[indexPath!.row]



        //UISwitch off

        if sender.isOn == false{

            selectedAlarm.toggle = false

            //선택한 알람 취소(처음 알림을 등록할 때 설정했던 이름으로 취소해주면 된다.)

            center.removePendingNotificationRequests(withIdentifiers: [selectedAlarm.identifier])

        }else{ //UISwitch on

            selectedAlarm.toggle = true

            //선택한 알람 등록

            registerAlarm(identifier: selectedAlarm.identifier)

        }

    }

}
```

- 스톱워치(StopWatchViewController)

스톱워치 기능으로 시작버튼을 누르면 0부터 1초씩 카운트 된다. 중단 버튼을 누르면 카운트가 멈추고, 다시 시작 버튼을 누르면 멈췄던 부분부터 다시 1초씩 카운트 된다. 카운트 되는 동안 랩 버튼을 누르면 그 시간이 TableView의 TableViewCell로 기록이 된다. 재설정 버튼을 누르면 다시 0초로 리셋된다.

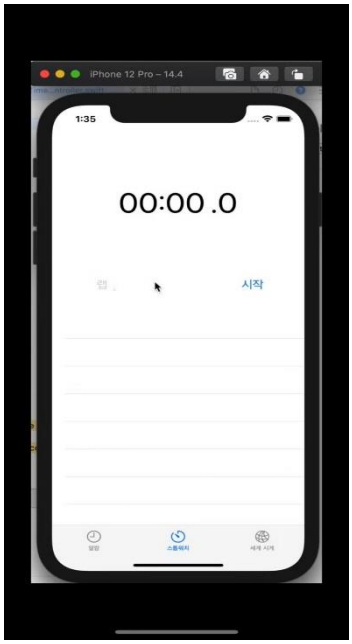


2. 스톱워치 첫 화면



2. 스톱워치 중단

3.스톱워치 시작 & 기록



4. 스톱워치 재설정 후(리셋 후)

0.1초마다 실시간으로 시간을 증가시키기 위해 Timer를 이용했다.

```
extension StopWatchViewController{

    //스톱워치 시작

    func startAction() {

        guard mainTimer == nil else { return }

        mainTimer = Timer.scheduledTimer(withTimeInterval: 0.1, repeats: true, block: {

            (_) in self.timeCount += 1

            DispatchQueue.main.async {

                let timeString = self.makeTimeLabel(count: self.timeCount)

                self.frontTime.text = timeString.0

                self.behindTime.text = ".₩(timeString.1)" } }) }

    func makeTimeLabel(count:Int) -> (String,String) { //return - (TimeLabel, decimalLabel)

        let decimalSec = count % 10

        let sec = (count / 10) % 60

        let min = (count / 10) / 60

        let sec_string = "₩(sec)".count == 1 ? "0₩(sec)" : "₩(sec)"

        let min_string = "₩(min)".count == 1 ? "0₩(min)" : "₩(min)"

        return ("₩(min_string):₩(sec_string)","₩(decimalSec)")

    }

}
```

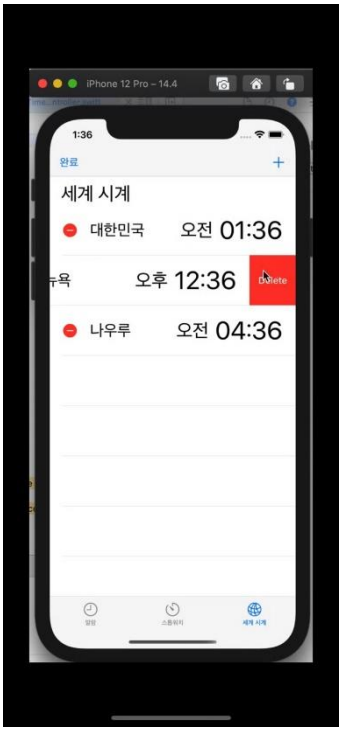
- 세계 시간(WorldTimeViewController -> SearchWorldTimeViewController)

사용자가 나라를 추가해 그 나라의 현재 시각을 알 수 있는 기능이다.

3. 세계시계 첫 화면

2. 세계시계 목록 검색

3. 시계시계 목록



4. 목록 삭제

첫번째 화면은 WorldTimeViewController이고, Bar Button Item을 이용해 상단에 편집버튼과 삽입버튼을 넣어주었다. 삽입버튼을 누르면 SearchWorldTimeViewController로 화면이 전환된다. 이 화면은 438개의 나라이름이 오름차순으로 정렬되어 TableView를 채우고, Search Bar에 입력하는 순간마다 입력된 글자와 일치하는 나라이름만 필터링 되어 다시 TableView를 채운다. 원하는 나라의 TableViewCell을 선택하면 다시 WorldTimeViewController로 화면이 전환되고, WorldTimeViewController의 TableView에 나라이름과 현재시간을 보여주는 TableViewCell로 채워진다. 편집버튼을 누르면 TableView가 편집모드가 되어 원하는 TableViewCell을 삭제할 수 있다. 대한민국의 시간은 TableView에 기본적으로 추가되어있다.

각 나라의 현재시간을 가져오기 위해 TimeZone을 이용했다.

```
//현재 시간 가져오기

extension WorldTimeViewController{

    func getWorldTime(){

        for tz in TimeZone.knownTimeZoneIdentifiers {

            let timeZone = TimeZone(identifier: tz) //해당 나라의 이름과 현재 시각의 정보가 담겨있다.

            var translatedName : String = timeZone?.localizedName(for: NSTimeZone.NameStyle.shortGeneric, locale: Locale(identifier: "ko_KR")) ?? "" //나라의 이름은 기본적으로 영어로 되어있기 때문에 한글로 바꾸어줬다.

            let date = DateFormatter()

            date.locale = Locale(identifier: "ko_KR")

            date.timeZone = timeZone

            date.dateFormat = "HH:mm" //시간을 HH:mm 형태로 포맷팅했다.

            var name =  translatedName.components(separatedBy: " 시간") //나라 이름만 추출했다.

            if name[0] == "Montreal"{name[0] = "몬트리올"} //몬트리올은 한글 변환이 적용이 안되어서 직접 변경해주었다.

            var time = date.string(from: Date())

            var worldTime = WorldTime(name: name[0], time: time)

            pair[name[0]] = time

        }

    }

}
```

검색기능을 위해 다음과 같은 알고리즘으로 구현했다.

```
//searchBar에 입력된 글자와 일치하는 나라의 이름만 보여지도록 필터링
extension SearchWorldTimeViewController:UISearchBarDelegate{
    func searchBar(_ searchBar: UISearchBar, textDidChange searchText: String) {
        //searchBar의 내용이 바뀔때마다 searchBar의 글자와 일치하는 나라이름만 filteredWorldTimes라는 배열로 따로 저장해준다.
        filteredWorldTimes = searchBar.text!.isEmpty ? nameArray : nameArray.filter({(dataString:String)->Bool in
            return dataString.range(of: searchBar.text!,options: .caseInsensitive) != nil
        })
        // filteredWorldTimes의 내용이 변경되었으니 TableView의 내용을 갱신한다.
        worldTimeTableView.reloadData()
    }
}
```

- 화면전환

FirstViewController -> ViewController,

WorldTimeViewController -> SearchWorldTimeViewController 으로 화면전환하는 부분은 Navigation Controller를 이용했고, FirstViewController, StopWatchViewController, WorldTimeViewController 이 세 화면을 왔다갔다 할 수 있게 하는 부분은 Tab Bar Controller 를 이용했다.

6. 결론

개발을 하면서 한학기동안 배웠던 내용만으로도 충분히 그럴듯한 앱을 만들 수 있는 자신을 보고 뿌듯했다. 다만, 데이터베이스를 이용하지 못한 부분이 좀 아쉽다. 또, 평소 어플 개발에 관심이 많았는데 이번 수업으로 되게 많은 걸 얻어간 느낌이다.