**Министерство образования и науки Кыргызской Республики**

Кыргызский государственный технический университет

Им. Разакова

САМОСТОЯТЕЛЬНАЯ РАБОТА

По дисциплине «Проектирование интерфейса пользователя»

Тема: Вариант №10 Разработка магазина бытовой техники OOO”TS”

Выполнил студент ПИ 1-16 Абакиров Нурсултан

Содержание

[Введение 3](#_Toc533410963)

[Постановка задачи 3](#_Toc533410964)

[Описание Программного Обеспечения 4](#_Toc533410965)

[Подключение к базе данных SQLite3 4](#_Toc533410966)

[Главное окно 8](#_Toc533410967)

[**Триггеры на нажатия:** 9](#_Toc533410968)

[Окно редактирования товаров 10](#_Toc533410969)

[Окно редактирования клиентов 12](#_Toc533410970)

[Окно редактирования заказов 14](#_Toc533410971)

[Список литературы 17](#_Toc533410972)

# **Введение**

В настоящее время многие организации используют информационные системы. Для работы на такой системе требуются настройки приложения и дополнительные ресурсы, а также умение человека работать с этим программным обеспечением. Используя интерфейс, организованный к базе данных, можно значительно облегчить задачу пользователям информационных систем. Поэтому организация такого интерфейса посредством языков программирования является сегодня актуальной задачей.

# **Постановка задачи**

Рассмотрим организацию интерфейса посредством языка программирования swift для создания программного обеспечения учета заказов, покупателей и товаров в магазине бытовой техники. Для реализации нашей задачи будем использовать программное обеспечение XCode Version 10.1 от Apple. В качестве базы данных будем использовать SQLite3

# **Описание Программного Обеспечения**

## **Подключение к базе данных SQLite3**

1. Скачать проект с сайта https://github.com/stephencelis/SQLite.swift
2. Перетащите файл SQLite.xcodeproj в свой проект.
3. На вкладке «Общие» своей цели нажмите кнопку «+» в разделе «Связанные фреймворки и библиотеки».
4. Выберите соответствующую SQLite.framework для вашей платформы.
5. Добавить

Для удобства работы с базой пишем свой контроллер db.swift

*//*

*// db.swift*

*// ts*

*//*

*// Created by user on 12/23/18.*

*// Copyright © 2018 user. All rights reserved.*

*//*

**import** Foundation

**import** SQLite

**class** Db {

**public** **let** product = Table("product")

**public** **let** customer = Table("customer")

**public** **let** order = Table("order")

**private** **let** id = Expression<Int>("id")

**private** **let** name = Expression<String>("name")

**private** **let** code = Expression<String>("code")

**private** **let** price = Expression<String>("price")

**private** **let** number = Expression<String>("number")

**private** **let** customer\_id = Expression<String>("customer\_id")

**private** **let** product\_id = Expression<String>("product\_id")

**private** **let** quantity = Expression<String>("quantity")

**private** **let** delivered = Expression<String>("delivered")

**private** **let** note = Expression<String>("note")

**public** **var** product\_max\_id: Int = 0

**public** **var** customer\_max\_id: Int = 0

**public** **var** order\_max\_id: Int = 0

**public** **var** db: Connection

**init**(path: String) {

print(path)

**do** {

db = **try** Connection(path)

**try** db.run(product.create (ifNotExists: **true**) { t **in**

t.column(id, primaryKey: **true**)

t.column(name)

t.column(code)

t.column(price)

})

**try** db.run(customer.create (ifNotExists: **true**) { t **in**

t.column(id, primaryKey: **true**)

t.column(name)

t.column(number)

})

**try** db.run(order.create(ifNotExists: **true**) {t **in**

t.column(id, primaryKey: **true**)

t.column(customer\_id)

t.column(product\_id)

t.column(quantity)

t.column(price)

t.column(delivered)

t.column(note)

})

} **catch** {

fatalError("\(error)")

}

}

**func** insertOrder(id: Int, customer\_id: String, product\_id: String, quantity: String, price: String, delivered: String, note: String) {

**let** insert = **self**.order.insert(**self**.id <- id, **self**.customer\_id <- customer\_id, **self**.product\_id <- product\_id, **self**.quantity <- quantity, **self**.price <- price, **self**.delivered <- delivered, **self**.note <- note)

**do** {

**try** **self**.db.run(insert)

} **catch** {

fatalError("\(error)")

}

}

**func** getOrders() -> [[String:String]] {

**var** data: [[String:String]] = []

**var** max\_id: Int = 0

**do** {

**for** row **in** **try** **self**.db.prepare(**self**.order) {

**if** row[**self**.id] > max\_id {

max\_id = row[**self**.id]

}

data.append(

[

"id": String(row[**self**.id]),

"customer": row[**self**.customer\_id],

"product": row[**self**.product\_id],

"quantity": row[**self**.quantity],

"price": row[**self**.price],

"delivered": row[**self**.delivered],

"note": row[**self**.note]

])

}

**self**.order\_max\_id = max\_id

**return** data

} **catch** {

fatalError("\(error)")

}

}

**func** deleteOrder(id: Int) {

**let** item = **self**.order.filter(**self**.id == id)

**do** {

**try** **self**.db.run(item.delete())

} **catch** {

fatalError("\(error)")

}

}

**func** updateOrder(id: Int, customer\_id: String, product\_id: String, quantity: String, price: String, delivered: String, note: String) {

**let** item = **self**.order.filter(**self**.id == id)

**do** {

**try** **self**.db.run(item.update(**self**.product\_id <- product\_id, **self**.customer\_id <- customer\_id, **self**.quantity <- quantity, **self**.price <- price, **self**.delivered <- delivered, **self**.note <- note))

} **catch** {

fatalError("\(error)")

}

}

//////////////////////////

**func** insertCustomer(id: Int, name: String, number: String) {

**let** insert = **self**.customer.insert(**self**.id <- id, **self**.name <- name, **self**.number <- number)

**do** {

**try** **self**.db.run(insert)

} **catch** {

fatalError("\(error)")

}

}

**func** getCustomers() -> [[String:String]] {

**var** data: [[String:String]] = []

**var** max\_id: Int = 0

**do** {

**for** row **in** **try** **self**.db.prepare(**self**.customer) {

**if** row[**self**.id] > max\_id {

max\_id = row[**self**.id]

}

data.append(

[

"id": String(row[**self**.id]),

"name": row[**self**.name],

"number": row[**self**.number]

])

}

**self**.customer\_max\_id = max\_id

**return** data

} **catch** {

fatalError("\(error)")

}

}

**func** deleteCostomer(id: Int) {

**let** item = **self**.customer.filter(**self**.id == id)

**do** {

**try** **self**.db.run(item.delete())

} **catch** {

fatalError("\(error)")

}

}

**func** updateCustomer(id: Int, name: String, number: String) {

**let** item = **self**.customer.filter(**self**.id == id)

**do** {

**try** **self**.db.run(item.update(**self**.name <- name, **self**.number <- number))

} **catch** {

fatalError("\(error)")

}

}

//////////////////////////

**func** insertProduct(id: Int, name: String, code: String, price: String) {

**let** insert = **self**.product.insert(**self**.id <- id, **self**.name <- name, **self**.code <- code, **self**.price <- price)

**do** {

**try** **self**.db.run(insert)

} **catch** {

fatalError("\(error)")

}

}

**func** getProduct(id: Int) -> [String:String]? {

**let** query = **self**.product.filter(**self**.id == id)

**do** {

**let** items = **try** db.prepare(query)

**for** item **in** items {

**return** [

"id": String(item[**self**.id]),

"name": item[**self**.name],

"code": item[**self**.code],

"price": item[**self**.price]

]

}

} **catch** {

fatalError("\(error)")

}

**return** **nil**

}

**func** getProducts() -> [[String:String]] {

**var** data: [[String:String]] = []

**var** max\_id: Int = 0

**do** {

**for** row **in** **try** **self**.db.prepare(**self**.product) {

**if** row[**self**.id] > max\_id {

max\_id = row[**self**.id]

}

data.append(

[

"id": String(row[**self**.id]),

"name": row[**self**.name],

"code": row[**self**.code],

"price": row[**self**.price]

])

}

**self**.product\_max\_id = max\_id

**return** data

} **catch** {

fatalError("\(error)")

}

}

**func** deleteProduct(id: Int) {

**let** product = **self**.product.filter(**self**.id == id)

**do** {

**try** **self**.db.run(product.delete())

} **catch** {

fatalError()

}

}

**func** updateProduct(id: Int, name: String, code: String, price: String) {

**let** product = **self**.product.filter(**self**.id == id)

**do** {

**try** **self**.db.run(product.update(**self**.name <- name, **self**.code <- code, **self**.price <- price))

} **catch** {

fatalError()

}

}

}

**let** documentsDirectory = NSSearchPathForDirectoriesInDomains(.documentDirectory, .userDomainMask, **true**)[0] **as** String

**let** db = Db(path: "\(documentsDirectory)/db.sqlite3")

## **Главное окно**

Создадим новый проект в XCode. В MainViewContoller расположим 4 элемента

- Label

- Button (orders)

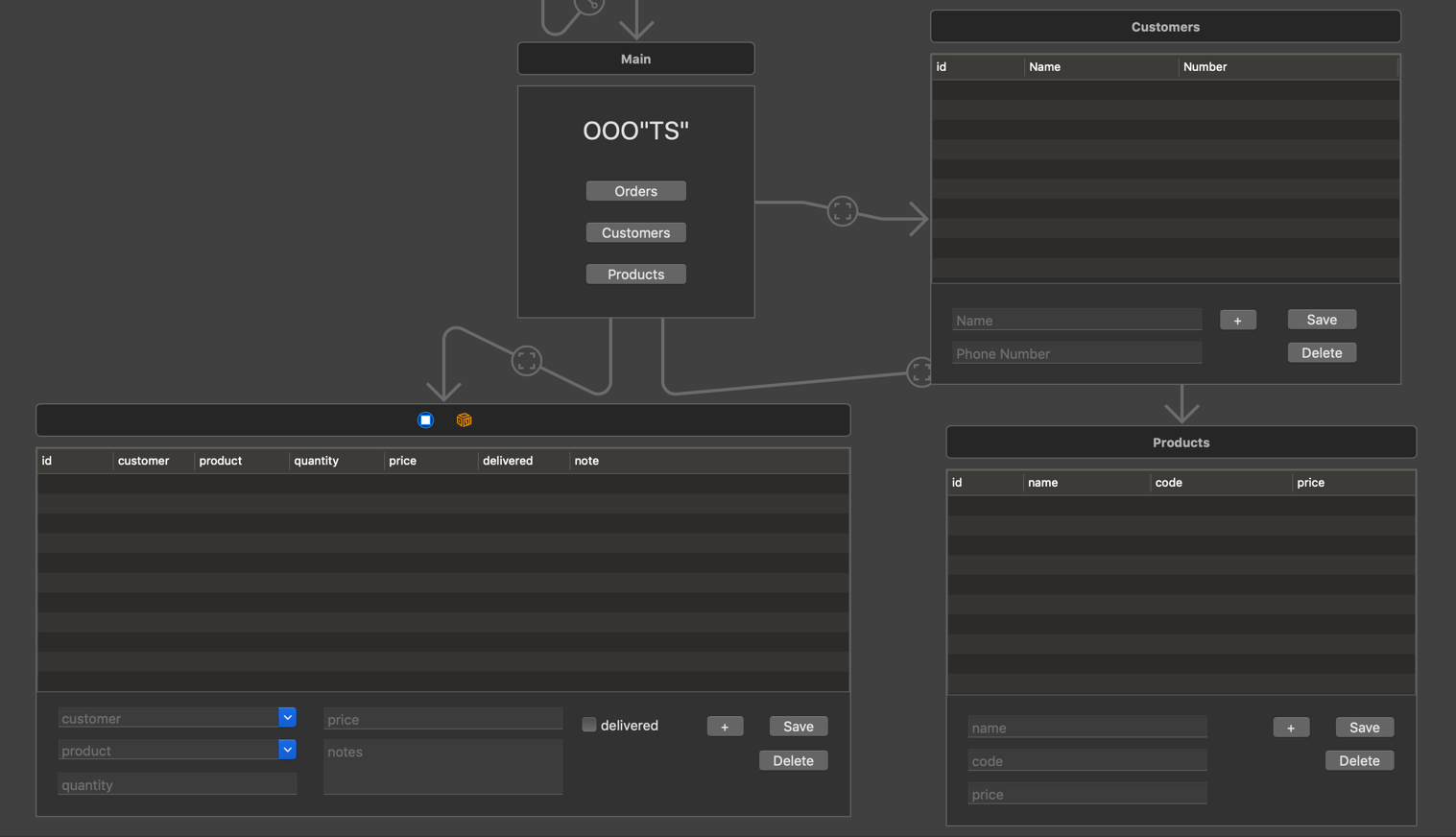
- Button (Customers)

- Button (Products)



### **Триггеры на нажатия:**

* Button (orders) -> OrderViewController
* Button (Customers) -> CustomerViewController
* Button (Products) -> ProductViewController

****

## **Окно редактирования товаров**



*//*

*// ProductViewContoller.swift*

*// ts*

*//*

*// Created by user on 12/23/18.*

*// Copyright © 2018 user. All rights reserved.*

*//*

**import** Cocoa

**class** ProductViewContoller: NSViewController {

**@IBOutlet** **weak** **var** nameInput: NSTextField!

**@IBOutlet** **weak** **var** tableView: NSTableView!

**@IBOutlet** **weak** **var** codeInput: NSTextField!

**@IBOutlet** **weak** **var** priceInput: NSTextField!

**private** **var** rowId: Int? = **nil**

**private** **var** lastId: Int = 1

**private** **var** data: [[String:String]] = []

**@IBAction** **func** onNew(**\_** sender: **Any**) {

**self**.clearInputs()

}

**@IBAction** **func** onSave(**\_** sender: **Any**) {

**if** **self**.rowId != **nil** {

db.updateProduct(id: **self**.rowId!, name: **self**.nameInput.stringValue, code: **self**.codeInput.stringValue, price: **self**.priceInput.stringValue)

} **else** {

db.product\_max\_id += 1

db.insertProduct(id: db.product\_max\_id, name: **self**.nameInput.stringValue, code: **self**.codeInput.stringValue, price: **self**.priceInput.stringValue)

}

**self**.clearInputs()

**self**.tableView.reloadData()

}

**@IBAction** **func** onDelete(**\_** sender: **Any**) {

**if** **self**.rowId != **nil** {

db.deleteProduct(id: **self**.rowId!)

}

**self**.clearInputs()

**self**.tableView.reloadData()

}

**override** **func** viewDidLoad() {

**super**.viewDidLoad()

**self**.tableView.delegate = **self**

**self**.tableView.dataSource = **self**

**self**.tableView.target = **self**

}

**func** clearInputs() {

**self**.rowId = **nil**

**self**.codeInput.stringValue = ""

**self**.nameInput.stringValue = ""

**self**.priceInput.stringValue = ""

}

}

**extension** ProductViewContoller: NSTableViewDataSource, NSTableViewDelegate {

**func** numberOfRows(in tableView: NSTableView) -> Int {

**self**.data = db.getProducts()

**return** **self**.data.count

}

**func** tableView(**\_** customersTableView: NSTableView, objectValueFor tableColumn: NSTableColumn?, row: Int) -> **Any**? {

**let** text = data[row][(tableColumn?.identifier.rawValue)!]

**return** text

}

**func** tableView(**\_** tableView: NSTableView, shouldSelectRow row: Int) -> Bool {

**let** item = **self**.data[row]

**self**.nameInput.stringValue = item["name"]!

**self**.codeInput.stringValue = item["code"]!

**self**.priceInput.stringValue = item["price"]!

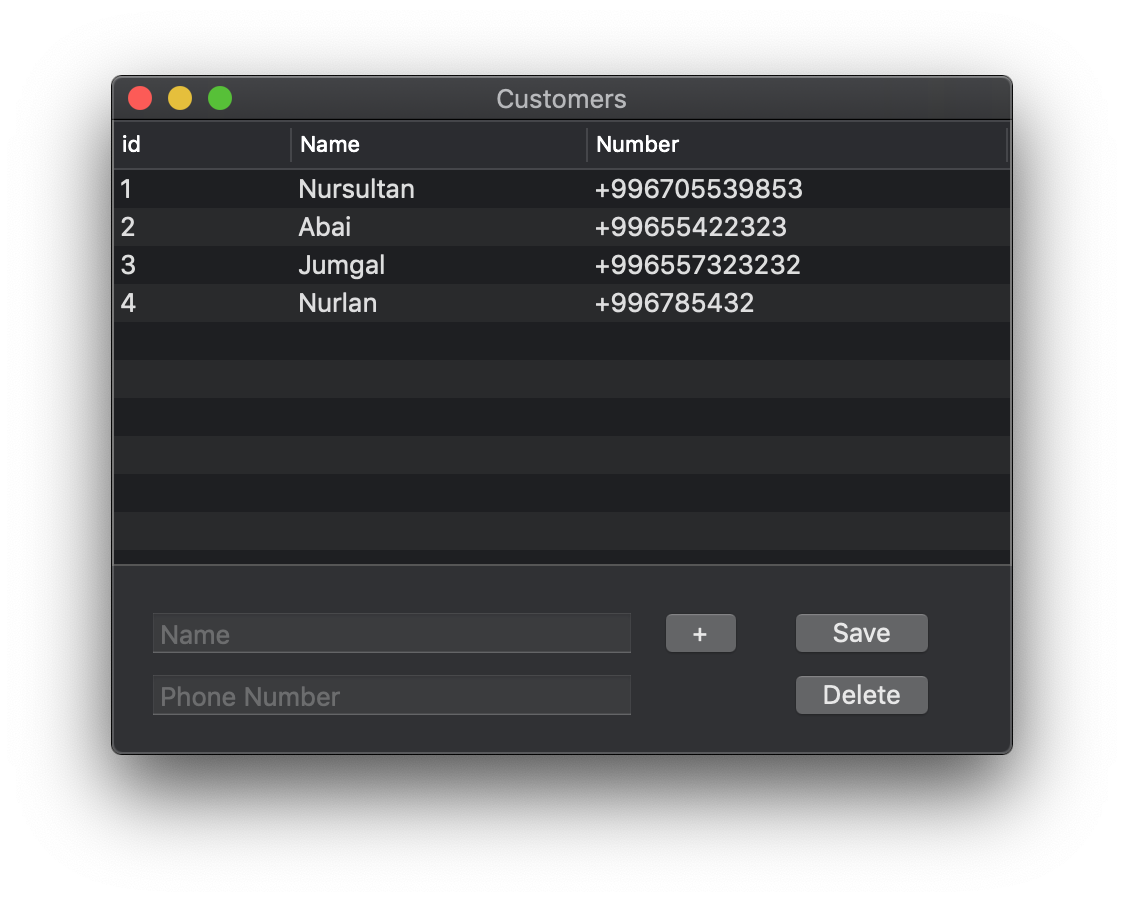
**self**.rowId = Int(item["id"]!)

**return** **true**

}

}

## **Окно редактирования клиентов**



*//*

*// CustomersViewController.swift*

*// ts*

*//*

*// Created by user on 12/22/18.*

*// Copyright © 2018 user. All rights reserved.*

*//*

**import** Cocoa

**class** CustomersViewController: NSViewController {

**@IBOutlet** **weak** **var** customersTableView: NSTableView!

**@IBOutlet** **weak** **var** nameInput: NSTextField!

**@IBOutlet** **weak** **var** numberInput: NSTextField!

**private** **var** rowId: Int? = **nil**

**private** **var** lastId: Int = 1

**private** **var** data: [[String:String]] = []

**override** **func** viewDidLoad() {

**super**.viewDidLoad()

**self**.customersTableView.delegate = **self**

**self**.customersTableView.dataSource = **self**

**self**.customersTableView.target = **self**

*// self.customersTableView.doubleAction*

*// Do view setup here.*

}

**@IBAction** **func** onSave(**\_** sender: NSButton) {

**if** **self**.rowId != **nil** {

**let** id = **self**.rowId!

db.updateCustomer(id: id, name: **self**.nameInput.stringValue, number: **self**.numberInput.stringValue)

} **else** {

db.customer\_max\_id += 1

db.insertCustomer(id: db.customer\_max\_id,name: **self**.nameInput.stringValue, number: **self**.numberInput.stringValue)

}

**self**.clearInputs()

**self**.customersTableView.reloadData()

}

**@IBAction** **func** onDelete(**\_** sender: **Any**) {

**if** **self**.rowId != **nil** {

db.deleteProduct(id: **self**.rowId!)

}

**self**.clearInputs()

**self**.customersTableView.reloadData()

}

**@IBAction** **func** onNew(**\_** sender: **Any**) {

**self**.clearInputs()

}

**func** clearInputs() {

**self**.rowId = **nil**

**self**.numberInput.stringValue = ""

**self**.nameInput.stringValue = ""

}

}

**extension** CustomersViewController: NSTableViewDataSource, NSTableViewDelegate {

**func** numberOfRows(in tableView: NSTableView) -> Int {

**self**.data = db.getCustomers()

**return** **self**.data.count

}

**func** tableView(**\_** customersTableView: NSTableView, objectValueFor tableColumn: NSTableColumn?, row: Int) -> **Any**? {

**let** text = data[row][(tableColumn?.identifier.rawValue)!]

**return** text

}

**func** tableView(**\_** tableView: NSTableView, shouldSelectRow row: Int) -> Bool {

**let** item = **self**.data[row]

**self**.nameInput.stringValue = item["name"]!

**self**.numberInput.stringValue = item["number"]!

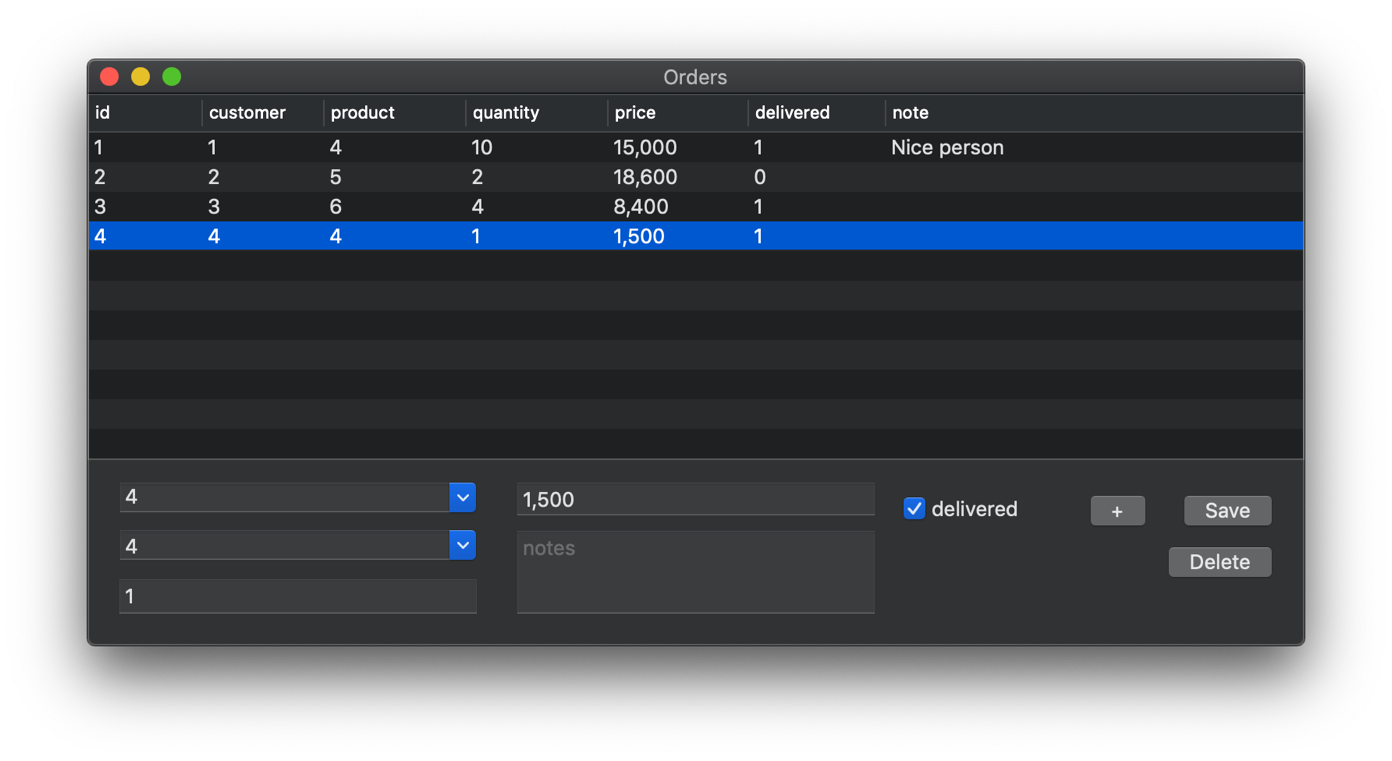
**self**.rowId = Int(item["id"]!)

**return** **true**

}

}

## **Окно редактирования заказов**



*//*

*// OrderViewController.swift*

*// ts*

*//*

*// Created by user on 12/23/18.*

*// Copyright © 2018 user. All rights reserved.*

*//*

**import** Cocoa

**class** OrderViewController: NSViewController {

**@IBOutlet** **weak** **var** tableView: NSTableView!

**@IBOutlet** **weak** **var** customerInput: NSComboBox!

**@IBOutlet** **weak** **var** productInput: NSComboBox!

**@IBOutlet** **weak** **var** quantityInput: NSTextField!

**@IBOutlet** **weak** **var** priceInput: NSTextField!

**@IBOutlet** **weak** **var** noteInput: NSTextField!

**@IBOutlet** **weak** **var** deliveredInput: NSButton!

**@IBOutlet** **weak** **var** downTest: NSComboBox!

**@IBAction** **func** qunatitySelected(**\_** sender: **Any**) {

**let** productId = productInput.stringValue

**if** productId.isEmpty{

**return**

}

**let** product = db.getProduct(id: Int(productId)!)

**self**.priceInput.doubleValue = **self**.quantityInput.doubleValue \* Double(product!["price"]!)!

}

**private** **var** rowId: Int? = **nil**

**private** **var** lastId: Int = 1

**private** **var** data: [[String:String]] = []

**@IBAction** **func** onNew(**\_** sender: **Any**) {

**self**.clearInputs()

}

**@IBAction** **func** onSave(**\_** sender: **Any**) {

**if** **self**.rowId != **nil** {

**let** id = **self**.rowId!

db.updateOrder(id: id, customer\_id: **self**.customerInput.stringValue, product\_id: **self**.productInput.stringValue, quantity: **self**.quantityInput.stringValue, price: **self**.priceInput.stringValue, delivered: **self**.deliveredInput.stringValue, note: **self**.noteInput.stringValue)

} **else** {

**let** id = db.order\_max\_id + 1

db.insertOrder(id: id, customer\_id: **self**.customerInput.stringValue, product\_id: **self**.productInput.stringValue, quantity: **self**.quantityInput.stringValue, price: **self**.priceInput.stringValue, delivered: **self**.deliveredInput.stringValue, note: **self**.noteInput.stringValue)

}

**self**.clearInputs()

**self**.tableView.reloadData()

}

**@IBAction** **func** onDelete(**\_** sender: **Any**) {

**if** **self**.rowId != **nil** {

db.deleteOrder(id: **self**.rowId!)

}

**self**.clearInputs()

**self**.tableView.reloadData()

}

**override** **func** viewDidLoad() {

**super**.viewDidLoad()

**self**.tableView.delegate = **self**

**self**.tableView.dataSource = **self**

**self**.tableView.target = **self**

**for** item **in** db.getCustomers() {

**self**.customerInput.addItem(withObjectValue: item["id"]!)

}

**for** item **in** db.getProducts() {

**self**.productInput.addItem(withObjectValue: item["id"]!)

}

}

**func** clearInputs() {

**self**.rowId = **nil**

**self**.customerInput.stringValue = ""

**self**.productInput.stringValue = ""

**self**.quantityInput.stringValue = ""

**self**.priceInput.stringValue = ""

**self**.noteInput.stringValue = ""

**self**.deliveredInput.stringValue = "0"

}

}

**extension** OrderViewController: NSTableViewDataSource, NSTableViewDelegate {

**func** numberOfRows(in tableView: NSTableView) -> Int {

**self**.data = db.getOrders()

**return** **self**.data.count

}

**func** tableView(**\_** customersTableView: NSTableView, objectValueFor tableColumn: NSTableColumn?, row: Int) -> **Any**? {

**let** text = data[row][(tableColumn?.identifier.rawValue)!]

**return** text

}

**func** tableView(**\_** tableView: NSTableView, shouldSelectRow row: Int) -> Bool {

**let** item = **self**.data[row]

**self**.customerInput.stringValue = item["customer"]!

**self**.productInput.stringValue = item["product"]!

**self**.quantityInput.stringValue = item["quantity"]!

**self**.priceInput.stringValue = item["price"]!

**self**.noteInput.stringValue = item["note"]!

**self**.deliveredInput.stringValue = item["delivered"]!

**self**.rowId = Int(item["id"]!)

**return** **true**

}

}

# **Список литературы**

<https://github.com/stephencelis/SQLite.swift>

<https://www.raywenderlich.com/830-macos-nstableview-tutorial>

<http://www.knowstack.com/swift-3-1-nstableview-complete-guide/>