СОДЕРЖАНИЕ

ВВЕДЕНИЕ 3

1 ЦЕЛИ И ЗАДАЧИ УЧЕБНОЙ ПРАКТИКИ 4

1.1 Основные требования по технике безопасности во время прохождения учебной практики 5

1.2 Нормативно-правовые документы 6

2 ПРЕДМЕТНАЯ ОБЛАСТЬ. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ. ПОСТАНОВКА ЗАДАЧИ 8

2.1 Описание предметной области 8

2.2 Группы пользователей информационной системы 9

2.3 Основные требования, предъявляемые к информационной системе 9

3 ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ «РЕМОНТ ПАССАЖИРСКИХ ВАГОНОВ» 11

3.1 Моделирование некоторых динамических аспектов системы 11

3.1.1 Создание диаграммы вариантов использования 12

3.1.2 Создание диаграммы деятельности 12

3.2 Разработка функциональной структуры информационной системы. Карта навигации по информационной системе 13

3.3 Разработка макета информационной системы «Ремонт пассажирских вагонов» 14

4 РАЗРАБОТКА ПРОТОТИПА ИНФОРМАЦИОННОЙ СИСТЕМЫ «РЕМОНТ ПАССАЖИРСКИХ ВАГОНОВ» 18

4.1 Моделирование и разработка базы данных 18

4.2 Разработка программных модулей информационной системы «Ремонт пассажирских вагонов» 22

5 РАЗРАБОТКА ТЕСТОВЫХ НАБОРОВ И ТЕСТОВЫХ СЦЕНАРИЕВ 45

5.1 Назначение эксперимента. Выбор и обоснование методики проведения тестирования 45

5.2 Технология тестирования 46

5.3 Результаты проведения тестирования 46

ЗАКЛЮЧЕНИЕ 47

БИБЛИОГРАФИЧЕСКИЙ СПИСОК 48

# ВВЕДЕНИЕ

Пассажирские вагоны играют ключевую роль в сфере железнодорожного транспорта, обеспечивая комфорт и безопасность путешествий. С течением времени эта отрасль претерпела значительное развитие, внедряя новые технологии и улучшая условия для пассажиров.

Начиная с классических вагонов с деревянными отделениями, современные пассажирские поезда предлагают разнообразные варианты, от купейных до открытых салонов. Инновации в дизайне и строительстве включают использование легких и прочных материалов, улучшенные системы вентиляции и климат контроля.

С развитием высокоскоростных поездов, появлением беспроводного интернета и интеллектуальных систем управления, пассажирские вагоны становятся не только местом транспортировки, но и платформой для передовых технологий. Интеграция умных решений, таких как системы безопасности и развлечений, сделала поездки более современными и удобными.

В целом, развитие пассажирских вагонов продолжает стремительно продвигаться вперед, объединяя комфорт, технологии и ответственность перед окружающей средой.

# ЦЕЛИ И ЗАДАЧИ УЧЕБНОЙ ПРАКТИКИ

Разработка программных модулей для системы ремонта пассажирских вагонов направлена на повышение эффективности, надежности и прозрачности процессов технического обслуживания. Целью является создание комплексной информационной системы, способствующей оптимизации работы и обеспечивающей высокий уровень безопасности и качества обслуживания пассажирского транспорта.

Задачи:

1. Автоматизация Учета Технического Состояния: Разработка модулей для автоматизации мониторинга и учета технического состояния пассажирских вагонов. Это включает в себя системы диагностики, сбора и анализа данных с датчиков.
2. Планирование Регулярных Технических Обслуживаний: Создание модулей для планирования регулярных технических обслуживаний в соответствии с нормативами и требованиями безопасности, что способствует предотвращению аварий и повышению долговечности оборудования.
3. Управление Ремонтными Работами: Реализация функционала по управлению ремонтными работами, включая планирование, выделение ресурсов, контроль сроков и мониторинг качества выполненных работ.
4. Интеграция с Существующими Системами: Обеспечение совместимости с другими информационными системами, используемыми в железнодорожной отрасли, для обмена данными и обеспечения единого информационного пространства.
5. Обеспечение Безопасности Данных: Разработка модулей для защиты конфиденциальности данных, связанных с техническим состоянием вагонов и ремонтными работами, чтобы предотвратить несанкционированный доступ.
6. Обучение и Поддержка Пользователей: Предоставление обучения и системы поддержки для пользователей, обеспечивая эффективное использование программных модулей.

Цель и задачи в совокупности направлены на повышение эффективности обслуживания и обеспечение безопасности и надежности пассажирского транспорта через современные информационные технологии.

Для достижения цели и задач будет разработана и спроектирована информационная система “Ремонт пассажирских вагонов” нацеленная на оптимизацию процесса внесения и хранения данных в этой предметной области, в рамках учебной практики по модулю ПМ 01 (Разработка модулей программного обеспечения для компьютерных систем).

## Основные требования по технике безопасности во время прохождения учебной практики

К работе в компьютерном классе допускаются лица, ознакомленные с данной инструкцией по технике безопасности и правилам поведения. Необходимо неукоснительно соблюдать правила по технике безопасности, т.к. нарушение этих правил может привести к поражению электрическим током, вызвать возгорание и навредить вашему здоровью.

* Работа обучающихся в компьютерном классе разрешается только в присутствии преподавателя (лаборанта).
* Во время занятий посторонние лица могут находиться в классе только с разрешения преподавателя.
* Во время перемен между уроками проводится обязательное проветривание компьютерного кабинета с обязательным выходом обучающихся из класса.

Каждый обучающийся в ответе за состояние своего рабочего места и сохранность размещенного на нем оборудования.

Обучающийся, допустивший нарушение по технике безопасности и правилам поведения в компьютерном классе, может быть привлечен к дисциплинарной ответственности. Если нарушение техники безопасности связано с причинением имущественного ущерба, обучающийся несет и материальную ответственность в установленном законом порядке.

## Нормативно-правовые документы

Оформление и содержание технического задания должно соответствовать требованиям стандарта «ГОСТ 19.201-78. ЕСПД. Общие положения.

Техническое задание оформляют в соответствии с ГОСТ 19.106-78 на листах формата 11 и 12 по ГОСТ 2.301-68, как правило, без заполнения полей листа. Номера листов (страниц) проставляются в верхней части листа над текстом.

Лист утверждения и титульный лист оформляют в соответствии с ГОСТ 19.104-78.

Информационную часть (аннотацию и содержание), лист регистрации изменений допускается в документ не включать.

Для внесения изменений или дополнений в техническое задание на последующих стадиях разработки про1раммы или программного изделия выпускают дополнение к нему.

Согласование и утверждение дополнения к техническому заданию проводят в том же порядке, который установлен для технического задания.

Техническое задание должно содержать следующие разделы:

* введение;
* основания для разработки;
* назначение разработки;
* требования к программе или программному изделию;
* требования к программной документации;
* технико-экономические показатели;
* стадии и этапы разработки;
* порядок контроля и приемки;
* в техническое задание допускается включать приложения.

В зависимости от особенностей программы или программного изделия допускается уточнять содержание разделов, вводить новые разделы или объединять отдельные из них.

Вывод по главе: были описаны, техника безопасности во время прохождения учебной практики; нормативно-правовые документы, которые отвечают за стандарты разработки; цели и задачи учебной практики, включая вводную часть, где описана тематика отрасли и её тенденции развития.

# ПРЕДМЕТНАЯ ОБЛАСТЬ. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ. ПОСТАНОВКА ЗАДАЧИ

Предметная область информационной системы — это материальная система или система, характеризующая элементы материального мира, информация о которой хранится и обрабатывается. Предметная область рассматривается как некоторая совокупность реальных объектов и связей между ними

## Описание предметной области

Необходимо разработать и спроектировать информационную систему для эффективного управления процессами технического обслуживания и ремонта пассажирских вагонов, направленной на повышение безопасности, надежности и оптимизации затрат.

Депо выполняет несколько видов ремонта. Деповской ремонт – после пробега вагоном 450 тыс. км. или два года эксплуатации (что наступит раньше). TO-2 – подготовка вагона к зимним или летним условиям эксплуатации. ТО-3 – текущее обслуживание – после пробега 150 тыс. км. или один год эксплуатации. Текущий ремонт – круглосуточно, при котором ремонтируются вагоны всех дорог России. Основные причины поступления вагона в текущий ремонт: неисправность колесной пары, неисправность буксового узла и т.д. Каждый вагон имеет уникальный номер. Тип вагона также имеет значение при ремонте: купейный, СВ, плацкартный, почтовый, багажный. Каждый вагон приписан к дирекции по обслуживанию пассажиров (ДОП-1, ДОП-2, ДОП-3 и т.д.). Текущий ремонт выполняют ремонтные бригады в четыре смены. Для выполнения остальных ремонтов привлекается, как правило, одна бригада. За высокое качество ремонта члены бригады получают премию.

Информационная система "Ремонт Пассажирских Вагонов" призвана содействовать повышению эффективности и безопасности в процессах обслуживания, а также уменьшению затрат за счет оптимизации ресурсов и раннего выявления потенциальных проблем.

## Группы пользователей информационной системы

Сотрудники, в зависимости от своих функциональных обязанностей, могут выполнять действия, направленные на мониторинг, управление, анализ данных и обеспечение безопасности в рамках проектируемой информационной системы "Ремонт Пассажирских Вагонов".

Роли и их Обязанности:

* + Гость: просмотр данных.
  + Администратор: управление системами аутентификации, авторизации и шифрования; оптимизация работы системы для обеспечения высокой производительности; полный доступ ко всей системе.
  + Сотрудник: получение информации о текущем состоянии вагонов, диагностических данных и рекомендациях по обслуживанию; участие в разработке графика технического обслуживания и ремонтных работ.
  + Бригадир: получение информации о текущем состоянии вагонов, диагностических данных и рекомендациях по обслуживанию; участие в разработке графика технического обслуживания и ремонтных работ.

## Основные требования, предъявляемые к информационной системе

* + Гибкость: система должна обладать высокой степенью редактирования, чтобы пользователи могли адаптировать её под различные потребности в рамках своих обязанностей; возможность легкого масштабирования системы при увеличении объема данных или числа пользователей.
  + Надежность: обеспечение бесперебойной работы системы с минимальным количеством сбоев и перерывов; регулярное создание резервных копий данных для предотвращения потери информации в случае сбоев.
  + Эффективность: обеспечение высокой производительности при обработке данных, особенно в условиях одновременного использования множества пользователей; эффективное использование вычислительных ресурсов, чтобы система функционировала быстро и эффективно.
  + Безопасность: реализация надежной системы аутентификации для проверки личности пользователей и установление строгих прав доступа; и использование шифрования для защиты передаваемой и хранимой чувствительной информации.

Вывод по главе: было описано, что такое предметная область и её описание; группы пользователей, которые будут пользоваться информационной системой и основные требования к ней.

# ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ «РЕМОНТ ПАССАЖИРСКИХ ВАГОНОВ»

Проектированием информационных систем называется многоступенчатый процесс их создания и/или модернизации путём применения упорядоченной совокупности методологий и инструментария. Проектирование (в отличие от моделирования) предполагает работу с пока несуществующим объектом и направлено на создание информационной системы в области:

* обработки объектов будущей базы данных;
* написания программ (в том числе – отчётных и экранных форм), обеспечивающих выполнение запросов к данным,
* выполнения учёта функционирования конкретной среды (технологии).

Если выделять стадию проектирования информационных систем в качестве отдельного этапа, то его можно разместить между этапами анализа и разработки. Однако на практике чёткое разделение на этапы, как правило, затруднено или невозможно, поскольку проектирование, формально начинаясь с определения цели проекта, часто продолжается на стадиях тестирования и реализации.

## Моделирование некоторых динамических аспектов системы

Диаграммы деятельности применяются для моделирования динамических аспектов поведения системы. Эти динамические аспекты могут включать деятельность на любом уровне абстракции в любом представлении системной архитектуры, включая классы (в том числе активные), интерфейсы, компоненты и узлы.

Использовать диаграммы деятельности для моделирования некоторых динамических аспектов системы вы можете в контексте почти любого моделируемого элемента. Однако чаще вы будете прибегать к таким диаграммам в контексте всей системы, подсистемы, операции или класса. Диаграмму деятельности можно присоединить к варианту использования, чтобы моделировать сценарий, и к кооперации, чтобы моделировать динамические аспекты поведения совокупности объектов.

При разработке программного обеспечения заблаговременное планирование и моделирование значительно упрощают программирование. На этапе создания концептуальной модели для описания функционального назначения используется диаграмма вариантов использования, а для описания логики процедур и бизнес-процессов используется диаграмма деятельности.

### Создание диаграммы вариантов использования

В приложении 1, на рисунке 1 представлен пример диаграммы вариантов использования информационной системы для ремонта пассажирских вагонов.

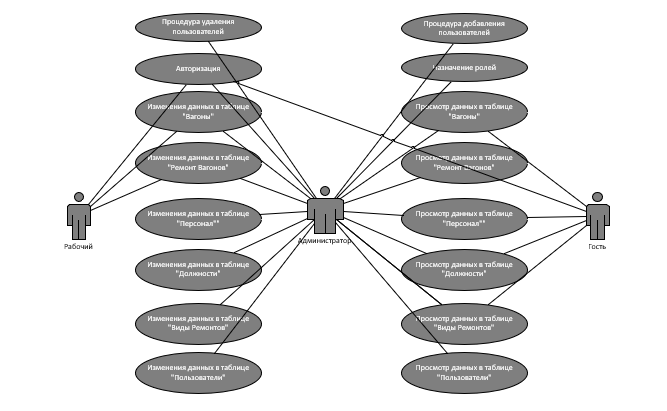


Рисунок 1 - Диаграмма вариантов использования

### Создание диаграммы деятельности

В приложении 2, на рисунке 2 представлена диаграмма деятельности информационной системы «Ремонт пассажирских вагонов».



Рисунок 2 – Диаграмма деятельности

## Разработка функциональной структуры информационной системы. Карта навигации по информационной системе

Карта навигации по информационной системе представлена в приложении 3 на рисунке 3 представлена карта переходов по информационной системе «Ремонт пассажирских вагонов»

Рисунок 3 – Карта переходов

## Разработка макета информационной системы «Ремонт пассажирских вагонов»

В приложении 4 на рисунке 4 представлено окно авторизации.

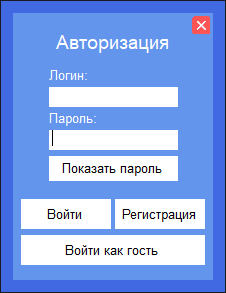


Рисунок 4 – Форма “Авторизация”

На этапе авторизации пользователь может ввести логин и пароль, чтобы продолжить работу в информационной системе.

В случае если пользователь еще не зарегистрирован в системе, он может сделать это, нажав на кнопку «Зарегистрироваться». В этом случае откроется окно регистрации, представленное в приложении 5 на рисунке 5.

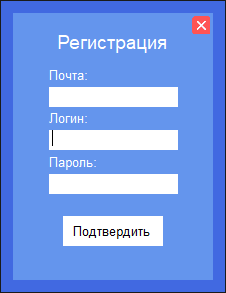


Рисунок 5 – Форма “Регистрация”

На этапе регистрации пользователю необходимо придумать свой логин и пароль.

Если сотрудник с введенными данными не зарегистрирован в базе данных, в качестве сотрудника, то пользователь не сможет зарегистрироваться, и, соответственно, продолжить работу, так как данная информационная система предназначена только для сотрудников и бригадиров ремонта пассажирских вагонов.

После того, как пользователь успешно прошел авторизацию, он может продолжить работу с таблицами на главной форме. В приложении 5 на рисунке 6 - 9 представлена главная форма, которая будет отображена, в зависимости от прав пользователя в системе.

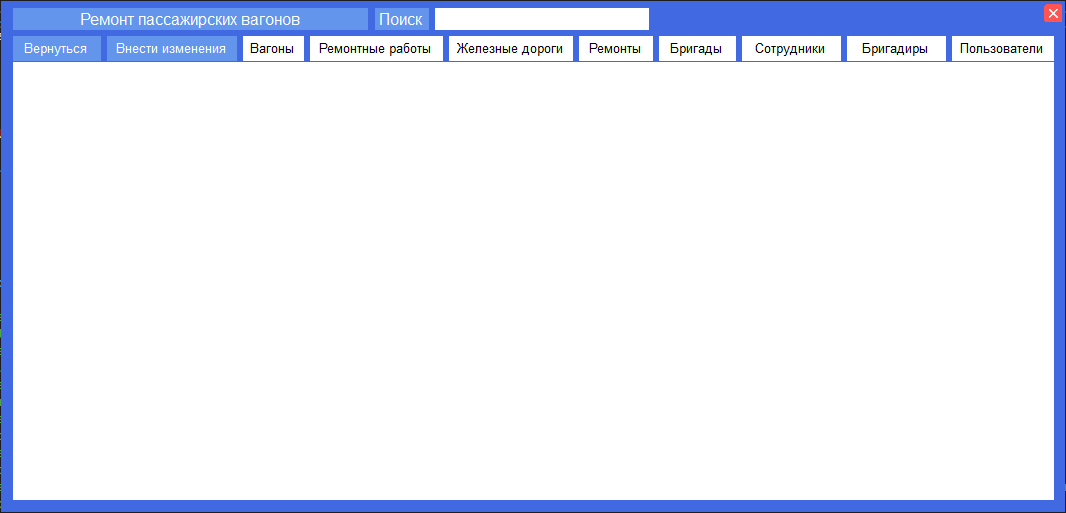


Рисунок 6 – Форма Администратора

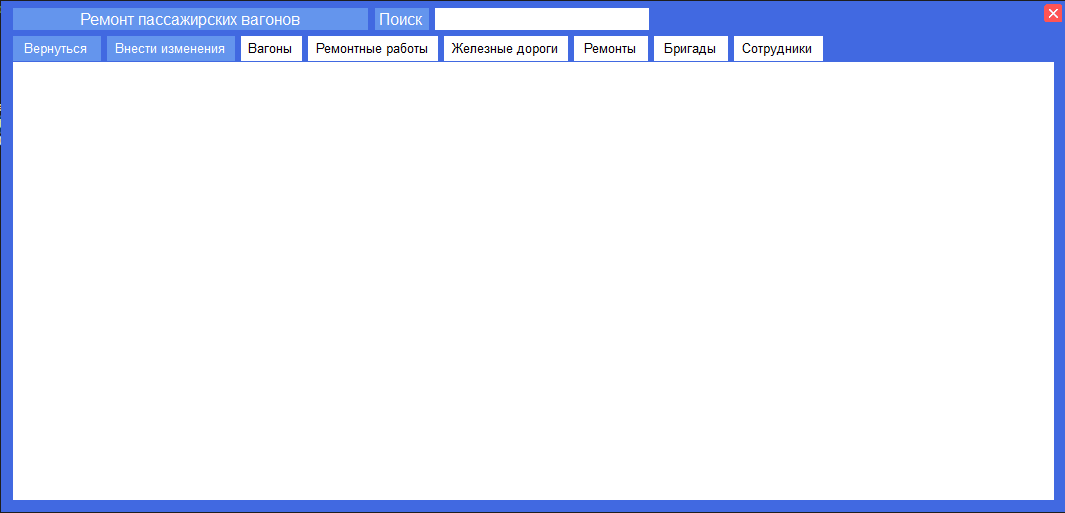


Рисунок 7 – Форма Бригадира

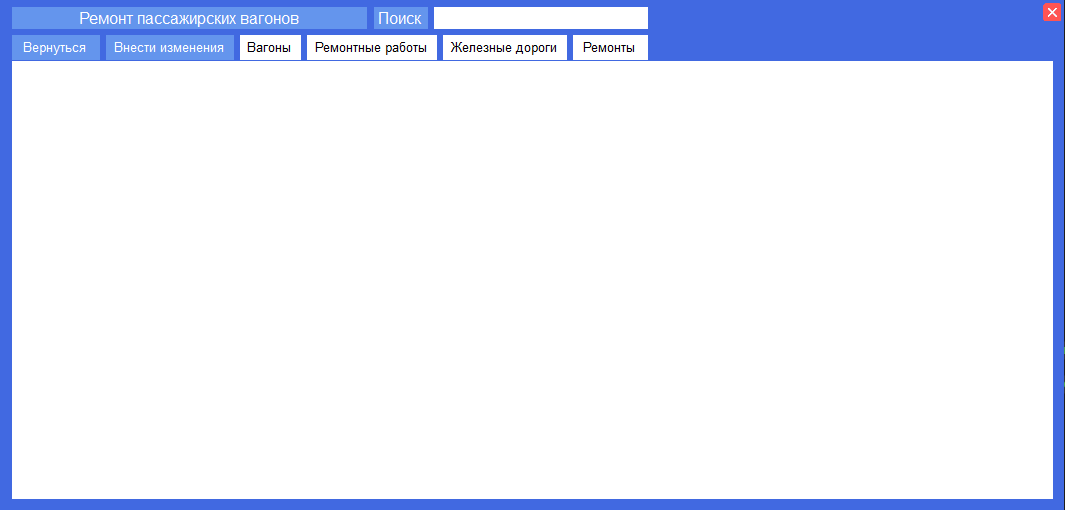


Рисунок 8 – Форма Сотрудника

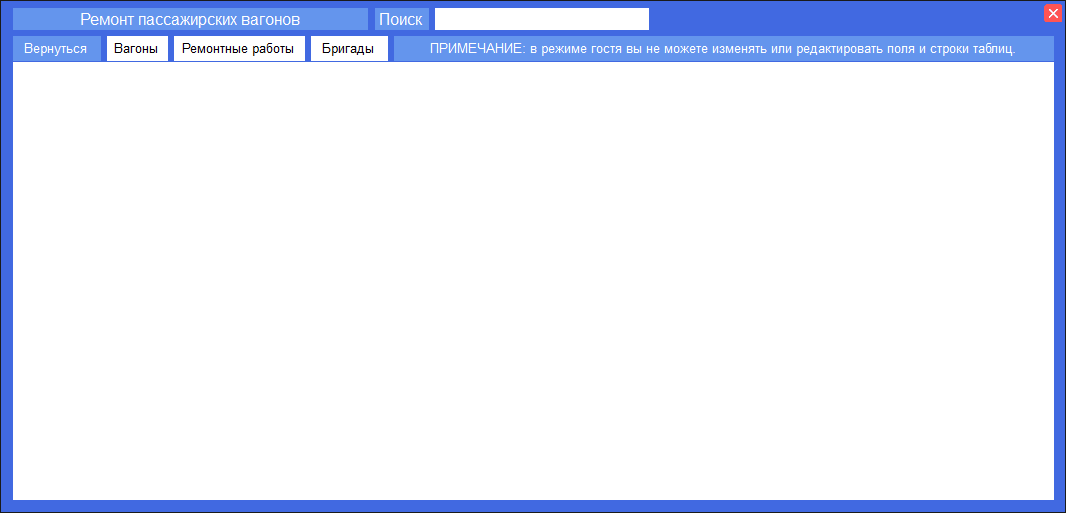


Рисунок 9 – Форма Гостя

Аналогичным образом в зависимости от роли пользователя выглядит главная форма, на которой расположены все необходимые таблицы. Например, для гостя будут скрыты кнопки редактирования, добавления и удаления записи, так как данной группе пользователей доступен только просмотр данных.

Дизайн выполнен в минималистичном стиле для того, чтобы пользователь с любым уровнем владения компьютером мог без препятствий работать с программным обеспечением.

Вывод по главе: была спроектирована диаграмма вариантов использования, диаграмма деятельности и макеты будущей информационной системы.

# РАЗРАБОТКА ПРОТОТИПА ИНФОРМАЦИОННОЙ СИСТЕМЫ «РЕМОНТ ПАССАЖИРСКИХ ВАГОНОВ»

Прототипирование — это один из этапов разработки, который заключается в продумывании содержания и расположения важных элементов интерфейса. Прототип — это моделирование конечного продукта. Это интерактивный макет, который может иметь любую степень точности. Основная цель создания прототипов – проверить, насколько последователен путь пользователя, и выявить препятствия, которые могут возникнуть в процессе его взаимодействия с продуктом.

Прототипы не только позволяют проверить удобство разрабатываемого продукта до начала написания кода, они также приводят к неожиданным открытиям и новым идеям, которые могут вывести проектируемый продукт на новый уровень.

## Моделирование и разработка базы данных

Исходя из анализа предметной области, можно выделить восемь сущностей: «Железные Дороги», «Вагоны», «Ремонты», «Ремонтные Работы», «Бригады», «Бригадиры», «Сотрудники», «Пользователи».

В приложении 6 на рисунке 10 представлена логическая модель данных базы данных «Ремонт пассажирских вагонов».

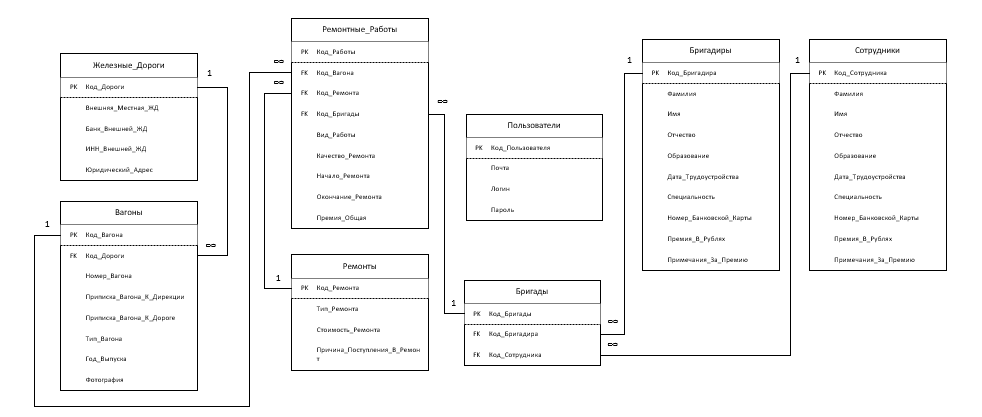


Рисунок 10 – Логическая модель данных

Создание базы данных происходит в MS SQL, в примере представлено создание базы данных в MS SQL Server.

С помощью раздела «Создание» и конструктора таблиц MS SQL созданы все таблицы и поля, представленные на логической модели данных.

Названия полей и их типы данных представлены в таблицах 1-8.

Таблица 1 – Поля таблицы «Железные Дороги»

|  |  |
| --- | --- |
| Название поля | Тип данных |
| Код Дороги | Числовой |
| Внешняя Местная ЖД | Логический |
| Банк Внешней ЖД | Текстовый |
| ИНН Внешней ЖД | Числовой |
| Юридический Адрес | Текстовый |

Таблица 2 – Поля таблицы «Вагоны»

|  |  |
| --- | --- |
| Название поля | Тип данных |
| Код Вагона | Числовой |
| Код Дороги | Числовой |
| Номер Вагона | Числовой |
| Приписка Вагона К Дирекции | Текстовый |
| Приписка Вагона К Дороге | Текстовый |
| Тип Вагона | Текстовый |
| Год Выпуска | Дата |
| Фотография | OLE |

Таблица 3 – Поля таблицы «Ремонты»

|  |  |
| --- | --- |
| Название поля | Тип данных |
| Код Ремонта | Числовой |
| Тип Ремонта | Текстовый |
| Стоимость Ремонта | Денежный |
| Причина Поступления В Ремонт | Текстовый |

Таблица 4 – Поля таблицы «Ремонтные Работы»

|  |  |
| --- | --- |
| Название поля | Тип данных |
| Код Работы | Числовой |
| Код Вагона | Числовой |
| Код Ремонта | Числовой |
| Код Бригады | Числовой |
| Вид Работы | Текстовый |
| Качество Ремонта | Логический |
| Начало Ремонта | Дата |
| Окончание Ремонта | Дата |
| Премия Общая | Числовой |

Таблица 5 – Поля таблицы «Бригады»

|  |  |
| --- | --- |
| Название поля | Тип данных |
| Код Бригады | Числовой |
| Код Бригадира | Числовой |
| Код Сотрудника | Числовой |

Таблица 6 – Поля таблицы «Бригадиры»

|  |  |
| --- | --- |
| Название поля | Тип данных |
| Код Бригадира | Числовой |
| Фамилия | Текстовый |
| Имя | Текстовый |
| Отчество | Текстовый |
| Образование | Текстовый |
| Дата Трудоустройства | Дата |
| Специальность | Текстовый |
| Номер Банковской Карты | Текстовый |
| Премия В Рублях | Денежный |
| Примечания. За Премию | Memo |

Таблица 7 – Поля таблицы «Сотрудники»

|  |  |
| --- | --- |
| Название поля | Тип данных |
| Код Сотрудника | Числовой |
| Фамилия | Текстовый |
| Имя | Текстовый |
| Отчество | Текстовый |
| Образование | Текстовый |
| Дата Трудоустройства | Дата |
| Специальность | Текстовый |
| Номер Банковской Карты | Текстовый |
| Премия В Рублях | Денежный |
| Примечания. За Премию | Memo |

Таблица 8 – Поля таблицы «Пользователи»

|  |  |
| --- | --- |
| Название поля | Тип данных |
| Код Пользователя | Числовой |
| Почта | Текстовый |
| Логин | Текстовый |
| Пароль | Числовой |

В приложении 6 на рисунке 11 представлены все созданные таблицы в обозревателе объектов базы данных.

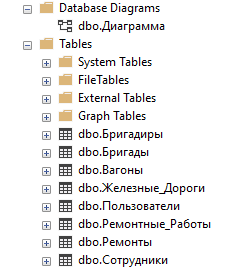


Рисунок 11 – Созданные таблицы

После того, как таблицы созданы, необходимо создать между ними связи.

В приложении 6 На рисунке 12 представлена физическая модель данных, отражающая все таблицы и их поля, а также связи между таблицами.

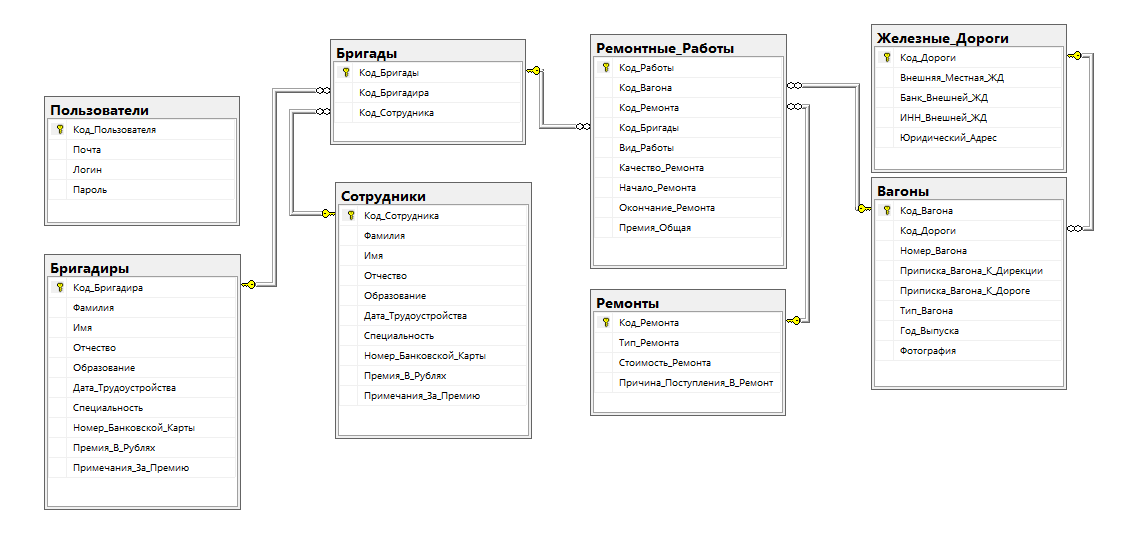


Рисунок 12 – Физическая модель базы данных

## Разработка программных модулей информационной системы «Ремонт пассажирских вагонов»

Разработанные в MS Visual Studio формы продемонстрированы в приложении 7 на рисунках 13-17.

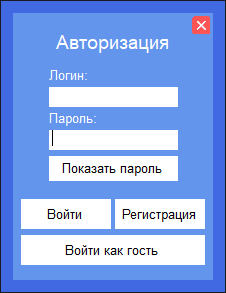


Рисунок 13 – Форма “Авторизация”

Свойства элементов формы Авторизация представлены в приложении 7 в таблице 9.

Таблица 9 – Свойства элементов формы Авторизация

|  |  |  |
| --- | --- | --- |
| Объект | Свойство | Значение |
| Авторизация | FormBorderStyle | None |
| Text | Ремонт пассажирских вагонов |
| StartPosition | CenterScreen |
| Size | 224; 291 |
| ShowIcon | False |

Событийно-управляемые процедуры в форме Авторизация представлены в приложении 7 таблице 10

Таблица 10 – Событийно-управляемые процедуры в форме «Авторизация»

|  |  |
| --- | --- |
| Объект | Событийно-управляемые процедуры |
| class Авторизация | public partial class Авторизация : Form  {  private Point mouseOffset;  private bool isMouseDown = false;  public Авторизация()  {  InitializeComponent();  }  } |
| textBox1\_KeyPress | private void textBox1\_KeyPress(object sender, KeyPressEventArgs e)  {  textBox1.MaxLength = 30;  if (char.IsDigit(e.KeyChar))  {  e.Handled = true;  MessageBox.Show("Поле не может содержать цифры!", "Ошибка");  }  } |
| textBox2\_KeyPress | private void textBox2\_KeyPress(object sender, KeyPressEventArgs e)  {  textBox2.MaxLength = 15;  if (!char.IsDigit(e.KeyChar) && (e.KeyChar != (char)Keys.Back))  {  e.Handled = true;  MessageBox.Show("Поле не может содержать буквы!", "Ошибка");  }  } |
| button1 | private void button1\_Click(object sender, EventArgs e)  {  if ((textBox1.Text == "") || (textBox2.Text == ""))  {  MessageBox.Show("Поля не должны быть пустыми.", "Ошибка");  }  else  {  SqlConnection sqlConnect = new SqlConnection("Data Source = sql; Initial Catalog = уП01\_ИСПП5\_Макуров\_ВА; Integrated Security = True");  sqlConnect.Open();  SqlDataAdapter da = new SqlDataAdapter("select \* from Пользователи", sqlConnect);  DataTable dt = new DataTable();  da.Fill(dt);  int flag = -1;  for (int i = 0; i < dt.Rows.Count; i++)  {  if ((dt.Rows[i]["Права\_Доступа"].ToString() == "Админ") && (dt.Rows[i]["Логин"].ToString() == textBox1.Text) && (dt.Rows[i]["Пароль"].ToString() == textBox2.Text))  flag = 0;  if ((dt.Rows[i]["Права\_Доступа"].ToString() == "Сотрудник") && (dt.Rows[i]["Логин"].ToString() == textBox1.Text) && (dt.Rows[i]["Пароль"].ToString() == textBox2.Text))  {  flag = 1;  }  if ((dt.Rows[i]["Права\_Доступа"].ToString() == "Бригадир") && (dt.Rows[i]["Логин"].ToString() == textBox1.Text) && (dt.Rows[i]["Пароль"].ToString() == textBox2.Text))  {  flag = 2;  }  }  if ((flag == 0) || (flag == 1) || (flag == 2))  {  if (flag == 0)  {  Hide();  Администратор UserForm1 = new Администратор();  UserForm1.Show();  }  if (flag == 1)  {  Hide();  Сотрудник UserForm2 = new Сотрудник();  UserForm2.Show();  }  if (flag == 2)  {  Hide();  Бригадир UserForm3 = new Бригадир();  UserForm3.Show();  }  }  else  MessageBox.Show("Пересоздайте пользователя! Такого пользователя не существует.", "Ошибка");  }  } |
| button2 | private void button2\_Click(object sender, EventArgs e)  {  Регистрация UserForm4 = new Регистрация();  UserForm4.Show();  } |
| button3 | private void button3\_Click(object sender, EventArgs e)  {  Hide();  Гость UserForm = new Гость();  UserForm.Show();  } |
| button4 | private void button4\_Click(object sender, EventArgs e)  {  if (button4.Text == "Показать пароль")  {  textBox2.PasswordChar = '\0';  button4.Text = "Скрыть пароль";  }  else if (button4.Text == "Скрыть пароль")  {  textBox2.PasswordChar = '\*';  button4.Text = "Показать пароль";  }  } |
| Авторизация\_MouseDown | private void Авторизация\_MouseDown(object sender, MouseEventArgs e)  {  int xOffset;  int yOffset;  if (e.Button == MouseButtons.Left)  {  xOffset = -e.X - SystemInformation.FrameBorderSize.Width;  yOffset = -e.Y - SystemInformation.CaptionHeight -  SystemInformation.FrameBorderSize.Height;  mouseOffset = new Point(xOffset, yOffset);  isMouseDown = true;  }  } |
| Авторизация\_MouseMove | private void Авторизация\_MouseMove(object sender, MouseEventArgs e)  {  if (isMouseDown)  {  Point mousePos = Control.MousePosition;  mousePos.Offset(mouseOffset.X, mouseOffset.Y);  Location = mousePos;  }  } |
| Авторизация\_MouseUp | private void Авторизация\_MouseUp(object sender, MouseEventArgs e)  {  if (e.Button == MouseButtons.Left)  {  isMouseDown = false;  }  } |
| pictureBox1 | private void pictureBox1\_Click(object sender, EventArgs e)  {  Application.Exit();  } |

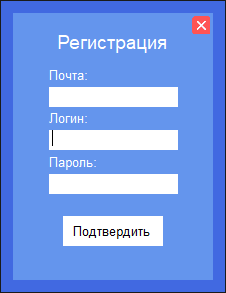


Рисунок 14 – Форма “Регистрация”

Свойства элементов формы Регистрация представлены в приложении 7 в таблице 11.

Таблица 11 – Свойства элементов формы Регистрация

|  |  |  |
| --- | --- | --- |
| Объект | Свойство | Значение |
| Регистрация | FormBorderStyle | None |
| Text | Ремонт пассажирских вагонов |
| StartPosition | CenterScreen |
| Size | 224; 291 |
| ShowIcon | False |

Событийно-управляемые процедуры в форме Регистрация представлены в приложении 7 таблице 12

Таблица 12 – Событийно-управляемые процедуры в форме «Регистрация»

|  |  |
| --- | --- |
| Объект | Событийно-управляемые процедуры |
| class Регистрация | public partial class Регистрация : Form  {  private Point mouseOffset;  private bool isMouseDown = false;  public Регистрация()  {  InitializeComponent();  }  } |
| textBox1\_KeyPress | private void textBox1\_KeyPress(object sender, KeyPressEventArgs e)  {  textBox1.MaxLength = 30;  if (char.IsDigit(e.KeyChar))  {  e.Handled = true;  MessageBox.Show("Поле не может содержать цифры!", "Ошибка");  }  } |
| textBox2\_KeyPress | private void textBox2\_KeyPress(object sender, KeyPressEventArgs e)  {  textBox2.MaxLength = 30;  if (char.IsDigit(e.KeyChar))  {  e.Handled = true;  MessageBox.Show("Поле не может содержать цифры!", "Ошибка");  }  } |
| textBox3\_KeyPress | private void textBox3\_KeyPress(object sender, KeyPressEventArgs e)  {  textBox3.MaxLength = 15;  if (!char.IsDigit(e.KeyChar) && (e.KeyChar != (char)Keys.Back))  {  e.Handled = true;  MessageBox.Show("Поле не может содержать буквы!", "Ошибка");  }  } |
| button1 | private void button1\_Click(object sender, EventArgs e)  {  SqlConnection sqlConnect = new SqlConnection("Data Source = sql; Initial Catalog = уП01\_ИСПП5\_Макуров\_ВА; Integrated Security = True");  sqlConnect.Open();  if ((textBox1.Text == "") || (textBox2.Text == ""))  {  MessageBox.Show("Поля не должны быть пустыми.", "Ошибка");  }  else  {  SqlDataAdapter da = new SqlDataAdapter($"INSERT INTO Пользователи (Почта, Логин, Пароль) VALUES ('{textBox1.Text}', '{textBox2.Text}', '{textBox3.Text}');", sqlConnect);  DataTable dt = new DataTable();  da.Fill(dt);  MessageBox.Show("Вы зарегистрированы, как сотрудник.", "Уведомление");  Hide();  Сотрудник UserForm = new Сотрудник();  UserForm.Close();  }  } |
| Регистрация\_MouseDown | private void Регистрация\_MouseDown(object sender, MouseEventArgs e)  {  int xOffset;  int yOffset;  if (e.Button == MouseButtons.Left)  {  xOffset = -e.X - SystemInformation.FrameBorderSize.Width;  yOffset = -e.Y - SystemInformation.CaptionHeight -  SystemInformation.FrameBorderSize.Height;  mouseOffset = new Point(xOffset, yOffset);  isMouseDown = true;  }  } |
| Регистрация\_MouseMove | private void Регистрация\_MouseMove(object sender, MouseEventArgs e)  {  if (isMouseDown)  {  Point mousePos = Control.MousePosition;  mousePos.Offset(mouseOffset.X, mouseOffset.Y);  Location = mousePos;  }  } |
| Регистрация\_MouseUp | private void Регистрация\_MouseUp(object sender, MouseEventArgs e)  {  if (e.Button == MouseButtons.Left)  {  isMouseDown = false;  }  } |
| pictureBox1 | private void pictureBox1\_Click(object sender, EventArgs e)  {  Hide();  Регистрация UserForm = new Регистрация();  UserForm.Close();  } |

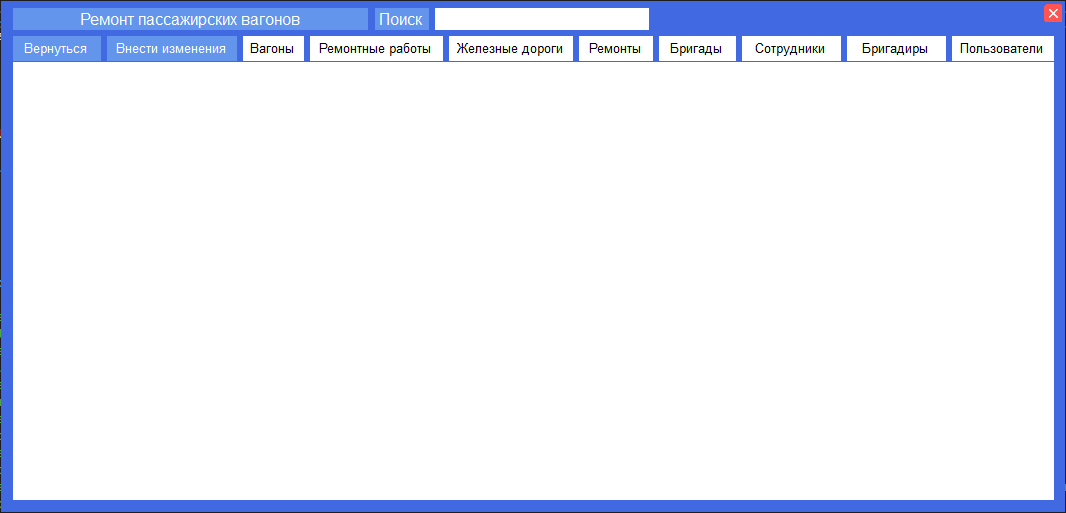


Рисунок 15 – Форма “Администратора”

Свойства элементов формы Администратор представлены в приложении 7 в таблице 13.

Таблица 13 – Свойства элементов формы Администратор

|  |  |  |
| --- | --- | --- |
| Объект | Свойство | Значение |
| Администратор | FormBorderStyle | None |
| Text | Ремонт пассажирских вагонов |
| StartPosition | CenterScreen |
| Size | 1064; 511 |
| ShowIcon | False |

Событийно-управляемые процедуры в форме Администратор представлены в приложении 7 таблице 14.

Таблица 14 – Событийно-управляемые процедуры в форме «Администратор»

|  |  |
| --- | --- |
| Объект | Событийно-управляемые процедуры |
| class Администратор | public partial class Администратор : Form  {  private Point mouseOffset;  private bool isMouseDown = false;  int s;  public Администратор()  {  InitializeComponent();  }  } |
| Select | public void Select(int i)  {  SqlConnection sqlConnect = new SqlConnection("Data Source = sql; Initial Catalog = уП01\_ИСПП5\_Макуров\_ВА; Integrated Security = True");  sqlConnect.Open();  switch (i)  {  case 0: dataGridView1.DataSource = вагоныBindingSource; break;  case 1: dataGridView1.DataSource = ремонтныеРаботыBindingSource; break;  case 2: dataGridView1.DataSource = железныеДорогиBindingSource; break;  case 3: dataGridView1.DataSource = ремонтыBindingSource; break;  case 4: dataGridView1.DataSource = бригадыBindingSource; break;  case 5: dataGridView1.DataSource = сотрудникиBindingSource; break;  case 6: dataGridView1.DataSource = бригадирыBindingSource; break;  case 7: dataGridView1.DataSource = пользователиBindingSource; break;  }  s = i;  } |
| button1 | private void button1\_Click(object sender, EventArgs e)  {  switch (s)  {  case 0: вагоныTableAdapter.Update(уП01\_ИСПП5\_Макуров\_ВАDataSet.Вагоны); break;  case 1: ремонтные\_РаботыTableAdapter.Update(уП01\_ИСПП5\_Макуров\_ВАDataSet.Ремонтные\_Работы); break;  case 2: железные\_ДорогиTableAdapter.Update(уП01\_ИСПП5\_Макуров\_ВАDataSet.Железные\_Дороги); break;  case 3: ремонтыTableAdapter.Update(уП01\_ИСПП5\_Макуров\_ВАDataSet.Ремонты); break;  case 4: бригадыTableAdapter.Update(уП01\_ИСПП5\_Макуров\_ВАDataSet.Бригады); break;  case 5: сотрудникиTableAdapter.Update(уП01\_ИСПП5\_Макуров\_ВАDataSet.Сотрудники); break;  case 6: бригадирыTableAdapter.Update(уП01\_ИСПП5\_Макуров\_ВАDataSet.Бригадиры); break;  case 7: пользователиTableAdapter.Update(уП01\_ИСПП5\_Макуров\_ВАDataSet.Пользователи); break;  }  } |
| Button2 | private void button2\_Click(object sender, EventArgs e)  {  Select(0);  } |
| Button3 | private void button3\_Click(object sender, EventArgs e)  {  Select(1);  } |
| Button4 | private void button4\_Click(object sender, EventArgs e)  {  Hide();  Авторизация UserForm = new Авторизация();  UserForm.Show();  } |
| Button5 | private void button5\_Click(object sender, EventArgs e)  {  Select(2);  } |
| Button6 | private void button6\_Click(object sender, EventArgs e)  {  Select(3);  } |
| Button7 | private void button7\_Click(object sender, EventArgs e)  {  Select(4);  } |
| Button8 | private void button8\_Click(object sender, EventArgs e)  {  Select(5);  } |
| Button9 | private void button9\_Click(object sender, EventArgs e)  {  Select(6);  } |
| Button10 | private void button10\_Click(object sender, EventArgs e)  {  Select(7);  } |
| pictureBox1 | private void pictureBox1\_Click(object sender, EventArgs e)  {  Application.Exit();  } |
| Администратор\_MouseDown | private void Администратор\_MouseDown(object sender, MouseEventArgs e)  {  int xOffset;  int yOffset;  if (e.Button == MouseButtons.Left)  {  xOffset = -e.X - SystemInformation.FrameBorderSize.Width;  yOffset = -e.Y - SystemInformation.CaptionHeight -  SystemInformation.FrameBorderSize.Height;  mouseOffset = new Point(xOffset, yOffset);  isMouseDown = true;  }  } |
| Администратор\_MouseMove | private void Администратор\_MouseMove(object sender, MouseEventArgs e)  {  if (isMouseDown)  {  Point mousePos = Control.MousePosition;  mousePos.Offset(mouseOffset.X, mouseOffset.Y);  Location = mousePos;  }  } |
| Администратор\_MouseUp | private void Администратор\_MouseUp(object sender, MouseEventArgs e)  {  if (e.Button == MouseButtons.Left)  {  isMouseDown = false;  }  } |
| Администратор\_Load | private void Администратор\_Load(object sender, EventArgs e)  {  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП01\_ИСПП5\_Макуров\_ВАDataSet.Пользователи". При необходимости она может быть перемещена или удалена.  this.пользователиTableAdapter.Fill(this.уП01\_ИСПП5\_Макуров\_ВАDataSet.Пользователи);  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП01\_ИСПП5\_Макуров\_ВАDataSet.Бригадиры". При необходимости она может быть перемещена или удалена.  this.бригадирыTableAdapter.Fill(this.уП01\_ИСПП5\_Макуров\_ВАDataSet.Бригадиры);  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП01\_ИСПП5\_Макуров\_ВАDataSet.Сотрудники". При необходимости она может быть перемещена или удалена.  this.сотрудникиTableAdapter.Fill(this.уП01\_ИСПП5\_Макуров\_ВАDataSet.Сотрудники);  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП01\_ИСПП5\_Макуров\_ВАDataSet.Бригады". При необходимости она может быть перемещена или удалена.  this.бригадыTableAdapter.Fill(this.уП01\_ИСПП5\_Макуров\_ВАDataSet.Бригады);  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП01\_ИСПП5\_Макуров\_ВАDataSet.Ремонты". При необходимости она может быть перемещена или удалена.  this.ремонтыTableAdapter.Fill(this.уП01\_ИСПП5\_Макуров\_ВАDataSet.Ремонты);  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП01\_ИСПП5\_Макуров\_ВАDataSet.Железные\_Дороги". При необходимости она может быть перемещена или удалена.  this.железные\_ДорогиTableAdapter.Fill(this.уП01\_ИСПП5\_Макуров\_ВАDataSet.Железные\_Дороги);  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП01\_ИСПП5\_Макуров\_ВАDataSet.Ремонтные\_Работы". При необходимости она может быть перемещена или удалена.  this.ремонтные\_РаботыTableAdapter.Fill(this.уП01\_ИСПП5\_Макуров\_ВАDataSet.Ремонтные\_Работы);  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП01\_ИСПП5\_Макуров\_ВАDataSet.Вагоны". При необходимости она может быть перемещена или удалена.  this.вагоныTableAdapter.Fill(this.уП01\_ИСПП5\_Макуров\_ВАDataSet.Вагоны);  } |
| textBox1\_TextChanged | private void textBox1\_TextChanged(object sender, EventArgs e)  {  for (int i = 0; i < dataGridView1.RowCount; i++)  {  dataGridView1.Rows[i].Selected = false;  for (int j = 0; j < dataGridView1.ColumnCount; j++)  if (dataGridView1.Rows[i].Cells[j].Value != null)  if (dataGridView1.Rows[i].Cells[j].Value.ToString().Contains(textBox1.Text))  {  dataGridView1.Rows[i].Selected = true;  break;  }  }  } |

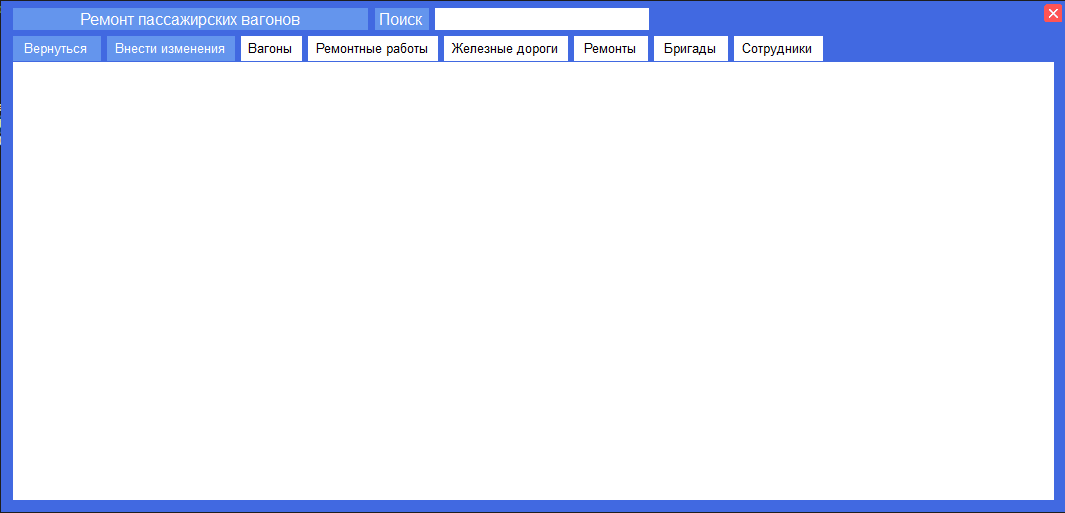


Рисунок 16 – Форма “Бригадира”

Свойства элементов формы Бригадир представлены в приложении 7 в таблице 15.

Таблица 15 – Свойства элементов формы Бригадир

|  |  |  |
| --- | --- | --- |
| Объект | Свойство | Значение |
| Бригадир | FormBorderStyle | None |
| Text | Ремонт пассажирских вагонов |
| StartPosition | CenterScreen |
| Size | 1064; 511 |
| ShowIcon | False |

Событийно-управляемые процедуры в форме Бригадир представлены в приложении 7 таблице 16.

Таблица 16 – Событийно-управляемые процедуры в форме «Бригадир»

|  |  |
| --- | --- |
| Объект | Событийно-управляемые процедуры |
| class Бригадир | public partial class Бригадир : Form  {  private Point mouseOffset;  private bool isMouseDown = false;  int s;  public Бригадир()  {  InitializeComponent();  }  } |
| Select | public void Select(int i)  {  SqlConnection sqlConnect = new SqlConnection("Data Source = sql; Initial Catalog = уП01\_ИСПП5\_Макуров\_ВА; Integrated Security = True");  sqlConnect.Open();  switch (i)  {  case 0: dataGridView1.DataSource = вагоныBindingSource; break;  case 1: dataGridView1.DataSource = ремонтныеРаботыBindingSource; break;  case 2: dataGridView1.DataSource = железныеДорогиBindingSource; break;  case 3: dataGridView1.DataSource = ремонтыBindingSource; break;  case 4: dataGridView1.DataSource = бригадыBindingSource; break;  case 5: dataGridView1.DataSource = сотрудникиBindingSource; break;  }  s = i;  } |
| Бригадир\_Load | private void Бригадир\_Load(object sender, EventArgs e)  {  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП01\_ИСПП5\_Макуров\_ВАDataSet.Сотрудники". При необходимости она может быть перемещена или удалена.  this.сотрудникиTableAdapter.Fill(this.уП01\_ИСПП5\_Макуров\_ВАDataSet.Сотрудники);  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП01\_ИСПП5\_Макуров\_ВАDataSet.Бригады". При необходимости она может быть перемещена или удалена.  this.бригадыTableAdapter.Fill(this.уП01\_ИСПП5\_Макуров\_ВАDataSet.Бригады);  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП01\_ИСПП5\_Макуров\_ВАDataSet.Ремонты". При необходимости она может быть перемещена или удалена.  this.ремонтыTableAdapter.Fill(this.уП01\_ИСПП5\_Макуров\_ВАDataSet.Ремонты);  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП01\_ИСПП5\_Макуров\_ВАDataSet.Железные\_Дороги". При необходимости она может быть перемещена или удалена.  this.железные\_ДорогиTableAdapter.Fill(this.уП01\_ИСПП5\_Макуров\_ВАDataSet.Железные\_Дороги);  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП01\_ИСПП5\_Макуров\_ВАDataSet.Ремонтные\_Работы". При необходимости она может быть перемещена или удалена.  this.ремонтные\_РаботыTableAdapter.Fill(this.уП01\_ИСПП5\_Макуров\_ВАDataSet.Ремонтные\_Работы);  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП01\_ИСПП5\_Макуров\_ВАDataSet.Вагоны". При необходимости она может быть перемещена или удалена.  this.вагоныTableAdapter.Fill(this.уП01\_ИСПП5\_Макуров\_ВАDataSet.Вагоны);  } |
| pictureBox1 | private void pictureBox1\_Click(object sender, EventArgs e)  {  Application.Exit();  } |
| Button4 | private void button4\_Click(object sender, EventArgs e)  {  Hide();  Авторизация UserForm = new Авторизация();  UserForm.Show();  } |
| Button8 | private void button8\_Click(object sender, EventArgs e)  {  switch (s)  {  case 0: вагоныTableAdapter.Update(уП01\_ИСПП5\_Макуров\_ВАDataSet.Вагоны); break;  case 1: ремонтные\_РаботыTableAdapter.Update(уП01\_ИСПП5\_Макуров\_ВАDataSet.Ремонтные\_Работы); break;  case 2: железные\_ДорогиTableAdapter.Update(уП01\_ИСПП5\_Макуров\_ВАDataSet.Железные\_Дороги); break;  case 3: ремонтыTableAdapter.Update(уП01\_ИСПП5\_Макуров\_ВАDataSet.Ремонты); break;  case 4: бригадыTableAdapter.Update(уП01\_ИСПП5\_Макуров\_ВАDataSet.Бригады); break;  case 5: сотрудникиTableAdapter.Update(уП01\_ИСПП5\_Макуров\_ВАDataSet.Сотрудники); break;  }  } |
| Button2 | private void button2\_Click(object sender, EventArgs e)  {  Select(0);  } |
| Button3 | private void button3\_Click(object sender, EventArgs e)  {  Select(1);  } |
| Button5 | private void button5\_Click(object sender, EventArgs e)  {  Select(2);  } |
| Button6 | private void button6\_Click(object sender, EventArgs e)  {  Select(3);  } |
| Button1 | private void button1\_Click(object sender, EventArgs e)  {  Select(4);  } |
| Button7 | private void button7\_Click(object sender, EventArgs e)  {  Select(5);  } |
| textBox1\_TextChanged | private void textBox1\_TextChanged(object sender, EventArgs e)  {  for (int i = 0; i < dataGridView1.RowCount; i++)  {  dataGridView1.Rows[i].Selected = false;  for (int j = 0; j < dataGridView1.ColumnCount; j++)  if (dataGridView1.Rows[i].Cells[j].Value != null)  if (dataGridView1.Rows[i].Cells[j].Value.ToString().Contains(textBox1.Text))  {  dataGridView1.Rows[i].Selected = true;  break;  }  }  } |
| Бригадир\_MouseDown | private void Бригадир\_MouseDown(object sender, MouseEventArgs e)  {  int xOffset;  int yOffset;  if (e.Button == MouseButtons.Left)  {  xOffset = -e.X - SystemInformation.FrameBorderSize.Width;  yOffset = -e.Y - SystemInformation.CaptionHeight -  SystemInformation.FrameBorderSize.Height;  mouseOffset = new Point(xOffset, yOffset);  isMouseDown = true;  }  } |
| Бригадир\_MouseMove | private void Бригадир\_MouseMove(object sender, MouseEventArgs e)  {  if (isMouseDown)  {  Point mousePos = Control.MousePosition;  mousePos.Offset(mouseOffset.X, mouseOffset.Y);  Location = mousePos;  }  } |
| Бригадир\_MouseUp | private void Бригадир\_MouseUp(object sender, MouseEventArgs e)  {  if (e.Button == MouseButtons.Left)  {  isMouseDown = false;  }  } |

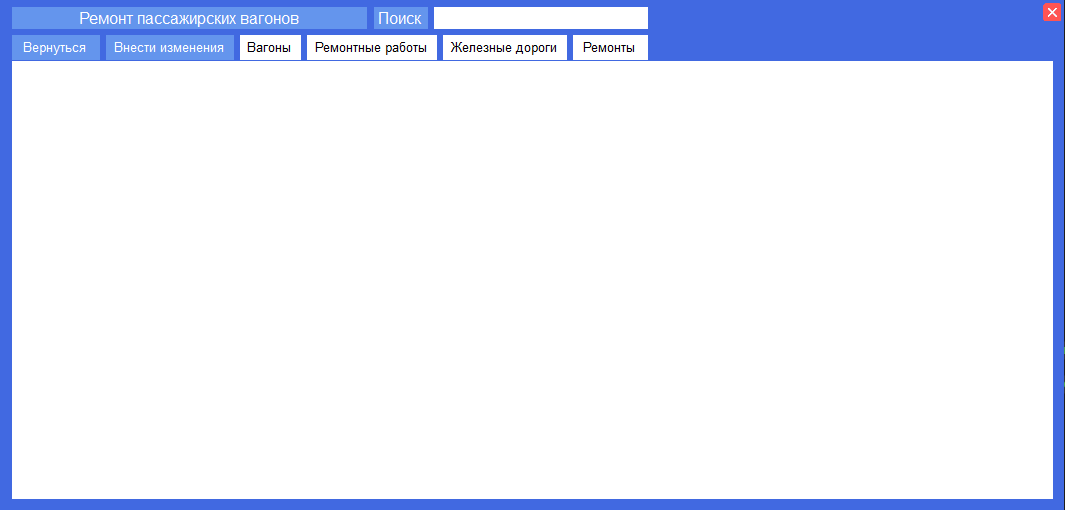


Рисунок 17 – Форма “Сотрудника”

Свойства элементов формы Сотрудник представлены в приложении 7 в таблице 17.

Таблица 17 – Свойства элементов формы Бригадир

|  |  |  |
| --- | --- | --- |
| Объект | Свойство | Значение |
| Сотрудник | FormBorderStyle | None |
| Text | Ремонт пассажирских вагонов |
| StartPosition | CenterScreen |
| Size | 1064; 511 |
| ShowIcon | False |

Событийно-управляемые процедуры в форме Сотрудник представлены в приложении 7 таблице 18.

Таблица 18 – Событийно-управляемые процедуры в форме «Сотрудник»

|  |  |
| --- | --- |
| Объект | Событийно-управляемые процедуры |
| class Сотрудник | public partial class Сотрудник : Form  {  private Point mouseOffset;  private bool isMouseDown = false;  int s;  public Сотрудник()  {  InitializeComponent();  }  } |
| Select | public void Select(int i)  {  SqlConnection sqlConnect = new SqlConnection("Data Source = sql; Initial Catalog = уП01\_ИСПП5\_Макуров\_ВА; Integrated Security = True");  sqlConnect.Open();  switch (i)  {  case 0: dataGridView1.DataSource = вагоныBindingSource; break;  case 1: dataGridView1.DataSource = ремонтныеРаботыBindingSource; break;  case 2: dataGridView1.DataSource = железныеДорогиBindingSource; break;  case 3: dataGridView1.DataSource = ремонтыBindingSource; break;  }  s = i;  } |
| Button4 | private void button4\_Click(object sender, EventArgs e)  {  Hide();  Авторизация UserForm = new Авторизация();  UserForm.Show();  } |
| Button2 | private void button2\_Click(object sender, EventArgs e)  {  Select(0);  } |
| Button3 | private void button3\_Click(object sender, EventArgs e)  {  Select(1);  } |
| Button5 | private void button5\_Click(object sender, EventArgs e)  {  Select(2);  } |
| Button6 | private void button6\_Click(object sender, EventArgs e)  {  Select(3);  } |
| pictureBox1 | private void pictureBox1\_Click(object sender, EventArgs e)  {  Application.Exit();  } |
| Сотрудник\_MouseDown | private void Сотрудник\_MouseDown(object sender, MouseEventArgs e)  {  int xOffset;  int yOffset;  if (e.Button == MouseButtons.Left)  {  xOffset = -e.X - SystemInformation.FrameBorderSize.Width;  yOffset = -e.Y - SystemInformation.CaptionHeight -  SystemInformation.FrameBorderSize.Height;  mouseOffset = new Point(xOffset, yOffset);  isMouseDown = true;  }  } |
| Сотрудник\_MouseMove | private void Сотрудник\_MouseMove(object sender, MouseEventArgs e)  {  if (isMouseDown)  {  Point mousePos = Control.MousePosition;  mousePos.Offset(mouseOffset.X, mouseOffset.Y);  Location = mousePos;  }  } |
| Сотрудник\_MouseUp | private void Сотрудник\_MouseUp(object sender, MouseEventArgs e)  {  if (e.Button == MouseButtons.Left)  {  isMouseDown = false;  }  } |
| Сотрудник\_Load | private void Сотрудник\_Load(object sender, EventArgs e)  {  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП01\_ИСПП5\_Макуров\_ВАDataSet.Ремонты". При необходимости она может быть перемещена или удалена.  this.ремонтыTableAdapter.Fill(this.уП01\_ИСПП5\_Макуров\_ВАDataSet.Ремонты);  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП01\_ИСПП5\_Макуров\_ВАDataSet.Железные\_Дороги". При необходимости она может быть перемещена или удалена.  this.железные\_ДорогиTableAdapter.Fill(this.уП01\_ИСПП5\_Макуров\_ВАDataSet.Железные\_Дороги);  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП01\_ИСПП5\_Макуров\_ВАDataSet.Ремонтные\_Работы". При необходимости она может быть перемещена или удалена.  this.ремонтные\_РаботыTableAdapter.Fill(this.уП01\_ИСПП5\_Макуров\_ВАDataSet.Ремонтные\_Работы);  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП01\_ИСПП5\_Макуров\_ВАDataSet.Вагоны". При необходимости она может быть перемещена или удалена.  this.вагоныTableAdapter.Fill(this.уП01\_ИСПП5\_Макуров\_ВАDataSet.Вагоны);  } |
| Button1 | private void button1\_Click(object sender, EventArgs e)  {  switch (s)  {  case 0: вагоныTableAdapter.Update(уП01\_ИСПП5\_Макуров\_ВАDataSet.Вагоны); break;  case 1: ремонтные\_РаботыTableAdapter.Update(уП01\_ИСПП5\_Макуров\_ВАDataSet.Ремонтные\_Работы); break;  case 2: железные\_ДорогиTableAdapter.Update(уП01\_ИСПП5\_Макуров\_ВАDataSet.Железные\_Дороги); break;  case 3: ремонтыTableAdapter.Update(уП01\_ИСПП5\_Макуров\_ВАDataSet.Ремонты); break;  }  } |
| textBox1\_TextChanged | private void textBox1\_TextChanged(object sender, EventArgs e)  {  for (int i = 0; i < dataGridView1.RowCount; i++)  {  dataGridView1.Rows[i].Selected = false;  for (int j = 0; j < dataGridView1.ColumnCount; j++)  if (dataGridView1.Rows[i].Cells[j].Value != null)  if (dataGridView1.Rows[i].Cells[j].Value.ToString().Contains(textBox1.Text))  {  dataGridView1.Rows[i].Selected = true;  break;  }  }  } |

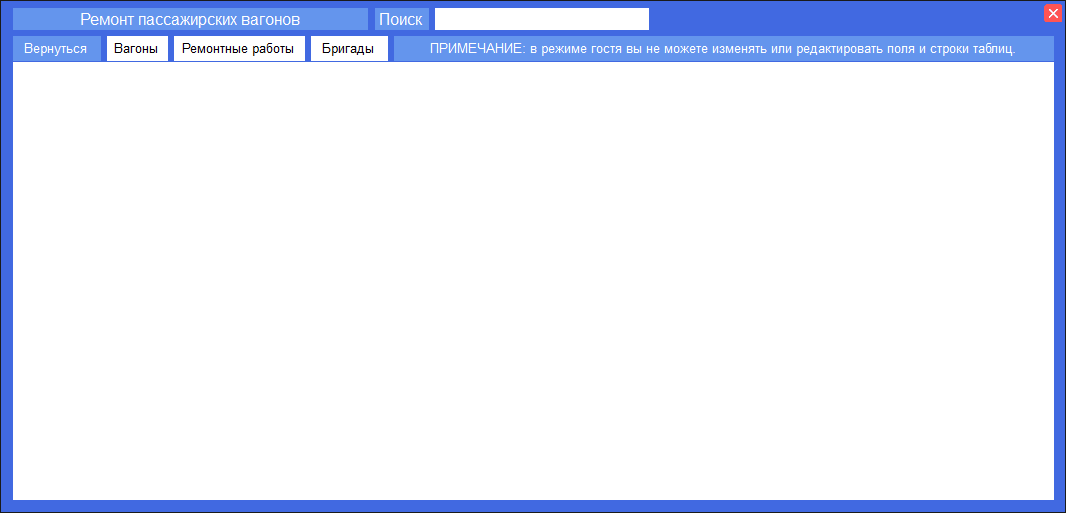


Рисунок 17 – Форма “Гостя”

Свойства элементов формы Гость представлены в приложении 7 в таблице 19.

Таблица 19 – Свойства элементов формы Гость

|  |  |  |
| --- | --- | --- |
| Объект | Свойство | Значение |
| Сотрудник | FormBorderStyle | None |
| Text | Ремонт пассажирских вагонов |
| StartPosition | CenterScreen |
| Size | 1064; 511 |
| ShowIcon | False |

Событийно-управляемые процедуры в форме Гость представлены в приложении 7 таблице 20.

Таблица 20 – Событийно-управляемые процедуры в форме «Гость»

|  |  |
| --- | --- |
| Объект | Событийно-управляемые процедуры |
| class Гость | public partial class Гость : Form  {  private Point mouseOffset;  private bool isMouseDown = false;  public Гость()  {  InitializeComponent();  }  } |
| Select | public void Select(int i)  {  SqlConnection sqlConnect = new SqlConnection("Data Source = sql; Initial Catalog = уП01\_ИСПП5\_Макуров\_ВА; Integrated Security = True");  sqlConnect.Open();  SqlDataAdapter[] sa = new SqlDataAdapter[3];  sa[0] = new SqlDataAdapter("select \* from Вагоны", sqlConnect);  sa[1] = new SqlDataAdapter("select \* from Ремонтные\_Работы", sqlConnect);  sa[2] = new SqlDataAdapter("select \* from Бригады", sqlConnect);  DataSet ds = new DataSet();  sa[i].Fill(ds);  dataGridView1.DataSource = ds.Tables[0];  } |
| pictureBox1 | private void pictureBox1\_Click(object sender, EventArgs e)  {  Application.Exit();  } |
| Button2 | private void button2\_Click(object sender, EventArgs e)  {  Select(0);  } |
| Button3 | private void button3\_Click(object sender, EventArgs e)  {  Select(1);  } |
| Button7 | private void button7\_Click(object sender, EventArgs e)  {  Select(2);  } |
| Button4 | private void button4\_Click(object sender, EventArgs e)  {  Hide();  Авторизация UserForm = new Авторизация();  UserForm.Show();  } |
| Гость\_MouseDown | private void Гость\_MouseDown(object sender, MouseEventArgs e)  {  int xOffset;  int yOffset;  if (e.Button == MouseButtons.Left)  {  xOffset = -e.X - SystemInformation.FrameBorderSize.Width;  yOffset = -e.Y - SystemInformation.CaptionHeight -  SystemInformation.FrameBorderSize.Height;  mouseOffset = new Point(xOffset, yOffset);  isMouseDown = true;  }  } |
| Гость\_MouseMove | private void Гость\_MouseMove(object sender, MouseEventArgs e)  {  if (isMouseDown)  {  Point mousePos = Control.MousePosition;  mousePos.Offset(mouseOffset.X, mouseOffset.Y);  Location = mousePos;  }  } |
| Гость\_MouseUp | private void Гость\_MouseUp(object sender, MouseEventArgs e)  {  if (e.Button == MouseButtons.Left)  {  isMouseDown = false;  }  } |
| textBox1\_TextChanged | private void textBox1\_TextChanged(object sender, EventArgs e)  {  for (int i = 0; i < dataGridView1.RowCount; i++)  {  dataGridView1.Rows[i].Selected = false;  for (int j = 0; j < dataGridView1.ColumnCount; j++)  if (dataGridView1.Rows[i].Cells[j].Value != null)  if (dataGridView1.Rows[i].Cells[j].Value.ToString().Contains(textBox1.Text))  {  dataGridView1.Rows[i].Selected = true;  break;  }  }  } |

Вывод по главе: была полностью спроектирована и разработана система базы данных, были полностью спроектированы и разработаны все программные модули информационной системы “Ремонт пассажирских вагонов”.

# РАЗРАБОТКА ТЕСТОВЫХ НАБОРОВ И ТЕСТОВЫХ СЦЕНАРИЕВ

Тестирование программного обеспечения – процесс исследования, испытания программного продукта, имеющий своей целью проверку соответствия между реальным поведением программы и её ожидаемым поведением на конечном наборе тестов, выбранных определённым образом.

Тестовый сценарий – это неавтоматизированный или автоматизированный сценарий, содержащий инструкции по реализации тестового набора. Тестовый сценарий может быть написан вручную (для выполнения человеком) либо полностью или частично автоматизирован.

## Назначение эксперимента. Выбор и обоснование методики проведения тестирования

В данном разделе проведены два вида тестирования. Провести анализ тестирования и выбрать как будет проходить тестирование ИС.

Модульное тестирование (Unit test). Цель: проверить, что код работает именно так, как должен (при заданных входных параметрах выдает предполагаемый результат).

Анализ покрытия кода тестами (Test coverage). Цель: проверить, что весь наш код отрабатывает при модульном тестировании, что нет не участвующих в тестировании участков кода.

Анализ стилистики кода. Цель: проверка кода на соблюдение стандартов разработки (отраслевых, корпоративных, принятых на проекте).

Анализ производительности (Benchmark, Profiling). Цель: проанализировать производительность ключевых операций, потребление памяти, утечку памяти.

## Технология тестирования

## Результаты проведения тестирования

Таблица 21 – Результаты тестирования

|  |  |  |
| --- | --- | --- |
| Число копий | Нагрузка на процессор (%) | Нагрузка на память (%) |
| 1 | 0,1 | 0,2 |
| 5 | 0,1 | 1 |
| 10 | 2,1 | 2 |
| 15 | 3,5 | 3 |
| 25 | 4,8 | 5 |

Исходя из результатов тестирования, можно прийти к выводу, что нагрузка, оказываемая на центральный процессор и память, не имеет критичных значений. Это значит, что показатель производительности информационной системы на приемлемом уровне, то есть сбои и длительная загрузка не должны помешать пользователю при работе с данным программным продуктом.

# ЗАКЛЮЧЕНИЕ

В результате учебной практики был проведен анализ предметной области, спроектирована и разработана информационная система «Ремонт пассажирских вагонов», серверная часть которой была реализована в MS SQL, а клиентская часть в MS Visual Studio.

В MS SQL была создана база данных для хранения информации о вагонах, персонале, должностях, видов ремонта и о самом ремонте вагонов.

Клиентская часть, реализованная в MS Visual Studio, отражает все хранящиеся в базе данных таблицы. Также разработаны некоторые запросы, позволяющие редактировать и удалять существующие данные, добавлять новые и осуществлять поиск по таблицам.

Цели учебной практики были достигнуты путем проведения анализа предметной области, проектирования информационной системы и разработки программных модулей.

Это позволяет сделать вывод, что поставленные цели и задачи выполнены в полном объеме.

В дальнейшем данная информационная система может быть модернизирована путем добавления формы личного кабинета, создания версии для слабовидящих или редизайна.

# БИБЛИОГРАФИЧЕСКИЙ СПИСОК