

Отчет по 1-му этапу проекта “Информационная система медицинских организаций города”

Системный анализ предметной области	2
Инфологическое проектирование	5
Диаграмма вариантов использования	5
Диаграмма 1. Диаграмма вариантов использования	5
Контекстная диаграмма	6
Диаграмма 2. Контекстная диаграмма	6
Диаграмма Сущность-Связь	7
Диаграмма 3. Диаграмма Сущность-Связь	7
Даталогическая модель	8
Диаграмма 4. Даталогическая модель	8
Архитектура приложения	9
Схема архитектуры приложения	10
Диаграмма 5. Схема архитектуры приложения	10
Инструкция по запуску приложения и работе с ним	10
Активный сервер. Триггеры	14
1. Описание бизнес-транзакций и ограничений целостности, требующих использование триггеров	14
2. Даталогическое описание работы триггеров	15
3. Код триггеров и фрагменты кода приложения, создающие триггеры	15
Фрагменты кода приложения, создающие триггеры:	16
4. Результаты отладки триггеров и их тестирования на тестовом наборе данных	18
Активный сервер. Хранимые процедуры и функции.	22
1. Описание бизнес-транзакций, требующих использования хранимых процедур и функций	22
2. Даталогическое описание работы хранимых процедур и функций	22
3. Код хранимых процедур и функций и фрагменты кода приложения, создающие и использующие хранимые процедуры и функции	22
Фрагменты кода приложения, создающие и использующие хранимые процедуры и функции:	23
4. Результаты отладки хранимых процедур и функций и их тестирования на тестовом наборе данных	25
Внешние схемы и интерфейсы	28
Объекты схемы базы данных, реализующие внешние схемы	28
Карты интерфейсов	43
Таблицы окон	46
Тестирование элементов внешней схемы	50
Система контроля доступа	59

Системный анализ предметной области

Приложение будет использоваться для хранения информации о медицинских организациях города: больницах и поликлиниках, а также информации о прохождении пациентами лечения в данных медицинских организациях. Жители города, зарегистрированные в перечисленных организациях, являются пациентами. В системе хранится общая информация о пациенте и номер его медицинского полиса. Амбулаторное лечение пациентов проводится в поликлиниках, при необходимости врач направляет пациента на стационарное лечение в больницу. Кроме того, в информационной системе хранится информация о врачебном персонале. Врачебный персонал обслуживает пациентов в больницах и поликлиниках.

Приложение смогут использовать следующие виды пользователей:

- пациенты (для получения информации о мед.организациях, поиска информации о врачах, записи на лечение)
- врачи (для получения списка записавшихся к ним пациентов, их персональной информации, сохранении результатов проведенного лечения)
- руководитель мед.организации (для контроля за эффективностью работы организаций, изменения кадрового состава мед.организации, получения статистики заболеваемости).

В системе присутствуют следующие бизнес-процессы:

- регистрация нового пациента: заполнение анкеты с общей информацией о пациенте, заполнение персональных данных, выдача учетных данных для входа в систему (пользователи: пациент, врач)
- проведение амбулаторного лечения: осмотр пациента, добавление записи в историю лечения и в историю болезней, при необходимости - назначение стационарного лечения или повторного амбулаторного лечения (пользователи: пациент, врач)
- проведение стационарного лечения: направление пациента в больницу, способную оказать необходимое лечение, выделение ему койки в палате, назначение операции и/или медикаментов, сбор сведений о состоянии пациента, проведение лабораторных исследований, по завершении выписки (пользователи: врач, пациент)
- сдача отчета: получение руководителем отчета о работе организации за определенный период (пользователи: руководитель мед.организации)
- принятие врача/увольнение врача, выдача учетных данных для входа в систему (пользователи: руководитель мед.организации, врач)
- получение статистики заболеваемости для принятия соответствующих карантинных мер.

Бизнес процессы, которые реализуют пользователи, информационные потоки, запросы и отчеты, с ними связанные:

- Пациент

- запись на лечение

(1) Получить список медицинских организаций

(2) Получить перечень врачей для конкретного медицинского учреждения

(3) Добавить новую запись к выбранному врачу в выбранную дату

- поиск врача

(1) Получить перечень врачей для конкретного медицинского учреждения

- поиск телефона или адреса медицинского учреждения

(1) Получить перечень всех больниц города

(2) Получить перечень всех поликлиник города

- Руководитель мед.организации

- контроль количества обслуживаемых пациентов

(1) Получить количество обслуженных за конкретный период времени пациентов текущей мед.организации

- получении данных о проведенном лечении

(1) Получить подробное описание лечения

- добавить в список врачей мед.организации нового сотрудника

(1) Добавить запись в список врачей мед.организации

- изменить данные сотрудника текущей медицинской организации

(1) Изменить запись о сотруднике

- удалить данные о сотруднике текущей медицинской организации после увольнения

(1) Удалить данные о сотруднике

- Врач

- ведение записей о проведенном лечении

(1) Добавить запись о лечении

- ведение учета истории болезней пациентов

(1) Добавить новую запись в историю болезней пациентов

- Внесение записи о новом пациенте

(1) Добавление персональных данных пациента

(2) Редактирование персональных данных пациента

(3) Удаление персональных данных пациента

Основные информационные объекты, которые будут храниться в системе и связи между этими объектами:

- Пациенты (проходят амбулаторное или стационарное лечение)
- Больницы (проводят стационарное лечение пациентов)
- Поликлиники (проводят амбулаторное лечение пациентов)
- Врачи (работают в больнице и/или поликлинике)

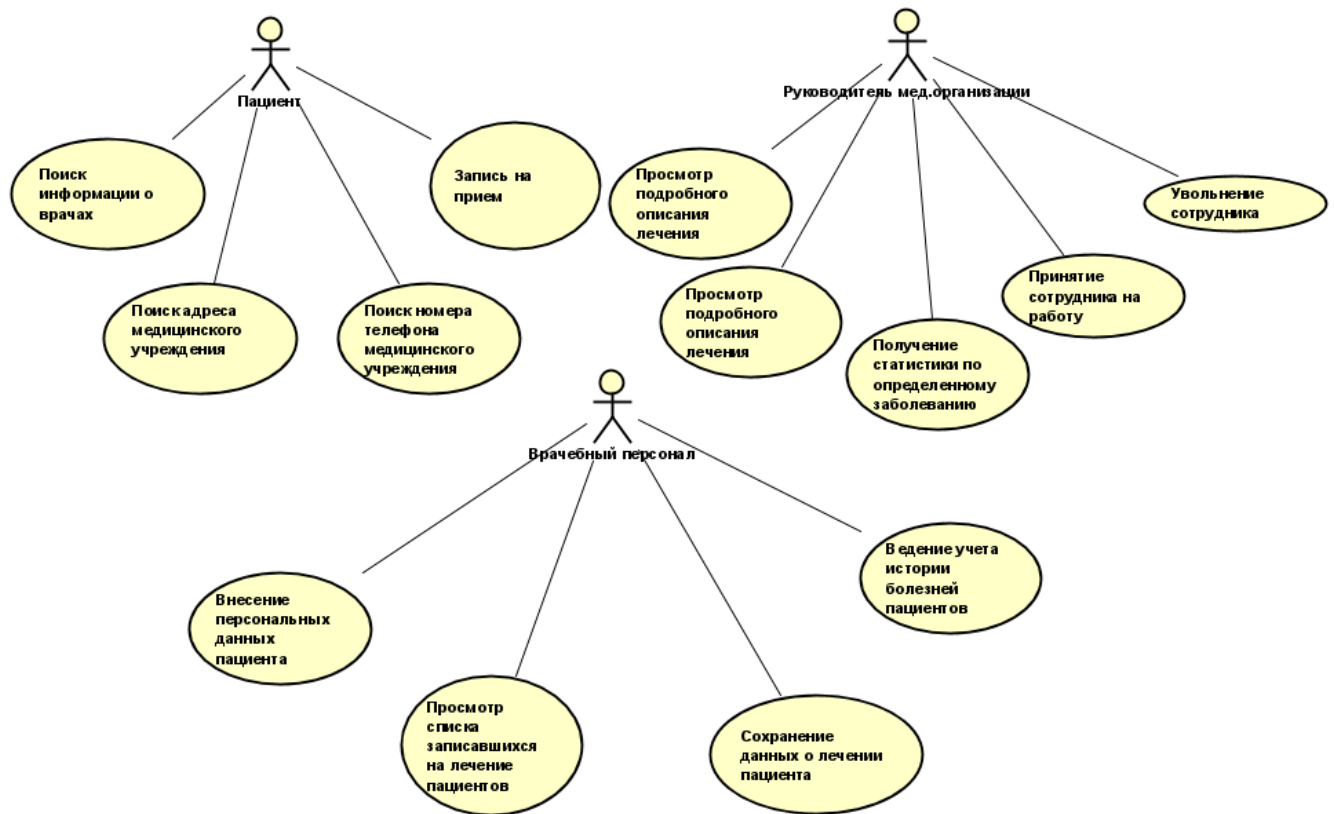
Основные ограничения, накладываемые на информацию исходной задачей:

- Все медицинские организации делятся на два типа: поликлиники и больницы
- Каждый врач может работать в больнице и/или поликлинике
- При добавлении записи о лечении пациента, его диагноз должен сохраняться в историю болезней
- После принятия пациента врачом его запись на прием отмечается как выполненная

Инфологическое проектирование

Диаграмма вариантов использования

Диаграмма 1. Диаграмма вариантов использования



В диаграмме отражены основные виды пользователей информационной системы:

- Пациент имеет возможность поиска в системе данных о врачах, контактов и адресов мед. учреждений, записи на прием.
- Руководитель мед. организаций может использовать данную систему для получения статистики по эффективности работы учреждения, просмотра подробного описания лечения пациентов, статистики по заболеваемости. При выполнении/невыполнении требований руководитель может принять решение об увольнении сотрудника и внести соответствующие изменения в систему. При принятии нового сотрудника на должность врача руководитель мед. организации вносит его в базу данных.
- Врач может добавить нового пациента в систему и внести его персональные данные, получить список пациентов, записавшихся на лечение и сохранить данные о проведенном лечении.

Контекстная диаграмма

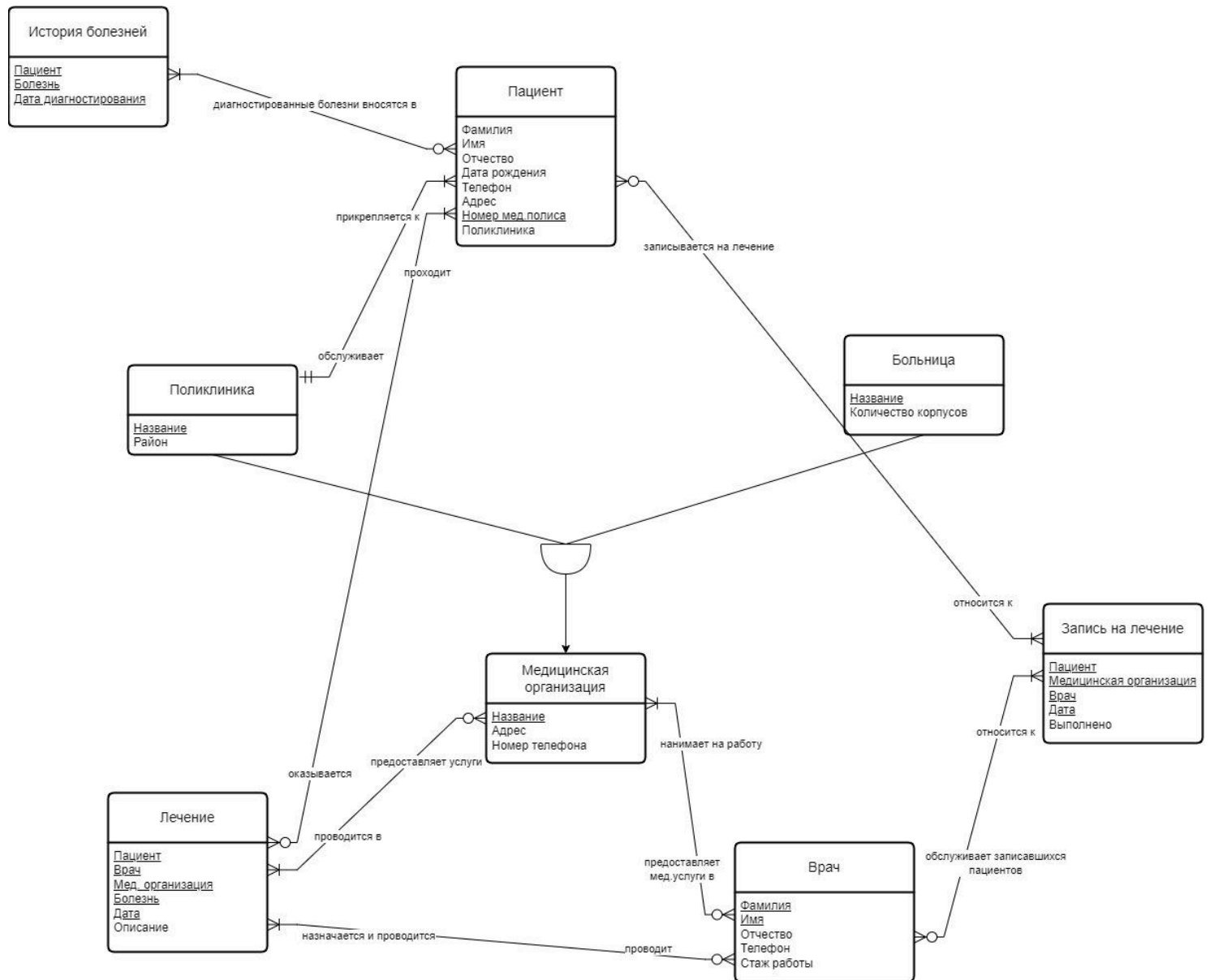
Диаграмма 2. Контекстная диаграмма



На контекстной диаграмме отображены основные виды пользователей системы и потоки данных, связывающие их с информационной системой. Стрелками показано, какие манипуляции проводятся над данными: получение или добавление/изменение. Врачи вносят в информационную систему персональные данные новых пациентов, фиксируют диагноз пациента и проведенное лечение. Из системы врачи получают персональные данные пациента, получают текущий список пациентов, записанных к ним на лечение. Пациенты через систему записываются на прием, а также могут получать перечень действующих мед. организаций, дополнительной информации о них и о врачебном персонале. Руководитель мед. организаций может вносить в систему данные о принятии/увольнении врачей и получать отчеты о выработке организации, проведенном лечении.

Диаграмма Сущность-Связь

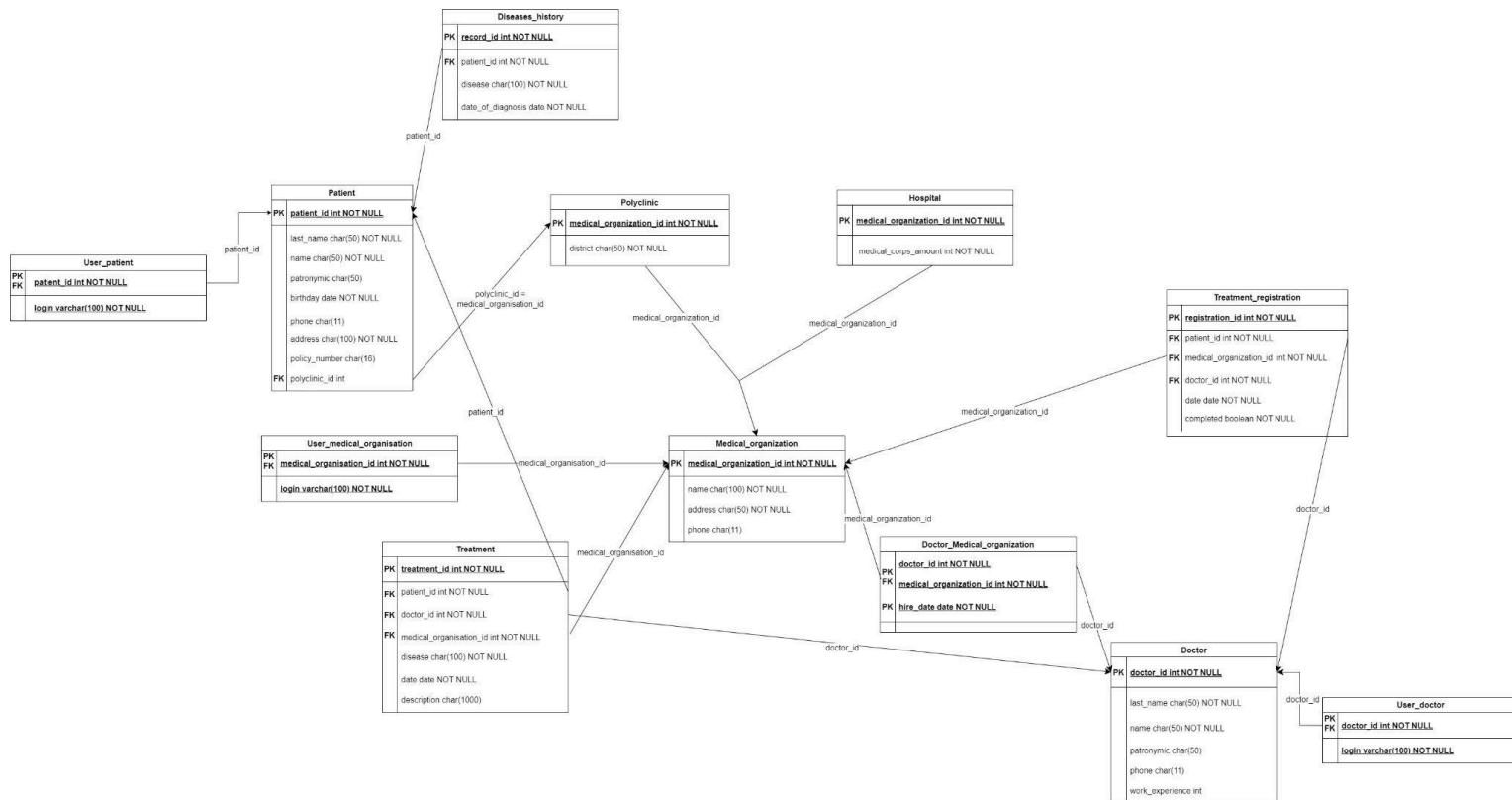
Диаграмма 3. Диаграмма Сущность-Связь



В диаграмме Сущность-Связь отражены основные сущности и отношения, которыми они связаны. Концы стрелок, соединяющих сущности, характеризуют вид связи (обязательная или нет, 1 к 1, 1 ко многим, много ко многим). В каждой сущности подчеркнуты атрибуты, которые являются ключевыми. Объекты некоторых сущностей можно однозначно идентифицировать лишь по набору атрибутов, поэтому в некоторых сущностях выделено несколько ключевых атрибутов. На диаграмме присутствует узел-дискриминатор, отражающий отношение категоризации. Сущность Медицинская организация является супертипом для сущностей Поликлиника и Больница. Сущность-супертип содержит атрибуты, общие для всех подтипов. Идентификатор сущности-супертипа и сущности-подтипа совпадают, что позволяет по нему восстановить полную информацию об объекте.

Даталогическая модель

Диаграмма 4. Даталогическая модель



Даталогическая модель построена на основе диаграммы Сущность-Связь. В данной модели добавлены типы атрибутов, определены первичные и внешние ключи. Стрелками обозначены связи таблиц посредством внешних ключей, на середине стрелок подписаны атрибуты, по которым они соединяются. Отношения много-ко-многим реализованы посредством введения дополнительной таблицы, содержащей их соответствия (на диаграмме это таблица *Doctor_Medical_organisation*). Эта таблица содержит составной первичный ключ, состоящий из идентификатора врача, идентификатора медицинской организации и даты принятия на должность (подразумевается, что один и тот же врач может несколько раз приниматься и увольняться). Таблицы, отражающие сущности, связанные отношением категоризации, связаны по идентификатору, который имеет одно и то же значение для экземпляра сущности-подтипа экземпляра сущности-подтипа, соответствующего ему (для поликлиники будет запись в таблице *Polyclinic* с названием обслуживаемого района и идентификатором, совпадающим с таблицей *Medical_organisation*, где хранится общая информация об учреждении). Те атрибуты таблиц, которые помечены ключевым словом NOT NULL являются обязательными, значения остальных могут отсутствовать. Таблицы *User_patient*, *User_doctor*, *User_medical_organisation* являются вспомогательными и содержат соответствия между логинами пользователей СУБД и идентификаторами тех сущностей, которые они представляют. Эти таблицы используются на этапе авторизации пользователя.

Архитектура приложения

Архитектура информационной системы построена таким образом, что прикладной компонент базы данных разделен на две части, одна из которых находится на сервере (RDA - Remote Data Access), а другая на клиенте (DBS - Database Server). Это означает, что часть данных и логики обработки данных находится на удаленном сервере, а другая часть - на клиентской стороне.

1. Серверная часть (RDA):

- На сервере располагается база данных.
- Здесь хранятся данные, выполняются запросы и обрабатывается логика базы данных.
- Сервер обеспечивает удаленный доступ к данным и предоставляет API для взаимодействия с ними.

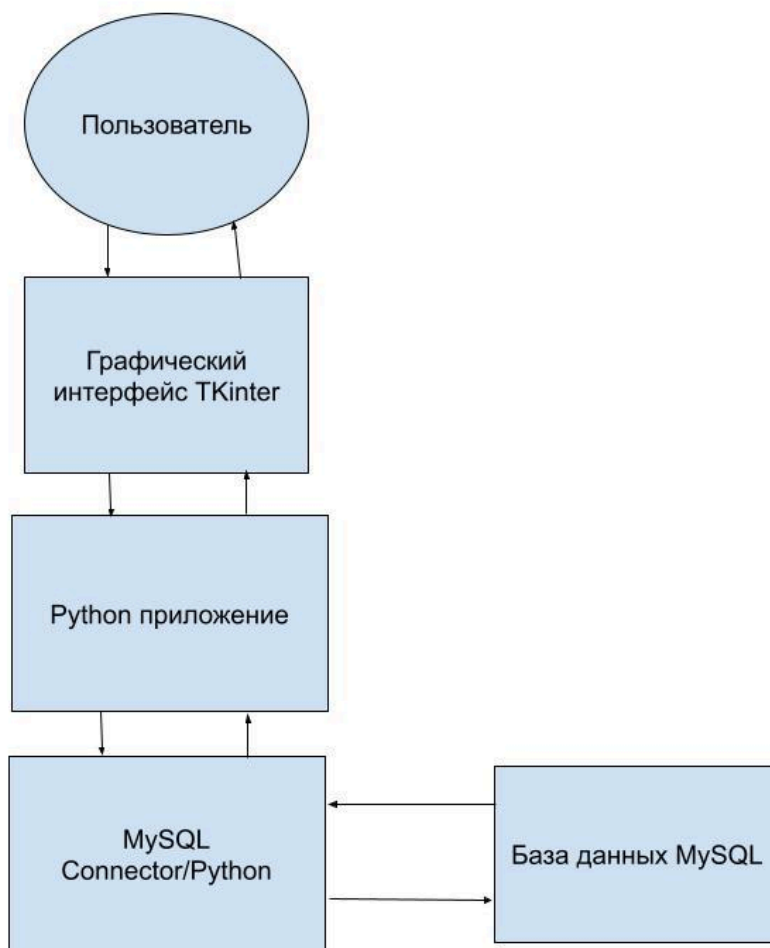
2. Клиентская часть (DBS):

- На клиентской стороне находится приложение или сервис, который использует данные из серверной части.
- Клиентская часть выполняет локальную обработку данных и часть логики приложения.
- Взаимодействие с сервером происходит через MySQL Connector/Python. С помощью него устанавливается соединение с базой данных, выполняются запросы к ней и обрабатываются результаты.

Такая архитектура позволяет распределить нагрузку между сервером и клиентом, обеспечивает гибкость в разработке и масштабировании системы. При изменении концептуальной модели данных не требуется изменение кода приложения, поскольку с помощью хранимых процедур, функций и представлений реализована логическая независимость данных.

Схема архитектуры приложения

Диаграмма 5. Схема архитектуры приложения



Инструкция по запуску приложения и работе с ним

Установка необходимых зависимостей и запуск приложения:

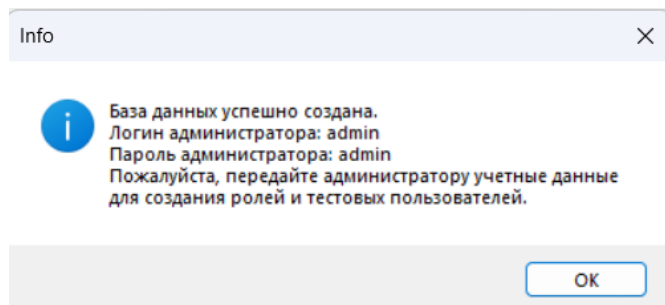
1. запустить MySQL сервер
2. убедиться, что на рабочем компьютере установлен python версии не ниже 3.10
3. для использования MySQL Connector/Python выполнить команду
\$pip install mysql-connector-python
4. для использования библиотеки TKinter выполнить команду
\$pip install tk
5. запустить приложение с помощью команды
\$python medical_organisations_db.py

Работа с системой

Внимание! Выход из системы осуществляется по кнопке “Выйти из системы”. Закрытие главных окон приведет к полному завершению работы приложения.

Первый запуск системы должен осуществляться пользователем root:

1. системный администратор запускает приложение
2. вводит логин и пароль пользователя root
3. Ожидает завершения развертывания базы данных. В случае успеха выводится сообщение с учетными данными администратора системы:



4. В случае ошибки выводится сообщение с текстом ошибки.

В ходе первого запуска приложение создаст на локальном сервере базу данных под названием “medical_organisations_db”, создаст в ней необходимые таблицы, установит ограничения целостности и создаст пользователя для администрирования системы.

Второй запуск системы должен осуществляться администратором системы:

1. администратор системы запускает приложение
2. вводит переданные ему системным администратором учетные данные (логин: admin, пароль: admin)
3. у администратора открывается окно с кнопками.

Внимание! Нажатие на кнопки должно осуществляться в строго определенном порядке.

Для этого некоторые кнопки изначально установлены в неактивное состояние.

4. В первую очередь администратор нажимает на кнопку “Создать роли”. В ходе этого создаются пользовательские роли и им выдаются права, необходимые и достаточные для использования системы. После этого все кнопки становятся активными.
5. Далее администратор может нажать на кнопку “Создать тестовых пользователей” для заполнения таблиц тестовыми данными. Либо администратор может оставить таблицы пустыми и нажать на кнопку “Управление медицинскими организациями” для создания пользователя-руководителя медицинской организации, который впоследствии сможет использовать систему.
6. При нажатии на кнопку “Управление медицинскими организациями” открывается окно с кнопками:
 - “Добавить поликлинику”: открывается окно с полями для ввода данных новой поликлиники. После ввода данных (обязательные для заполнения поля помечены *) администратор нажимает “Сохранить”, после чего либо выводится сообщение об успешном создании медицинской организации с учетными данными ее руководителя (с этими данными пользователь может входить в систему в роли руководителя созданной медицинской организации), либо выводится сообщение об ошибке. После этого пользователи-пациенты смогут записаться на прием в созданную больницу и просматривать информацию о ней.

- “Добавить больницу”: открывается окно с полями для ввода данных новой больницы. После ввода данных (обязательные для заполнения поля помечены *) администратор нажимает “Сохранить”, после чего либо выводится сообщение об успешном создании медицинской организации с учетными данными ее руководителя (с этими данными пользователь может входить в систему в роли руководителя созданной медицинской организации), либо выводится сообщение об ошибке. После этого пользователи-пациенты смогут записаться на прием в созданную больницу и просматривать информацию о ней.
- “Удалить медицинскую организацию”: открывается окно со списком всех медицинских организаций, зарегистрированных в системе. Администратор нажатием выделяет строку с медицинской организацией и нажимает “Удалить”. После этого удаленная медицинская организация будет недоступна для записи на прием и просмотра ее сведений и ее руководитель не сможет входить в систему.
- “Выйти из системы”: эта кнопка закрывает текущее соединение с базой данных и вернет пользователя к окну для настройки и авторизации.

Внимание! После создания тестовых пользователей возможна авторизация под созданными пользователями любых ролей (логины и пароли тестовых пользователей задаются в скрипте, который запускает администратор). Если создание тестовых пользователей не проводится, то администратор сначала добавляет (через выше описанные шаги) медицинскую организацию, получает учетные данные для ее руководителя и передает ему. Затем руководитель может войти в систему и добавить в свою организацию новых сотрудников (врачей). Врачи получают от руководителя учетные данные и могут войти в систему и добавлять новых пациентов. Пациент входит в систему с учетными данными, которые ему передал врач, зарегистрировавший его в системе.

Описание работы приложения:

Для пользователя “Руководитель медицинской организации”:

на главном окне расположены кнопки

- “Получить список обслуженных за промежуток времени пациентов”:
 - вверху пользователь вводит начальную и конечную даты в заданном формате, нажимает “Найти”, чтобы получить список пациентов
 - пользователь может выделить интересующую его строку и нажать “Получить подробное описание” для открытия окна с подробной информацией о выбранном лечении.
- “Получить количество пациентов, перенёсших заболевание:
 - вверху пользователь вводит название заболевания и нажимает “Найти” для получения количества пациентов текущей мед.организации, проходивших лечение указанного заболевания
- “Сотрудники”:
 - для добавления нового сотрудника пользователь заполняет поля ввода (обязательные поля перечислены в подсказках в верхней части окна) и нажимает “Сохранить”. После этого в системе создается пользователь с ролью врача и открывается окно с указанием его учетных данных.

- для добавления в качестве сотрудника текущей медицинской организации врача, который уже является сотрудником другой медицинской организации, пользователь нажимает “Добавить сотрудника из другой медицинской организации”. При этом открывается окно со списком всех сотрудников, не зарегистрированных в системе в качестве сотрудников текущей медицинской организации. Пользователь выделяет строку с нужным сотрудником и нажимает “Добавить”. При этом НЕ создается новый пользователь-врач. Учетные данные выбранного врача не меняются, меняется лишь то, что к выбранному врачу пациенты теперь смогут записываться на прием в текущую медицинскую организацию.
- для обновления данных врача пользователь выделяет строку с нужным врачом, заполняет только те поля, которые требуют изменения и нажимает “Обновить”.
- для удаления данных врача пользователь выделяет строку с нужным врачом и нажимает “Удалить”. После этого выбранной пользователь-врач удаляется из системы и больше не имеет возможности входа в нее.
- “Выйти из системы”: закроет текущее соединение с базой данных и вернет пользователя к окну для настройки и авторизации.

Для пользователя “Врач”:

на главном окне расположены кнопки

- “Пациенты”:
 - для добавления нового пациента пользователь заполняет поля ввода (обязательные поля перечислены в подсказках в верхней части окна) и нажимает “Сохранить”. После этого в системе создается пользователь с ролью пациента и открывается окно с указанием его учетных данных.
 - для обновления данных пациента пользователь выделяет строку с нужным пациентом, заполняет только те поля, которые требуют изменения и нажимает “Обновить”.
 - для удаления данных пациента пользователь выделяет строку с нужным пациентом и нажимает “Удалить”. После этого выбранной пользователь-пациент удаляется из системы и больше не имеет возможности входа в нее.
- “Список записавшихся пациентов”:
 - пользователь выбирает медицинскую организацию, для которой хочет получить список пациентов (из тех, в которых врач является сотрудником), в окне, открывшемся по кнопке “Выбрать”,
 - затем нажимает “Выполнить”, чтобы получить список пациентов на текущий день.
 - Пользователь может выделить строку с нужным пациентом и нажать “Принять пациента” для перехода к окну для сохранения данных о проведенном лечении. После ввода данных в этом окне (обязательные поля указаны в подсказках в верхней части окна) пользователь нажимает “Сохранить запись”. Это приводит к тому, что запись помечается как выполненная, и больше не показывается в текущем списке пациентов, записанных на прием.
- “Выйти из системы”: закроет текущее соединение с базой данных и вернет пользователя к окну для настройки и авторизации.

Для пользователя “Пациент”:

на главном окне расположены кнопки

- “Записаться на прием”:
 - пользователь выбирает медицинскую организацию в окне, открывшемся по кнопке “Выбрать” напротив поля “Медицинская организация”,
 - затем пользователь выбирает врача в окне, открывшемся по кнопке “Выбрать” напротив поля “Врач”,
 - затем пользователь вводит дату и нажимает “Записаться” для сохранения.

Внимание! Запись на прием на прошедшие даты недоступна. В этом случае будет выведено сообщение об ошибке и запись не сохранится в системе.

В списке ниже выводятся все записи пользователя на текущий день и позже.
- “Информация о поликлиниках”:
 - пользователь просматривает информацию о поликлиниках.
 - нажатие на заголовок любого столбца списка отсортирует его. Повторное нажатие сменит направление сортировки.
- “Информация о больницах”:
 - пользователь просматривает информацию о больницах.
 - нажатие на заголовок любого столбца списка отсортирует его. Повторное нажатие сменит направление сортировки.
- “Информация о врачах”:
 - пользователь просматривает информацию о врачах.
 - нажатие на заголовок любого столбца списка отсортирует его. Повторное нажатие сменит направление сортировки.
- “Выйти из системы”: закроет текущее соединение с базой данных и вернет пользователя к окну для настройки и авторизации.

Активный сервер. Триггеры

1. Описание бизнес-транзакций и ограничений целостности, требующих использование триггеров

Следующие бизнес-транзакции и ограничения целостности требуют использования триггеров:

- При оказании пациенту стационарного или амбулаторного лечения автоматически должна вноситься запись о его текущем диагнозе в общую историю болезней
- При изменении сведений об уже проведенном лечении пациента (изменение пациента, его диагноза или даты диагностирования) должно обновить соответствующую запись в общей истории болезней
- При добавлении сведений о проведенном лечении, соответствующая ему запись пациента на прием должна быть отмечена как выполненная
- При удалении объекта сущности Поликлиника соответствующая ей запись должна удалиться и из списка медицинских организаций (не может существовать медицинской организации, не относящегося ни к поликлинике, ни к больнице)

- При удалении объекта сущности Больница соответствующая ей запись должна удалиться и из списка медицинских организаций (не может существовать медицинской организации, не относящегося ни к поликлинике, ни к больнице)
- Персональные данные врачей (фамилия, имя, отчество) должны вноситься в систему в единообразном виде: первая буква заглавная, остальные строчные
- Персональные данные пациентов (фамилия, имя, отчество) должны вноситься в систему в единообразном виде: первая буква заглавная, остальные строчные

2. Даталогическое описание работы триггеров

Описанные выше триггеры будут работать следующим образом:

- 1) Наличие в схеме базы данных таблицы История болезней свидетельствует о денормализованном состоянии базы данных: данные, находящиеся в таблице Лечение дублируются также и в таблицу История болезней. Для поддержания целостности данных требуется добавить триггер, который будет после добавления/обновления данных в таблице Лечение вносить соответствующие изменения в таблицу История болезней.
- 2) При добавлении записи в таблицу Лечение в соответствующей ей записи в таблице Запись на лечение флаг completed устанавливается равным 1.
- 3) При удалении записей из таблиц Поликлиника и Больница автоматически должна удалиться соответствующая запись из таблицы Медицинские организации.
- 4) При добавлении и обновлении персональных данных пациентов и сотрудников значения в поля Фамилия, Имя, Отчество заносятся с заглавной буквы для унифицированного представления.

3. Код триггеров и фрагменты кода приложения, создающие триггеры

Код триггеров:

1.1	<pre>CREATE TRIGGER treatment_after_insert AFTER INSERT \ ON Treatment \ FOR EACH ROW \ INSERT INTO Diseases_history(patient_id, disease, date_of_diagnosis) \ VALUES (NEW.patient_id, NEW.disease, NEW.date);</pre>
1.2	<pre>CREATE TRIGGER treatment_after_update AFTER UPDATE \ ON Treatment \ FOR EACH ROW \ UPDATE Diseases_history SET patient_id = NEW.patient_id, \ disease = NEW.disease, date_of_diagnosis = NEW.date \ WHERE patient_id = OLD.patient_id and disease = OLD.disease \ and date_of_diagnosis = OLD.date</pre>
1.3	<pre>CREATE TRIGGER complete_treatment AFTER INSERT \ ON Treatment \ FOR EACH ROW \</pre>

	<pre> UPDATE Treatment_registration SET completed = 1 \ WHERE Treatment_registration.patient_id = NEW.patient_id and Treatment_registration.doctor_id = NEW.doctor_id \ and Treatment_registration.medical_organisation_id = NEW.medical_organisation_id and Treatment_registration.date = NEW.date </pre>
2.1	<pre> CREATE TRIGGER polyclinic_after_delete AFTER DELETE \ ON Polyclinic \ FOR EACH ROW \ DELETE FROM Medical_organisation \ WHERE OLD.medical_organisation_id = Medical_organisation.medical_organisation_id; </pre>
2.2	<pre> CREATE TRIGGER hospital_after_delete AFTER DELETE \ ON Hospital \ FOR EACH ROW \ DELETE FROM Medical_organisation \ WHERE OLD.medical_organisation_id = Medical_organisation.medical_organisation_id; </pre>
3.1	<pre> CREATE TRIGGER doctor_name_before_insert \ BEFORE INSERT ON Doctor \ FOR EACH ROW \ SET NEW.last_name = CONCAT(UCASE(SUBSTRING(NEW.last_name, 1, 1)),LCASE(SUBSTRING(NEW.last_name, 2))), \ NEW.name = CONCAT(UCASE(SUBSTRING(NEW.name, 1, 1)),LCASE(SUBSTRING(NEW.name, 2))), \ NEW.patronymic = CONCAT(UCASE(SUBSTRING(NEW.patronymic, 1, 1)),LCASE(SUBSTRING(NEW.patronymic, 2))); </pre>
3.2	<pre> CREATE TRIGGER patient_name_before_insert \ BEFORE INSERT ON Patient \ FOR EACH ROW \ SET NEW.last_name = CONCAT(UCASE(SUBSTRING(NEW.last_name, 1, 1)),LCASE(SUBSTRING(NEW.last_name, 2))), \ NEW.name = CONCAT(UCASE(SUBSTRING(NEW.name, 1, 1)),LCASE(SUBSTRING(NEW.name, 2))), \ NEW.patronymic = CONCAT(UCASE(SUBSTRING(NEW.patronymic, 1, 1)),LCASE(SUBSTRING(NEW.patronymic, 2))); </pre>

Фрагменты кода приложения, создающие триггеры:

1.1	<pre> mycursor.execute("CREATE TRIGGER treatment_after_insert AFTER INSERT \ ON Treatment \ </pre>
-----	--

	<pre> FOR EACH ROW \ INSERT INTO Diseases_history(patient_id, disease, date_of_diagnosis) \ VALUES (NEW.patient_id, NEW.disease, NEW.date);") </pre>
1.2	<pre> mycursor.execute("CREATE TRIGGER treatment_after_update AFTER UPDATE \ ON Treatment \ FOR EACH ROW \ UPDATE Diseases_history SET patient_id = NEW.patient_id, \ disease = NEW.disease, date_of_diagnosis = NEW.date \ WHERE patient_id = OLD.patient_id and disease = OLD.disease \ and date_of_diagnosis = OLD.date;") </pre>
1.3	<pre> mycursor.execute("CREATE TRIGGER complete_treatment AFTER INSERT \ ON Treatment \ FOR EACH ROW \ UPDATE Treatment_registration SET completed = 1 \ WHERE Treatment_registration.patient_id = NEW.patient_id and Treatment_registration.doctor_id = NEW.doctor_id \ and Treatment_registration.medical_organisation_id = NEW.medical_organisation_id and Treatment_registration.date = NEW.date") </pre>
2.1	<pre> mycursor.execute("CREATE TRIGGER polyclinic_after_delete AFTER DELETE \ ON Polyclinic \ FOR EACH ROW \ DELETE FROM Medical_organisation \ WHERE OLD.medical_organisation_id = Medical_organisation.medical_organisation_id;") </pre>
2.2	<pre> mycursor.execute("CREATE TRIGGER hospital_after_delete AFTER DELETE \ ON Hospital \ FOR EACH ROW \ DELETE FROM Medical_organisation \ WHERE OLD.medical_organisation_id = Medical_organisation.medical_organisation_id;") </pre>
3.1	<pre> mycursor.execute("CREATE TRIGGER doctor_name_before_insert \ BEFORE INSERT ON Doctor \ FOR EACH ROW \ SET NEW.last_name = CONCAT(UCASE(SUBSTRING(NEW.last_name, 1, 1)),LCASE(SUBSTRING(NEW.last_name, 2))), \ NEW.name = CONCAT(UCASE(SUBSTRING(NEW.name, 1, 1)),LCASE(SUBSTRING(NEW.name, 2))), \ NEW.patronymic = CONCAT(UCASE(SUBSTRING(NEW.patronymic, 1, 1)),LCASE(SUBSTRING(NEW.patronymic, 2)));") </pre>
3.2	<pre> mycursor.execute("CREATE TRIGGER patient_name_before_insert \ BEFORE INSERT ON Patient \ FOR EACH ROW \ SET NEW.last_name = CONCAT(UCASE(SUBSTRING(NEW.last_name, 1, 1)),LCASE(SUBSTRING(NEW.last_name, 2))), \ </pre>

	NEW.name = CONCAT(UCASE(SUBSTRING(NEW.name, 1, 1)),LCASE(SUBSTRING(NEW.name, 2))),\ NEW.patronymic = CONCAT(UCASE(SUBSTRING(NEW.patronymic, 1, 1)),LCASE(SUBSTRING(NEW.patronymic, 2)));"
--	--

4. Результаты отладки триггеров и их тестирования на тестовом наборе данных

Отладка триггеров через sql-менеджер:

1.1 `INSERT INTO Treatment(patient_id, doctor_id, medical_organisation_id, disease, date)`
`VALUES (3, 7, 1, 'ОРЗ', '2024-03-18');`

`SELECT * FROM Treatment;`

treatment_id	patient_id	doctor_id	medical_organisation_id	disease	date
1	1	6	8	Пневмония	2024-02-27
2	3	2	2	Грипп	2024-01-08
3	2	3	9	Туберкулез	2024-03-12
4	3	7	10	Перелом руки	2024-03-13
5	4	10	4	Ангина	2024-03-06
8	1	9	3	Стоматит	2024-03-17
9	3	7	4	Герпесная инфекция	2024-02-05
10	3	5	1	ОРВИ	2024-03-15
11	3	7	1	ОРЗ	2024-03-18

`SELECT * FROM Diseases_history;`

record_id	patient_id	disease	date_of_diagnosis
1	1	Пневмония	2024-02-27
2	3	Грипп	2024-01-08
3	2	Туберкулез	2024-03-12
4	3	Перелом руки	2024-03-13
5	4	Ангина	2024-03-06
6	2	Миопия	2024-01-13
7	2	Аллергия	2024-02-04
8	1	Стоматит	2024-03-17
9	3	Герпесная инфекция	2024-02-05
10	3	ОРВИ	2024-03-15
11	3	ОРЗ	2024-03-18

+-----+-----+-----+-----+

1.2 UPDATE Treatment SET patient_id = 1 WHERE doctor_id = 5;

SELECT * FROM Treatment;

+-----+-----+-----+-----+-----+-----+
| treatment_id | patient_id | doctor_id | medical_organisation_id | disease | date |
+-----+-----+-----+-----+-----+-----+
1	1	6	8	Пневмония	2024-02-27
2	3	2	2	Грипп	2024-01-08
3	2	3	9	Туберкулез	2024-03-12
4	3	7	10	Перелом руки	2024-03-13
5	4	10	4	Ангина	2024-03-06
8	1	9	3	Стоматит	2024-03-17
9	3	7	4	Герпесная инфекция	2024-02-05
10	1	5	1	ОРВИ	2024-03-15
11	3	7	1	ОРЗ	2024-03-18
+-----+-----+-----+-----+-----+-----+

SELECT * FROM Diseases_history;

+-----+-----+-----+-----+
| record_id | patient_id | disease | date_of_diagnosis |
+-----+-----+-----+-----+
1	1	Пневмония	2024-02-27
2	3	Грипп	2024-01-08
3	2	Туберкулез	2024-03-12
4	3	Перелом руки	2024-03-13
5	4	Ангина	2024-03-06
6	2	Миопия	2024-01-13
7	2	Аллергия	2024-02-04
8	1	Стоматит	2024-03-17
9	3	Герпесная инфекция	2024-02-05
10	1	ОРВИ	2024-03-15
11	3	ОРЗ	2024-03-18
+-----+-----+-----+-----+

1.3 insert into treatment(patient_id, doctor_id, medical_organisation_id, disease, date) VALUES (4, 4, 'Ангина', '2024-05-19');

select * from treatment_registration;

+-----+-----+-----+-----+-----+-----+
| registration_id | patient_id | doctor_id | medical_organisation_id | date | completed |
+-----+-----+-----+-----+-----+-----+
| 1 | 1 | 6 | 8 | 2024-02-27 | 1 |
+-----+-----+-----+-----+-----+-----+

2	3	2	2	2024-01-07	0
...					
22	1	2	2	2024-05-04	0
23	4	4	4	2024-05-19	1
+-----+-----+-----+-----+-----+					

2.1 **DELETE FROM Polyclinic**
WHERE medical_organisation_id = 6;

SELECT* FROM Polyclinic;

medical_organisation_id	district
+-----+-----+	
7	Советский
8	Октябрьский
9	Советский
10	Советский
+-----+-----+	

SELECT* FROM Medical_organisation;

medical_organisation_id	name	address	phone
+-----+-----+-----+-----+			
1	Центральная клиническая больница	Пирогова, 25/1	73833304321
2	Бердская центральная городская больница	Новосибирская, 10	73834155610
3	Городская клиническая больница 12	Морской проспект, 25	88002000200
4	Новосибирская клиническая центральная районная больница	Магистральная, 3а	79061959432
5	Городская клиническая больница 2	Ползунова, 21	73833639507
7	ЦНМТ	Пирогова, 25/4	73833630183
8	Поликлиническое отделение 1	Шукшина, 3	73833389747
9	Городская клиническая поликлиника 14	Демакова, 2	73833047444
10	Клиника Санитас	Николаева, 12/3	73832336600
+-----+-----+-----+-----+			

2.2 **DELETE FROM Hospital**
WHERE medical_organisation_id = 5;

SELECT* FROM Hospital;

medical_organisation_id	medical_corps_amount
+-----+-----+	
1	2
2	4

3	1
4	1

```
SELECT* FROM Medical_organisation;
```

medical_organisation_id	name	address	phone
1	Центральная клиническая больница	Пирогова, 25/1	73833304321
2	Бердская центральная городская больница	Новосибирская, 10	73834155610
3	Городская клиническая больница 12	Морской проспект, 25	88002000200
4	Новосибирская клиническая центральная районная больница	Магистральная, 3а	79061959432
7	ЦНМТ	Пирогова, 25/4	73833630183
8	Поликлиническое отделение 1	Шукшина, 3	73833389747
9	Городская клиническая поликлиника 14	Демакова, 2	73833047444
10	Клиника Санитас	Николаева, 12/3	73832336600

3.1

```
INSERT INTO Doctor(last_name, name, patronymic, phone, work_experience)
VALUES ('Ключихин', 'ИГОРЬ', 'ПЕТРОВИЧ', '75617492759', 50);
```

```
SELECT * FROM Doctor;
```

doctor_id	last_name	name	patronymic	phone	work_experience
1	Иванов	Иван	Иванович	79991234567	650
2	Петров	Петр	Петрович	79876543210	4200
3	Сидорова	Ольга	Владимировна	79261112233	300
4	Козлов	Алексей	Сергеевич	79105554433	3900
5	Михайлова	Екатерина	Андреевна	79157778899	3200
6	Михайлов	Семен	Игоревич	73583874861	1040
7	Иванова	Анна	Сергеевна	79123456789	879
8	Петров	Игорь	Владимирович	79234567890	20
9	Сидорова	Елена	Александровна	79345678901	410
10	Козлов	Артем	Дмитриевич	79456789012	5450
11	Михайлова	Ольга	Николаевна	79567890123	4000
12	Ключихин	Игорь	Петрович	75617492759	50

3.2

```
INSERT INTO Patient(last_name, name, patronymic, birthday,
phone, address, policy_number, polyclinic_id)
VALUES ('Кошечкина', 'МАРиНа', 'АЛЕКСАНДРОВНА', '2000-02-28', '72856154818', 'Ленина
3', '1234927591759174', 8);
```

```
SELECT * FROM Patient;
```

```
+-----+-----+-----+-----+-----+-----+-----+-----+
| patient_id | last_name | name | patronymic | birthday | phone | address | policy_number | polyclinic_id |
|-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | Смирнова | Ольга | Александровна | 1987-11-25 | 79998887766 | город Новосибирск, ул. Кирова 20 | 1234567891234567 | 7 |
| 2 | Кузнецов | Игорь | Петрович | 1995-04-03 | 79876543210 | город Новосибирск, пр. Ленина 25 | 9876543219876543 | 9 |
| 3 | Николаева | Мария | Владимировна | 1980-09-15 | 79112223344 | город Новосибирск, ул. Пушкина 5 | 5432167895432167 | NULL |
| 4 | Павлов | Артем | Сергеевич | 1983-06-30 | 79255554433 | город Новосибирск, ул. Гагарина 15 | 2468135792468135 | 8 |
| 5 | Лебедев | Иван | Дмитриевич | 1979-02-20 | 79167778899 | город Новосибирск, пр. Ленина 30 | 1357924681357924 | 10 |
| 6 | Кошечкина | Марина | Александровна | 2000-02-28 | 72856154818 | Ленина 3 | 1234927591759174 | 8 |
+-----+-----+-----+-----+-----+-----+-----+-----+
```

Активный сервер. Хранимые процедуры и функции.

1. Описание бизнес-транзакций, требующих использования хранимых процедур и функций

В системе присутствуют следующие бизнес-транзакции, требующие использования хранимых процедур и функций:

- контроль количества обслуживаемых пациентов в конкретной медицинской организации за заданный промежуток времени
- просмотр списка пациентов, записавшихся на лечение к определенному врачу в определенную медицинскую организацию (врач может одновременно работать в нескольких медицинских организациях)
- подсчет количества пациентов, перенесших определенное заболевание

2. Даталогическое описание работы хранимых процедур и функций

Описанные выше хранимые процедуры и функции будут работать следующим образом:

- процедура будет принимать идентификатор медицинской организации, начальную и конечную даты в качестве параметров и выдавать количество пациентов, обслуженных медицинской организацией за заданный период времени
- процедура будет принимать идентификатор врача и медицинской организации и возвращать список пациентов, записанных на лечение к указанному врачу на текущую дату (когда была вызвана процедура)
- функция будет принимать название заболевания и возвращать число - количество пациентов, у которых было диагностировано данное заболевание

3. Код хранимых процедур и функций и фрагменты кода приложения, создающие и использующие хранимые процедуры и функции

Код хранимых процедур и функций:

1	<pre>CREATE PROCEDURE medical_organisations_db.get_treatment_statistics(IN current_medical_organisation_id INT, IN start_date DATE, IN end_date DATE) SELECT * FROM Treatment WHERE medical_organisation_id = current_medical_organisation_id AND (DATE(date) BETWEEN DATE(start_date) AND DATE(end_date));</pre>
2	<pre>CREATE PROCEDURE medical_organisations_db.get_registered_patients_list(\ IN current_doctor_id INT, IN current_medical_organisation_id INT) SELECT Treatment_registration.registration_id, Treatment_registration.patient_id, Patient.last_name, Patient.name, Patient.patronymic, Treatment_registration.date FROM Treatment_registration INNER JOIN Patient ON Treatment_registration.patient_id = Patient.patient_id WHERE doctor_id = current_doctor_id AND medical_organisation_id = current_medical_organisation_id AND date = CURRENT_DATE();</pre>
3	<pre>CREATE FUNCTION get_patients_count_by_disease(current_disease VARCHAR(100)) RETURNS INT READS SQL DATA BEGIN DECLARE patients_count INT; SELECT COUNT(DISTINCT patient_id) INTO patients_count FROM Treatment WHERE disease = current_disease; RETURN patients_count; END</pre>

Фрагменты кода приложения, создающие и использующие хранимые процедуры и функции:

1	<pre> mycursor.execute("CREATE PROCEDURE medical_organisations_db.get_treatment_statistics(\ IN current_medical_organisation_id INT, \ IN start_date DATE, \ IN end_date DATE) \ SELECT * \ FROM Treatment \ WHERE medical_organisation_id = current_medical_organisation_id \ AND (DATE(date) BETWEEN DATE(start_date) AND DATE(end_date));") ... val = (current_medical_organisation_id, start_date, end_date) mycursor.callproc("get_treatment_statistics", val) for result in mycursor.stored_results(): rows = result.fetchall() </pre>
2	<pre> mycursor.execute("CREATE PROCEDURE medical_organisations_db.get_registered_patients_list(\ IN current_doctor_id INT, \ IN current_medical_organisation_id INT) \ SELECT Treatment_registration.registration_id, Treatment_registration.patient_id, \ Patient.last_name, Patient.name, Patient.patronymic, Treatment_registration.date \ FROM Treatment_registration \ INNER JOIN Patient ON Treatment_registration.patient_id = Patient.patient_id \ WHERE doctor_id = current_doctor_id \ AND medical_organisation_id = current_medical_organisation_id \ AND date = CURRENT_DATE();") ... val = (current_doctor_id, current_medical_organisation_id) mycursor.callproc("get_registered_patients_list", val) for result in mycursor.stored_results(): rows = result.fetchall() </pre>
3	<pre> mycursor.execute("SET autocommit = 0") create_function_query = "" CREATE FUNCTION get_patients_count_by_disease(current_disease VARCHAR(100)) RETURNS INT READS SQL DATA </pre>


```

BEGIN
    DECLARE patients_count INT;
    SELECT COUNT(DISTINCT patient_id) INTO patients_count
    FROM Treatment
    WHERE disease = current_disease;
    RETURN patients_count;
END
"""
mycursor.execute(create_function_query)
mycursor.execute("SET autocommit = 1")

...

sql = "SELECT get_patients_count_by_disease(%s);"
val = (disease, )
mycursor.execute(sql, val)
rows = mycursor.fetchall()

```

4. Результаты отладки хранимых процедур и функций и их тестирования на тестовом наборе данных

Отладка через sql-менеджер:

1) `SELECT * from Treatment;`

treatment_id	patient_id	doctor_id	medical_organisation_id	disease	date
1	1	6	8	Пневмония	2024-02-27
2	3	2	2	Грипп	2024-01-08
3	2	3	9	Туберкулез	2024-03-12
4	3	7	10	Перелом руки	2024-03-13
5	4	10	4	Ангина	2024-03-06
8	1	9	3	Стоматит	2024-03-17
9	3	7	4	Герпесная инфекция	2024-02-05
10	1	5	1	ОРВИ	2024-03-15
11	3	7	1	ОРЗ	2024-03-18

`CALL get_treatment_statistics(1, '2024-03-01', '2024-03-16');`

treatment_id	patient_id	doctor_id	medical_organisation_id	disease	date
10	1	5	1	ОРВИ	2024-03-15

Результат выполнения в приложении:

Получить количество обслуженных пациентов за промежуток времени

Идентификатор медицинской организации:

4

Начальная дата:

2024-01-01

Конечная дата:

2024-04-01

treatment_id; patient_id; doctor_id; medical_organisation_id; disease; date

5; 4; 10; 4; Ангина; 2024-03-06;

9; 3; 7; 4; Герпесная инфекция; 2024-02-05;

2) SELECT * FROM Treatment_registration;

registration_id	patient_id	doctor_id	medical_organisation_id	date
1	1	6	8	2024-02-27
2	3	2	2	2024-03-24
3	2	3	9	2024-03-12
4	3	7	10	2024-03-24
5	4	10	4	2024-03-06
6	4	8	6	2024-01-11
7	2	4	5	2024-02-04
8	1	9	3	2024-03-17
9	3	7	4	2024-03-24
10	3	5	1	2024-03-24

CALL get_registered_patients_list(7, 4);

registration_id	patient_id	last_name	name	patronymic	date
9	3	Николаева	Мария	Владимировна	2024-03-24

Результат выполнения в приложении:

Получить список записавшихся пациентов

Идентификатор врача:

7

Идентификатор медицинской организации:

4

registration_id; patient_id; last_name; name; patronymic; date

9; 3; Николаева; Мария; Владимировна; 2024-03-31


3) SELECT * FROM Treatment;

treatment_id	patient_id	doctor_id	medical_organisation_id	disease	date
1	1	6	8	Пневмония	2024-02-27
2	3	2	2	Грипп	2024-01-08
3	2	3	9	Туберкулез	2024-03-12
4	3	7	10	Перелом руки	2024-03-13
5	4	10	4	Ангина	2024-03-06

8	1	9	3	Стоматит	2024-03-17
9	3	7	4	Герпесная инфекция	2024-02-05
10	1	5	1	Грипп	2024-03-15
11	3	7	1	ОРЗ	2024-03-18

```
SELECT get_patients_count_by_disease('Грипп');
+-----+
| get_patients_count_by_disease('Грипп') |
+-----+
|                2 |
+-----+
```

Результат выполнения в приложении:


Получить список пациентов, перенесших заболевание

Название болезни:

Грипп

Число пациентов, перенесших заболевание

1

Внешние схемы и интерфейсы

В системе реализованы внешние схемы для 4-ех ролей:

Роль 1. Пациент

- поиск информации о врачах: представление, связывающее таблицы Doctor, Doctor_Medical_organisation и Medical_organisation, доступное только для чтения.
- поиск адреса медицинского учреждения, поиск номера телефона медицинского учреждения: представления, связывающие таблицы Polyclinic и Hospital с таблицей Medical_organisation, доступные только для чтения.
- Запись на амбулаторное лечение: зеркальное представление таблицы Treatment_registration, доступное для записи и чтения только тех записей в таблице, которые относятся к данному пациенту.

Роль 2. Руководитель медицинской организации

- контроль количества обслуживаемых пациентов: вызов хранимой процедуры get_treatment_statistics().
- контроль заболеваемости: вызов хранимой процедуры get_patients_count_by_disease().
- принятие сотрудника на работу: представление, связывающее таблицы Doctor и Doctor_Medical_organisation, доступное для чтения и записи только тех записей, которые относятся к медицинской организации, которой руководит данный пользователь.
- увольнение сотрудника: представление, связывающее таблицы Doctor и Doctor_Medical_organisation, доступное для чтения и записи только тех записей, которые относятся к медицинской организации, которой руководит данный пользователь.

Роль 3. Врач

- внесение персональных данных пациента: зеркальное представление таблицы Patient.
- просмотр списка записавшихся на лечение пациентов: вызов хранимой процедуры get_registered_patients_list().
- добавление информации о лечении пациентов: хранимая процедура add_treatment().
- ведение учета истории болезней: происходит автоматически за счет срабатывания триггера.

Роль 4. Администратор системы

Объекты схемы базы данных, реализующие внешние схемы

Роль 1. Пациент

Тип	Назначение	Код
Представление	Предоставляет доступ на чтение части информации о врачах (пациенту достаточно фамилии, имени и отчества врача и медицинской организации, где он работает)	CREATE VIEW Doctor_PatientUser_View AS SELECT d.doctor_id, d.last_name, d.name, d.patronymic, mo.name as medical_organisation_name, mo.address

		FROM Doctor as d INNER JOIN Doctor_Medical_organisation as dmo ON d.doctor_id = dmo.doctor_id INNER JOIN Medical_organisation as mo ON dmo.medical_organisation_id = mo.medical_organisation_id
Представление	Предоставляет доступ на чтение информации о поликлиниках, совмещенной с общей информацией о медицинских организациях	CREATE VIEW Polyclinic_PatientUser_View AS SELECT Polyclinic.medical_organisation_id, name, address, phone, district FROM Medical_organisation INNER JOIN Polyclinic ON Medical_organisation.medical_organisation_id = Polyclinic.medical_organisation_id
Представление	Предоставляет доступ на чтение информации о больницах, совмещенной с общей информацией о медицинских организациях	CREATE VIEW Hospital_PatientUser_View \ AS SELECT Hospital.medical_organisation_id, \ name, \ address, \ phone, \ medical_corps_amount \ FROM Medical_organisation INNER JOIN Hospital ON Medical_organisation.medical_organisation_id = Hospital.medical_organisation_id
Представление	Зеркальное представление таблицы для записи на приём	CREATE VIEW Treatment_registration_PatientUser_View AS SELECT patient_id, doctor_id, medical_organisation_id, date FROM Treatment_registration
Хранимая процедура	Выдает актуальный список записей на приём для текущего пациента	CREATE PROCEDURE medical_organisations_db.get_patients_registrations(IN patient_id INT) SELECT d.last_name, d.name, d.patronymic, mo.name as medical_organisation_name, mo.address, t.date FROM Treatment_registration as t INNER JOIN Doctor as d ON t.doctor_id = d.doctor_id INNER JOIN Medical_organisation as mo ON t.medical_organisation_id = mo.medical_organisation_id WHERE t.patient_id = patient_id AND date >= CURRENT_DATE();

Хранимая процедура	Выдает список врачей, работающих в выбранной медицинской организации	<pre>CREATE PROCEDURE medical_organisations_db.get_doctors_by_medical_organisation(IN medical_organisation_id INT) SELECT d.doctor_id, d.last_name, d.name, d.patronymic FROM Doctor_Medical_organisation as dmo INNER JOIN Doctor as d ON dmo.doctor_id = d.doctor_id INNER JOIN Medical_organisation as mo ON dmo.medical_organisation_id = mo.medical_organisation_id WHERE mo.medical_organisation_id = medical_organisation_id;</pre>
Хранимая процедура	Возвращает идентификатор записи в таблице Patient, который соответствует текущему пользователю-пациенту	<pre>mycursor.execute("CREATE PROCEDURE medical_organisations_db.get_patient_user_id(\ login varchar(100) \) \ SELECT patient_id \ FROM User_patient AS u \ WHERE u.login = login;")</pre>

Роль 2. Руководитель медицинской организации

Тип	Назначение	Код
Представление	Предоставляет полный доступ на чтение и запись к данным о врачах	<pre>CREATE VIEW Doctor_MedicalOrganisationUser_View AS SELECT Doctor.doctor_id, last_name, name, patronymic, phone, work_experience, medical_organisation_id FROM Doctor INNER JOIN Doctor_medical_organisation on Doctor.doctor_id = Doctor_Medical_organisation.doctor_id</pre>
Хранимая процедура	Связывает существующую запись врача с текущей медицинской организацией	<pre>CREATE PROCEDURE medical_organisations_db.add_existed_doctor(IN doctor_id INT, IN medical_organisation_id INT) INSERT INTO Doctor_Medical_Organisation(doctor_id, medical_organisation_id, hire_date) VALUES(doctor_id, medical_organisation_id, CURRENT_DATE());</pre>
Хранимая процедура	Обновляет данные о враче,	CREATE PROCEDURE

	<p>работающем в текущей медицинской организации</p>	<pre> medical_organisations_db.update_doctor_in_medical_o rganisation(doctor_id int, last_name varchar(50), name varchar(50), patronymic varchar(50), phone varchar(11), work_experience int, medical_organisation_id int) UPDATE Doctor AS d INNER JOIN Doctor_Medical_organisation AS dmo ON d.doctor_id = dmo.doctor_id SET d.last_name = last_name, d.name = name, d.patronymic = patronymic, d.phone = phone, d.work_experience = work_experience, dmo.medical_organisation_id = medical_organisation_id WHERE d.doctor_id = doctor_id AND d.doctor_id = dmo.doctor_id; </pre>
Хранимая процедура	<p>Удаляет врача из сотрудников текущей медицинской организации</p>	<pre> mycursor.execute("SET autocommit = 0") create_function_query = "" CREATE PROCEDURE medical_organisations_db.delete_doctor_from_medical _organisation(IN doctor_id INT) BEGIN DECLARE deleted_user_login VARCHAR(255); DELETE FROM Doctor_Medical_organisation WHERE Doctor_Medical_organisation.doctor_id = doctor_id; SELECT login INTO deleted_user_login FROM User_doctor WHERE User_doctor.doctor_id = doctor_id; DELETE FROM User_doctor WHERE User_doctor.login = deleted_user_login AND User_doctor.doctor_id = doctor_id; SET @create_user_query = CONCAT('DROP USER IF EXISTS ', deleted_user_login, '@\\\\"localhost\\\\"'); PREPARE create_user_stmt FROM @create_user_query; EXECUTE create_user_stmt; </pre>

		<pre> DEALLOCATE PREPARE create_user_stmt; COMMIT; END; """" mycursor.execute(create_function_query) mycursor.execute("SET autocommit = 1") </pre>
Хранимая процедура	Добавляет врача в качестве сотрудника текущей медицинской организации	<pre> mycursor.execute("SET autocommit = 0") create_function_query = """" CREATE PROCEDURE medical_organisations_db.insert_doctor_to_medical_or ganisation(last_name varchar(50), name varchar(50), patronymic varchar(50), phone varchar(11), work_experience int, medical_organisation_id int) BEGIN DECLARE new_id INT; DECLARE new_user_name VARCHAR(150); INSERT INTO Doctor(last_name, name, patronymic, phone, work_experience) VALUES(last_name, name, patronymic, phone, work_experience); SET new_id = LAST_INSERT_ID(); INSERT INTO Doctor_Medical_organisation(doctor_id, medical_organisation_id, hire_date) VALUES(new_id, medical_organisation_id, CURRENT_DATE()); SET new_user_name = CONCAT(last_name, '_ ', name, '_ ', new_id); SET @create_user_query = CONCAT('CREATE USER ', new_user_name, '@\\\\"localhost\\\\" IDENTIFIED BY \\\\'', new_user_name, '\\\\"'); PREPARE create_user_stmt FROM @create_user_query; EXECUTE create_user_stmt; DEALLOCATE PREPARE create_user_stmt; SET @create_user_query = CONCAT('GRANT \\\\'doctor_role\\\\" TO ', new_user_name, '@\\\\"localhost\\\\"'); PREPARE create_user_stmt FROM @create_user_query; </pre>

		<pre> EXECUTE create_user_stmt; DEALLOCATE PREPARE create_user_stmt; SET @create_user_query = CONCAT('SET DEFAULT ROLE ALL TO ', new_user_name, '@\\\\"localhost\\\\"'); PREPARE create_user_stmt FROM @create_user_query; EXECUTE create_user_stmt; DEALLOCATE PREPARE create_user_stmt; INSERT INTO User_doctor(login, doctor_id) VALUES(new_user_name, new_id); COMMIT; END ===== mycursor.execute(create_function_query) mycursor.execute("SET autocommit = 1") </pre>
Хранимая функция	Выдает количество пациентов текущей медицинской организации, которые проходили лечение выбранного заболевания	<pre> CREATE FUNCTION get_patients_count_by_disease(current_disease VARCHAR(100), current_medical_organisation_id INT) RETURNS INT READS SQL DATA BEGIN DECLARE patients_count INT; SELECT COUNT(DISTINCT patient_id) INTO patients_count FROM Treatment WHERE disease = current_disease AND medical_organisation_id = current_medical_organisation_id; RETURN patients_count; END </pre>
Хранимая процедура	Выдает список лечений, произведенных текущей медицинской организацией за выбранный промежуток времени	<pre> CREATE PROCEDURE medical_organisations_db.get_treatment_statistics(IN current_medical_organisation_id INT, IN start_date DATE, IN end_date DATE) SELECT p.last_name AS patient_last_name, p.name AS patient_name, p.patronymic AS patient_patronymic, d.last_name AS doctor_last_name, d.name AS doctor_name, d.patronymic AS doctor_patronymic, disease, date FROM Treatment t INNER JOIN Patient p ON t.patient_id = p.patient_id INNER JOIN Doctor d ON t.doctor_id = d.doctor_id INNER JOIN Medical_organisation mo </pre>

		ON t.medical_organisation_id = mo.medical_organisation_id WHERE t.medical_organisation_id = current_medical_organisation_id AND (DATE(date) BETWEEN DATE(start_date) AND DATE(end_date));
Хранимая процедура	Возвращает описание лечения по идентификаторы записи	mycursor.execute("CREATE PROCEDURE medical_organisations_db.get_treatment_description(\ IN treatment_id INT) \ SELECT t.description \ FROM Treatment t \ WHERE t.treatment_id = treatment_id")
Хранимая процедура	Возвращает идентификатор записи в таблице Medical_organisation, который соответствует текущему пользователю-руководителю	mycursor.execute("CREATE PROCEDURE medical_organisations_db.get_doctor_user_id(\ login varchar(100) \) \ SELECT doctor_id \ FROM User_doctor AS u \ WHERE u.login = login;")
Хранимая процедура	Сохраняет новую запись в таблице Medical_organisation и Polyclinic, создает для нее пользователя с ролью Руководитель медицинской организации	mycursor.execute("SET autocommit = 0") create_function_query = "" CREATE PROCEDURE medical_organisations_db.add_polyclinic(name varchar(100), address varchar(100), phone varchar(11), district varchar(50)) BEGIN DECLARE new_id INT; DECLARE new_user_name VARCHAR(150); DECLARE EXIT HANDLER FOR SQLEXCEPTION BEGIN ROLLBACK; RESIGNAL SET MESSAGE_TEXT = 'Error creating user'; END; START TRANSACTION; INSERT INTO Medical_organisation(name, address, phone) VALUES(name, address, phone); SET new_id = LAST_INSERT_ID(); INSERT INTO

		<pre>Polyclinic(medical_organisation_id, district) VALUES(new_id, district); SET new_user_name = CONCAT(name, '_', new_id); BEGIN DECLARE user_count INT; DECLARE CONTINUE HANDLER FOR SQLSTATE '42000' BEGIN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Error creating user.'; END; SET user_count = (SELECT COUNT(*) FROM mysql.user WHERE User = new_user_name); IF user_count > 0 THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Error creating user.'; ELSE SET @create_user_query = CONCAT('CREATE USER ', new_user_name, '@\\'localhost\\' IDENTIFIED BY \\', new_user_name, '\\'); PREPARE create_user_stmt FROM @create_user_query; EXECUTE create_user_stmt; DEALLOCATE PREPARE create_user_stmt; SET @create_user_query = CONCAT('GRANT \\'medical_organisation_role\\' TO ', new_user_name, '@\\'localhost\\'); PREPARE create_user_stmt FROM @create_user_query; EXECUTE create_user_stmt; DEALLOCATE PREPARE create_user_stmt; SET @create_user_query = CONCAT('SET DEFAULT ROLE ALL TO ', new_user_name, '@\\'localhost\\'); PREPARE create_user_stmt FROM @create_user_query; EXECUTE create_user_stmt; DEALLOCATE PREPARE create_user_stmt; INSERT INTO User_medical_organisation(login, medical_organisation_id) VALUES(new_user_name,</pre>
--	--	---

		<pre> new_id); COMMIT; END IF; END; END; """" mycursor.execute(create_function_query) mycursor.execute("SET autocommit = 1") </pre>
Хранимая процедура	<p>Сохраняет новую запись в таблице Medical_organisation и Hospital, создает для нее пользователя с ролью Руководитель медицинской организации</p>	<pre> mycursor.execute("SET autocommit = 0") create_function_query = """" CREATE PROCEDURE medical_organisations_db.add_hospital(name varchar(100), address varchar(100), phone varchar(11), medical_corps_amount varchar(50)) BEGIN DECLARE new_id INT; DECLARE new_user_name VARCHAR(150); DECLARE EXIT HANDLER FOR SQLEXCEPTION BEGIN ROLLBACK; RESIGNAL SET MESSAGE_TEXT = 'Error creating user'; END; START TRANSACTION; INSERT INTO Medical_organisation(name, address, phone) VALUES(name, address, phone); SET new_id = LAST_INSERT_ID(); INSERT INTO Hospital(medical_organisation_id, medical_corps_amount) VALUES(new_id, medical_corps_amount); SET new_user_name = CONCAT(name, '_ ', new_id); BEGIN DECLARE user_count INT; DECLARE CONTINUE HANDLER FOR SQLSTATE '42000' BEGIN SIGNAL SQLSTATE '45000' SET </pre>

		<pre> MESSAGE_TEXT = 'Error creating user.'; END; SET user_count = (SELECT COUNT(*) FROM mysql.user WHERE User = new_user_name); IF user_count > 0 THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Error creating user.'; ELSE SET @create_user_query = CONCAT('CREATE USER ', new_user_name, '@\\'localhost\\' IDENTIFIED BY \\', new_user_name, '\\'); PREPARE create_user_stmt FROM @create_user_query; EXECUTE create_user_stmt; DEALLOCATE PREPARE create_user_stmt; SET @create_user_query = CONCAT('GRANT \\'medical_organisation_role\\' TO ', new_user_name, '@\\'localhost\\'); PREPARE create_user_stmt FROM @create_user_query; EXECUTE create_user_stmt; DEALLOCATE PREPARE create_user_stmt; SET @create_user_query = CONCAT('SET DEFAULT ROLE ALL TO ', new_user_name, '@\\'localhost\\'); PREPARE create_user_stmt FROM @create_user_query; EXECUTE create_user_stmt; DEALLOCATE PREPARE create_user_stmt; INSERT INTO User_medical_organisation(login, medical_organisation_id) VALUES(new_user_name, new_id); COMMIT; END IF; END; END; "" mycursor.execute(create_function_query) mycursor.execute("SET autocommit = 1") </pre>
Хранимая процедура	Удаляет данные о медицинской организации и	<pre> mycursor.execute("SET autocommit = 0") create_function_query = "" </pre>

	пользователя, которые с ней связан	<pre> CREATE PROCEDURE medical_organisations_db.delete_medical_organisation (IN medical_organisation_id INT) BEGIN DECLARE deleted_user_login VARCHAR(255); DECLARE num_records INT; START TRANSACTION; SELECT login INTO deleted_user_login FROM User_medical_organisation WHERE User_medical_organisation.medical_organisation_id = medical_organisation_id; DELETE FROM User_medical_organisation WHERE User_medical_organisation.login = deleted_user_login AND User_medical_organisation.medical_organisation_id = medical_organisation_id; SET @create_user_query = CONCAT('DROP USER IF EXISTS ', deleted_user_login, '@\\\\"localhost\\\\"'); PREPARE create_user_stmt FROM @create_user_query; EXECUTE create_user_stmt; DEALLOCATE PREPARE create_user_stmt; DELETE FROM Medical_organisation WHERE Medical_organisation.medical_organisation_id = medical_organisation_id; COMMIT; END; **** mycursor.execute(create_function_query) mycursor.execute("SET autocommit = 1") </pre>
--	------------------------------------	---

Роль 3. Врач

Тип	Назначение	Код
Представление	Предоставляет доступ на чтение и запись к данным о пациентах, а также о поликлинике, к которой они прикреплены	<pre> CREATE VIEW Patient_DoctorUser_View AS SELECT p.patient_id, p.last_name, p.name, p.patronymic, p.birthday, </pre>

		<p>p.phone, p.address, p.policy_number, mo.name as Polyclinic_name FROM Patient AS p INNER JOIN Medical_organisation AS mo ON mo.medical_organisation_id = p.polyclinic_id</p>
Представление	Выдает список поликлиник, включая общую информацию о них как о медицинских организациях	<pre>CREATE VIEW Polyclinic_DoctorUser_View AS SELECT Polyclinic.medical_organisation_id, name, address FROM Medical_organisation INNER JOIN Polyclinic ON Medical_organisation.medical_organisation_id = Polyclinic.medical_organisation_id</pre>
Хранимая процедура	Выдает список медицинских организаций, в которых работает текущий врач	<pre>CREATE PROCEDURE medical_organisations_db.get_medical_organisations _by_doctor(IN doctor_id INT) SELECT mo.medical_organisation_id, mo.name, mo.address FROM Doctor_Medical_organisation as dmo INNER JOIN Doctor as d ON dmo.doctor_id = d.doctor_id INNER JOIN Medical_organisation as mo ON dmo.medical_organisation_id = mo.medical_organisation_id WHERE d.doctor_id = doctor_id;</pre>
Хранимая процедура	Выдает список пациентов, записанных на прием к текущему врачу в текущий день	<pre>CREATE PROCEDURE medical_organisations_db.get_registered_patients_list (IN current_doctor_id INT, IN current_medical_organisation_id INT) SELECT Patient.last_name, Patient.name, Patient.patronymic FROM Treatment_registration INNER JOIN Patient ON Treatment_registration.patient_id = Patient.patient_id WHERE doctor_id = current_doctor_id AND medical_organisation_id = current_medical_organisation_id AND date = CURRENT_DATE();</pre>
Хранимая процедура	Добавляет записи о пациентах, включая информацию о поликлинике, к которой они прикреплены	<pre>mycursor.execute("SET autocommit = 0") create_function_query = "" CREATE PROCEDURE medical_organisations_db.add_patient_to_polyclinic(</pre>

		<pre>last_name varchar(50), name varchar(50), patronymic varchar(50), birthday date, phone varchar(11), address varchar(100), policy_number varchar(16), polyclinic_id int) BEGIN DECLARE new_id INT; DECLARE new_user_name VARCHAR(150); INSERT INTO Patient(last_name, name, patronymic, birthday, phone, address, policy_number, polyclinic_id) VALUES(last_name, name, patronymic, birthday, phone, address, policy_number, polyclinic_id); SET new_id = LAST_INSERT_ID(); SET new_user_name = CONCAT(last_name, '_', name, '_', new_id); SET @create_user_query = CONCAT('CREATE USER ', new_user_name, '@\\localhost\\' IDENTIFIED BY '\\', new_user_name, '\\'); PREPARE create_user_stmt FROM @create_user_query; EXECUTE create_user_stmt; DEALLOCATE PREPARE create_user_stmt; SET @create_user_query = CONCAT('GRANT \\patient_role\\ TO ', new_user_name, '@\\localhost\\'); PREPARE create_user_stmt FROM @create_user_query; EXECUTE create_user_stmt; DEALLOCATE PREPARE create_user_stmt; SET @create_user_query = CONCAT('SET DEFAULT ROLE ALL TO ', new_user_name, '@\\localhost\\'); PREPARE create_user_stmt FROM @create_user_query; EXECUTE create_user_stmt; DEALLOCATE PREPARE create_user_stmt; INSERT INTO User_patient(login, patient_id) VALUES(new_user_name, new_id);</pre>
--	--	---

		COMMIT; END """" mycursor.execute(create_function_query) mycursor.execute("SET autocommit = 1")
Хранимая процедура	Обновляет данные о пациенте, включая информацию о поликлинике, к которой он прикреплен	CREATE PROCEDURE medical_organisations_db.update_patient_in_polyclinic(patient_id int, last_name varchar(50), name varchar(50), patronymic varchar(50), birthday date, phone varchar(11), address varchar(100), policy_number varchar(16), polyclinic_id int) UPDATE Patient AS p SET p.last_name = last_name, p.name = name, p.patronymic = patronymic, p.birthday = birthday, p.phone = phone, p.address = address, p.policy_number = policy_number, p.polyclinic_id = IF(polyclinic_id != -1, polyclinic_id, p.polyclinic_id) WHERE p.patient_id = patient_id
Хранимая процедура	Удаляет запись о пациенте	mycursor.execute("SET autocommit = 0") create_function_query = """" CREATE PROCEDURE medical_organisations_db.delete_patient(id int) BEGIN DECLARE deleted_patient_login VARCHAR(255); DELETE FROM Patient WHERE patient_id = id; SELECT login INTO deleted_patient_login FROM User_patient WHERE User_patient.patient_id = id; SET @drop_patient_query = CONCAT('DROP USER IF EXISTS ', deleted_patient_login, '@\\\\"localhost\\"'); PREPARE drop_patient_stmt FROM @drop_patient_query; EXECUTE drop_patient_stmt; DEALLOCATE PREPARE

		<pre> drop_patient_stmt; DELETE FROM User_patient WHERE User_patient.login = deleted_patient_login AND User_patient.patient_id = id; COMMIT; END; """" mycursor.execute(create_function_query) mycursor.execute("SET autocommit = 1") </pre>
Хранимая процедура	Добавляет запись о проведенном лечении с текущей датой	<pre> CREATE PROCEDURE medical_organisations_db.add_treatment(patient_id int, doctor_id int, medical_organisation_id int, disease varchar(100), description varchar(100)) INSERT INTO Treatment(patient_id, doctor_id, medical_organisation_id, disease, date, description) VALUES (patient_id, doctor_id, medical_organisation_id, disease, CURRENT_DATE(), description); </pre>
Хранимая процедура	Возвращает идентификатор записи в таблице Doctor, который соответствует текущему пользователю-врачу	<pre> myscursor.execute("CREATE PROCEDURE medical_organisations_db.get_medical_organisation_ user_id(\ login varchar(100) \) \ SELECT medical_organisation_id \ FROM User_medical_organisation AS u \ WHERE u.login = login;") </pre>

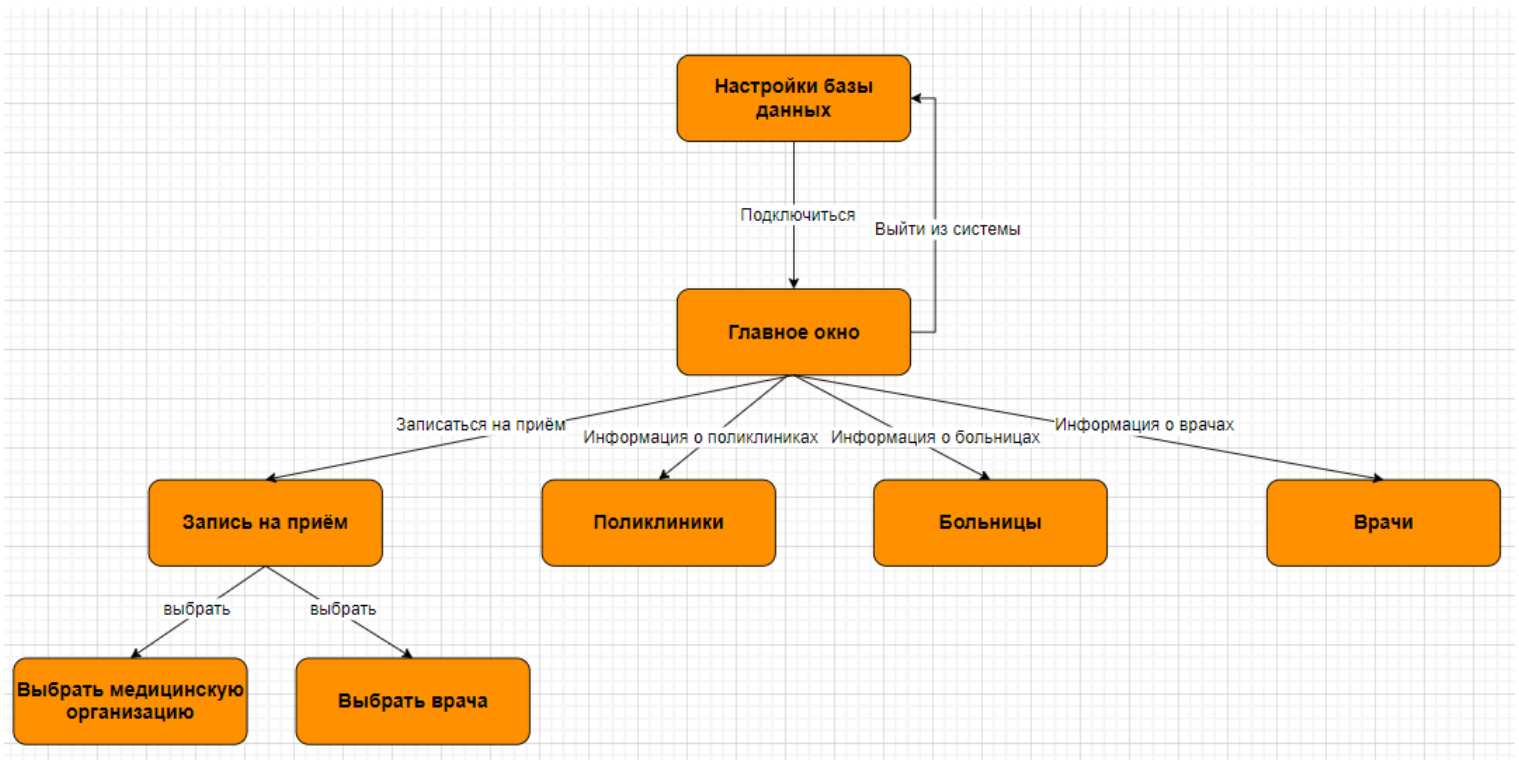
Дополнительная роль. Администратор системы

Тип	Назначение	Код
Представление	Просмотр минимально необходимого количества информации для поиска медицинской организации, которую нужно удалить	<pre> myscursor.execute("CREATE VIEW Medical_Organisation_AdminUser_View \ AS SELECT medical_organisation_id, \ name, \ address \ FROM Medical_organisation") </pre>

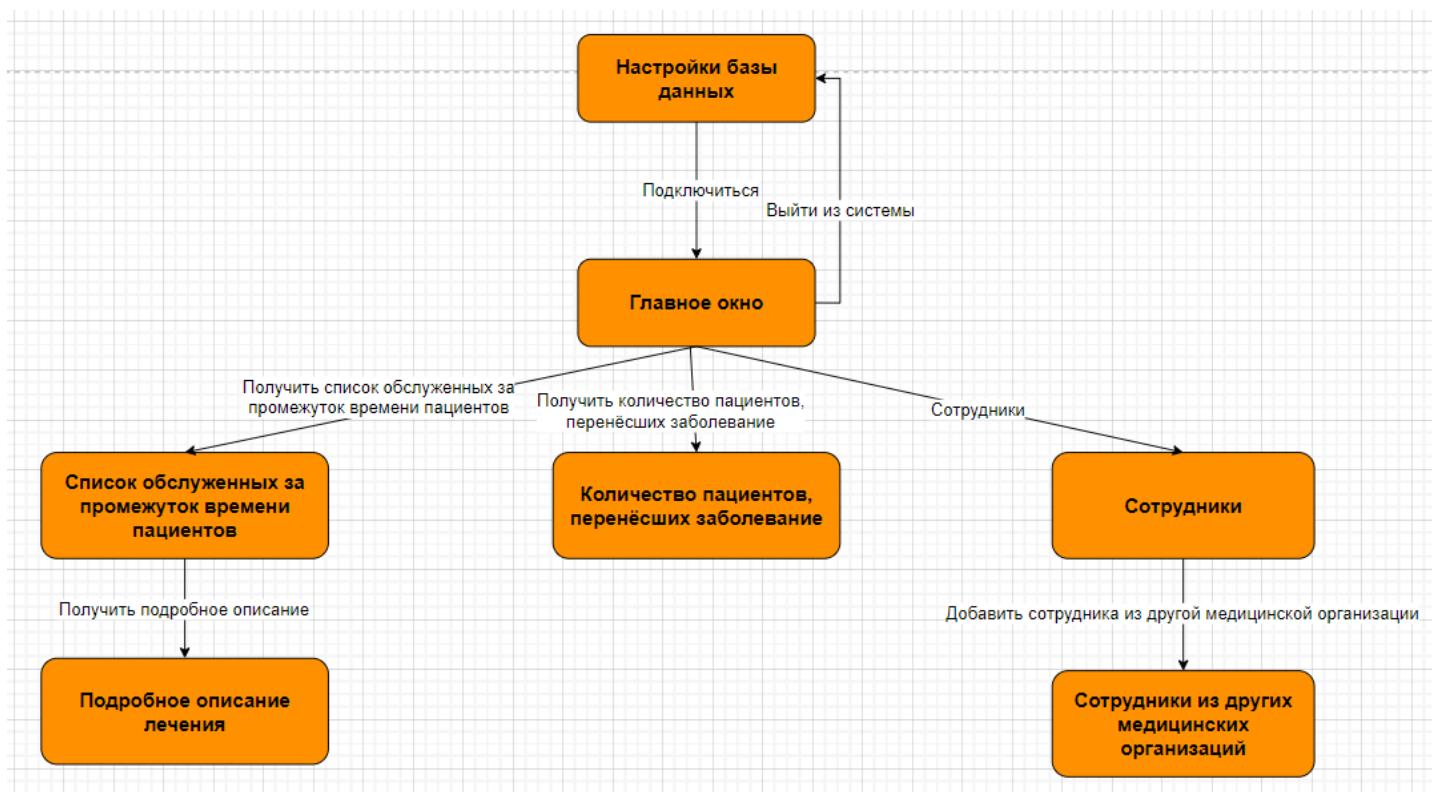
Карты интерфейсов

На данных диаграммах представлены окна, доступные каждой из ролей. На стрелках подписаны кнопки, нажатие на которые вызывает открытие окон.

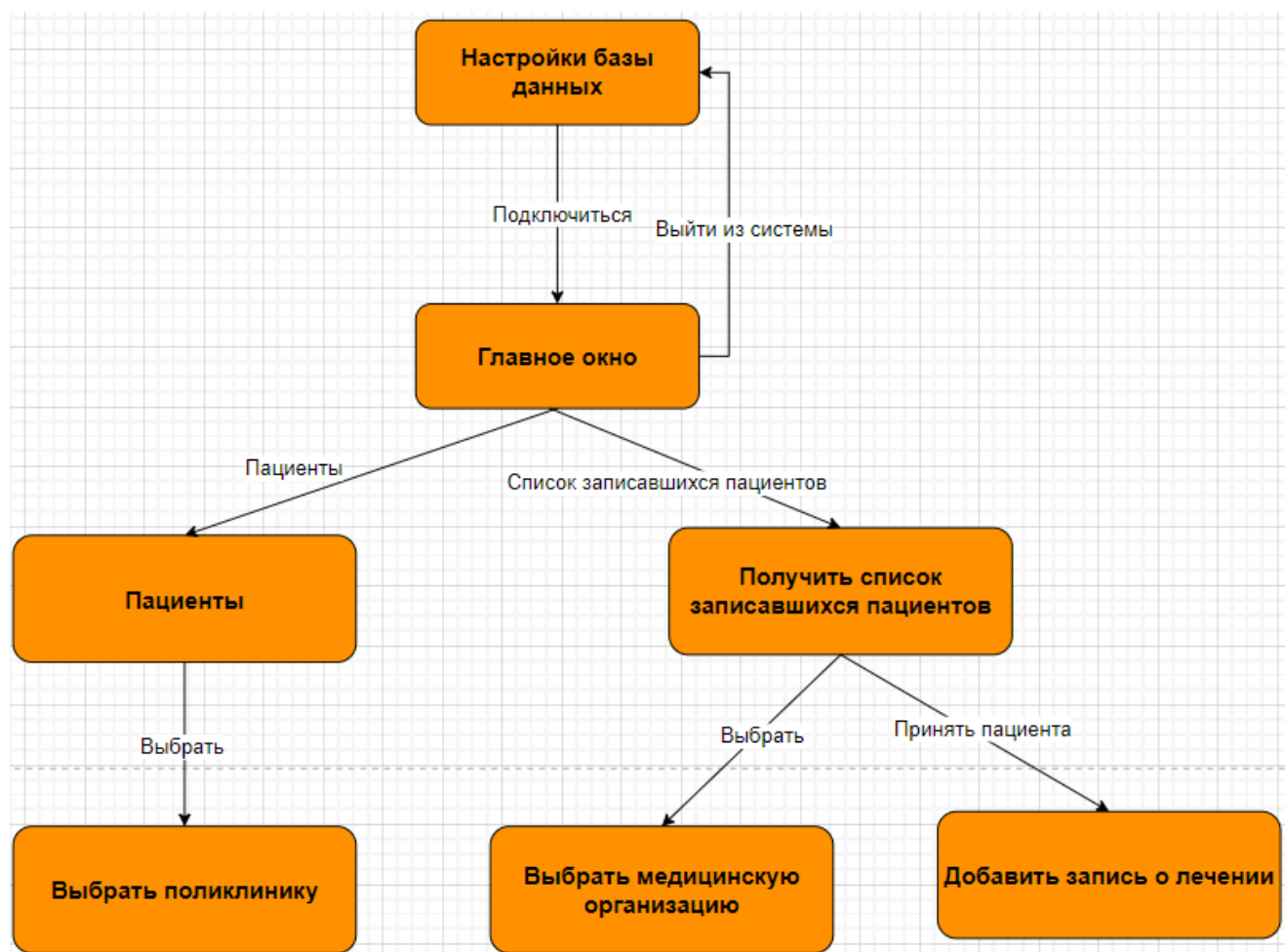
Роль 1. Пациент



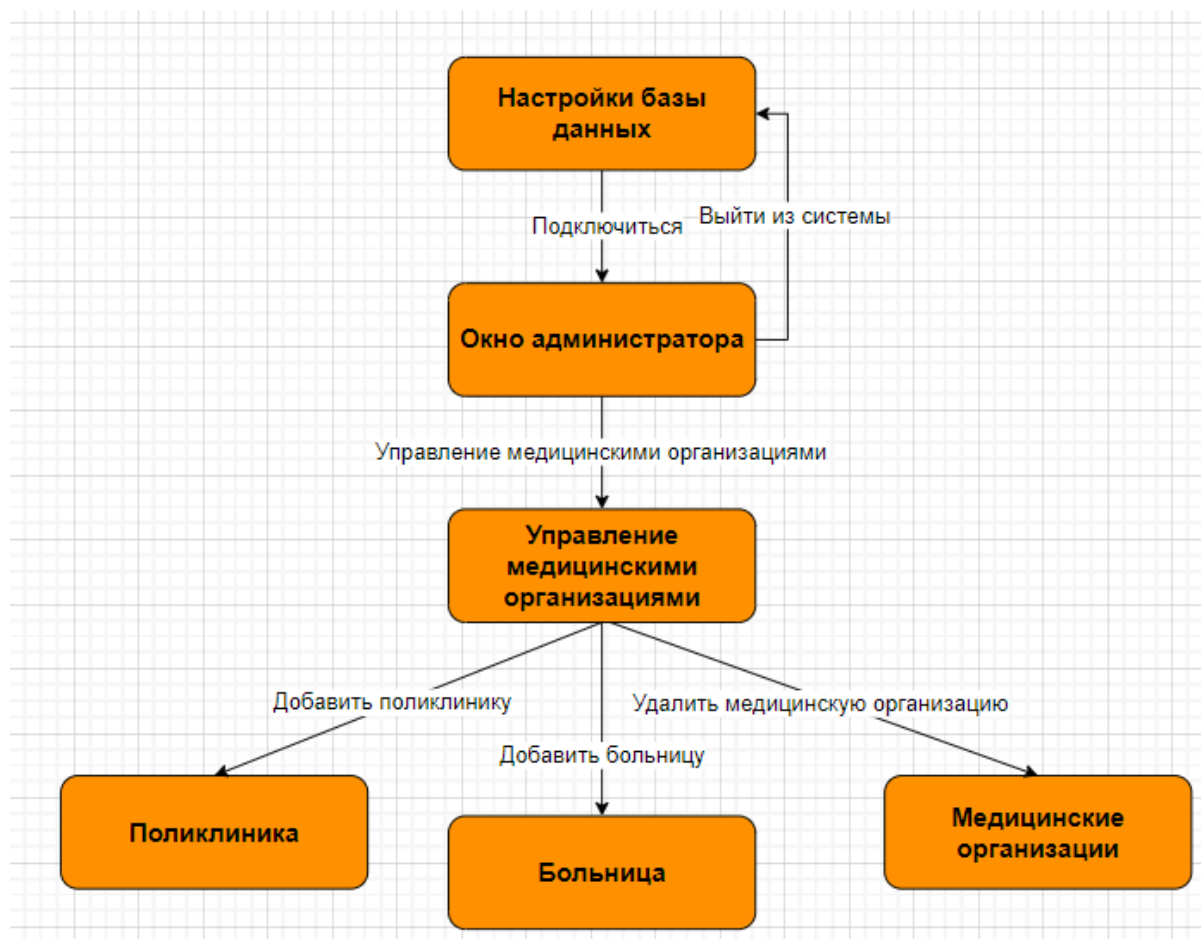
Роль 2. Руководитель медицинской организации



Роль 3. Врач



Дополнительная роль. Администратор системы



Таблицы окон

Роль 1. Пациент

Название окна	Назначение	Родители	Потомки	Модальность
Настройки базы данных	Ввод конфигурации локального сервера с базой данных: хоста, порта, пользователя, пароля	-	Главное окно, Error	нет
Главное окно	Кнопки для перехода к основным разделам системы	Настройки базы данных	Запись на прием, Поликлиники, Больницы, Врачи, Настройки базы данных	нет
Запись на приём	Просмотр текущих записей пользователя, добавление	Главное окно	Выбрать медицинскую	нет

	новых записей		организацию, Выбрать врача	
Выбрать медицинскую организацию	Выбор медицинской организации из списка путем выделения и нажатия на кнопку “Выбрать медицинскую организацию”	Запись на приём	-	да
Выбрать врача	Выбор врача из списка путем выделения и нажатия на кнопку “Выбрать врача”	Запись на приём	-	да
Поликлиники	Просмотр списка поликлиник	Главное окно	-	нет
Больницы	Просмотр списка больниц	Главное окно	-	нет
Врачи	Просмотр списка врачей	Главное окно	-	нет

Роль 2. Руководитель медицинской организации

Название окна	Назначение	Родители	Потомки	Модальность
Настройки базы данных	Ввод конфигурации локального сервера с базой данных: хоста, порта, пользователя, пароля	-	Главное окно, Error	нет
Главное окно	Кнопки для перехода к основным разделам системы	Настройки базы данных	Список обслуженных за промежуток времени пациентов, Количество пациентов, перенёсших заболевание, Сотрудники, Настройки базы данных	нет
Список обслуженных за промежуток времени пациентов	Получение списка пациентов, обслуженных текущей медицинской организацией за выбранный промежуток времени	Главное окно	Подробное описание лечения	нет
Подробное описание лечения	Просмотр описания проведенного лечения	Список обслуженных за промежуток	-	нет

		времени пациентов		
Количество пациентов, перенёсших заболевание	Получение числа пациентов текущей медицинской организацией, прошедших лечение указанного заболевания	Главное окно	-	нет
Сотрудники	Просмотр, добавление, изменение и удаление информации о сотрудниках текущей медицинской организации	Главное окно	Сотрудники из других медицинских организаций	нет
Сотрудники из других медицинских организаций	Выбор и добавление сотрудника из других медицинских организаций	Сотрудники	-	нет

Роль 3. Врач

Название окна	Назначение	Родители	Потомки	Модальность
Настройки базы данных	Ввод конфигурации локального сервера с базой данных: хоста, порта, пользователя, пароля	-	Главное окно, Error	нет
Главное окно	Кнопки для перехода к основным разделам системы	Настройки базы данных	Список обслуженных за промежуток времени пациентов, Количество пациентов, перенёсших заболевание, Сотрудники, Настройки базы данных	нет
Пациенты	Просмотр, добавление, изменение и удаление данных о пациентах	Главное окно	Выбрать поликлинику	нет
Выбрать поликлинику	Выбор по названию поликлиники, к которой прикреплён пациент	Пациенты	-	да

Получить список записавшихся пациентов	Получение списка пациентов, записанных в текущий день на прием к текущему врачу в медицинскую организацию, которую он выбрал из тех, в которых он является сотрудником	Главное окно	Выбрать медицинскую организацию, Добавить запись о лечении	нет
Выбрать медицинскую организацию	Выбор медицинской организации из тех, в которых текущий врач является сотрудником	Получить список записавшихся пациентов	-	да
Добавить запись о лечении	Добавление новой записи о проведенном лечении	Получить список записавшихся пациентов	-	да

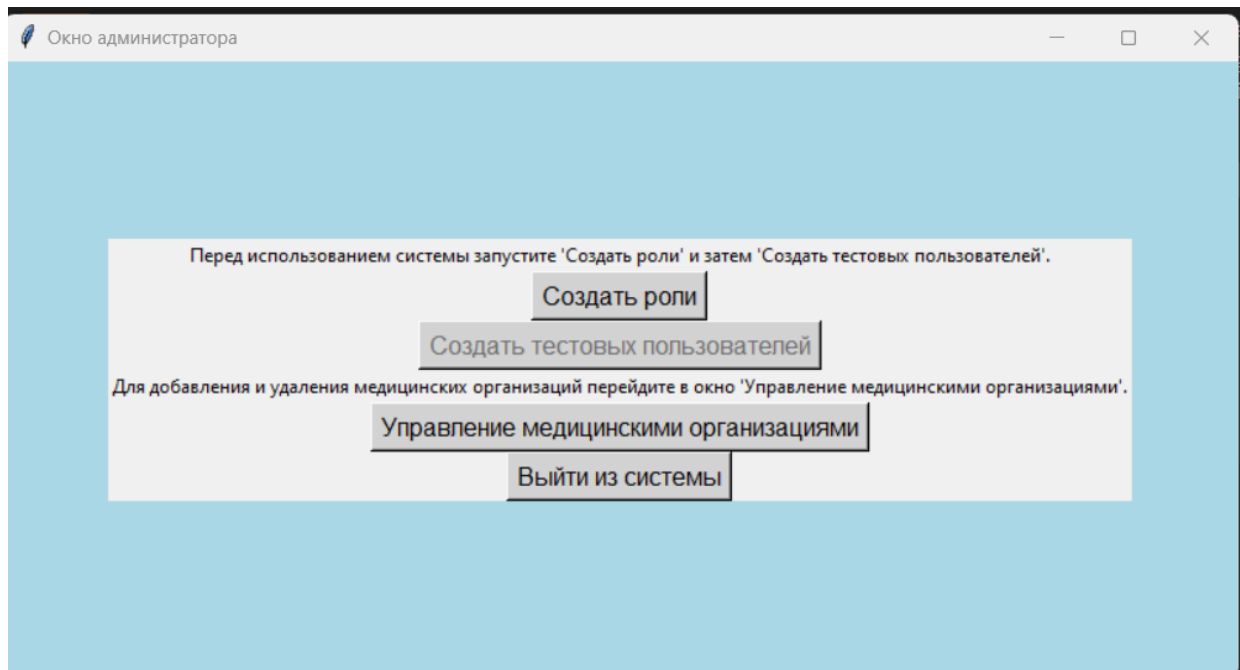
Дополнительная роль. Администратор

Название окна	Назначение	Родители	Потомки	Модалность
Настройки базы данных	Ввод конфигурации локального сервера с базой данных: хоста, порта, пользователя, пароля	-	Окно администратора, Еггог, Настройки базы данных	нет
Окно администратора	Кнопки для запуска создания ролей и создания тестовых пользователей	Настройки базы данных	Управление медицинскими организациями, Настройки базы данных	нет
Управление медицинскими организациями	Кнопки для перехода к добавлению и удалению медицинских организаций	Окно администратора	Поликлиники, Больница, Медицинские организации	нет
Поликлиника	Добавление новой поликлиники в систему	Управление медицинскими организациями		нет
Больница	Добавление новой поликлиники в систему	Управление медицинскими организациями		нет
Медицинские организации	Выбор медицинской организации для удаления из системы	Управление медицинскими организациями		нет

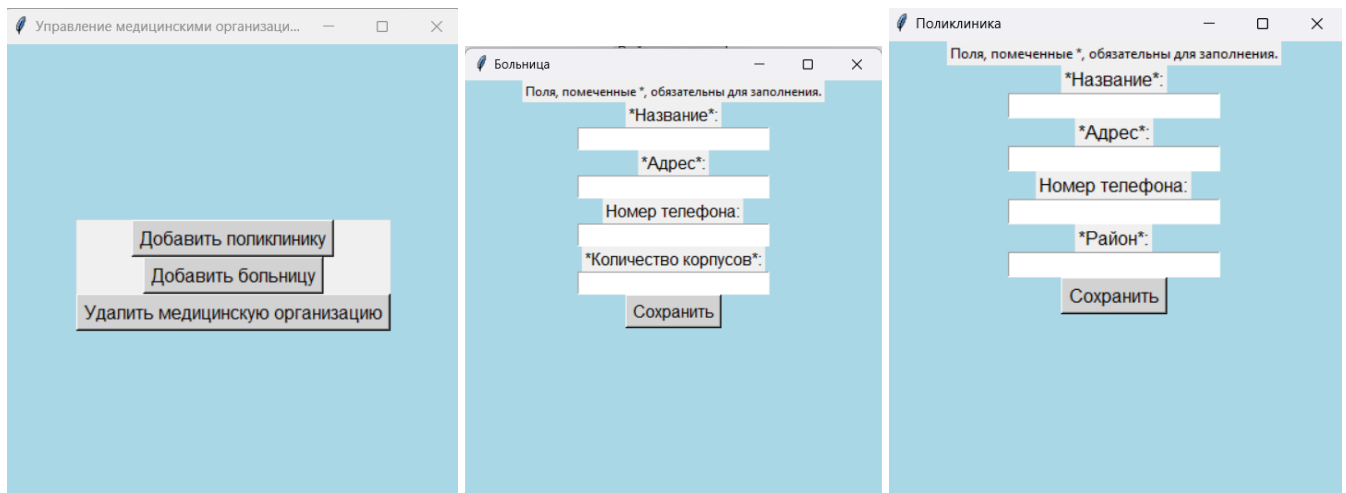
Тестирование элементов внешней схемы

Администратор системы

Окно администратора



Окна Управление медицинскими организациями, Поликлиника, Больница



Окно Медицинские организации

Медицинские организации

Удалить

	id	name	address
1		Центральная клиническая больница	Пирогова, 25/1
2		Бердская центральная городская больница	Новосибирская, 10
3		Городская клиническая больница №12	Морской проспект, 25
4		Новосибирская клиническая центральная районная больница	Магистральная, 3а
5		Городская клиническая больница №2	Ползунова, 21
6		Консультативно-диагностическая поликлиника №2	Морской проспект, 25
7		ЦНМТ	Пирогова, 25/4
8		Поликлиническое отделение №1	Шукина, 3
9		Городская клиническая поликлиника №14	Демакова, 2
10		Клиника Санитас	Николаева, 12/3

Нажмите на заголовок столбца для сортировки. Повторное нажатие сменит направление сортировки.

Роль “Пациент”

Запись на прием:

Запись на приём

Выберите медицинскую организацию, затем врача и дату для записи на прием.

Медицинская организация: не выбрано

Врач: не выбрано

Дата (YYYY-MM-DD): не выбрано

Текущие записи:

Выбрать


Выбораться

Выбрать

last_name	name	patronymic	medical_organisation_name	address	date
-----------	------	------------	---------------------------	---------	------

Нажмите на заголовок столбца для сортировки. Повторное нажатие сменит направление сортировки.

Выбрать медицинскую организацию:


Выбрать медицинскую организацию

Выбрать медицинскую организацию

medical_organisation_name	address
Консультативно-диагностическая ЦНМТ	Морской проспект, 25 Пирогова, 25/4
Поликлиническое отделение №1	Шукшина, 3
Городская клиническая поликлиника	Демакова, 2
Клиника Санитас	Николаева, 12/3
Центральная клиническая больница	Пирогова, 25/1
Бердская центральная городская больница	Новосибирская, 10
Городская клиническая больница	Морской проспект, 25
Новосибирская клиническая центральная больница	Магистральная, 3а
Городская клиническая больница	Ползунова, 21

Нажмите на заголовок столбца для сортировки. Повторное нажатие сменит направление сортировки.

Выбрать врача:

Выбрать врача

Выбрать врача

last_name	name	patronymic
Михайлов	Семен	Игоревич

Нажмите на заголовок столбца для сортировки. Повторное нажатие сменит направление сортировки.

Поликлиники:

Поликлиники					
name		address	phone	district	
Консультативно-диагностическая поликлиника №2 ЦНМТ		Морской проспект, 25	73833066657	Советский	
Поликлиническое отделение №1		Пирогова, 25/4	73833630183	Советский	
Городская клиническая поликлиника №14		Шукшина, 3	73833389747	Октябрьский	
Клиника Санитас		Демакова, 2	73833047444	Советский	
		Николаева, 12/3	73832336600	Советский	

Нажмите на заголовок столбца для сортировки. Повторное нажатие сменит направление сортировки.

Больницы:

Больницы				
name		address	phone	medical_corps_amount
Центральная клиническая больница		Пирогова, 25/1	73833304321	2
Бердская центральная городская больница		Новосибирская, 10	73834155610	4
Городская клиническая больница №12		Морской проспект, 25	88002000200	1
Новосибирская клиническая центральная районная бс		Магистральная, 3а	79061959432	1
Городская клиническая больница №2		Ползунова, 21	73833639507	6

Нажмите на заголовок столбца для сортировки. Повторное нажатие сменит направление сортировки.

Врачи:

Врачи					
	last_name	name	patronymic	medical_organisation_name	address
Иванов		Иван	Иванович	Городская клиническая поликлиника №14	Демакова, 2
Петров		Петр	Петрович	Бердская центральная городская больница	Новосибирская, 10
Сидорова		Ольга	Владимировна	Городская клиническая больница №2	Ползунова, 21
Козлов		Алексей	Сергеевич	Центральная клиническая больница	Пирогова, 25/1
Козлов		Алексей	Сергеевич	Новосибирская клиническая центральная ф	Магистральная, 3а
Михайлова		Екатерина	Андреевна	Городская клиническая больница №2	Ползунова, 21
Михайлов		Семен	Игоревич	Поликлиническое отделение №1	Шукушина, 3
Иванова		Анна	Сергеевна	Городская клиническая поликлиника №14	Демакова, 2
Сидорова		Елена	Александровна	Центральная клиническая больница	Пирогова, 25/1
Козлов		Артём	Дмитриевич	ЦНМТ	Пирогова, 25/4
Козлов		Артём	Дмитриевич	Клиника Санитас	Николаева, 12/3
Пашков		Виктор	Сергеевич	Центральная клиническая больница	Пирогова, 25/1

Нажмите на заголовок столбца для сортировки. Повторное нажатие сменит направление сортировки.

Роль “Руководитель медицинской организации”
Список обслуженных за промежуток времени пациентов:

Список обслуженных за промежуток времени пациентов

Введите начальную и конечную дату периода, за который хотите получить отчёт. После этого вы сможете выделить запись для получения подробного описания.

Начальная дата (YYYY-MM-DD):

2024-01-01

Конечная дата (YYYY-MM-DD):

2024-06-01

Найти

Получить подробное описание

patient	doctor	disease	date
Николаева Мария Владимировна	Петров Петр Петрович	Грипп	2024-01-08
Павлов Артём Сергеевич	Михайлова Екатерина Андреевна	ОРВИ	2024-03-10
Смирнова Ольга Александровна	Михайлова Екатерина Андреевна	ОРВИ	2024-03-07
Волков Илья Иванович	Пашков Виктор Сергеевич	Грипп	2024-05-20

Нажмите на заголовок столбца для сортировки. Повторное нажатие сменит направление сортировки.

Количество пациентов, перенесших заболевание:

Количество пациентов, перенесших заболевание

Введите название заболевание, для которого хотите получить статистику.

Название заболевания:

ОРВИ

Найти

Количество пациентов

2

Сотрудники:

Сотрудники

Для добавления нового сотрудника введите его данные и нажмите 'Сохранить'. Поля Фамилия, Имя обязательны для заполнения.

Для обновления данных сотрудника выделите запись для изменения, заполните необходимые поля и нажмите 'Обновить'.

Для удаления данных сотрудника выделите запись и нажмите 'Удалить'.

Для добавления сотрудника, который уже зарегистрирован в другой медицинской организации, перейдите в окно 'Добавить сотрудника из другой медицинской организации'.

Добавить сотрудника из другой медицинской организации

Фамилия:

Имя:

Отчество:

Номер телефона:

Стаж работы:

Сохранить

Обновить

Удалить

doctor_id	last_name	name	patronymic	phone	work_experience
4	Козлов	Алексей	Сергеевич	79105554433	3900
9	Сидорова	Елена	Александровна	79345678901	410
12	Пашков	Виктор	Сергеевич		

Нажмите на заголовок столбца для сортировки. Повторное нажатие сменит направление сортировки.

Роль “Врач”

Пациенты:

Пациенты

Для добавления нового пациента введите его данные и нажмите 'Сохранить'. Поля Фамилия, Имя, Дата рождения, Адрес, Поликлиника обязательны для заполнения.

Для обновления данных пациента выделите запись для изменения, заполните необходимые поля и нажмите 'Обновить'.

Для удаления данных пациента выделите запись и нажмите 'Удалить'.

Фамилия:

Имя:

Отчество:

Дата рождения (YYYY-MM-DD):

Номер телефона:

Адрес:

Номер медицинского полиса:

Поликлиника:

Сохранить

Обновить

Удалить

Выбрать

patient_id	name	birthday	phone	address	policy_number	polyclinic_id
1	Смирнова Ольга Александровна	1987-11-25	79998887766	город Новосибирск, ул. Кирова 20	1234567891234567	ЦНМТ
3	Николаева Мария Владимировна	1980-09-15	79112223344	город Новосибирск, ул. Пушкина 1	5432167895432167	Консультативно-диагностическая г
4	Павлов Артем Сергеевич	1983-06-30	79255554433	город Новосибирск, ул. Гагарина 1	2468135792468135	Поликлиническое отделение №1
5	Лебедев Иван Дмитриевич	1979-02-20	79167778899	город Новосибирск, пр. Ленина 30	1357924681357924	Клиника Санитас
6	Волков Илья Иванович	2000-01-01		ленина 8		Городская клиническая поликлини

Нажмите на заголовок столбца для сортировки. Повторное нажатие сменит направление сортировки.

Выбрать поликлинику:

Выбрать поликлинику

Выбрать поликлинику

polyclinic_name	address
Консультативно-диагностическая г	Морской проспект, 25
ЦНМТ	Пирогова, 25/4
Поликлиническое отделение №1	Шукшина, 3
Городская клиническая поликлини	Демакова, 2
Клиника Санитас	Николаева, 12/3

Нажмите на заголовок столбца для сортировки. Повторное нажатие сменит направление сортировки.

Получить список записавшихся пациентов:

Получить список записавшихся пациентов

Выберите медицинскую организацию, для которой хотите получить список пациентов.

Медицинская организация:

Выбрать

1

Выполнить

Принять пациента

Идентификатор	Фамилия	Имя	Отчество	
6	Волков	Илья	Иванович	

Добавить запись о лечении:

Добавить запись о лечении

Поля, помеченные *, обязательны для заполнения.

Заболевание:

Грипп

Описание:

Красное горло
Температура 38

Сохранить запись

Система контроля доступа

Система ролей в приложении построена следующим образом:

- Системный администратор первым входит в систему под логином и паролем пользователя root и исполняет скрипт для создания и разворачивания базы данных, всех ее таблиц и ограничений целостности, триггеров, хранимых процедур и функций, а также создания роли администратора и пользователя-администратора.
- Администратор следующим входит в систему под своим логином и паролем (admin, admin), сначала запускает скрипт для создания пользовательских ролей, а затем запускает скрипт для создания тестовых пользователей. После администратор системы может добавлять руководителей медицинских организаций, а также удалять их.
- Руководитель медицинской организации входит в систему под своим логином и паролем (выданными администратором) и в окне Сотрудники может: 1) добавлять новых сотрудников, что автоматически создаст нового пользователя с ролью Врач и вернет его учетные данные (эти данные руководитель должен сообщить сотруднику) или 2) удалить сотрудника, что автоматически удалит соответствующего ему пользователя с ролью Врач.
- Врач входит в систему под своим логином и паролем (выданными его руководителем) и в окне Пациенты аналогично может: 1) добавить нового пациента, что автоматически создаст для него нового пользователя с ролью Пациент и вернет его учетные данные (врач должен сообщить их пациенту) или 2) удалить пациента, что автоматически удалит соответствующего ему пользователя.
- Обладание одним пользователем несколькими ролями не предусмотрено. Для этого можно создать несколько учетных записей для разных ролей.
- В систему добавлены таблицы User_Medical_organisation, User_Doctor, User_Patient для хранения соответствия между логинами пользователей и идентификаторами в таблицах Medical_organisation, Doctor, Patient. При входе в систему сначала определяется роль пользователя (SELECT CURRENT_ROLE()), а затем по этим таблицам определяется соответствующих пользователю идентификатор записи в таблице.

Наборы привилегий, необходимые ролям пользователей

1. Руководитель медицинской организации:
 - право на создание пользователей и назначение им прав (создание пользователей-врачей)
 - право на чтение и запись в таблицу User_Doctor (сохранение учетных данных пользователей-врачей)
 - право на чтение для представления Doctor_MedicalOrganisationUser_View (просмотр данных о сотрудниках)
 - право на вызов процедуры get_treatment_statistics() и get_treatment_description() (получение списка пациентов, обслуженных сотрудниками подчиненной организации за выбранный период)
 - право на вызов процедуры add_existed_doctor() (добавление сотрудника из другой медицинской организации)
 - право на вызов процедуры update_doctor_in_medical_organisation() (обновление данных о сотрудниках)

- право на вызов процедуры `delete_doctor_from_medical_organisation()` (удаление данных о сотрудниках)
- право на вызов процедуры `insert_doctor_to_medical_organisation()` (добавление данных о сотрудниках)
- право на вызов процедуры `get_medical_organisation_user_id()` (получение собственного идентификатора в таблице `Medical_organisation`)
- право на вызов функции `get_patients_count_by_disease()` (получение статистики заболеваемости)

2. Врач

- право на чтение для представления `Patient_DoctorUser_View` (просмотр данных о пациентах)
- право на чтение для представления `Polyclinic_DoctorUser_View` (выбор поликлиники для прикрепления пациента)
- право на вызов процедуры `get_medical_organisations_by_doctor()` (получения списка мед. организаций, в которых работает сотрудник)
- право на вызов процедуры `get_registered_patients_list()` (получения списка записавшихся на прием пациентов)
- право на вызов процедуры `add_patient_to_polyclinic()` (добавление данных о пациенте и прикрепление его к поликлинике)
- право на вызов процедуры `update_patient_in_polyclinic()` (обновления данных о пациенте)
- право на вызов процедуры `delete_patient()` (удаление данных о пациенте)
- право на вызов процедуры `add_treatment()` (добавление сведений о лечении пациента)
- право на вызов процедуры `get_doctor_user_id()` (получение собственного идентификатора в таблице `Doctor`)

3. Пациент

- право на чтение для представления `Doctor_PatientUser_View` (просмотр данных о врачах)
- право на чтение для представления `Polyclinic_PatientUser_View` (просмотр данных о поликлиниках)
- право на чтение для представления `Hospital_PatientUser_View` (просмотр данных о больницах)
- права на запись и чтение для представления `Treatment_registration_PatientUser_View` (запись на прием)
- право на вызов процедуры `get_patients_registrations()` (и получение списка текущих записей)
- право на вызов процедуры `get_doctors_by_medical_organisation()` (выбор врача из определенной мед. организации)
- право на вызов процедуры `get_patient_user_id()` (получение собственного идентификатора в таблице `Patient`)

Кроме того, предусмотрены 2 дополнительные роли для администрирования системы:

1) Системный администратор

- право на удаление и создание базы данных
- право на создание таблиц
- право на создание внешних ключей
- право на создание ролей
- право на создание пользователя-администратора

2) Администратор системы

- право на создание пользовательских ролей
- право на создание пользователей

Скрипты для создания ролей и привилегий и назначение их пользователям

1. Администратор

```
mycursor.execute("DROP USER IF EXISTS 'admin'@'localhost';")
mycursor.execute("DROP ROLE IF EXISTS 'admin_role';")
mycursor.execute("CREATE ROLE 'admin_role';")
mycursor.execute("GRANT CREATE USER ON *.* TO 'admin_role' WITH
GRANT OPTION;")
mycursor.execute("GRANT ALL PRIVILEGES ON
medical_organisations_db.* TO 'admin_role' WITH GRANT OPTION;")
mycursor.execute("GRANT SUPER ON *.* TO 'admin_role' WITH GRANT
OPTION;")

mycursor.execute("CREATE USER 'admin'@'localhost' IDENTIFIED BY
'admin';")
mycursor.execute("GRANT 'admin_role' TO 'admin'@'localhost';")
mycursor.execute("SET DEFAULT ROLE ALL TO 'admin'@'localhost';")
```

2. Руководитель медицинской организации

```
mycursor.execute("GRANT CREATE USER ON *.* TO 'medical_organisation_role';")
mycursor.execute("GRANT SUPER ON *.* TO
'medical_organisation_role';")
mycursor.execute("GRANT SELECT, INSERT, DELETE ON
medical_organisations_db.User_Doctor TO 'medical_organisation_role';")
mycursor.execute("GRANT SELECT ON
medical_organisations_db.Doctor_MedicalOrganisationUser_View TO
'medical_organisation_role';")

mycursor.execute("GRANT EXECUTE ON PROCEDURE
medical_organisations_db.get_treatment_statistics TO 'medical_organisation_role';")
mycursor.execute("GRANT EXECUTE ON PROCEDURE
medical_organisations_db.get_treatment_description TO 'medical_organisation_role';")
mycursor.execute("GRANT EXECUTE ON PROCEDURE
medical_organisations_db.add_existed_doctor TO 'medical_organisation_role';")
mycursor.execute("GRANT EXECUTE ON PROCEDURE
medical_organisations_db.update_doctor_in_medical_organisation TO
'medical_organisation_role';")
mycursor.execute("GRANT EXECUTE ON PROCEDURE
```

```

medical_organisations_db.delete_doctor_from_medical_organisation TO
'medical_organisation_role;")
        mycursor.execute("GRANT EXECUTE ON PROCEDURE
medical_organisations_db.insert_doctor_to_medical_organisation TO
'medical_organisation_role;")
        mycursor.execute("GRANT EXECUTE ON PROCEDURE
medical_organisations_db.get_medical_organisation_user_id TO
'medical_organisation_role;")

        mycursor.execute("GRANT EXECUTE ON FUNCTION
medical_organisations_db.get_patients_count_by_disease TO
'medical_organisation_role;")

```

3. Врач

```

mycursor.execute("GRANT SELECT ON medical_organisations_db.Patient_DoctorUser_View TO
'doctor_role;")
        mycursor.execute("GRANT SELECT ON
medical_organisations_db.Polyclinic_DoctorUser_View TO 'doctor_role;")

        mycursor.execute("GRANT EXECUTE ON PROCEDURE
medical_organisations_db.get_medical_organisations_by_doctor TO 'doctor_role;")
        mycursor.execute("GRANT EXECUTE ON PROCEDURE
medical_organisations_db.get_registered_patients_list TO 'doctor_role;")
        mycursor.execute("GRANT EXECUTE ON PROCEDURE
medical_organisations_db.add_patient_to_polyclinic TO 'doctor_role;")
        mycursor.execute("GRANT EXECUTE ON PROCEDURE
medical_organisations_db.update_patient_in_polyclinic TO 'doctor_role;")
        mycursor.execute("GRANT EXECUTE ON PROCEDURE
medical_organisations_db.delete_patient TO 'doctor_role;")
        mycursor.execute("GRANT EXECUTE ON PROCEDURE
medical_organisations_db.add_treatment TO 'doctor_role;")
        mycursor.execute("GRANT EXECUTE ON PROCEDURE
medical_organisations_db.get_doctor_user_id TO 'doctor_role;")

```

4. Пациент

```

mycursor.execute("GRANT SELECT ON medical_organisations_db.Doctor_PatientUser_View TO
'patient_role;")
        mycursor.execute("GRANT SELECT ON
medical_organisations_db.Polyclinic_PatientUser_View TO 'patient_role;")
        mycursor.execute("GRANT SELECT ON
medical_organisations_db.Hospital_PatientUser_View TO 'patient_role;")
        mycursor.execute("GRANT SELECT, INSERT, UPDATE, DELETE ON
medical_organisations_db.Treatment_registration_PatientUser_View TO 'patient_role;")

        mycursor.execute("GRANT EXECUTE ON PROCEDURE
medical_organisations_db.get_patients_registrations TO 'patient_role;")
        mycursor.execute("GRANT EXECUTE ON PROCEDURE

```

```
medical_organisations_db.get_doctors_by_medical_organisation TO 'patient_role;")
mycursor.execute("GRANT EXECUTE ON PROCEDURE
medical_organisations_db.get_patient_user_id TO 'patient_role;")
```

Приложение было протестировано на корректность выдачи прав и привилегий (с помощью проверки данных в таблице mysql.user и из графического интерфейса), а также на корректность авторизации пользователя для возможности дальнейшего использования всех функциональных возможностей системы.