МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования

Гомельский государственный технический университет имени

П.О. Сухого

Факультет автоматизированных и информационных систем

Кафедра «Информатика»

специальность 1-40 04 01 «Информатика и технологии программирования»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к дипломной работе**

**на тему**

**«Программный комплекс учета лабораторных анализов в лаборатории промышленного предприятия»**

Разработал студент гр. ИП-41                        Хрол Д.А.

(подпись) (Ф.И.О.)

Руководитель работы                        Шибеко В.Н.

(подпись) (ученое звание, ученая степень, Ф.И.О.)

Консультант   
по экономической части                          профессор, д.э.н., Голуб В.А.

(подпись) (ученое звание, ученая степень, Ф.И.О.)

Консультант по охране труда

и технике безопасности                        профессор, д.т.н., Кудин В.П.

(подпись) (ученое звание, ученая степень, Ф.И.О.)

Нормоконтроль                         Самовендюк Н.В.

(подпись) (ученое звание, ученая степень, Ф.И.О.)

Рецензент                        профессор, д.т.н., кафедры ИТ, ГГТУ

им. П.О. Сухого, Мурашко И.А.

(подпись) (ученое звание, ученая степень, должность,

организация, Ф.И.О.)

Дипломная работа ( \_\_\_\_\_\_с.) допущена к защите   
в Государственной экзаменационной комиссии.

Зав. кафедрой   
«Информатика»                        доцент, к.т.н, Трохова Т.А.

(подпись) (ученое звание, ученая степень, Ф.И.О.)

Гомель 2020

**Лист задания**

**РЕФЕРАТ**

ПРОГРАММНЫЙ КОМПЛЕКС УЧЕТА ЛАБОРАТОРНЫХ АНАЛИЗОВ В ЛАБОРАТОРИИ ПРОМЫШЛЕННОГО ПРЕДПРИЯТИЯ: дипломная работа / Д. А. Хрол. – Гомель : ГГТУ им. П.О. Сухого, 2020. – Дипломная работа: 72 страницы, 29 рисунков, 26 таблиц, 10 источников, 5 приложений.

Ключевые слова: заказчик, передел, конечный продукт, испытание, статус продукта, вывод данных.

Объектом разработки является программный продукт, направленный на ведение и контроль лабораторных анализов.

Цель работы: создание программного комплекса учета лабораторных анализов в лаборатории промышленного предприятия на базе операционной системы Windows.

В процессе работы было сделано следующее: выполнен анализ существующего программного обеспечения по выполнению планирования личного времени, создана база данных для хранения и обработки данных каналов, планов канала и пользователей.

Основными функциями разрабатываемого программного комплекса являются:

* регистрация продукта или передела;
* фиксирование результатов набора испытаний над переделом или конечным продуктом;
* принятие и фиксирование статуса продукта;
* вывод статистики результатов испытаний по качеству передела или конечного продукта.

Элементом новизны является отсутствие аналогов, возможность получить подробную информацию о качестве продукта на разных этапах его производства, определение неисправности в производстве, в случае брака.

Областью практического применения является применение этого программного обеспечения на промышленных предприятиях планового подхода.

Студент-дипломник подтверждает, что дипломная работа выполнена самостоятельно, приведенный в дипломной работе материал объективно отражает состояние разрабатываемого объекта, пояснительная записка проверена в системе «Антиплагиат» «АО "Антиплагиат"» (режим доступа: https://www.antiplagiat.ru/). Процент оригинальности составляет 78.83%. Все заимствованные из литературных и других источников, теоретические и методологические положения и концепции сопровождаются ссылками на источники, указанные в «Списке использованных источников».

**РЕЗЮМЕ**

Тема дипломной работы «Программный комплекс учета лабораторных анализов в лаборатории промышленного предприятия».

В процессе выполнения данной дипломной работы был разработан программный продукт, который позволяет вести учет лабораторных анализов и отслеживать качество выпускаемой продукции.

Объектом разработки является программный продукт, направленный на ведение и контроль лабораторных анализов.

Поставленная задача была выполнена в полном объеме.

**РЭЗЮМЭ**

Тэма дыпломнай працы «Праграмны комплекс ўліку лабараторных аналізаў ў лабараторыі прамысловага прадпрыемства».

У працэсе выканання дадзенай дыпломнай працы быў распрацаваны праграмны прадукт, які дазваляе весці ўлік лабараторных аналізаў і адсочваць якасць выпускаемай прадукцыі.

Аб'ектам распрацоўкі з'яўляецца праграмны прадукт, накіраваны на вядзенне і кантроль лабараторных аналізаў.

Пастаўленая задача была выканана ў поўным аб'ёме.

**SUMMARY**

The subject of degree work – "The software package for laboratory analysis in the laboratory of an industrial enterprise."

In the course of implementation of the degree work, I have executed software that allows you to keep track of laboratory tests and monitor the quality of products.

Subject of development is a software product aimed at maintaining and monitoring laboratory tests.

The task has been carried out successfully and completely.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 7](#_Toc42791323)

[1 АНАЛИТИЧЕСКИЙ ОБЗОР СУЩЕСТВУЮЩИХ МЕТОДОВ И СРЕДСТВ АВТОМАТИЗАЦИИ ЛАБОРАТОРНЫХ ИСПЫТАНИЙ НА ПРЕДПРИЯТИИ 8](#_Toc42791324)

[1.1 Обзор существующих систем автоматизации 8](#_Toc42791325)

[1.2 Анализ современного языка программирования C# 9](#_Toc42791326)

[1.3 Анализ используемых технологий для реализации поставленной задачи 11](#_Toc42791327)

[1.4 Анализ инструментальных средств автоматизации разработки и тестирования 15](#_Toc42791328)

[1.5 Техническое задание для клиент-серверного программного продукта «Программный комплекс учета лабораторных анализов в лаборатории промышленного предприятия» 18](#_Toc42791329)

[2 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И АЛГОРИТМЫ 19](#_Toc42791330)

[2.1 Анализ предметной области 19](#_Toc42791331)

[2.2 Алгоритм работы программного комплекса 19](#_Toc42791332)

[2.3 Функциональная модель программного комплекса 19](#_Toc42791333)

[2.4 Информационная модель программного комплекса 22](#_Toc42791334)

[3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ КОМПЛЕКСА 31](#_Toc42791335)

[3.1 Общая структура клиентской части приложения 31](#_Toc42791336)

[3.2 Уровень представления клиентской части приложения 32](#_Toc42791337)

[3.3 Уровень бизнес-логики клиентской части приложения 40](#_Toc42791338)

[3.4 Уровень данных в приложении 45](#_Toc42791339)

[4 ТЕСТИРОВАНИЕ, ВЕРИФИКАЦИЯ И ВАЛИДАЦИЯ ПРОГРАММНОГО КОМПЛЕКСА 46](#_Toc42791340)

[5 ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ ДИПЛОМНОЙ РАБОТЫ 50](#_Toc42791341)

[5.1 Расчёт общей трудоемкости разработки программного обеспечения 50](#_Toc42791342)

[5.2 Расчёт затрат на разработку программного продукта 52](#_Toc42791343)

[5.3 Формирование цены при создании программного обеспечения 59](#_Toc42791344)

[5.4 Расчет годового экономического эффекта от использования нового ПО……………………………………………………………………………….61](#_Toc42791345)

[5.5 Оценка конкурентоспособности программного продукта 62](#_Toc42791346)

[6 ОХРАНА ТРУДА И ТЕХНИКА БЕЗОПАСНОСТИ 64](#_Toc42791347)

[6.1 Общие положения инструкции по оказанию первой доврачебной помощи 64](#_Toc42791348)

[6.2 Признаки для определения состояния здоровья пострадавшего 65](#_Toc42791349)

[6.3 Комплекс реанимационных мероприятий 65](#_Toc42791350)

[6.3.1 Искусственное дыхание 66](#_Toc42791351)

[6.3.2 Наружный массаж сердца 67](#_Toc42791352)

[ЗАКЛЮЧЕНИЕ 69](#_Toc42791353)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 70](#_Toc42791354)

[ПРИЛОЖЕНИЕ A 71](#_Toc42791355)

[ПРИЛОЖЕНИЕ Б 72](#_Toc42791356)

[ПРИЛОЖЕНИЕ В 73](#_Toc42791357)

[ПРИЛОЖЕНИЕ Г 74](#_Toc42791358)

[ПРИЛОЖЕНИЕ Д 75](#_Toc42791359)

# **ВВЕДЕНИЕ**

В настоящее время ни одно промышленное предприятие не обходится без планового и чёткого тестирования своей продукции на различных этапах его производства.

Целью дипломной работы является создание программного комплекса учета лабораторных анализов в лаборатории промышленного предприятия на базе операционной системы Windows. В дальнейшем, под испытаниями будут предполагаться анализы переделов и конечной продукции. Предполагается использование разрабатываемого программного продукта на заводах и фабриках промышленного производства. У большинства таких предприятий есть общие критерии – серийность и плановый выпуск продукции. Эти критерии позволяют создать общий макет системы, которая подходит всем таким предприятиям.

Задачами дипломной работы являются:

* изучение методик разработки клиент-серверных приложений;
* классификация ролей пользователей и их ролевые политики;
* изучение методов реализации серверной части для приложения;
* проектирование структуры приложения, базы данных для хранения информации, формирование пользовательских правил для доступа к ресурсам и функциям приложения;
* разработка программных модулей: обеспечивающих авторизацию и аутентификацию пользователей; работу с данными с помощью графического интерфейса;
* верификация и опытная эксплуатация разработанного программного обеспечения.

# **1 АНАЛИТИЧЕСКИЙ ОБЗОР СУЩЕСТВУЮЩИХ МЕТОДОВ И СРЕДСТВ АВТОМАТИЗАЦИИ ЛАБОРАТОРНЫХ ИСПЫТАНИЙ НА ПРЕДПРИЯТИИ**

## **Обзор существующих систем автоматизации**

Многие предприятия создают свои локальные программы для автоматизации производственных процессов. Так, например, на БМЗ, для автоматизации производственных процессов, используются различные небольшие приложения на языке Delphi. Такой подход, создания многих небольших программ, позволяет быстро устранять неполадки или обновлять программы в случае необходимости. Все эти программы связаны с серверной частью, которая выражена в виде СУБД Oracle. Автоматизация, представленная в таком виде, имеет строго локальный характер, её нельзя будет применить в подобном предприятии без масштабных изменений.

Лабораторная информационная менеджмент-система (ЛИМС) I-LDS.

Функциональные возможности ЛИМС I-LDS позволили:

* формировать электронные лабораторные журналы, что способствует систематизации аналитических данных;
* архивировать результаты лабораторных анализов, что предоставило широкие возможности для анализа долговременных данных;
* перейти на электронные журналы, что привело к отказу от кропотливого процесса ведения "бумажных" журналов.

Автоматизация формирования отчетных документов экономит время на их подготовку, повышает их достоверность и существенно сокращает время на поиск необходимой информации. Дизайнер отчетов в I-LDS позволяет генерировать неограниченное количество самых разных форм отчетов:

* получение статистических данных по качеству сырья и материалов, полуфабрикатов и готовой продукции с группировкой по периодам (сутки, декада, месяц, и т.д.) либо по настраиваемым параметрам, например, по сменам;
* составление сводок по соблюдению требований норм технологического режима, учет процента отклонений и т.д.

Таким образом, ЛИМС I-LDS, став информационным ядром контроля качества, способствует реализации основных функций центра аналитики и контроля качества (ЦАKK) и обеспечивает выполнение следующих задач:

* оптимизация и повышение эффективности работы сотрудников лаборатории, своевременное предоставление руководству корректной информации о качестве работы лаборатории;
* единообразие выполнения функций лабораториями ЦАKK;
* автоматизация документооборота;
* оперативное информирование персонала заинтересованных служб предприятия о результатах испытаний для своевременного реагирования на возникшие несоответствия;
* стандартизация бизнес-процессов лаборатории за счет планирования ее деятельности и рационального использования ресурсов (персонала, приборов, оборудования, реагентов и стандартных образцов);
* исполнение процедур в соответствии с требованиями нормативных документов [1].

## **Анализ современного языка программирования C#**

С# был разработан благодаря усилиям Андерса Хейлсберга - создателя компилятора, который лег в основу Turbo Pascal и языка программирования Delphi. Первая версия языка увидел свет в июне 2000 года (возможно, что майкрософт хотела отметить так новое тысячелетие), а окончательная версия вышла в 2002 году вместе с Visual Studio. Сейчас C# стало одним из самых популярных языков программирования, начавшим даже немного опережать своего предшественника.

C# нельзя рассматривать только как язык, это часть большой системы, которая включает в себя ОС Windows, среду разработки Visual Studio (и другие инструменты), ну и, конечно, саму Microsoft, которая обеспечивает поддержку этого языка. C# будет существовать и развиваться, пока существует Microsoft и пока люди пользуются Windows, а это будет ещё очень и очень долго.

Поскольку язык принадлежит компании Microsoft, то применяется практически во всех продуктах, которые были разработаны или куплены. Рассмотрим наиболее интересные из них.

**Mono** - проект, который был посвящен свободной реализации C# и .NET. То есть позволял бы писать на C#, например, для Linux и Mac OS X. На основе Mono была создана XamarinStudio, которая позволяет создавать мобильные приложения на C#, не используя родные(нативные) для языков платформы (Java и Objective-C). Была выкуплена у автора (Мигеля де Иказа) и внедрена в Visual Studio.

**DirectX** - API для программирования под Windows, чаще всего используется при программировании игр. При помощи DirectX вы можете написать, скажем, отличный трехмерный шутер.

**Unity** - кроссплатформенный игровой движок, который позволяет создавать двухмерные и трехмерные игры. Очень популярен в среде инди-разработчиков, но берется на вооружение и крупными компания. Так, например, Hearthstone создан на Unity.

Плюсы языка программирования C#:

* поддержка Microsoft. В отличии от Java, которой не пошел на пользу переход в собственность Oracle, C# хорошо развивается благодаря усилиям Microsoft;
* в последнее время много совершенствуется. Так как C# был создан позже, чем Java и другие языки, то требовалось очень много доработать. Также это касается популяризации и бесплатности - было обещано открыть исходный код, а инструменты (Visual Studio, Xamarin) стали бесплатными для частных лиц и небольших компаний;
* много синтаксического сахара. Синтаксический сахар - это такие конструкции, которые созданы для облегчения написания и понимания кода (особенно если это код другого программиста) и не играют роли при компиляции;
* средний порог вхождения. Синтаксис похожий на C, С++ или Java облегчает переход для других программистов. Для новичков это также один из самых перспективных языков для изучения;
* Xamarin. Благодаря покупке Xamarin на C# теперь можно писать под Android и iOS. Это, несомненно, большой плюс, так как их собственная мобильная ОС (Windows Phone) не завоевала большой популярности;
* добавлено функциональное программирование (F#);
* большое сообщество программистов;
* много вакансий на должность C# программиста в любом регионе.

Минусы языка программирования C#:

* ориентированность, в основном, только на .NET (на Windows платформу);
* бесплатность только для небольших компании, учащихся и программистов-одиночек. Для больших команд покупка лицензий обойдется недешево. Поэтому если у вас есть своя фирма, то придется раскошелиться;
* сохранили оператор go to.

С языком программирования C# взаимодействуют множество технологий, которые будут описаны ниже.

**LINQ (Language Integrated Query)** - язык интегрированных запросов, напоминающий SQL, в языках для платформы .NET.

**Microsoft Visual Studio** - один из самых известных продуктов от Microsoft. Среда разработки, которая позволяет писать код на C#, С++ и некоторых других (при установке плагинов). Существует версия VS Community для некоммерческого использования и небольших команд разработчиков.

**ASP.NET** - технология для веб-приложений (и сервисов), продолжение старой технологии ASP (Active Server Pages).

**TPL (Task Parallel Library)** - библиотека для создания многопоточных приложений.

**WCF(Windows Communication Foundation)** - среда выполнения и набор API-интерфейсов для создания систем, которые обеспечивают обмен сообщениями между службами и клиентами.

**WPF (Windows Presentation Foundation)** - система для построения приложений с визуальный интерфейсом для Windows, позволяет добавлять двухмерную трехмерную графику, анимацию и.т.д.

**Resharper** - плагин для Visual Studio, который позволяет обеспечивает дополнительные функции при написании кода - анализ и подсветка ошибок и опечаток, улучшенную навигацию по коду и статический анализ кода (без выполнения программы)

**F#** - язык для функционального программирования под .NET

Данный язык программирования и сопутствующие его технологии полностью подходят для выполнения поставленной задачи. Далее будут подробнее описаны выбранные мною технологии, взаимодействующие с данным языком программирования. [2].

## **Анализ используемых технологий для реализации поставленной задачи**

Для создания программного комплекса учёта лабораторных анализов в лаборатории промышленного предприятия будет использована среда разработки Visual Studio 2019 Community, программная платформа .NET Framework с использованием интеллектуальной клиентской технологии Windows Forms. Серверная часть программного комплекса будет выражена в виде базы данных на Microsoft SQL Server. Выбранные технологии обладают достаточным функционалом для выполнения поставленной задачи.

**Microsoft Visual Studio** - полнофункциональная интегрированная среда разработки (IDE) с поддержкой популярных языков программирования, среди которых С, C++, VB.NET, C#, F#, JavaScript, Python.

Функциональность Visual Studio охватывает все этапы разработки программного обеспечения, предоставляя современные инструменты для написания кода, проектирования графических интерфейсов, сборки, отладки и тестирования приложений. Возможности Visual Studio могут быть дополнены путем подключения необходимых расширений.

Редактор кода Visual Studio поддерживает подсветку синтаксиса, вставку фрагментов кода, отображение структуры и связанных функций. Существенно ускорить работу помогает технология IntelliSense - автозавершение кода по мере ввода.

Встроенный отладчик Visual Studio используется для поиска и исправления ошибок в исходном коде, в том числе на низком аппаратном уровне. Инструменты диагностики позволяют оценить качество кода с точки зрения производительности и использования памяти.

Дизайнер форм Visual Studio незаменим при разработке программ с графическим интерфейсом, помогая спроектировать внешний вид будущего приложения и работу каждого элемента интерфейса.

Наконец, Visual Studio предоставляет комплекс инструментов для автоматизации тестирования приложений в части проверки работы интерфейсов, модульного и нагрузочного тестирования.

Для командных проектов Visual Studio предлагает поддержку групповой работы, позволяя выполнять совместное редактирование и отладку любой части кода в реальном времени, а в качестве системы управления версиями использовать Team Foundation или Git [3].

**Платформа .NET Framework** состоит из общеязыковой среды выполнения (среды CLR) и библиотеки классов .NET Framework. Основой платформы .NET Framework является среда CLR. Среду выполнения можно считать агентом, который управляет кодом во время выполнения и предоставляет основные службы, такие как управление памятью, управление потоками и удаленное взаимодействие. При этом средой накладываются условия строгой типизации и другие виды проверки точности кода, обеспечивающие безопасность и надежность. Фактически основной задачей среды выполнения является управление кодом. Код, который обращается к среде выполнения, называют управляемым кодом, а код, который не обращается к среде выполнения, называют неуправляемым кодом. Библиотека классов является комплексной объектно-ориентированной коллекцией повторно используемых типов, которые применяются для разработки приложений — начиная с обычных приложений, запускаемых из командной строки, и приложений с графическим интерфейсом (GUI) и заканчивая приложениями, использующими последние технологические возможности ASP.NET, такие как веб-формы и веб-службы XML.

Среда CLR управляет памятью, выполнением потоков, выполнением кода, проверкой безопасности кода, компиляцией и другими системными службами. Эти средства являются внутренними для управляемого кода, который выполняется в среде CLR.

По соображениям безопасности управляемым компонентам присваиваются разные степени доверия, зависящие от ряда факторов, в число которых входит их происхождение (например, Интернет, сеть предприятия или локальный компьютер). Это означает, что управляемый компонент может или не может выполнять операции доступа к файлам, операции доступа к реестру или другие важные функции, даже если он используется в одном и том же активном приложении.

Среда выполнения также обеспечивает надежность кода, реализуя инфраструктуру строгой типизации и проверки кода, которую называют системой общих типов (CTS). Система общих типов обеспечивает самоописание всего управляемого кода. Различные языковые компиляторы корпорации Microsoft и независимых изготовителей создают управляемый код, удовлетворяющий системе общих типов. Это означает, что управляемый код может принимать другие управляемые типы и экземпляры, при этом обеспечивая правильность типов и строгую типизацию.

Среда выполнения может размещаться в высокопроизводительных серверных приложениях, таких как Microsoft SQL Server и службы IIS (Internet Information Services). Такая инфраструктура позволяет использовать управляемый код для написания собственной логики программ, пользуясь при этом высочайшей производительностью лучших производственных серверов, которые поддерживают размещение среды выполнения.

Библиотека классов платформы .NET Framework представляет собой коллекцию типов, которые тесно интегрируются со средой CLR. Библиотека классов является объектно-ориентированной. Она предоставляет типы, от которых управляемый код пользователя может наследовать функции. Это не только упрощает работу с типами .NET Framework, но и сокращает время изучения новых средств платформы .NET Framework. Кроме того, компоненты независимых производителей можно легко объединять с классами платформы .NET Framework.

Например, в классах коллекций .NET Framework реализуется набор интерфейсов для разработки пользовательских классов коллекций. Пользовательские классы коллекций легко объединяются с классами .NET Framework.

Как и ожидается от объектно-ориентированной библиотеки классов, типы .NET Framework позволяют решать типовые задачи программирования, включая работу со строками, сбор данных, подключение к базам данных и доступ к файлам. В дополнение к этим обычным задачам библиотека классов содержит типы, поддерживающие многие специализированные сценарии разработки [4].

**Windows Forms** — это интеллектуальная клиентская технология для .NET Framework, набор управляемых библиотек, упрощающих типичные задачи приложения, такие как чтение и запись в файловую систему. При использовании среды разработки, такой как Visual Studio, можно создавать Windows Forms интеллектуальные клиентские приложения, которые отображают сведения, запрашивают ввод данных от пользователей и обмениваются данными с удаленными компьютерами по сети.

В Windows Forms форма — это визуальная поверхность, на которой выводится информация для пользователя. Обычно приложение Windows Forms строится путем помещения элементов управления на форму и написания кода для реагирования на действия пользователя, такие как щелчки мыши или нажатия клавиш. Элемент управления — это отдельный элемент пользовательского интерфейса, предназначенный для отображения или ввода данных.

При выполнении пользователем какого-либо действия с формой или одним из ее элементов управления создается событие. Приложение реагирует на эти события с помощью кода и обрабатывает события при их возникновении.

Windows Forms включает широкий набор элементов управления, которые можно добавлять на формы: текстовые поля, кнопки, раскрывающиеся списки, переключатели и даже веб-страницы. Если существующий элемент управления не удовлетворяет потребностям, в Windows Forms можно создать пользовательские элементы управления с помощью класса [UserControl](https://docs.microsoft.com/ru-ru/dotnet/api/system.windows.forms.usercontrol).

В состав Windows Forms входят многофункциональные элементы пользовательского интерфейса, позволяющие воссоздавать возможности таких сложных приложений, как Microsoft Office. Используя элементы управления [ToolStrip](https://docs.microsoft.com/ru-ru/dotnet/api/system.windows.forms.toolstrip) и [MenuStrip](https://docs.microsoft.com/ru-ru/dotnet/api/system.windows.forms.menustrip), можно создавать панели инструментов и меню, содержащие текст и рисунки, подменю и другие элементы управления, такие как текстовые поля и поля со списками.

С помощью конструктор Windows Forms перетаскивания в Visual Studio можно легко создавать Windows Forms приложения. Достаточно выделить элемент управления курсором и поместить его в нужное место на форме. Для преодоления трудностей, связанных с выравниванием элементов управления, конструктор предоставляет такие средства, как линии сетки и линии привязки. А также при использовании Visual Studio или компиляции из командной строки можно использовать элементы управления [FlowLayoutPanel](https://docs.microsoft.com/ru-ru/dotnet/api/system.windows.forms.flowlayoutpanel), [TableLayoutPanel](https://docs.microsoft.com/ru-ru/dotnet/api/system.windows.forms.tablelayoutpanel) и [SplitContainer](https://docs.microsoft.com/ru-ru/dotnet/api/system.windows.forms.splitcontainer) для создания расширенных макетов форм за меньшее время [5].

**Microsoft SQL Server** является одной из наиболее популярных систем управления базами данных (СУБД) в мире. Данная СУБД подходит для самых различных проектов: от небольших приложений до больших высоконагруженных проектов.

SQL Server был создан компанией Microsoft. Первая версия вышла в 1987 году. А текущей версией является версия 19, которая вышла в 2019 году.

SQL Server долгое время был исключительно системой управления базами данных для Windows, однако начиная с версии 16 эта система доступна и на Linux.

SQL Server характеризуется такими особенностями как:

* производительность. SQL Server работает очень быстро;
* надежность и безопасность. SQL Server предоставляет шифрование данных;
* простота. С данной СУБД относительно легко работать и вести администрирование.

Центральным аспектом в MS SQL Server, как и в любой СУБД, является база данных. База данных представляет хранилище данных, организованных определенным способом. Нередко физически база данных представляет файл на жестком диске, хотя такое соответствие необязательно. Для хранения и администрирования баз данных применяются системы управления базами данных (database management system) или СУБД (DBMS). И как раз MS SQL Server является одной из такой СУБД.

Для организации баз данных MS SQL Server использует реляционную модель. Эта модель баз данных была разработана еще в 1970 году Эдгаром Коддом. А на сегодняшний день она фактически является стандартом для организации баз данных.

Реляционная модель предполагает хранение данных в виде таблиц, каждая из которых состоит из строк и столбцов. Каждая строка хранит отдельный объект, а в столбцах размещаются атрибуты этого объекта.

Для идентификации каждой строки в рамках таблицы применяется первичный ключ (primary key). В качестве первичного ключа может выступать один или несколько столбцов. Используя первичный ключ, мы можем ссылаться на определенную строку в таблице. Соответственно две строки не могут иметь один и тот же первичный ключ.

Через ключи одна таблица может быть связана с другой, то есть между двумя таблицами могут быть организованы связи. А сама таблица может быть представлена в виде отношения ("relation").

Для взаимодействия с базой данных применяется язык SQL (Structured Query Language). Клиент (например, внешняя программа) отправляет запрос на языке SQL посредством специального API. СУБД должным образом интерпретирует и выполняет запрос, а затем посылает клиенту результат выполнения [6].

Существуют так же различные средства SQL, которые предоставляют графический пользовательский интерфейс (GUI) для управления базой данных. Например, для создания сущностей и создания связей между ними в базе данных, мною было использовано средство **SQL Server Management Studio (SSMS)**. Это средство предоставляет управление экземпляром SQL Server или базой данных с полной поддержкой GUI. Возможности доступа, настройки, администрирования и разработки всех компонентов SQL Server, Базы данных SQL Azure и хранилища данных SQL, а также управления ими. Среда SSMS предоставляет единую полнофункциональную служебную программу, которая сочетает в себе обширную группу графических инструментов с рядом отличных редакторов сценариев для доступа к службе SQL Server для разработчиков и администраторов баз данных всех профессиональных уровней [7].

## **Анализ инструментальных средств автоматизации разработки и тестирования**

Rational Rose представляет собой CASE средство проектирования и разработки информационных систем и программного обеспечения для управления предприятиями. Как и другие CASE средства (ARIS, BPwin, ERwin) его можно применять для анализа и моделирования бизнес процессов. Первая версия этого продукта была выпущена компанией Rational Software. В дальнейшем Rational Rose был куплен IBM.

Принципиальное отличие Rational Rose от других средств заключается в объектно-ориентированном подходе. Графические модели, создаваемые с помощью этого средства, основаны на объектно-ориентированных принципах и языке UML (Unified Modeling Language). Инструменты моделирования Rational Rose позволяют разработчикам создавать целостную архитектуру процессов предприятия, сохраняя все взаимосвязи и управляющие воздействия между различными уровнями иерархии.

Моделирование бизнес процессов в Rational Rose выполняется за счет применения различных аспектов. Каждый из этих аспектов концентрирует внимание на определенных характеристиках и возможностях процессов.

К таким аспектам относятся:

* вариант использования (use case). Этот аспект дает возможность понять, каким образом действуют участники процесса и за счет этого определить их взаимодействие и влияние на процесс. Для построения моделей процесса в рамках данного аспекта применяются Use-case диаграммы, диаграммы последовательностей, диаграммы совместной работы и диаграммы действий;
* логический аспект. С помощью этого аспекта можно определить функциональные требования процессов. Он задает логическую взаимосвязь между классами элементов процессов. Для построения моделей применяются диаграммы классов и диаграммы состояний;
* составляющие элементы. Этот аспект обращает внимание на состав элементов процесса и их распределение при создании информационной системы. Модели в этом аспекте строятся с помощью диаграммы компонентов. Она содержит информацию об элементах процесса и программном обеспечении;
* ввод в действие. Этот аспект показывает схему процесса в привязке к аппаратному обеспечению информационной системы. Для построения моделей применяется только одна диаграмма – диаграмма топологии;

Последние версии Rational Rose содержат несколько программных продуктов, которые обеспечивают широкие возможности по моделированию бизнес процессов. Пользователи могут создавать графические модели процессов, приближенные к потребностям бизнеса.

Rational Rose обеспечивает следующие возможности моделирования бизнес процессов:

* поддержка объектного моделирования. Применение принципов объектного моделирования и языка UML позволяет приблизить модели процессов к требованиям бизнеса и упрощает вид моделей;
* структурное представление элементов. Модели процессов и их элементы могут быть представлены в виде графической структуры, наглядно отображающий их состав и взаимосвязи;
* интеграция моделей. За счет применения единого языка UML, Rational Rose позволяет объединить модели бизнес процесса, модели приложений и модели данных;
* интеграция с программными продуктами. Для расширения возможностей моделирования и анализа бизнес процессов в Rational Rose реализована возможность интеграции с другими программными продуктами, например, Microsoft Visual Studio;
* открытая архитектура. Она позволяет дополнять существующий инструментарий программы новыми функциями и возможностями;
* обратное проектирование. Эта возможность позволяет на основе имеющегося программного кода построить понятийную модель. Для целей моделирования бизнес процессов данная возможность может быть полезна, если моделируемый процесс автоматизирован.

Основное преимущество данного CASE средства связано с объектным принципом моделирования. За счет его применения можно максимально сблизить представления различных специалистов, которые осуществляют моделирование бизнес процессов и работают с моделями. Помимо этого, есть и преимущества, обусловленные удобством работы с программным пакетом Rational Rose.

Преимуществами Rational Rose являются:

* поддержка командной работы. В этом CASE средстве реализована простая поддержка всех участников проекта. Пользователи могут работать со своими собственными уникальными моделями и в своем собственном окружении без смены рабочего места, при этом сохраняется взаимосвязь с общими моделями;
* управление моделями. Все создаваемые модели могут быть легко изменены. Изменения в одной модели автоматически отражаются во взаимосвязанных моделях. Для управления моделями применяется система контроля версий и управления конфигурацией. Это позволяет легко проводить изменения в любых моделях бизнес процессов;
* контроль ошибок. Rational Rose обеспечивает отслеживание ошибок, возникающих при моделировании. Это позволяет исправить ошибки с учетом их наследования и передачи на очередной уровень моделирования;
* документирование моделей. Пользователи могут создавать необходимые им отчеты и документы по моделям процессов. Документы формируются под потребности пользователя и могут настраиваться для применения к разным моделям;

управление конфигурацией. Пользователи могут настраивать конфигурацию интерфейса и части приложений под свои потребности. В Rational Rose применяется графический пользовательский интерфейс (GUI), за счет которого можно настроить необходимое окружение для комфортной работы [8].

## **Техническое задание для клиент-серверного программного продукта «Программный комплекс учета лабораторных анализов в лаборатории промышленного предприятия»**

Цель разработки: разработать приложение, предназначенное для автоматизации учёта лабораторных испытаний промышленного предприятия.

Основные функции подлежащие реализации:

* приём и регистрация партии образцов на определенный набор испытаний;
* проведение над партией образцов определенного набора испытаний с фиксацией результатов;
* принятие решения о качестве образцов партии на основе полученных результатов набора испытаний над образцами партии;
* формирование отчётности по качеству выпускаемой продукции.

Приложение должно иметь следующую структуру и функциональность:

* работать под управлением операционной системы Windows;
* иметь принцип работы, схожий с приложениями на предприятиях (наличие разделения доступа к функционалу приложения по ролям, автоматизация учёта лабораторных испытаний);
* предоставлять интерфейс для создания и редактирования набора испытаний; предоставлять отчётность по проведённым испытаниям по выпускаемой продукции.

Создание клиентской части при помощи платформы .NET Framework с использованием Windows Forms. Серверная часть выражена в виде базы данных на Microsoft SQL Server.

# **АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И АЛГОРИТМЫ**

## **Анализ предметной области**

На сегодняшний день ни одно крупное промышленное предприятие не обходится без автоматизированных систем, в том числе систем, позволяющих автоматизировать процессы тестирования продукции на всём протяжении её создания.

Автоматизация позволяет значительно ускорить процессы создания, тестирования продукции на предприятии, а также формирование отчётности.

Ещё одним важным плюсом таких систем является быстрое нахождение сбоя в производстве и его причины.

У большинства таких предприятий есть общие критерии – серийность и плановый выпуск продукции. Это позволяет создать общий макет системы, которая подходит всем таким предприятиям.

## **Алгоритм работы программного комплекса**

Идея работы алгоритма программного комплекса состоит в контроле качества производимой продукции на каждом её этапе. Рамки качества переделов и конечной продукции задаются заказчиком. По итогу производства конечного продукта заказчик должен получить сведения о результатах испытаний конечного продукта и его переделов на всех этапах производства. Так же программный комплекс должен помогать выяснять причину брака.

Для реализации данного алгоритма необходимо составить функциональную и информационную модели программного комплекса. Данные модели описаны ниже.

## **Функциональная модель программного комплекса**

Приложение будет содержать в себе трёхуровневую архитектуру (рисунок 2.1) – клиент-серверная архитектура, в которой разделяются функции представления, обработки и хранения данных.

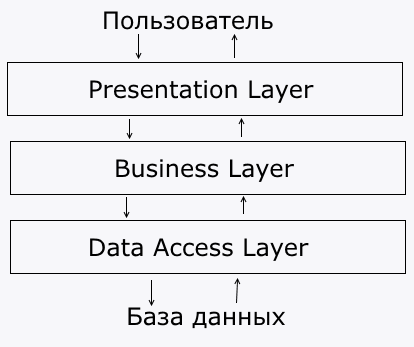


Рисунок 2.1 – Диаграмма архитектуры приложения

Уровень представления напрямую взаимодействует с пользователем. Этот уровень включает методы получения введённых данных пользователем, компоненты пользовательского интерфейса, которые могут быть выражены в виде различных кнопок, текстовых полей, таблиц, и так далее. Далее полученные данные или указания от пользователя передаются уровню бизнес-логики.

Уровень бизнес-логики содержит набор компонентов, которые отвечают за обработку полученных от уровня представлений данных, реализует всю необходимую логику приложения, все вычисления, взаимодействует с базой данных и передает уровню представления результат обработки.

Уровень доступа к данным хранит модели, описывающие используемые сущности, также здесь размещаются специфичные классы для работы с разными технологиями доступа к данным.

Уровень доступа к данным не зависит от других уровней, уровень бизнес-логики зависит от уровня доступа к данным, а уровень представления не зависит от уровня бизнес-логики.

Такой подход, разделения приложения на слои, даст возможность перейти в будущем от десктопного приложения, работающего в сетевой среде предприятия к Web-решениям с единой бизнес-логикой. Слой доступа к данным также позволит рассматривать альтернативные решения без существенной модификации приложения.

Для обеспечения правильной работы всей системы программы доступ к её функционалу следует распределить по ролям. Таким образом явно выделяется 4 роли, которые будут взаимодействовать с программой.

Функционал роли «Контролёр»:

* добавление новой партии продуктов и связывание с ними других существующих продуктов;
* в случае ошибочного добавления какого-либо продукта, возможность редактирования или удаления его;
* просмотр партий продуктов, окончивших проходить испытания;
* принятия решения о качестве партии или отдельных его продуктов;
* просмотр данных в БД.

Функционал роли «Лаборант»:

* добавление записи о проведённом испытании;
* редактирование или удаление информации об проведённых испытаниях над продуктом, который ещё не был утверждён контролёром;
* просмотр данных в БД.

Функционал роли «Заведующий»:

* составление отчётности о качестве выпускаемой продукции;
* просмотр данных в БД.

Функционал роли «Администратор»:

* ведение таблиц в БД;
* просмотр данных в БД.

В качестве среды, для разработки и отображения функциональной структуры программы, будет использована Rational Rose. Эта среда использует язык UML, который предоставляет объектно-ориентированный подход построения диаграмм (рисунок 2.2).

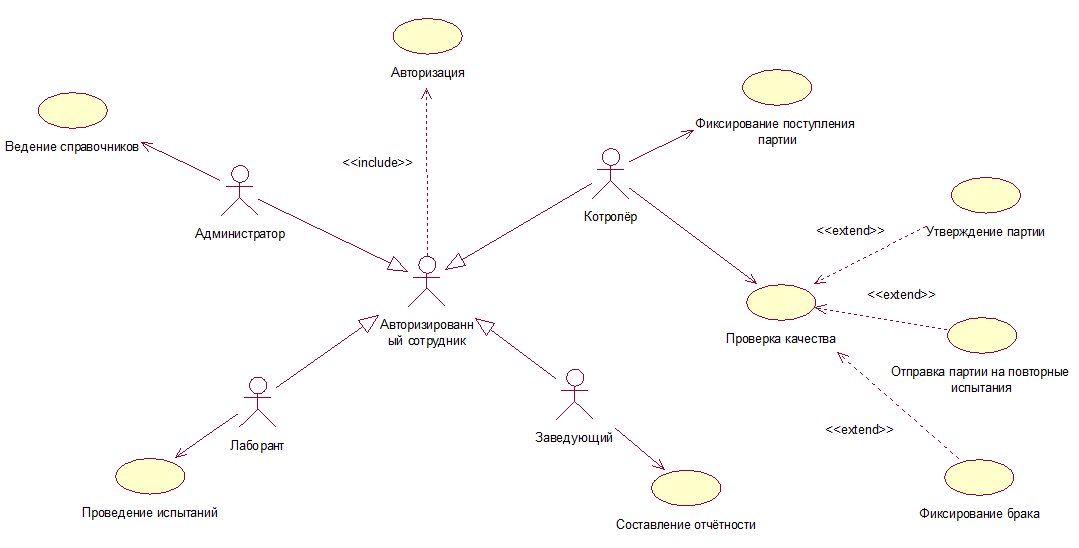


Рисунок 2.2 – Диаграмма прецедентов

## **Информационная модель программного комплекса**

На основе анализа предметной области и решения применить трёхуровневый подход к разработке функциональной модели программного комплекса определим набор сущностей и их свойства, после чего спроектировать базу данных.

Для отображения информационной модели для разрабатываемого продукта были выделены следующие сущности:

* сущность «Цех» хранит в себе информацию о цехах, в которых будет установлено данное ПО;
* сущность «Помещение» хранит в себе информацию о помещениях, в которых работают сотрудники, находится оборудование как для производства, так и для тестирования продукции;
* сущность «Оборудование» хранит в себе информацию об оборудовании, которое позволяет производить и тестировать продукцию;
* сущность «Роль» хранит в себе информацию о всех ролях сотрудников, связанных с производством или тестированием продукции на предприятии и уровень доступа для них;
* сущность «Сотрудник» хранит в себе информацию о каждом сотруднике предприятия, связанных с производством или тестированием продукции;
* сущность «Название шаблона испытания» хранит в себе информацию о типах испытаний, проводимых над выпускаемой продукцией на каждом её этапе производства;
* сущность «Связи оборудования с названием шаблона испытания» хранит в себе информацию о связях сущностей «Оборудование» и «Название шаблона испытания», которая описывает, на каком оборудовании может проводится тот или иной тип испытания;
* сущность «Заказчик» хранит в себе информацию о заказчиках, их контактную информацию;
* сущность «Шаблон продукта» хранит в себе информацию о названии шаблона продукта, которое является основой для фактического продукта;
* сущность «Связи шаблонов продукта» хранит в себе информацию о связях шаблонов продуктов в производственной цепочке;
* сущность «Шаблон испытания» хранит в себе информацию об типе испытания, принадлежность его к шаблону продукта и требования конкретного заказчика;
* сущность «Продукт» хранит в себе информацию о фактическом продукте, который есть или был в производственной цепочке, информацию о рабочем и станке, на котором был создан;
* сущность «Связи продуктов» хранит в себе информацию о связях фактических продуктов в производственной цепочке;
* сущность «Испытание» хранит в себе информацию о полученных результатах в ходе испытания определённого типа над определённым фактическим продуктом.

Физическая модель базы данных определяет способ размещения данных в среде хранения и способы доступа к этим данным, которые поддерживаются на физическом уровне. Построение физической модели базы данных производится на основе логической модели, которая и была представлена выше.

В таблице 2.1 описано соответствие сущностей логической модели и таблиц физической. Подробное описание каждой сущности приведено в таблицах 2.2 – 2.15.

Таблица 2.1 – Соответствие сущностей логической и физической моделей

|  |  |
| --- | --- |
| Сущность | Таблица |
| Цех | Workshop |
| Помещение | Room |
| Оборудование | Equipment |
| Роль | Role |
| Сотрудник | Employee |
| Название шаблона испытания | NameTestTemplate |
| Связи оборудования с названием шаблона испытания | EquipmentNTTRelations |
| Заказчик | Customer |
| Шаблон продукта | ProductTemplate |
| Связи шаблонов продукта | ProductTemplateRelations |
| Шаблон испытания | TestTemplate |
| Продукт | Product |
| Связи продуктов | ProductRelations |
| Испытание | Test |

Таблица 2.2 – Таблица Workshop

| Имя столбца | Тип данных | Разрешено значение NULL | Уникальное значение |
| --- | --- | --- | --- |
| ID | int | false | true |
| NameWorkshop | nvarchar(50) | false | true |

В таблице 2.2 поле NameWorkshop имеет уникальное значение, так как, для большей информативности и безопасности, пользователи будут работать с ним, вместо поля ID.

Таблица 2.3 – Таблица Room

| Имя столбца | Тип данных | Разрешено значение NULL | Уникальное значение |
| --- | --- | --- | --- |
| ID | int | false | true |
| NameRoom | nvarchar(50) | false | true |
| WorkPhone | nchar(20) | true | true |
| WorkshopID | int | false | false |

В таблице 2.3 поля NameRoom, WorkPhone имеют уникальные значения, так как, для большей информативности и безопасности, пользователи будут работать с ними, вместо поля ID.

Таблица 2.4 – Таблица Equipment

| Имя столбца | Тип данных | Разрешено значение NULL | Уникальное значение |
| --- | --- | --- | --- |
| ID | int | false | true |
| NameEquipment | nvarchar(50) | false | false |
| Number | bigint | false | true |
| RoomID | int | false | false |
| Description | nvarchar(50) | true | false |

В таблице 2.4 поле Number имеет уникальное значение, так как, для большей информативности и безопасности, пользователи будут работать с ним, вместо поля ID.

Таблица 2.5 – Таблица Role

| Имя столбца | Тип данных | Разрешено значение NULL | Уникальное значение |
| --- | --- | --- | --- |
| ID | int | false | true |
| NameRole | nvarchar(50) | false | true |
| Code | int | false | true |

В таблице 2.5 поля NameRole, Code имеют уникальные значения, так как, для большей информативности и безопасности, пользователи будут работать с ними, вместо поля ID.

Таблица 2.6 – Таблица Employee

| Имя столбца | Тип данных | Разрешено значение NULL | Уникальное значение |
| --- | --- | --- | --- |
| ID | int | false | true |
| Login | varchar(50) | false | true |
| Password | varchar(50) | false | false |
| RoleID | int | false | false |
| Name | nvarchar(50) | false | false |
| Surname | nvarchar(50) | false | false |
| Patronymic | nvarchar(50) | false | false |
| RoomID | int | false | false |
| Active | int | false | false |
| RegistrationDate | datetime | false | false |

В таблице 2.6 поле Login имеет уникальное значение, так как, для большей информативности и безопасности, пользователи будут работать с ним, вместо поля ID.

Таблица 2.7 – Таблица NameTestTemplate

| Имя столбца | Тип данных | Разрешено значение NULL | Уникальное значение |
| --- | --- | --- | --- |
| ID | int | false | true |
| Name | nvarchar(50) | false | true |

В таблице 2.7 поле Name имеет уникальное значение, так как, для большей информативности и безопасности, пользователи будут работать с ним, вместо поля ID.

Таблица 2.8 – Таблица EquipmentNTTRelations

| Имя столбца | Тип данных | Разрешено значение NULL | Уникальное значение |
| --- | --- | --- | --- |
| ID | int | false | true |
| EquipmentID | int | false | false |
| NameTestTemplateID | int | false | false |

Таблица 2.9 – Таблица Customer

| Имя столбца | Тип данных | Разрешено значение NULL | Уникальное значение |
| --- | --- | --- | --- |
| ID | int | false | true |
| NameCustomer | nvarchar(50) | false | true |
| Email | nvarchar(50) | false | true |
| Phone | nchar(20) | false | true |

В таблице 2.9 поля NameCustomer, Email, Phone имеют уникальные значения, так как, для большей информативности и безопасности, пользователи будут работать с ними, вместо поля ID.

Таблица 2.10 – Таблица ProductTemplate

| Имя столбца | Тип данных | Разрешено значение NULL | Уникальное значение |
| --- | --- | --- | --- |
| ID | int | false | true |
| NameProductTemplate | nvarchar(50) | false | true |

В таблице 2.10 поле NameProductTemplate имеет уникальное значение, так как, для большей информативности и безопасности, пользователи будут работать с ним, вместо поля ID.

Таблица 2.11 – Таблица ProductTemplateRelations

| Имя столбца | Тип данных | Разрешено значение NULL | Уникальное значение |
| --- | --- | --- | --- |
| ID | int | false | true |
| ProductTemplateID | int | false | false |
| NextProductTemplateID | int | false | false |

Таблица 2.12 – Таблица TestTemplate

| Имя столбца | Тип данных | Разрешено значение NULL | Уникальное значение |
| --- | --- | --- | --- |
| ID | int | false | true |
| NameTestTemplateID | int | false | false |
| CustomerID | int | false | false |
| ProductTemplateID | int | false | false |
| MaxValue | float | true | false |
| RecomendValue | float | true | false |
| MinValue | float | true | false |

Таблица 2.13 – Таблица Product

| Имя столбца | Тип данных | Разрешено значение NULL | Уникальное значение |
| --- | --- | --- | --- |
| ID | bigint | false | true |
| ProductTemplateID | int | false | false |
| Number | bigint | false | true |
| NumberBatch | int | false | false |
| EmployeeID | int | false | false |
| EquipmentID | int | false | false |
| CustomerID | int | false | false |
| ProductionDate | datetime | false | false |
| TestStatus | int | false | false |
| ControllerID | int | false | false |

В таблице 2.13 поле Number имеет уникальное значение, так как, для большей информативности и безопасности, пользователи будут работать с ним, вместо поля ID.

Таблица 2.14 – Таблица ProductRelations

| Имя столбца | Тип данных | Разрешено значение NULL | Уникальное значение |
| --- | --- | --- | --- |
| ID | bigint | false | true |
| ProductID | bigint | false | false |
| NextProductID | bigint | false | false |

Таблица 2.15 – Таблица Test

| Имя столбца | Тип данных | Разрешено значение NULL | Уникальное значение |
| --- | --- | --- | --- |
| ID | bigint | false | true |
| TestTemplateID | int | false | false |
| ProductID | bigint | false | false |
| EmployeeID | int | false | false |
| EquipmentID | int | false | false |
| Value | float | false | false |
| TestDate | datetime | false | false |

Соединив связями выше перечисленные сущности получим диаграмму связей сущностей в базе данных (рисунок 2.3).

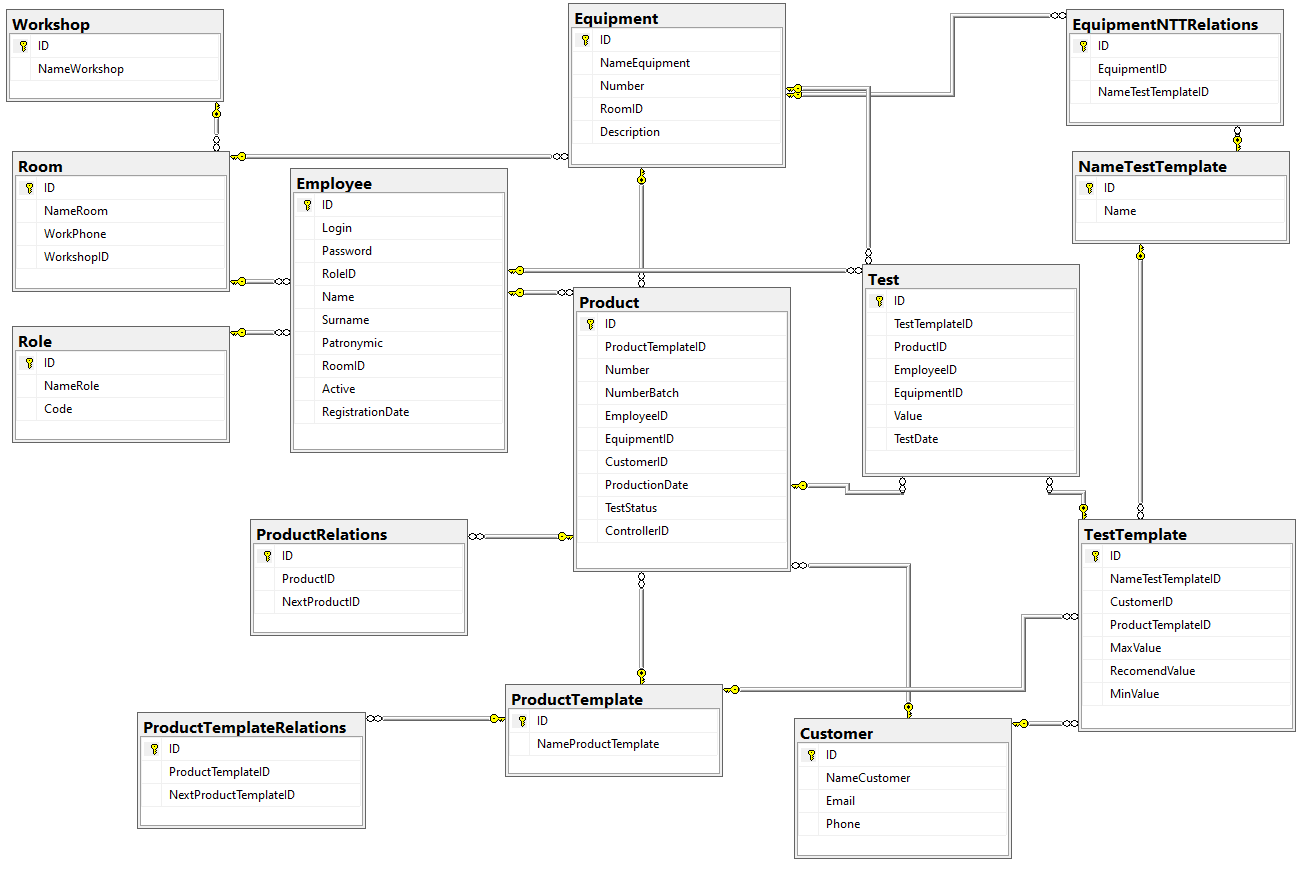


Рисунок 2.3 – Диаграмма связей сущностей в базе данных

Ключевыми сущностями для контроля качества производимой продукции являются:

* Customer – сущность заказчика, от которого зависят записи в сущности TestTempalte;
* ProductTemplate – сущность шаблонов продуктов и переделов, это шаблоны производимой продукции и переделов на производстве;
* NameTestTemplate – сущность типов испытаний, в ней указаны единицы измерения и технология проведения испытания;
* TestTemplate – сущность шаблонов испытаний, которые формируются от предпочтения заказчика, типа испытания и направлены на определённый шаблон продукта;
* Product – сущность переделов и продуктов, хранит в себе переделы и продукты которые уже были созданы на производстве. Архивация и очистка сущности должна производится через определённый промежуток времени, например, в конце каждого года;
* Test – сущность испытаний, которая хранит в себе результаты испытаний, которые были проведены над конкретным продуктом или переделом, ссылается на шаблон испытания.

# **ПРОГРАММНАЯ РЕАЛИЗАЦИЯ КОМПЛЕКСА**

* 1. Общая структура клиентской части приложения

Компоненты клиентской части разделены на три уровня (рисунок 3.1):

* уровень представления включает в себя все формы пользовательского интерфейса и механизмы реагирования на действия пользователя;
* уровень бизнес-логики включает классы, которые отвечают за логику работы всех пользовательских форм и обрабатывают полученные от них данные;
* уровень данных включает в себя модели, которые имеют такую же структуру данных что и сущности в базе данных. Эти модели помогают уровню бизнес-логики работать с данными из базы данных.

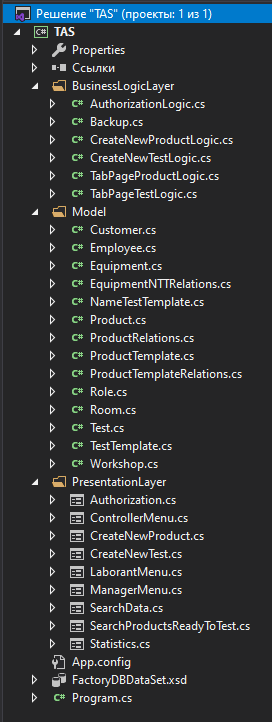


Рисунок 3.1 – Структура уровней в клиентской части программы

* 1. Уровень представления клиентской части приложения

Уровень представления содержит следующие формы:

* Authorization – начальная форма для авторизации пользователя;
* ControllerMenu – форма выбора действия для роли «Контролёр»;
* CreateNewProduct – форма контролёра для корректного фиксирования производства продуктов партии с сохранением данных в базу данных;
* SearchReadyToCheckProducts – форма контролёра для удобного отслеживания продуктов партий, которые завершили полный набор испытаний и готовы к проверке результатов испытаний контролёром;
* LocalStatistics – форма просмотра статистики испытаний по определённому типу продукта и определённому типу испытаний;
* LaborantMenu – форма выбора действий для роли «Лаборант»;
* CreateNewTest – форма лаборанта для корректного выбора типа проводимого испытания над конкретным продуктом из партии с сохранением полученных результатов испытания в базу данных;
* SearchData – форма поиска данных по фильтрам для всех ролей, но ограничивающая внесение контролируемых изменений в данные по ролям;
* ManagerMenu – форма выбора действий для роли «Заведующий»;
* GlobalStatistics – форма просмотра полной статистики испытаний по определённому конечному типу продукта для заказчика;

Форма Authorization (рисунок 3.2) – это начальная форма, которая открывается перед пользователем при запуске программы. Файл Program.cs хранит в себе глобальные значения такие как строка подключения к базе данных, код роли пользователя и его ID (рисунок 3.3). До авторизации пользователь имеет код роли 0 и ID равный -1, что означает, что пользователь пока не имеет роли.

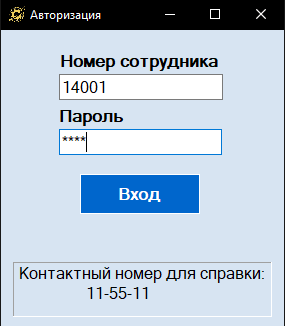


Рисунок 3.2 – вид формы Authorization

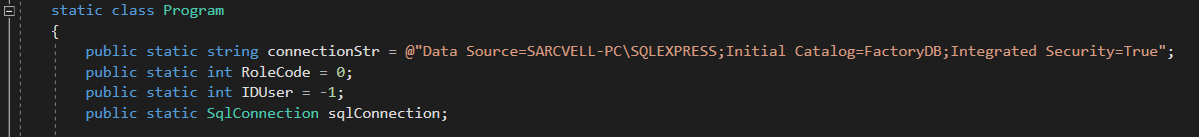


Рисунок 3.3 – вид глобальных переменных в файле Program.cs

Для получения одной из 4-ёх ролей нужно ввести свой логин и пароль. В качестве логина выступает номер сотрудника.

Форма ControllerMenu (рисунок 3.4) открывается пользователю после успешной авторизации в роле «Контролёр». Данная форма включает в себя все необходимые действия для контролёра. При нажатии на кнопку «Создать новый продукт» пользователь перейдёт в форму CreateNewProduct для корректного фиксирования производства продуктов партии с сохранением данных в базу данных. При нажатии на кнопку «Поиск продуктов готовых к проверке» пользователь перейдёт в форму SearchReadyToCheckProducts для удобного отслеживания продуктов партий, которые завершили полный набор испытаний и готовы к проверке результатов испытаний контролёром. При нажатии на кнопку «Поиск» пользователь перейдёт в форму SearchData для поиска данных по фильтрам для всех ролей, но ограничивающая внесение контролируемых изменений в данные по ролям.

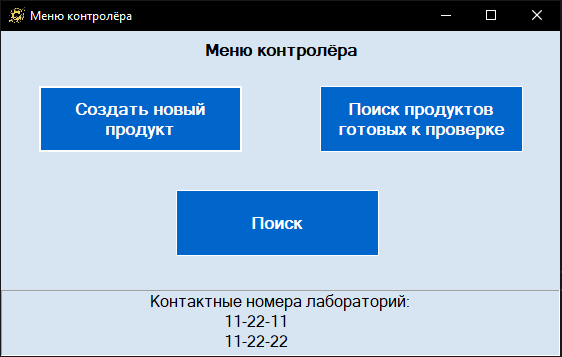


Рисунок 3.4 – вид формы ControllerMenu

Форма CreateNewProduct (рисунок 3.5) сменяет форму ControllerMenu после нажатия на той соответствующей кнопки описанной выше. Данная форма позволяет пользователю под ролью «Контролёр» корректно фиксировать произведённые продукты партии с сохранением данных в базу данных. При загрузке формы, форма обращается к уровню бизнес-логики и в классе TabPageProductLogic вызывает методы GetArrayProductTemplateName и GetArrayNameCustomer, которые обращаются к данным базы данных в виде SQL запроса и возвращают массив названий продуктов и имён заказчиков соответственно. Эти меры необходимы для избегания ошибочного ввода информации. При нажатии на кнопку «Добавить» происходит добавление номера, или как ещё называют кода продукции в список номеров продуктов, которые являлись сырьём или составляющей частью создаваемого продукта. При ошибочном вводе номера продукта можно удалить выделенный номер продукта из списка, нажав на кнопку «Удалить выбранное». При нажатии на кнопку «Автозаполнение» происходит выбор нужного имени заказчика, если все предыдущие продукты, входящий в данный, имели одинакового заказчика. При нажатии кнопки «Сохранить» форма отправляет введённые данные пользователем на уровень бизнес-логики. Далее, после обработки этих данных, уровень бизнес-логики возвращает сообщение с результатом выполнения операции. Ответом такого сообщения может быть, как сообщение об успешном добавлении нового продукта, так и ошибка, с уточняющей информацией.

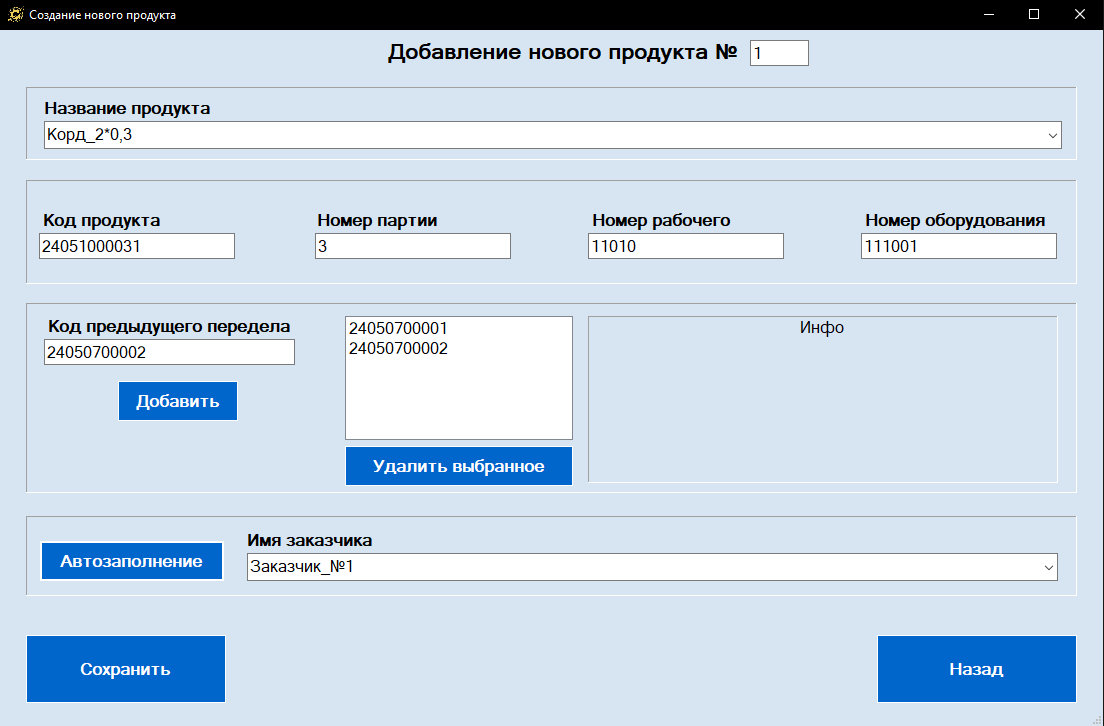


Рисунок 3.5 – вид формы CreateNewProduct

Форма SearchReadyToCheckProducts (рисунок 3.6) сменяет форму ControllerMenu после нажатия на той соответствующей кнопки описанной выше. Эта форма позволяет удобно отслеживать продукты партий, которые завершили полный набор испытаний и готовы к проверке результатов испытаний контролёром. Форма содержит функционал поиска партий продукции, которые завершили полный набор испытаний и готовы к проверке результатов. Данная форма обладает различными фильтрами для удобного поиска.

При загрузке данной формы, происходит обращение к уровню бизнес-логики для загрузки массива названий продуктов и названий испытаний, а также названий заказчиков.

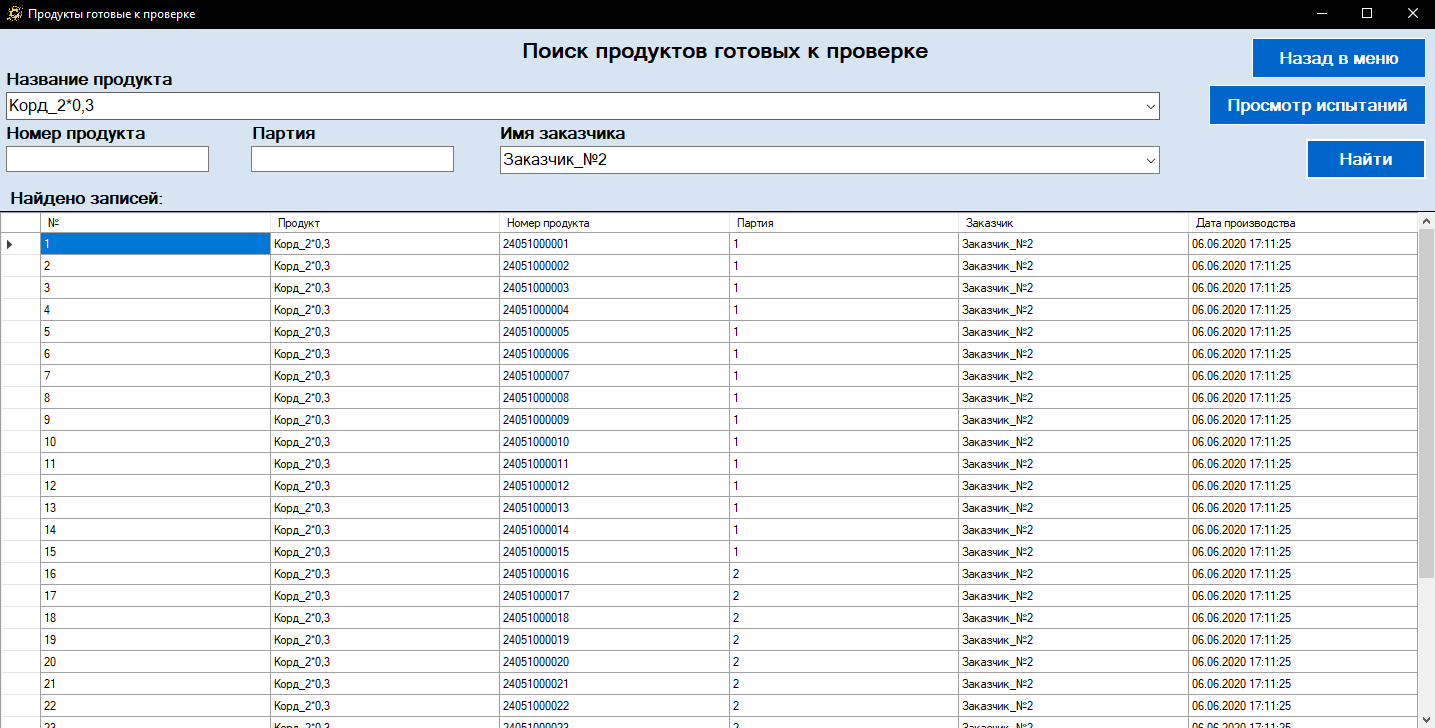


Рисунок 3.6 – вид формы SearchReadyToCheckProducts список продуктов

Форма LocalStatistics (рисунок 3.7) сменяет форму SearchReadyToCheckProducts после нажатия на той соответствующей кнопки. Эта форма позволяет просмотреть статистику испытаний по определённому типу продукта и определённому типу испытаний. Необходимо ввести имя заказчика, нажать кнопку «Определить набор продуктов», выбрать из списка нужный продукт, или ввести сразу номер продукта, над которым будет проводиться испытание. Далее, нажать на кнопку «Определить набор испытаний» и выбрать из списка название испытания. После этого, нужно указать номер партии или диапазон даты производства партий данного типа продуктов. Далее, после обработки выбранных данных, уровень бизнес-логики возвращает список записей, проходящий через пользовательские фильтры. В таблице предоставлена вся необходимая информация, чтобы оценить качество партии.

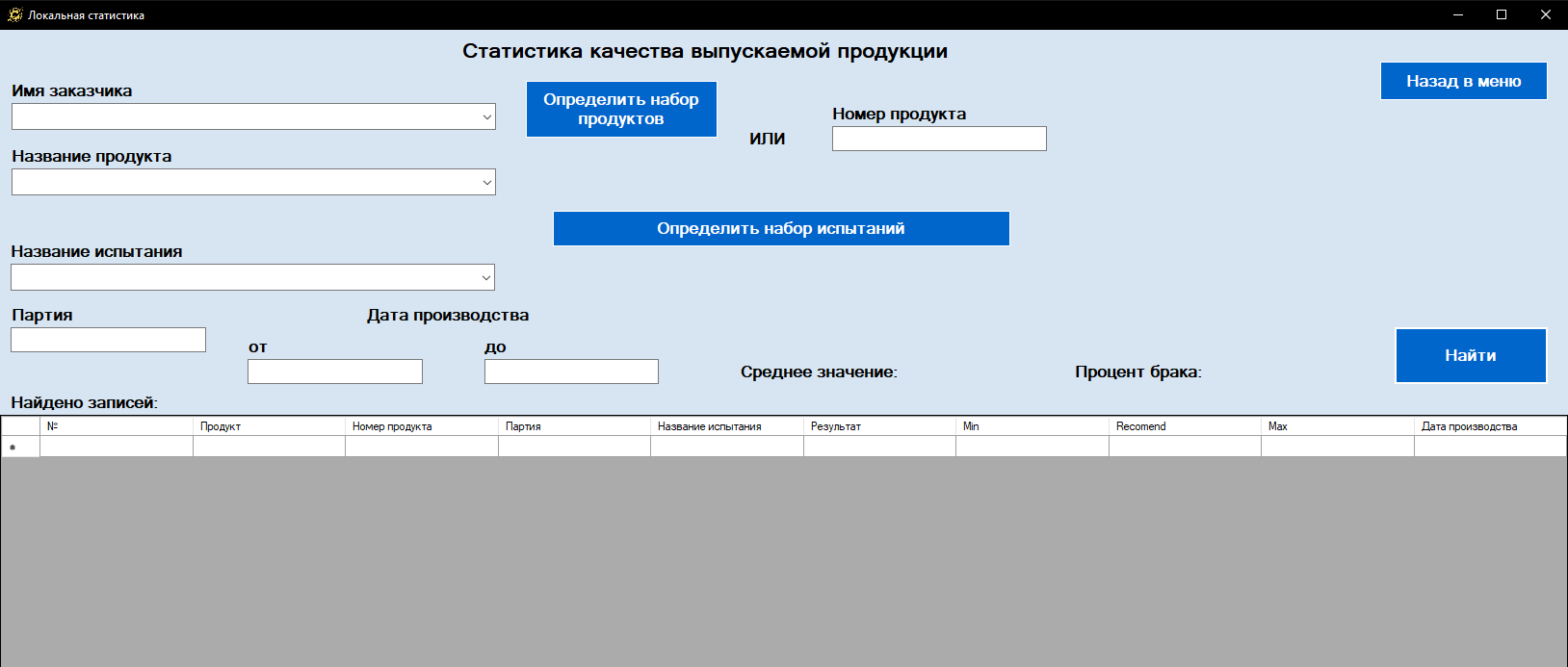


Рисунок 3.7 – вид формы LocalStatistics

Форма LaborantMenu (рисунок 3.8) открывается пользователю после успешной авторизации в роле «Лаборант». Данная форма включает в себя все необходимые действия для лаборанта. При нажатии на кнопку «Записать результаты испытания» пользователь перейдёт в форму CreateNewTest для корректного выбора типа проводимого испытания над конкретным продуктом из партии с сохранением полученных результатов испытания в базу данных.

При нажатии на кнопку «Просмотр статистики по типу испытания» пользователь перейдёт в форму LocalStatistics для просмотра статистики испытаний по определённому типу продукта и определённому типу испытаний. При нажатии на кнопку «Поиск» пользователь перейдёт в форму SearchData для поиска данных по фильтрам для всех ролей, но ограничивающая внесение контролируемых изменений в данные по ролям.

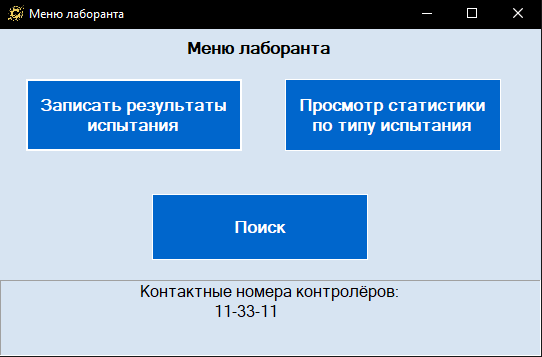


Рисунок 3.8 – вид формы LaborantMenu

Форма CreateNewTest (рисунок 3.9) сменяет форму ControllerMenu после нажатия на той соответствующей кнопки описанной выше. Эта форма позволяет корректно выбрать тип проводимого испытания над конкретным продуктом из партии с сохранением полученных результатов испытания в базу данных. Необходимо ввести номер продукта, над которым будет проводиться испытание. После этого, при нажатии на кнопку «Определить набор типов испытаний», форма обратится к уровню бизнес-логики и получит массив названий испытаний, которые можно провести над введённым продуктом. После проведения испытания пользователем заполняются поля «Номер оборудования» и «Результат испытания». При нажатии кнопки «Сохранить» форма отправляет введённые данные пользователем на уровень бизнес-логики. Далее, после обработки этих данных, уровень бизнес-логики возвращает сообщение с результатом выполнения операции. Ответом такого сообщения может быть, как сообщение об успешном добавлении результатов испытания над продуктом, так и ошибка, с уточняющей информацией.

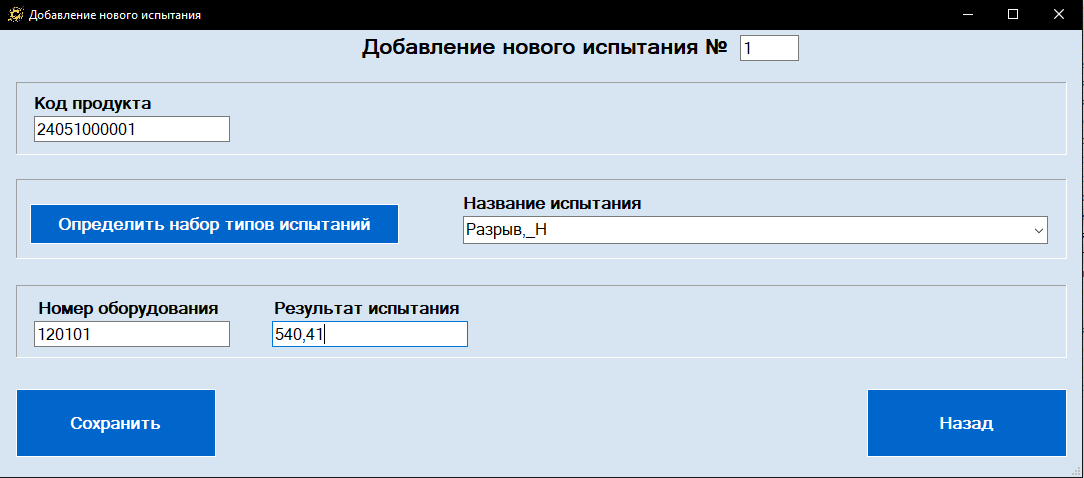


Рисунок 3.9 – вид формы CreateNewTest

Форма SearchData (рисунок 3.10, рисунок 3.11) сменяет форму с меню пользователя на любой роли после нажатия на той соответствующей кнопки описанной выше. Данная форма позволяет производить поиск различных данных по фильтрам для всех ролей, но ограничивающая внесение контролируемых изменений в данные по ролям. Форма содержит элемент tabControl, который позволяет переключаться между поиском различных данных. При загрузке данной формы, происходит обращение к уровню бизнес-логики для загрузки массива названий продуктов, испытаний, заказчиков. Если пользователь открыл страницу поиска под ролью с расширенным доступом для этой страницы, то он увидит дополнительные кнопки для редактирования данных. Для внесения изменений в данных необходимо настроить фильтры поиска и нажать кнопку «Найти», затем, выбрав строку с данными нужной записи нажать на кнопку «Разблокировать». Далее станет возможным внесение изменений в эту строку. После окончания внесения изменений необходимо нажать на кнопку «Сохранить», это вызовет обращение к уровню бизнес-логики, где, проверив внесённые данные пользователем, будет выполнена или отклонена команда пользователя на обновление записи в базе данных и выдано соответствующее сообщение пользователю.

В случае необходимости отмены редактирования записи данных в таблице следует нажать кнопку «Отмена», это вернёт изначальные данные и заблокирует строку в таблице для редактирования вновь. При необходимости удаления записи данных в таблице следует нажать на кнопки «Разблокировать» а затем «Удалить».

Всё взаимодействие с данными в базе данных выполняется через уровень бизнес-логики, где контролируются действия пользователя над данными в базе данных.

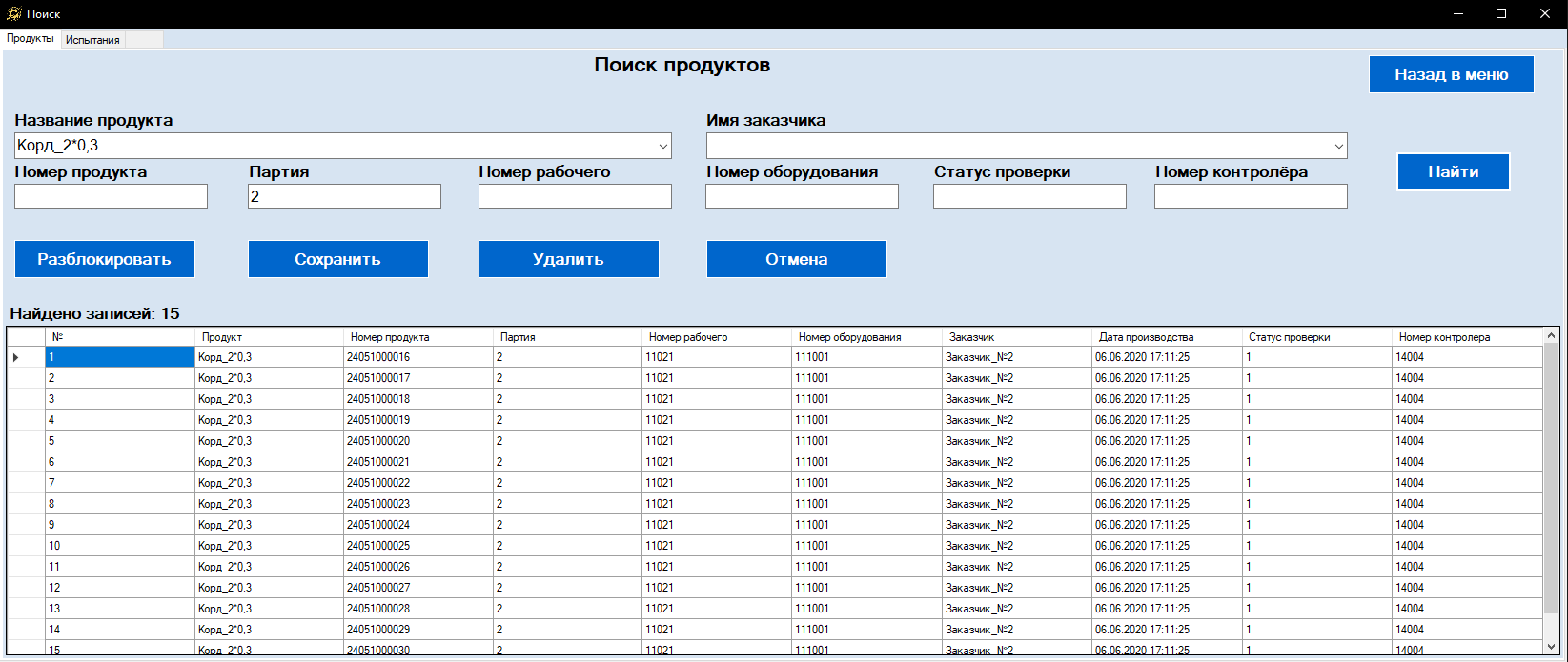


Рисунок 3.10 – вид формы SearchData продукты в роле контролёра

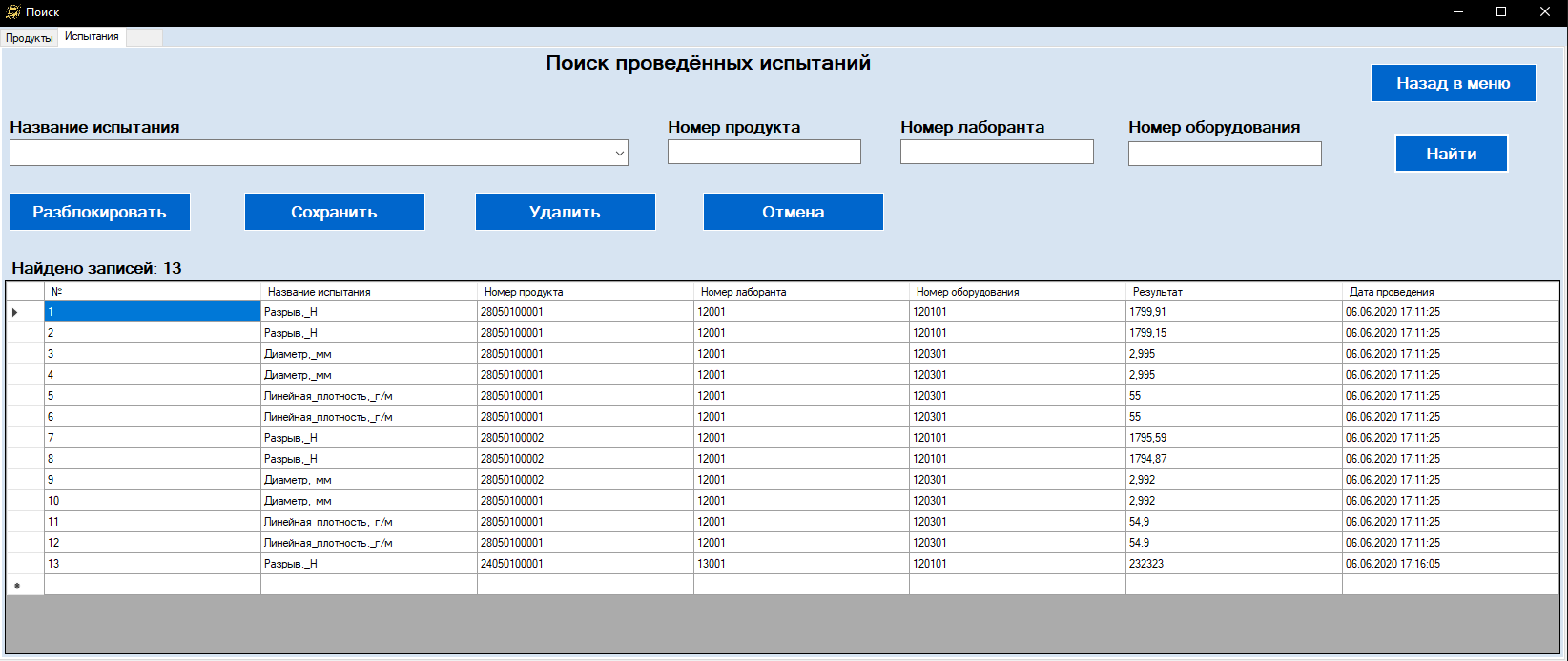


Рисунок 3.11 – вид формы SearchData испытания в роле лаборанта

Форма ManagerMenu (рисунок 3.12) открывается пользователю после успешной авторизации в роле «Заведующий». Данная форма включает в себя все необходимые действия для заведующего. При нажатии на кнопку «Просмотр статистики по типу испытания» пользователь перейдёт в форму LocalStatistics для просмотра статистики испытаний по определённому типу продукта и определённому типу испытаний. При нажатии на кнопку «Просмотр итоговой статистики» пользователь перейдёт в форму GlobalStatistics для просмотра глубокой статистики испытаний по определённому типу продукта для заказчика. При нажатии на кнопку «Поиск» пользователь перейдёт в форму SearchData для поиска данных по фильтрам для всех ролей, но ограничивающая внесение контролируемых изменений в данные по ролям.

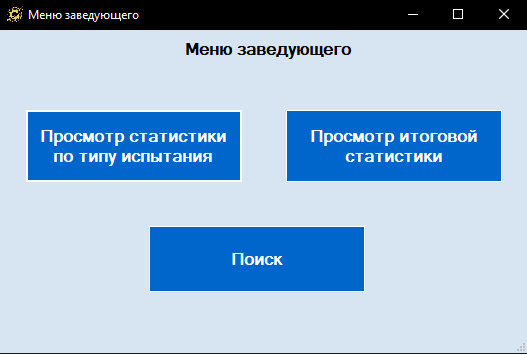


Рисунок 3.12 – вид формы ManagerMenu в роле заведующего

Форма GlobalStatistics (рисунок 3.13) сменяет форму ManagerMenu после нажатия на той соответствующей кнопки. Эта форма позволяет просмотреть статистику испытаний по определённому типу конечного продукта для конкретного заказчика на всех этапах его производства. Для этого необходимо ввести имя заказчика и тип продукта, далее указать номер партии или интервал времени производства партий. Далее, после обработки выбранных данных, уровень бизнес-логики возвращает список записей, проходящий через пользовательские фильтры. В таблице предоставлена вся необходимая информация, чтобы оценить качество партии.

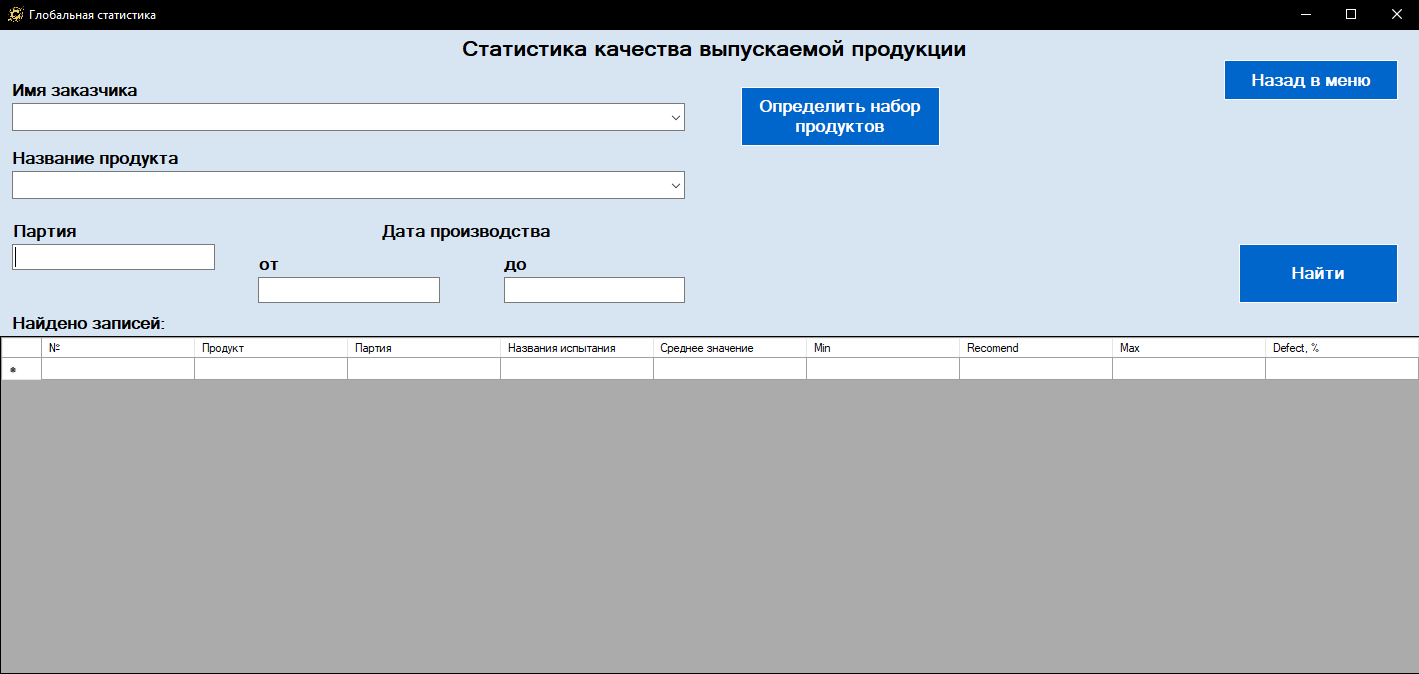


Рисунок 3.13 – вид формы GlobalStatistics

* 1. Уровень бизнес-логики клиентской части приложения

Уровень бизнес-логики содержит следующие классы:

* AuthorizationLogic – класс, отвечающий за механизмы корректной обработки и работы с данными уровня представления в виде формы Authorization, ControllerMenu, LaborantMenu;
* CreateNewProductLogic – класс, отвечающий за механизмы корректной обработки и работы с данными уровня представления в виде формы CreateNewProduct;
* CreateNewTestLogic – класс, отвечающий за механизмы корректной обработки и работы с данными уровня представления в виде формы CreateNewTest;
* TabPageProductLogic – класс, отвечающий за механизмы корректной обработки и работы с данными уровня представления в виде форм SearchData, CreateNewProduct, CreateNewTest, SearchProductReadyToTest;
* TabPageTestLogic – класс, отвечающий за механизмы корректной обработки и работы с данными уровня представления в виде форм SearchData, SearchProductReadyToTest;

Методы классов бизнес-логики приведены в таблицах 3.1-3.5.

Таблица 3.1 – Методы класса AuthorizationLogic

|  |  |
| --- | --- |
| Прототип метода | Описание |
| public static int Login(string login, string password) | Возвращает код номера ошибки при неудачной авторизации, 0 – успешная авторизация |
| public static bool SearchEmployeeIDByLoginANDPassword(string login, string password, Employee employee) | Проверяет, есть ли пользователь в БД с таким логином и паролем |
| public static int GetCodeRoleByID(int idRole) | При нахождении пользователя в БД, возвращает код роли из БД |
| public static List<string> GetPhonesByCodeRole(int codeRole) | Возвращает список номеров телефонов для справки |
| public static int GetRoleIDByCodeRole(int code) | Возвращает ID сущности Role по коду роли |
| public static List<int> GetEmployeeIDByRoleID(int idRole) | Возвращает список ID сотрудников по ID сущности Role |
| public static List<int> GetRoomIDByEmployeeID(List<int> idEmployeeList) | Возвращает список ID помещений по списку ID сущности Employee |

Таблица 3.2 – Методы класса CreateNewProductLogic

|  |  |
| --- | --- |
| Прототип метода | Описание |
| public static bool CheckCorrectNumberPrevProduct(string numberPrevProduct, List<string> listNumberPrevProduct) | Проверяет на корректность добавления номеров продуктов, входящих в создаваемый продукт |
| public static string AutoNameCustomer(List<string> listNumberPrevProduct, string nameProductTemplate) | Возвращает общего заказчика у всех продуктов, входящих в создаваемый продукт |
| public static int CheckNumberPrevProduct(List<int> listProductTemplateID, List<int> listNumberPrevProduct) | Проверяет, верный ли набор номеров продуктов, входящих в создаваемый продукт. Возврат  1- ошибка, 0 – нет ошибки |
| public static int CheckCustomerOnPrevProduct(List<int> listNumberPrevProduct) | Проверяет, есть ли общий заказчик у продуктов, входящих в создаваемый продукт. 1- нету,  0 - есть |
| public static string GetGeneralNameCustomer(string numberPrevProduct) | Возвращает имя общего заказчика среди продуктов, входящих в создаваемый продукт |
| public static string SaveNewProduct(string nameProductTemplate, string numberProduct, string numberBatch,  string numberEpmloyee, string numberEquipment, List<string> listNumberPrevProduct, string nameCustomer) | Основной метод сохранения нового продукта, который вызывает остальные методы в классе для проверок. Возвращает сообщение об успешном завершении операции или ошибке |
| public static bool CheckCorrectOnNumber(string number) | Возвращает результат, можно ли конвертировать введённую строку в тип int или long |
| public static List<int> GetArrayIDPrevProductTemplate(int idProductTemplate) | Возвращает массив ID шаблонов продуктов, которые входят в состав шаблона продукта, передаваемого в качестве атрибута |
| public static void SetRelationsProduct(int numberPrevProduct, int newProductID) | Записывает связи между продуктами в сущность ProductRelations |

Таблица 3.3 – Методы класса CreateNewTestLogic

|  |  |
| --- | --- |
| Прототип метода | Описание |
| public static List<string> AutoNameTestTemplate(string numberProduct) | Возвращает список названий шаблонов испытаний в зависимости от номера продукта |
| public static bool CheckCorrectOnNumber(string number) | Возвращает результат, можно ли конвертировать строку в int или long |
| public static bool CheckCorrectOnDouble(string number) | Возвращает результат, можно ли конвертировать строку в double |
| public static Product GetProductDataByProductID(long id) | Запись данных в модель о продукте по его ID |
| public static List<int> GetNameTestTemplateIDList  ByProductTemplateIDANDCustomerID(int idProductTemplate, int idCustomer) | Получение списка названий типов тестов, которые можно проводить над данным продуктом |
| public static string SaveNewTest(string nameTestTemplate, string numberProduct, string numberEquipment, string valueTest) | Сохраняет введённую запись пользователем с использованием проверяющих на корректность методов. Возвращает сообщение о результате выполнения операции |

Таблица 3.4 – Методы класса TabPageProductLogic

|  |  |
| --- | --- |
| Прототип метода | Описание |
| public static string[] GetArrayProductTemplateName() | Возвращает список названий шаблонов продуктов |
| public static string[] GetArrayNameCustomer() | Возвращает список названий заказчиков |
| public static List<Product> GetProductListByFilters(string nameProductTemplate,  string numberProduct, string numberBatch, string numberEmployee, string numberEquipment, string nameCustomer, string testStatus, string numberController) | Возвращает список продуктов проходящих фильтры пользователя |
| public static bool CheckCorrectOnNumber(string number) | Возвращает результат, может ли строка конвертироваться в int или long |
| public static int GetProductTemplateID  ByNameProductTemplate(string nameProductTemplate) | Возвращает ID шаблона продукта по его имени |
| public static long GetProductIDByNumberProduct(string numberProduct) | Возвращает ID продукта по его номеру |
| public static int GetEmployeeIDByNumberEmployee(string numberEmployee) | Возвращает ID сотрудника по его номеру |
| public static int GetEquipmentIDByNumberEquipment(string numberEquipment) | Возвращает ID оборудования по его номеру |
| public static int GetCustomerIDByNameCustomer(string nameCustomer) | Возвращает ID заказчика по его имени |
| public static string GetNameProductTemplateByProductTemplateID(int idProductTemplate) | Возвращает название шаблона продукта по его ID |
| public static string GetNumberEmployeeByEmployeeID(int idEmployee) | Возвращает номер сотрудника роли «Рабочий» по его ID |
| public static int GetNumberEquipmentByEquipmentID(int idEquipment) | Возвращает номер оборудования по его ID |
| public static string GetNameCustomerByCustomerID(int idCustomer) | Возвращает имя заказчика по его ID |
| public static string GetNumberControllerByControllerID(int idController) | Возвращает номер сотрудника роли «Контролёр» по его ID |
| public static string UpdateProductData(List<string> productData) | Обновляет запись в сущности Product, возвращает сообщение об результате выполнения операции |
| public static void DeleteProductData(string numberProduct) | Удаляет запись из сущности Product по номеру продукта |

Таблица 3.5 – Методы класса TabPageTestLogic

|  |  |
| --- | --- |
| Прототип метода | Описание |
| public static string[] GetArrayNameTestTemplate() | Возвращает список названий шаблонов испытаний |
| public static List<Test> GetTestListByFilters(string nameTestTemplate, string numberProduct, string numberEmployee, string numberEquipment) | Возвращает список записей сущности Test прошедших фильтры пользователя |
| public static bool CheckCorrectOnNumber(string number) | Возвращает результат, может ли строка конвертироваться в тип int или long |
| public static int GetNameTestTemplateID  ByNameTestTemplate(string name) | Возвращает ID названия шаблона испытания по его наименованию |
| public static string GetNameTestTemplate  ByNameTestTemplateID(int id) | Возвращает наименование названия шаблона испытания по его ID |
| public static long GetNumberProductByProductID(long id) | Возвращает номер продукта по его ID |

* 1. Уровень данных в приложении

Уровень данных в клиентской части приложения представлен в виде моделей, через которые происходит корректное считывание данных из базы данных и, при необходимости, занесение в базу данных целого списка записей. Модели названы в соответствии с названием сущностей базы данных (рисунок 3.1). Имеют ту же структуру и типы данных. Нужно отметить, что тип данных double, в языке программирования c#, имеет аналог типа данных в базе данных float, а тип long, аналогичен bigint.

Уровень данных в серверной части представлен в виде базы данных, которая описана во второй главе (рисунок 2.3). Для подключения клиентского приложения к серверу базы данных используются строка подключения в классе Program (рисунок 3.14).

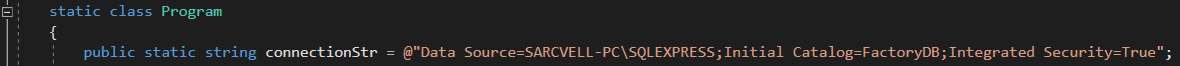


Рисунок 3.14 – строка подключения к базе данных в классе Program

# **ТЕСТИРОВАНИЕ, ВЕРИФИКАЦИЯ И ВАЛИДАЦИЯ ПРОГРАММНОГО КОМПЛЕКСА**

При запуске клиентского приложения пользователю будет предложено авторизоваться (рисунок 4.1). Если пользователь введёт неверный логин (номер сотрудника) или пароль, то он получит сообщение об ошибке (рисунок 4.2). В случае возникновения трудностей со входом внизу формы указан телефон администратора.

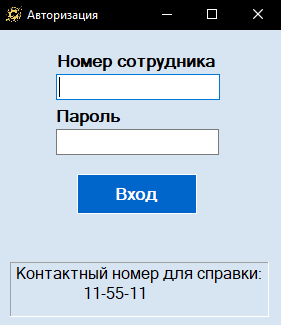


Рисунок 4.1 – форма авторизации

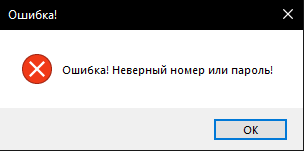


Рисунок 4.2 – сообщение об ошибке авторизации

После успешного входа пользователя в роле «Контролёр» он перейдёт на форму меню для контролёра (рисунок 4.3). В этой форме есть полный набор действий необходимых для работы контролёра. При необходимости связаться с лабораториями испытаний продукции внизу формы указаны рабочие номера.

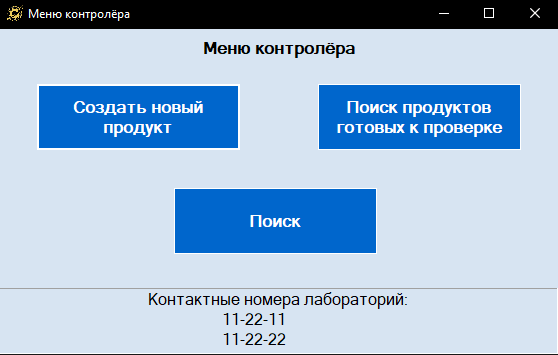


Рисунок 4.3 – форма меню контролёра

После перехода по кнопке «Создать новый продукт» на форму создания продукта (рисунок 4.4), пользователю необходимо заполнить все поля формы. В случае неправильного ввода данных возникают соответствующие сообщения об ошибках (рисунки 4.5-4.10). При успешном создании продукта возникнет соответствующее сообщение (рисунок 4.11).

При выходе из формы через нажатие кнопки «Назад» возникнет диалоговое окно для подтверждения выхода (рисунок 4.12).

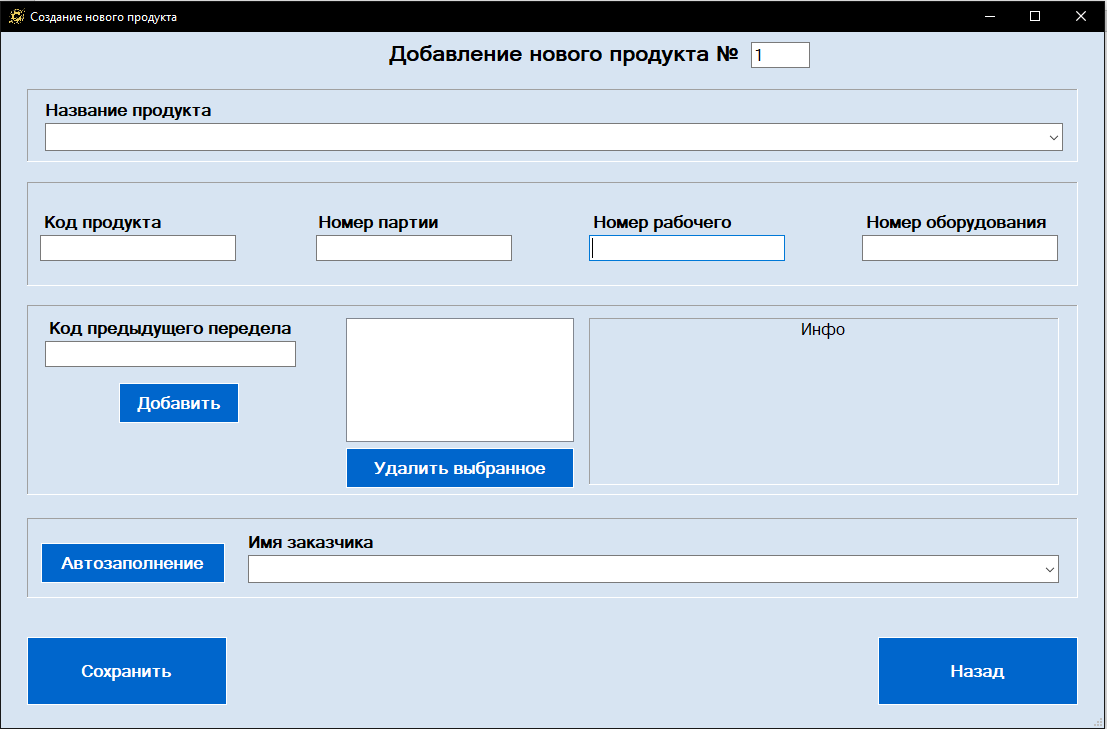
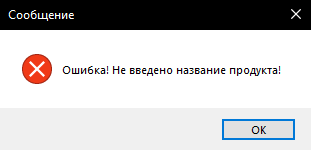
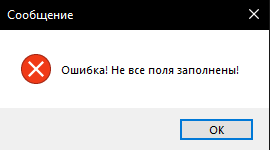
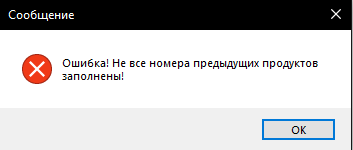
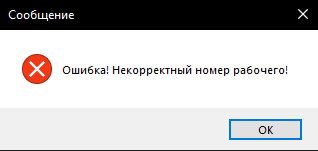


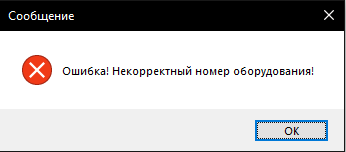
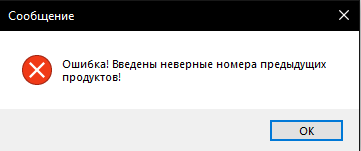
Рисунок 4.4 – форма создания нового продукта

Рисунки 4.5-4.6 – образцы типов сообщений об ошибке

Рисунки 4.7-4.8 – образцы типов сообщений об ошибке

Рисунки 4.9-4.10 – образцы типов сообщений об ошибке

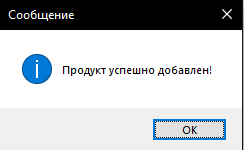


Рисунок 4.11 – вид сообщения об успешном добавлении продукции

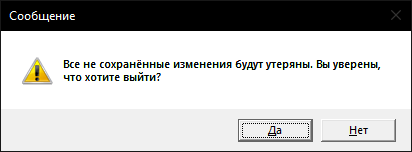


Рисунок 4.11 – вид сообщения об успешном добавлении продукции

Подобный набор проверок на корректность используется во всех остальных пользовательских формах.

В таблице 4.1 подытожим работу основных пользовательских форм программного комплекса и их функционал.

Таблица 4.1 – функционал основных пользовательских форм

|  |  |
| --- | --- |
| Название формы | Выполняемый функционал |
| Authorization | Предоставление набора функционала программного комплекса в зависимости от роли авторизированного пользователя. Безопасность от посторонних. |
| CreateNewProduct | Контролируемое создание записи в сущность Product в роли «Контролёр». |
| CreateNewTest | Контролируемое создание записи в сущность Test в роли «Лаборант». |
| SearchReadyToCheckProducts | Вывод переделов или конечных продуктов, окончивших прохождение набора необходимых испытаний, контролируемое обновление статуса передела или конечного продукта в роли «Контролёр». |
| LocalStatistics | Вывод результатов испытаний по типу продукта и типу испытаний. |
| SearchData | Поиск записей различных сущностей в базе данных по фильтрам с возможностью контролируемого обновления записей по ролям. |
| GlobalStatistics | Вывод полной статистики испытаний конечного продукта и его переделов в роли «Заведующий». |

Из таблицы видно, что весь необходимый функционал был реализован в программном комплексе в полной мере.

1. ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ ДИПЛОМНОЙ РАБОТЫ
   1. Расчёт общей трудоемкости разработки программного обеспечения

Общий объём трудоёмкости разработки системы по автоматизации учёта лабораторных испытаний на промышленном предприятии () определяется исходя из количества и объёма функций, реализуемых программой, по каталогу функций ПО по формуле (5.1):

, (5.1)

где – объем отдельной функции ПО;

– общее число функций.

Анализируя разработанную программу, уточнённый объём ПО () определяем по формуле (5.2):

, (5.2)

где – уточнённый объём отдельной функции ПО в строках исходного кода (LOC).

Сравнение исходного и уточнённых объёмов строк исходного кода представлены в приложении Б.

Разработанное в ходе выполнения дипломной работы приложение относится к третьей категории сложности.

На основании принятого к расчёту (уточнённого) объёма () и категории сложности ПО определяется нормативная трудоемкость ПО () выполняемых работ, которая приведена в таблице 5.1.

Таблица 5.1 – Нормативная трудоёмкость на разработку ПО ()

|  |  |  |
| --- | --- | --- |
| Уточнённый объем, | 3-я категория сложности ПО | Номер нормы |
| 2050 | 91 | 31 |

Дополнительные затраты труда, связанные с повышением сложности разрабатываемого ПО, учитываются посредством коэффициента повышения сложности ПО () (формула 5.3).

*,* (5.3)

где – коэффициент соответствующий степени повышения сложности;

– количество учитываемых характеристик.

Новизна разработанного ПО определяется путем экспертной оценки данных, полученных при сравнении характеристик разрабатываемого ПО с имеющимися аналогами. Влияние фактора новизны на трудоёмкость учитывается путем умножения нормативной трудоёмкости на соответствующий коэффициент, учитывающий новизну ПО (). Разработанная программа обладает категорией новизны A, а значение .

Современные технологии разработки компьютерных программ предусматривают широкое использование коробочных продуктов (пакетов, модулей, объектов). Степень использования в разрабатываемом ПО стандартных модулей определяется их удельным весом в общем объеме ПО.

В данном программном комплексе используется до 20% стандартных модулей, что соответствует значению коэффициента .

Приложение разработано на языке C#, что соответствует коэффициенту функционирования в локальных сетях, учитывающему средства разработки ПО, .

Значения коэффициентов удельных весов трудоёмкости стадий разработки ПО в общей трудоемкости ПО определяются с учётом установленной категории новизны ПО.

При этом сумма значений коэффициентов удельных весов всех стадий в общей трудоёмкости равна единице. Значения коэффициентов приведены в таблице 5.2.

Таблица 5.2 – Значения коэффициентов удельных весов трудоёмкости стадий разработки ПО в общей трудоёмкости ПО

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Категория новизны ПО | Без применения CASE-технологии | | | | |
| Стадии разработки ПО | | | | |
| ТЗ | ЭП | ТП | РП | ВН |
| Значения коэффициентов | | | | |
|  |  |  |  |  |
| А | 0,12 | 0,21 | 0,32 | 0,26 | 0,09 |

Нормативная трудоёмкость ПО () выполняемых работ по стадиям разработки корректируется с учетом коэффициентов: повышения сложности ПО (), учитывающих новизну ПО (), учитывающих степень использования стандартных модулей (), средства разработки ПО ().

Коэффициенты вводятся на всех стадиях разработки, а коэффициент вводится только на стадии РП.

Общая трудоёмкость разработки ПО () определяется суммированием нормативной (скорректированной) трудоёмкости ПО по стадиям разработки формуле (5.4):

, (5.4)

где – нормативная (скорректированная) трудоёмкость разработки ПО на -й стадии (чел/дней);

– количество стадий разработки.

Результаты расчётов по определению нормативной и скорректированной трудоёмкости ПО по стадиям разработки и общую трудоёмкость разработки ПО () представлены в приложении В.

* 1. Расчёт затрат на разработку программного продукта

В состав затрат на разработку системы по автоматизации учёта лабораторных испытаний промышленного предприятия входят следующие статьи расходов:

* затраты труда на создание программного продукта (затраты по основной, дополнительной заработной плате и соответствующие отчисления) ();
* затраты на изготовление эталонного экземпляра ();
* затраты на технологию (затраты на приобретение и освоение программных средств, используемых при разработке программного продукта; затраты на ПО, используемое как эталон) ();
* затраты на машинное время (расходы на содержание и эксплуатацию технических средств разработки, эксплуатации и сопровождения) ();
* затраты на материалы (информационные носители) ();
* затраты на энергию, на использование каналов связи (для отдельных видов);
* общепроизводственные расходы (затраты на управленческий персонал, на содержание помещений) ();
* непроизводственные (коммерческие) расходы (затраты, связанные с рекламой, поиском заказчиков, поставками конкретных экземпляров) ().

В приложении Г приведены значения основных параметров, необходимых для расчёта затрат на разработку программного продукта.

Суммарные затраты на разработку ПО () определяются по формуле (5.5):

, (5.5)

Расходы на оплату труда разработчиков с отчислениями определяются по формуле (5.6):

, (5.6)

где – основная заработная плата разработчиков, руб.;

– дополнительная заработная плата разработчиков, руб.;

– сумма отчислений от заработной платы (социальные нужды, страхование от несчастных случаев), руб.

Основная заработная плата разработчиков рассчитывается по формуле (5.7):

, (5.7)

где Сср\_час – средняя часовая тарифная ставка;

То – общая трудоемкость разработки, чел-час;

Кув – коэффициент, учитывающий доплаты стимулирующего характера.

Средняя часовая тарифная ставка определяется по формуле (5.8):

, (5.8)

где Счi - часовая тарифная ставка разработчика i – й категории;

ni – количество разработчиков i-й категории.

где Tcт – базовая ставка;

k – тарифный коэффициент.

.

.

Дополнительная заработная плата определяется по формуле (5.9):

, (5.9)

где – норматив отчислений на дополнительную заработную плату разработчиков.

Проводим сравнение с минимальной заработной платой (далее – МЗП) (375 руб.).

Рассчитываем оплату труда разработчиков с отчислениями за время выполнения проекта ().

Отчисления от основной и дополнительной заработной платы (отчисления на социальные нужды и обязательное страхование) рассчитываются по формуле (5.10):

, (5.10)

где – процент отчислений на социальные нужды и обязательное страхование от суммы основной и дополнительной заработной платы ().

Затраты машинного времени определяются по формуле (5.11):

, (5.11)

где – стоимость 1 часа машинного времени (руб./ч.);

– коэффициент мультипрограммности, показывающий распределение времени работы ЭВМ в зависимости от количества пользователей ЭВМ;

=1;

– машинное время ЭВМ, необходимое для разработки и отладки проекта (ч.).

Стоимость машино-часа определяется по формуле (5.12):

, (5.12)

где – затраты на заработную плату обслуживающего персонала с учетом всех отчислений, (руб. в год);

– стоимость аренды помещения под размещение вычислительной техники, (руб. в год);

– амортизационные отчисления за год, (руб. в год);

– затраты на электроэнергию, (руб. в год);

– затраты на материалы, необходимые для обеспечения нормальной работы ПЭВМ (вспомогательные), (руб. в год);

– затраты на текущий и профилактический ремонт ЭВМ (руб. в год);

– прочие затраты, связанные с эксплуатацией ПЭВМ (руб. в год);

– действительный фонд времени работы ЭВМ (час/год).

Все статьи затрат формируются в расчете на единицу ПЭВМ.

Затраты на заработную плату обслуживающего персонала () определяются по формуле (5.13):

, (5.13)

,

,

,

где – основная заработная плата обслуживающего персонала, руб.;

– дополнительная заработная плата обслуживающего персонала, руб.;

– сумма отчислений от заработной платы (социальные нужды, страхование от несчастных случаев), руб.;

– количество обслуживаемых ПЭВМ, шт.;

– месячная тарифная ставка *i*-го работника, руб.;

– численность обслуживающего персонала, чел.;

– процент дополнительной заработной платы обслуживающего персонала от основной;

– процент отчислений на социальные нужды и обязательное страхование от суммы основной и дополнительной заработной платы.

Тарифная ставка 5-го разряда обслуживающего персонала:

Годовые затраты на аренду помещения () определяются по формуле (5.14):

, (5.14)

где – средняя годовая ставка арендных платежей, руб./м2;

– площадь помещения, м2;

– количество ПЭВМ, шт.

Сумма годовых амортизационных отчислений () определяется по формуле (5.15):

, (5.15)

где – затраты на приобретение (стоимость) единицы ПЭВМ, руб;

– коэффициент, характеризующий дополнительные затраты, связанные с доставкой, монтажом и наладкой оборудования, от ;

– балансовая стоимость ЭВМ, руб;

– норма амортизации, %.

Стоимость электроэнергии, потребляемой за год, (ЗЭП) определяется по формуле (5.16):

, (5.16)

где М – паспортная мощность ПЭВМ, (кВт), М = 0,09 кВт;

– стоимость одного кВт-часа электроэнергии, руб;

– действительный годовой фонд времени работы ПЭВМ, , согласно производственному календарю на 2020 год.

Затраты на материалы (), необходимые для обеспечения нормальной работы ПЭВМ составляют около 1% от балансовой стоимости ЭВМ и определяются формулой (5.17):

, (5.17)

где – затраты на приобретение (стоимость) ЭВМ,(руб);

– коэффициент, характеризующий дополнительные затраты, связанные с доставкой, монтажом и наладкой оборудования, от ;

– коэффициент, характеризующий затраты на вспомогательные материалы ().

Затраты на текущий и профилактический ремонт () принимаются равными 5% от балансовой стоимости ЭВМ и рассчитываются по формуле (5.18):

, (5.18)

где – коэффициент, характеризующий затраты на текущий и профилактический ремонт ().

Прочие затраты, связанные с эксплуатацией ЭВМ (), состоят из амортизационных отчислений на здания, стоимости услуг сторонних организаций, составляют 5 % от балансовой стоимости и рассчитываются по формуле (5.19):

, (5.19)

где – коэффициент, характеризующий размет прочих затрат, связанных с эксплуатацией ЭВМ ().

Для расчета машинного времени ЭВМ ( в часах), необходимого для разработки и отладки проекта, следует использовать формулу (5.20):

, (5.20)

где – срок реализации стадии «Рабочий проект» (РП), 8 дней;

– срок реализации стадии «Ввод в действие» (ВП), 2 дня;

– продолжительность рабочей смены, (ч.), ;

– количество рабочих смен, .

При написании дипломной работы были использованы Visual Studio 2019 и Microsoft SQL Server поэтому затраты на технологию () и изготовление эталонного экземпляра () будут нулевыми.

Затраты на материалы (носители информации и пр.), необходимые для обеспечения нормальной работы ПЭВМ рассчитываются по формуле (5.21):

, (5.21)

где – затраты на приобретение ЭВМ, руб;

– коэффициент, характеризующий дополнительные затраты, связанные с доставкой, монтажом и наладкой оборудования, от ;

– коэффициент, характеризующий затраты материалы ().

Общепроизводственные затраты рассчитываются по формуле (5.22):

, (5.22)

где – норматив общепроизводственных затрат.

Непроизводственные затраты рассчитываются по формуле (5.23):

, (5.23)

где – норматив непроизводственных затрат.

Итого получаем суммарные затраты на разработку:

Результаты расчетов приведены в приложении Д.

* 1. Формирование цены при создании программного обеспечения

Оптовая цена ПО () определяется следующей формулой (5.24):

, (5.24)

,

где – себестоимость ПО, руб.;

– прибыль от реализации ПО, руб.;

– уровень рентабельности ПО, % ().

Прогнозируемая отпускная цена ПО с НДС рассчитывается по формуле (5.25):

, (5.25)

Налог на добавленную стоимость () рассчитывается по формуле (5.26):

, (5.26)

где – ставка налога на добавленную стоимость, %,

Результаты расчетов формирования цены на разработку программы приведены в таблице 5.3.

Таблица 5.3 – Расчет формирования цены на разработку программы

|  |  |
| --- | --- |
| Наименование статьи расходов | Значение |
| Суммарные затраты на разработку ПО |  |
| Полная себестоимость |  |
| Прибыль от реализации ПО |  |
| Отпускная цена ПО без НДС |  |
| Налог на добавленную стоимость |  |
| Отпускная цена ПО с НДС |  |

* 1. Расчет годового экономического эффекта от использования нового ПО

Для того, чтобы рассчитать годовой экономический эффект от использования нового ПО необходимо такие параметры как заработная плата специалиста, работающего с программой, стоимость 1 часа работы этого специалиста и время, сэкономленное при использовании программы. Все параметры и расчеты приведены в таблице 5.4.

Таблица 5.4 – Расчет годового экономического эффекта от использования нового ПО

|  |  |  |
| --- | --- | --- |
| Показатель | Базовый (до использования ПО) | Новый (при использовании ПО) |
| Время подготовки исходных данных к электронной обработке (ввода в день), мин. | 660 | 60 |
| Дневная экономия времени на ввод и обработку данных, мин. |  | |
| Заработная плата специалиста, работающего с программой в месяц, руб. | 1800 | |
| Стоимость 1 часа работы специалиста, руб. |  | |
| Число пользователей, работающих с программой (например, преподаватель, выдающий студентам задания) | 1 | |
| Годовая экономия рабочего времени на ввод и обработку данных, часов |  | | |
| Годовая экономия средств на оплату труда, руб. |  | | |

Возможность приобретения и использования аналогичного ПО предполагает необходимость ежемесячного обновления программных модулей и ежегодное приобретение лицензий. Таким образом совокупная стоимость использования аналогичного ПО в течении 5 лет составит следующую сумму:

Использование разработанного ПО предполагает единоразовый платеж с последующим автономным использованием, который составит 24606,25 руб.

Таким образом, экономия на цене приобретения в течении 5 лет составит , а 1 года

Далее, на основе полученной годовой экономии от использования ПО, рассчитаем срок окупаемости с учетом экономии в цене приобретения и без неё.

С учетом экономии в цене приобретения срок окупаемости составит:

Без учета экономии в цене приобретения срок окупаемости будет следующим:

* 1. Оценка конкурентоспособности программного продукта

Конкурентоспособность товара – это степень его соответствия выбранному рынку по коммерческим, техническим и экономическим показателям, обеспечивающим возможность сбыта товара на этом рынке. Это те характеристики, которые выгодно отличают данный товар от товаров-конкурентов.

На момент разработки программного комплекса нет аналогов-конкурентов, поэтому не требуется оценка конкурентоспособности.

Все данные приведены в итоговой таблице 5.5.

Таблица 5. – Технико-экономические показатели проекта

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | Наименование показателя | | Единица измерения | | Проектный вариант | |
| Показатели затрат на разработку | | | | | | |
| 2 | Общая трудоемкость разработки ПО | | чел.- дн | | 131 | |
| 3 | Затраты на разработку программы | | руб. | |  | |
| 3.1 | Затраты на оплату труда разработчиков | | руб. | |  | |
| 3.2 | Затраты машинного времени | | руб. | |  | |
| 3.3 | Затраты на изготовление эталонного экземпляра | | руб. | | 0 | |
| 3.4 | Затраты на технологию | | руб. | | 0 | |
| 3.5 | Затраты на материалы | | руб. | |  | |
| 3.6 | Общепроизводственные затраты | | руб. | |  | |
| 3.7 | Непроизводственные (коммерческие) затраты | | руб. | |  | |
| Показатели стоимости | | | | | | |
| 4 | Отпускная цена ПО с НДС | | руб. | |  | |
| Показатели экономической эффективности | | | | | | |
| 5 | | Экономия в цене приобретения в год | | руб. | |  |
| 6 | | Годовая экономия от использования ПО | | руб. | |  |
| 7 | | Простой срок окупаемости | | лет | |  |
| 8 | | Простой срок окупаемости с учётом экономии на приобретении | | лет | |  |

Таким образом, по результатам проведенной мною оценки установлено, что реализация проекта обоснована и является экономически целесообразной. Об этом свидетельствует простой срок окупаемости 0,9 года или с учётом экономии на приобретении 0,91 года.

# **ОХРАНА ТРУДА И ТЕХНИКА БЕЗОПАСНОСТИ**

Согласно Закону Республики Беларусь «Об охране труда», охрана труда – система обеспечения безопасности жизни и здоровья работающих в процессе трудовой деятельности, включающая правовые, социально-экономические, организационные, технические, психофизиологические, санитарно-гигиенические, лечебно-профилактические, реабилитационные и иные мероприятия и средства, где работающие – граждане Республики Беларусь, иностранные граждане и лица без гражданства, работающие по трудовым договорам и (или) гражданско-правовым договорам, на основе членства (участия) в юридических лицах любых организационно-правовых форм, а также привлекаемые к выполнению работ (оказанию услуг) юридическими лицами в порядке и на условиях, установленных законодательством [9].

## **Общие положения инструкции по оказанию первой доврачебной помощи**

Первая доврачебная помощь — это комплекс мероприятий, направленных на восстановление или сохранение жизни и здоровья пострадавшего. Её должен оказывать тот, кто находится рядом с пострадавшим (взаимопомощь), или сам пострадавший (самопомощь) до прибытия медицинского работника. Ответственность за организацию обучения по оказанию первой доврачебной помощи в оздоровительной организации возлагается на руководителя и/или ответственных должностных лиц.

Для того чтобы первая доврачебная помощь была эффективной, в оздоровительной организации должны быть:

* аптечки с набором необходимых медикаментов и медицинских средств для оказания первой доврачебной помощи;
* плакаты, изображающие приемы оказания первой доврачебной помощи пострадавшим при несчастных случаях и проведении искусственного дыхания и наружного массажа сердца.

Оказывающий помощь должен знать основные признаки нарушения жизненно важных функций организма человека, а также уметь освободить пострадавшего от действия опасных и вредных факторов, оценить состояние пострадавшего, определить последовательность применяемых приемов первой доврачебной помощи, при необходимости использовать подручные средства при оказании помощи и транспортировке пострадавшего.

Последовательность действий при оказании первой помощи пострадавшему:

* устранение воздействия на организм пострадавшего опасных и вредных факторов (освобождение его от действия электрического тока, гашение горящей одежды, извлечение из воды и т. д.);
* оценка состояния пострадавшего;
* определение характера травмы, создающей наибольшую угрозу для жизни пострадавшего, и последовательности действий по его спасению;
* выполнение необходимых мероприятий по спасению пострадавшего в порядке срочности (восстановление проходимости дыхательных путей; проведение искусственного дыхания, наружного массажа сердца; остановка кровотечения; иммобилизация места перелома; наложение повязки и т. п.);
* поддержание основных жизненных функций пострадавшего до прибытия медицинского персонала;
* вызов скорой медицинской помощи или врача либо принятие мер для транспортировки пострадавшего в ближайшую медицинскую организацию.

В случае невозможности вызова медицинского персонала на место происшествия необходимо обеспечить транспортировку пострадавшего в ближайшую медицинскую организацию. Перевозить пострадавшего можно только при устойчивом дыхании и пульсе.

В том случае, когда состояние пострадавшего не позволяет его транспортировать, необходимо поддерживать его основные жизненные функции до прибытия медицинского работника [10].

## **Признаки для определения состояния здоровья пострадавшего**

Признаки, по которым можно быстро определить состояние здоровья пострадавшего, следующие:

* сознание: ясное, отсутствует, нарушено (пострадавший заторможен или возбужден);
* цвет кожных покровов и видимых слизистых оболочек (губ, глаз): розовые, синюшные, бледные.
* дыхание: нормальное, отсутствует, нарушено (неправильное, поверхностное, хрипящее);
* пульс на сонных артериях: хорошо определяется (ритм правильный или неправильный), плохо определяется, отсутствует;
* зрачки: расширенные, суженные [10].

## **Комплекс реанимационных мероприятий**

Если у пострадавшего отсутствуют сознание, дыхание, пульс, кожный покров синюшный, а зрачки расширенные, следует немедленно приступить к восстановлению жизненно важных функций организма путем проведения искусственного дыхания и наружного массажа сердца. Требуется заметить время остановки дыхания и кровообращения у пострадавшего, время начала проведения искусственного дыхания и наружного массажа сердца, а также продолжительность реанимационных мероприятий и сообщить эти сведения прибывшему медицинскому персоналу [10].

## **Искусственное дыхание**

         Искусственное дыхание проводится в тех случаях, когда пострадавший не дышит или дышит очень плохо (редко, судорожно, как бы со всхлипыванием), а также если его дыхание постоянно ухудшается независимо от того, чем это вызвано: поражением электрическим током, отравлением, утоплением и т. д. Наиболее эффективным способом искусственного дыхания является способ «изо рта в рот» или «изо рта в нос», так как при этом обеспечивается поступление достаточного объема воздуха в легкие пострадавшего.

         Способ «изо рта в рот» или «изо рта в нос» основан на применении выдыхаемого оказывающим помощь воздуха, который насильно подается в дыхательные пути пострадавшего и физиологически пригоден для дыхания пострадавшего. Воздух можно вдувать через марлю, платок и т. п. Этот способ искусственного дыхания позволяет легко контролировать поступление воздуха в легкие пострадавшего по расширению грудной клетки после вдувания и последующему спаданию ее в результате пассивного выдоха.

         Для проведения искусственного дыхания пострадавшего следует уложить на спину, расстегнуть стесняющую дыхание одежду и обеспечить проходимость верхних дыхательных путей, которые в положении на спине при бессознательном состоянии закрыты запавшим языком. Кроме того, в полости рта может находиться инородное содержимое (рвотные массы, песок, ил, трава и т. п.), которое необходимо удалить указательным пальцем, обернутым платком (тканью) или бинтом, повернув голову пострадавшего набок.

         После этого оказывающий помощь располагается сбоку от головы пострадавшего, одну руку подсовывает под его шею, а ладонью другой руки надавливает на лоб, максимально запрокидывая голову. При этом корень языка поднимается и освобождает вход в гортань, а рот пострадавшего открывается. Оказывающий помощь наклоняется к лицу пострадавшего, делает глубокий вдох открытым ртом, затем полностью плотно охватывает губами открытый рот пострадавшего и делает энергичный выдох, с некоторым усилием вдувая воздух в его рот; одновременно он закрывает нос пострадавшего щекой или пальцами руки, находящейся на лбу. При этом обязательно следует наблюдать за грудной клеткой пострадавшего, которая должна подниматься. Как только грудная клетка поднялась, нагнетание воздуха приостанавливают, оказывающий помощь приподнимает свою голову, происходит пассивный выдох у пострадавшего. Для того чтобы выдох был более глубоким, можно несильным нажатием руки на грудную клетку помочь воздуху выйти из легких пострадавшего.

         Если у пострадавшего хорошо определяется пульс и необходимо проводить только искусственное дыхание, то интервал между искусственными вдохами должен составлять 5 с, что соответствует частоте дыхания 12 раз в минуту.

         Кроме расширения грудной клетки хорошим показателем эффективности искусственного дыхания может служить порозовение кожных покровов и слизистых оболочек, а также выхода пострадавшего из бессознательного состояния и появление у него самостоятельного дыхания.

         При проведении искусственного дыхания оказывающий помощь должен следить за тем, чтобы вдуваемый воздух попадал в легкие, а не в желудок пострадавшего. При попадании воздуха в желудок, о чем свидетельствует вздутие живота «под ложечкой», осторожно надавливают ладонью на живот между грудиной и пупком. При этом может возникнуть рвота, поэтому необходимо повернуть голову и плечи пострадавшего набок (лучше налево), чтобы очистить его рот и глотку.

         Если челюсти пострадавшего плотно стиснуты и открыть рот не удается, следует проводить искусственное дыхание по способу «изо рта в нос».

         Маленьким детям вдувают воздух одновременно в рот и нос. Чем меньше ребенок, тем меньше воздуха нужно ему для вдоха и тем чаще следует производить вдувание по сравнению со взрослым человеком (до 15—18 раз в мин).

         При появлении первых слабых вдохов у пострадавшего следует приурочить проведение искусственного вдоха к моменту начала у него самостоятельного вдоха.

Прекращают искусственное дыхание после восстановления у пострадавшего достаточно глубокого и ритмичного самостоятельного дыхания.

         Нельзя отказываться от оказания помощи пострадавшему и считать его умершим при отсутствии таких признаков жизни, как дыхание или пульс. Делать вывод о смерти пострадавшего имеет право только медицинский работник[10].

## **Наружный массаж сердца**

Показанием к проведению наружного массажа сердца является остановка сердечной деятельности, для которой характерно сочетание следующих признаков: бледность или синюшность кожных покровов, потеря сознания, отсутствие пульса на сонных артериях, прекращение дыхания или судорожные, неправильные вдохи. При остановке сердца, не теряя ни секунды, пострадавшего надо уложить на ровное жесткое основание: скамью, пол, в крайнем случае подложить под спину доску.

Если помощь оказывает один человек, он располагается сбоку от пострадавшего и, наклонившись, делает два быстрых энергичных вдувания (по способу «изо рта в рот» или «изо рта в нос»), затем разгибается, оставаясь на этой же стороне от пострадавшего, ладонь одной руки кладет на нижнюю половину грудины (отступив на два пальца выше от ее нижнего края), а пальцы приподнимает. Ладонь второй руки он кладет поверх первой поперек или вдоль и надавливает, помогая наклоном своего корпуса. Руки при надавливании должны быть выпрямлены в локтевых суставах.

Надавливать следует быстрыми толчками так, чтобы смещать грудину на 4—5 см, продолжительность надавливания не более 0,5 с, интервал между отдельными надавливаниями не более 0,5 с.

В паузах руки с грудины не снимают (если помощь оказывают два человека), пальцы остаются приподнятыми, руки полностью выпрямленными в локтевых суставах.

Если оживление производит один человек, то на каждые два глубоких вдувания (вдоха) он производит 15 надавливаний на грудину, затем снова делает два вдувания и опять повторяет 15 надавливаний и т. д. За минуту необходимо сделать не менее 60 надавливаний и 12 вдуваний, т. е. выполнить 72 манипуляции, поэтому темп реанимационных мероприятий должен быть высоким.

Опыт показывает, что больше всего времени затрачивается на искусственное дыхание. Нельзя затягивать вдувание: как только грудная клетка пострадавшего расширилась, его надо прекращать.

При правильном выполнении наружного массажа сердца каждое надавливание на грудину вызывает появление пульса в артериях.

Оказывающие помощь должны периодически контролировать правильность и эффективность наружного массажа сердца по появлению пульса на сонных или бедренных артериях. При проведении реанимации одним человеком ему следует через каждые 2 мин прерывать массаж сердца на 2-3 сек. для определения пульса на сонной артерии.

Если в реанимации участвуют два человека, то пульс на сонной артерии контролирует тот, кто проводит искусственное дыхание. Появление пульса во время перерыва массажа свидетельствует о восстановлении деятельности сердца (наличии кровообращения). При этом следует немедленно прекратить массаж сердца, но продолжать проведение искусственного дыхания до появления устойчивого самостоятельного дыхания. При отсутствии пульса необходимо продолжать делать массаж сердца.

Искусственное дыхание и наружный массаж сердца необходимо проводить до восстановления устойчивого самостоятельного дыхания и деятельности сердца у пострадавшего или до его передачи медицинскому персоналу.

Длительное отсутствие пульса при появлении других признаков оживления организма (самостоятельное дыхание, сужение зрачков, попытки пострадавшего двигать руками и ногами и др.) служит признаком фибрилляции сердца. В этих случаях необходимо продолжать делать искусственное дыхание и массаж сердца пострадавшему до передачи его медицинскому персоналу [10].

# **ЗАКЛЮЧЕНИЕ**

В ходе выполнения данной дипломной работы были пройдены следующие важные этапы по созданию полноценного программного комплекса:

* аналитический обзор существующих аналогов, современных технологий и сред программирования, который позволил получить необходимую информацию для разработки постановки задачи на проектирование, выделить разработанный продукт среди аналогов;
* анализ предметной области и разработка алгоритмов работы программного комплекса позволили определить необходимые инструменты и технологии в работе, способ хранения данных, архитектуру программного комплекса, его функционал и распределение доступа к нему;
* программная реализация комплекса, в результате завершения которой программный комплекс был разделён на 3 разноуровневых слоя, что дало возможность в будущем изменить способы реализации пользовательского интерфейса и получения данных;
* тестирование программного комплекса, в результате была проверена корректность и стабильность работы программного обеспечения в процессе его использования.

В результате выполнения работы было создано уникальное на данный момент программное обеспечение, реализующее функционал регистрации продукта, фиксирование результатов набора испытаний и вывод отчётности по результатам испытаний. Использование данного программного комплекса позволит контролировать качество выпускаемой продукции на каждом его этапе производства и оперативно выяснять причину брака.

В дальнейшем планируется выполнять поддержку и улучшение функционала разработанного программного обеспечения, а также добавление функционала, позволяющего отслеживать не только качество, но и сроки производства того или иного продукта для каждого заказчика.

Поставленные цели данной дипломной работы были полностью достигнуты путём выполнения всех поставленных задач.

# **СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ**

1. Официальный сайт Индасофт. [Электронный ресурс] – Режим доступа: <https://www.indusoft.ru/media/articles/2115/>.
2. Сайт learn-code.ru. [Электронный ресурс] – Режим доступа: <https://learn-code.ru/yazyki-programmirovaniya/c-sharp>.
3. Сайт open-file.ru. [Электронный ресурс] – Режим доступа: <https://open-file.ru/programs/microsoft-visual-studio>.
4. Сайт docs.microsoft.com. [Электронный ресурс] – Режим доступа: <https://docs.microsoft.com/ru-ru/dotnet/framework/get-started/overview>.
5. Сайт docs.microsoft.com. [Электронный ресурс] – Режим доступа: <https://docs.microsoft.com/ru-ru/dotnet/framework/winforms>.
6. Сайт metanit.com. [Электронный ресурс] – Режим доступа: <https://metanit.com/sql/sqlserver/1.1.php>.
7. Сайт docs.microsoft.com. [Электронный ресурс] – Режим доступа: https://docs.microsoft.com/ru-ru/sql/tools.
8. Сайт kpms.ru. [Электронный ресурс] – Режим доступа: <https://www.kpms.ru/Automatization/Rational_Rose.htm>.
9. Сайт helper.by. [Электронный ресурс] – Режим доступа: <http://www.helper.by/chto-takoe-oxrana-truda-i-v-chem-sostoit-ee-socialno-ekonomicheskoe-znachenie.html>.
10. Сайт dzhmao.ru. [Электронный ресурс] – Режим доступа: <https://www.dzhmao.ru/info/instruktsiya-po-okazaniyu-pervoy-dovrachebnoy-pomoshchi-.php>.

ПРИЛОЖЕНИЕ A

(обязательное)

**Исходный код разработанного приложения**

ПРИЛОЖЕНИЕ Б

(справочное)

**Каталог функций программного обеспечения**

|  |  |  |  |
| --- | --- | --- | --- |
| Код функций | Наименование (содержание) функций | Объем функции строк исходного кода (LOC) | |
| По каталогу () | уточненный () |
| Ввод, анализ входной информации | | | |
| 101 | Организация ввода информации | 130 | 60 |
| 102 | Контроль, предварительная обработка и ввод информации | 490 | 300 |
| 109 | Управление вводом-выводом | 1970 | 50 |
| Формирование, ведение и обслуживание базы данных | | | |
| 206 | Манипулирование данными | 7860 | 440 |
| 207 | Организация поиска и поиск в базе данных | 4720 | 200 |
| Тестирование, проведение тестовых испытаний прикладных программ, вспомогательные программные функции | | | |
| 601 | Проведение тестовых испытаний прикладных программ в интерактивном режиме | 3780 | 1000 |
|  | Итого: | 18950 | 2050 |

ПРИЛОЖЕНИЕ В

(справочное)

**Расчет общей трудоемкости разработки ПО**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Показатели | Стадии разработки | | | | | Итого |
| ТЗ | ЭП | ТП | РП | ВН |  |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Общий объем ПО (), кол-во строк LOC | - | - | - | - | - | 18950 |
| Общий уточненный объем ПО (), кол-во строк LOC | - | - | - | - | - | 2050 |
| Категория сложности разрабатываемого ПО | - | - | - | - | - | 3 |
| Нормативная трудоемкость разработки ПО (), чел./дн. | - | - | - | - | - | 91 |
| Коэффициент повышения сложности ПО () | 1,14 | 1,14 | 1,14 | 1,14 | 1,14 | - |
| Коэффициент, учитывающий новизну ПО () | 1,0 | 1,0 | 1,0 | 1,0 | 1,0 | - |
| Коэффициент, учитывающий степень использования стандартных модулей () | - | - | - | 0,9 | - | 0,9 |
| Коэффициент, учитывающий средства разработки ПО() | 1,3 | 1,3 | 1,3 | 1,3 | 1,3 | - |
| Коэффициенты удельных весов трудоемкости стадий разработки ПО (, , , , ) | 0,12 | 0,21 | 0,32 | 0,26 | 0,09 | 1,0 |
| Распределение скорректированной (с учетом , , , ) трудоемкости ПО по стадиям, чел./дн. | 9 | 16 | 25 | 18 | 7 | - |
| Общая трудоемкость разработки ПО (), чел./дн. | - | - | - | - | - | 131 |

ПРИЛОЖЕНИЕ Г

(справочное)

**Параметры для расчета производственных затрат на разработку ПО**

|  |  |  |
| --- | --- | --- |
| Параметр | Единица измерения | Значение |
| Базовая ставка | руб. | 185 |
| Разряд разработчика | – | 1 |
| Тарифный коэффициент 5-го разряда | – | 1,29 |
| Коэффициент | – | 1,79 |
| Норматив отчислений на доп. зарплату разработчиков () | % | 10 |
| Численность обслуживающего персонала | чел. | 1 |
| Разряд обслуживающего персонала | – | 5 |
| Базовая ставка 5-го разряда | – | 159,63 |
| Коэффициент | – | 2,48 |
| Средняя годовая ставка арендных платежей () | руб./м2 | 195.8 |
| Площадь помещения () | м2 | 9 |
| Количество ПЭВМ () | шт. | 1 |
| Затраты на приобретение единицы ПЭВМ | руб. | 1600 |
| Стоимость одного кВт-часа электроэнергии () | руб. | 0,2909 |
| Затраты на технологию () | руб. | – |
| Норматив общепроизводственных затрат () | % | 50 |
| Норматив непроизводственных затрат () | % | 70 |

ПРИЛОЖЕНИЕ Д

(справочное)

**Расчет суммарных затрат на разработку ПО**

|  |  |
| --- | --- |
| Статья затрат | Итого |
|
| Затраты на оплату труда разработчиков (), руб. |  |
| Затраты машинного времени (), руб. |  |
| Стоимость машино-часа, руб/ч |  |
| Сумма годовых амортизационных отчислений, руб. |  |
| Действительный годовой фонд времени работы ПЭВМ, дн. | 2048 |
| Затраты на текущий и профилактический ремонт, руб. | 50,4 |
| Прочие затраты, связанные с эксплуатацией ЭВМ, руб. | 50,4 |
| Машинное время ЭВМ, ч. | 80 |
| Затраты на изготовление эталонного экземпляра (), руб. | 0 |
| Затраты на технологию (), руб. | 0 |
| Общепроизводственные затраты () |  |
| Непроизводственные (коммерческие) затраты () |  |
| Затраты на материалы (), руб. |  |
| Суммарные затраты на разработку ПО () |  |