**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ**

**РОССИЙСКОЙ ФЕДЕРАЦИИ**

федеральное государственное автономное образовательное

учреждение высшего образования

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**

**ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

Инженерная школа информационных технологий и робототехники

Отделение информационных технологий

Направление «Информатика и вычислительная техника»

Курсовая работа по дисциплине

**«ТЕХНОЛОГИИ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ»**

Веб-сервис для записи к врачу

Выполнил:

Студент группы 8ВМ03 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.В. Кавешников

Проверил:

Доцент ОИТ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.Н. Поляков

Томск 2022

# Содержание

[Содержание 2](#_Toc33597905)

[Введение 3](#_Toc33597906)

[1 Требования к программе 4](#_Toc33597907)

[1.1 Введение 4](#_Toc33597908)

[1.2 Основания для разработки 4](#_Toc33597909)

[1.3 Назначение разработки 4](#_Toc33597910)

[1.4 Требования к программе или программному изделию 4](#_Toc33597911)

[1.4.1 Требования к функциональным характеристикам 4](#_Toc33597912)

[1.4.2 Требования к надежности 5](#_Toc33597913)

[1.4.3 Условия эксплуатации 5](#_Toc33597914)

[1.4.4 Требования к составу и параметрам технических средств 5](#_Toc33597915)

[1.4.5 Требования к информационной и программной совместимости 5](#_Toc33597916)

[1.4.6 Требования к маркировке и упаковке 5](#_Toc33597917)

[1.4.7 Требования к транспортированию и хранению 5](#_Toc33597918)

[1.5 Требования к программной документации 5](#_Toc33597919)

[1.6 Технико-экономические показатели 6](#_Toc33597920)

[1.7 Стадии и этапы разработки 6](#_Toc33597921)

[1.7.1 Стадии разработки 6](#_Toc33597922)

[1.7.2 Этапы разработки 6](#_Toc33597923)

[1.7.3 Содержание работ по этапам 6](#_Toc33597924)

[1.8 Порядок контроля и приемки 7](#_Toc33597925)

[1.8.1 Виды испытаний 7](#_Toc33597926)

[1.8.2 Общие требования к приемке работы 7](#_Toc33597927)

[1.9 Варианты использования 7](#_Toc33597928)

[1.9.1 Сценарии 8](#_Toc33597929)

[2 Анализ 10](#_Toc33597930)

[3 Проектирование 11](#_Toc33597931)

[4 Реализация 12](#_Toc33597932)

[4.1 Тестирование 12](#_Toc33597933)

[4.2 Непрерывная интеграция 12](#_Toc33597934)

[4.3 Развёртывание 12](#_Toc33597935)

[5 Документация 13](#_Toc33597936)

[Заключение 14](#_Toc33597937)

[Список использованных источников 15](#_Toc33597938)

# Введение

Лёгкая атлетика – олимпийский вид спорта, включающий бег, ходьбу, прыжки и метания. Объединяет следующие дисциплины: беговые виды, спортивную ходьбу, технические виды (прыжки и метания), многоборья, пробеги (бег по шоссе) и кроссы (бег по пересечённой местности). Один из основных и наиболее массовых видов спорта. Легкую атлетику называют королевой спорта. [1]

Рост спортивных результатов во многих легкоатлетических дисциплинах объясняется не только постоянным совершенствованием методики подготовки и ее научного обеспечения, но и достижениями научно-технического прогресса, благодаря которому появляется все более эффективный спортивный инвентарь – шесты, копья и др., покрытия беговых дорожек и секторов для прыжков и метаний, спортивная обувь и др. [2]

Рассмотрим то, как проходят соревнования по лёгкой атлетике:

1. Прибытие спортсменов на место проведения и переодевание.
2. Парад участников. Сначала участники выстраиваются в две шеренги и идут с табличками своей организации. Затем главный судья командует "стоп" и "направо/налево", спортсмены поворачиваются к нему лицом.
3. Награждение заслуженных тренеров и спортсменов, выполнивших высокий разряд на всероссийских соревнованиях.
4. Звучит гимн России и равнение на флаг (если он есть).
5. Разминка. Она длиться длится 40-50 мин и включает в себя беговые упражнения и растяжку.
6. Пробные забеги. Участники бегут дистанция, на которую заявились.
7. Далее спортсмен подходит к судье и получает номер забега и дорожки.
8. Ожидание забега
9. Когда стартер (судья с пистолетом) говорит «снять тренировочные костюмы и встать по дорожкам» бегун идет на свою позицию. Там он может настроить колодки (приспособления для низкого старта) и сделать пару выходов с них.
10. После команды «на старт» спортсмены встают на свои колодки не шевелясь. На команду «внимание» чуть приподнимают корпус вверх. На выстрел из пистолета бегут до финиша.
11. На финише судья смотрят кто под каким номером бежал и записывают его результат.
12. Потом все расходятся, надевают костюмы и ждут награждения или финального забега.

В качестве языка разработки используется Python 3.7, так как данный язык входит в топ-3 по популярности в 2019 году согласно рейтингам TOIBE, PYPL, GitHub, StackOverflow, RedMonk. [3] В качестве средства разработки выбран PyCharm Community Edition 2020.2.3. Это интегрированная среда разработки для языка программирования Python от компании JetBrains на основе IntelliJ IDEA. [4] PyCharm предоставляет умную проверку кода, быстрое выявление ошибок и оперативное исправление, вкупе с автоматическим рефакторингом кода, и богатыми возможностями в навигации. [5] Также используются пакеты Django (веб-сервис) и sqlite3 (связь с БД).

Системой контроля версий выступает Git, а в качестве веб-хостинга выбран GitHub (ссылка на проект – <https://github.com/BossOfCreeps/Software-Development-Technologies>). GitHub является самой популярный сервисом для хранения кода. [6].

Для размещения сервиса используется виртуальная машина на Google Cloud. Для хранения информации используется база данных (БД). Выбрана SQLite3, размещенная так же на этом сервере.

# Требования к программе

## Введение

Наименование: «Веб-сервис для записи на прием к врачу». Программа будет применяться для записи на прием к врачу больницы и просмотра информации о встречах.

## Основания для разработки

Основаниями для разработки является «Требования к сдачи курсового проекта по дисциплине “Технологии разработки программного обеспечения”», утверждённой Национальным исследовательским Томским Политехническим университетом.

Наименование темы разработки – «Веб-сервис для записи на прием к врачу». Условное обозначение темы разработки (шифр темы) отсутствует.

## Назначение разработки

Программа должна облегчить получение информации участникам соревнований:

* Пациентам – облегчить процедуру записи на прием к врачу и получение информации о последующих встречах
* Врачам – получение расписания предстоящих встреч

## Требования к программе или программному изделию

### Требования к функциональным характеристикам

#### Требования к составу выполняемых функций

Программа должна обеспечивать возможность выполнения перечисленных ниже функций:

* Функция регистрации пациента пользователем
* Функция авторизации
* Функция занесении информации пациентом о встрече с врачом
* Функция просмотра пациентом информации о предстоящих встречах
* Функция просмотра врачом информации о предстоящих встречах
* Функция создания пользователя пациента/врача администратором
* Функция просмотра списка пользователей администратором
* Функция удаления пользователей администратором
* Функция просмотра актуальных текущих встреч администратором
* Функция просмотра прошедших встреч администратором
* Функция удаления записей о встречах

#### Требования к организации входных и выходных данных

Входные данные программы должны быть логичными при занесении (например, не вписывать имя вместо времени). Данные могут поступать на русском, английских языках, а также поддерживаются числа и даты там, где это необходимо (время записи, дата записи). Данные не должны быть пустыми. Нельзя осуществить запись к врачу на уже прошедшую дату и время.

#### Требования к временным характеристикам

Требования к временным характеристикам программы не предъявляются.

### Требования к надежности

Требования к обеспечению надежного (устойчивого) функционирования программы не предъявляются.

### Условия эксплуатации

Требования к климатическим условиям не предъявляется. Программа не требует проведения каких-либо видов обслуживания. Отсутствует необходимость в персонале для работы.

### Требования к составу и параметрам технических средств

Необходимо устройство с доступом Интернет и браузером с поддержкой HTML.

### Требования к информационной и программной совместимости

#### Требования к информационным структурам и методам решения

Требования к информационным структурам (файлов) на входе и выходе, а также к методам решения не предъявляются.

#### Требования к исходным кодам и языкам программирования

Исходные коды программы должны быть реализованы на языке Python 3.7. В качестве интегрированной среды разработки программы должна быть использована среда PyCharm Community Edition 2020.2.3

#### Требования к программным средствам, используемым программой

Системные программные средства, используемые программой, должны быть представлены лицензионной локализованной версией операционной системой Ubuntu 16.04 с установленными БД sqlite3 и средой python3 (версия 3.7), а также пакетами Django, sqlite3.

#### Требования к защите информации и программ

Требования к защите информации и программ не предъявляются.

### Требования к маркировке и упаковке

Требования к маркировке и упаковке не предъявляются.

### Требования к транспортированию и хранению

Требования к транспортированию и хранению не предъявляются.

## Требования к программной документации

В состав программной документации должны входить:

* Техническое задание
* Программа и методика испытаний
* Руководство системного программиста

## Технико-экономические показатели

Ориентировочная экономическая эффективность не рассчитывается.

## Стадии и этапы разработки

### Стадии разработки

Разработка должна быть проведена в три стадии:

* техническое задание;
* технический (и рабочий) проекты;
* внедрение.

### Этапы разработки

На стадии «Техническое задание» должен быть выполнен этап разработки, согласования и утверждения настоящего технического задания.

На стадии «Технический (и рабочий) проект» должны быть выполнены перечисленные ниже этапы работ:

* разработка программы;
* разработка программной документации;
* испытания программы.

На стадии «Внедрение» должен быть выполнен этап разработки «Подготовка и передача программы».

### Содержание работ по этапам

На этапе разработки техзадания должны быть выполнены перечисленные ниже работы:

* постановка задачи;
* определение и уточнение требований к техническим средствам;
* определение требований к программе;
* определение стадий, этапов и сроков разработки программы и документации на нее;
* выбор языков программирования;
* согласование и утверждение технического задания.

На этапе разработки программы должна быть выполнена работа по программированию (кодированию) и отладке программы.

На этапе разработки программной документации должна быть выполнена разработка программных документов в соответствии с требованиями ГОСТ 19.101-77.

На этапе испытаний программы должны быть выполнены перечисленные ниже виды работ:

* разработка, согласование и утверждение программы и методики испытаний;
* проведение приемо-сдаточных испытаний;
* корректировка программы и программной документации по результатам испытаний.

На этапе подготовки и передачи программы должна быть выполнена работа по подготовке и передаче программы и программной документации в эксплуатацию на объектах заказчика.

## Порядок контроля и приемки

### Виды испытаний

Приемосдаточные испытания должны проводиться на объекте заказчика в срок до 01.06.2022.

Ход проведения приемо-сдаточных испытаний заказчик и исполнитель документируют в протоколе испытаний.

### Общие требования к приемке работы

На основании протокола испытаний исполнитель совместно с заказчиком подписывают акт приемки-сдачи программы в эксплуатацию.

## Варианты использования

Модель вариантов использования служит для того, чтобы достигнуть взаимопонимания между разработчиками и заказчиками по назначению, возможностям и технологии использования будущей информационной системы, т. е. определить границы ее применения.

Так как заказчик принимает активное участие в построении этой модели, она должна быть описана на его языке, т. е. с употреблением терминологии, принятой в рассматриваемой предметной области. Кроме того, для повышения взаимопонимания рекомендуется в данной модели не отражать механизма реализации функций. Ведь эта специфическая информация лишь отвлекает внимание заказчика от главной цели – формирования требований.

Построение этой модели необходимо для выявления:

* актеров, взаимодействующих с системой;
* вариантов использования системы;
* нефункциональных требований.

Диаграмма вариантов использования является исходным концептуальным представлением системы в процессе ее проектирования и разработки. Данная диаграмма состоит из актеров, вариантов использования и отношений между ними. При построении диаграммы могут использоваться также общие элементы нотации: примечания и механизмы расширения.

Суть данной диаграммы состоит в следующем: проектируемая система представляется в виде множества актеров, взаимодействующих с системой с помощью так называемых вариантов использования. При этом актером называется любой объект, субъект или система, взаимодействующая с моделируемой системой извне. В свою очередь вариант использования – это спецификация сервисов (функций), которые система предоставляет актеру. Другими словами, каждый вариант использования определяет некоторый набор действий, совершаемых системой при взаимодействии с актером. При этом в модели никак не отражается то, каким образом будет реализован этот набор действий.

В проектируемой системе можно выделить следующих актеров:

* гость;
* пользователь(пациент);
* пользователь(врач);
* администатор;

Для гостя доступны следующие варианты использования системы:

* регистрация(пациента);
* авторизация;
* просмотр контактной информации;

Для пациента доступны варианты использования:

* создание записи к врачу;
* просмотр информации об актуальных записях к врачу;

Для врача доступны следующие варианты использования:

* просмотр информации об актуальных встречах с пациентами;

Для администратора доступны следующие варианты использования:

* регистрация пациента или врача;
* просмотр списка пользователей;
* удаление пользователя;
* просмотр всех актуальных записей к врачу;
* просмотр прошедших записей к врачу;
* удаление записей к врачу;

Цель построения диаграммы – документирование требований к системе в общем в виде.

Правила построения:

1. Не моделировать взаимодействие между актёрами
2. Не соединять два варианта использования
3. Каждый вариант использования должен быть инициирован актёром

Основные элементы [7]:

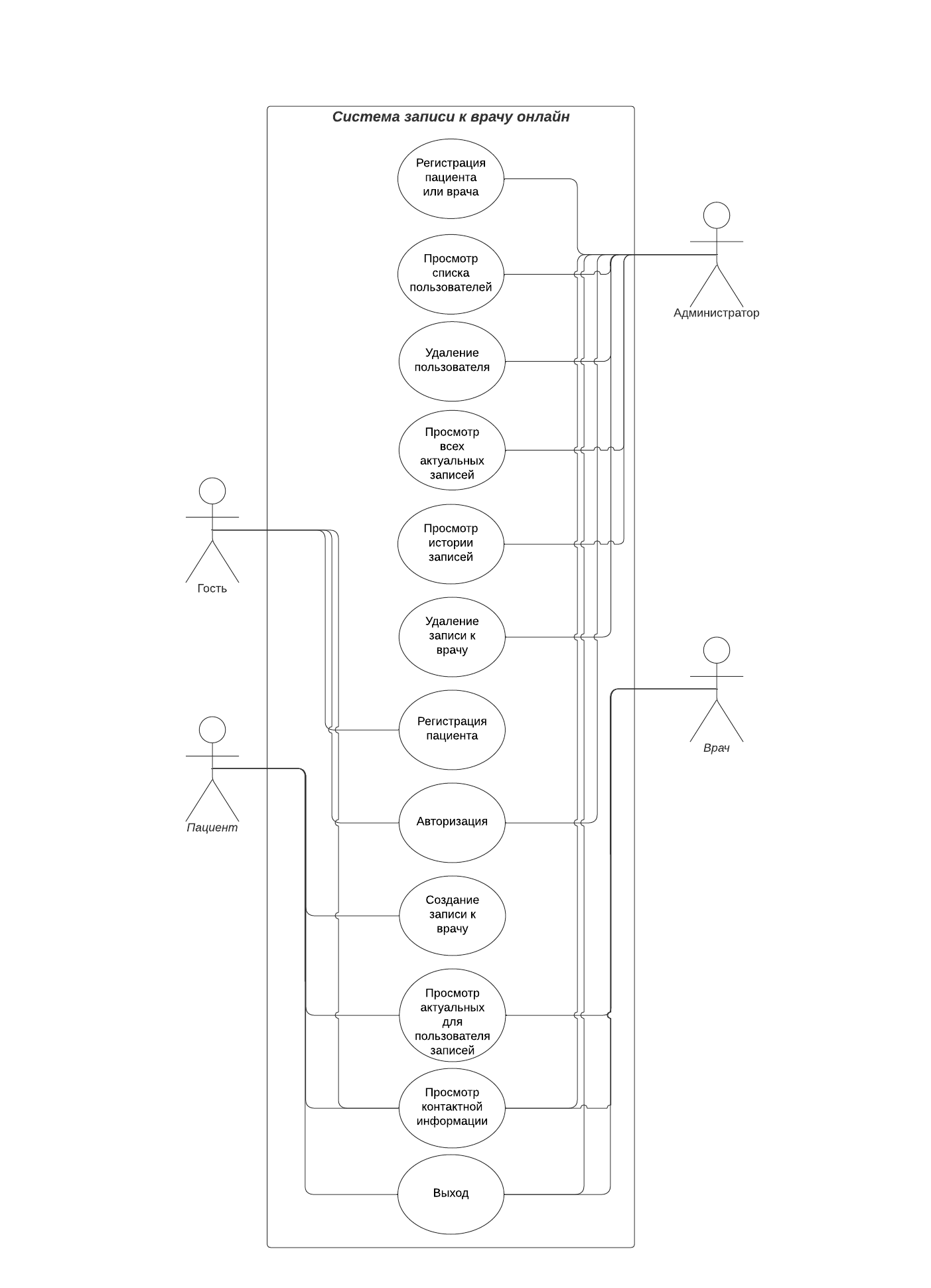
* Имя – наименование, чтобы четко и однозначно понимать назначение
* Краткое описание – цель
* Актёры – участвующие лица:
  + Главные актёры – инициируют
  + Второстепенные актёры – взаимодействуют с ВИ во время исполнения
* Предусловия – условия, которые должны быть выполнены до начала ВИ; ограничения на состояние системы
* Основной поток – диалог актёра и системы
* Альтернативные потоки – обрабатывают ошибки, ветвления, прерывания, исключения основного потока
* Постусловия – состояние системы после окончания ВИ

Рассмотрим варианты использования данной системы в таблице 1.

1. Варианты использования системы

|  |  |
| --- | --- |
| Имя | Регистрация пользователя (пациента) |
| Краткое описание | Неавторизованный пользователь зайдя на страницу регистрации, заполняет форму регистрации и отправляет данные на сервер |
| Актёры | Гость |
| Предусловие | Неавторизованный пользователь |
| Основной поток | Неавторизованный пользователь (гость) зайдя на первую страницу авторизации переходит на страницу регистрации где осуществляет заполнения формы и отправку её данных |
| Альтернативный поток | При неправильной регистрации (введение невалидных либо уже существующих данных) система сообщает об ошибке |
| Постусловия | В базу данных добавляется запись о новом пациенте |
|  |  |
| Имя | Авторизация пользователя |
| Краткое описание | Неавторизованный пользователь (гость) осуществляет процесс авторизации |
| Актёры | Гость |
| Предусловие | Неавторизованный пользователь |
| Основной поток | Гость заходит на страницу авторизации, вводит в веб форму логин и пароль, после чего отправляет данные формы на сервер, где происходит проверка данных и если данные верны, то происходит успешная авторизация |
| Альтернативный поток | В случае ввода некорректных данных (неверный логин или пароль) система сообщит об ошибке |
| Постусловия | Авторизованный пользователь |
|  |  |
| Имя | Занесение информации о записи к врачу |
| Краткое описание | Пациент через веб-форму заполняет данные о предстоящей встрече |
| Актёры | Пациент |
| Предусловие | Пользователь авторизованный как пациент |
| Основной поток | Пациент выбирает профиль специалиста, дату и время встречи |
| Альтернативный поток | При вводе неправильных данных (у специалиста заняты данные дата и время) система сообщает об ошибке |
| Постусловия | База данных с новой записью о встречах, На персональной странице пациента/врача показан список актуальных для него встреч |
|  | |
| Имя | Получение информации о предстоящих встречах |
| Краткое описание | Пациенты и врачи, зайдя на персональную страницу, могут получить информацию о предстоящих встречах |
| Актёры | Пациент, врач |
| Предусловие | Пользователь авторизованный как пациент либо как врач |
| Основной поток | Пациент или врач вводит данные для авторизации и переходит на персональную страницу. На странице он получает информацию о предстоящих встречах |
| Альтернативный поток | При отсутствии удовлетворительных записей в БД система выдаст пустую страницу |
| Постусловия | На персональной странице пациента/врача показана информация о встречах |
|  |  |
| Имя | Регистрация пользователя (врача) |
| Краткое описание | Администратор на административной странице осуществляет процесс регистрации врача |
| Актёры | Администратор |
| Предусловие | Пользователь, авторизованный как администратор |
| Основной поток | Администратор на административной странице заполняет данные веб формы (логин, пароль, имя, врачебный профиль) |
| Альтернативный поток | При неправильной регистрации (введение невалидных либо уже существующих данных) система сообщает об ошибке |
| Постусловия | В базу данных добавляется запись о новом враче |
|  |  |
| Имя | Просмотр списка пользователей |
| Краткое описание | Администратор, зайдя на административную страницу, просматривает таблицу, содержащую список пользователей |
| Актёры | Администратор |
| Предусловие | Пользователь, авторизованный как администратор |
| Основной поток | Администратор, зайдя на административную страницу, просматривает таблицу, содержащую список пользователей |
| Альтернативный поток | При отсутствии пользователей Администратор видит пустую таблицу |
| Постусловия | На административной странице информация о пользователях |
|  |  |
| Имя | Удаления пользователя |
| Краткое описание | Администратор осуществляет удаления пользователя |
| Актёры | Администратор |
| Предусловие | Пользователь, авторизованный как администратор, наличие пользователей в соответствующей таблице БД |
| Основной поток | Администратор, зайдя на административную страницу, просматривает таблицу, содержащую список пользователей и в этом списке выбирает пользователя для удаления, нажимая кнопку “x” в списке на соответствующей пользователю строке |
| Альтернативный поток | При отсутствии пользователей либо отображении неактуальной информации о пользователях при попытке удаления пользователя будет показана ошибка |
| Постусловия | База данных с удаленной записью о пользователе |
|  |  |
| Имя | Просмотр списка втреч |
| Краткое описание | Администратор, зайдя на административную страницу, просматривает таблицу, содержащую список встреч пациентом с врачами |
| Актёры | Администратор |
| Предусловие | Пользователь, авторизованный как администратор |
| Основной поток | Администратор, зайдя на административную страницу, просматривает таблицу, содержащую список из актуальных и прошедших встреч пациентом с врачами |
| Альтернативный поток | При отсутствии актуальных встреч Администратор видит пустую таблицу с текущими встречами |
| Постусловия | Административная страница с информацией о встречах |
|  |  |
| Имя | Удаления встречи |
| Краткое описание | Администратор осуществляет удаления встречи |
| Актёры | Администратор |
| Предусловие | Пользователь, авторизованный как администратор, наличие встреч в соответствующей таблице БД |
| Основной поток | Администратор, зайдя на административную страницу, просматривает таблицу, содержащую список встреч и в этом списке выбирает встречу для удаления, нажимая кнопку “x” в списке на соответствующей встрече строке |
| Альтернативный поток | При отсутствии встреч либо отображении неактуальной информации о пользователях при попытке удаления пользователя будет показана ошибка |
| Постусловия | База данных с удаленной записью о встрече |
|  |  |
| Имя | Просмотр контактной информации |
| Краткое описание | Просмотр контактной информации о предприятии |
| Актёры | Гость, пациент, врач, Администатор |
| Предусловие | Рабочая система |
| Основной поток | Любой пользователь с любой страницы может перейти по вкладке в верхне-левом углу страницы на страницу, содержащую контактную информацию о предприятии |
| Альтернативный поток | отсутствует |
| Постусловия | Страница с контактной информацией |
|  |  |
| Имя | Выход |
| Краткое описание | Выход пользователем из системы |
| Актёры | Пациент, врач, администатор |
| Предусловие | Авторизованный пользователь |
| Основной поток | Авторизованный пользователь, находясь на своей странице, нажимает на ссылку «Выход» в верхне-левом углу и тем сам осуществляет выход из системы |
| Альтернативный поток | отсутствует |
| Постусловия | Неавторизованный пользователь |

Изобразим варианты использования на на use case диаграмме (рисунок 1)



1. Use case диаграмма

# Анализ

([глава 6](http://yandex.ru/yandsearch?text=UML+2+%D0%B8+%D1%83%D0%BD%D0%B8%D1%84%D0%B8%D1%86%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9+%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81))

Номинальный объём: 8 страниц

Основное отличие модели вариантов использования от модели анализа состоит в том, что при построении первой основное внимание уделяется определению функциональных требований системы, а при построении второй – их уточнению с учетом внутренней архитектуры проектируемой системы. В связи с этим на второй стадии может использоваться более формальный и специфичный язык.

Построение этой модели необходимо:

для выявления внутренней архитектуры (определения подсистем и основных классов);

для поиска альтернативных вариантов реализации системы (подсистем) и выбора основного;

для уточнения всех требований к системе.

При разработке модели анализа рекомендуется построить следующие диаграммы: классов анализа; последовательности.

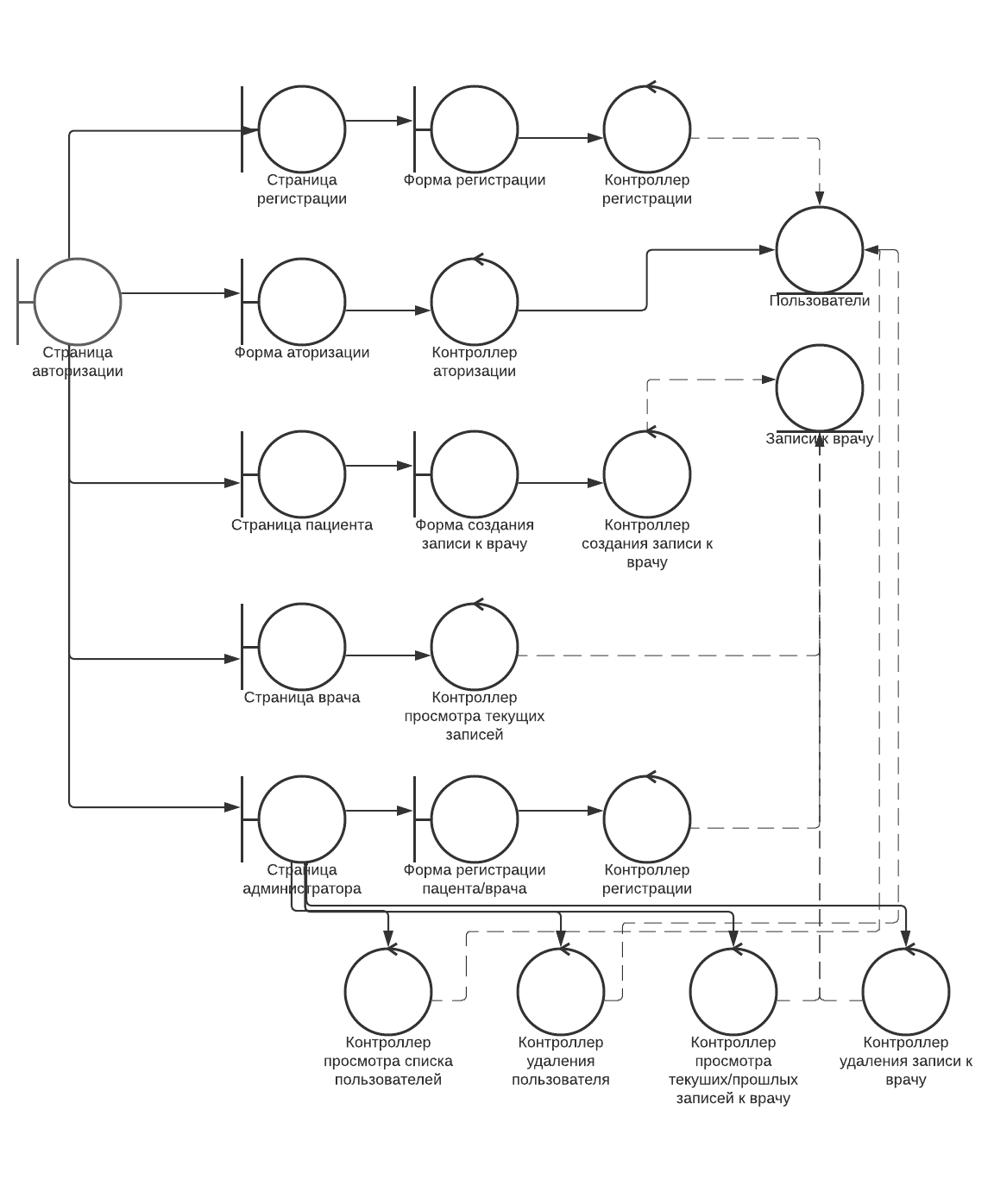
Изначально определяются состав подсистем и связи между ними, модель предметной области в виде диаграммы классов анализа. Для вариантов использования определяется их реализация в виде диаграмм взаимодействия. Реализация отдельного варианта использования требует участия и взаимодействия определенных экземпляров актеров и классов. Наиболее подходящий инструмент для описания такого взаимодействия – это диаграммы последовательности.

Диаграмма последовательностей относится к диаграммам взаимодействия UML, описывающим поведенческие аспекты системы, но рассматривает взаимодействие объектов во времени.

Предлагаемое содержание:

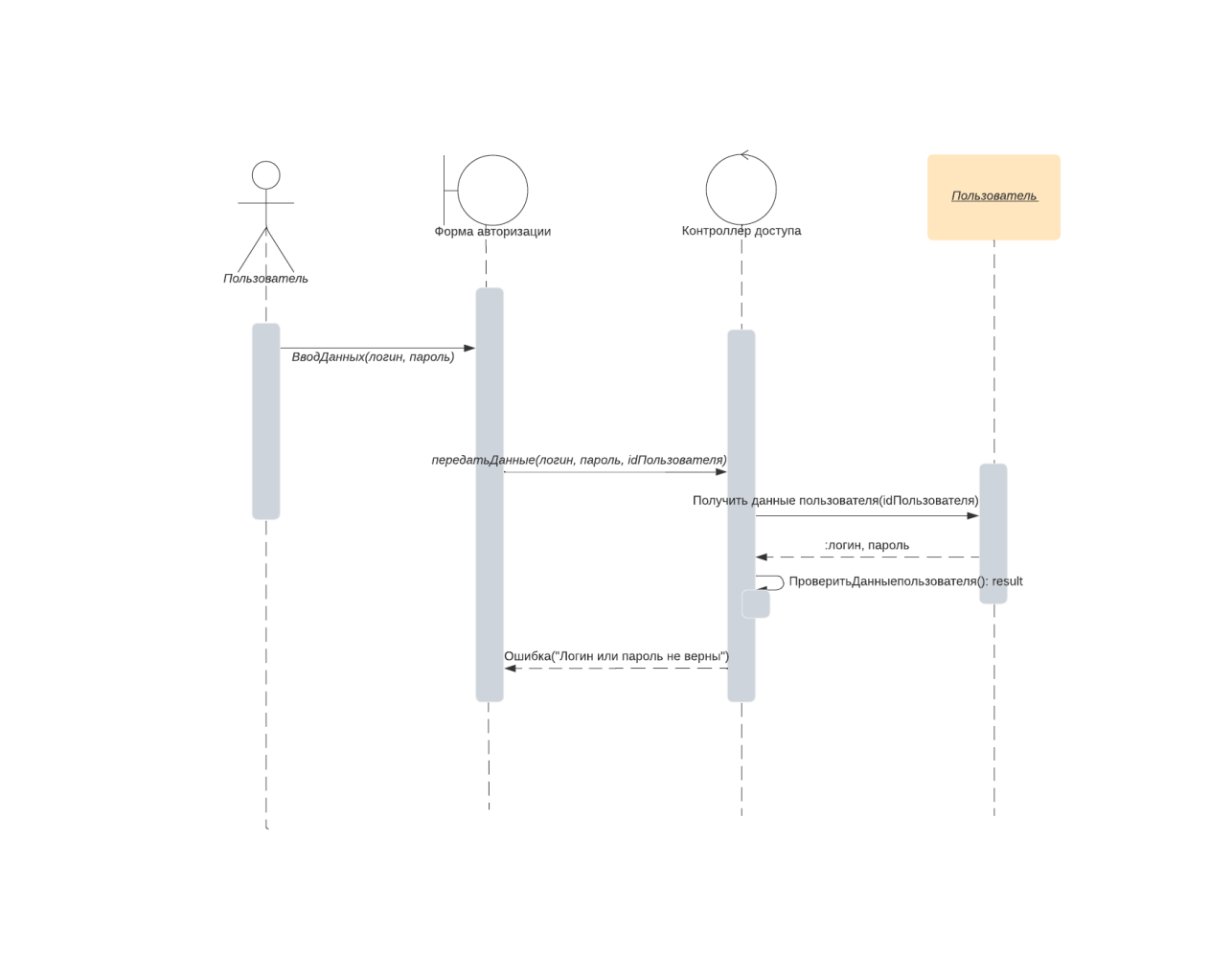
* Классы анализа: диаграмма с описанием назначения всех классов и отношений между ними ([главы 7-10](http://yandex.ru/yandsearch?text=UML+2+%D0%B8+%D1%83%D0%BD%D0%B8%D1%84%D0%B8%D1%86%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9+%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81)).

На рисунке 2 представлена диаграмма классов анализа проектируемого приложения. В левой части приведены граничные классы, по середине располагаются управляющие классы, в правой части находятся классы сущности. Граничные классы, отображаемые на данной диаграмме будут представлены представлениями (Template). Управляющие классы будут представлены контроллерами (View). Классы сущности будут представлены моделями (Model) согласно методике MTV.



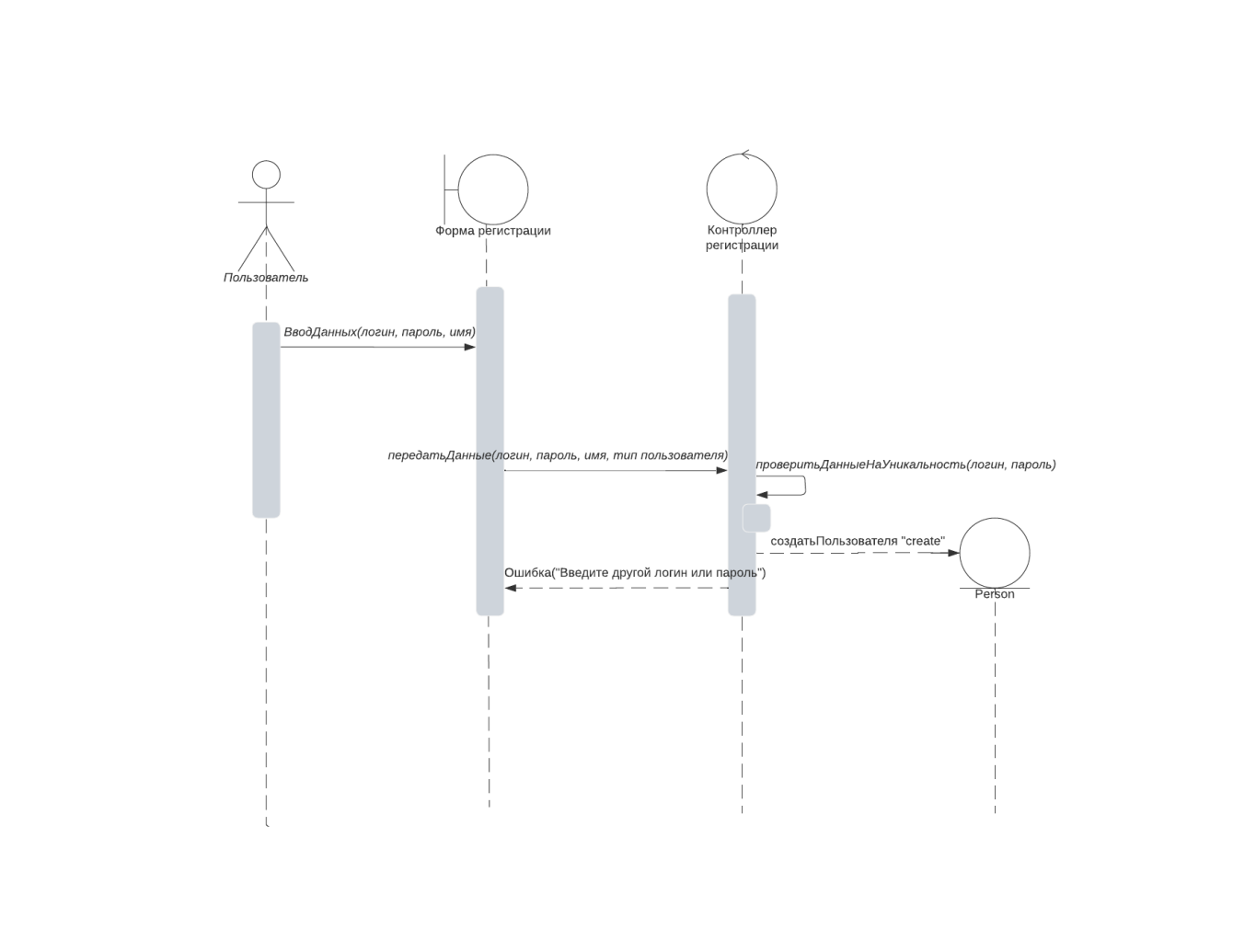
1. Диаграмма классов анализа

Диаграммы последовательностей для вариантов использования с участием классов анализа и выявленных для них операций ([глава 12](http://yandex.ru/yandsearch?text=UML+2+%D0%B8+%D1%83%D0%BD%D0%B8%D1%84%D0%B8%D1%86%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9+%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81)).



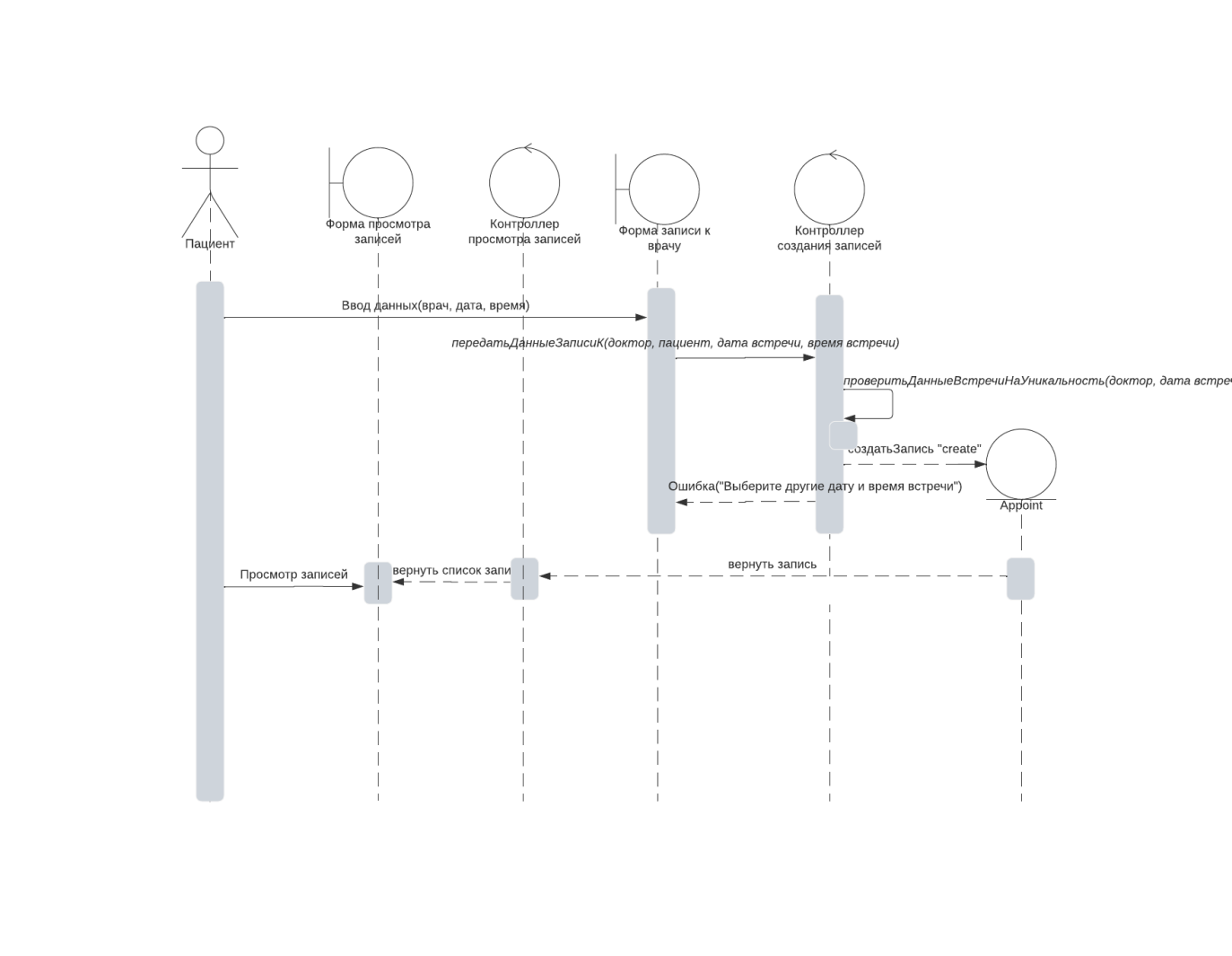
1. Диаграмма последовательности. Авторизация

На рисунке 3 представлена диаграмма последовательности «Авторизация». При авторизации изначально неавторизованный пользователь т.е. «Гость» начинает свое взаимодействие с системой входа путем ввода в форму авторизации логина и пароля. После отправки данных формы происходит их сравнение с записями БД на соответствие (логин, пароль). В случае успешной авторизации происходит переход на персональную страницу пациента, если пользователь – пациент или на страницу врача, если пользователь – врач или на административную страницу, если пользователь – администратор.



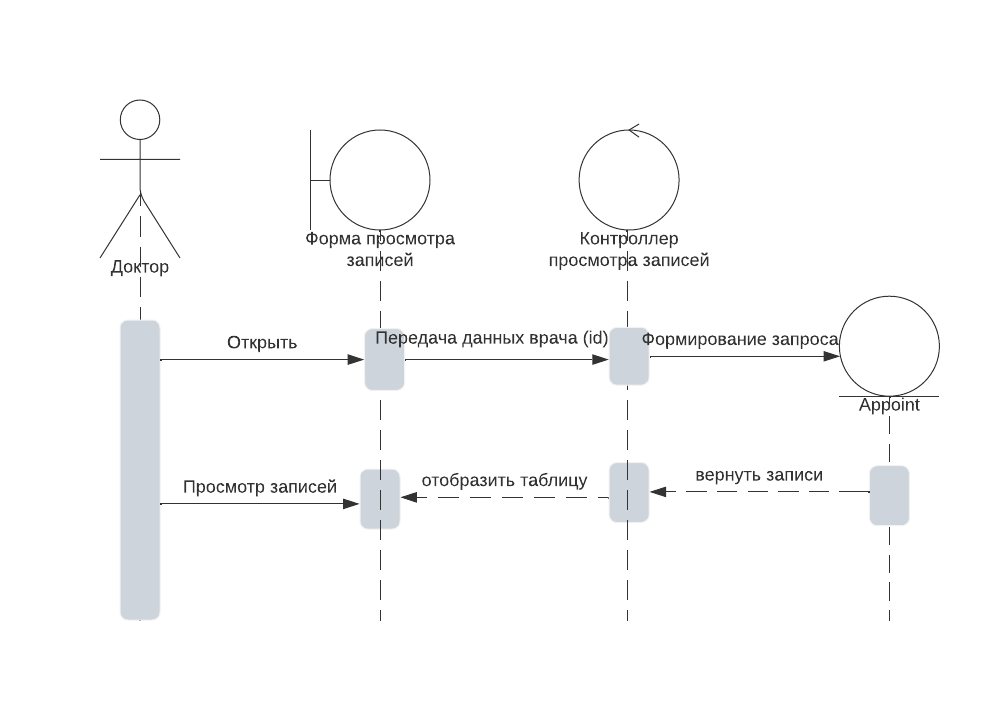
1. Диаграмма последовательностей. Регистрация

На рисунке 4 представлена диаграмма последовательности «Регистрация». Изначально неавторизованный пользователь, находящийся на странице авторизации, осуществляет переход на страницу регистрации. Далее гость заполняет форму регистрации, содержающую логин, пароль, имя. После чего отправляет данные формы на сервер, где происходит проверка данных формы (уникальность пары (логин, пароль)). В случае успешной проверки происходит создание нового пользователя и переход на страницу авторизации. В противном случае происходит уведомление об ошибке «Введите другой логин или пароль».



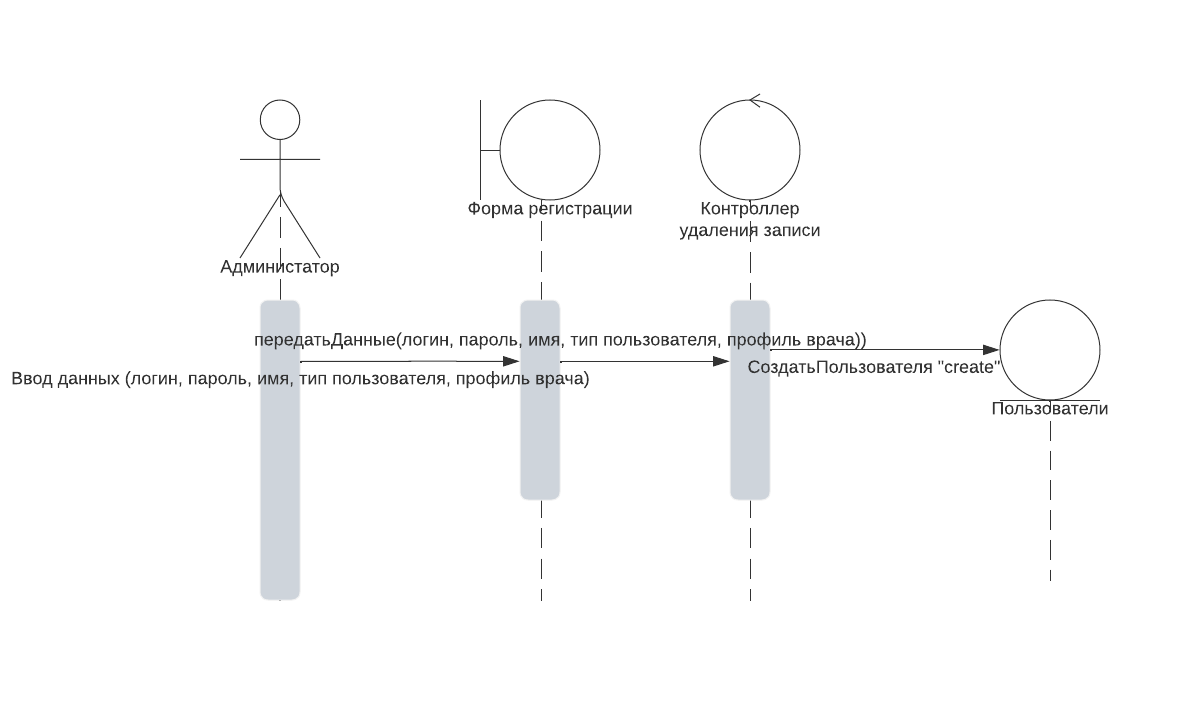
1. Диаграмма последовательностей. Создание записи к врачу

На рисунке 5 представлена диаграмма последовательностей для последовательности создания записи к врачу. Пользователь, авторизовавшийся как пациент, переходит на свою персональную страницу. Далее он заполняет данные формы записи к врачу, содержащие данные о направлении врача, конкретного специалиста, даты встречи и времени встречи, после чего отправляет их на сервер, где происходит их проверка (на то свободен ли конкретный специалист в выбранную дату и время). В случае если проверка прошла успешно создается новая запись к врачу и происходит переход к персональной странице пациента, на которой он может просмотреть список актуальных записей к врачам. В противном случае будет выдана ошибка «Выберите другие дату и время встречи».



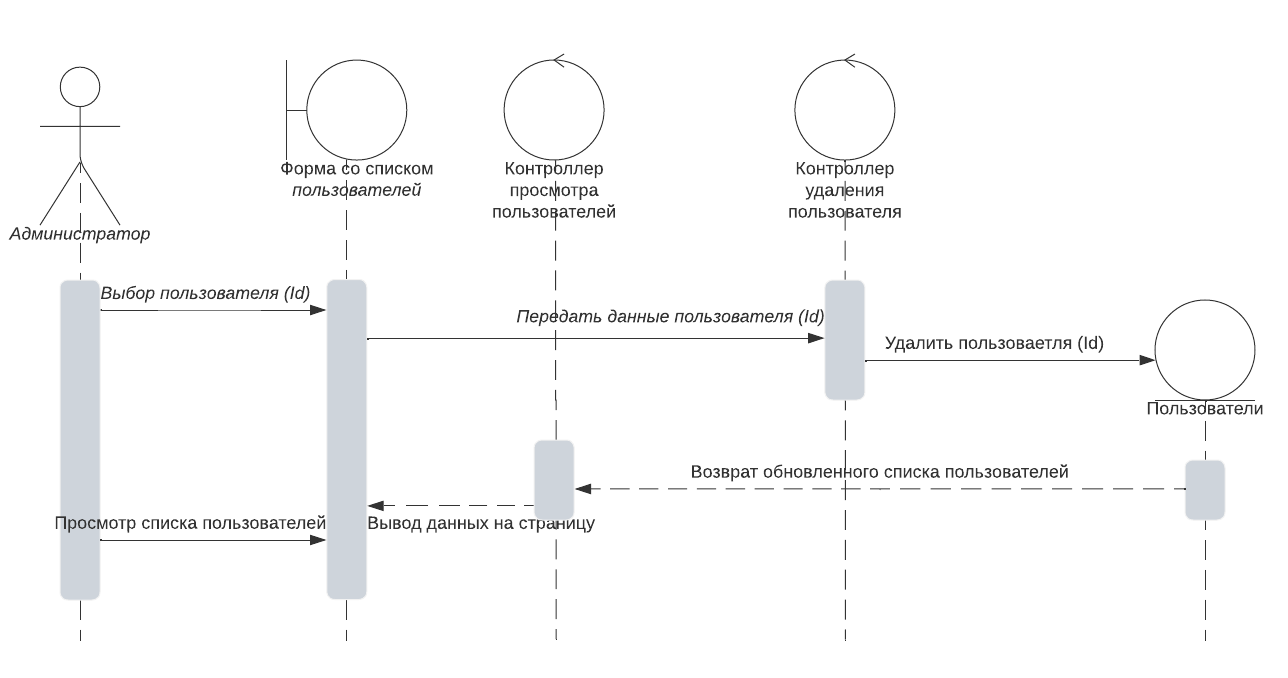
1. Диаграмма последовательностей. Просмотр записей врачом

На рисунке 6 представлена диаграмма последовательности «Просмотр записей врачом». Пользователь, авторизовавшись как доктор, передает свои данные (id) на контроллер просмотра записей, который осуществляет запрос к бд с данными встреч, после чего выводит их на персональную страницу врача в виде таблицы.



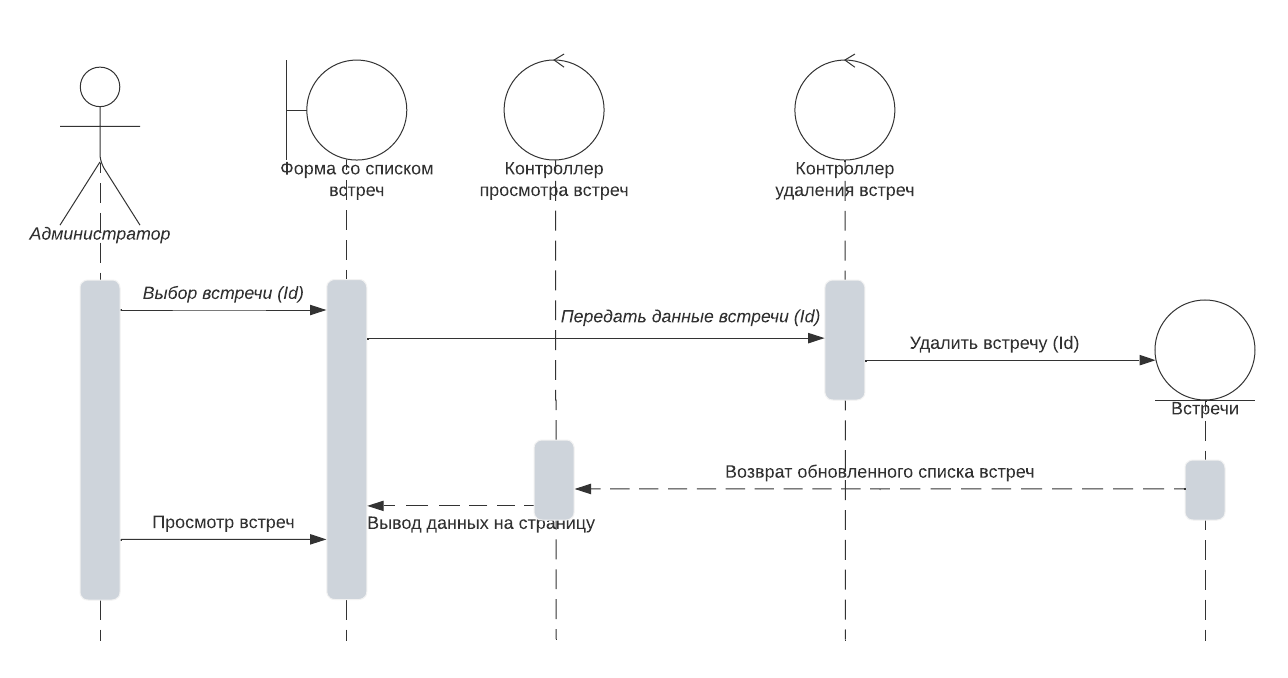
1. Диаграмма последовательностей. Создания пользователя (врача)

На рисунке 7 представлена диаграмма последовательности создания пользователя – врача администратором. Пользователь, авторизованный как администратор, находясь на административной странице заполняет данные формы регистрации указывая(логин, пароль, имя и врачебный профиль пользователя) после чего отправляет данные формы на сервер, где проходит проверка данных на уникальность (логин, пароль). В случае успешной проверки происходит создания нового пользователя. В случае ошибки происходит вывод ошибке о некорректных данных “Ошибка введите другой логин или пароль”.



1. Диаграмма последовательностей. Удаление пользователя

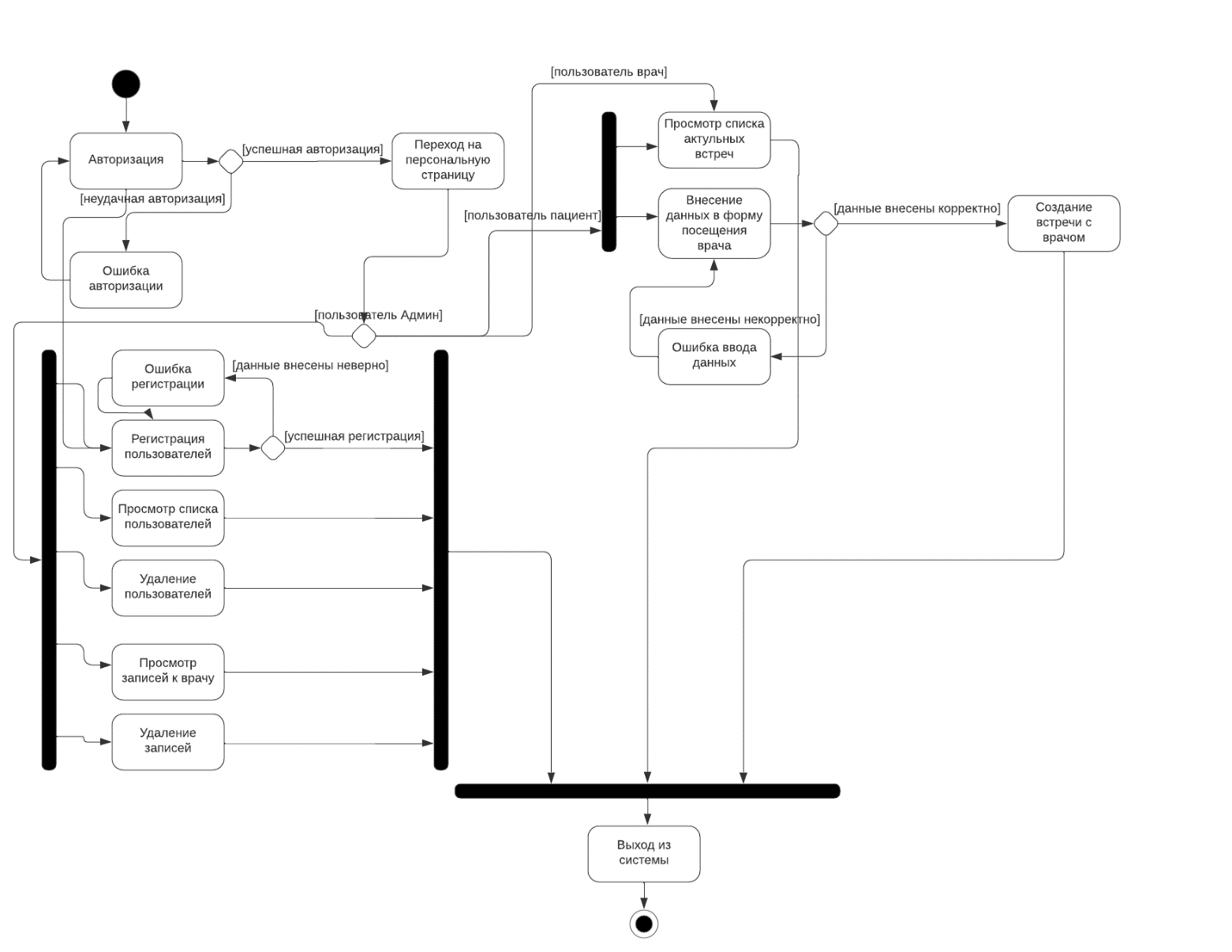
На рисунке 8 представлена диаграмма последовательности удаления пользователя администратором. Пользователь, авторизованный как администратор, находясь на административной странице пользователь просматривая список пользователей, выбирает пользователя для удаления, нажав соответствующую кнопку “x” после чего происходит удаление выбранного пользователя и отображение обновленного списка пользователей на административной странице.



1. Диаграмма последовательностей. Удаление записи к врачу

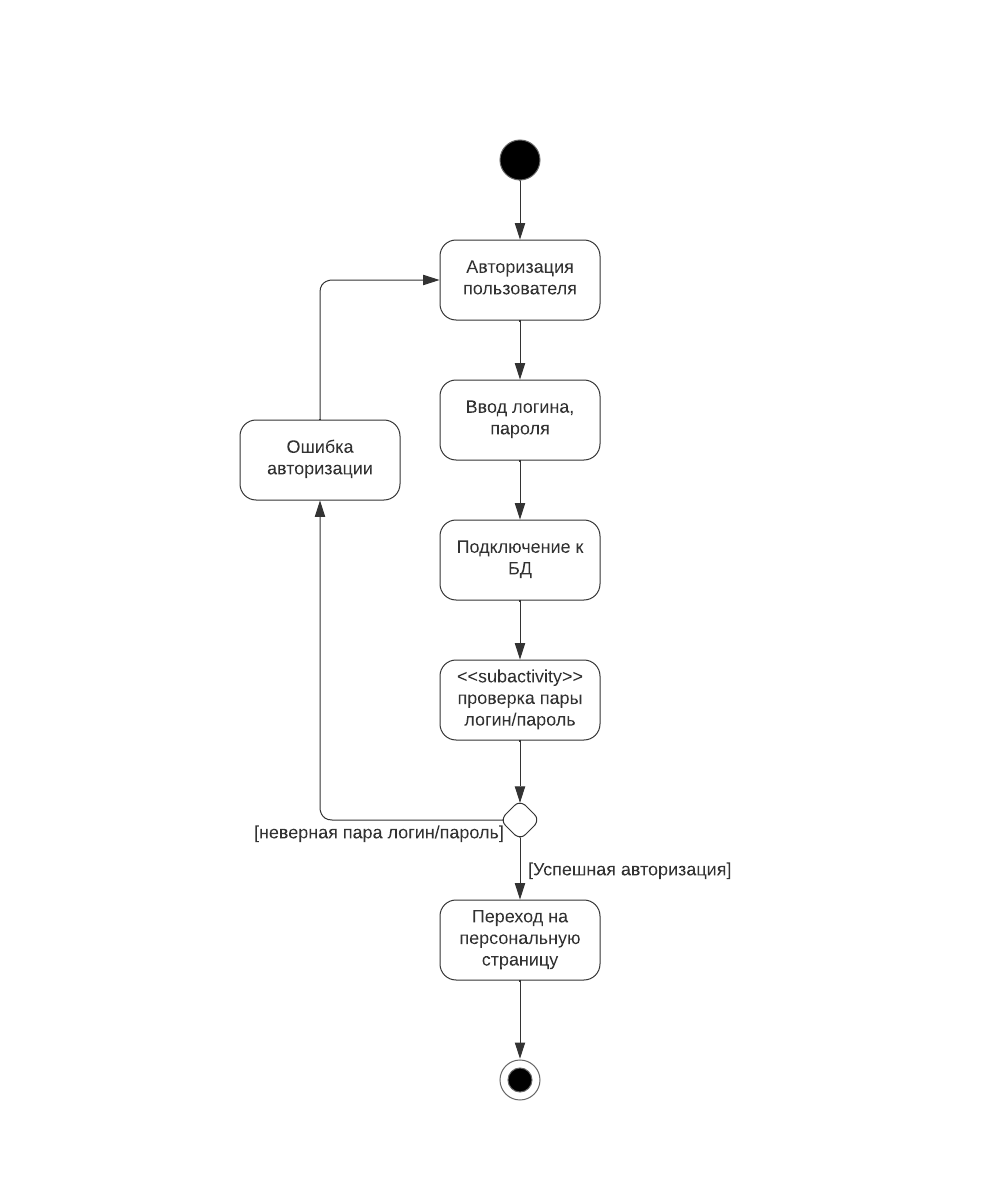
На рисунке 9 представлена диаграмма последовательности удаления записи к врачу администратором. Пользователь, авторизованный как администратор, находясь на административной странице пользователь просматривая список встреч пациентов с врачами, выбирает встречу для удаления, нажав соответствующую кнопку “x” соответствующую выбранной встрече после чего происходит отправка данных об id пользователя контроллеру, который обращается к базе данных и удаляет запись с выбранным id. После чего происходит отображение обновленного списка встреч на административной странице.

* Диаграммы деятельности для описания поведения программы ([глава 14](http://yandex.ru/yandsearch?text=UML+2+%D0%B8+%D1%83%D0%BD%D0%B8%D1%84%D0%B8%D1%86%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9+%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81)).



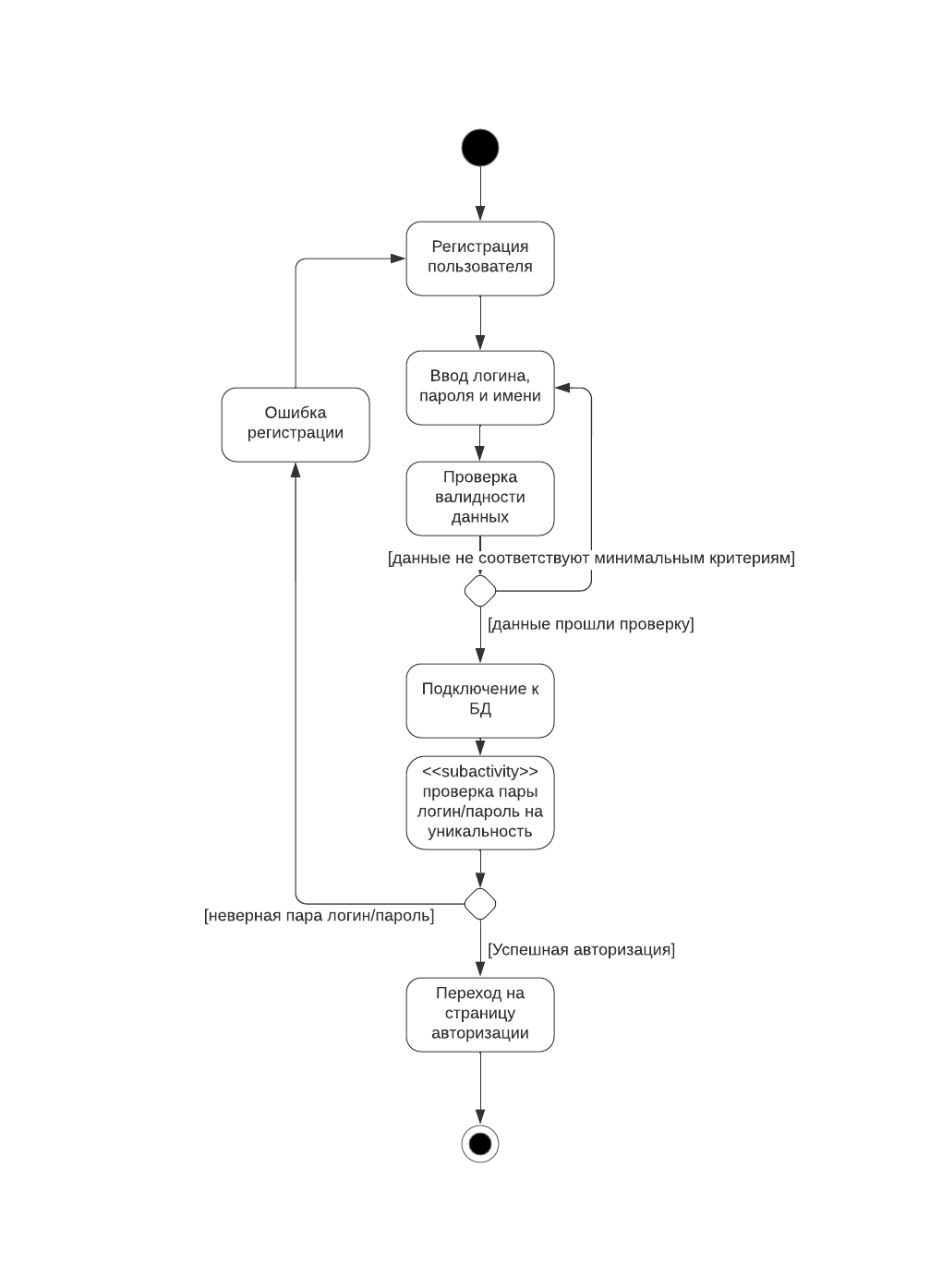
1. Диаграма деятельности программы

На рисунке 10 представлена общая диаграмма деятельности программы, которая отображает все возможные последовательности действий пользователя, а также поведение системы, переход из одной подсистемы в другую.



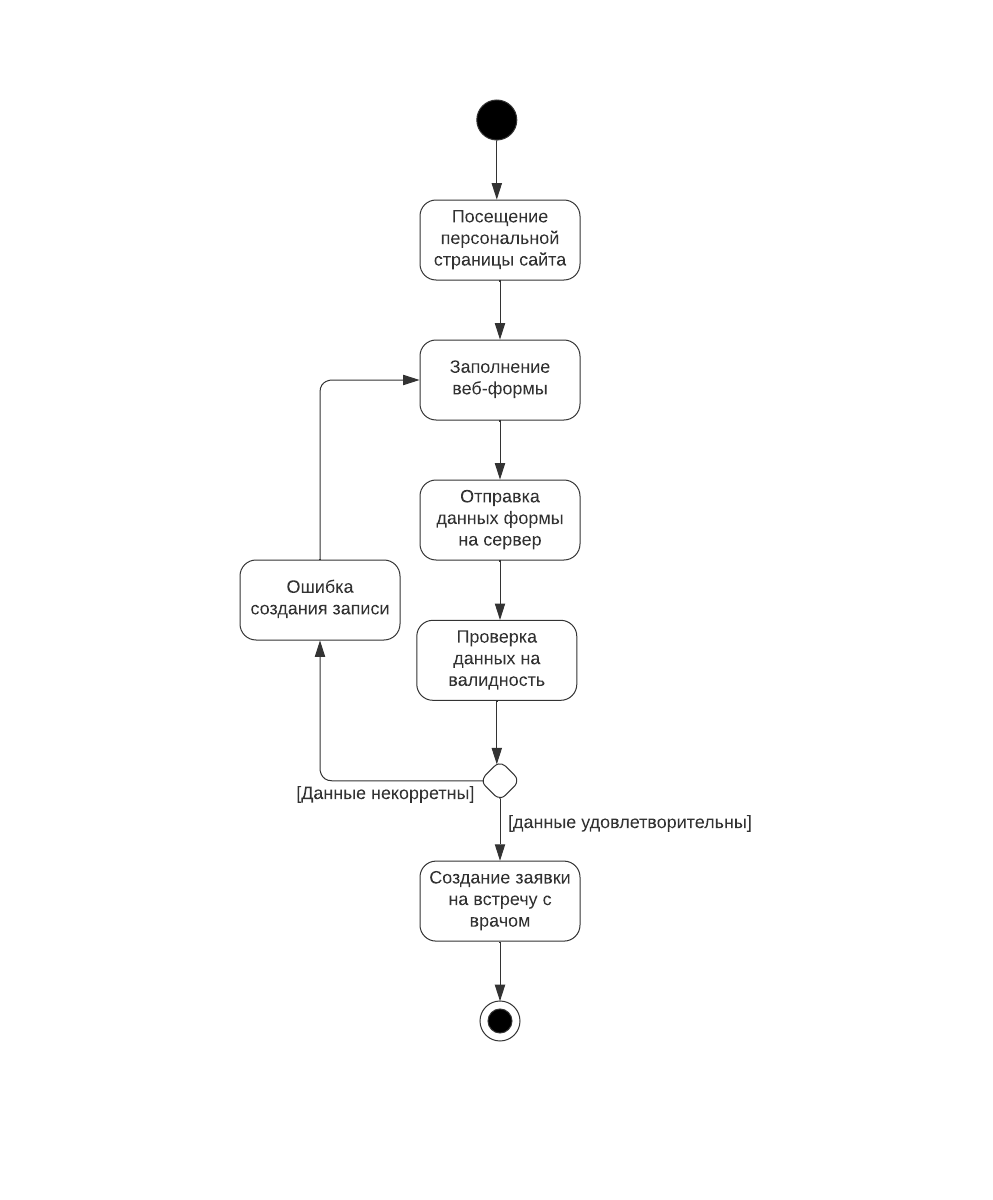
1. Диаграма деятельности. Авторизация

На рисунке 11 представлена диаграмма деятельности, которая отображает последовательность действий пользователя, а также поведение системы при авторизации пользователя.



1. Диаграмма деятельности. Регистрация

На рисунке 12 представлена диаграмма деятельности, которая отображает последовательность действий пользователя, а также поведение системы при регистрации пользователя.



1. Диаграмма деятельности. Создание заявки на встречу с врачом

На рисунке 13 представлена диаграмма деятельности, которая отображает последовательность действий пользователя, а также поведение системы при попытке создания записи к врачу.

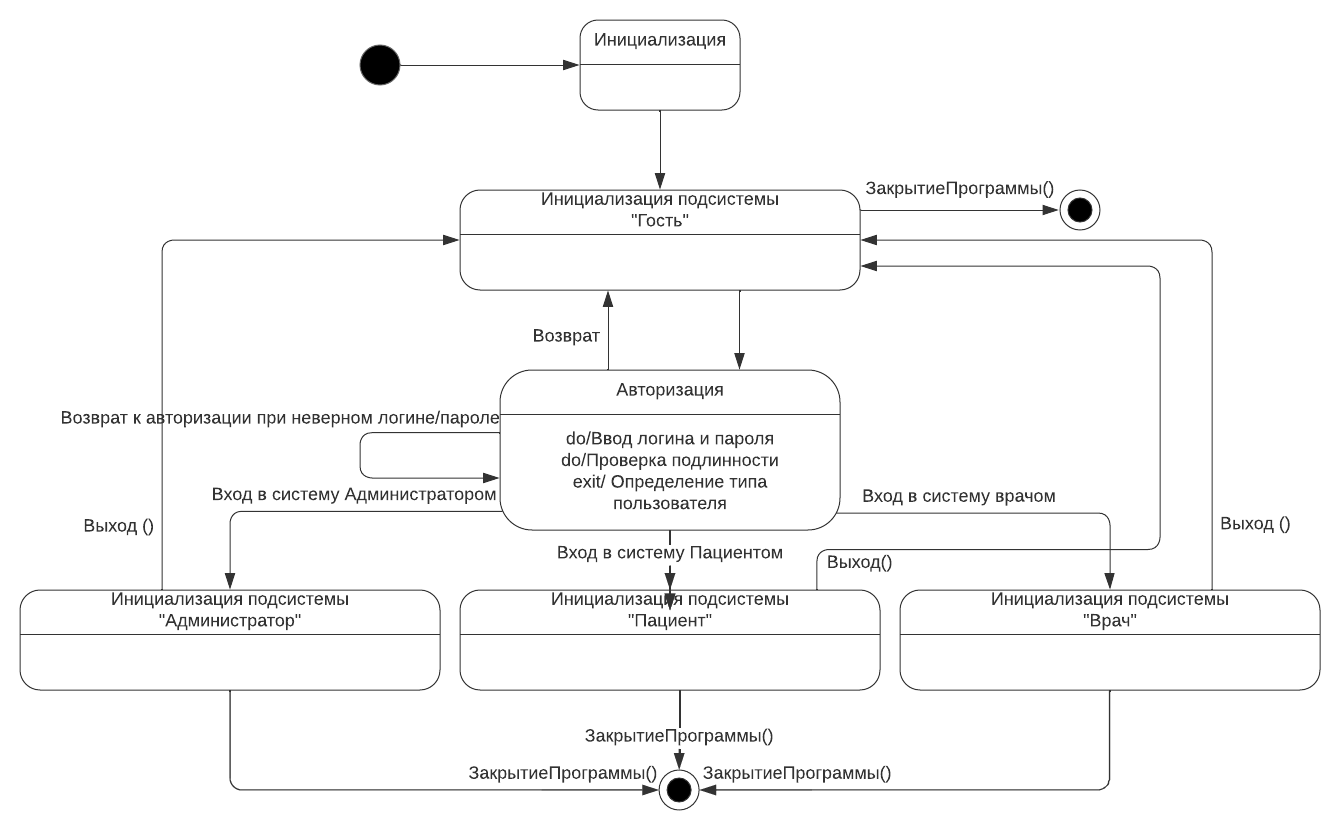
* Конечные автоматы для динамических классов анализа ([глава 21](http://yandex.ru/yandsearch?text=UML+2+%D0%B8+%D1%83%D0%BD%D0%B8%D1%84%D0%B8%D1%86%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9+%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81)).

Диаграммы автоматов используются для описания поведения, реализуемого в рамках варианта использования, или поведения экземпляра сущности (класса, объекта, компонента, узла или системы в целом). Поведение моделируется через описание возможных состояний экземпляра сущности и переходов между ними на протяжении его жизненного цикла, начиная от создания и заканчивая уничтожением.

Диаграмма автоматов представляет собой связный ориентированный граф, вершинами которого являются состояния, а дуги служат для обозначения переходов из состояния в состояние.

Под состоянием понимается ситуация в ходе жизни экземпляра сущности, когда эта ситуация удовлетворяет некоторому условию, экземпляр выполняет некоторые операции или ждет наступления некоторого события. В UML различают два вида операций: действие и деятельность. Действие – это атомарная операция, выполнение которой не может быть прервано, приводящая к смене состояния или возвращающая значение. Примерами действий служат операции создания или уничтожения объекта, расчет факториала и т. д. Деятельность – это составная операция, реализуемая экземпляром в конкретном состоянии, выполнение которой может быть прервано. В частности, под деятельностью можно понимать процедуры расчета допускаемых скоростей или шифрования данных.

В проектируемой системе существует три типа пользователей так как каждый из них имеет различные полномочия, то необходимо разделить интерфейс предоставляемый для каждой категории пользователей.



1. Диаграмма состояний. Авторизация

На рисунке 14 представлена контексная диаграмма автоматов системы Авторизации из которой видно, что при открытии страницы сайта, посетитель автоматически попадает в категорию Гость, после чего он проходит Авторизацию, производит ввод логина и пароля, которые проверяются на подлинность.

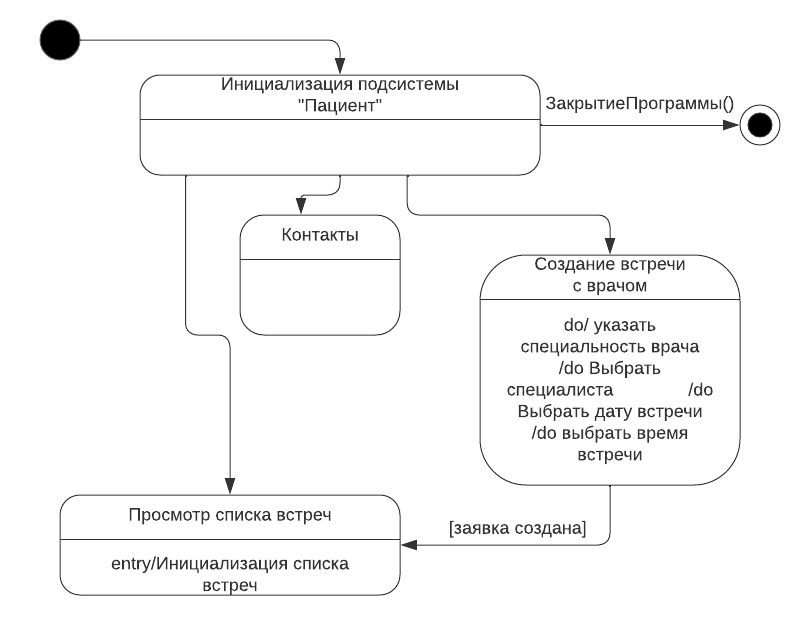
В случае, если проверка прошла успешно посетителю сайта входит в соответсвующую подсистему. Если проверка не пройдена происходит возврат к окну авторизации с уведомление об ошибке.

Используемые условные обозначения:

ЗакрытиеПрограммы () – закрытие программы;

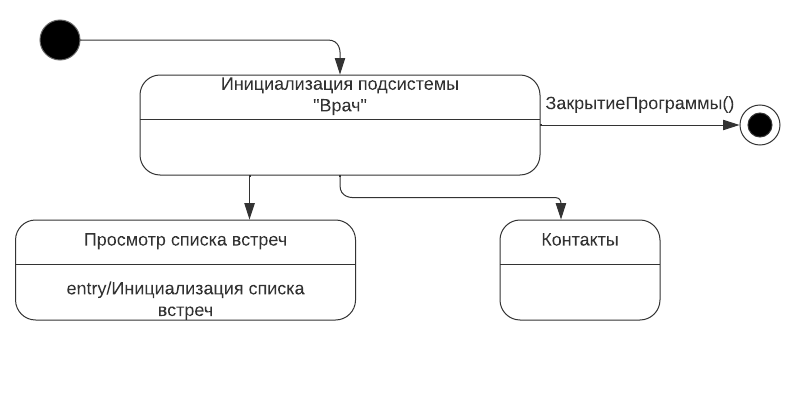
Выход() – выход из подсистемы.

Рассмотрим более подробно работу с подсистемами «Администратор», «Пациент» и «Врач».



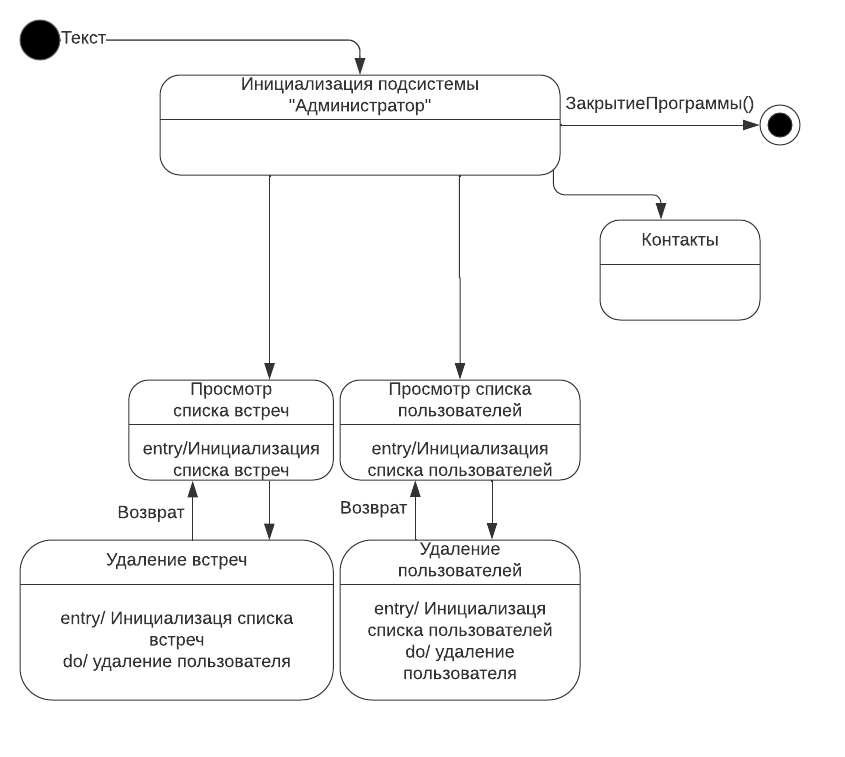
1. Диаграмма состояний. Подсистема «Пациент»

На рисунке 15 представлена диаграмма автоматов ддя подсистемы «Пациент». После инициалзиции посетитель сайта автоматически переходит на персональную страницу пацента, накоторой расположена форма для создания записи к врачу, в которой пациент выбирает направление врача, специалиста, дату и время предстоящей встречи. Ниже расположен список актуальных для пациента встреч с врачом. Пациент также может перейти в раздел контакты и посмотреть контактную информацию.



1. Диаграмма состояний. Подсистема «Врач»

На рисунке 16 представлена диаграмма автоматов для подсистемы «Врач». Во многом она похожа на подсистему «Пользователь» за исключением отсутствия в ней формы для создания записи к врачу.



1. Диаграмма состояний. Подсистема «Админ»

На рисунке 17 представлена диаграмма автоматов для подсистемы «Администратор». Данная подсистема предоставляет особый функционал для работы с объектами приложения: регистрация пользователей, возможность просмотра списка пользователей и их удаление, возможность просмотра списка встреч и их удаление.

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя класса** | Person | | |
| **Описание** | Person отвечает за данные пользователя | | |
| **Ответственности** | Для обслуживания информации о пользователе | | |
| **Атрибуты** | **Имя** | **Описание** | **Тип** |
| Имя | Имя пользователя | Varchar(20) |
| Логин | Логин | Varchar(20) |
| Пароль | Пароль | Varchar(20) |
| Тип пользователя | Тип пользователя (пациент или врач) | Bool |
| Дата регистрации | Дата регистрации пользователя | Datetime |

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя класса** | docprofile | | |
| **Описание** | docprofile отвечает за данные о медицинких направлениях | | |
| **Ответственности** | Для обслуживания информации о специализации врача | | |
| **Атрибуты** | **Имя** | **Описание** | **Тип** |
| Профиль | Медицинский профиль/направление | Varchar(20) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя класса** | Doctor | | |
| **Описание** | Doctor отвечает за данные о враче | | |
| **Ответственности** | Для обслуживания информации о враче | | |
| **Атрибуты** | **Имя** | **Описание** | **Тип** |
| user | OneToOneField(Person) | Int |
| profile | Медицинский профиль | Varchar(20) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя класса** | Patient | | |
| **Описание** | Patient отвечает за данные о пациенте | | |
| **Ответственности** | Для обслуживания информации о пациенте | | |
| **Атрибуты** | **Имя** | **Описание** | **Тип** |
| user | OneToOneField(Person) | Int |

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя класса** | Appoint | | |
| **Описание** | Patient отвечает за данные о встрече/записи | | |
| **Ответственности** | Для обслуживания информации о встрече | | |
| **Атрибуты** | **Имя** | **Описание** | **Тип** |
| doctor | Внешний ключ (Doctor) | Int |
| patient | Внешний ключ (Patient) | Int |
| appointdate | Дата встречи | Date |
| appointtime | Время встречи | Time |
| registered | Дата регистрации встречи | Datetime |

# Проектирование

([глава 16](http://yandex.ru/yandsearch?text=UML+2+%D0%B8+%D1%83%D0%BD%D0%B8%D1%84%D0%B8%D1%86%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9+%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81))

Номинальный объём: 8 страниц

Предлагаемое содержание:

Модель проектирования представляет собой высокоуровневое (на уровне подсистем) и низкоуровневое (на уровне классов, если речь идет об использовании объектно-ориентированного подхода) описание программной системы. Назначение модели проектирования заключается в создании полного детализированного описания внутренней архитектуры и алгоритмов работы системы. Модель проектирования включает в себя диаграммы классов и диаграммы деятельности. В данной главе рассматриваются диаграммы деятельности, а диаграммы классов будут рассмотрены в третьей главе, посвящённой проектированию структуры базы данных.

При моделировании поведения системы возникает необходимость не только представить процесс изменения ее состояний, но и детализировать особенности алгоритмической и логической реализации выполняемых системой операций. Для описания поведения системы и ее отдельных элементов (поведенческих моделей) в UML предусмотрено четыре вида диаграмм.

* Проектные классы ([главы 17, 18](http://yandex.ru/yandsearch?text=UML+2+%D0%B8+%D1%83%D0%BD%D0%B8%D1%84%D0%B8%D1%86%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9+%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81)).
* Диаграммы последовательностей для операций проектных классов ([глава 12](http://yandex.ru/yandsearch?text=UML+2+%D0%B8+%D1%83%D0%BD%D0%B8%D1%84%D0%B8%D1%86%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9+%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81)).
* Конечные автоматы для динамических проектных классов ([глава 21](http://yandex.ru/yandsearch?text=UML+2+%D0%B8+%D1%83%D0%BD%D0%B8%D1%84%D0%B8%D1%86%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9+%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81)).
* Компоненты и программные интерфейсы ([глава 19](http://yandex.ru/yandsearch?text=UML+2+%D0%B8+%D1%83%D0%BD%D0%B8%D1%84%D0%B8%D1%86%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9+%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81)).

# Реализация

([глава 23](http://yandex.ru/yandsearch?text=UML+2+%D0%B8+%D1%83%D0%BD%D0%B8%D1%84%D0%B8%D1%86%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9+%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81))

Номинальный объём: 8 страниц

Основная цель, преследуемая при построении данной модели, – получение работоспособной версии системы. Помимо непосредственного написания программного кода будущей системы, на данной стадии окончательно определяется логическая и физическая организация классов в виде компонентов и подсистем, а также топология распределенной информационной системы. Таким образом, при разработке модели преследуются цели:

определение окончательного состава, структуры и кода классов;

распределение классов по компонентам и подсистемам;

определение топологии распределенной системы и распределение подсистем по узлам сети;

планирование итераций (версий) сборки системы;

сборка версий системы.

Диаграмма компонентов позволяет определить состав программных компонентов, в роли которых может выступать исходный, бинарный и исполняемый код, а также установить зависимости между ними. При разработке диаграмм компонентов преследуются цели:

спецификация общей структуры исходного кода системы;

спецификация исполнимого варианта системы.

Данная диаграмма обеспечивает согласованный переход от логического к физическому представлению системы в виде программных компонентов. Одни компоненты могут существовать только на этапе компиляции программного кода, другие – на этапе его исполнения. Основными элементами диаграммы являются компоненты, интерфейсы и зависимости между ними. Кроме этого, на ней могут отображаться ключевые классы, входящие в компоненты.

## Тестирование

* Модульное тестирование – тестирование каждого метода в отдельности с использованием заглушек.
* Интеграционное тестирование – то же, что модульное тестирование, но без использования заглушек.
* Системное тестирование – проверка всей программы в целом.
* Тестирование интерфейса
  + автоматизированное ([Coded UI tests](http://msdn.microsoft.com/en-us/library/dd286726.aspx), [Selenium WebDriver](http://docs.seleniumhq.org/projects/webdriver/), [Watir WebDriver](http://watirwebdriver.com/));
  + ручное (проверка всех заявленных вариантов использования).

## Непрерывная интеграция

* Описание того, как была выполнена автоматизация сборки (make, msbuild и т.п.), включающее конфигурационный файл системы автоматизации сборки с пояснениями.
* Описание процесса настройки системы непрерывной интеграции (например, [Jenkins](http://jenkins-ci.org/), TFS) для автоматического тестирования и сборки проекта при появлении изменений в системе управления версиями.

## Развёртывание

* Описание процесса развёртывания программы, проиллюстрированное диаграммой развёртывания ([глава 24](http://yandex.ru/yandsearch?text=UML+2+%D0%B8+%D1%83%D0%BD%D0%B8%D1%84%D0%B8%D1%86%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9+%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81)).

# Документация

Номинальный объём: 8 страниц

# Заключение

Номинальный объём: 1 полная страница

# Список использованных источников

1. <https://geekbrains.ru/posts/2019_lang_trends>
2. <https://ru.wikipedia.org/wiki/PyCharm>
3. <https://python-scripts.com/pycharm-download>
4. [https://ru.wikipedia.org/wiki/GitHub](https://ru.wikipedia.org/wiki/GitHub#cite_note-30)
5. <https://sysana.wordpress.com/2011/03/15/use-case-specification-structure/>
6. Стратегии расширения Django User Model [электронный ресурс]. Url : https://habr.com/ru/post/313764/