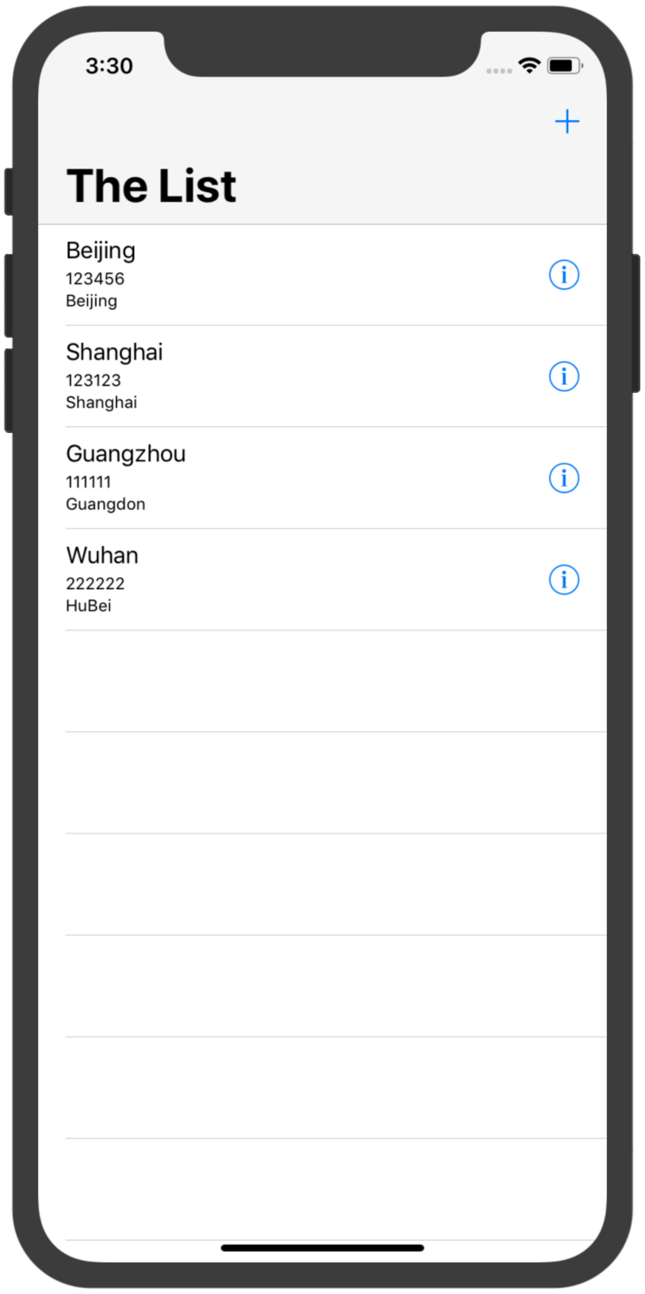
Assignment 3

2016302580149 赵世晗

In this assignment, the course requires us to build up the city model in the database and uses the TableView to show it. Here is the result performance about my program:



3:30

+

The List

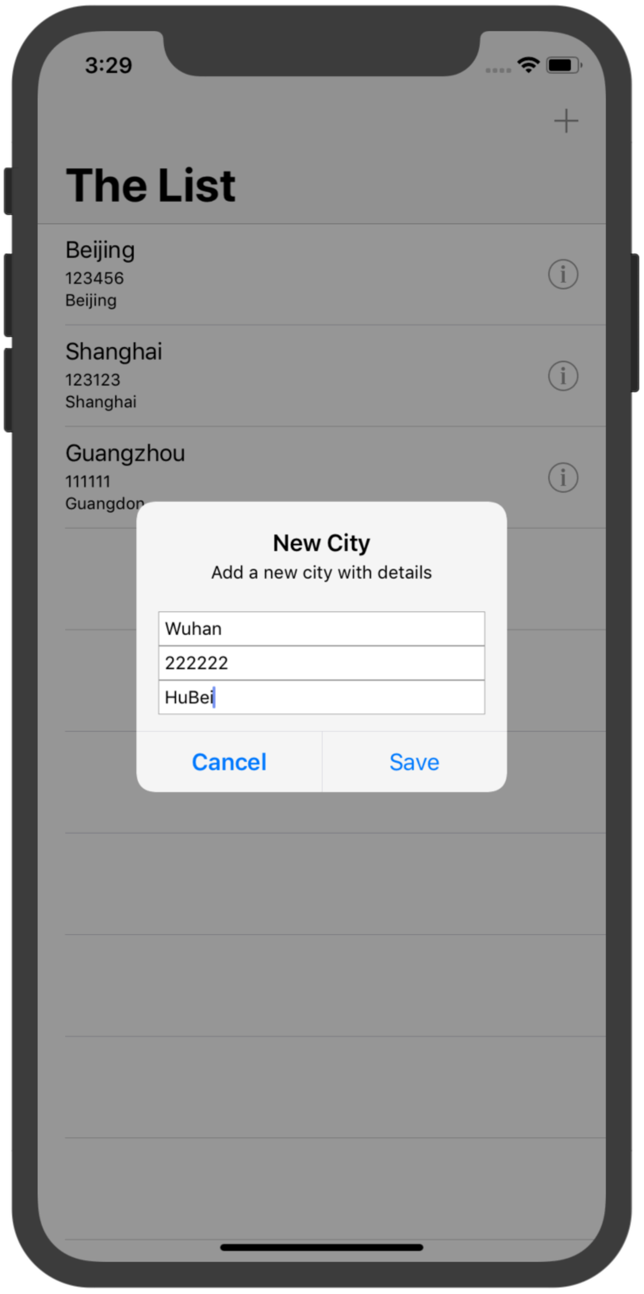
Beijing
123456
Beijing

Shanghai
123123
Shanghai

Guangzhou
111111
Guangdon

Wuhan
222222
HuBei

As for add list to the current table, here is the appearance:


3:29

The List

Beijing
123456
Beijing

Shanghai
123123
Shanghai

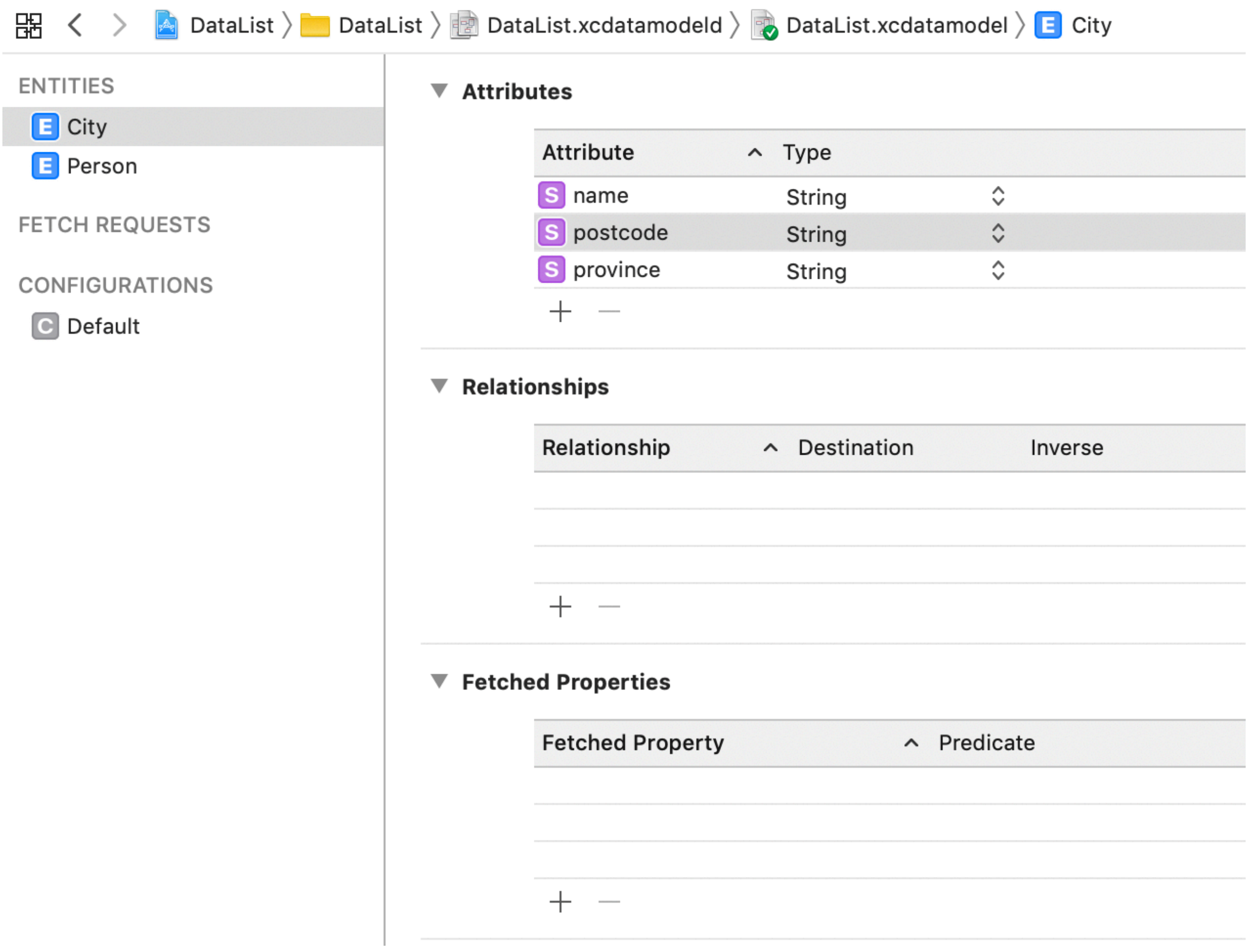
Guangzhou
111111
Guangdon

New City
Add a new city with details

Wuhan
222222
HuBei

The whole program can be divided into several parts. The step one is to build the data entity "city" for the Xcode project to display. Here we import the core data to achieve the persistent data store in the hard device rather than build up a temporary array list variable in the memory. The data modal contains three main property: name, postcode, province, as showing below.

All the data information can be define in the xcdatamodel:



DataList 〉 DataList 〉 DataList.xcdatamodeld 〉 DataList.xcdatamodel 〉 City

ENTITIES
- City
- Person

FETCH REQUESTS

CONFIGURATIONS
- Default

Attributes

Attribute	Type
name	String
postcode	String
province	String

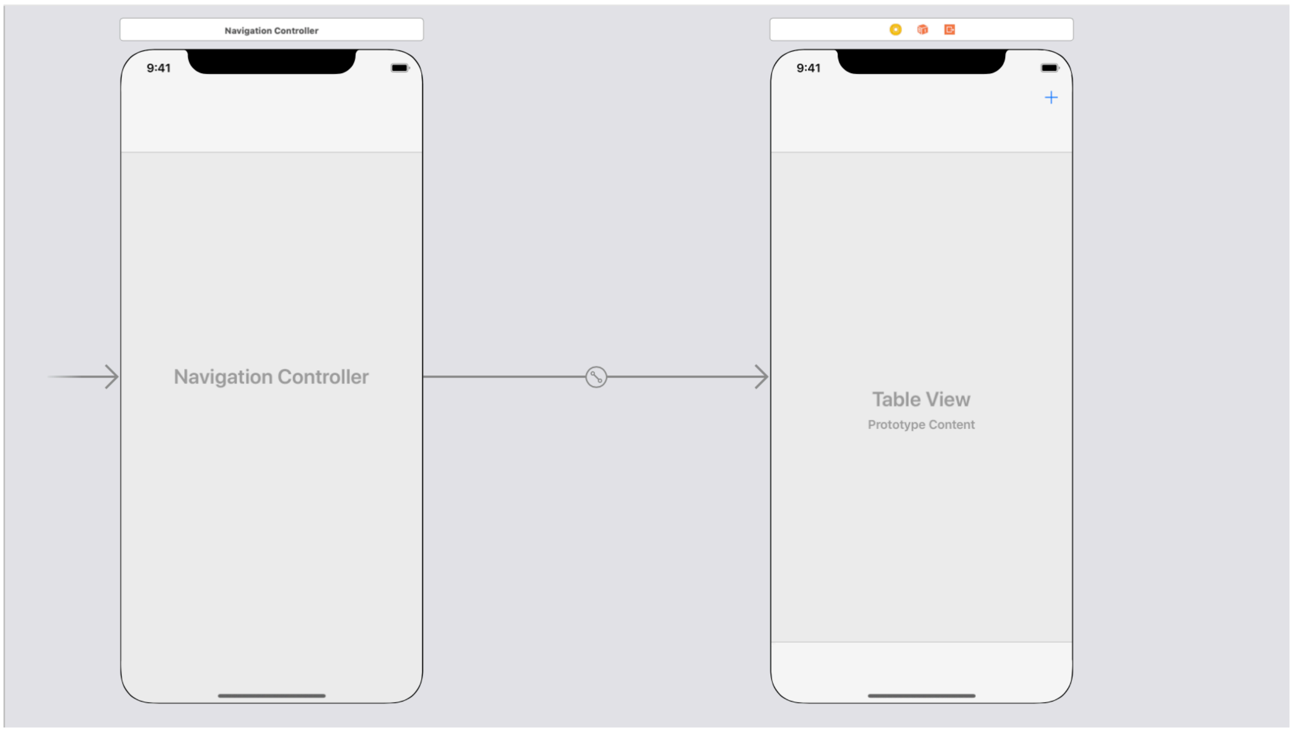
Relationships

Relationship	Destination	Inverse

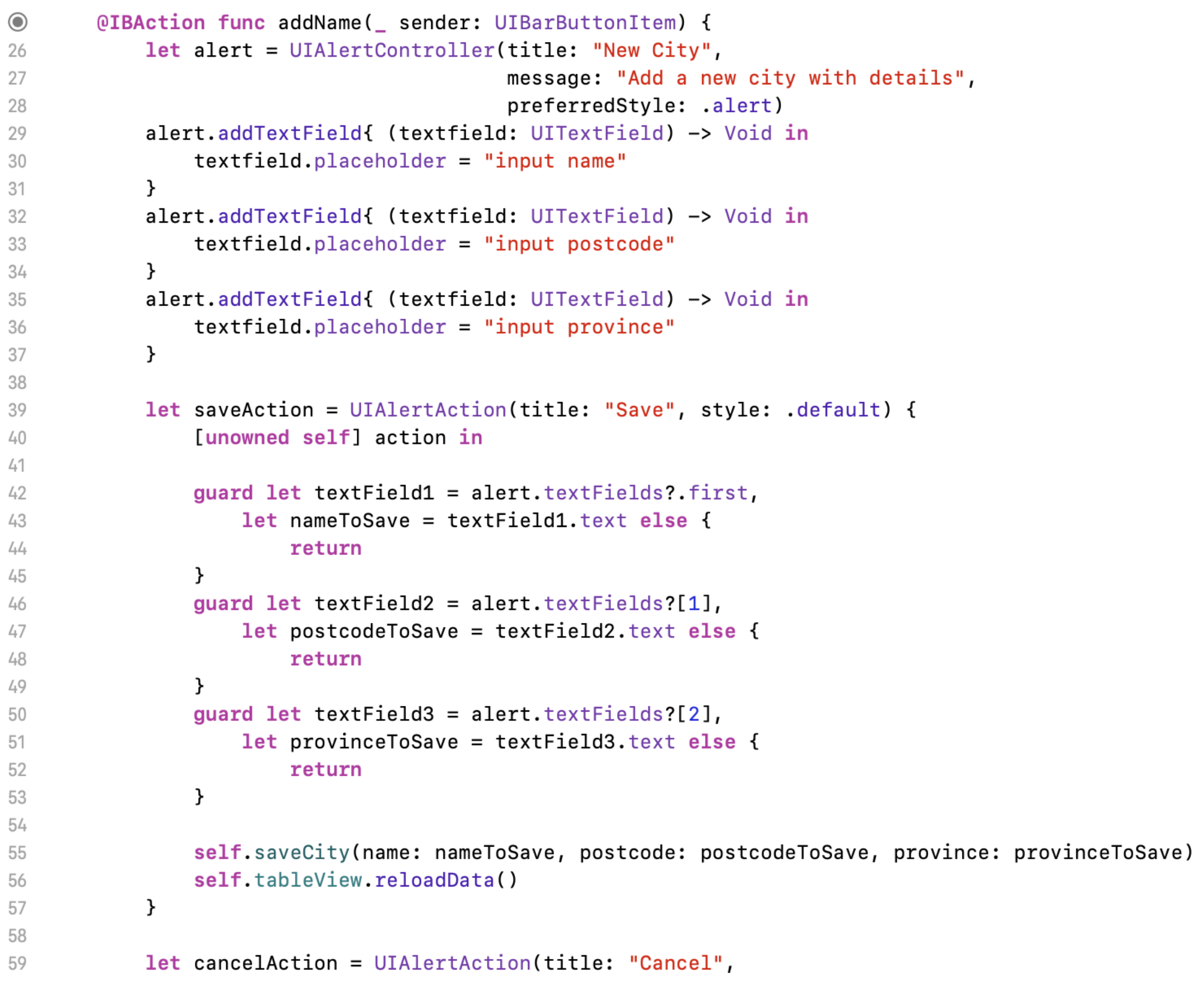
Fetched Properties

Fetched Property	Predicate

On top of that, we need to display this information by TableView. Here we create a TableView controller and use the UITableViewCell to achieve the display. The main storyboard is just like this:



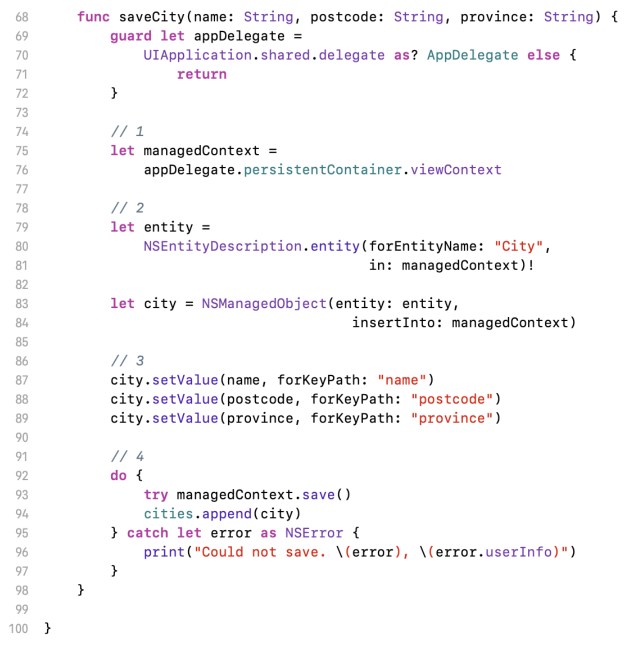
How to add new data item to the current database? Here we use the IBAction to establish the trigger action between bar add button and function in the ViewController.swift, here is the source code to display, we use the addTextField method to the UIAlertController to allow the user input the string value of the data item in the textfield, then we transfer these parameters to the saveCity function.



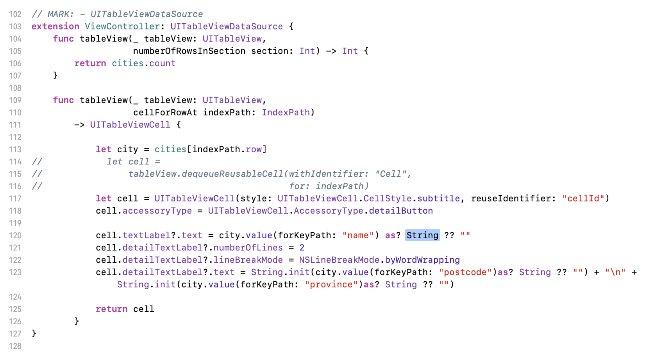
```swift
@IBAction func addName(_ sender: UIBarButtonItem) {
    let alert = UIAlertController(title: "New City",
                                  message: "Add a new city with details",
                                  preferredStyle: .alert)
    alert.addTextField{ (textfield: UITextField) -> Void in
        textfield.placeholder = "input name"
    }
    alert.addTextField{ (textfield: UITextField) -> Void in
        textfield.placeholder = "input postcode"
    }
    alert.addTextField{ (textfield: UITextField) -> Void in
        textfield.placeholder = "input province"
    }

    let saveAction = UIAlertAction(title: "Save", style: .default) {
        [unowned self] action in

        guard let textField1 = alert.textFields?.first,
            let nameToSave = textField1.text else {
                return
        }
        guard let textField2 = alert.textFields?[1],
            let postcodeToSave = textField2.text else {
                return
        }
        guard let textField3 = alert.textFields?[2],
            let provinceToSave = textField3.text else {
                return
        }

        self.saveCity(name: nameToSave, postcode: postcodeToSave, province: provinceToSave)
        self.tableView.reloadData()
    }

    let cancelAction = UIAlertAction(title: "Cancel",
```

We define the NSManagedObject as a global variable that store the data from the xcdatamodeld. When it comes to the save function, we just grasp the object from the ArrayList and use set method to change the value:

How about the final display? We use the UITableCell to display the each row about the data, here I use a little trick (with cell.textLabel and cell.detailTextLabel) that can allow us to display the multiple line information. We extend the ViewController to UITableViewDataSource and return the each data row we edited.



```swift
    func saveCity(name: String, postcode: String, province: String) {
        guard let appDelegate =
            UIApplication.shared.delegate as? AppDelegate else {
                return
        }

        // 1
        let managedContext =
            appDelegate.persistentContainer.viewContext

        // 2
        let entity =
            NSEntityDescription.entity(forEntityName: "City",
                                       in: managedContext)!

        let city = NSManagedObject(entity: entity,
                                   insertInto: managedContext)

        // 3
        city.setValue(name, forKeyPath: "name")
        city.setValue(postcode, forKeyPath: "postcode")
        city.setValue(province, forKeyPath: "province")

        // 4
        do {
            try managedContext.save()
            cities.append(city)
        } catch let error as NSError {
            print("Could not save. \(error), \(error.userInfo)")
        }
    }

}
```



```swift
// MARK: - UITableViewDataSource
extension ViewController: UITableViewDataSource {
    func tableView(_ tableView: UITableView,
                   numberOfRowsInSection section: Int) -> Int {
        return cities.count
    }

    func tableView(_ tableView: UITableView,
                   cellForRowAt indexPath: IndexPath)
        -> UITableViewCell {

            let city = cities[indexPath.row]
//            let cell =
//                tableView.dequeueReusableCell(withIdentifier: "Cell",
//                                              for: indexPath)
            let cell = UITableViewCell(style: UITableViewCell.CellStyle.subtitle, reuseIdentifier: "cellId")
            cell.accessoryType = UITableViewCell.AccessoryType.detailButton

            cell.textLabel?.text = city.value(forKeyPath: "name") as? String ?? ""
            cell.detailTextLabel?.numberOfLines = 2
            cell.detailTextLabel?.lineBreakMode = NSLineBreakMode.byWordWrapping
            cell.detailTextLabel?.text = String.init(city.value(forKeyPath: "postcode")as? String ?? "") + "\n" +
                String.init(city.value(forKeyPath: "province")as? String ?? "")

            return cell
        }
}
```