



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Лабораторная работа № 1

По предмету: «Проектирование программного обеспечения»

Тема: Программная реализация доступа к данным

Студент: Тимонин Антон

Группа ИУ7-626

Оценка (баллы) _____

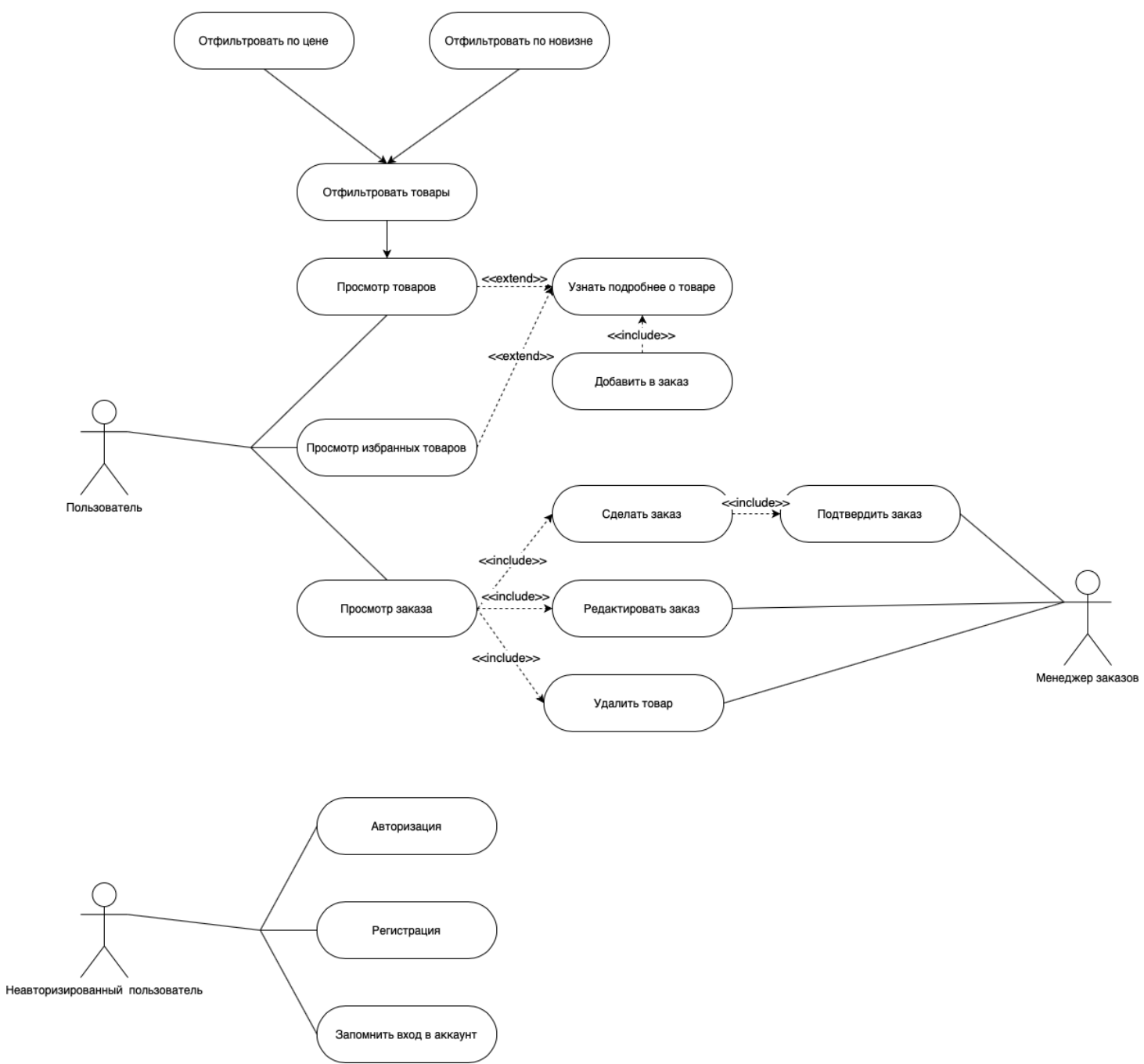
Преподаватель Бекасов Д. Е.

Москва.
2020 г.

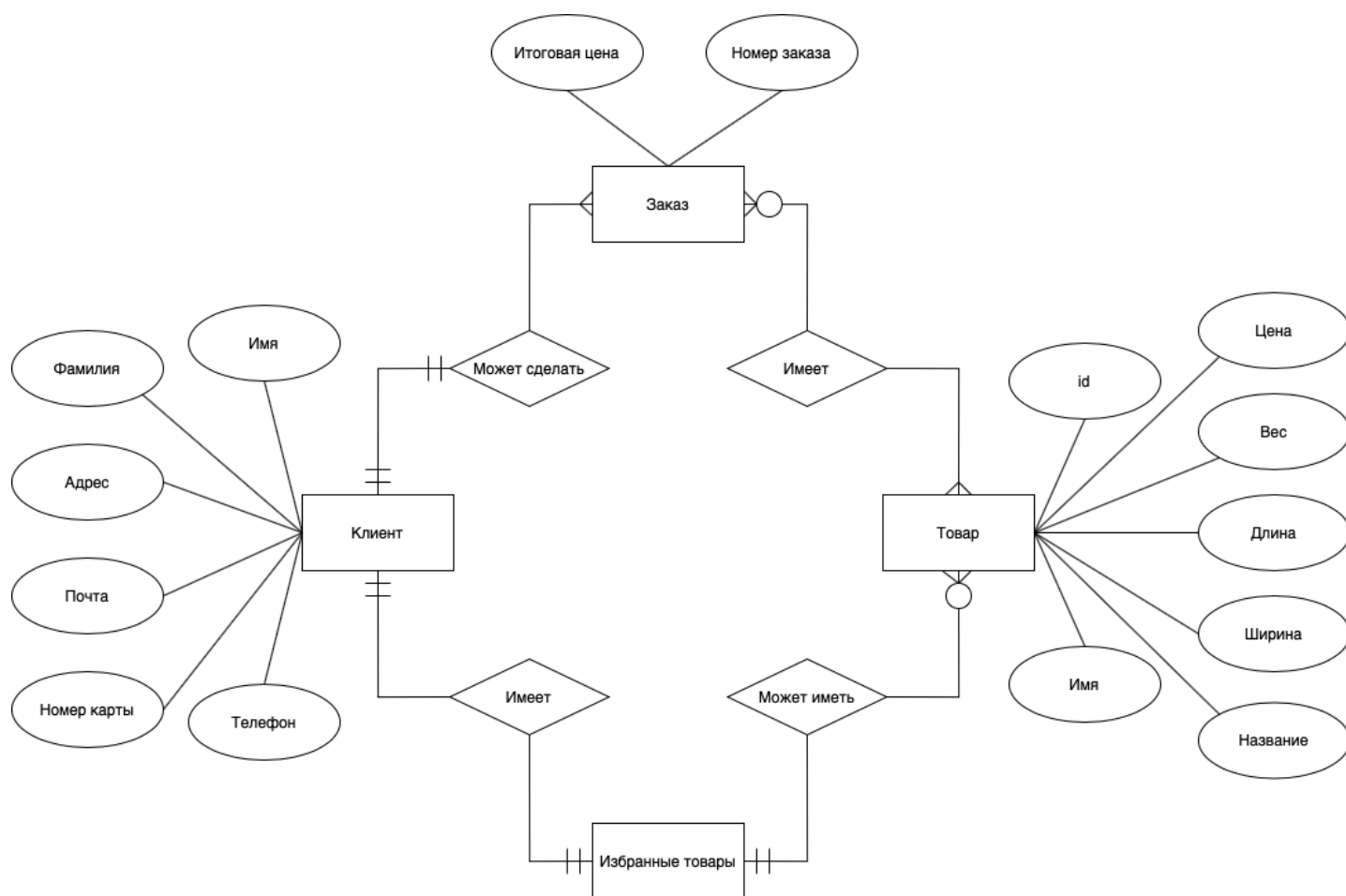
Оглавление

Use - Case диаграмма	3
ER - диаграмма сущностей	4
Технологический стек	5
Структурная схема программы	6
UML диаграмма классов	7
Программная реализация компонента доступа к данным	8

Use - Case диаграмма



ER - диаграмма сущностей



Технологический стек

Используемые технологии и подходы

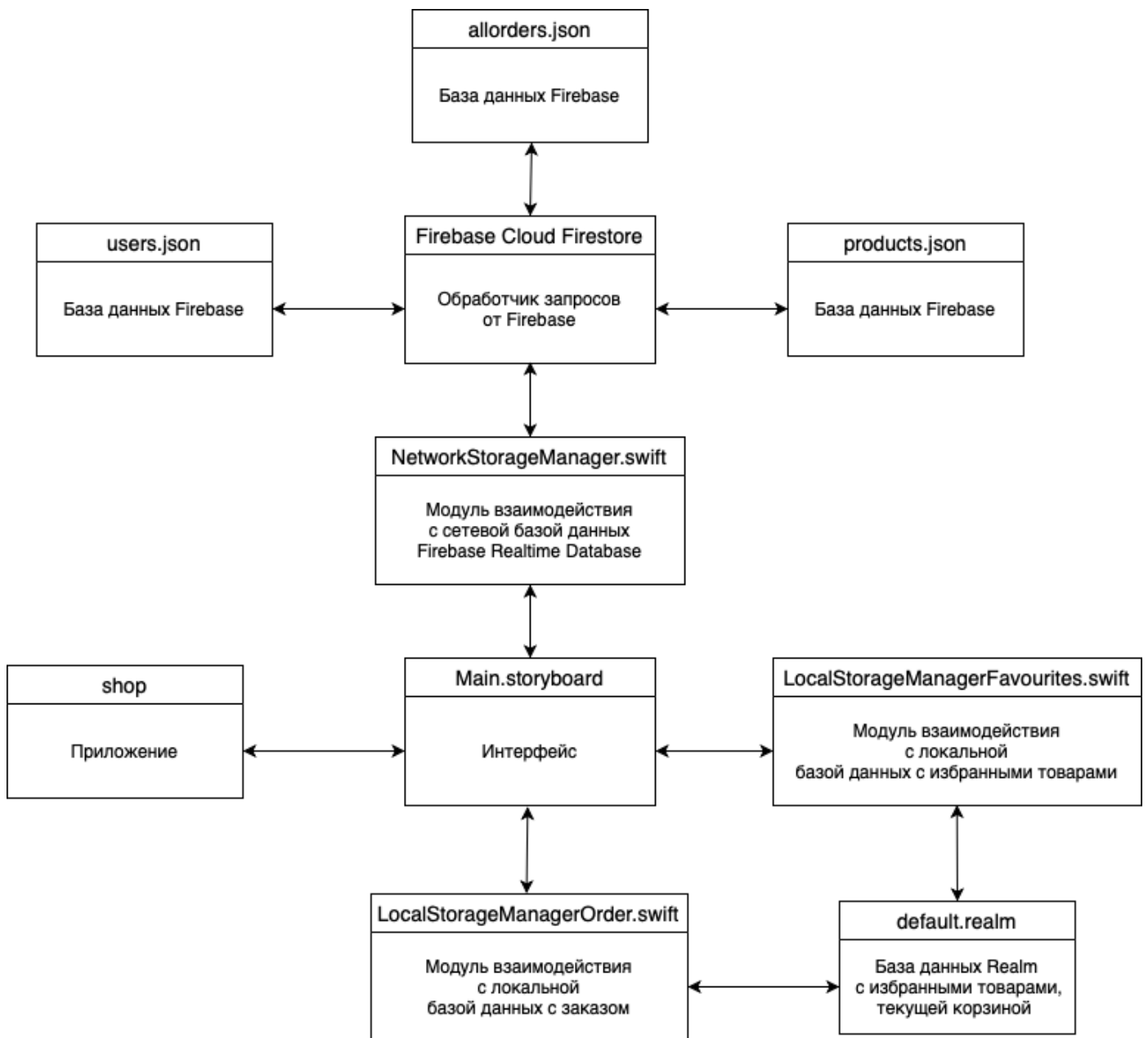
Бэкенд

- Firebase (auth, realtime database) - нереляционная бд
- Realtime - нереляционная бд
- Kingfisher - библиотеке Swift для загрузки и кэширования изображений из интернета
- MVC - архитектура
- Github - система контроля версий

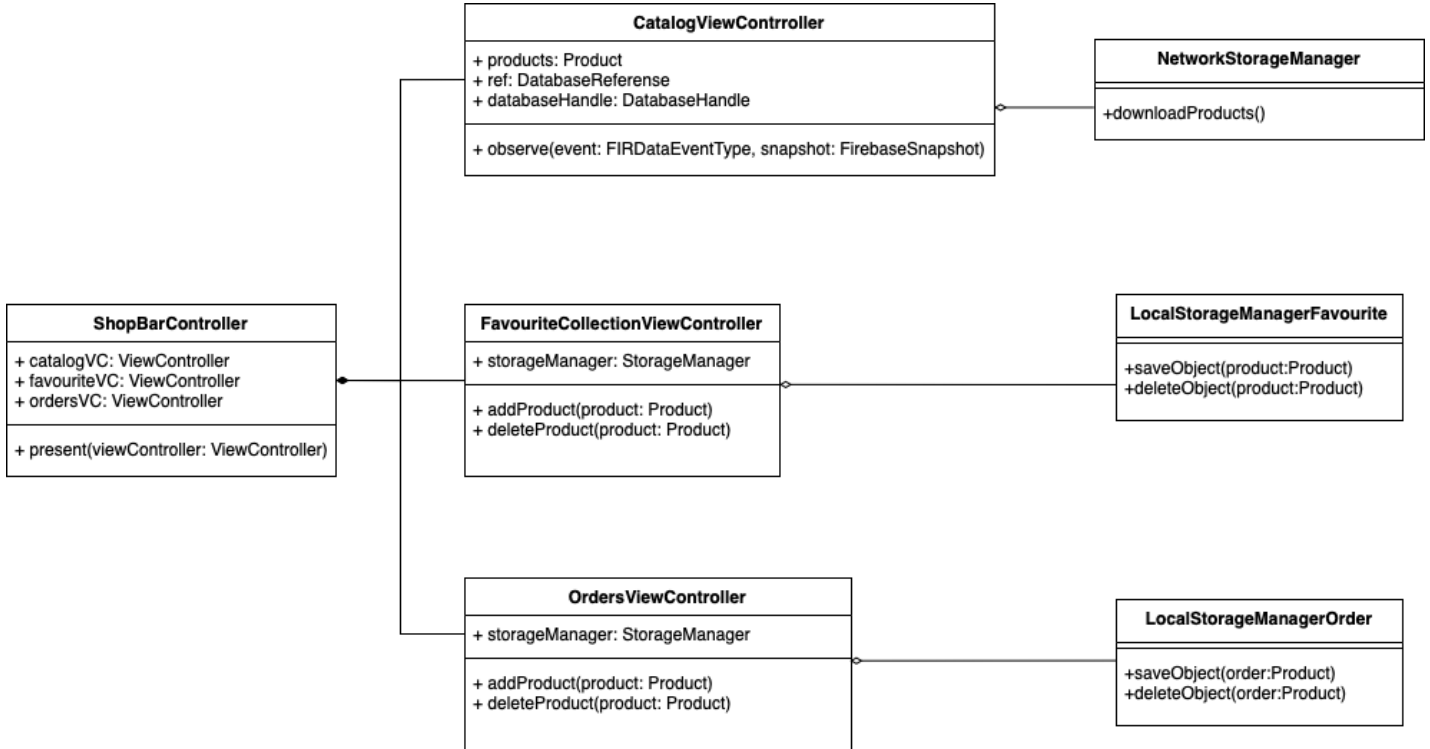
Фронтэнд

- UIKit - фреймворк для построения пользовательского интерфейса

Структурная схема программы



UML диаграмма классов



Программная реализация компонента доступа к данным

Листинг 1. Класс для взаимодействия с локальной базой данных Realm

```
import RealmSwift

let realm = try! Realm()

class StorageManager {

    static func saveObject(_ product: Product) {
        try! realm.write {
            realm.add(product)
        }
    }

    static func deleteObject(_ product: Product) {
        try! realm.write {
            realm.delete(product)
        }
    }
}
```

Листинг 2. CatalogViewController - класс с экраном каталога товаров, лежащих в Firebase. Первоначальная загрузка экрана с выгрузкой товаров их сетевой бд.

```
class CatalogViewController: UICollectionViewController {
    private let reuseIdentifier = "showSubject"
    var products = [Product]()

    var ref:DatabaseReference?
    var databaseHandle: DatabaseHandle?

    override func viewDidLoad() {
        super.viewDidLoad()

        ref = Database.database().reference()
        self.databaseHandle = self.ref?.child("subject").observe(.value,
with: {[weak self] (snapshot) in
            let subjects = snapshot.value as? [[String:Any]]
            let subjectCount = subjects?.count

            for i in 0..
```



```

                                imageURLString: imageURL!)
        self?.products.append(product)
        self?.collectionView.reloadData()
    })
}
}
}

```

Листинг 3. Структура данных, в которой хранятся все товары

```

struct Product {
    var id: Int = 0
    var price: Int = 0
    var weight: Int = 0
    var length: Int? = 0
    var width: Int? = 0
    var name: String = ""
    var imageURLString: String?

    convenience init(id: Int,
                    price: Int,
                    weight: Int,
                    length: Int,
                    width: Int,
                    name: String,
                    imageURLString: String) {

        self.init()
        self.id = id
        self.price = price
        self.weight = weight
        self.length = length
        self.width = width
        self.name = name
        self.imageURLString = imageURLString
    }
}

```