Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ

КАФЕДРА КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ И ПРОГРАММНОЙ ИНЖЕНЕРИИ

КУРСОВАЯ РАБОТА   
ЗАЩИЩЕНА С ОЦЕНКОЙ

РУКОВОДИТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ст. преподаватель |  |  |  | Е.О. Шумова |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОМУ ПРОЕКТУ |
| Разработка системы классов для обеспечения работы поликлиники |
| по дисциплине: ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4831 |  |  |  | К. А. Корнющенков |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2020

Оглавление

[Задание на курсовой проект 3](#_Toc58160704)

[1 Постановка задачи 4](#_Toc58160705)

[1.1 Анализ предметной области 4](#_Toc58160706)

[1.2 Формулировка технического задания 4](#_Toc58160707)

[2 Проектирование классов 5](#_Toc58160708)

[2.1 UML диаграмма классов 5](#_Toc58160709)

[2.2 Классы сущностей 5](#_Toc58160710)

[2.3 Управляющие классы 5](#_Toc58160711)

[2.4 Интерфейсные классы 6](#_Toc58160712)

[2.5 Используемые паттерны проектирования 6](#_Toc58160713)

[3 Разработка приложения 7](#_Toc58160714)

[3.1 Описание технической реализации 7](#_Toc58160715)

[3.2 Разработка интерфейса приложения 7](#_Toc58160716)

[3.3 Реализация классов 18](#_Toc58160717)

[4 Тестирование 23](#_Toc58160718)

[Заключение 30](#_Toc58160719)

[Список использованных источников 30](#_Toc58160720)

# Задание на курсовой проект

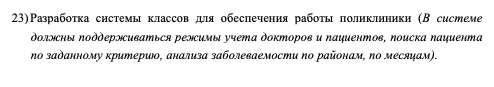
В ходе курсового проекта необходимо разработать приложение, позволяющее протестировать взаимодействие объектов классов, спроектированных и реализованных студентом для решения конкретной задачи при заданных критериях.

При разработке программного обеспечения следует использовать шаблоны проектирования. Также предпочтение должно быть отдано графическому приложению.

При этом подлежат разработке следующие вопросы:

а) анализ задания;  
б) разработка классов  
в) разработка тестового приложения;  
г) оформление пояснительной записки по результатам выполнения проекта д) создание презентации к проекту

Вариант задания:



# 1 Постановка задачи

# 1.1 Анализ предметной области

Разработка информационной системы для обеспечения работы поликлиники важная часть для упрощения жизни врачей и клиентов данного медицинского заведения. Каждый из нас сталкивался с необходимостью записаться к врачу или необходимостью посмотреть какую-то информацию об истории своих болезней, операций или хотя бы давность прививок. Но зачастую это сложно, ибо нету какой-то единой платформы, где это можно узнать быстро и удобно. Данная информационная система должна позволить посетителям легко и просто записаться к врачам, аналитикам отслеживать статистику заболеваемости по разным параметрам. А также врачи могут смотреть историю болезней и всю необходимую информацию посетителей через одну информационную систему, что упрощает и оптимизирует их рутинную работу, позволяя сосредоточиться на пациенте, не тратя лишняя время и силы на бумажную волокиту.

# 1.2 Формулировка технического задания

В ходе выполнения курсовой работы необходимо разработать информационную систему для обеспечения полноценной работы поликлиники. Приложение должно делиться на 3 блока. Первый блок отвечает и предоставляет функционал для работы врачей: добавление нового врача, изменение данных, удаление сотрудника, а также возможность просматривать историю посещений и поиск врачей по заданному параметру (по профессии, по ФИО). Второй блок отвечает за работу с пациентами. В данном блоке должен быть аналогичный функционал, что и у врачей. Это добавление нового пациента, редактирование и удаление данных, поиск по поиск врачей по заданному параметру (по симптомам, по ФИО). А также у пациента должна быть возможность записаться к выбранному врачу, указав район, в котором он живет и симптом, который у него проявляется. Третья вкладка отвечает за предоставление данных статистики. Статистика делится на 2 части. Первая диаграмма показывает динамику заболеваемости, количество обращений по районам города, а вторая количество обращений по дням недели. Все данные должны автоматически сохраняться при закрытии информационной системы и сами загружаться при открытии приложения.

В разрабатываемом приложении необходимо хранить у пациентов:

- ФИО

- Симптомы

- Район проживания

- Изображение пациента

В разрабатываемом приложении необходимо хранить у врачей:

- ФИО

- Профессия

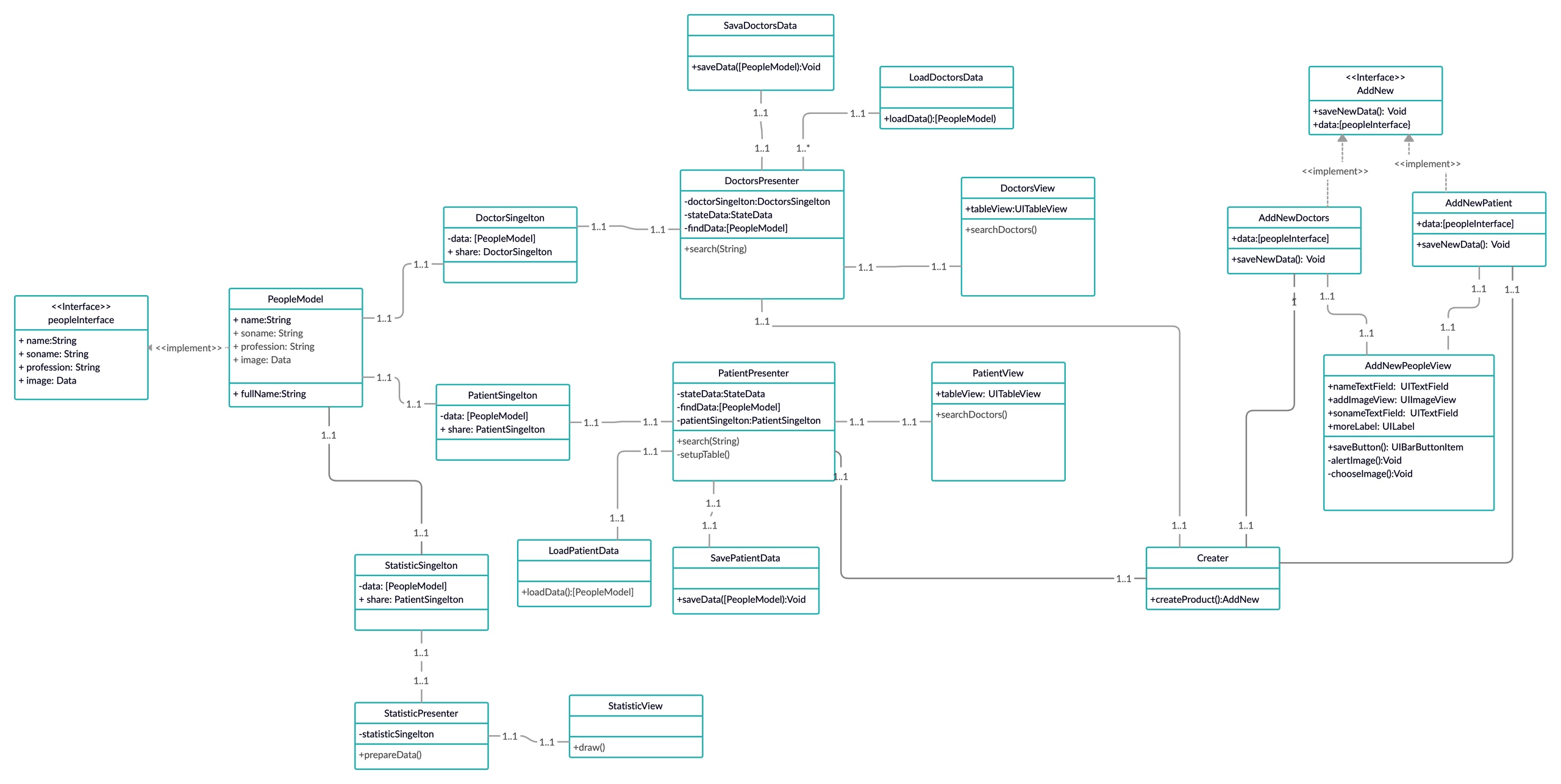
- Изображение врача

- история пациентов, которые посетили данного врача

Статистика формируется автоматически на основе данных врачей и пациентов. Новые данные в статистику добавляются в момент, когда пациент закончил запись на прием.

# 2 Проектирование классов

# 2.1 UML диаграмма классов



# 2.2 Классы сущностей

DoctorModel – класс сущности, который используется в качестве модели для хранении данных об одном докторе.

PatientModel – класс сущности, который используется в качестве модели для хранении данных об одном посетителе.

StatisticModel – класс сущности, который используется в качестве модели для хранении данных для статистике.

# 2.3 Управляющие классы

# 

DoctorSingelton – класс для хранения и работы с данными о врачах.

PatientSingelton – класс для хранения и работы с данными о пациентах.

StatisticSingelton – класс для работы с данными о статистике.

StatisticPresenter – класс для подготовки данных статистики для отображения.

LoadPatientData – класс для загрузки сохраненных данных о пациентах.

SaveDoctorsData – класс для сохранения данных о докторах.

DoctorsPresenter – класс для подготовки данных о докторах для отображения.

PatientPresenter – класс для подготовки данных о пациентах для отображения.

SavePatientData – класс для сохранения данных о пациентах.

LoadDoctorsData – класс для загрузки сохраненных данных о докторах.

Creater – класс для создания объекта интерфейса AddNew.

AddNewDoctors – класс обработки новых данных о докторах.

AddNewPatient – класс обработки новых данных о пациентах.

# 2.4 Интерфейсные классы

StatisticView – класс для работы с графическим интерфейсов статистики.

DoctorsView – класс для работы с графическим интерфейсов связанный с информацией о докторах.

PatientView – класс для работы с графическим интерфейсов связанный с информацией о пациентах.

AddNewPeopleView – класс для работы с графическим интерфейсов для добавления новых данных.

# 2.5 Используемые паттерны проектирования

В программе используются 3 проектирующих паттерна программирования Одиночка для удобного хранения и передачи данных между классами. Данный паттерн реализуется в классах DoctorSingelton, PatientSingelton, StatisticSingelton.

Так же в программе Фабричный метод, который служит для создания объектов, которые поддерживают интерфейс AddNew. Фабричный метод реализован с помощью интерфейса AddNew, который поддерживают 2 класса AddNewDoctors и AddNewPatien и для создания и выбора необходимо объекта есть специальный класс Creater.

# 

# 3 Разработка приложения

# 3.1 Описание технической реализации

Для реализации данного приложения был выбран следующий стек технологий:

Язык программирования Swift, для создания UI- компонентов библиотека данного языка UIKit, которая позволяет создавать императивный UI, а для работы с данными UserDefaults, что является контейнером для хранения значений по ключу. Конечный продукт будет представлен в виде мобильного приложения под платформу IOS.

# 3.2 Разработка интерфейса приложения

**DoctorsController** – интерфейсный класс, который отображает список всех докторов.

Данный класс содержит данные свойства:

- tableView: UITableView – таблица, которая содержит ячейки с данными о докторах

- presenter: DoctorsPresenter – ссылка на класс DoctorsPresenter

Данный класс содержит данные методы:

- viewDidLoad() – метод жизненного цикла, который срабатывает при открытии экрана

- viewWillAppear() – метод жизненного цикла, который срабатывает в когда экран стал активен и отобразился

- searchDoctors(sender: UIBarButtonItem) – метод, который срабатывает при нажатии на кнопку поиска

- prepare(for segue: UIStoryboardSegue, sender: Any?) – метод, который срабатывает при переходе и передает данные на новый экран

- tableView(\_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int – метод, который возвращает количество ячеек в таблице

- tableView(\_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell – метод, который возвращает контент каждой ячейки

- tableView(\_ tableView: UITableView, commit editingStyle: UITableViewCell.EditingStyle, forRowAt indexPath: IndexPath) – метод, который срабатывает при удалении ячейки

- tableView(\_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) – метод, который срабатывает при нажатии на ячейку



Иллюстрация интерфейса класса DoctorsController

**PatientController**– интерфейсный класс, который отображает список всех пациентов.

Данный класс содержит данные свойства:

- tableView: UITableView – таблица, которая содержит ячейки с данными о докторах

- presenter: PatientPresenter– ссылка на класс PatientPresenter

Данный класс содержит данные методы:

- viewDidLoad() – метод жизненного цикла, который срабатывает при открытии экрана

- viewWillAppear() – метод жизненного цикла, который срабатывает в когда экран стал активен и отобразился

- searchPatient (sender: UIBarButtonItem) – метод, который срабатывает при нажатии на кнопку поиска

- prepare(for segue: UIStoryboardSegue, sender: Any?) – метод, который срабатывает при переходе и передает данные на новый экран

- tableView(\_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int – метод, который возвращает количество ячеек в таблице

- tableView(\_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell – метод, который возвращает контент каждой ячейки

- tableView(\_ tableView: UITableView, commit editingStyle: UITableViewCell.EditingStyle, forRowAt indexPath: IndexPath) – метод, который срабатывает при удалении ячейки

- tableView(\_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) – метод, который срабатывает при нажатии на ячейку



Иллюстрация интерфейса класса PatientController

**StatisticController**– интерфейсный класс, который отображает статистику.

Данный класс содержит данные свойства:

- talbeView: UITableView - таблица, которая содержит ячейки с данными о выбранной статистике

- segmentControl: UISegmentedControl – контроллер, который позволяет выбрать ту статистику, которую мы хотим отобразить

- presenter: StatisticPresenter – ссылка на класс StatisticPresenter

Данный класс содержит данные методы:

- viewDidLoad() – метод жизненного цикла, который срабатывает при открытии экрана

- viewWillAppear() – метод жизненного цикла, который срабатывает в когда экран стал активен и отобразился

- chengeValue() – метод, который позволяет отслеживать изменения segmentControl

- tableView(\_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int – метод, который возвращает количество ячеек в таблице

- tableView(\_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell – метод, который возвращает контент каждой ячейки

- tableView(\_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) – метод, который срабатывает при нажатии на ячейку

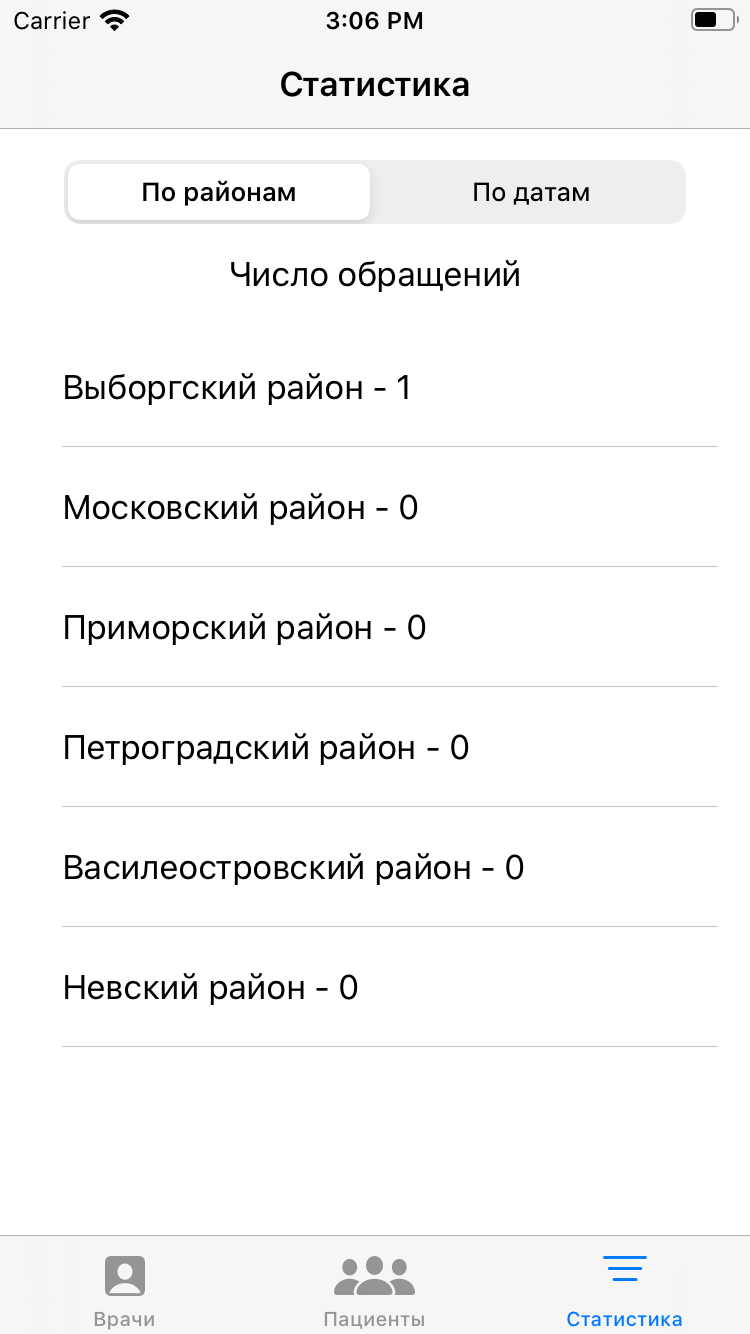


Иллюстрация интерфейса класса StatisticController

**PickerController**– интерфейсный класс, который дает возможность выбрать район для записи к врачу.

Данный класс содержит данные свойства:

- presenter: PickerPresenter – ссылка на класс PickerPresenter

- picker: UIPickerView – пикер для выбора данных

Данный класс содержит данные методы:

- viewDidLoad() - метод жизненного цикла, который срабатывает при открытии экрана

- add() – метод для сохранения выбранных данных

- numberOfComponents(in pickerView: UIPickerView) -> Int – метод, который возвращает количество столбцов в picker

- pickerView(\_ pickerView: UIPickerView, numberOfRowsInComponent component: Int) -> Int – метод, который возвращает количество строк в picker

- pickerView(\_ pickerView: UIPickerView, titleForRow row: Int, forComponent component: Int) -> String? – метод, который возвращает значение для каждой строки picker

- pickerView(\_ pickerView: UIPickerView, didSelectRow row: Int, inComponent component: Int) – метод, который срабатывает при изменении значения активного элемента picker

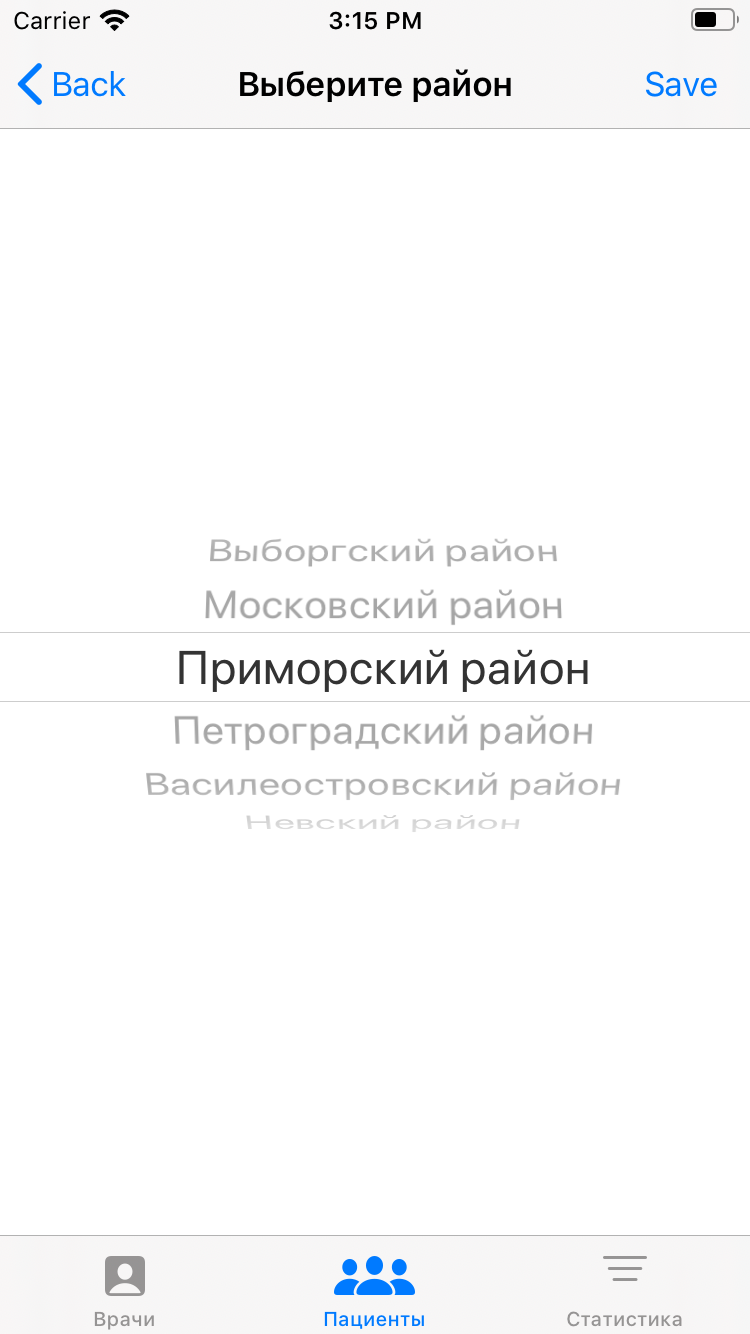


Иллюстрация интерфейса класса PickerController

**EditDoctorsController**– интерфейсный класс, который дает возможность исправиль информацию о выбранном докторе.

Данный класс содержит данные свойства:

- presenter: EditDoctorsPresenter – ссылка на класс EditDoctorsPresenter

- nameTextField: UITextField – текстовое поля для указания имени

- sonameTextField: UITextField – текстовое поля для указания фамилии

- proffessionTextField: UITextField – текстовое поле для указания профессии

- professionLabel: UILabel - вспомагательный ярлык

- peopleImageView: UIImageView – изображения человека

- imageTapGestureRecognizer: UITapGestureRecognizer – элемент нажатия на изображение

Данный класс содержит данные методы:

- viewDidLoad() – метод жизненного цикла, который срабатывает при открытии экрана

- tapPersonImageView() – метод, который срабатывает при нажатии на изображение

- saveAction() – метод, который срабатывает при нажатии на кнопку save

- alertImage() – метод, который открывает alert

- chooseImage(sourse:UIImagePickerController.SourceType) – метод для загрузки нового изображения

- imagePickerController(\_ picker: UIImagePickerController, didFinishPickingMediaWithInfo info: [UIImagePickerController.InfoKey : Any]) – метод, который срабатывает после загружки нового изображения

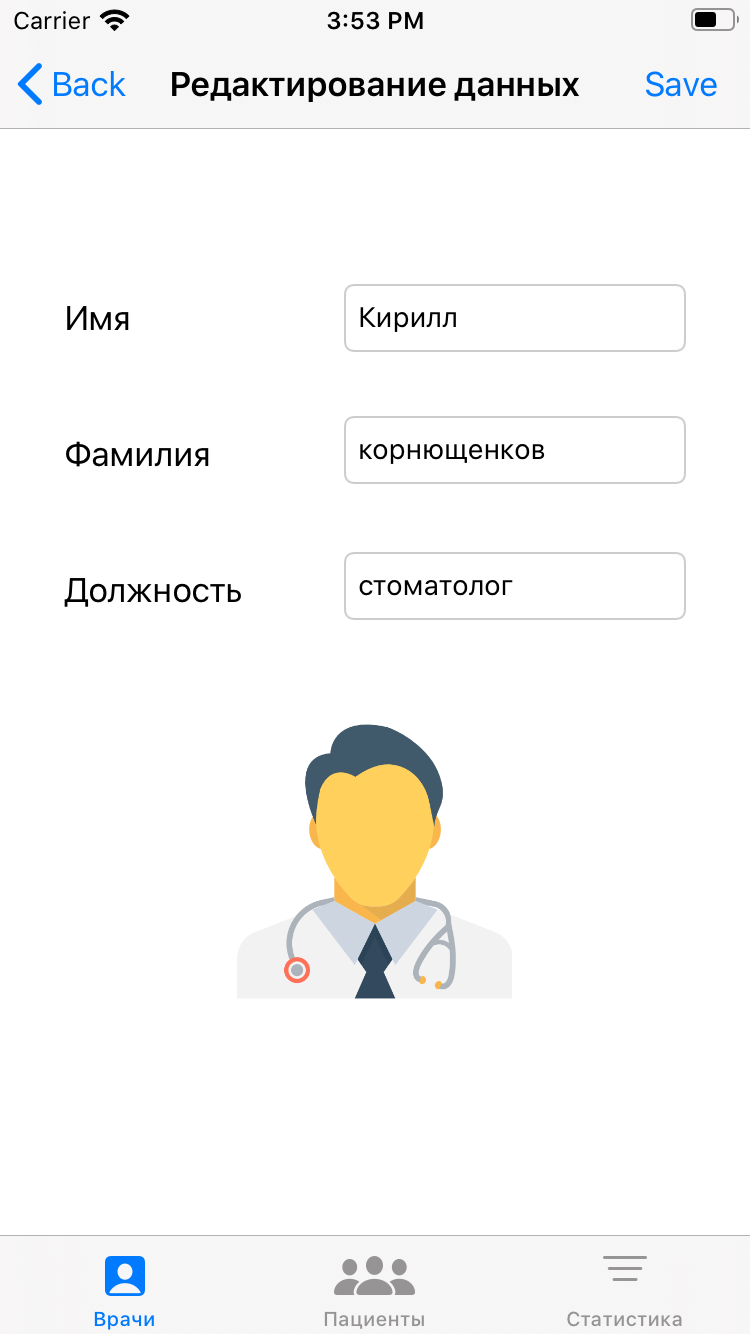


Иллюстрация интерфейса класса EditDoctorsController

**WritePatientToDoctorController**– интерфейсный класс, который дает возможность записать пациента к врачу.

Данный класс содержит данные свойства:

- tableView: UITableView - таблица, которая содержит ячейки с данными

- presenter: WritePatientToDoctorPresenter – ссылка на класс WritePatientToDoctorPresenter

Данный класс содержит данные методы:

- tableView(\_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int – метод, который возвращает количество ячеек в таблице

- tableView(\_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell – метод, который возвращает контент каждой ячейки

- tableView(\_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) – метод, который срабатывает при нажатии на ячейку



Иллюстрация интерфейса класса WritePatientToDoctorController

**EditPatientController**– интерфейсный класс, который дает возможность исправиль информацию о выбранном пациенте.

Данный класс содержит данные свойства:

- presenter: EditPatientPresenter– ссылка на класс EditPatientPresenter

- nameTextField: UITextField – текстовое поля для указания имени

- infoTextField: UITextField - текстовое поля для указания симптомов

- sonameTextField: UITextField – текстовое поля для указания фамилии

- peopleImageView: UIImageView – изображения человека

- imageTapGestureRecognizer: UITapGestureRecognizer – элемент нажатия на изображение

Данный класс содержит данные методы:

- viewDidLoad() – метод жизненного цикла, который срабатывает при открытии экрана

- tapPersonImageView() – метод, который срабатывает при нажатии на изображение

- saveAction() – метод, который срабатывает при нажатии на кнопку save

- alertImage() – метод, который открывает alert

- chooseImage(sourse:UIImagePickerController.SourceType) – метод для загрузки нового изображения

- imagePickerController(\_ picker: UIImagePickerController, didFinishPickingMediaWithInfo info: [UIImagePickerController.InfoKey : Any]) – метод, который срабатывает после загружки нового изображения

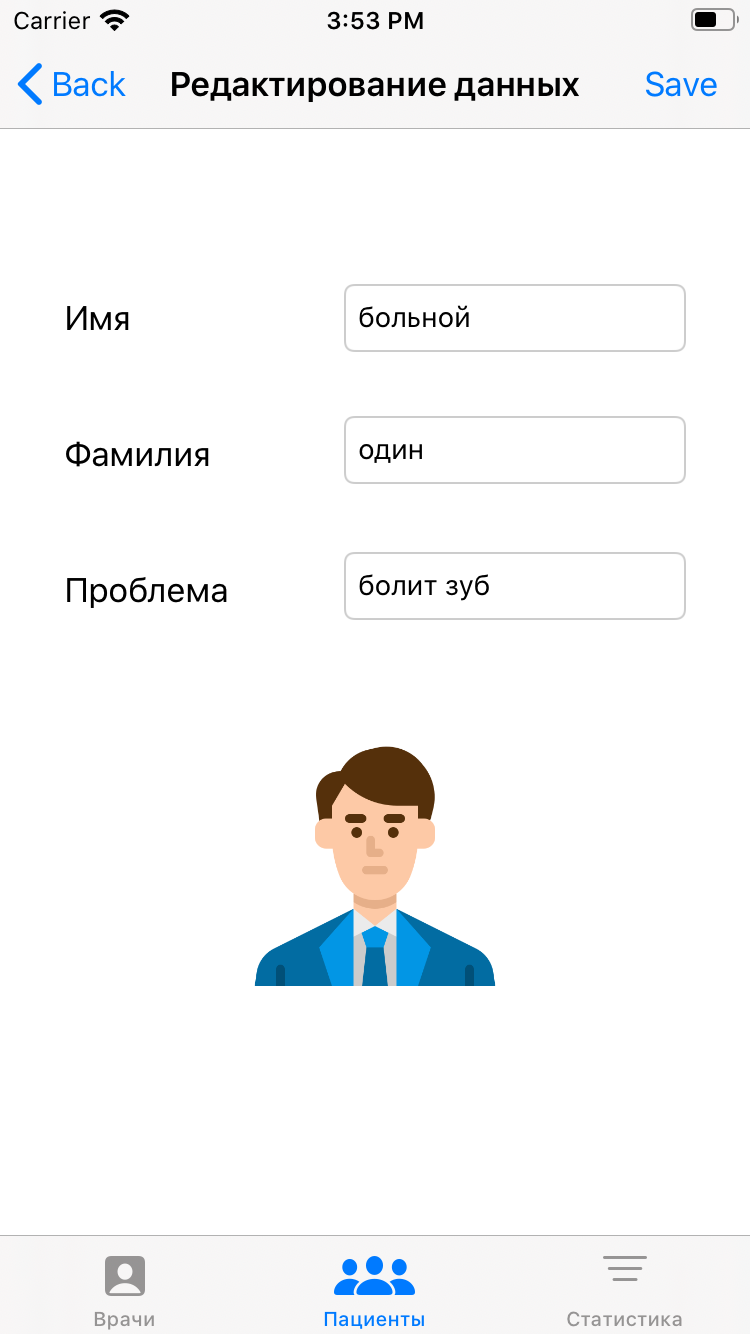


Иллюстрация интерфейса класса EditPatientController

**AddNewPeopleController**– интерфейсный класс, который дает возможность исправиль информацию о выбранном пациенте.

Данный класс содержит данные свойства:

- addImageView: UIImageView – изображение нового пользователя

- nameTextField: UITextField – текстовое поле для имени

- somaneTextField: UITextField – текстовое поле для фамилии

- professionTextField: UITextField – текстовое поля для профессии

- someHelpersLabel: UILabel – вспомогающий ярлык

- imageTapGestureRecognizer: UITapGestureRecognizer - элемент нажатия на изображение

- presenter: AddNewPeoplePresenter – ссылка на класс AddNewPeoplePresenter

Данный класс содержит данные методы:

-viewDidLoad() - метод жизненного цикла, который срабатывает при открытии экрана

- saveButton(\_ sender: UIBarButtonItem) – метод, который срабатывает при нажатии на кнопку save

- tapPersonImageView() – метод, при который срабатывает при нажатии на изображение

- alertImage() – метод, который отображает alert

- chooseImage(sourse:UIImagePickerController.SourceType) – метод для выбора аватара пользователя

- imagePickerController(\_ picker: UIImagePickerController, didFinishPickingMediaWithInfo info: [UIImagePickerController.InfoKey : Any]) – метод для установки нового аватара пользователя

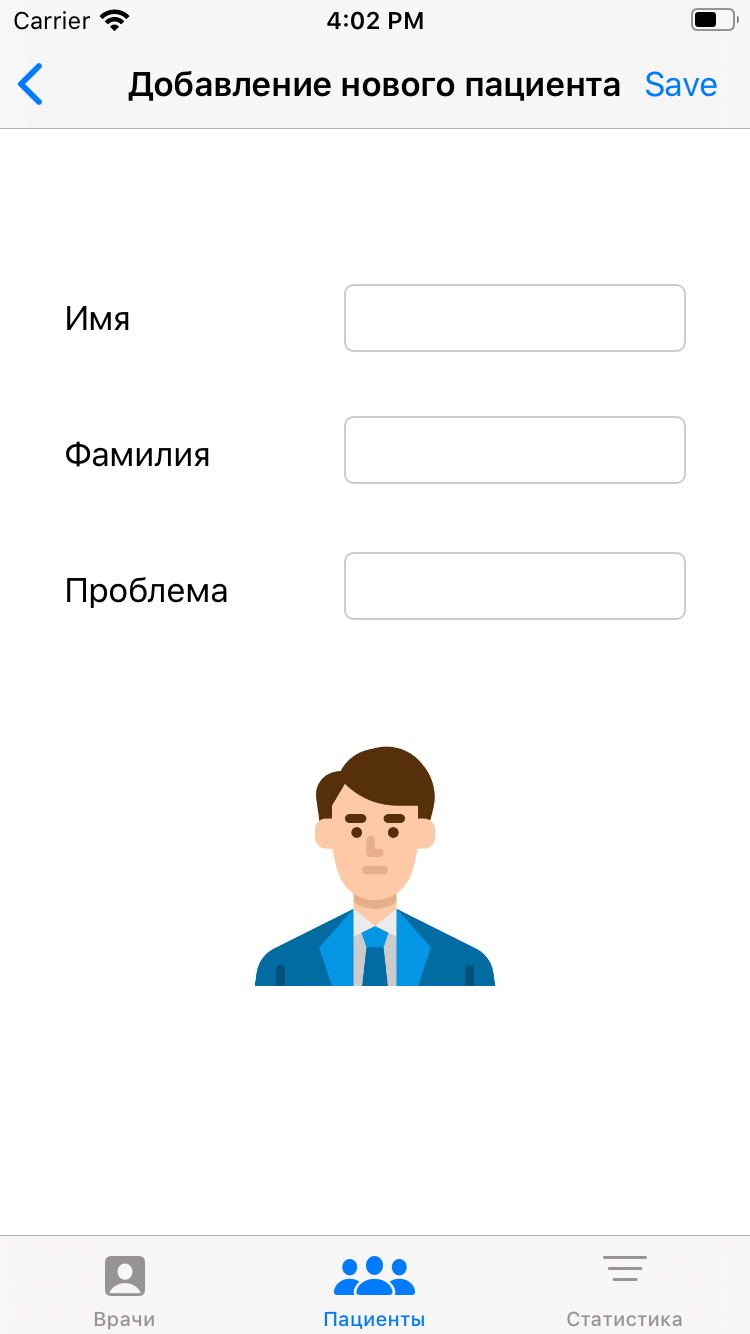


Иллюстрация интерфейса класса **AddNewPeopleController**

**DetailDoctorsController**– интерфейсный класс, который отображает детальную информацию о докторе.

Данный класс содержит данные свойства:

- collectionView: UICollectionView

- fullName: UILabel

- personImage: UIImageView

- profession: UILabel

- presenter: DetailDoctorsPresenter

Данный класс содержит данные методы:

- viewDidLoad() – метод жизненного цикла, который срабатывает при открытии экрана

- viewWillAppear() – метод жизненного цикла, который срабатывает в когда экран стал активен и отобразился

- editProfiel() – метод срабатывает при нажатии на кнопку edit

- collectionView(\_ collectionView: UICollectionView, numberOfItemsInSection section: Int) -> Int – возвращает данные ячейки collectionView

- collectionView(\_ collectionView: UICollectionView, cellForItemAt indexPath: IndexPath) -> UICollectionViewCell – возвращает данные ячейки collectionView

- collectionView(\_ collectionView: UICollectionView, layout collectionViewLayout: UICollectionViewLayout, sizeForItemAt indexPath: IndexPath) -> CGSize – возвращает размер ячейки collectionView

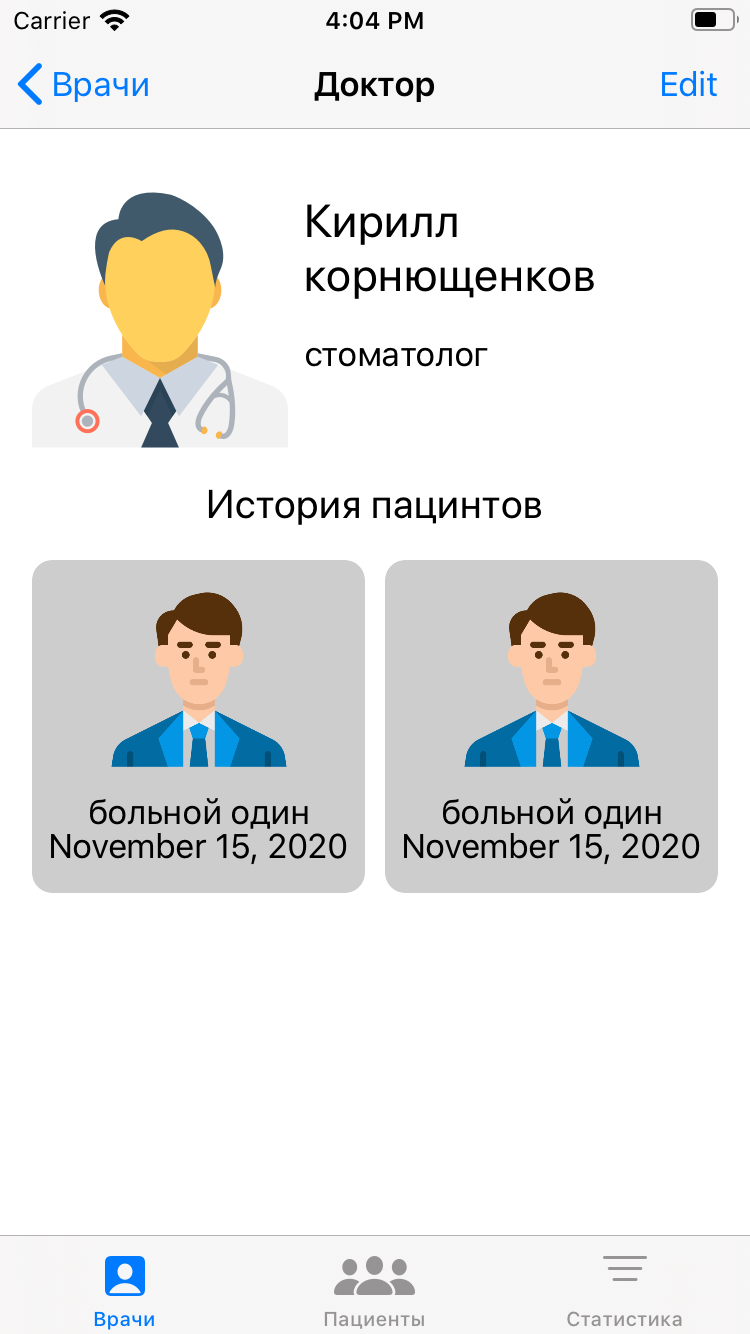


Иллюстрация интерфейса класса DetailDoctorsController

**DetailPersonPresenter** – интерфейсный класс, который отображает детальную информацию о пациентах.

Данный класс содержит данные свойства:

- presenter: DetailPersonPresenter – ссылка на класс DetailPersonPresenter

- peopleImageView: UIImageView – изображения пациента

- fullNameLabel: UILabel – полное имя пользователя

- professionLabel: UILabel – симптом пользователя

Данный класс содержит данные методы:

- viewDidLoad() – метод жизненного цикла, который срабатывает при открытии экрана

- viewWillAppear() – метод жизненного цикла, который срабатывает в когда экран стал активен и отобразился

- writePeople(\_ sender: UIButton) - метод, который срабатывает при нажатии на кнопку “Запись к врачу”

- editProfile() – метод, который срабатывает при нажатии на кнопку edit

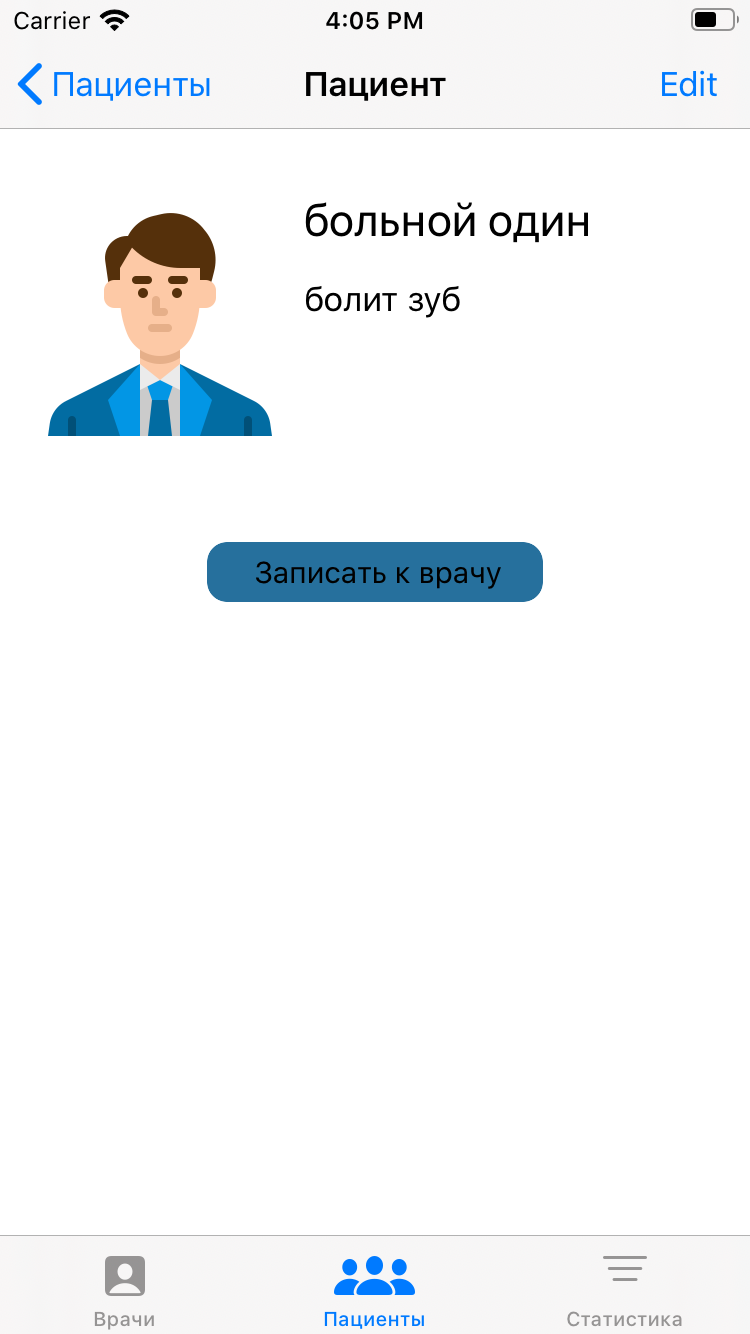


Иллюстрация интерфейса класса DetailPersonPresenter

# 3.3 Реализация классов

**DoctorsPresenter**– управляющий класс, который хранит данные о врачах.

Данный класс содержит данные свойства:

- doctorSingelton: DoctorsSingelton – ссылка на класс DoctorsSingelton

- stateData:StateData – объект перечисления StateData

- findData:[PeopleModel] – найденные данные после поиска

**PatientPresenter**– управляющий класс, который хранит данные о пациентах.

Данный класс содержит данные свойства:

- patientSingelton: PatientSingelton– ссылка на класс PatientSingelton

- stateData:StateData – объект перечисления StateData

- findData:[PeopleModel] – найденные данные после поиска

**StatisticPresenter**– управляющий класс, который хранит данные о статистике.

Данный класс содержит данные свойства:

- statisticData: StatisticSingelton– ссылка на класс PatientSingelton

- presentData: presentDATA – объект перечисления presentDATA

- enum presentDATA – перечисление, которое содержит 2 поля: rayon, day

**PickerPresenter**– управляющий класс, который хранит данные о выбранном районе.

Данный класс содержит данные свойства:

- data: StatisticSingelton– ссылка на класс StatisticSingelton

- active: Int – свойство, которое отвечает за индекс выбранного района

**EditPatientPresenter**– управляющий класс, который хранит данные, которые необходимы при редактировании профиля пациента.

Данный класс содержит данные свойства:

- data: PatientSingelton– ссылка на класс PatientSingelton

- active: Int – свойство, которое хранит id текущего пациента

**EditDoctorsPresenter**– управляющий класс, который хранит данные, которые необходимы при редактировании профиля доктора.

Данный класс содержит данные свойства:

- data: DoctorsSingelton– ссылка на класс DoctorsSingelton

- active: Int – свойство, которое хранит id текущего доктора

**AddNewPeoplePresenter**– управляющий класс, который хранит данные, которые необходимы при добавлении нового пользователя.

Данный класс содержит данные свойства:

- doctorSingelton: DoctorsSingelton– ссылка на класс DoctorsSingelton

- patientSingelton: PatientSingelton– ссылка на класс PatientSingelton

- stateAddPeople: StateAddPeople – значение перечисления StateAddPeople, которое отвечает з то, кого мы добавляем (врача или пациента)

**WritePatientToDoctorPresenter**– управляющий класс, который хранит данные, которые необходимы при записи пациента к врачу.

Данный класс содержит данные свойства:

- indexTapPeople: Int – id выбранного пользователя

- doctors: DoctorsSingelton – ссылка на класс DoctorsSingelton

- patient: PatientSingelton– ссылка на класс PatientSingelton

**DetailDoctorsPresenter**– управляющий класс, который хранит данные о враче.

Данный класс содержит данные свойства:

- index: Int – id выбранного врача

- data: DoctorsSingelton – ссылка на класс DoctorsSingelton

- patient: PatientSingelton– ссылка на класс PatientSingelton

**DetailPersonPresenter**– класс, который хранит данные о пациенте.

Данный класс содержит данные свойства:

- index: Int – id выбранного врача

- data: PatientSingelton– ссылка на класс PatientSingelton

**DoctorsSingelton**– управляющий класс, который реализует паттерн одиночка.

Данный класс содержит данные свойства:

- data: [PeopleModel] – данные о докторах

- share: DoctorsSingelton – ссылка на класс DoctorsSingelton

Данный класс содержит данные методы:

- private init() – приватный конструктор

**PatientSingelton**– управляющий класс, который реализует паттерн одиночка.

Данный класс содержит данные свойства:

- data: [PeopleModel] – данные о пацинтах

- share: PatientSingelton – ссылка на класс PatientSingelton

Данный класс содержит данные методы:

- private init() – приватный конструктор

**StatisticSingelton**– управляющий класс, который реализует паттерн одиночка.

Данный класс содержит данные свойства:

- pickerData: [String] – список районов

- count: [Int] = [0,0,0,0,0,0] – счетчик кол-ва записей в каждый район

- weekday: [Int] = [0,0,0,0,0,0,0]- счетчик кол-ва записей по дням недели

- day: [String] – список дней недели

- share: PatientSingelton – ссылка на класс StatisticSingelton

Данный класс содержит данные методы:

- private init() – приватный конструктор

**SaveDoctors** –класс, который реализует сохранение данных для докторов.

Данный класс содержит данные методы:

- saveData – метод для сохранения данных

- saveDataHistory – метод для сохранения истории посещений

**SavePatient** –класс, который реализует сохранение данных для пациентов.

Данный класс содержит данные методы:

- saveData – метод для сохранения данных

**SaveStatistic**–класс, который реализует сохранение данных для статистики.

Данный класс содержит данные методы:

- saveData – метод для сохранения данных

**LoadStatistic**–класс, который реализует загрузку данных для статистики.

Данный класс содержит данные методы:

- loadData– метод для загрузки данных

**LoadPatient**–класс, который реализует загрузку данных для пациентов.

Данный класс содержит данные методы:

- loadData– метод для загрузки данных

**LoadDoctors**–класс, который реализует загрузку данных для докторов.

Данный класс содержит данные методы:

- loadData– метод для загрузки данных

- loadDataHistory – метод для загрузки историй посещения врачей

**DoctorModel**–структура, которая хранит данные врачей.

Данный класс содержит данные свойства:

- name: String – имя врача

- soname: String – фамилия врача

- profession: String – профессия врача

- image: UIImage – изображение врача

**PatientModel**–структура, которая хранит данные пациентов.

Данный класс содержит данные свойства:

- name: String – имя пользователя

- soname: String – фамилия пользователя

- problem: String – проблема пользователя

- image: UIImage – изображение пользователя

- data: Date - дата посещения врача

**StatisticModel**–структура, которая хранит данные для статистики.

Данный класс содержит данные свойства:

- pickerData: [String] – массив, который хранит название районов

- count: [Int] – массив, который хранит количество обращений в каждый район

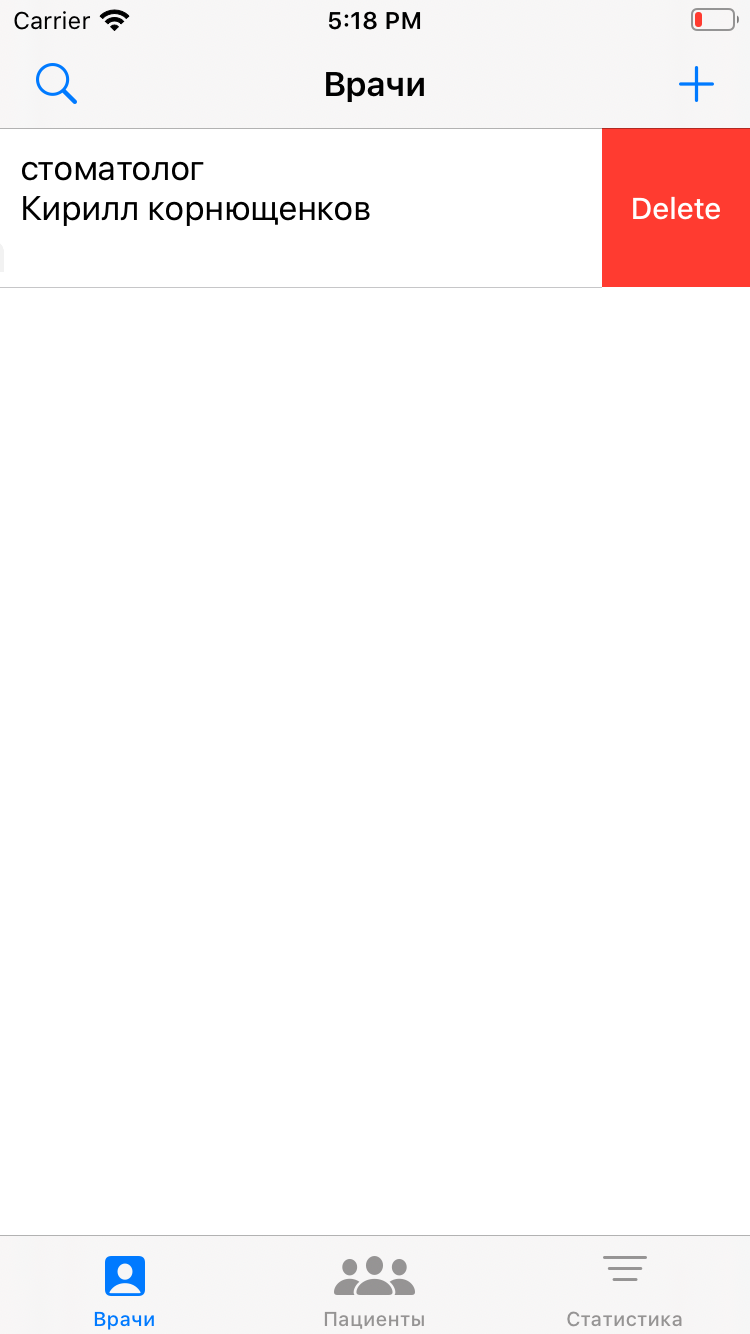
- weekday: [Int] – массив, который хранит количество обращений по дням недели

- day: [String] – массив, который хранит название дней недели

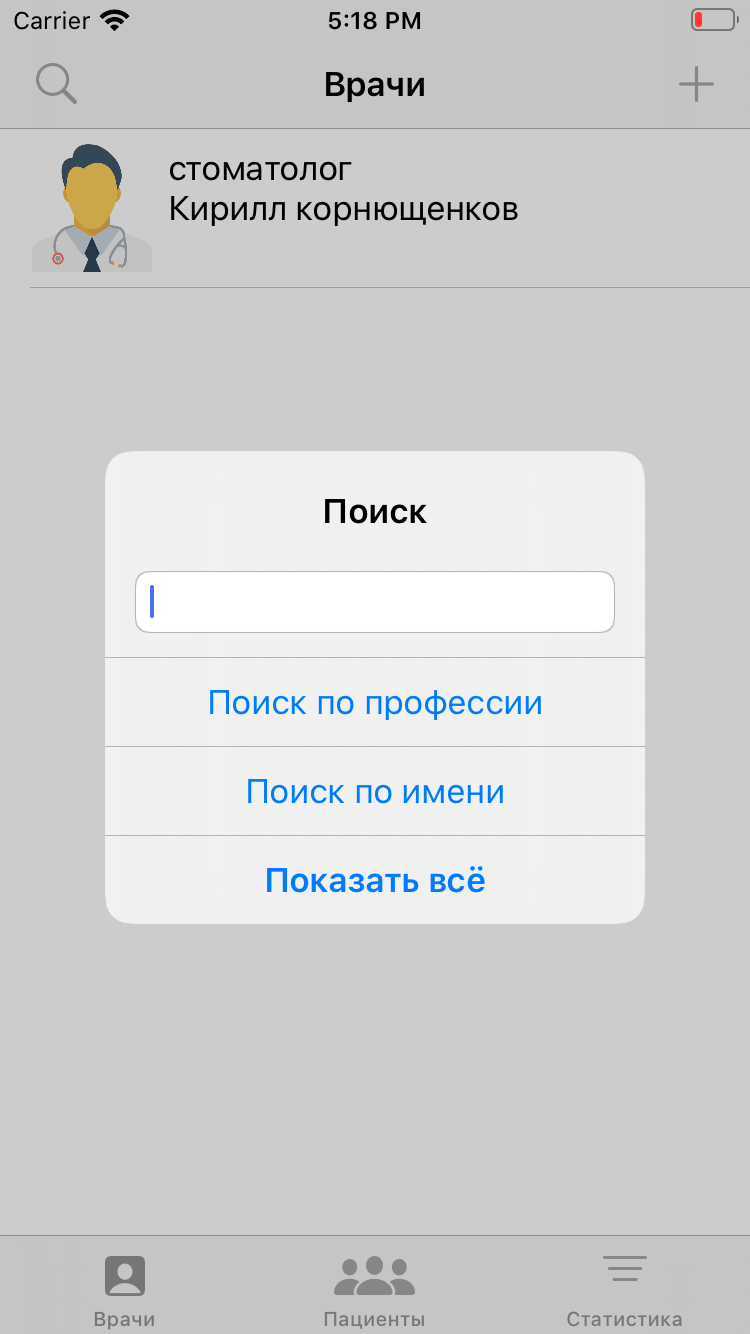
# 4 Тестирование



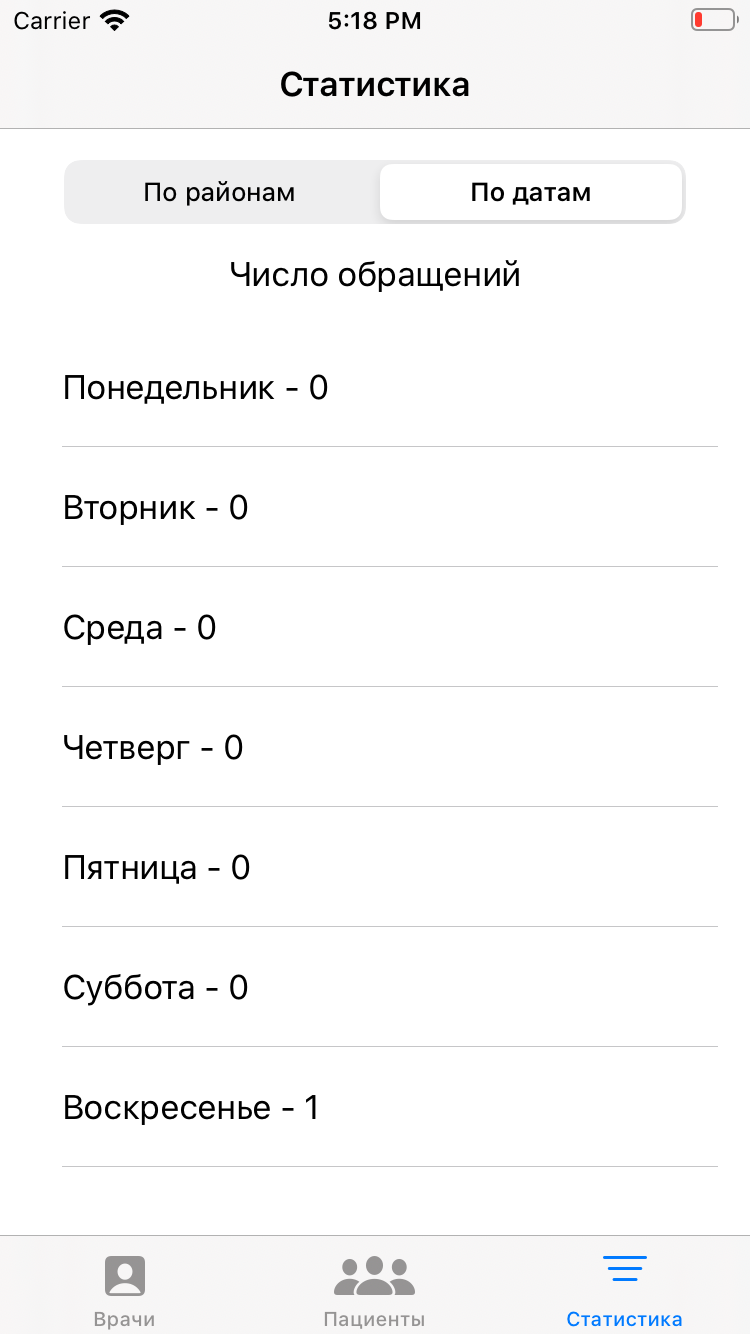
Стартовый экран приложения, на котором изображен список всех врачей



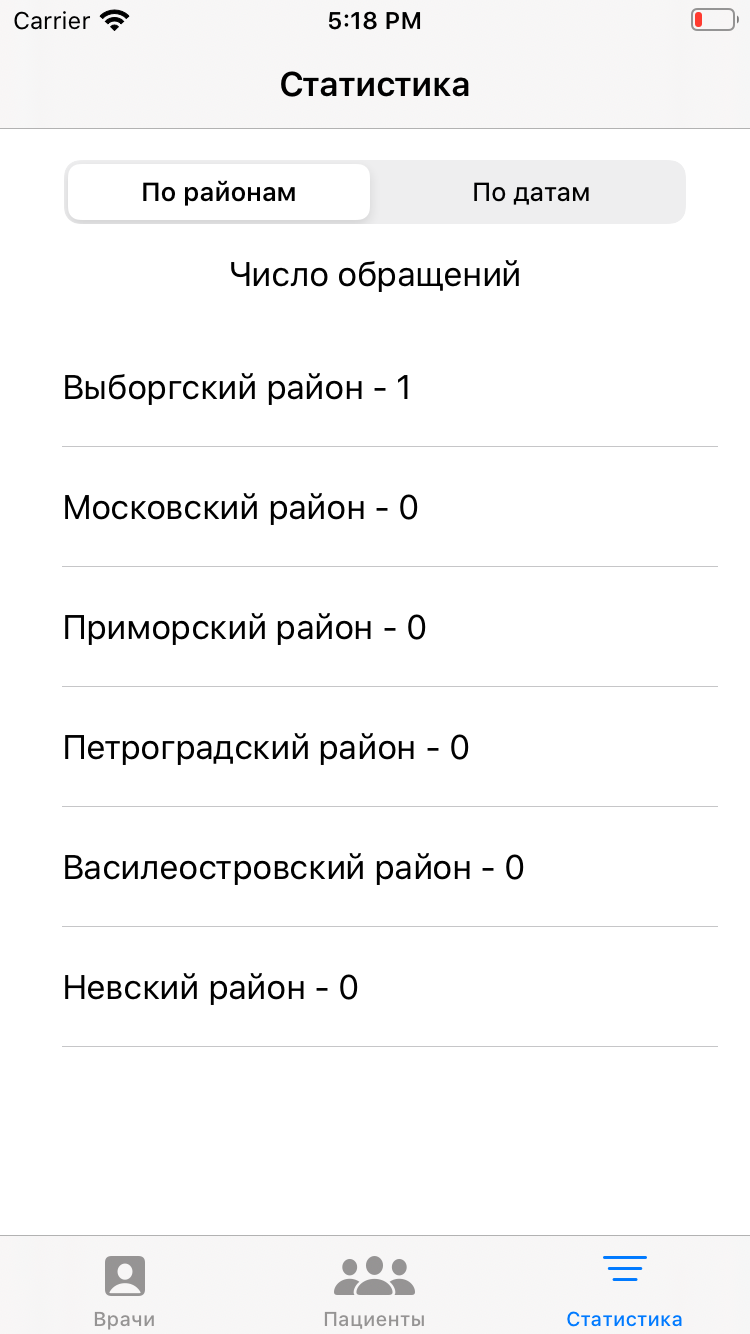
При свайпе влево появляется возможность удалить врача из нашей базы данных



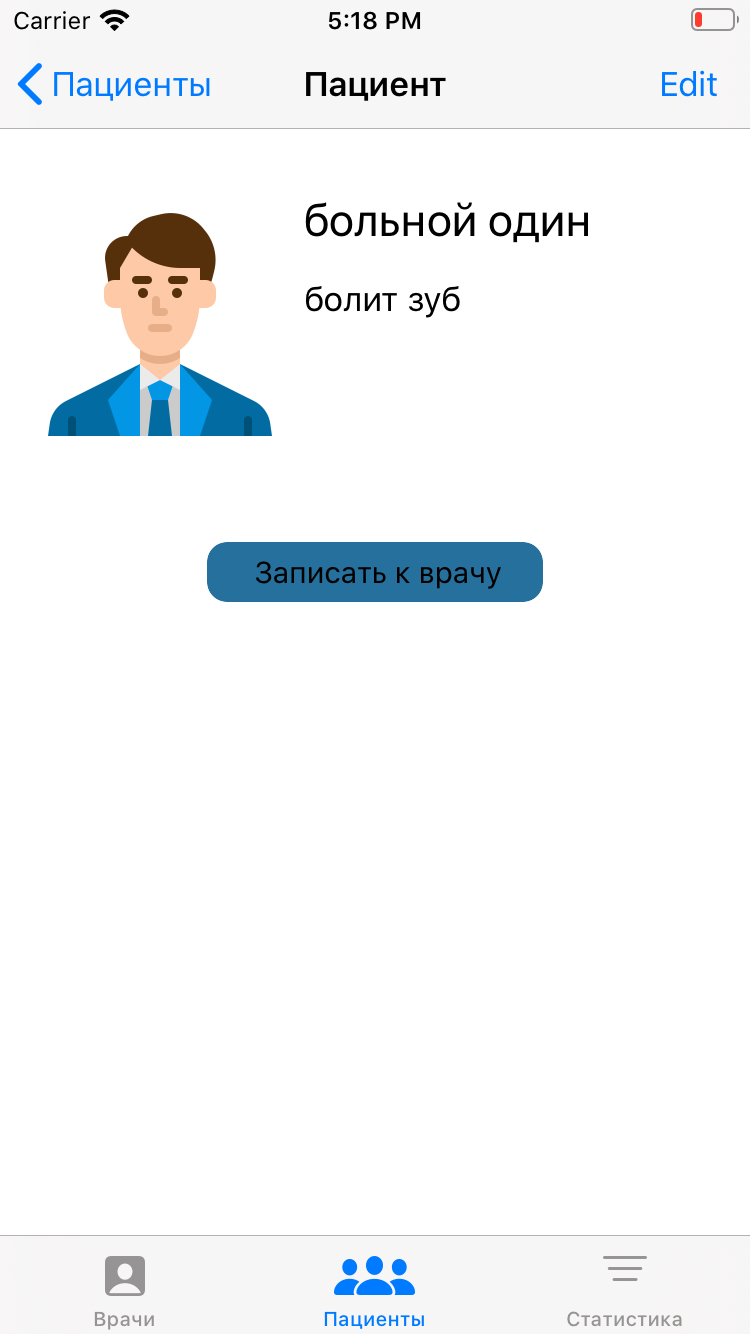
При нажатии на кнопку поиска всплывает Alert, где можно указать искомую информацию и выбрать по какому полю будет осуществляться поиск или показать всё информацию



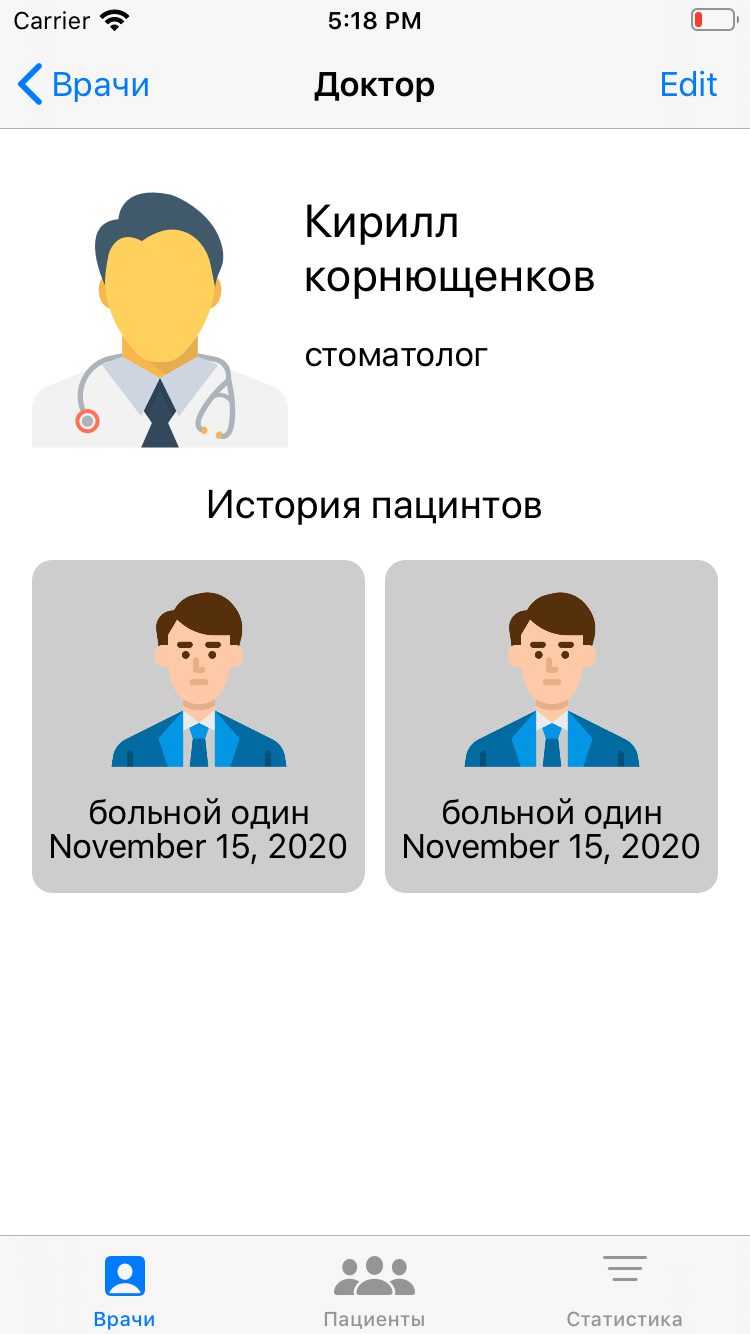
При переходе на экран статистике можно увидеть количество записей в каждый день недели



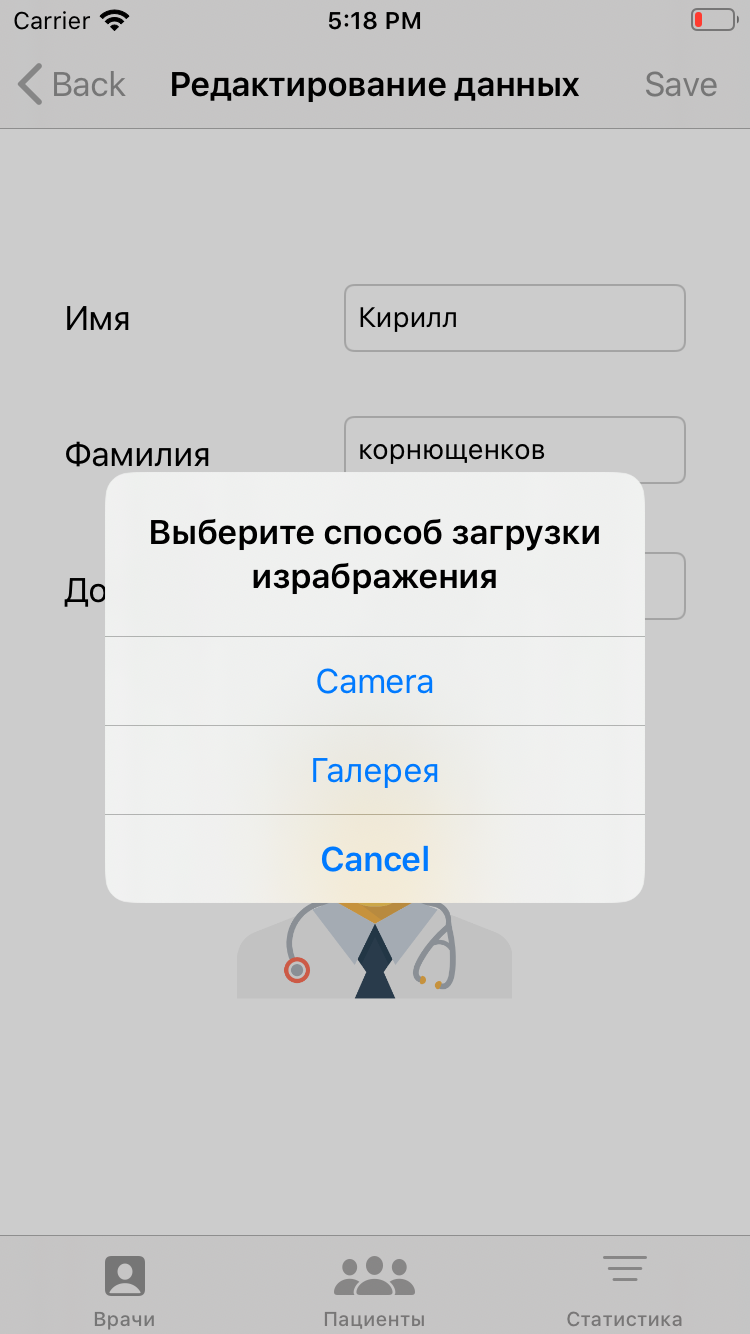
При переключении на пункт “по районам” отображается количество записей в каждом район



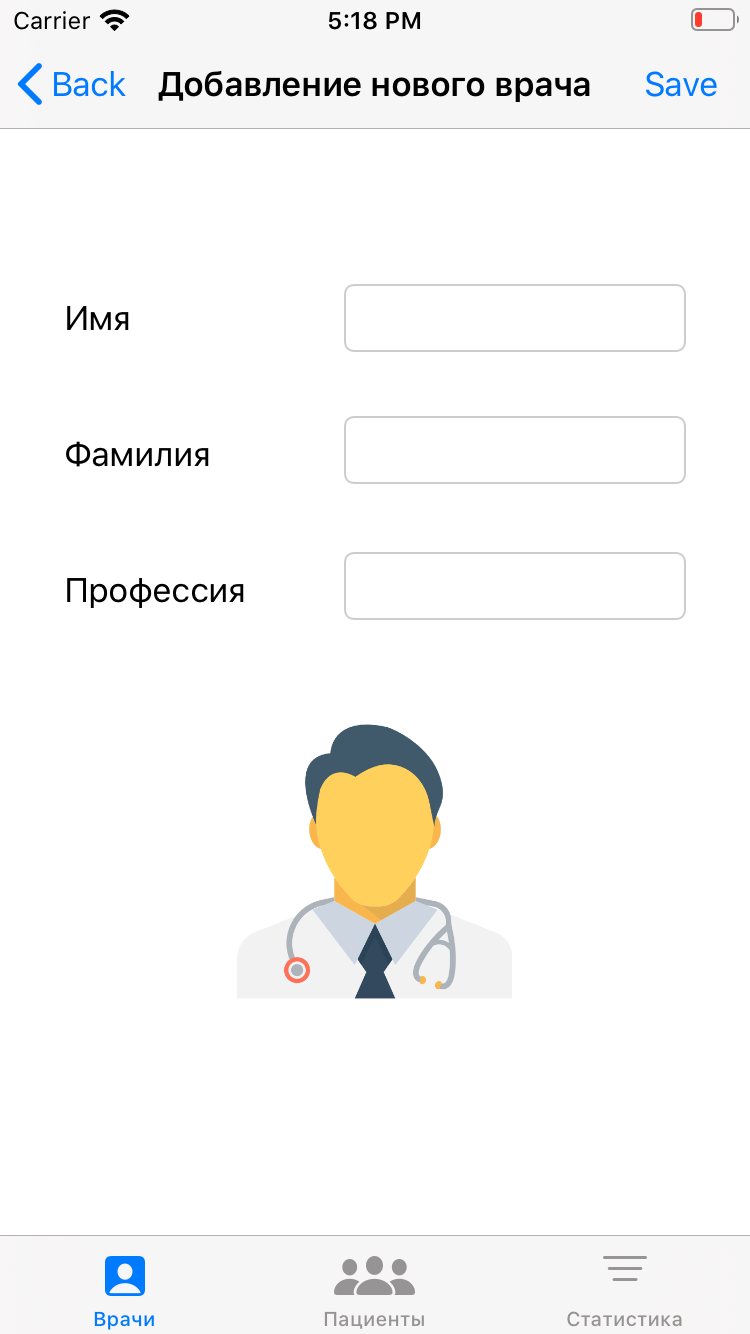
При переходе на экран Пациенты и нажатии на пациента открывается экран с детальной информацией



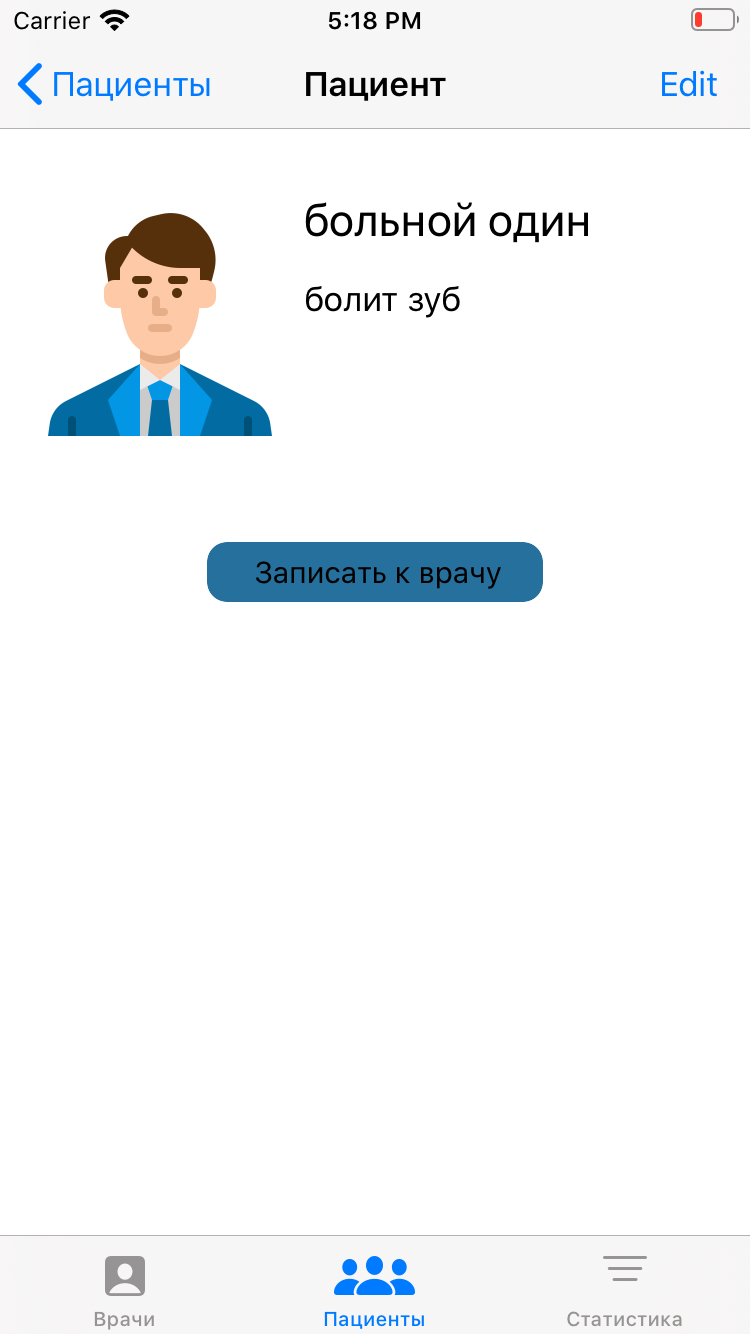
Аналогичный экран есть для врачей, где отображается детальная информация и история пациентов, которые посетили данного врача.



При редактировании данных пользователя/врача можно нажать на изображения и выбрать пункт с помощью чего хотят загрузить новую фотографию



Так же есть экран добавления нового врача/пользователя, где можно ввести данные и сохранить их. Если какое-то из 3 полей пустое, то система не даст нажать кнопку Сохранить



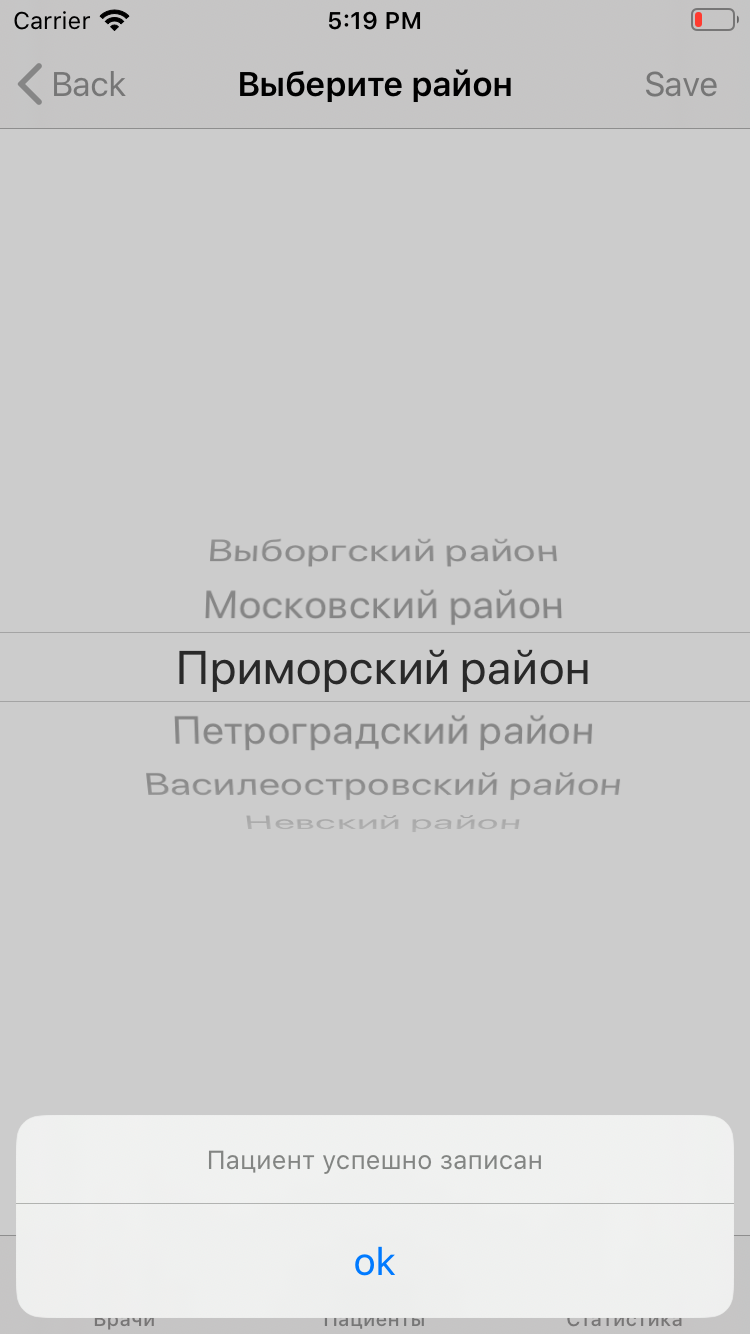
На экране детальной информации о пациенте есть кнопка записи к врачу



При нажатии на кнопку Запись к врачу открывается экран, где можно выбрать нужного врача



Затем необходимо выбрать район для записи



При нажатии на кнопку Save система уведомляет нас об успешной записи пользователя к врачу

# 

# Заключение

В результате конечный пользователь получает готовую информационную систему для обеспечения работы поликлиники. Данное приложение написано на языке Swift, с использование библиотеки UIKit для написания и создания графического интерфейса.

К достоинствам приложения можно отнести простой и понятный интерфейс.

К недостаткам слабая переиспользуемость компонентов и не лучший выбор организации хранения данных из-за чего могут возникнуть проблемы при масштабируемости проекта.

# Список использованных источников

1. E.O. Шумова Методические указания к выполнению курсового проекта — СПб ГУАП, 2020
2. Swift. Основы разработки приложений под iOS и macOS | Усов Василий
3. Кнут Д.Е. Искусство программирования. В 3-х томах. М."Вильямс", 2000.