**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 5](#_Toc12520)

[1 ОБЩАЯ ЧАСТЬ 6](#_Toc17100)

[1.1 Общие сведения о предприятии 6](#_Toc4647)

[1.2 Описание организационной структуры 6](#_Toc20213)

[1.3 Описание существующей информационной системы 7](#_Toc10781)

[2 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА 8](#_Toc27996)

[2.1 Диаграммы UML и IDEF 8](#_Toc18736)

[2.2 Диаграмма потоков данных 11](#_Toc6642)

[2.3 Диаграмма классов 12](#_Toc15391)

[3 ВЫБОР СРЕДСТВ РАЗРАБОТКИ 14](#_Toc5291)

[4 УСТАНОВКА, НАСТОЙКА СЕРВЕРА БАЗЫ ДАННЫХ 16](#_Toc6270)

[5 РАЗРАБОТКА БАЗЫ ДАННЫХ, ПРИЛОЖЕНИЯ 21](#_Toc17552)

[5.1 Таблицы базы данных, структура. Нормализация базы данных 21](#_Toc14828)

[5.2 Схема базы данных 22](#_Toc19604)

[5.3 Интерфейс приложения 23](#_Toc24086)

[5.4 Реализация поиска в таблице базы данных 29](#_Toc10256)

[5.5 Реализация фильтрации данных 29](#_Toc32205)

[5.6 Реализация запросов к таблице базы данных 30](#_Toc5130)

[5.7 Хранимые процедуры 32](#_Toc13228)

[5.8 Импорт,экспорт информации базы данных 34](#_Toc29611)

[5.9 Резервное копирование, восстановление информации базы данных 36](#_Toc17707)

[5.10 Формирование отчётов для печати 38](#_Toc4513)

[6 ТЕСТИРОВАНИЕ, СОПРОВОЖДЕНИЕ И ОБСЛУЖИВАНИЕ ПРОГРАММНОГО ПРОДУКТА 39](#_Toc18707)

[6.1 Установка серверного программного обеспечения, программного обеспечения рабочих станций, системные требования 39](#_Toc27954)

[6.2 Настройка антивирусного программного обеспечения, Файрвола 41](#_Toc32420)

[6.3 Результаты тестирования 42](#_Toc31055)

[7 АДМИНИСТРИРОВАНИЕ, ЗАЩИТА БАЗЫ ДАННЫХ 45](#_Toc17143)

[7.1 Учетные записи сервера, базы данных 45](#_Toc9458)

[7.2 Правила, роли, ограничения 45](#_Toc15966)

[ЗАКЛЮЧЕНИЕ 47](#_Toc32628)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 48](#_Toc29134)

[ГЛОССАРИЙ 49](#_Toc8410)

[СПИСОК АББРЕВИАТУР 50](#_Toc19611)

[ПРИЛОЖЕНИЯ 51](#_Toc32025)

[Приложение А. Структуры БД 51](#_Toc29964)

[Приложение Б. Программный код 52](#_Toc17120)

[Приложение В. Руководство оператора 55](#_Toc654)

[Приложение Г. Анализ существующих аналогичных ПП 57](#_Toc10613)

**ВВЕДЕНИЕ**

Автоматизация учёта анализов пациента больницы является актуальной задачей, поскольку она способствует повышению качества медицинской помощи и сокращению времени нахождения пациентов в амбулаторных учреждениях. Для этого необходимо разработать специальное приложение, которое будет связывать пациентов с медицинскими специалистами и обеспечивать надёжность и конфиденциальность данных.

Разработка такого приложения позволит улучшить обслуживание пациентов, уменьшить очереди и повысить удобство предоставления медицинской помощи, а также обеспечить эффективность и качество работы медицинского персонала.

Объектом исследования: является ГБУЗ РХ РКБ имени Г.Я. Ремишевской.

Предметом исследования: является приложение по учету анализов пациентов ГБУЗ РХ РКБ имени Г.Я. Ремишевской.

Целью дипломной работы является создание приложения по учету анализов пациентов ГБУЗ РХ РКБ имени Г.Я. Ремишевской.

Для выполнения цели необходимо выполнить следующие задачи:

* Настройка и проектирование систем управления баз данных;
* Реализация клиентского приложения;
* Использование и реализация объектов баз данных;
* Освоение администрирования базы данных.

В ходе дипломной работы будут рассмотрены различные аспекты разработки и администрирования базы данных организации, включая проектирование, настройку, реализацию и администрирование системы управления базами данных. Кроме того, будут рассмотрены вопросы разработки и реализации клиентского приложения, а также использования и реализации объектов базы данных.

1. **ОБЩАЯ ЧАСТЬ**
   1. **Общие сведения о предприятии**

Республиканская клиническая больница имени Г. Я. Ремишевской - многопрофильное лечебно-профилактическое учреждение, оказывающее специализированную медицинскую помощь населению Республики Хакасия, Тывы и Юга Красноярского края[9].

Предметом деятельности Учреждения является оказание первичной медико-санитарной помощи, специализированной, в том числе высокотехнологичной, медицинской помощи, скорой, в том числе скорой специализированной, медицинской помощи вне медицинской организации.

Целями деятельности Учреждения являются: обеспечение населения квалифицированной диагностической, профилактической, лечебной медицинской помощью; практическая подготовка врачей-ординаторов и среднего медицинского персонала; обеспечение условий для проведения научно-исследовательской деятельности.

Реквизиты ГБУЗ РХ «РКБ имени Г.Я. Ремишевской» :

655012, Республика Хакасии, г. Абакан, проспект Ленина, дом 23.

тел./факс: (3902) 248-263, 248-262.

E-mail: gbuzrkb@yandex.ru

* 1. **Описание организационной структуры**

В структуре учреждения функционируют: диагностическая служба, республиканская консультативная поликлиника, стационар с хирургической, терапевтической и педиатрической службами, а также центры республиканского уровня: диабетологический центр, центр амбулаторного диализа, антирабический центр, клинико-диагностический центр для беременных, сосудистый центр[10].

Также организация имеет несколько подразделений, каждое из которых обладает определенной функциональностью и обслуживает различные потребности пациентов:

Организация имеет четыре подразделения:

1. Круглосуточный и дневной стационар:

* Предоставляет круглосуточную и плановую медицинскую помощь в различных медицинских направлениях, включая кардиологию, терапию, неврологию, хирургию, гинекологию и реанимацию.

1. Поликлиническое отделение:

* Оказывает консультативно-диагностическую и лечебную помощь амбулаторным пациентам, включая регистрацию, прием, направление к специалистам и оказание медицинских услуг.

1. Администрация:

* Управляет всей организацией, включая стратегическое планирование, финансовое управление и контроль качества медицинских услуг.

1. Центр медицинской реабилитации и спортивной медицины:

* Предоставляет комплексную медицинскую и спортивную реабилитацию, включая услуги по восстановлению после травм и заболеваний, а также спортивную медицину.
  1. **Описание существующей информационной системы**

Существующая информационная система в больнице включает в себя аппаратно-программное обеспечение, предназначенное для автоматизации различных аспектов работы лечебно-профилактического учреждения. Основные требования к этой системе включают организацию данных, защиту от несанкционированного доступа, администрирование базы данных, работоспособность программного обеспечения и наличие графического пользовательского интерфейса.

Приоритетные направления автоматизации управления медицинским учреждением включают учёт расходов и управление ресурсами, ввод системы электронного документооборота, ведение медицинской статистики, формирование медицинских регистров населения и баз данных пациентов.

Однако в существующей информационной системе отсутствует специализированный модуль для учета проводимых анализов. В настоящее время учет анализов осуществляется вручную с использованием традиционных средств, что может приводить к неэффективности, ошибкам и задержкам в процессе.

1. **ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА**
   1. **Диаграммы UML и IDEF0**

UML-диаграмма – это графическое представление системы, которое используется для моделирования, проектирования и документирования программного обеспечения и других сложных систем[2].

UML-диаграммы используются для:

* Визуализации системы. Предоставление наглядного представления структуры, поведения и взаимосвязей системы;
* Моделирования системы. Создание абстрактных моделей системы, которые можно использовать для анализа и проектирования;
* Документирования системы. Запись требований, дизайна и реализации системы в понятном и структурированном формате;
* Общения с заинтересованными сторонами. Обеспечение общего языка для обсуждения и понимания системы между разработчиками, аналитиками и другими заинтересованными сторонами;
* Генерации кода. UML-диаграммы могут использоваться для автоматической генерации кода, что экономит время и снижает вероятность ошибок.

В UML-диаграмме используются две сущности: пациент и медработник. Пациент может записаться на сдачу анализов, а также получить результаты сданных анализов. Медработник может записать пациента на анализы, составить медицинскую карту пациента для сохранения данных о пациенте в реестре больницы. Также медработник работает с данными пациентов, может добавлять, удалять или редактировать их. Для удобства работы с данными может воспользоваться фильтрацией или поиском.

На основе описанной предметной области и требованиям к программному приложению составлена UML-диаграмма, представленная на рисунке 2.1.1.

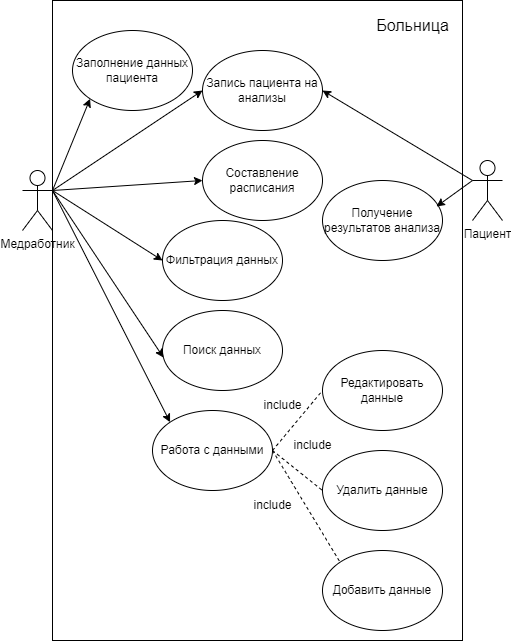


Рисунок 2.1.1 – UML диаграмма

IDEF0 — методология функционального моделирования (англ. function modeling) и графическая нотация, предназначенная для формализации и описания бизнес-процессов[14]. Отличительной особенностью IDEF0 является ее акцент на соподчиненность объектов. В IDEF0 рассматриваются логические отношения между работами, а не их временная последовательность .

Преимущества IDEF0 диаграмм включают в себя[5]:

* Дисциплина и отсутствие ошибок. Стандарт IDEF0 предполагает строгие рамки и правила. Такой подход дисциплинирует, а привычка действовать в рамках стандарта помогает избежать ошибок по невнимательности. Любые нарушения стандарта становятся сразу заметны;
* Наглядность. Вы сами начинаете понимать, как работает та или иная система, и можете также наглядно пояснить, где в этой системе «тонкие места» и как ваши решения помогут избавиться от них;
* Взаимопонимание и отсутствие разночтений. При обсуждении работы компании с использованием функциональной модели у вас имеются наглядные и понятные интуитивно блоки задач с управляющими элементами. Кроме того, функциональное моделирование предполагает создание в случае необходимости глоссария, в котором раскрываются условные обозначения и термины. В результате вы с клиентом, руководителем, другими сотрудниками при обсуждении проблемы говорите на одном языке.

В соответствии с задачей по учету анализов пациентов, согласно ФЗ РФ, уставу организации и данных с медицинских карт медицинский работник на своей рабочей станции обрабатывает данные о пациенте для последующей записи пациента на сдачу анализов и внесения изменений в медицинскую карту пациента.

На рисунке 2.1.2 представлена IDEF0.

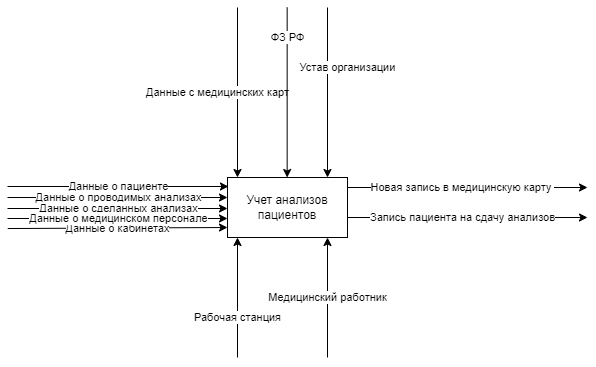


Рисунок 2.1.2 – IDEF0

В соответствии с задачей по учету анализов пациентов, согласно ФЗ РФ, уставу организации и данных с медицинских карт медицинский работник на своей рабочей станции обрабатывает данные о пациенте для последующей записи пациента на сдачу анализов и внесения изменений в медицинскую карту пациента. Пациент в свою очередь сдаёт анализы в назначенное ему время или день, после проведения исследований и получения результата, он может посмотреть их в справке или же в своей медицинской карточке.

На рисунке 2.1.3 представлена декомпозиция IDEF0.

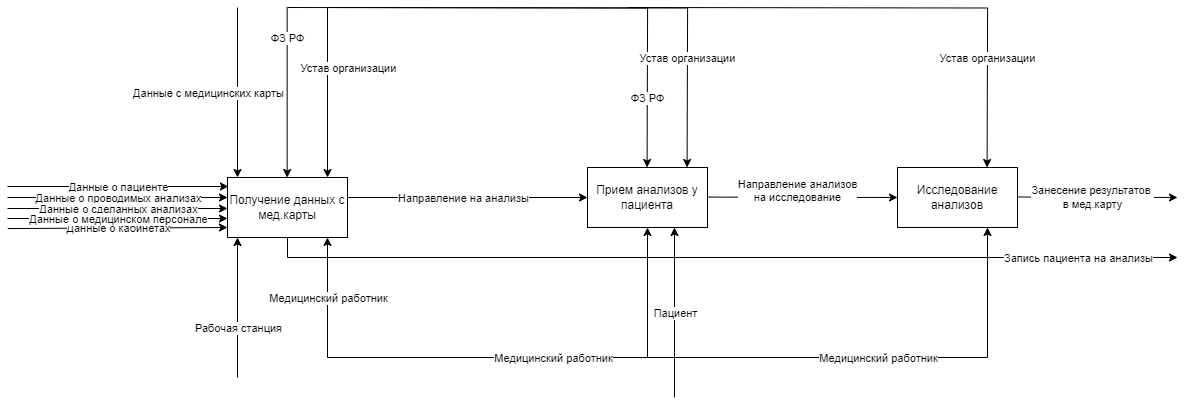


Рисунок 2.1.3 – Декомпозиция IDEF0

* 1. **Диаграмма потоков данных**

Диаграмма потоков данных (DFD) – это графическое представление потоков данных в информационной системе. Она используется для иллюстрации того, как данные передаются через систему, обрабатываются и хранятся.

Основная цель диаграммы потоков данных –понять, как информация перемещается в системе, какие процессы её обрабатывают и какие данные влияют на результаты этих процессов. Она помогает в моделировании бизнес-процессов, идентификации потенциальных проблем и оптимизации работы системы.

Преимущества диаграмм потоков данных:

* Простота восприятия: DFD предоставляют простой и понятный способ визуализации потоков данных и процессов в системе, что облегчает понимание её работы;
* Анализ безопасности и контроля доступа: Они помогают идентифицировать уязвимости в системе и разрабатывать стратегии для обеспечения безопасности данных и контроля доступа.

Диаграммы потоков данных являются важным инструментом для анализа, проектирования и управления информационными системами, помогая организациям эффективно использовать свои ресурсы и оптимизировать свои бизнес-процессы.

Медицинский персонал работает с данными пациентов в приложение, которое связано с базой данных больницы,может добавлять,удалять или редактировать их. Для удобства работы с данными может воспользоваться фильтрацией или поиском.

В ходе проектирования и анализа требований к разрабатываемому программному обеспечению была составлена DFD диаграмма, представленная на рисунке 2.2.1.



Рисунок 2.2.1 – DFD диаграмма

* 1. **Диаграмма классов**

Диаграмма классов – это статическая структуральная диаграмма в языке унифицированного моделирования (UML), которая отображает классы системы, их атрибуты, операции и отношения между ними.

Диаграммы классов используются для:

* Представления статической структуры системы, включая классы, атрибуты и операции;
* Моделирования отношений между классами, таких как наследование, агрегация и композиция;
* Документирования структуры и поведения системы.

Преимущества диаграмм классов

* Улучшенное понимание системы: Диаграммы классов обеспечивают визуальное представление статической структуры системы, упрощая ее понимание;
* Модульность и повторное использование: способствуют модульному проектированию, позволяя повторно использовать классы и отношения в различных частях системы.

Для учета анализов пациента была разработана диаграмма классов, которая отображает классы системы и их атрибуты, а также отношения между ними. Все это позволяет понять структуру реализуемой базы данных больницы по учету анализов пациентов.

В ходе проведения проектирования, была составлена диаграмма классов, которая представлена на рисунке 2.3.1.

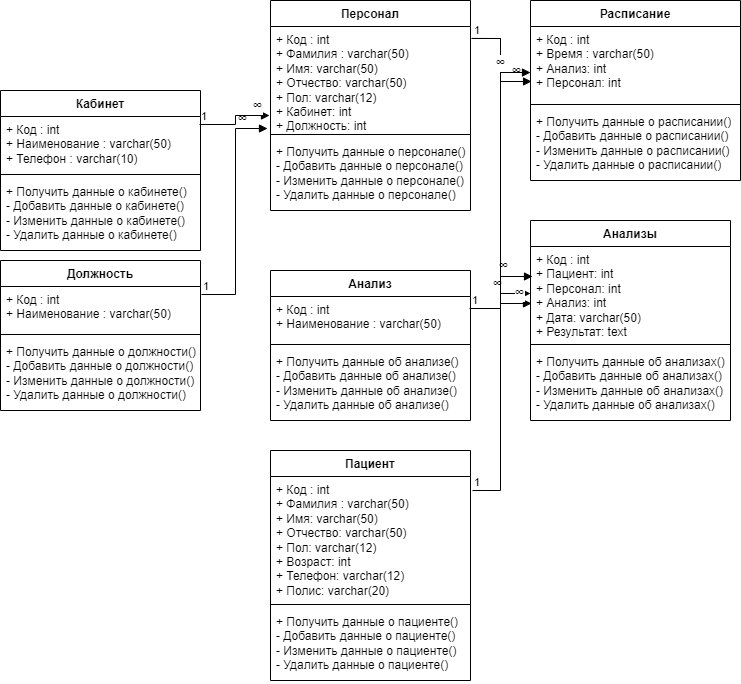


Рисунок 2.3.1 – Диаграмма классов

1. **ВЫБОР СРЕДСТВ РАЗРАБОТКИ**

Для разработки программного модуля была выбрана кроссплатформенная интегрированная среда разработки программного обеспечения для платформы .NET JetBrains Rider со спецификациями Avalonia, а также система управления реляционными базами данных MySQL[8][7].

Avalonia и MySQL были выбраны для разработки приложения по следующим причинам:

Avalonia:

* Многоплатформенность: Avalonia позволяет создавать приложения, которые работают на нескольких платформах, включая Windows, macOS и Linux. Это позволяет создавать универсальные решения, которые могут быть использованы на различных устройствах и операционных системах;
* XAML: Avalonia использует XAML для описания пользовательского интерфейса, что делает его удобным для разработки и облегчает создание сложных интерфейсов;
* Открытый исходный код: Avalonia является открытым исходным кодом, что означает, что разработчики могут свободно использовать и модифицировать его в соответствии с их потребностями.

MySQL:

* Устойчивость и надежность: MySQL является надежным и стабильным реляционным СУБД, которая используется для хранения данных в приложениях разного масштаба;
* Поддержка множества языков программирования: MySQL может быть использован в сочетании с различными языками программирования, такими как C#, Java, Python и другими, что делает его гибким и универсальным решением для хранения данных;
* Широкая поддержка и сообщество: MySQL имеет обширную документацию и поддержку сообщества, что упрощает процесс разработки и решает проблемы, с которыми могут столкнуться разработчики.

Таким образом, данные средства были выбраны для разработки программного модуля благодаря их гибкости, надежности, поддержке сообщества и возможности использования в сочетании с различными языками программирования.

Avalonia - это открытая, кроссплатформенная графическая система отображения. Она входит в .NET Foundation и является одним из самых активных проектов сообщества в области IT. Она используется для создания продвинутых и красивых интерфейсов. Avalonia поддерживает множество платформ.

На рисунке 3.1 изображен пример интерфейса JetBrains Rider: Avalonia.

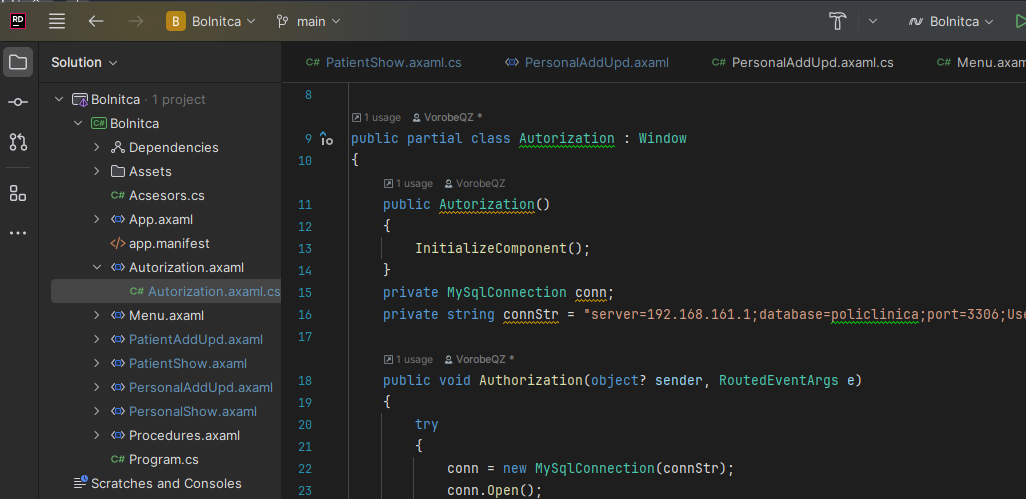


Рисунок 3.1 – Интерфейс JetBrains Rider: Avalonia

MySQL — одна из наиболее используемых систем управления базами данных. MySQL управляет реляционными базами данных, то есть такими, в которых таблицы связаны между собой. MySQL имеет клиент-серверную архитектуру. Это означает, что база данных хранится в одном источнике — на сервере. А связываться с ней могут клиенты — сторонние устройства. Клиенты отправляют запросы к БД, а потом получают от сервера информацию.

На рисунке 3.2 изображен пример интерфейса СУБД MySQL.

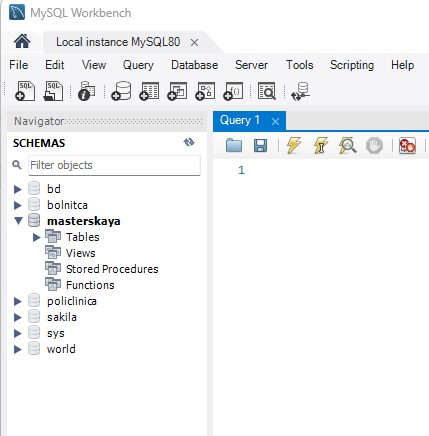


Рисунок 3.2 – Интерфейс СУБД MySQL

1. **УСТАНОВКА, НАСТОЙКА СЕРВЕРА БАЗЫ ДАННЫХ**

Установка операционной системы Linux Ubuntu Server 24.04 LTS представляет собой процесс развертывания программного обеспечения на компьютере с целью обеспечения функционирования в качестве сервера. Данная операционная система предоставляет широкий спектр функций и возможностей для управления сетевыми сервисами, хранения данных и выполнения задач в среде серверного окружения[6].

Процесс установки начинается с загрузки образа установочного диска операционной системы Ubuntu Server 24.04 LTS и его записи на запускаемый носитель, такой как USB-флеш-накопитель. Затем компьютер загружается с этого носителя, и пользователь следует инструкциям мастера установки.

Установка Ubuntu Server встречает с выбора языка, на котором будет произведена дальнейшая установка. Язык по умолчанию выбран английский.

На рисунке 4.1 изображен выбор языка.

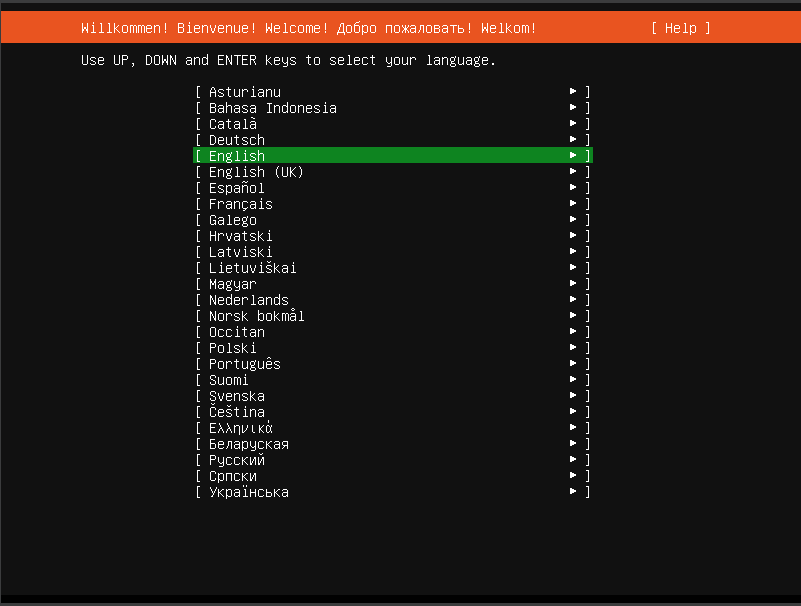


Рисунок 4.1 – Выбор языка

После выбора языка появляется следующее окно с выбором раскладки клавиатуры, здесь также по умолчанию выбран английский.

На рисунке 4.2 изображено окно с выбором раскладки клавиатуры.

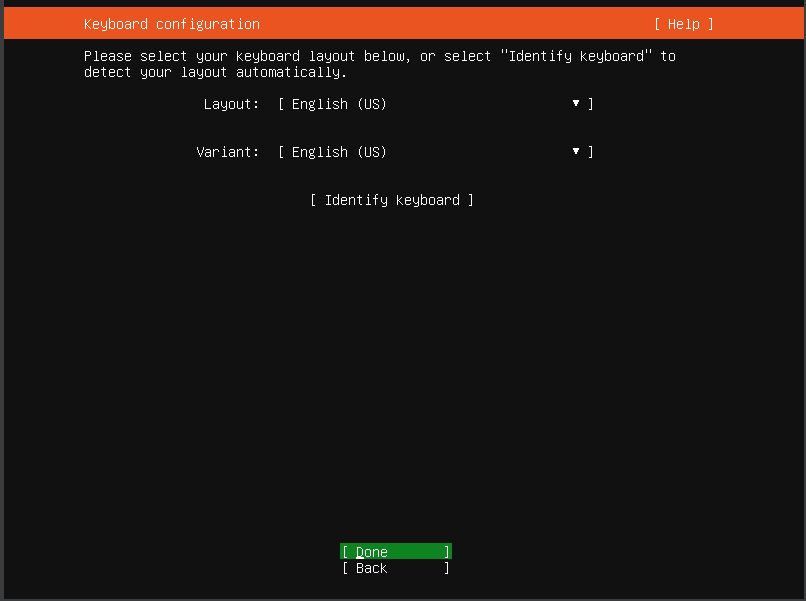


Рисунок 4.2 – Выбор раскладки клавиатуры

После выбора раскладки клавиатуры необходимо выбрать тип установки сервера.

На рисунке 4.3 изображен тип установки сервера.

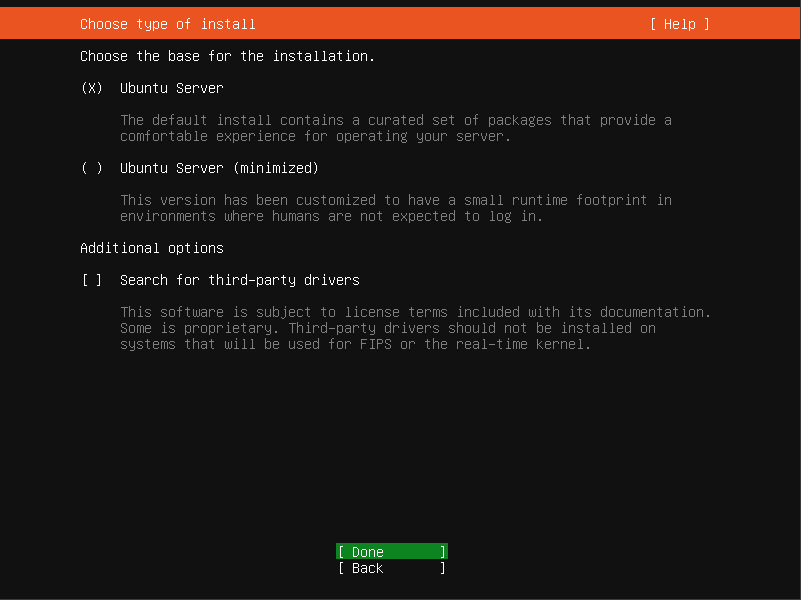


Рисунок 4.3 – Выбор типа установки сервера

После выбора типа установки появляется окно с настройкой сети. Сетевые интерфейсы по умолчанию настроены на DHCP.

На рисунке 4.3 изображена настройка сети.

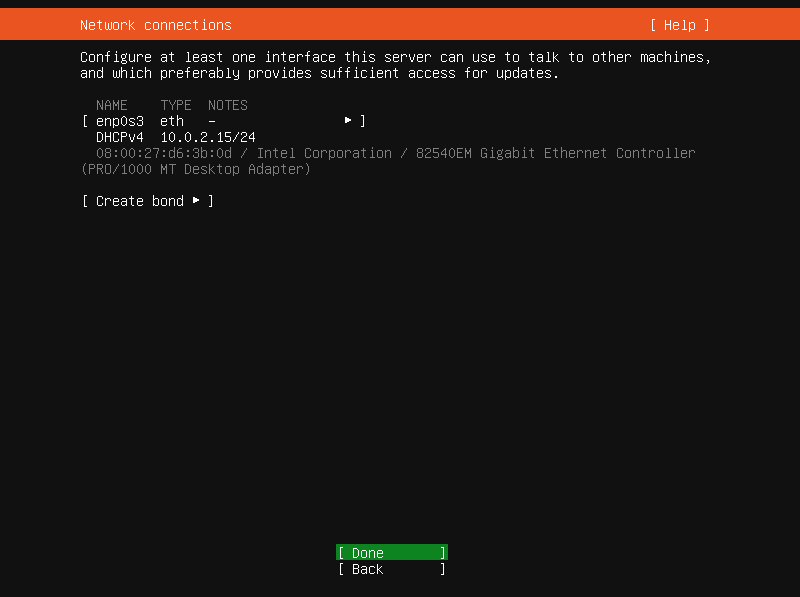


Рисунок 4.4 – Настройка сети

После настройки сети появится окно разметки дисков.

На рисунке 4.5 изображена настройка разметки дисков.

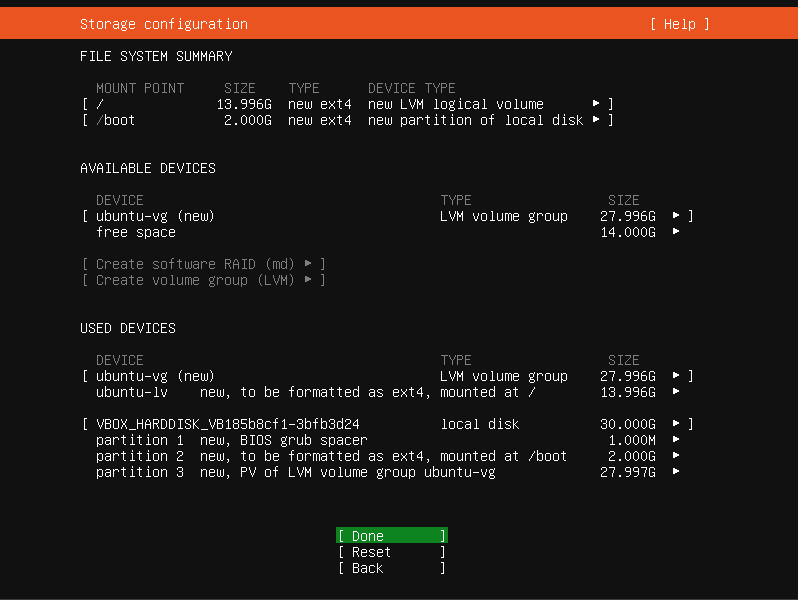


Рисунок 4.5 – Настройка разметки дисков

После чего уже переходим непосредственно к настройке учетной записи. В этом окне нужно ввести информацию о пользователе и сервере, придумать пароль, который в дальнейшем будем вводить в процессе работы.

На рисунке 4.6 изображена настройка учетной записи.

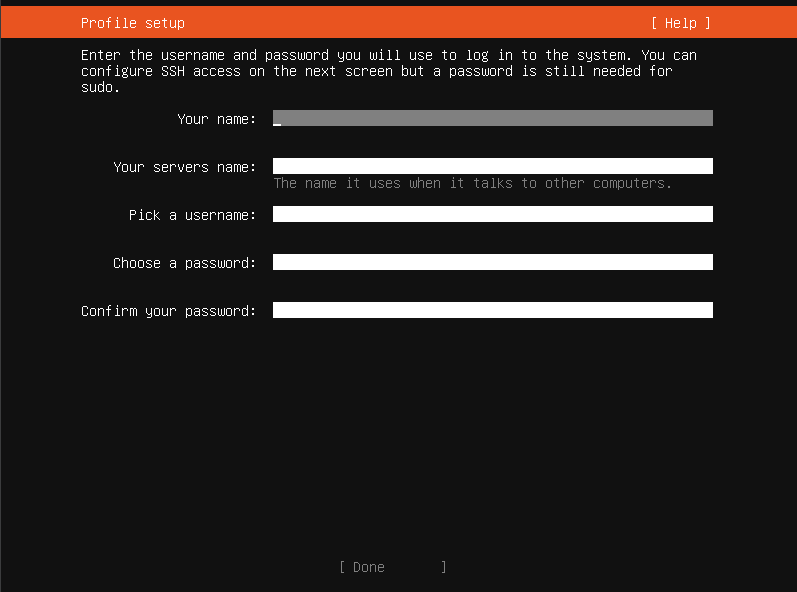


Рисунок 4.6 – Настройка учетной записи

После создания учётной записи начнётся установка ОС.

На рисунке 4.7 изображен процесс установки ОС.

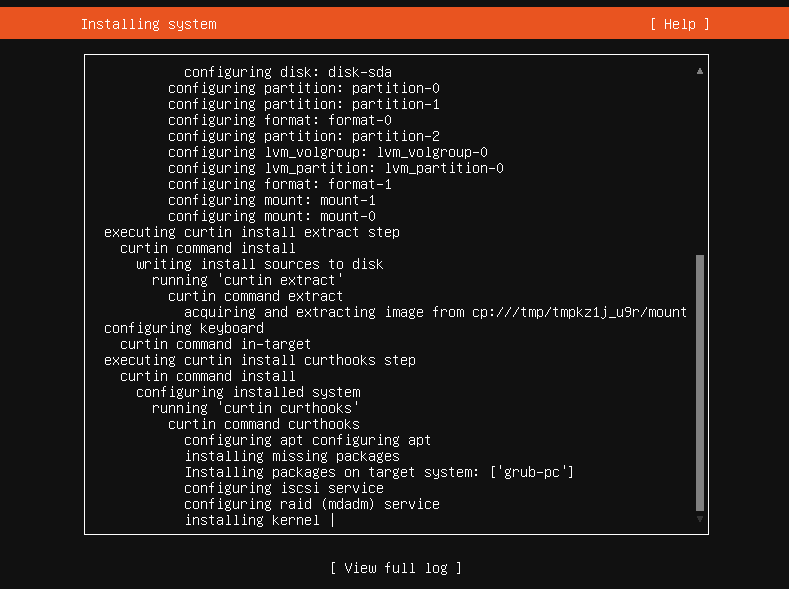


Рисунок 4.7 – Процесс установки

Следующим этапом является установка самого сервера базы данных на выбранном сервере. Этот процесс требует внимательного изучения инструкций и следования им, чтобы гарантировать корректную установку и последующую работоспособность сервера. В ходе процесса установки были указаны основные параметры, такие как имя хоста, порт, аутентификация пользователей и другие настройки, которые будут влиять на работу сервера в дальнейшем[1].

На рисунке 4.8 изображена установка сервера БД.

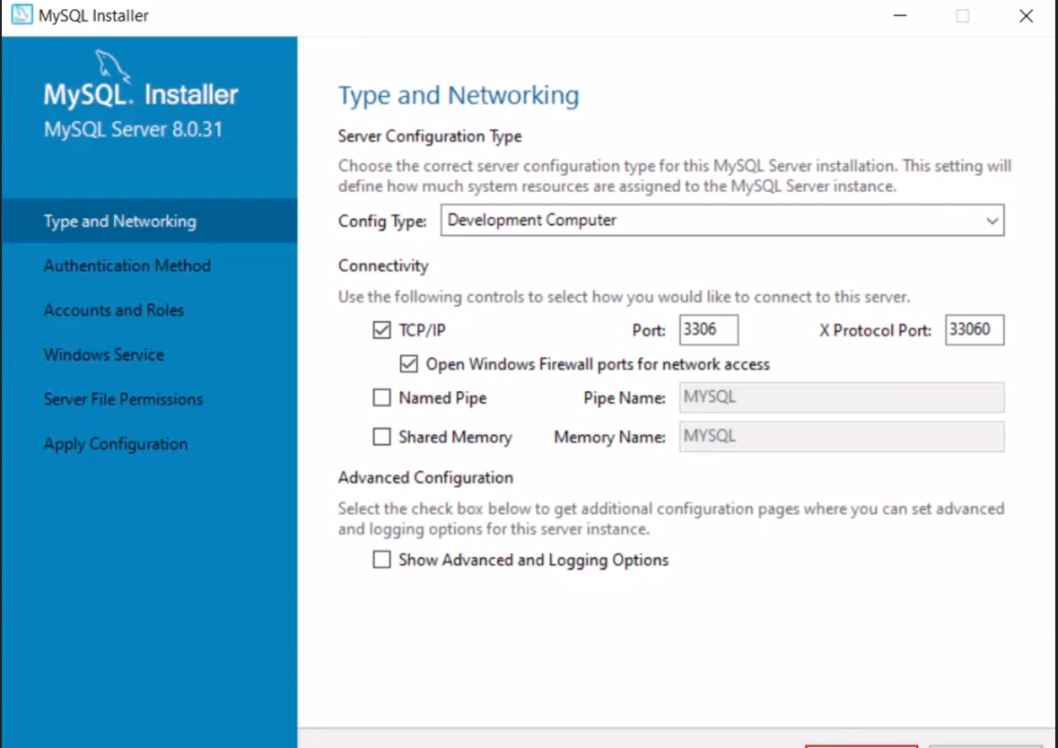


Рисунок 4.8 – Установка сервера БД

После установки сервера, необходимо настроить параметры базы данных в соответствии с требованиями проекта. Это включает в себя настройку различных параметров, таких как размеры таблиц, типы данных, ограничения и многое другое.

После настройки параметров сервера базы данных, была создана база данных, которая была необходима для реализации информационной системы предприятия. В рамках создания базы данных были разработаны и созданы необходимые таблицы, содержащие информацию об учителях, учениках, расписания занятий и других сущностях, которые были нужны для функционирования системы.

На заключительном этапе, проверяется работоспособность сервера базы данных и корректность настроек. Это включает в себя проверку различных функций базы данных, таких как создание, чтение, обновление и удаление данных, а также проверка параметров базы данных на соответствие требованиям проекта. При этом были проведены тестовые запросы к базе данных, чтобы убедиться в правильности функционирования сервера и отсутствии ошибок. Также был проведен анализ производительности сервера базы данных, чтобы оценить его возможности в условиях реальной нагрузки.

1. **РАЗРАБОТКА БАЗЫ ДАННЫХ, ПРИЛОЖЕНИЯ**
   1. **Таблицы базы данных, структура. Нормализация базы данных**

В рамках преддипломной практики, после успешной установки и настройки сервера базы данных MySQL, следующим шагом является создание необходимых таблиц в MySQL Workbench.

Первым делом запускаем MySQL Workbench и подключаемся к серверу базы данных MySQL, используя необходимые учётные данные и настройки.

После успешного подключения, переходим на вкладку «Schema» в левой части окна программы. Здесь видим список существующих схем. Если необходимая база данных еще не создана, нажмите на кнопку «Create Schema», данном меню можно настроить имя базы данных, а также ее кодировку. По окончанию настраивания параметров, необходимо нажать «Apply», после чего база данных создастся и появится в списке существующих баз[13].

На рисунке 5.1.1 изображено меню создания БД.

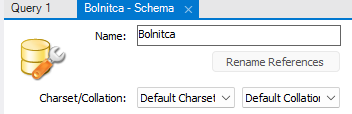


Рисунок 5.1.1 – Меню создания БД

Выбираем созданную базу данных и нажимаем на кнопку «Create Table», чтобы начать создание новой таблицы. В открывшемся окне «Create Table» вводим имя таблицы и добавляем необходимые столбцы, указав их имена, типы данных и другие свойства, например, является ли столбец ключевым, есть ли у него ограничение NOT NULL и т.д.

На рисунке 5.1.2 изображено создание таблицы «персонал».

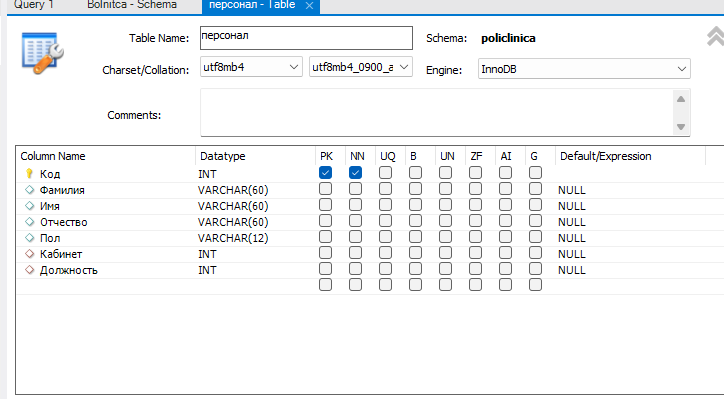


Рисунок 5.1.2 – Создание таблицы «персонал»

Создание остальных таблиц БД были произведены аналогичным способом.

На рисунке 5.1.3 изображены все таблицы БД.

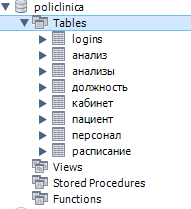


Рисунок 5.1.3– Все созданные таблицы БД

* 1. **Схема базы данных**

Для разработки базы данных была применена СУБД MySQL. На Схеме базы данных, отражающей архитектуру БД показано семь таблиц: «Кабинет», «Анализ», «Анализы», «Должность», «Пациент», «Персонал», «Расписание».

Таблица «Должность» содержит информацию о должностях медицинского персонала. Она включает идентификатор, наименование должности.

Таблица «Кабинет» содержит информацию о кабинете. Она включает идентификатор ,телефон закреплённый за кабинетом, наименование кабинета.

Таблица «Персонал» содержит информацию о персонале в медицинском учреждении. Она включает идентификатор, имя, фамилия, отчество, пол, идентификатор кабинета, идентификатор должности.

Таблица «Пациент» содержит информацию о пациентах. Она включает идентификатор, имя, фамилия, отчество, пол, возраст, телефон и полис.

Таблица «Анализ» содержит информацию о проводимых анализах в медицинском учреждении. Она включает идентификатор и наименование анализа.

Таблица «Анализы» содержит информацию об назначенных анализах пациенту и их результаты. Она включает идентификатор,идентификатор анализа, идентификатор пациента,идентификатор персонала, дата проведения анализа, результат анализа.

Таблица «Расписание» содержит информацию о расписании проводимых анализах. Она включает идентификатор,идентификатор персонала, идентификатор анализа, время.

На рисунке 5.2.1 изображена схема БД.

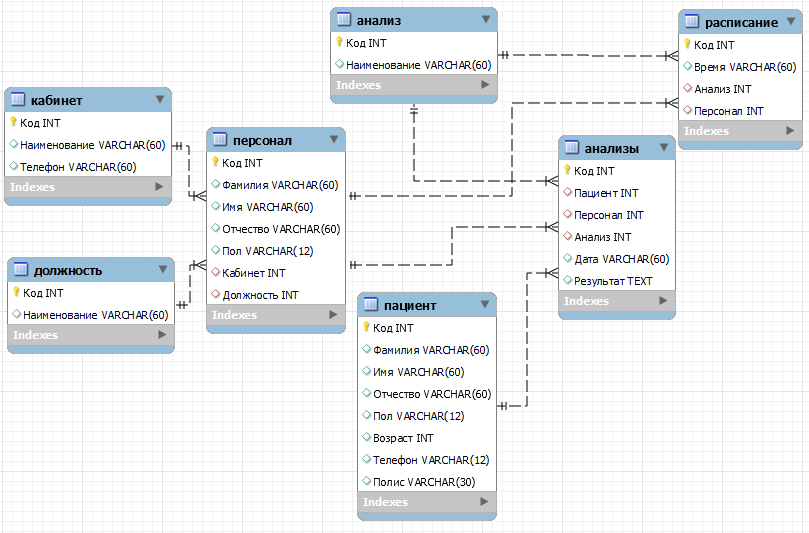


Рисунок 5.2.1– Схема БД

* 1. **Интерфейс приложения**

Технология WPF является часть экосистемы платформы .NET и представляет собой подсистему для построения графических интерфейсов. С помощью WPF будет разрабатываться приложение [3].

Приложение имеет окно авторизации, где пользователь может ввести данные своей учетной записи. Данные учетных записей хранятся в базе данных в таблице «logins».

На рисункe 5.3.1 представлено окно авторизации.

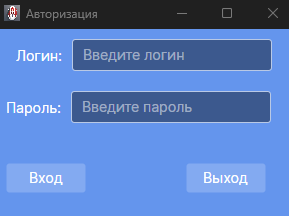


Рисунок 5.3.1 – Окно авторизации приложения

В случае если пользователь вводите неверные логин или пароль появляется надпись, информирующая пользователя об этом.

На рисункe 5.3.2 представлено окно авторизации после неправильного ввода данных авторизации.

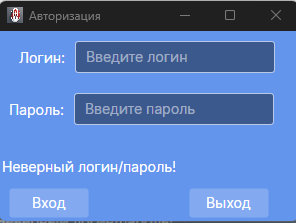


Рисунок 5.3.2 – Окно авторизации после неправильного ввода логина и пароля

После успешной авторизации пользователь получает доступ к просмотру таблиц данных. Каждая таблица отображается в отдельной форме, к которой можно перейти, нажав соответствующую кнопку.

На рисункe 5.3.3 представлено меню для перехода между формами приложения для дальнейшей работы с таблицами.

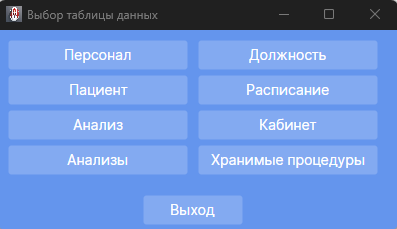


Рисунок 5.3.3 – Меню

Например, форма «Персонал» содержит элемент для табличного отображения данных. Пользователь может просматривать, фильтровать, сортировать и редактировать данные. Кнопка «Удалить» удаляет выбранную запись из базы данных. Кнопки «Добавить» и «Изменить» открывают новую форму, где пользователь может ввести данные для добавления или изменения записи.

На рисункe 5.3.4 представлена форма «Персонал».

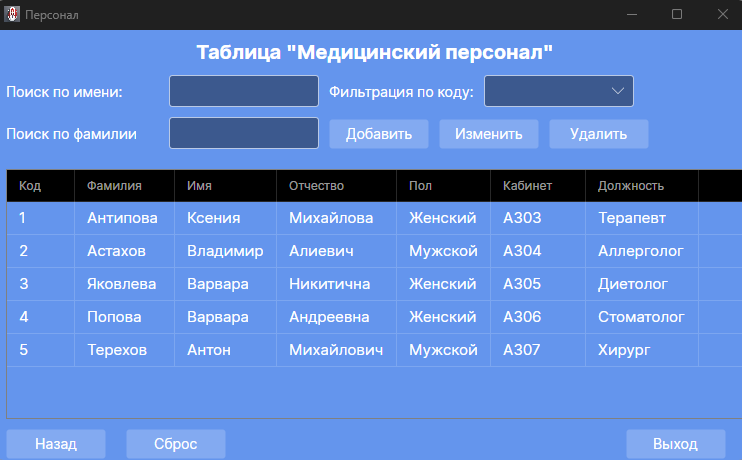


Рисунок 5.3.4 – Форма для отображения таблицы «Персонал»

Для добавления новой записи пользователь нажимает на функциональную кнопку «Добавить». После нажатия открывается новая форма, в которой пользователь должен внести данные в специальные поля.

На рисункe 5.3.5 представлена форма для добавления новой записи.

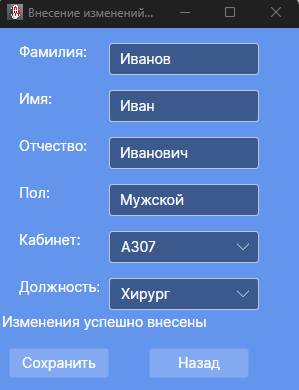


Рисунок 5.3.5 – Форма добавления для таблицы «Персонал»

По завершению пользователь нажимает на кнопку «Сохранить» и возвращается на форму с таблицей «Персонал» с помощью кнопки «Назад».

На рисункe 5.3.6 представлена форма после добавления новой записи.

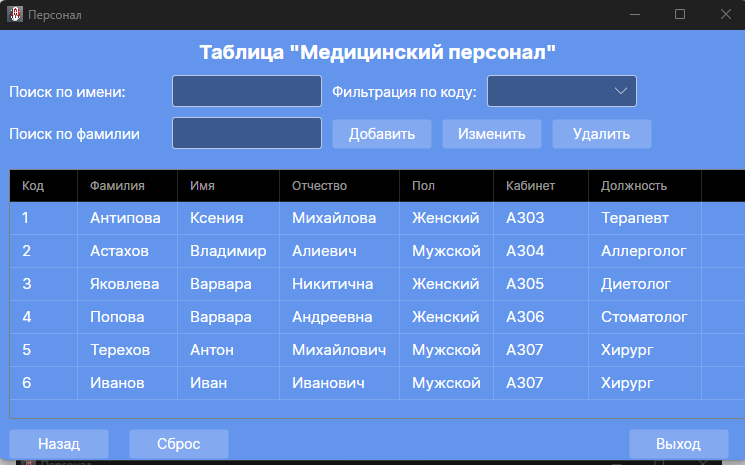


Рисунок 5.3.6 – Форма после добавления новых данных

В случае изменения каких-то данных или ввода неверных данных, пользователь может изменить ошибочную запись. Для этого пользователь должен выбрать запись которую он хочет изменить.

На рисункe 5.3.7 представлена форма с выделенной записью для изменения.

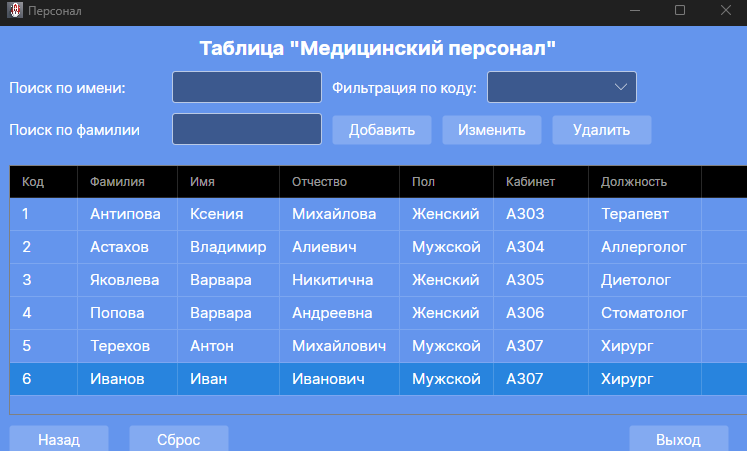


Рисунок 5.3.7 – Форма с выделенной записью для изменения

После нажимаем на кнопку «Изменить» и в специальной форме вносим изменения в выбранную запись.

На рисункe 5.3.8 представлена форма для редактирования запись.

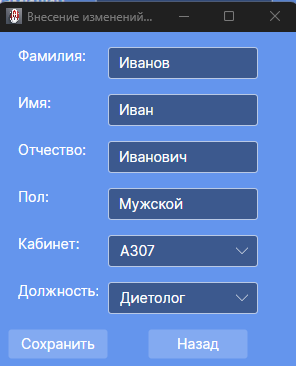


Рисунок 5.3.8 – Форма для изменения записи\

По завершению пользователь нажимает на кнопку «Сохранить» и возвращается на форму с таблицей «Персонал» с помощью кнопки «Назад».

На рисункe 5.3.9 представлена форма после изменения записи.

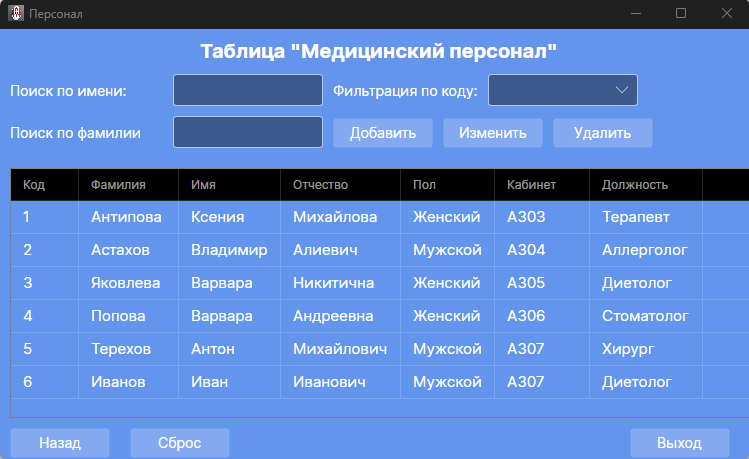


Рисунок 5.3.9 – Форма после изменения ошибочных данных

Для того чтобы удалить запись пользователь должен выбрать запись, которую он хочет удалить и после нажать на кнопку «Удалить».

На рисункe 5.3.10 представлена форма после удаления записи.

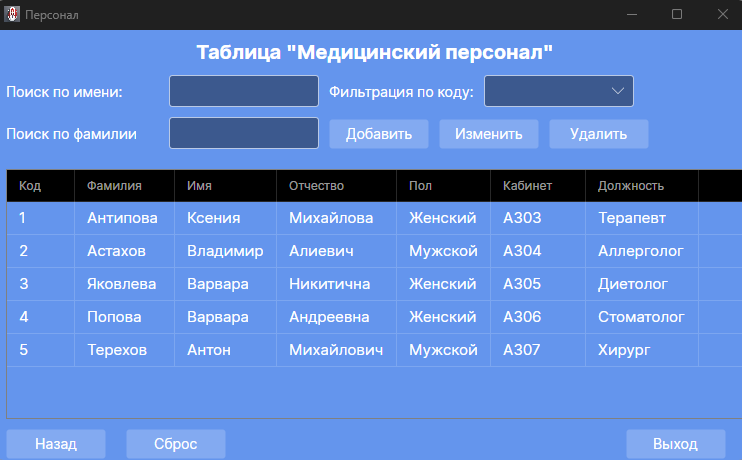


Рисунок 5.3.10 – Форма после удаления внесённых данных

На остальных таблицах добавление, удаление, редактирование происходит аналогичным способом.

* 1. **Реализация поиска в таблице базы данных**

Для поиска реализован метод перебора данных по выбранным столбцам, после ввода данных для поиска будут отображаться только те записи, соответствующие результата поиска. Поиск работает в реальном времени,после начала ввода запроса в специальное поле. Кнопка сброса позволяет сбросить параметры и отобразить все записи.

На рисунках 5.4.1-5.4.2 представлен поиск по одному и двум столбцам.

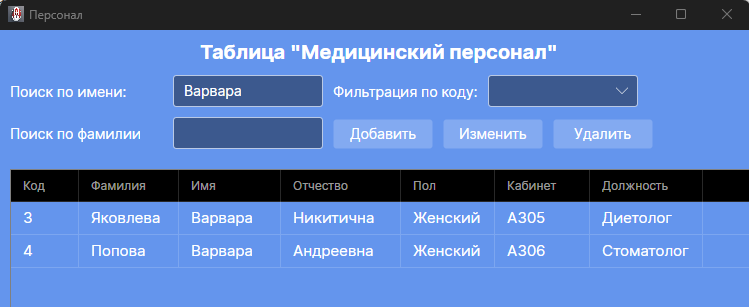


Рисунок 5.4.1 – Форма после поиска по имени

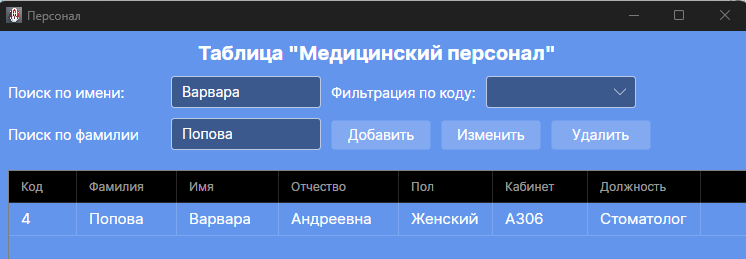


Рисунок 5.4.2 – Форма после поиска по имени и фамилии

На остальных таблицах поиск выполнен аналогичным способом.

* 1. **Реализация фильтрации данных**

Для фильтрации данных предусмотрено поле со списком, где пользователь может выбрать параметр фильтрации. После выбора параметра будут отображаться только записи, соответствующие выбранному параметру. Кнопка сброса позволяет сбросить параметры фильтрации и отобразить все записи.

На рисунке 5.5.1 представлен фильтрация.

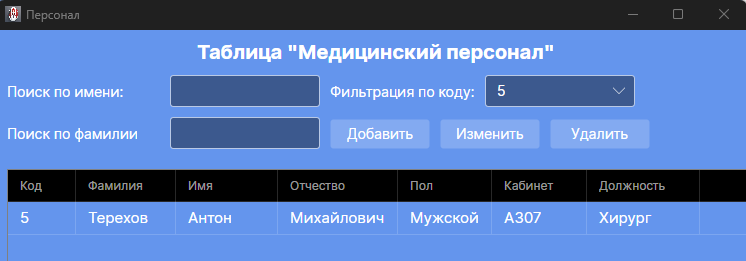


Рисунок 5.5.1 – Форма после фильтрации по коду

На остальных таблицах фильтрация выполнена аналогичным способом.

* 1. **Реализация запросов к таблице базы данных**

Запрос в СУБД - это запрос, выполняемый пользователем или приложением для извлечения данных, хранящихся в базе данных, или манипулирования ими. Этот запрос обычно формулируется с использованием языка структурированных запросов (SQL) или интерфейса запросов, предоставляемого СУБД.

Реализация запросов SQL может включать в себя несколько концепций и методов, таких как объединение таблиц, подзапросы, использование агрегатных функций, оконные функции и другие. Вот несколько примеров сложных запросов SQL, которые могут включать в себя указанные методы:

INNER JOIN – команда, которая объединяет данные из разных таблиц на основе совпадения значений в указанных столбцах.

WHERE – команда, которая фильтрует строки в результирующем наборе данных на основе заданных условий.

GROUP BY – команда, которая группирует строки в результирующем наборе данных на основе значений указанных столбцов и позволяет выполнять агрегацию данных внутри каждой группы.

LIKE SQL — это оператор, который используется для поиска строк, содержащих определённый шаблон символов.

Запрос «LIKE» выводит все значения из таблицы «Персонал», которые в поле «Имя» начинаются на букву «В».

На рисунке 5.6.1 изображен SQL запрос с использованием операнда «Like» и его результат.

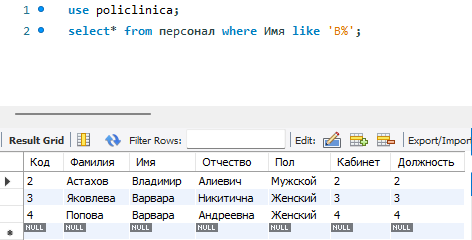


Рисунок 5.6.1 – SQL запрос и его результат

Запрос «WHERE» выводит все значения из таблицы «Персонал», где значение в столбце пол является «Мужской»

На рисунке 5.6.2 изображен SQL запрос с использованием операнда «WHERE» и его результат.

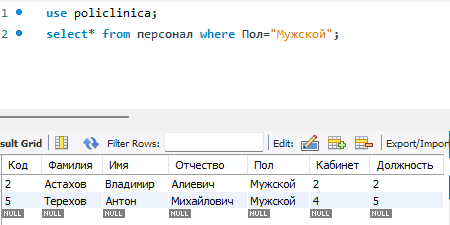


Рисунок 5.6.2 – SQL запрос и его результат

Запрос «GROUP BY» объединяет значения в поле «Имя» из таблицы «Персонал» и добавляет новый столбец с количеством повторяющихся имен.

На рисунке 5.6.3 изображен SQL запрос с использованием операнда «GROUP BY» и его результат.

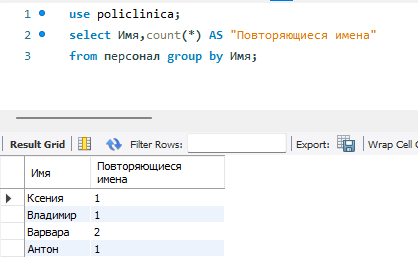


Рисунок 5.6.3 – SQL запрос и его результат

Запрос «INNER JOIN» объединяет данные из таблиц «Персонал», «Кабинет» и «Должность».

На рисунке 5.6.4 изображен SQL запрос с использованием операнда «INNER JOIN» и его результат.

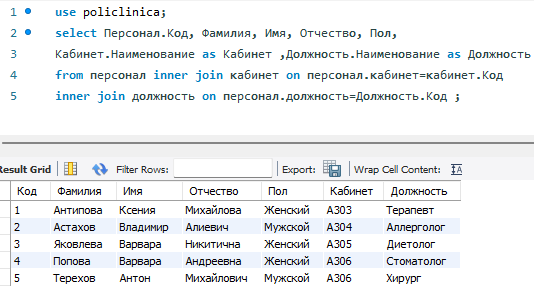


Рисунок 5.6.4 – SQL запрос и его результат

* 1. **Хранимые процедуры**

Хранимая процедура ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA" \o "Английский язык) Stored procedure) — объект [базы данных](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85" \o "База данных), представляющий собой набор [SQL](https://ru.wikipedia.org/wiki/SQL" \o "SQL)-инструкций, который компилируется один раз и хранится на сервере[15]. Хранимые процедуры очень похожи на обыкновенные [процедуры](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D0%B4%D1%83%D1%80%D0%B0_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)" \o "Процедура (программирование)) [языков](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F" \o "Язык программирования) высокого уровня, у них могут быть входные и выходные параметры и локальные [переменные](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)" \o "Переменная (программирование)), в них могут производиться числовые вычисления и операции над символьными данными, результаты которых могут присваиваться переменным и параметрам. В хранимых процедурах могут выполняться стандартные операции с базами данных (как [DDL](https://ru.wikipedia.org/wiki/DDL" \o "DDL), так и [DML](https://ru.wikipedia.org/wiki/DML" \o "DML)). Кроме того, в хранимых процедурах возможны циклы и ветвления, то есть в них могут использоваться инструкции управления процессом исполнения.

Хранимые процедуры похожи на определяемые пользователем функции (UDF). Основное различие заключается в том, что пользовательские функции можно использовать как и любое другое выражение в SQL запросе, в то время как хранимые процедуры должны быть вызваны с помощью функции CALL:

Хранимые процедуры могут возвращать множества результатов, то есть результаты запроса SELECT. Такие множества результатов могут обрабатываться, используя курсоры, другими хранимыми процедурами, возвращая указатель результирующего множества, либо же приложениями. Хранимые процедуры могут также содержать объявленные переменные для обработки данных и курсоров, которые позволяют организовать цикл по нескольким строкам в таблице. Стандарт SQL предоставляет для работы выражения IF, LOOP, REPEAT, CASE и многие другие. Хранимые процедуры могут принимать переменные, возвращать результаты или изменять переменные и возвращать их, в зависимости от того, где переменная объявлена.

Первая хранимая процедура заменяет все кодовые значения в таблице «Расписание» на присвоенные им текстовые значения.

На рисунке 5.7.1 представлен SQL-запрос на создание хранимой процедуры.

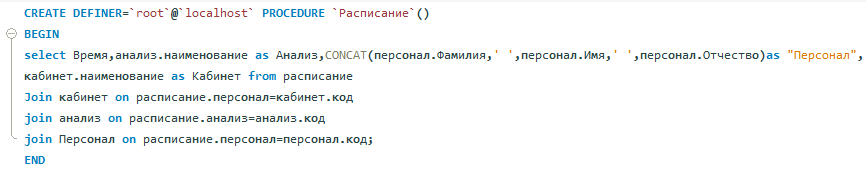


Рисунок 5.7.1 – SQL-запрос на создание хранимой процедуры

После создания хранимой процедуры можно ее выполнить.

На рисунке 5.7.2 представлен результат выполения хранимой процедуры.



Рисунок 5.7.2 – Результат выполнения хранимой процедуры

Вторая хранимая процедура заменяет все кодовые значения в таблице «Персонал» на присвоенные им текстовые значения.

На рисунке 5.7.3 представлен SQL-запрос на создание хранимой процедуры.

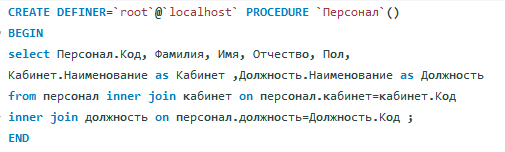


Рисунок 5.7.3 – SQL-запрос на создание хранимой процедуры

После создания хранимой процедуры можно ее выполнить.

На рисунке 5.7.4 представлен результат выполения хранимой процедуры.

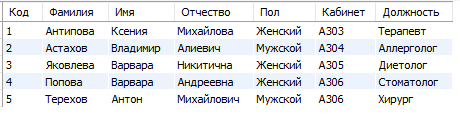


Рисунок 5.7.4 – Результат выполнения хранимой процедуры

Все созданные хранимые процедуры сохраняются в специальном каталоге базы данных. В дальнейшем из этого каталога их можно выполнить или же отредактировать.

На рисунке 5.7.5 представлен каталог с созданными хранимыми процедурами.



Рисунок 5.7.5 – Каталог с созданными хранимыми процедурами

* 1. **Импорт, экспорт информации базы данных**

Импорт и экспорт данных в MySQL - это процессы, связанные с перемещением данных в базе данных MySQL или из нее[12].

Импорт данных в MySQL - это процесс добавления данных из внешних источников в базу данных MySQL. Это может быть сделано с помощью различных инструментов и методов, таких как командная строка, графические интерфейсы, или специализированные программы.

Экспорт данных из MySQL - это процесс извлечения данных из базы данных MySQL и сохранения их в другом формате, который может быть использован другими системами или программами. Это также может быть сделано с помощью различных инструментов и методов, таких как mysqldump, SELECT INTO OUTFILE, или другие программы.

Оба процесса являются важными для управления данными в MySQL и могут быть использованы для различных целей, таких как резервное копирование, миграция данных, или обмен данными с другими системами.

Для того чтобы выполнить экспорт данных в специальном меню открываем инструмент «Data Export». Здесь необходимо выбрать базу данных, из которой мы хотим экспортировать данные, при этом выбрать необходимые таблицы. При этом необходимо сразу указав путь к файлу, в котором будет храниться наша экспортированная база данных.

На рисунке 5.8.1 изображен экспорт базы данных с помощью функционала СУБД «My SQL Workbench».

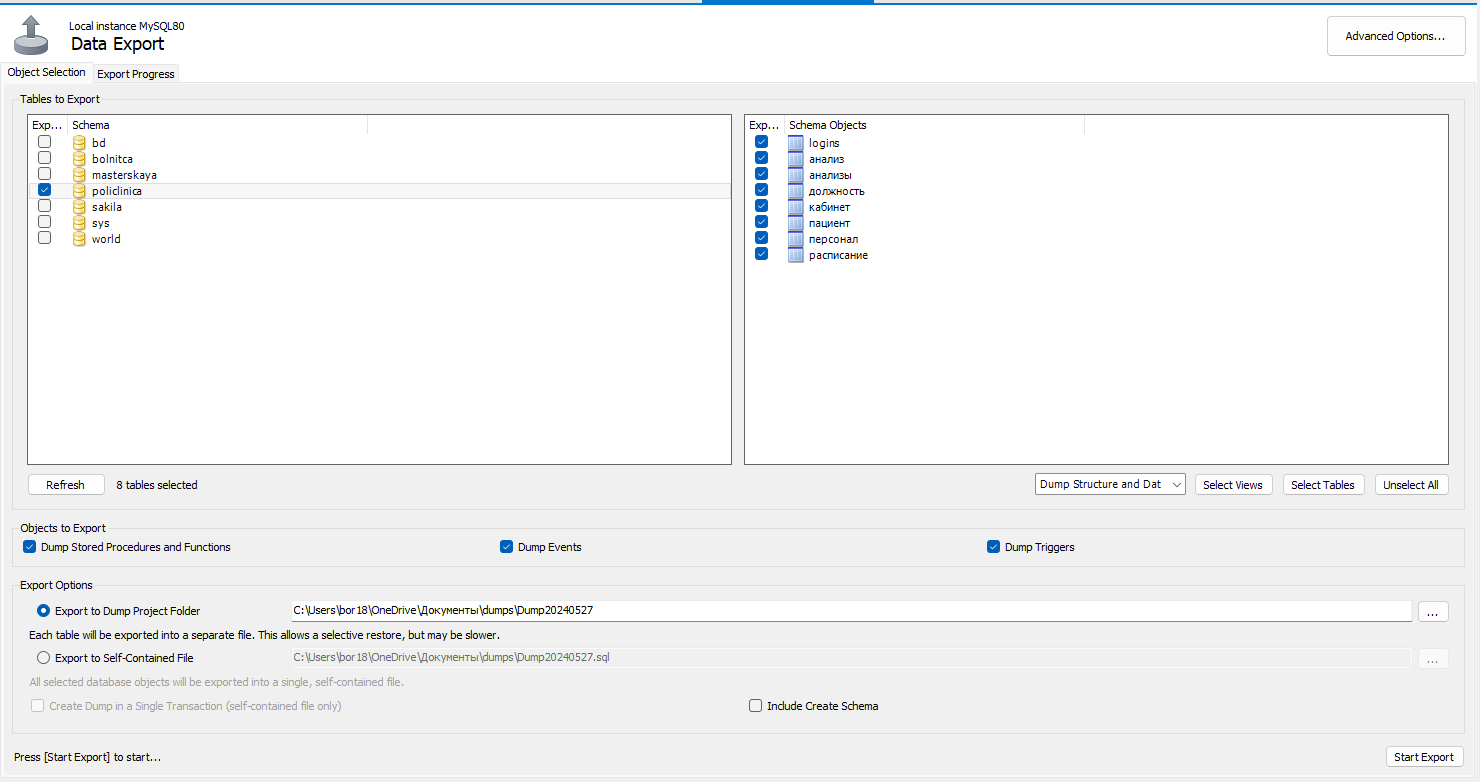


Рисунок 5.8.1 – Экспорт базы данных

На рисунке 5.8.2 изображено успешное завершение экспорта базы данных.

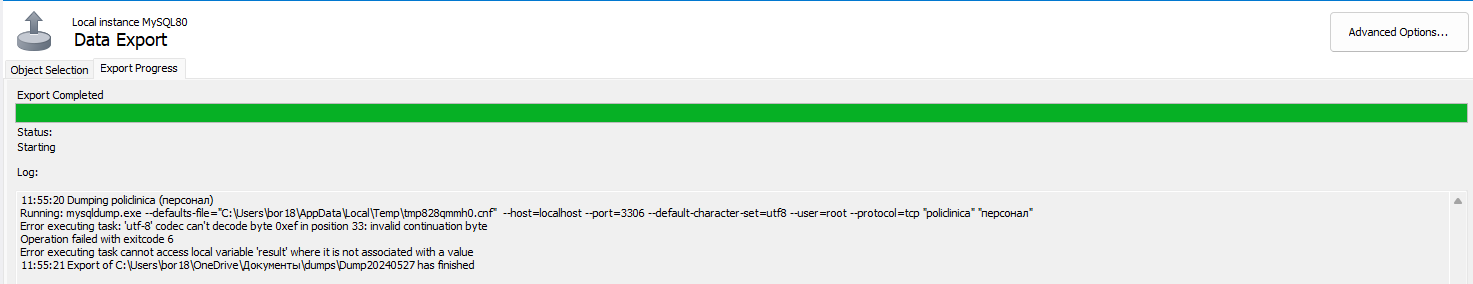


Рисунок 5.8.2 – Успешное завершение экспорта БД

На рисунке 5.8.3 изображен импорт базы данных с помощью функционала СУБД «My SQL Workbench».

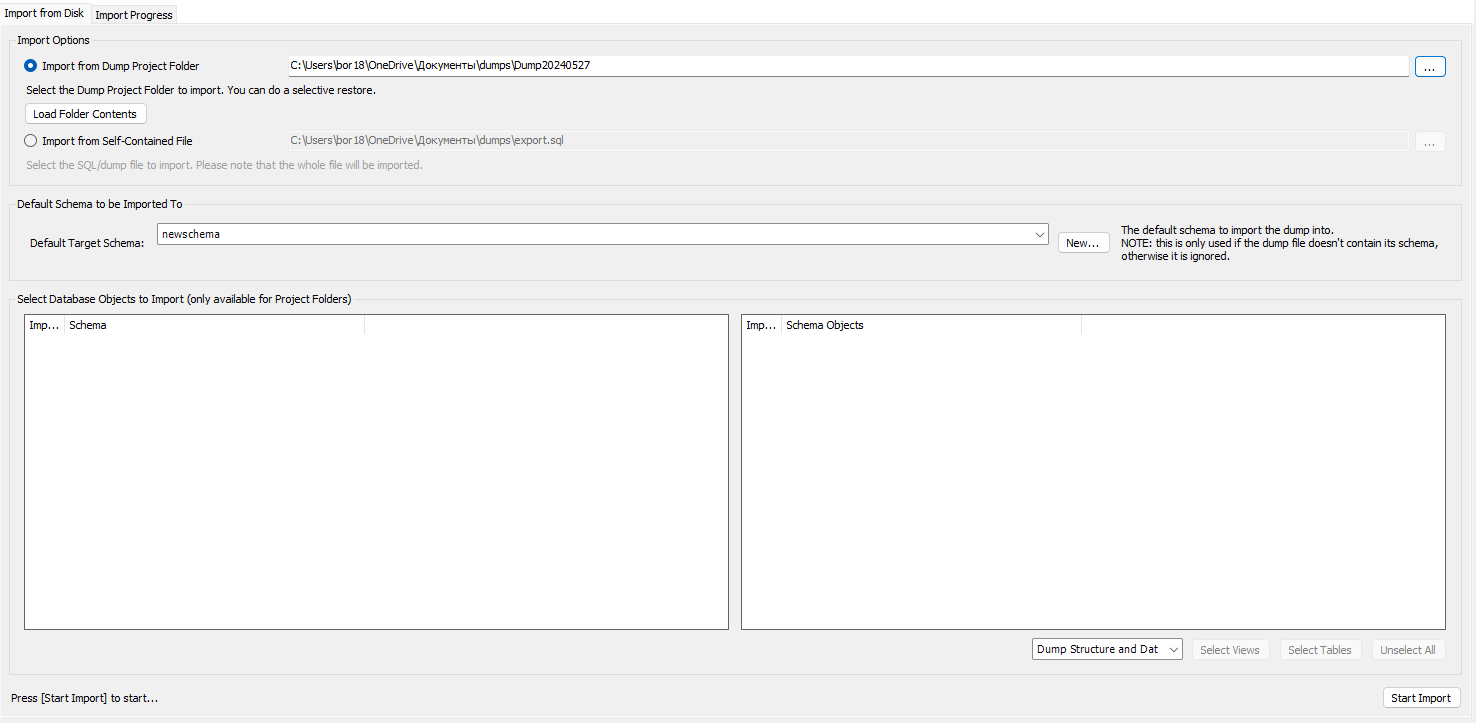


Рисунок 5.8.3 – Импорт базы данных

На рисунке 5.8.4 изображено успешное завершение импорта базы данных.

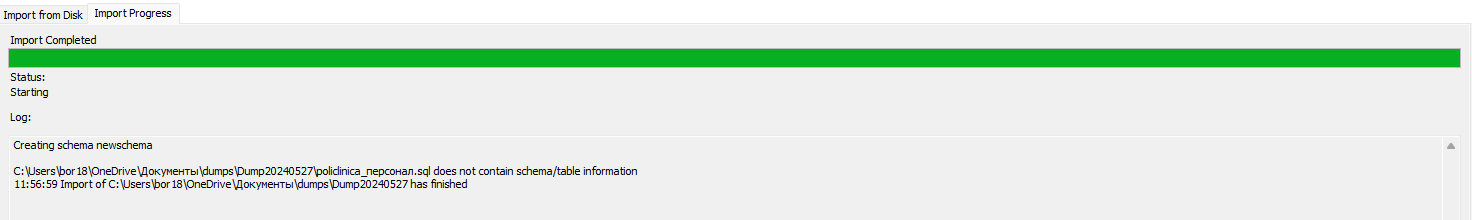


Рисунок 5.8.4 – Успешное завершение импорта БД

* 1. **Резервное копирование, восстановление информации базы данных**

Резервное копирование и восстановление данных в базе данных MySQL является неотъемлемой частью обеспечения надежности и защиты информации. Для уменьшения риска необратимой потери данных, необходимо периодически создавать копии баз данных. Благодаря этому, можно будет восстановить данные после различных видов сбоев:

* Сбой носителя;
* Ошибки пользователей БД (например, удаление таблицы по ошибке);
* Сбои оборудования;
* Стихийные бедствия.

Кроме того, резервные копии баз данных полезны и при выполнении повседневных административных задач, например для копирования баз данных с одного сервера на другой.

Резервное копирование было выполнено на основе экспорта базы данных. Для этого был использован инструмент "Data Export", который позволяет выбрать необходимые данные и указать путь к файлу, в котором будет храниться резервная копия.

После того как резервное копирование было выполнено успешно. Необходимо проверить работоспособность резервной копии. Для осуществления проверки была создана тестовая база данных для импорта, что позволило убедиться в корректности восстановления данных из резервной копии.

На рисунке 5.9.1 изображено создание базы данных для дальнейшего импорта.

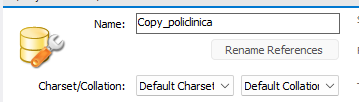


Рисунок 5.9.1 – Создание БД

После успешного создания тестовой базы данных был выполнен импорт данных при помощи инструмента «Data Import» и ранее созданной резервной копии.

На рисунке 5.9.2 изображен импорт базы данных с помощью функционала СУБД «My SQL Workbench».

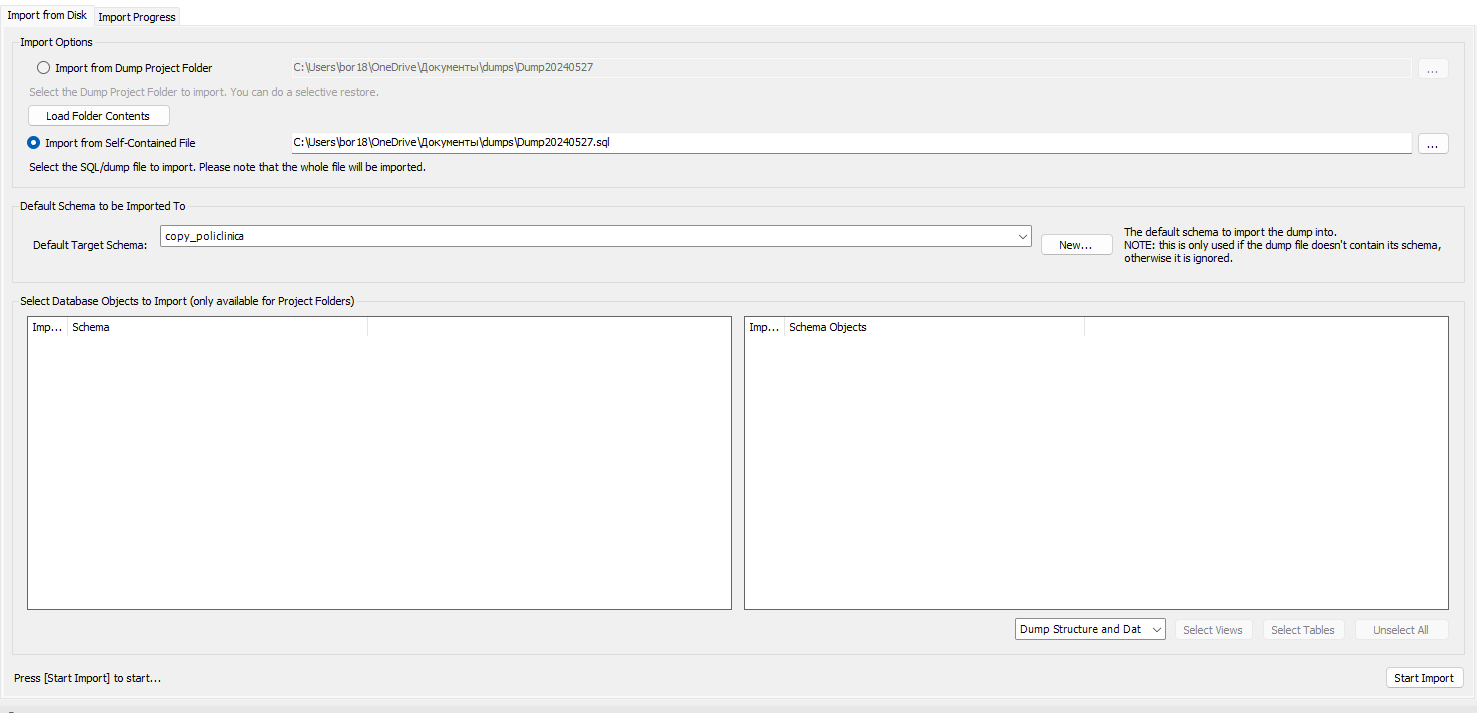


Рисунок 5.9.2 – Импорт базы данных

На рисунке 5.9.3 изображено успешное проведённое резервное копирование.

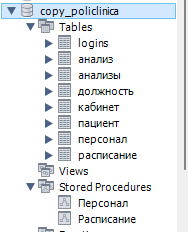


Рисунок 5.9.3 – Восстановление резервной копии

* 1. **Формирование отчётов для печати**

В приложении была реализована возможность формирования отчёта для печати. Это функция поможет сэкономить время для переноса данных необходимых для печати. Для того чтобы сформировать отчёт необходимо перейти на специальную форму,нажать на кнопку «Печать».

На рисунке 5.10.1 изображено форма с возможностью формирования отчёта.

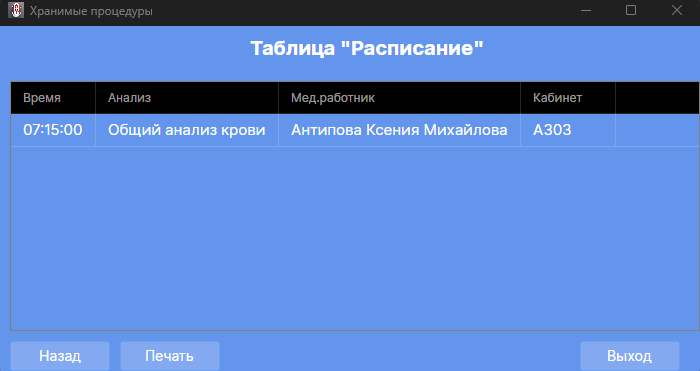


Рисунок 5.10.1 – Форма с возможностью формирования отчёта

После нажатия кнопки на рабочем столе создаётся файл формата .xlsx, который можно открыть и уже его распечатать.

На рисунке 5.10.2 изображен автоматически сформированный отчёт.

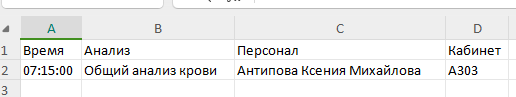


Рисунок 5.10.2 – Отчёт

1. **ТЕСТИРОВАНИЕ, СОПРОВОЖДЕНИЕ И ОБСЛУЖИВАНИЕ ПРОГРАММНОГО ПРОДУКТА**
   1. **Установка серверного программного обеспечения, программного обеспечения рабочих станций, системные требования**

Системные требования к серверу базы данных:

* Процессор: Intel Core или Xeon 3 ГГц (или Dual Core 2 ГГц) или аналогичный процессор AMD;
* Оперативная память: 4 ГБ (рекомендуется использовать 6 ГБ);
* Графические ускорители: nVidia или ATI с поддержкой OpenGL 1.5 или выше;
* Дисковая подсистема: RAID-массив из 4 дисков по 80 ГБ;
* ОС: Linux Ubuntu 22.04 или аналогичные операционные системы.

Установка серверного программного обеспечения MySQL включает в себя несколько этапов, которые необходимо выполнить для настройки и запуска сервера баз данных MySQL на компьютере. MySQL - это популярная система управления базами данных, которая используется для хранения, управления и извлечения данных в различных приложениях и системах.

MySQL Workbench — это официальная интегрированная среда разработки (IDE), созданная специально для MySQL корпорацией Oracle, которая владеет и поддерживает базы данных MySQL и MariaDB. Она построена на базе среды Eclipse IDE, поэтому вы получаете все преимущества этой платформы, включая мощный редактор кода, интеграцию с системой контроля версий, подсветку синтаксиса и многое другое. Workbench также доступен в виде интегрированного решения для Microsoft Windows и других операционных систем.

В данной работе мы установим Workbench на ОС Ubuntu, с графическим интерфейсом.

Прежде всего, необходимо обновить пакеты системы.

На рисунке 6.1.1 изображена команда обновления системы.



Рисунок 6.1.1 – Обновление системы

После чего уже установим сам MySQLServer.

На рисунке 6.1.2 изображена команда установки MySQLServer.



Рисунок 6.1.2 – Установка MySQLServer

Далее, когда установка завершится, проверим работоспособность сервера.

На рисунке 6.1.3 изображена проверка MySQLServer.

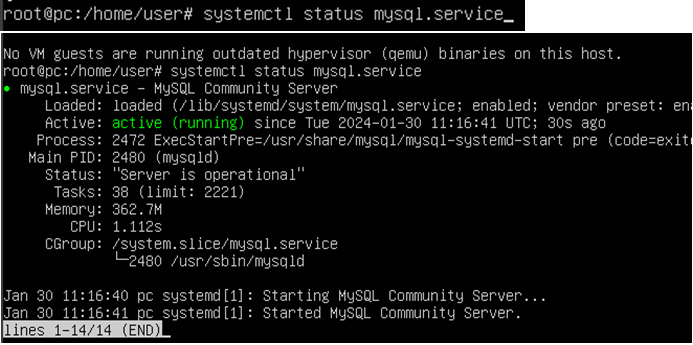


Рисунок 6.1.3 – Проверка MySQLServer

После этого необходимо установить графическую оболочку.

На рисунке 6.1.4 изображена установка графической оболочки.

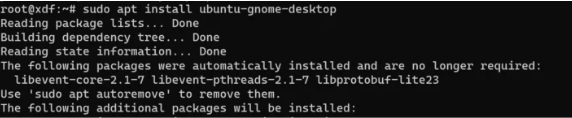


Рисунок 6.1.4 – Установка графической оболочки

Дожидаемся окончания процесса установки, и теперь можем установить workbench.

На рисунке 6.1.5 изображена установка workbench.

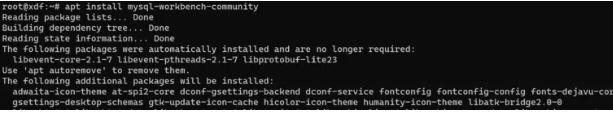


Рисунок 6.1.5 – Установка workbench

Теперь нам необходимо войти в GUI-интерфейс, для этого можно перезапустить ОС. После включения системы пройдем первоначальную настройку, настроем свою учетную запись, имя и другие важные моменты. После настройки MySql Workbench появиться на панели приложений.

После нажатия на иконку мы видим страницу приветствия с картой наших подключений, системы автоматически распознают сервер, но если вы хотите управлять удаленными БД, вы можете добавить их, поставив плюс рядом с подключением MySQL.

На рисунке 6.1.6 изображен интерфейс установленного MySQL Workbench.



Рисунок 6.1.6 – MySQL Workbench

* 1. **Настройка антивирусного программного обеспечения, Файрвола**

Настройка антивирусного программного обеспечения на Ubuntu Server необходима для защиты системы от вредоносных программ, таких как вирусы, черви, троянские кони и другие виды вредоносного кода. Хотя Linux-системы, в том числе Ubuntu, менее подвержены атакам вирусов по сравнению с другими операционными системами, такими как Windows, все же существует риск заражения, особенно когда речь идет о серверных системах.

В целом, ClamAV - это мощный инструмент для обнаружения и борьбы с вирусами на платформе Linux, который комбинирует в себе профессиональные возможности с открытостью и бесплатностью, делая его популярным выбором среди пользователей и администраторов Linux-систем.

Перед тем как установить антивирус, нам необходимо установить репозиторий EPEL. Так как нам нужно будет использовать высокие привилегии в дальнейшей работе, мы переключимся на пользователя «root» с помощью команды: «sudo –i», а затем обновляем индексы репозитория и пакеты.

На рисунке 6.2.1 изображена команда обновления пакетов.



Рисунок 6.2.1– Обновление пакетов

Теперь переходим к установке «clamav».

На рисунке 6.2.2 изображена установка «clamav».



Рисунок 6.2.2 – Установка «clamav»

После завершения установки убеждаемся что анивирус работает

На рисунке 6.2.3 изображена проверка работоспособности антивируса.

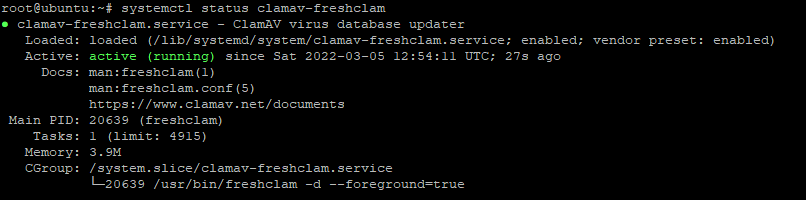


Рисунок 6.2.3 – Запущенный антивирус

После того как установка прошла успешно, перейдем к тому, как пользоваться антивирусом. Он запускается из командной строки. Вы можете проверить как отдельные файлы, так и папки. Для этого вводим команду: clamscan <имя документа/путь к папке>.

* 1. **Результаты тестирования**

Smoke-тестирование – это тип тестирования программного обеспечения, который проводится для проверки основных функций и функциональности приложения[4]. Цель такого тестирования заключается в том, чтобы убедиться, что программное обеспечение достаточно стабильно для дальнейшего тестирования.

Особенности Smoke-тестирования:

* Быстрое выполнение: Тесты разрабатываются и выполняются быстро, чтобы обеспечить быструю обратную связь;
* Точечная проверка: Проверяет конкретные области приложения, не охватывая все функциональные возможности;
* Регулярное выполнение: Обычно выполняется после внесения изменений в программное обеспечение, таких как исправление ошибок или добавление новых функций.

Предназначение Smoke-тестирование:

* Выявления критических дефектов: Определяет, может ли приложение выполнять основные функции без сбоев;
* Предотвращения дальнейшего тестирования неисправного кода: Гарантия, что программное обеспечение стабильно для дальнейшего тестирования;
* Определения приоритета дефектов: Помогает определить приоритетность ошибок на основе их влияния на основные функции;
* Подтверждения сборок: Проверка того, что собранное программное обеспечение достаточно стабильно для передачи в другие среды ;
* Мониторинга устойчивости: Регулярное выполнение Smoke-тестов обеспечивает постоянный контроль над стабильностью приложения.

Преимущества Smoke-тестирования:

* Снижает риски тестирования неустойчивого кода;
* Экономит время и ресурсы;
* Повышает уверенность в стабильности приложения перед развертыванием.

Тест кейс № 1 - Авторизация

|  |  |
| --- | --- |
| Идентификатор: | Тест-кейс №1 |
| Название: | Авторизация |
| Предусловие: | Имеется подключение к базе данных и знание логина и пароля |
| Описание: | Медицинский работник авторизуется с помощью логина и пароля |
| Шаги: | 1. Ввести логин и пароль; 2. Нажать на кнопку «Вход». |
| Ожидаемый результат: | Успешная авторизация и отображение следующей формы |
| Фактический результат: | Успешная авторизация и отображение следующей формы |
| Статус: | Пройден успешно |
| Постусловие: | Вывод текста об ошибке в консоль |

Тест кейс № 2 - Добавление данных через форму

|  |  |
| --- | --- |
| Идентификатор: | Тест-кейс №2 |
| Название: | Добавление данных через форму |
| Предусловие: | Подключение к базе, открыта форма с данным функционалом |
| Описание: | Медицинский работник добавляет новую запись в таблицу с которой он работает, посредством заполнения необходимых данных в отдельной форме добавления. |
| Шаги: | 1. Нажать на кнопку «Добавить»; 2. Заполнить поля формы в соответствии с полями базы данных; 3. Нажать на кнопку «Сохранить». |
| Ожидаемый результат: | Добавление записи с указанными в текстовых полях значениями |
| Фактический результат: | Добавление записи с указанными в текстовых полях значениями |
| Статус: | Пройден успешно |
| Постусловие: | Вывод текста об ошибке в консоль |

Тест кейс № 3 - Удаление записи из таблицы данных

|  |  |
| --- | --- |
| Идентификатор: | Тест-кейс №3 |
| Название: | Удаление записи из таблицы данных |
| Предусловие: | Подключение к базе, открыта форма с данным функционалом |
| Описание: | Медицинский работник удаляет запись в таблице с которой он работает. |
| Шаги: | 1. Выбрать необходимую запись; 2. Нажать кнопку «Удалить». |
| Ожидаемый результат: | Выбранная запись удалена из базы данных |
| Фактический результат: | Выбранная запись удалена из базы данных |
| Статус: | Пройден успешно |
| Постусловие: | Вывод текста об ошибке в консоль |

Тест кейс № 4 - Поиск

|  |  |
| --- | --- |
| Идентификатор: | Тест-кейс №4 |
| Название: | Поиск |
| Предусловие: | Подключение к базе, открыта форма с данным функционалом |
| Описание: | Медицинский работник вводит значения для поиска по определённому столбцу в таблице. |
| Шаги: | 1. Вести значения для поиска; |
| Ожидаемый результат: | Отображение только тех записей, которые соответствуют результату поиска |
| Фактический результат: | Отображение только тех записей, которые соответствуют результату поиска |
| Статус: | Пройден успешно |
| Постусловие: | Вывод текста об ошибке в консоль |

Тест кейс № 5 - Редактирование данных через форму

|  |  |
| --- | --- |
| Идентификатор: | Тест-кейс №5 |
| Название: | Редактирование данных через форму |
| Предусловие: | Подключение к базе, открыта форма с данным функционалом |
| Описание: | Медицинский работник редактирует запись в таблице с которой он работает,посредством заполнения необходимых данных в отдельной форме редактирования. |
| Шаги: | 1. Выбрать необходимую для изменения запись; 2. Нажать на кнопку «Изменить»; 3. Заполнить необходимые поля правильными данными вместо перенесённых данных; 4. Нажать на кнопку «Сохранить». |
| Ожидаемый результат: | Изменение данных записи |
| Фактический результат: | Изменение данных записи |
| Статус: | Пройден успешно |
| Постусловие: | Вывод текста об ошибке в консоль |

1. **АДМИНИСТРИРОВАНИЕ, ЗАЩИТА БАЗЫ ДАННЫХ**
   1. **Учетные записи сервера, базы данных**

В MySQL есть два основных типа пользователей: «root» и обычные пользователи «user». «Root» - это суперпользователь или администратор базы данных MySQL. Обычные пользователи «User» - это пользователи с ограниченными правами, которые используют базу данных для выполнения определенных задач, таких как выполнение запросов, чтение и запись данных[10].

Для MySQL были реализован простой скрипт на добавление пяти пользователей. В момент добавления пользователей при помощи скрипта, MySQL сам добавляет для них пароли, которые уже находятся в зашифрованном состоянии.

На рисунке 7.1.1 изображено добавление пользователей.



Рисунок 7.1.1 – Добавление пользователей при помощи скрипта

* 1. **Правила, роли, ограничения**

Привилегии, предоставленные учетной записи MySQL, определяют, какие операции может выполнять учетная запись. Привилегии MySQL различаются в зависимости от контекста, в котором они применяются, и на разных уровнях работы:

* Административные привилегии позволяют пользователям управлять работой сервера MySQL. Эти привилегии являются глобальными, поскольку они не относятся к конкретной базе данных;
* Привилегии базы данных применяются к базе данных и ко всем объектам в ней. Эти привилегии могут быть предоставлены для конкретных баз данных или глобально, чтобы они применялись ко всем базам данных;
* Права доступа к объектам базы данных, таким как таблицы, индексы, представления и [хранимые процедуры/функции](https://docs-python.ru/packages/klient-bd-mysql/khranimye-protsedury-funktsii-bd-mysql/" \o "MySQL: Хранимые процедуры и функции#.), могут быть предоставлены для определенных объектов в базе данных (например, для всех таблиц в базе данных).

Пользователь root:

* Роль: root - это суперпользователь или администратор базы данных MySQL;
* Права: root имеет неограниченные права на выполнение любых операций в базе данных, включая создание и удаление пользователей, управление правами доступа, создание, изменение и удаление баз данных, таблиц и объектов;
* Доступ: root имеет полный доступ ко всем базам данных на сервере MySQL;
* Имя пользователя: по умолчанию имя пользователя root является "root", но оно может быть изменено при установке или настройке сервера.

Обычные пользователи (user):

* Роль: обычные пользователи - это пользователи с ограниченными правами, которые используют базу данных для выполнения определенных задач, таких как выполнение запросов, чтение и запись данных;
* Права: права обычных пользователей ограничены и определяются при их создании. Администратор базы данных может назначить пользователям различные уровни доступа к базам данных, таблицам и другим объектам;
* Доступ: обычные пользователи могут иметь доступ к одной или нескольким базам данных, но их доступ ограничен определенными правами;
* Имя пользователя: имена обычных пользователей могут быть любыми, и их можно создавать и управлять ими с помощью root или другого администратора базы данных.

В целом, пользователь root имеет полный контроль над базой данных и может выполнять любые операции, в то время как обычные пользователи имеют ограниченный доступ и могут выполнять только те операции, которые им разрешены администратором базы данных.

MySQL предоставляет набор ограничений, которые вы можете включить в определения ваших таблиц для обеспечения целостности данных [11]. Ограничения позволяют лучше контролировать типы данных, которые могут быть добавлены в базу данных. Например, вы можете использовать ограничения, чтобы установить уникальность столбца или допустимость NULL-значений, указать значение по умолчанию для столбца или проверить, что данные попадают в определенный интервал допустимых значений.

MySQL поддерживает шесть основных типов ограничений для обеспечения целостности данных: PRIMARY KEY, NOT NULL, DEFAULT, CHECK, UNIQUE и FOREIGN KEY.

**ЗАКЛЮЧЕНИЕ**

В процессе проведения дипломной работы была проведено исследование предметной области, в результате которого были выбраны оптимальные средства разработки и системы управления базами данных. В результате реализации приложения на языке программирования C#, удалось создать функциональное решение, обеспечивающее добавление, удаление и редактирование данных в базе через интегрированную платформу. Благодаря использованию различных элементов управления, была реализована возможность осуществлять фильтрацию и поиск данных, что значительно упростило работу с информацией и повысило эффективность управления данными.

Таким образом, цель преддипломной практики, которая заключалась в создании приложения по учету анализов пациентов ГБУЗ РХ РКБ имени Г.Я. Ремишевской при помощи платформы C# достигнута.

Выполнены следующие задачи:

* Настройка и проектирование систем управления баз данных;
* Реализация клиентского приложения;
* Использование и реализация объектов баз данных;
* Освоение администрирования базы данных.

Выводы о проделанной работе могут быть следующими:

Автоматизация процессов в области оказания медицинской помощи пациентов способствует развитию уровня качества, скорости и точности оказанных услуг.

Возможные сферы внедрения программного средства включают в себя:

* Возможность формирования личного кабинета пациента на основе данных о пациенте;
* Упрощение коммуникации между персоналом и пациентом.

Перспективы развития программного средства могут включать в себя:

* Расширение функционала для включения дополнительных возможностей;
* Развитие веб-приложения для удобного доступа к функционалу;

Таким образом, приложение может значительно улучшить процесс оказания услуг в области медицины и облегчить взаимодействие между медицинским персоналом и пациентами, что делает его перспективным и востребованным программным средством в области медицины.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Eternalhost.net. Режим доступа [Электронный ресурс]: https://eternalhost.net/base/vps-vds/ustanovka-mysql-ubuntu (дата обращения 03.05.2024);
2. Habr.com. Режим доступа [Электронный ресурс]: https://habr.com/ru/articles/511798/ (дата обращения 23.04.2024);
3. Habr.com. Режим доступа [Электронный ресурс]: https://habr.com/ru/articles/505036/ (дата обращения 30.04.2024);
4. Ibs-qa.ru. Режим доступа [Электронный ресурс]: https://ibs-qa.ru/services/functional-testing/smoke-testing/ (дата обращения 04.05.2024);
5. Infostart Режим доступа [Электронный ресурс]: https://infostart.ru/pm/640962/ (дата обращения 06.05.2024);
6. Itproblog.ru. Режим доступа [Электронный ресурс]: https://itproblog.ru/установка-ubuntu-server-22-04-lts/ (дата обращения 25.04.2024);
7. Jetbrains.com. Режим доступа [Электронный ресурс]: https://www.jetbrains.com/rider/documentation/ (дата обращения 24.04.2024);
8. MySQL.com. Режим доступа [Электронный ресурс]: https://www.mysql.com/ (дата обращения 26.04.2024);
9. Resbol.mz19.ru. Режим доступа [Электронный ресурс]: https://resbol.mz19.ru/org/ (дата обращения 19.04.2024);
10. Resbol.mz19.ru. Режим доступа [Электронный ресурс]: https://resbol.mz19.ru/org/about/structure.php (дата обращения 19.04.2024);
11. SQL-ex Режим доступа [Электронный ресурс]: https://sql-ex.ru/blogs/?/Rabota\_s\_ogranicheniJami\_v\_MySQL.html (дата обращения 20.05.2024)
12. Timeweb Режим доступа [Электронный ресурс]: https://timeweb.com/ru/community/articles/kak-importirovat-i-eksportirovat-bazy-dannyh-v-mysql-i-mariadb-1 (дата обращения 20.05.2024)
13. Timeweb.cloud. Режим доступа [Электронный ресурс]: https://timeweb.cloud/tutorials/mysql/kak-sozdat-bazu-dannyh-i-novogo-polzovatelya-v-mysql (дата обращения 06.05.2024);
14. Wikipedia Режим доступа [Электронный ресурс]: https://ru.wikipedia.org/wiki/IDEF0 (дата обращения 06.05.2024);
15. Wikipedia Режим доступа [Электронный ресурс]: https://ru.wikipedia.org/wiki/Хранимая\_процедура (дата обращения 20.05.2024);

**ГЛОССАРИЙ**

Авалония (от англ: «Avalonia») – Реализация UI-фреймворка, основанная на XAML, которая позволяет разрабатывать кроссплатформенные приложения для Windows, Linux и macOS. Avalonia предоставляет мощные инструменты для построения сложных пользовательских интерфейсов и поддерживает теснуграцию с .NET Core.

Авторизация (от англ: «Authorization») – Процесс проверки личности пользователя перед предоставлением доступа к системе или определенным функциям приложения.

Администратор (от англ: «Admin») – Пользователь с расширенными правами, который может выполнять все функции приложения, включая добавление, редактирование, поиск, фильтрацию и удаление данных.

Данные (от англ: «Data») – Данные представляют собой информацию, которая обрабатывается или передается между процессами.

Клиентское приложение (от англ: «Client application») – Приложение, которое устанавливается и выполняется на устройстве пользователя, предоставляя ему доступ к серверным службам и данным через сеть.

Пользователь (от англ: «User») – Сущность, имеющая ограниченный доступ к функциям приложения. Пользователь может просматривать данные и выполнять поиск с фильтрацией, но не может вносить изменения.

Процессы (от англ: «Processes») – Они представляют операции или функции, которые преобразуют входные данные в выходные данные.

Роль (от англ: «Role») – Набор прав и функций, определяющих, какие действия может выполнять пользователь в системе.

Хранилища данных (от англ: «Data Stores») – Это места хранения данных внутри системы.

MySQL – Открытая система управления реляционными базами данных, которая использует язык SQL для управления данными.

SQL – Язык структурированных запросов, который используется для взаимодействия с реляционными базами данных.

XAML – Язык разметки, используемый для описания пользовательского интерфейса в приложениях .NET. XAML позволяет разработчикам создавать сложные пользовательские интерфейсы, используя декларативный синтаксис и привязку данных.

**СПИСОК АББРЕВИАТУР**

БД - База данных;

ИС - Информационная система;

ОС - Операционная система;

ПО - Программное обеспечение;

СУБД — Система управления базами данных.

DFD - Диаграмма потоков данных (Data Flow Diagram);

EPEL - Extra Packages for Enterprise Linux (дополнительные пакеты для Enterprise Linux);

GUI - Графический пользовательский интерфейс (Graphical User Interface);

LVM - Логический менеджер объёмов (Logical Volume Manager);

LTS - Долгосрочная поддержка (Long-Term Support);

MVVM - Модель-представление-виджет (Model-View-ViewModel);

RAID - Массивы независимо отказоустойчивых дисков (Redundant Array of Independent Disks);

UML - Унифицированный язык моделирования (Unified Modeling Language);

XAML - Язык разметки для приложений платформы Windows (EXtensible Application Markup Language).

**ПРИЛОЖЕНИЯ**

**Приложение А. Структуры БД**

В контексте дипломной работы была разработана база данных, которая включает в себя набор таблиц, каждая из которых отвечает за определённый аспект функционирования медицинского учреждения. Структура базы данных между таблицами:

Таблица «logins» содержит информацию о пользователях . В ней хранятся данные о их идентификаторе, логине, пароле.

Таблица «Должность» содержит информацию о должностях медицинского персонала. Она включает идентификатор, наименование должности.

Таблица «Кабинет» содержит информацию о кабинете. Она включает идентификатор ,телефон закреплённый за кабинетом, наименование кабинета.

Таблица «Персонал» содержит информацию о персонале в медицинском учреждении. Она включает идентификатор, имя, фамилия, отчество, пол, идентификатор кабинета, идентификатор должности.

Таблица «Пациент» содержит информацию о пациентах. Она включает идентификатор, имя, фамилия, отчество, пол, возраст, телефон и полис.

Таблица «Анализ» содержит информацию о проводимых анализах в медицинском учреждении. Она включает идентификатор и наименование анализа.

Таблица «Анализы» содержит информацию об назначенных анализах пациенту и их результаты. Она включает идентификатор,идентификатор анализа, идентификатор пациента,идентификатор персонала, дата проведения анализа, результат анализа.

Таблица «Расписание» содержит информацию о расписании проводимых анализах. Она включает идентификатор,идентификатор персонала, идентификатор анализа, время.

На рисунке 1 изображена схема БД.

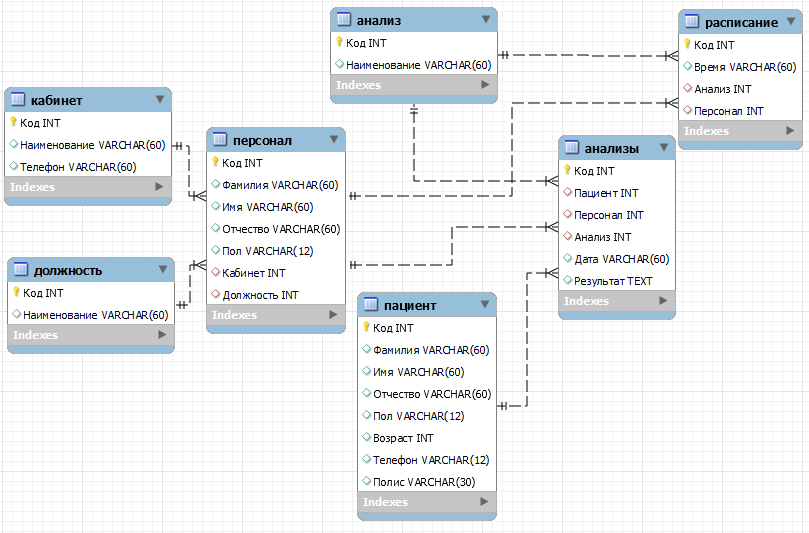


Рисунок 1 – Схема БД

**Приложение Б. Программный код**

В листинге 1 представлен код для отображения таблицы.

Листинг 1 - Код для отображения таблицы

public void FillCmb()  
{  
 patient = new List<Patient>();  
 conn = new MySqlConnection(connStr);  
 conn.Open();  
 MySqlCommand command = new MySqlCommand("SELECT \* FROM пациент", conn);  
 MySqlDataReader reader = command.ExecuteReader();  
 while (reader.Read() && reader.HasRows)  
 {  
 var currentPatient = new Patient()  
 {  
 Код = reader.GetInt32("Код"),  
 Фамилия = reader.GetString("Фамилия"),  
 Имя = reader.GetString("Имя"),  
 Отчество = reader.GetString("Отчество"),

Пол = reader.GetString("Пол"),  
 Возраст = reader.GetInt32("Возраст"),  
 Телефон = reader.GetString("Телефон"),  
 Полис = reader.GetString("Полис")  
 };  
 patient.Add(currentPatient);  
 }  
 conn.Close();  
 CmbNum.ItemsSource = patient;  
}

В листинге 2 представлен код для реализации метода удаления данных.

Листинг 2 - Код для реализации кнопки реализации метода удаления данных

Patient currentPatient = DataGrid.SelectedItem as Patient;  
if (currentPatient == null)  
{  
 return;  
}  
conn = new MySqlConnection(connStr);  
conn.Open();  
string sql = "DELETE FROM пациент WHERE Код = " + currentPatient.Код;  
MySqlCommand cmd = new MySqlCommand(sql, conn);  
cmd.ExecuteNonQuery();  
conn.Close();  
patient.Remove(currentPatient);  
ShowTable("SELECT \* FROM пациент;");

В листинге 3 представлен код для реализации метода добавления данных.

Листинг 3 - Код для реализации кнопки реализации метода добавления данных

var code = personals.FirstOrDefault(x => x.Код == CurrentPersonal.Код);  
if (code == null)  
{  
 try  
 {  
 conn = new MySqlConnection(connStr);  
 conn.Open();  
 string add = "INSERT INTO персонал VALUES (" + Convert.ToInt32(Код.Text)+ ", '" + Фамилия.Text + "', '" + Имя.Text + "', '" + Отчество.Text + "', '" + Пол.Text + "', " + Convert.ToInt32(Кабинет.Text ) + ", " + Convert.ToInt32(Должность.Text )+");";  
 MySqlCommand cmd = new MySqlCommand(add, conn);  
 cmd.ExecuteNonQuery();  
 conn.Close();  
 }  
 }

В листинге 4 представлен код для реализации метода редактирования данных.

Листинг 4 - Код для реализации кнопки реализации метода редактирования данных

conn = new MySqlConnection(connStr);  
conn.Open();  
string upd = "UPDATE персонал SET Фамилия = '" + Фамилия.Text + "', Имя = '" + Имя.Text + "', Отчество = '" + Отчество.Text + "', Пол = '" + Пол.Text + "', Кабинет = "+ Convert.ToInt32(Кабинет.Text) + ", Должность = "+ Convert.ToInt32(Должность.Text) + " WHERE Код = " + Convert.ToInt32(Код.Text) + ";";  
MySqlCommand cmd = new MySqlCommand(upd, conn);  
cmd.ExecuteNonQuery();  
conn.Close();

В листинге 5 представлен код для реализации поиска.

Листинг 5 - Код для реализации поиска

var search = patient;  
search = search.Where(x => x.Имя.Contains(Search1.Text)).ToList();  
DataGrid.ItemsSource = search;

В листинге 6 представлен код для реализации авторизации.

Листинг 6 - Код для реализации авторизации

conn = new MySqlConnection(connStr);  
conn.Open();  
string check ="SELECT \* FROM logins WHERE Login = '" + Login.Text + "' AND Password ='" + Password.Text + "' LIMIT 1";   
MySqlCommand cmd = new MySqlCommand(check, conn);  
cmd.ExecuteNonQuery();  
MySqlDataAdapter dataAdapter = new MySqlDataAdapter(cmd);  
DataTable dt = new DataTable();  
dataAdapter.Fill(dt);  
if (dt.Rows[0][0].ToString() == "1"||dt.Rows[0][0].ToString() == "2" )  
{  
 var menu = new Bolnitca.Menu();  
 this.Hide();  
 menu.Show();   
}

В листинге 7 представлен код интерфейса страницы.

Листинг 7 - Код интерфейса страницы

<Grid>  
 <StackPanel Orientation="Vertical">  
 <TextBlock Margin = "10 10 0 0" FontSize = "20" FontWeight="Bold"> Таблица "Медицинский персонал"</TextBlock>  
 <StackPanel Orientation="Horizontal">  
 <TextBlock VerticalAlignment="Center" Margin = "10 10 0 0"> Поиск по имени: </TextBlock>  
 <TextBox Name="Search1" TextChanged="Searchname" Width="200" Margin ="33 10 0 0"></TextBox>  
 <TextBlock VerticalAlignment="Center"  
 x:Name="FilterNum" Margin ="10 10 0 0">Фильтрация по коду: </TextBlock>  
 <ComboBox Name="CmbNum" Width="200" Margin ="10 10 0 0"   
 x:CompileBindings="False"  
 SelectionChanged="CmbNum\_OnSelectionChanged">  
 <ComboBox.ItemTemplate>  
 <DataTemplate>  
 <TextBlock Text="{Binding Код}"></TextBlock>  
 </DataTemplate>

Продолжение листинга 7  
 </ComboBox.ItemTemplate>  
 </ComboBox>  
 </StackPanel>   
 <StackPanel Orientation="Vertical">  
 <StackPanel Orientation="Horizontal">  
 <TextBlock VerticalAlignment="Center" Margin = "10 10 0 0"> Поиск по фамилии: </TextBlock>  
 <TextBox Name="Search2" TextChanged="Searchlastname" Width="200" Margin ="33 10 0 0"></TextBox>  
 <StackPanel Margin="10 12 00 0">  
 <Button Name="Add" Content="Добавить" Click="AddData"></Button>  
 </StackPanel>  
 <StackPanel Margin="10 12 00 0">  
 <Button Name="Update" Content="Изменить" Click="EditData"></Button>  
 </StackPanel>  
 <StackPanel Margin="10 12 00 0">  
 <Button Name="Delete" Content="Удалить" Click="DeleteData"></Button>  
 </StackPanel>  
 </StackPanel>  
 </StackPanel>   
 <DataGrid Name="DataGrid" x:CompileBindings="False" Margin ="10 20 0 0" Height="250">  
 <DataGrid.Columns>  
 <DataGridTextColumn Header="Код" Binding="{Binding Код}"></DataGridTextColumn>  
 <DataGridTextColumn Header="Фамилия" Binding="{Binding Фамилия}"></DataGridTextColumn>  
 <DataGridTextColumn Header="Имя" Binding="{Binding Имя}"></DataGridTextColumn>  
 <DataGridTextColumn Header="Отчество" Binding="{Binding Отчество}"></DataGridTextColumn>  
 <DataGridTextColumn Header="Пол" Binding="{Binding Пол}"></DataGridTextColumn>  
 <DataGridTextColumn Header="Кабинет" Binding="{Binding Кабинет}"></DataGridTextColumn>  
 <DataGridTextColumn Header="Должность" Binding="{Binding Должность}"></DataGridTextColumn>  
 </DataGrid.Columns>  
 </DataGrid>  
 <StackPanel Orientation="Horizontal" Margin="10 20 20 0">  
 <Button Name="Back" Content="Назад" Click="Back\_OnClick"></Button>  
 <StackPanel Margin="100 0 00 0">  
 <Button Name="Reset" Content="Сброс" Click="Reset\_OnClick"></Button>  
 </StackPanel>  
 </StackPanel>  
 </StackPanel>  
</Grid>

**Приложение В. Руководство оператора**

Авторизация:

* При первом запуске программы вы увидите окно авторизации;
* Вводим логин и пароль;
* Нажимаем кнопку «Войти».

Если введённые данные верны, программа перейдет в меню. В случае введения неправильного логина или пароля появится уведомление и потребуется ввести коректные значения.

Меню:

В меню программы расположены все таблицы базы данных, достаточно лишь выбрать необходимую таблицу нажатием опредленной функциональной кнопки.

Окно программы с необходимой таблицей состоит из следующих элементов:

* Таблица с данными из базы «MySQL»;
* Поле для ввода значений на поиск;
* Выпадающий список для фильтрации;
* Кнопки для добавления, редактирования, удаления.

Работа с данными:

Добавление данных:

* Нажимаем кнопку «Добавить» на панели инструментов;
* Заполняем поля в открывшемся окне;
* Нажимаем кнопку «Сохранить».

Если данные успешно добавлены, в поле сообщений появится соответствующее сообщение. В противном случае будет отображена ошибка.

Редактирование данных:

* Выделяем строку в таблице, которую хотим отредактировать;
* Нажимаем кнопку «Редактировать» на панели инструментов;
* Изменяем данные в открывшемся окне;
* Нажимаем кнопку «Сохранить».

Если данные успешно отредактированы, в поле сообщений появится соответствующее сообщение. В противном случае будет отображена ошибка.

Удаление данных:

* Выделяем строку в таблице, которую хотим удалить;
* Нажимаем кнопку «Удалить» на панели инструментов.

Если данные успешно удалены, в поле сообщений появится соответствующее сообщение. В противном случае будет отображена ошибка.

Поиск:

* Вводим нужное значение в поле для поиска.
* Получаем нужные значения

Важно чтобы введенное значение в поле для поиска совпадало со значением, для которого предназначено поле.

Для обновления окна на панели инструментов находится специальная кнопка «Сброс ». При ее нажатии окно возвращается к первоначальному состоянию.

Фильтрация:

* Выбираем необходимое значение из выпадающего списка;
* Получаем нужное значение.

Завершение работы:

* Нажмите кнопку «Выход» на панели инструментов.
* Приложение завершает свою работу

**Приложение Г. Анализ существующих аналогичных ПП**

На данный момент используются два варианта цифровых данных пациента, ЭМК и ЭМЗ.

**Электронные медицинские карты** являются цифровыми аналогами традиционных бумажных историй болезни. Они предназначены для обеспечения целостного представления истории болезни пациента и взаимодействия между различными поставщиками медицинских услуг и организациями здравоохранения.

ЭМК хранят широкий спектр данных о пациентах, включая личные демографические данные, историю болезни, текущие и прошлые диагнозы, лекарства, аллергии, записи о прививках и семейную медицинскую историю. Этот подробный профиль пациента помогает медицинским работникам принимать обоснованные решения.

Одним из основных преимуществ ЭМК является их доступность. Уполномоченные медицинские работники могут безопасно получать доступ к электронным медицинским картам из разных мест, включая больницы, клиники и даже удаленно, благодаря безопасным учетным данным и усовершенствованному шифрованию.

ЭМК спроектированы так, чтобы быть совместимыми, то есть они могут передавать данные и обмениваться ими с различными системами здравоохранения и поставщиками услуг. Такая совместимость обеспечивает беспрепятственный обмен данными между организациями здравоохранения, способствуя лучшей координации лечения и улучшению результатов лечения пациентов.

ЭМК часто содержат инструменты поддержки клинических решений, которые помогают медицинским работникам принимать обоснованные решения. Эти инструменты могут предупреждать о потенциальном взаимодействии лекарств, напоминать поставщикам медицинских услуг о рекомендуемых обследованиях и предлагать научно обоснованные рекомендации по лечению.

ЭМК могут хранить радиологические изображения, результаты лабораторных исследований и другие диагностические данные, что упрощает поставщикам медицинских услуг доступ и анализ этих важных фрагментов информации.

ЭМК могут включать в себя платежные и административные данные, что упрощает управление финансами здравоохранения и снижает административные накладные расходы для организаций здравоохранения.

На рисунке 2 изображено приложение с использованием электронной медицинской карты.

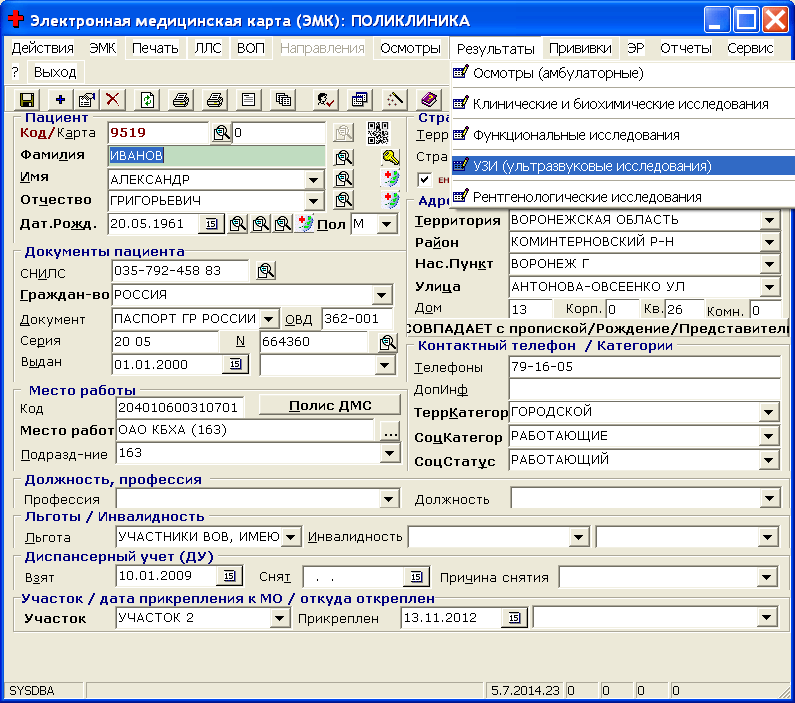


Рисунок 2 – Пример приложения с использованием электронной медицинской карты

**Электронные медицинские записи** имеют много общего с ЭМК, но обычно их объем ограничен конкретным медицинским учреждением или практикой.

ЭМЗ в основном используются в одном медицинском учреждении или практике, например в больнице, клинике или частной практике. Они предназначены для сбора и управления информацией о пациентах, полученной в конкретной организации.

ЭМЗ играют важную роль в обеспечении ориентированной на пациента помощи в пределах конкретного медицинского учреждения. Они позволяют медицинским работникам внутри организации получать доступ к актуальной информации о пациентах, оптимизировать рабочие процессы и обеспечивать непрерывность лечения.

ЭМЗ хранят важные данные о пациентах, включая истории болезни, диагнозы, методы лечения и результаты анализов. Они позволяют эффективно извлекать эту информацию во время приема пациентов, уменьшая необходимость в бумажных записях и ручном вводе данных.

Хотя ЭМЗ могут не иметь такого же уровня совместимости, как ЭМК, они все же могут интегрироваться с другими системами в том же медицинском учреждении, такими как лабораторные системы и программное обеспечение для выставления счетов.

На рисунке 3 изображен пример приложения с электронной медицинской записью.

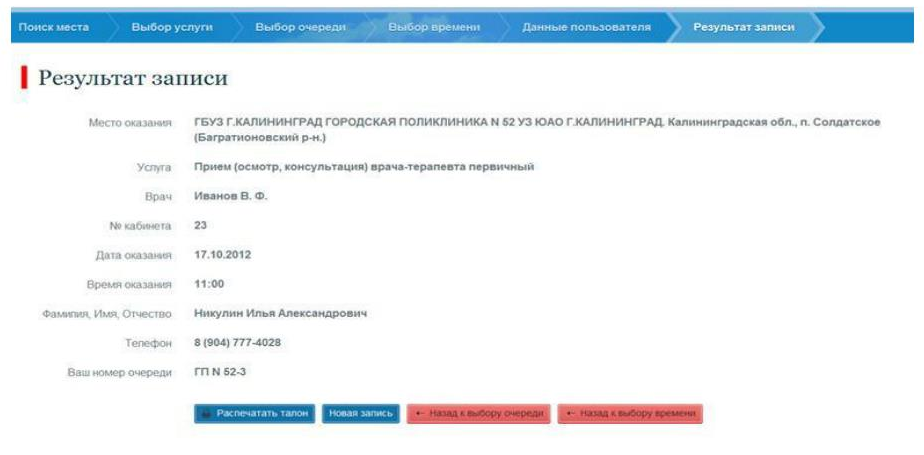


Рисунок 3 – Пример приложения с электронной медицинской записью

**1С Медицина** – программный продукт, который автоматизирует деятельность медицинских учреждений. Предназначен для организаций, которые оказывают медпомощь как в стационарных, так и в амбулаторно-поликлинических условиях. В частности, софт подходит для городских/районных/областных больниц, а также диспансеров разных специализаций.

Приложение помогает вести учет оказанной медпомощи и взаиморасчеты с контрагентами, управлять потоками пациентов. С его помощью можно автоматизировать деятельность регистратуры, приемного отделения, кассы. Также доступ к программе предоставляется врачебному и среднему медперсоналу, статистической службе.

Продукт учитывает все особенности бизнес-процессов подразделений и упрощает сотрудникам выполнение ежедневных рутинных задач.

1С Медицина создает единое информационное пространство медучреждения. Доступ к данным разделяется по ролевому принципу. Также предусмотрена возможность ведения учета по нескольким организациям в единой информационной базе.

В программе можно создать и вести несколько медкарт для одного пациента: стоматологическую, амбулаторную и др. Каждый медработник получает доступ к определенному типу карт. Дополнительно в приложении предусмотрены гибкие механизмы квотирования, с помощью которых можно устанавливать лимит на объемы оказываемой медпомощи.

Также данное решение позволяет:

* вести учет пациентов, которые поступили, были госпитализированы или выписаны;
* получать оперативные сводки о движении пациентов и количества свободных коек в отделении;
* вести учет и анализировать функционирование коечного фонда;
* вести учет оказанных врачами медицинских услуг;
* формировать управленческие и регламентированные отчеты.

Записывать пациентов на прием может как регистратура, так и врач. По данным предварительной записи составляется график работы кабинетов, нормы загрузки докторов, перечень выполняемых услуг.

На рисунке 4 изображен пример приложения 1С Медицина .

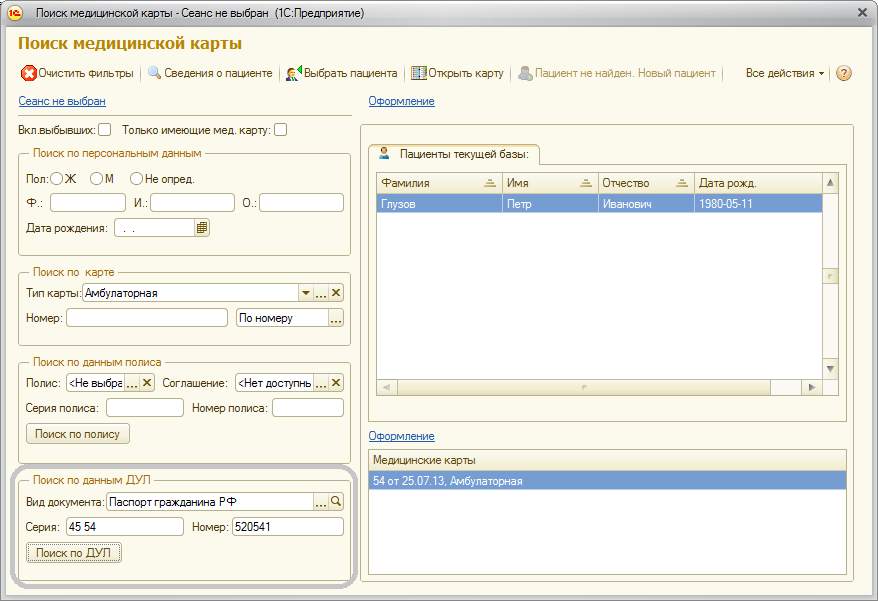


Рисунок 4 – Пример приложения 1С Медицина