МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**
Кафедра математической кибернетики и компьютерных наук

РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ ДЛЯ СБОРА, ОБРАБОТКИ И ВИЗУАЛИЗАЦИИ ИНФОРМАЦИИ ИЗ ЧЕКОВ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 451 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Тихонова Владислава Денисовича

Научный руководитель
доцент, к. ф.-м. н. _____ А. С. Иванова

Заведующий кафедрой
к. ф.-м. н., доцент _____ С. В. Миронов

Саратов 2023

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Магазины — неотъемлемая часть нашей жизни. Мы покупаем в них самые разные вещи, начиная от продуктов и заканчивая бытовой техникой. Однако, увлекшись покупками, очень просто потратить больше, чем ожидалось. Данное мобильное приложение поможет держать траты в магазинах под контролем.

Целью данной работы является разработка мобильного приложения для сбора, обработки и визуализации информации из чеков под операционную систему iOS на языке программирования Swift, а именно, количество, цена и наименования продуктов, название и адрес магазина, сколько было оплачено картой и наличными, какое количество денег ушло на уплату НДС 10% и 20%.

В результате написания работы должны быть решены следующие задачи:

- изучение работы REST API на языке Swift;
- разработка механизма получения данных с nalog.ru;
- разработка механизма считывания QR-кода;
- планирование и создание базы данных;
- реализация взаимодействия с базой данных;
- визуализация данных;

1 Основные технологии, используемые при разработке приложения

1.1 Язык программирования Swift

Swift - открытый, компилируемый язык программирования общего назначения, разработанный Apple и выпущенный в 2014 году. Он был создан в качестве замены для предыдущего языка программирования Apple Objective -C по причине того, что Objective -C устарел.

1.1.1 Особенности языка Swift

— **Скорость.** Swift был создан с заделом на скорость. Разработчики из Apple говорят, что Swift в 2,6 раза быстрее, чем Objective -C, и в 8,4 раза быстрее, чем Python.
— **Open source сообщество.** У языка Swift существует большое сообщество и немалое количество сторонних инструментов.
— **Безопасность.** Синтаксис языка Swift побуждает писать чистый и последовательный код, обеспечивает защиту от ошибок и улучшает читаемость.

1.1.2 Преимущества использования Swift для разработки

— Улучшенная безопасность и производительность.
— Быстрый процесс разработки.
— Совместимость с Objective -C.
— Легче планировать приложение наперед.
— Уменьшенное использование памяти.
— Автоматическое управление оперативной памятью.
— Full stack потенциал и поддержка разных устройств.
— Крупное сообщество.

1.1.3 Недостатки использования Swift для разработки

Однако, Swift не является идеальным языком программирования и имеет свои недостатки:

— Отсутствие поддержки ранних версий iOS
— Плохая совместимость со сторонними инструментами и IDE.
— Сравнительная новизна языка.

```swift
let vegetable = "red pepper"
switch vegetable {
case "celery":
    let vegetableComment = "Add some raisins and
        make ants on a log."
case "cucumber", "watercress":
    let vegetableComment = "That would make a good
        tea sandwich."
case let x where x.hasSuffix("pepper"):
    let vegetableComment = "Is it a spicy \(x)?"
default:
    let vegetableComment = "Everything tastes good
        in soup."
}
```

Рисунок 1 – Пример кода на языке Swift

2 Описание приложения

Данное приложение состоит из двух экранов добавления чека в базу данных — с помощью фотографии QR кода, напечатанного на чеке или вручную, пяти экранов, визуализирующих различную информацию о чеках, экрана с информацией о разработчиках , а также промежуточных экранов меню.

Для разработки данного приложения было использовано изображение из общедоступных источников.

2.1 Дополнительно написанные классы

Для удобства разработки приложения были написаны несколько классов.

2.1.1 GetReceiptInfo

Класс GetReceiptInfo используется для авторизации и отправки запроса к API для получения информации из чека по QR коду. Содержит в себе девять констант: HOST, DEVICE_OS, CLIENT_VERSION, DEVICE_ID, ACCEPT, USER_AGENT, ACCEPT_LANGUAGE, CLIENT_SECRET, OS, которые используются для передачи в теле запроса.

Помимо этого, содержит в себе четыре переменные: session_id, refresh_token, phone, code, в которых будут храниться уникальные идентификатор сессии и токен обновления, получаемые после авторизации, номер телефона и код из СМС, соответственно.

Также, присутствуют четыре функции: send_phone_number, confirm_phone_number, get_ticket_id, get_ticket, которые используются для авторизации номера, подтверждении номера при помощи кода из СМС, получении идентификатора чека и получении информации из чека по идентификатору.

2.1.2 CheckBox

Класс CheckBox наследуется от стандартного класса UIKit UIButton. Используется для отображения пустого квадрата и квадрата с галочкой внутри. Содержит в себе две переменные типа UIImage, checkedImage и uncheckedImage, checkedImage содержит изображение квадрата с галочкой внутри, uncheckedImage — изображение пустого квадрата. Также, присутствует переменная isChecked, имеющая функцию, меняющую изображение, в зависимости от статуса переменной.

2.1.3 Product

Класс Product используется для загрузки информации из json файла и передачи в базу данных. Содержит в себе четыре переменные: name типа String, хранящую в себе название продукта из чека, price типа Double, хранящую в себе цену продукта из чека, quantity типа Double, хранящую в себе количество единиц продукта из чека и sum типа Double, хранящую в себе количество денег, заплаченных за данный продукт из чека.

Также, присутствует функция, инициализирующая экземпляр класса Product.

2.1.4 Shop

Класс Shop наследуется от стандартного класса ObjectiveC NSObject и стандартного класса MapKit MKAnnotationView. Используется для визуализации магазинов на карте. Содержит в себе три переменные: title типа String?, хранящую в себе название магазина, coordinate типа CLLocationCoordinate2D, хранящую в себе координаты магазина на карте и info типа String, хранящую в дополнительную информацию о магазине.

Также, присутствует функция, инициализирующая экземпляр класса Shop.

2.2 Основная часть программы

В данной части будут рассмотрены все экраны приложения, перечислены элементы, присутствующие на них, и функции, описанные в них.

Ниже можно увидеть UML диаграмму экранов приложения(2).

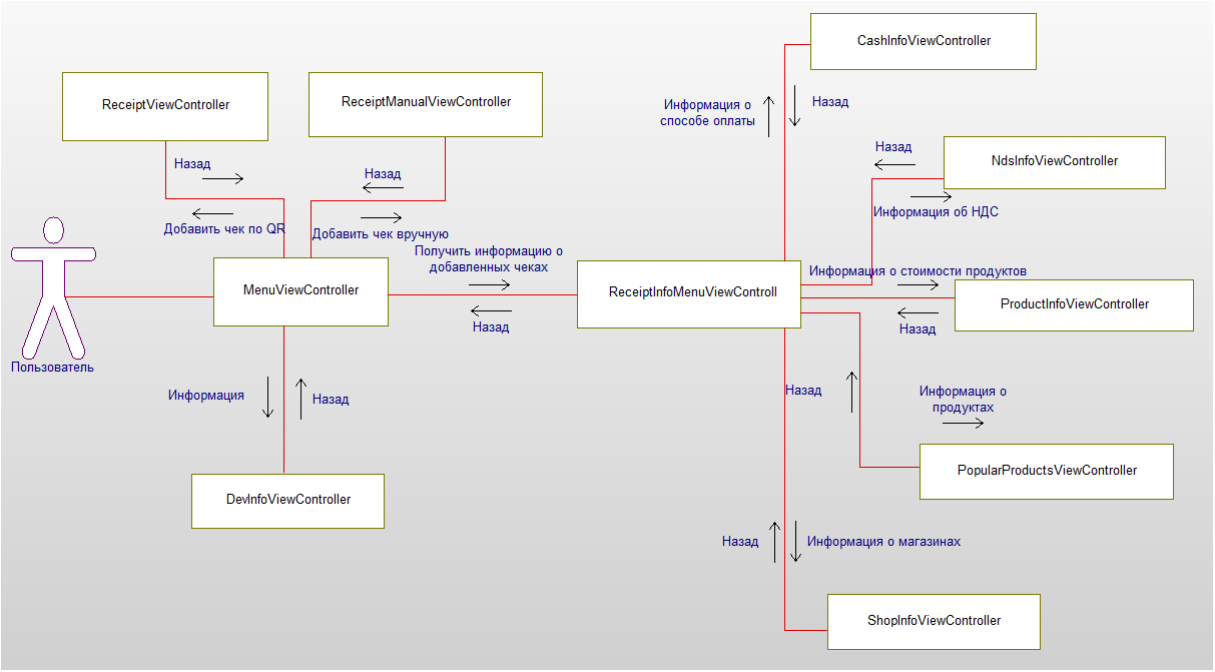

Рисунок 2 – UML диаграмма экранов приложения.

2.2.1 MenuViewController — Главное меню

Данный экран служит главным меню приложения. На нем доступно четыре кнопки: «Добавить чек по QR», «Добавить чек вручную», «Получить информацию о добавленных чеках», «Информация о разработчике», один лейбл, содержащий название, и одно фоновое изображение. Также в классе экрана определены функции assignbackground, initButtons, initLabels, инициализирующие данные элементы.

Кнопка «Добавить чек по QR» перемещает на экран добавления чека с помощью QR кода ReceiptViewController (см. раздел 2.2.3).

Кнопка «Добавить чек вручную» перемещает на экран добавления чека вручную ReceiptManualViewController (см. раздел 2.2.4).

Кнопка «Получить информацию о добавленных чеках» перемещает на экран выбора типа визуализируемой информации, полученной из чеков ReceiptInfoMenuViewController (см. раздел 2.2.5).

Кнопка «Информация о разработчике» перемещает на экран с информацией о разработчике DevInfoViewController (см. раздел 2.2.2).

Функция assignBackground создает фоновое изображение с нужными параметрами и размещает его на экране.

Функция initButtons инициализирует свойства всех кнопок на экране.

Функция initLabels инициализирует свойства лейбла на экране.

2.2.2 DevInfoViewController — Экран с разработчиками приложения

Данный экран используется для показа разработчика приложения. На экране доступна одна кнопка: «Назад», одно фоновое изображение и один лейбл, содержащий информацию о разработчике. Также в файле сцены определены функции assignBackground и initLabels создающие данные элементы.

Кнопка «Назад» возвращает на экран MenuViewController — в главное меню (см. раздел 2.2.1).

Функция assignBackground создает фоновое изображение с нужными параметрами и размещает его на экране.

Функция initLabels инициализирует свойства лейбла на экране.

2.2.3 ReceiptViewController — Экран добавления чека с помощью QR кода

На этом экране можно выбрать фотографию QR -кода из галереи, произвести аутентификацию при помощи мобильного телефона и добавить данные из чека в базу данных. Изначально на экране доступно две кнопки: «Назад», «Выберите фотографию QR кода из галереи», лейбл, содержащий название данного экрана, три лейбла, содержащих шаги, необходимые для добавления информации из чека в базу данных, три чекбокса, соответствующих вспомогательным лейблам, и фоновое изображение.

Кнопка «Назад» возвращает на экран MenuViewController — в главное меню (см. раздел 2.2.1).

Кнопка «Выберите фотографию QR кода из галереи» открывает галерею.

После выбора изображения на экране появляется три новых элемента: кнопка «Далее», лейбл с инструкцией к дальнейшему действию и текстовое поле для ввода номера телефона, необходимого для аутентификации. Помимо этого, в первом чекбоксе появляется галочка. При этом, с экрана пропадает кнопка «Выберите фотографию QR кода из галереи».

После введения номера телефона в текстовое поле и нажатия на кнопку «Далее» на введенный номер телефона приходит сообщение с кодом подтверждения от Федеральной Налоговой Службы РФ. После этого обновляется текст на лейбле с инструкцией и заглушка на текстовом поле. Помимо этого, во втором чекбоксе появляется галочка.

После ввода кода подтверждения из смс и нажатия на кнопку «Далее» с

экрана пропадает лейбл с инструкцией, текстовое поле и кнопка «Далее». Вместо них, на экране появляется лейбл со статусом операции добавления в базу, лейбл с количеством продуктов, содержащихся в чеке, и таблица с названиями этих продуктов. Также, в третьем чекбоксе появляется галочка.

Также в классе экрана определены функции assignbackground, initTables, initLabels, initButtons, initCheckBoxes, initTextFields, инициализирующие элементы на экране, detectQRCode, расшифровывающая QR code на фотографии, функции imagePickerController, imagePickerControllerDidCancel, отвечающие за открытие галереи и выбор изображения, две вспомогательные функции tableView, отвечающие за отображение ячеек в таблице, и функция ProceedButtonPressed, которая запускается при нажатия на кнопку «Далее» и отвечает за авторизацию и добавление информации в базу данных.

Функция initCheckBoxes инициализирует свойства чекбоксов на экране.

Функция initTextFields инициализирует свойства текстовых полей на экране.

Функция detectQRCode выполняет расшифровку QR кода на изображении.

Помимо этого, в классе определены пять вспомогательных переменных:

receiptGetter — переменная, которая инициализируется экземпляром класса GetReceiptInfo, используется для получения информации из чека при помощи REST API.

stateVariable — переменная, которая изначально инициализируется как False, используется для распознавания между ожиданием ввода телефонного номера и ожиданием ввода кода подтверждения.

qrImage — переменная, которая изначально инициализируется как nil, используется для хранения изображения QR кода, выбранного пользователем в галерее.

qrText — переменная, которая изначально инициализируется как пустая строка, используется для хранения расшифрованной информации из QR кода, выбранного пользователем в галерее.

products — переменная, которая изначально инициализируется как пустая массив, используется для экземпляров класса Product (2.1.3), получаемых из чека.

2.2.4 ReceiptManualViewController — Экран добавления чека вручную

На этом экране можно вручную ввести данные из чека в соответствующие поля и добавить их в базу данных. Изначально на экране доступно две

кнопки «Ввод», одна для добавления информации о продуктах, другая — для добавления информации о чеке и магазине, два лейбла, содержащих инструкции для ввода, тринадцать лейблов и соответствующих им текстовых полей, необходимых для ввода информации, и фоновое изображение.

После введения информации о продукте и нажатия на верхнюю кнопку «Ввод», на верхний лейбл выводится изображение о статусе добавления информации о продукте в базу данных.

После введения информации о чеке и магазине и нажатия на нижнюю кнопку «Ввод», на нижний лейбл выводится изображение о статусе добавления информации о чеке и магазине в базу данных.

Также в классе экрана определены функции assignbackground, initLabels, initButtons, initFields, инициализирующие элементы на экране, AddProduct и AddReceipt, вызываемые по нажатию на определенную кнопку и отвечающие за добавление информации в базу данных.

Функция AddProduct вызывается после нажатия на верхнюю кнопку «Ввод», используется для добавления информации о продукте в базу данных.

Функция AddReceipt вызывается после нажатия на нижнюю кнопку «Ввод», используется для добавления информации о чеке и магазине в базу данных.

2.2.5 ReceiptInfoMenuViewController — Экран выбора типа визуализируемой информации о чеке

Данный экран используется для выбора типа визуализируемой информации о чеке. На экране доступно шесть кнопок: «Информация о способе оплаты», «Информация об НДС», «Информация о стоимости продуктов», «Информация о продуктах», «Информация о магазинах», «Назад», одно фоновое изображение и один лейбл, содержащий информацию об экране. Также в файле сцены определены функции assignbackground, initLabels, initButtons, инициализирующие элементы на экране.

Кнопка «Назад» возвращает на экран MenuViewController — в главное меню (см. раздел 2.2.1).

Кнопка «Информация о способе оплаты» перемещает на экран CashInfoViewController — на экран визуализации информации о способе оплаты (см. раздел 2.2.6).

Кнопка «Информация об НДС» перемещает на экран NdsInfoViewController — на экран визуализации информации об НДС (см. раз-

дел 2.2.7).

Кнопка «Информация о стоимости продуктов» перемещает на экран ProductInfoViewController — на экран визуализации информации о стоимости продуктов (см. раздел 2.2.8).

Кнопка «Информация о продуктах» перемещает на экран PopularProductsViewController — на экран визуализации информации о самых популярных продуктах (см. раздел 2.2.9).

Кнопка «Информация о магазинах» перемещает на экран ShopInfoViewController — на экран визуализации информации о самых популярных магазинах (см. раздел 2.2.10).

Функция assignBackground создает фоновое изображение с нужными параметрами и размещает его на экране.

Функция initButtons инициализирует свойства всех кнопок на экране.

Функция initLabels инициализирует свойства лейбла на экране.

2.2.6 CashInfoViewController — Экран визуализации информации о способе оплаты

Данный экран используется для визуализации информации о наличном и безналичном способах оплаты в абсолютном и процентном соотношении при помощи графиков. На экране доступны две кнопки: «Продолжить», «Назад», одно фоновое изображение, один лейбл, содержащий инструкции, колесо выбора времени и два графика. Также в файле сцены определены функции assignbackground, initLabels, initButtons, initCharts, initTimePickers, инициализирующие элементы на экране, функция ButtonPressed, вызываемая после нажатия на кнопку и отвечающая за извлечение информации из базы данных, функции customizePieChart и customizeBarChart, отвечающие за настройку графиков, вспомогательные функции numberOfComponents и pickerView, отвечающие за наполнение колеса выбора времени.

Кнопка «Назад» возвращает на экран ReceiptInfoMenuViewController — в меню выбора типа визуализируемой информации о чеке (см. раздел 2.2.5).

После выбора нужного пользователю временного промежутка из следующих доступных: 1 месяц, 3 месяца, 6 месяцев, 1 год, на колесе выбора времени и нажатия на кнопку «Продолжить» происходит обновление графиков.

Функция ButtonPressed посылает запрос к базе данных в соответствии с выбранным пользователем промежутком времени и вызывает функции настрой-

ки графиков.

Функция customizePieChart настраивает первый граф.

Функция customizeBarChart настраивает второй граф.

Помимо этого, в классе определены четыре вспомогательных переменных: TimePickerData — переменная, которая инициализируется массивом строк, используется для наполнения колеса выбора времени значениями.

types — переменная, которая инициализируется массивом строк, используется для настройки графиков.

percents — переменная, которая инициализируется пустым массивом, используется для настройки графиков.

values — переменная, которая инициализируется пустым массивом, используется для настройки графиков.

2.2.7 NdsInfoViewController — Экран визуализации информации о типе уплаченного НДС

Данный экран используется для визуализации информации о наличном и безналичном способах оплаты в абсолютном и процентном соотношении при помощи графиков. Изначально на экране доступны две кнопки: «Продолжить», «Назад», одно фоновое изображение, один лейбл, содержащий инструкции, колесо выбора времени и два графика. Также в файле сцены определены функции assignbackground, initLabels, initButtons, initCharts, initTimePickers, инициализирующие элементы на экране, функция ButtonPressed, вызываемая после нажатия на кнопку и отвечающая за извлечение информации из базы данных, функции customizePieChart и customizeBarChart, отвечающие за настройку графиков, вспомогательные функции numberOfComponents и pickerView, отвечающие за наполнение колеса выбора времени.

Кнопка «Назад» возвращает на экран ReceiptInfoMenuViewController — в меню выбора типа визуализируемой информации о чеке (см. раздел 2.2.5).

После выбора нужного пользователю временного промежутка на колесе выбора времени из следующих доступных: 1 месяц, 3 месяца, 6 месяцев, 1 год, и нажатия на кнопку «Продолжить» происходит обновление графиков, появляется лейбл с информацией и текстовое поле с общим количеством уплаченного НДС за выбранный пользователем промежуток времени. Помимо этого, с экрана пропадает лейбл, содержащий инструкции, кнопка «Продолжить» и колесо выбора времени.

Функция ButtonPressed посылает запрос к базе данных в соответствии с выбранным пользователем промежутком времени и вызывает функции настройки графиков.

Функция customizePieChart настраивает первый граф.

Функция customizeBarChart настраивает второй граф.

Помимо этого, в классе определены четыре вспомогательных переменных: TimePickerData — переменная, которая инициализируется массивом строк, используется для наполнения колеса выбора времени значениями.

types — переменная, которая инициализируется массивом строк, используется для настройки графиков.

percents — переменная, которая инициализируется пустым массивом, используется для настройки графиков.

values — переменная, которая инициализируется пустым массивом, используется для настройки графиков.

2.2.8 ProductInfoViewController — Экран визуализации информации о количестве потраченных денег

Данный экран используется для визуализации информации о количестве потраченных денег по месяцам в абсолютном и процентном соотношении при помощи графиков. Изначально на экране доступны две кнопки: «Продолжить», «Назад», одно фоновое изображение, один лейбл, содержащий инструкции, колесо выбора времени и два графика. Также в файле сцены определены функции assignbackground, initLabels, initButtons, initCharts, initTimePickers, инициализирующие элементы на экране, функция ButtonPressed, вызываемая после нажатия на кнопку и отвечающая за извлечение информации из базы данных, функции customizePieChart и customizeBarChart, отвечающие за настройку графиков, вспомогательные функции numberOfComponents и pickerView, отвечающие за наполнение колеса выбора времени, и функция colorsOfCharts, отвечающая за создание цветов для графиков.

Кнопка «Назад» возвращает на экран ReceiptInfoMenuViewController — в меню выбора типа визуализируемой информации о чеке (см. раздел 2.2.5).

После выбора нужного пользователю временного промежутка на колесе выбора времени из следующих доступных: 1 месяц, 3 месяца, 6 месяцев, 1 год, и нажатия на кнопку «Продолжить» происходит обновление графиков, появляется лейбл с информацией и текстовое поле с общим количеством потраченных

денег за выбранный пользователем промежуток времени. Помимо этого, с экрана пропадает лейбл, содержащий инструкции, кнопка «Продолжить» и колесо выбора времени.

Функция ButtonPressed посылает запрос к базе данных в соответствии с выбранным пользователем промежутком времени и вызывает функции настройки графиков.

Функция customizePieChart настраивает первый граф.

Функция customizeBarChart настраивает второй граф.

Функция colorsOfCharts создает случайный массив разных цветов для окраски графов.

Помимо этого, в классе определены четыре вспомогательных переменных: TimePickerData — переменная, которая инициализируется массивом строк, используется для наполнения колеса выбора времени значениями.

types — переменная, которая инициализируется массивом строк, используется для настройки графиков.

percents — переменная, которая инициализируется пустым массивом, используется для настройки графиков.

values — переменная, которая инициализируется пустым массивом, используется для настройки графиков.

2.2.9 PopularProductsViewController — Экран визуализации информации о популярных продуктах по сумме и количеству

Данный экран используется для визуализации информации о популярных продуктах по сумме и количеству при помощи таблицы. Изначально на экране доступны две кнопки: «Продолжить», «Назад», одно фоновое изображение, один лейбл, содержащий инструкции, колесо выбора времени, таблица и два сегментированных элемента. Также в файле сцены определены функции assignbackground, initLabels, initButtons, initTables, initDatePickers, initSegmentedControls, инициализирующие элементы на экране, функция ButtonPressed, вызываемая после нажатия на кнопку и отвечающая за извлечение информации из базы данных, вспомогательные функции numberOfComponents и pickerView, отвечающие за наполнение колеса выбора времени, и вспомогательные функции tableView, отвечающие за оформление и наполнение таблицы.

Кнопка «Назад» возвращает на экран ReceiptInfoMenuViewController — в меню выбора типа визуализируемой информации о чеке (см. раздел 2.2.5).

После выбора нужного пользователю временного промежутка на колесе выбора времени из следующих доступных: 1 месяц, 3 месяца, 6 месяцев, 1 год, выбора количества продуктов, выводимых на экран, выбора, по какому признаку будет производиться сортировка продуктов: сумма или количество, и нажатия на кнопку «Продолжить» происходит обновление таблицы.

Функция ButtonPressed посылает запрос к базе данных в соответствии с выбранным пользователем промежутком времени и вызывает функции настройки графиков.

Помимо этого, в классе определены три вспомогательных переменных: tableCellLength — переменная, которая инициализируется числом, используется для настройки размера ячейки таблицы.

productsArray — переменная, которая инициализируется пустым массивом, используется хранения информации о продуктах, извлеченной из таблицы.

TimePickerData — переменная, которая инициализируется массивом строк, используется для наполнения колеса выбора времени значениями.

2.2.10 ShopInfoViewController — Экран визуализации информации о популярных магазинах

Данный экран используется для визуализации информации о популярных магазинах по количеству покупок при помощи карты. На экране доступны две кнопки: «Продолжить», «Назад», одно фоновое изображение, один лейбл, содержащий инструкции, карта и сегментированный элемент. Также в файле сцены определены функции assignbackground, initLabels, initButtons, initSegmentedControls, инициализирующие элементы на экране, функция ButtonPressed, вызываемая после нажатия на кнопку и отвечающая за извлечение информации из базы данных, и функция coordinates, отвечающие за сопоставление адреса магазина его координатам на карте.

Кнопка «Назад» возвращает на экран ReceiptInfoMenuViewController — в меню выбора типа визуализируемой информации о чеке (см. раздел 2.2.5).

После выбора количества выводимых на карту магазинов и нажатия на кнопку «Продолжить» происходит обновление карты.

Функция ButtonPressed посылает запрос к базе данных в соответствии с выбранным пользователем промежутком времени и вызывает функции настрой-

ки графиков.

Функция coordinates возвращает координаты переданного адреса.

ЗАКЛЮЧЕНИЕ

Было разработано мобильное приложение для сбора, обработки и визуализации информации из чеков под операционную систему iOS на языке программирования Swift.

В результате написания работы должны быть решены следующие задачи:

- было произведено изучение работы REST API на языке Swift;
- был разработан механизм получения данных с nalog.ru;
- была произведена разработка механизма считывания QR-кода;
- была распланирована и создана база данных;
- были реализованы взаимодействия с базой данных;
- была произведена визуализация данных;