МИНИСТЕРСТВО СЕЛЬСКОГО ХОЗЯЙСТВА РФ

ФГБОУ ВО «Кубанский государственный

аграрный университет имени И.Т. Трубилина»

Кафедра информационных систем

**КУРСОВАЯ РАБОТА**

по дисциплине: Методы хранения и анализа данных

на тему: «Разработка архитектуры информационной подсистемы «Получение прогноза процента брака»

выполнила студентка группы ПИ1901

*Хоружая Алина Максимовна*

Допущена к защите\_\_\_\_\_\_\_\_

Руководитель проекта: *к.э.н., доцент Кумратова Альфира Менлигуловна*

Нормоконтролер \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Кумратова А.М.

(подпись, расшифровка подписи)

Защищен \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Оценка\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата)

Члены комиссии \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Краснодар – 2022

МИНИСТЕРСТВО СЕЛЬСКОГО ХОЗЯЙСТВА РФ

ФГБОУ ВО «Кубанский государственный

аграрный университет имени И.Т. Трубилина»

Кафедра информационных систем

**УТВЕРЖДАЮ**

Зав. кафедрой информационных систем

Профессор Попова Е.В.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_» \_\_\_\_\_\_\_\_\_\_2021 г.

**ЗАДАНИЕ**

**на курсовую работу**

Студентке группы ПИ1901 3 курса

Факультета: Прикладной информатики

Специальность: 09.03.03 – Прикладная информатика, направленность

«Менеджмент проектов в области информационных технологий, создание и поддержка информационных систем»

*Хоружей Алине Максимовне*

Тема проекта: «Разработка архитектуры информационной подсистемы «Получение прогноза процента брака»

Содержание задания: «разработать информационную систему получения прогноза процента брака изготавливаемой продукции»

Объем работы: **44** страницы

Рекомендуемая литература: Кумратова А.М. Методы хранения и анализа данных: учеб. пособие / А. М. Кумратова, И. И. Василенко. – Краснодар: КубГАУ, 2021. – 183 с.

Срок выполнения проекта: с «\_\_\_» \_\_\_\_\_\_\_\_\_\_ по «\_\_» \_\_\_\_\_\_\_ 2022 г.

Срок защиты: «\_\_» \_\_\_\_\_\_\_\_ 2022г.

Дата выдачи задания: «\_\_» \_\_\_\_\_\_\_\_\_ 2022 г. Дата сдачи проекта на кафедру: «\_\_\_» \_\_\_\_\_\_\_ 2022 г.

Руководитель проекта \_\_\_\_\_\_\_\_\_\_\_\_\_\_ к.э.н., доцент Кумратова Альфира Менлигуловна

Задание принял студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, дата)

Краснодар – 2022

**РЕФЕРАТ**

Курсовая работа содержит 44 страницы, 31 рисунок, 1 таблицу, 5 используемых источников, 1 приложение.

Ключевые слова: ИНФОРМАЦИОННАЯ СИСТЕМА, ПРОГНОЗ, БАЗА ДАННЫХ, MYSQL, WINDOWS FORM, CRUD.

Цель работы – разработка приложения для получения прогноза % брака изготавливаемой продукции в среде MS Visual Studio 2019 на языке программирования C#.

Предмет исследования – автоматизация процесса получение прогноза.

Основные результаты:

* спроектированы UML – модели по предметной области;
* спроектирована структура и функционал информационной системы;
* реализована информационная система получения прогноза на базе СУБД MySQL WorkBench 8.0 CE;
* разработано руководство пользователя.

**СОДЕРЖАНИЕ**

[**ВВЕДЕНИЕ** 5](#_Toc103638721)

[**1.** **АНАЛИТИЧЕСКАЯ ЧАСТЬ** 6](#_Toc103638722)

[**1.1.** **Экономическая сущность задачи** 6](#_Toc103638723)

[**2.** **Практическая реализация** 16](#_Toc103638724)

[**3.** **Реализация приложения** 23](#_Toc103638725)

[**3.1.1.** **Форма «Главная страница»** 24](#_Toc103638726)

[**3.1.2.** **Форма «ТП»** 25](#_Toc103638727)

[**3.1.3.** **Форма «Добавление ТП»** 27](#_Toc103638728)

[**3.1.4.** **Форма «Удаление ТП»** 29](#_Toc103638729)

[**3.1.5.** **Ключевая форма «Прогнозирование % брака»** 30](#_Toc103638730)

[**Заключение** 33](#_Toc103638731)

[**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ** 34](#_Toc103638732)

[**ПРИЛОЖЕНИЕ 1. ЛИСТИНГ ПРОГРАММЫ** 35](#_Toc103638733)

# **ВВЕДЕНИЕ**

Уровень информационной и технической оснащенности различных предприятий довольно высок, но не у всех корректно работают приложения для работы с ИС.

Для разработки данного приложения для работы с БД было выбрано предприятие ООО «ГлавСтрой Усть-Лабинск», конкретнее – подразделение ОТК и лабораторных исследований. Все данные отдел заносит вручную в бумажные тетради. Такой процесс необходимо цифровизировать. Для этого необходимо создать приложение на основе Windows Forms.

Целью курсовой работы является создание информационной системы для быстрого и легкого получения прогноза % брака.

Для достижения поставленной цели необходимо выполнить следующие задачи:

* проанализировать бизнес-процессы предприятия;
* определиться с функционалом и наполненностью системы;
* разработать БД;
* разработать интерфейс для работы с БД.

# **АНАЛИТИЧЕСКАЯ ЧАСТЬ**

## **Экономическая сущность задачи**

Постановка задачи (описание предметной области). Предприятие осуществляет производство и продажу строительного материала – газобетонного блока. Предприятию необходимо просчитывать процент брака в продукции, чтоб контролировать производственный план. Для описание бизнес-процессов используем концепцию жизненного цикла (ЖЦ) и за основу возьмем язык UML; построим контекстную диаграмму процессов (Use Case Diagram).

Диаграмма должна отвечать на вопрос «что должно делаться в системе, и кто участник этих процессов». Окончательный вид диаграммы вариантов использования показан на рисунке 1.

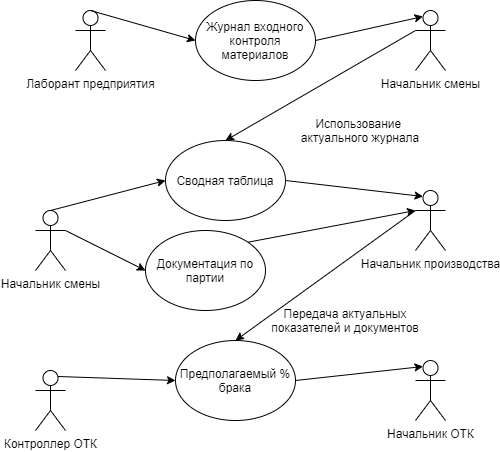


Рисунок 1 - ДВИ задачи получение прогноза % брака

1. Лаборант предприятия проводит опыты над полученным сырьем (вода, песок, глина, цемент, известь, иногда алюминиевая крошка) и заполняет журнал входного контроля материалов.
2. Т.к. предприятие осуществляет деятельность 24\7, лаборант должен отчитаться Начальнику смены о проделанной работе. Далее, начальник смены формирует сводную таблицу и составляет документацию по партии и передает их начальнику производства.
3. На следующем этапе, начальник производства передает всю документацию Контроллёру ОТК (отдел технического контроля), который на основе этих данных, прогнозирует % брака и отчитывается перед начальником ОТК.

Диаграмма вариантов использования и действующих лиц позволяет просто и эффективно представить поставленную перед нами задачу: диаграмма описывает, какой функционал доступен каждой группе пользователей, какие действия они совершают и с какой документацией взаимодействуют.

Теперь построим модель бизнес-процессов, чтоб ответить на вопрос «как это должно делать в системе». В диаграмме взаимодействия действующим лицом будет Начальник ОТК.

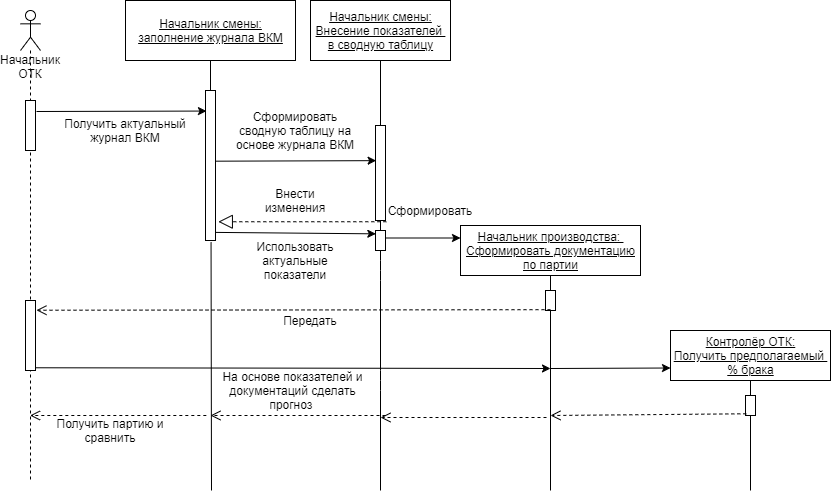


Рисунок 2 - диаграмма последовательности

**Объекты:**

1. Начальник смены: заполнение журнала ВКМ;
2. Начальник смены: внесение показателей в сводную таблицу;
3. Начальник производства: сформировать документацию по партии;
4. Контроллёр ОТК: получить предполагаемый % брака.

**Сообщения:**

1. Получить актуальный журнал ВКМ;
2. Сформировать сводную таблицу на основе журнала ВКМ;
3. Использовать актуальные показатели и сформировать документацию по партии;
4. На основе показателей и документации сделать прогноз.

**Возвращаемые результаты**:

1. Внести изменения;
2. Передать документацию по партии в случае некорректности введенных показателей;
3. Получить партию и сравнить предполагаемый % брака с действительным.

Диаграмма последовательности позволяет более детально описать логики сценариев использования, это способ документирования проекта с точки зрения сценария использования.

Следующий этап – создание корпоративной диаграммы.

Ключевой участник – Начальник ОТК.

1. Начальник ОТК открывает полученный от начальника смены журнал ВКМ;
2. Вместе с начальником смены формирует сводную таблицу;
3. Получает сводную таблицу и проводит анализ;
4. Дополняет сводную таблицу (при необходимости) и формирует вместе с начальником производства документацию по партии;
5. Сформированную документацию сохраняют и утверждают;
6. В случае нахождения ошибок в документации, ее возвращают на 2 этап;
7. Начальник ОТК по номеру партии и документации получает предполагаемый % брака (прогнозирует).

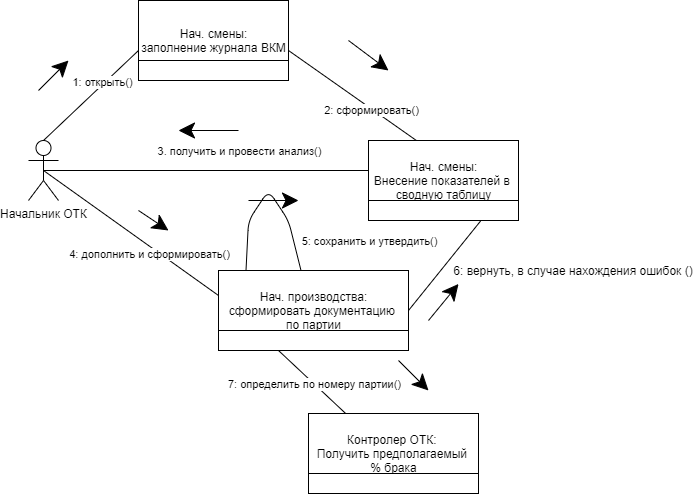
****

Рисунок 3 - кооперативная диаграмма

Кооперативная диаграмма отображает поток событий через конкретный сценарий варианта использования, а также заостряет внимание на связях между объектами.

Далее построим диаграмму Состояний для класса «Получить предполагаемый % брака».

Основной этап (состояние) состоит из 3 шагов:

1. Entry / номер партии
2. Do / получить и изучить журналы ВКМ, сводную таблицу
3. Do / получить и изучить документацию по партии

Состояние - осуществляем анализ полученной информации и переходим к формированию прогноза.

Далее возможны 2 сценария развития событий:

1. Состояние - Прогноз получен
2. Состояние - Прогноз не получен. Тогда исправляются ошибки, вносятся изменения и документация повторно подается на анализ и прогнозирование.

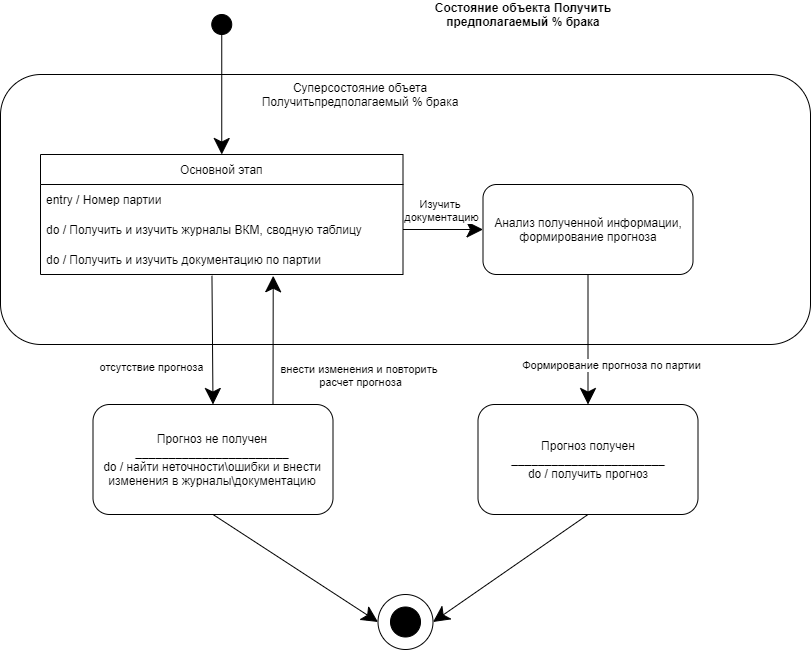
****

Рисунок 4 - диаграмма Состояние объекта "Получить предполагаемый % брака"

Диаграмма состояний показывает, как объект переходит из одного состояния в другое. Также можно отследить ЖЦ объекта. От других диаграмм эта диаграмма отличается тес, что описывает процесс изменения состояний экземпляра, т.е. характеризуется его реакцией на внешние события.

Теперь построим диаграмму активности для варианта использования «Получить прогноз % брака».

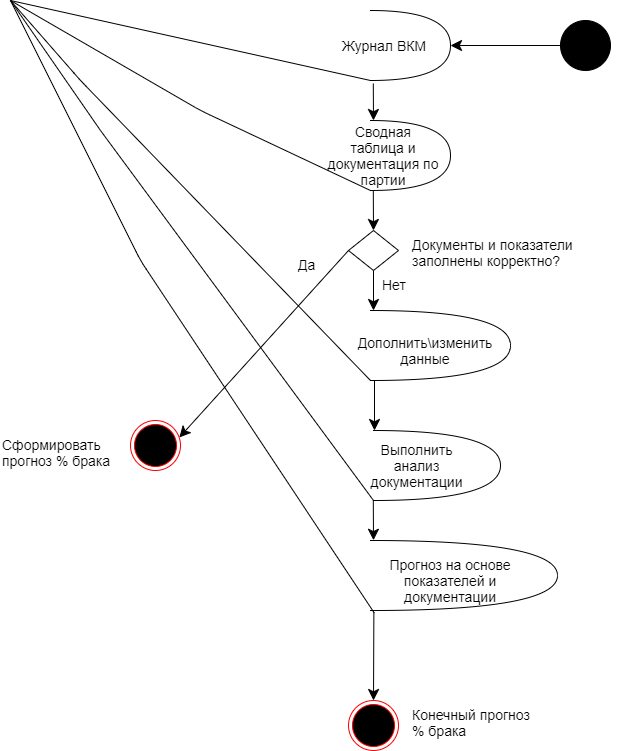


Рисунок 5 - диаграмма активности для варианта использования "Получить прогноз % брака"

1. Журнал ВКМ
2. Сводная таблица и документация по партии
3. Документы и показатели имеются? «ДА» - 1 активность - сформировать прогноз % брака (Exit); «НЕТ» - дополнить\изменить данные
4. Выполнить анализ документации
5. Прогноз на основе показателей и документации
6. 2 активность - Конечный прогноз % брака (Exit)

Следующий этап – добавление классов и пакетов.

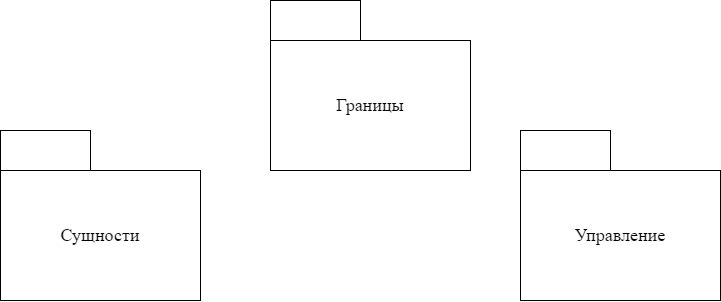
****

Рисунок 6 - диаграмма пакетов

**Сущности:** Заполнение сводной таблицы; Прогноз % брака.

**Границы:** Заполнение журнала ВКМ.

**Управление:** Документация по партии; Внесение правок в документацию.

Далее произведем уточнение методов свойств и классов и создадим описание связей между классами.



Рисунок 7 - диаграмма описания связей между классами

1. Для класса «заполнение журнала ВКМ» введу следующие сигнатуры: открыть()

сформировать()

1. Для класса «заполнение сводной таблицы» введу следующие сигнатуры:

Получить()

Открыть()

Проанализировать()

Сформировать()

1. Проведу ассоциацию от класса «заполнение журнала ВКМ» к классу «заполнение сводной таблицы» 1 к 1
2. Для класса «документации по партии» введу следующие сигнатуры:

Получить()

Дополнить и сформировать()

Сохранить и утвердить()

Вернуть, при ошибках()

1. Проведу ассоциацию от класса «заполнение сводной таблицы» к классу «документация по партии» 1 к 1
2. Для класса «внесение правок в документацию» введу следующие сигнатуры:

Получить()

Открыть()

Исправить()

Сформировать()

1. Проведу ассоциацию от класса «документация по партии» к классу «внесение правок в документацию» Zero or One (0.1)
2. Проведу ассоциацию от класса «внесение правок в документацию» к классу «заполнение сводной таблицы» Zero or One (0.1)
3. Для класса «Прогноз % брака» введу следующие сигнатуры»

Получить()

Определить по номеру партии()

1. Проведу ассоциацию от класса «документация по партии» к классу «прогноз % брака» 1:M (1 ко многим)

Важным моментом является создание диаграмм с исключением кириллизованного текста в информации классов (см. рис. 8 и 9)

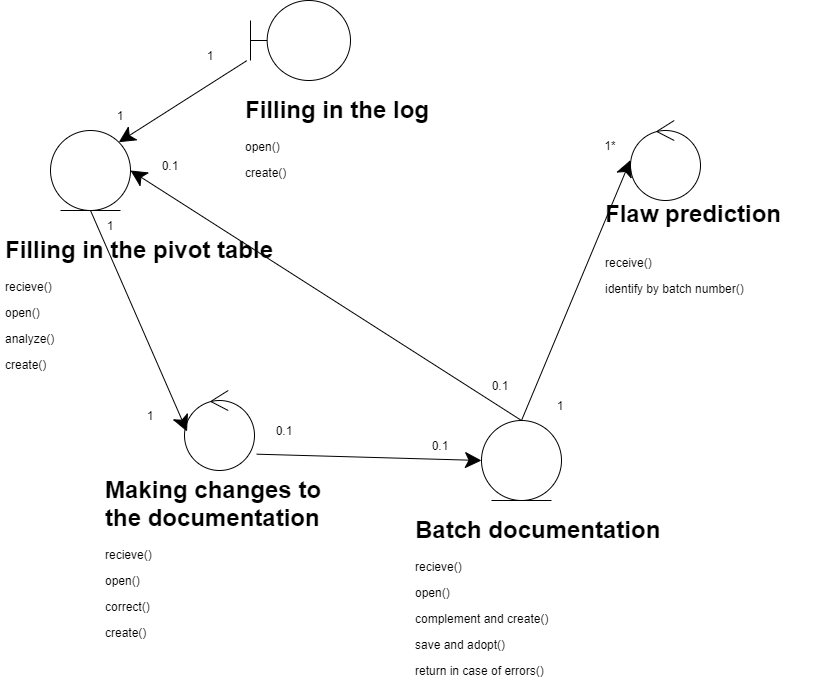
****

Рисунок 8 - диаграмма исключения кириллизованного текста

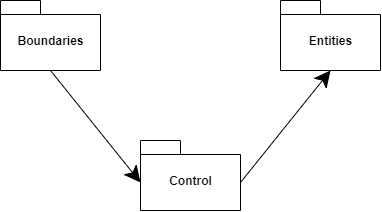


Рисунок 9 - диаграмма компонентов

**Связь компонентов с классами:**

**Entities:**

Filling in the pivot table

Flaw prediction

**Boundaries:**

Filling in the log

**Control:**

Batch documentation

Making changes to the documentation

# **Практическая реализация**

* 1. Выбор инструментальной среды разработки

В качестве СУБД была выбрана MySQL 8.0, которая является одной из самых популярных СУБД на сегодняшний день является. Эта серверная система проста в освоении. К тому же, в интернете без каких-либо усилий можно найти руководства по освоению системы, плагины и расширения, которые помогут освоиться в этой среде.

Следует отметить следующие преимущества:

1. Простота в использовании.
2. Обширный функционал.
3. Безопасность.
4. Масштабируемость. Можно использовать для работы с малыми и большими объемами данных.
5. Скорость.

Приложение ведения БД разработано в Visual Studio 2019 (самая актуальная версия среды).

Microsoft Visual Studio имеет поддержку технологии Windows Forms, которая значительно упрощает типичные задачи приложения (чтение и запись в файловую систему). При использовании среды VS можно создавать WF клиентские приложения, которые будут отображать сведения, запрашивать данные от пользователя. Среда включает в себя редактор исходного кода с поддержкой технологии IntelliSense (автодополнение) и возможностью простейшего рефакторинга (процесс изменения внутренней структуры программы) кода. Встроенный отладчик на 2 уровнях: исходного кода и машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных.

Приложение написано на объектно-ориентированном языке программирования - C#, который был разработан в 1998-2001 годах, как язык разработки приложений для платформы Microsoft .NET Framework. В дальнейшем был стандартизирован. По синтаксису он близок к языку С++, который лежал в основе изучения курса АиООП в течение всего первого курса. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, свойства, обобщённые типы и методы, итераторы. Переняв многое от своих предшественников — языков C++, Delphi, Модула, Smalltalk и, в особенности, Java — С#, опираясь на практику их использования, исключает некоторые модели, зарекомендовавшие себя как проблематичные при разработке программных систем, например, C# в отличие от C++ не поддерживает множественное наследование классов (между тем допускается множественное наследование интерфейсов).

* 1. Логическое проектирование

Создадим визуальное представление о всей информационной системе, чтобы показать типы данных, которые используются и хранятся в системе, отношения между этими типами данных, способы группировки и организации данных, их форматы и атрибуты.

Для этого определим сущности, ключевые свойства каждой сущности, связи между сущностями, сопоставим атрибуты, назначим ключи и определим степень нормализации и проверим модель данных (см. рис. ).

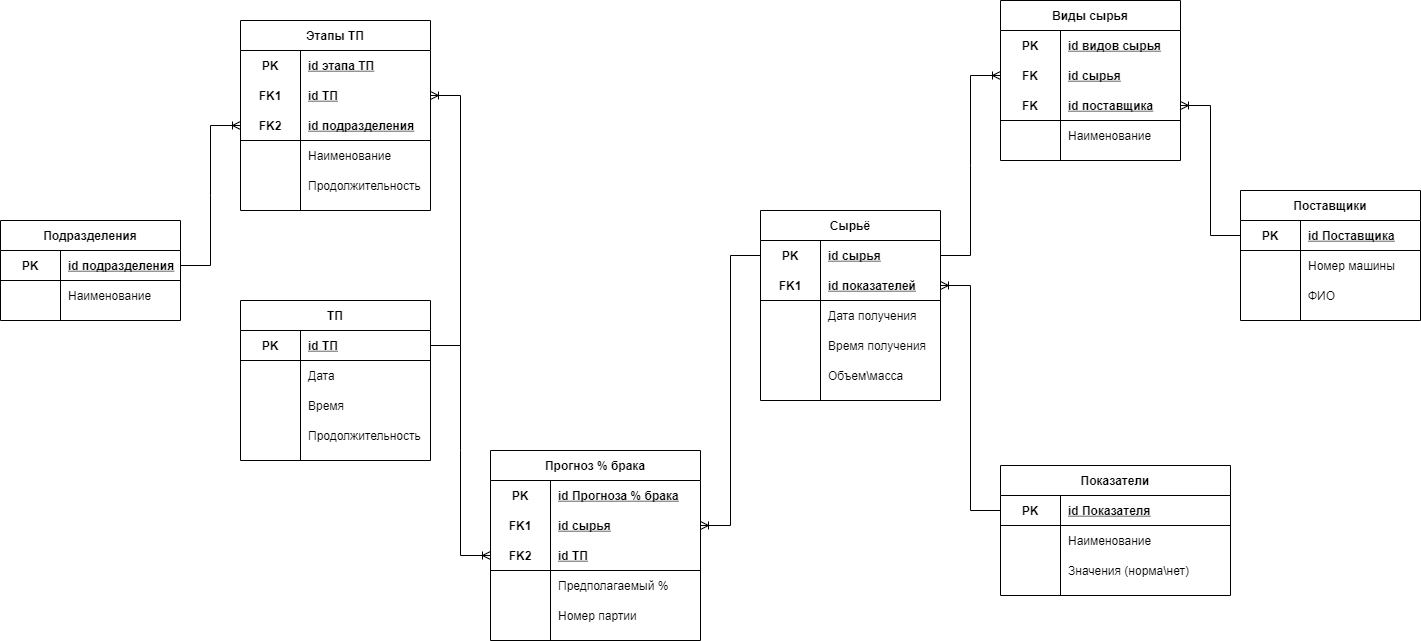


Рисунок 10 - инфологическая модель

Теперь представим модель в табличной форме.

Таблица 1. Даталогическая модель

|  |  |  |  |
| --- | --- | --- | --- |
| Сущность | Имя атрибута | Тип значения | Связь |
| devisions (подразделения) | iddevisions | INTEGER | stages\_tp |
| name | TEXT |  |
| indicators (показатели) | idindicators | INTEGER | raw |
| name | TEXT |  |
| normal | TINYINT(1) |  |
| prediction\_percent\_defect (прогнозирование % брака) | idpretdiction\_percent\_defect | INTEGER |  |
| idraw | INEGER | raw |
| idtp | INTEGER |  |
| assumed\_pecent | FLOAT |  |
| batch\_number | INTEGER |  |
| provider (поставщик) | idprovider | INEGER | types\_of\_raw |
| number\_car | TEXT |  |
| FIO | TEXT |  |
| raw (сырьё) | idraw | INTEGER | prediction\_percent\_defect |
| idindicators | INTEGER | indicators |
| date\_of\_receipt | DATE() |  |
| time\_of\_receipt | TIME() |  |
| Volume\_weight | INTEGER |  |
| stages\_tp (стадии ТП) | idstages\_tp | INTEGER |  |
| idtp | INTEGER | tp |
| iddevisions | INTEGER | devisions |
| name | TEXT |  |
| duration\_min | INTEGER |  |
| tp (ТП – технологический процесс) | idtp | INTEGER | stages\_tp |
| number | INTEGER |  |
| date | DATE() |  |
| time | TIME() |  |
| duration\_min | INTEGER |  |

* 1. Проектирование базы данных

В системе MySQL сформирована база данных – prediction\_percent, содержащая в себе 8 основных таблиц с данными: devisions (подразделения), indicators (показатели), prediction\_percent\_defect (прогноз % брака), provider (поставщики), raw (сырье), stages\_tp (стадии ТП), tp (ТП), types\_of\_raw (типы сырья).

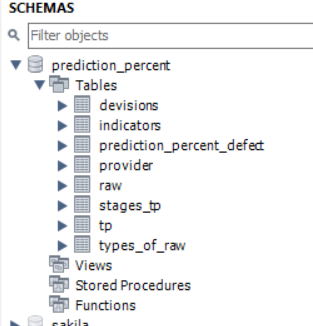


Рисунок 11 - таблицы БД

Таблица Подразделения состоит из 2 полей: iddevisions (id подразделения) PK, name\_devisions (название подразделения).

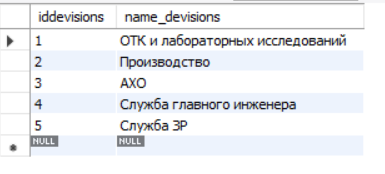


Рисунок 12 - таблица Подразделения

Таблица Показатели состоит из 3 полей: idindicators (id показателя) PK, name\_i (наименование показателя), normal (состояние: нормальное или нет)

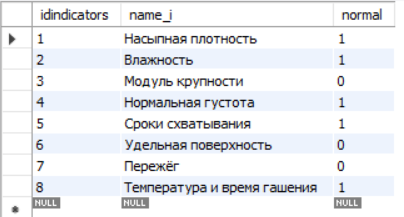


Рисунок 13 - таблица Показатели

Таблица Прогнозирование % брака является самой главной таблицей, которая и определяет связь M:M. Состоит из 5 полей: idprediction\_percent\_defect (id прогнозирования % брака) PK, idraw (id сырья) FK1, idtp (id технологического процесса) FK2, assumed\_percent (предположительный процент), batch\_number (номер партии).

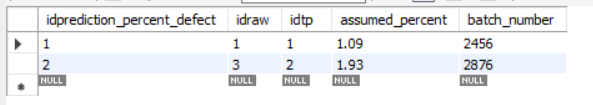


Рисунок 14 - прогнозирование процента брака

Таблица Поставщики состоит из 3 полей: idprovider (id поставщика) PK, number\_car (номер машины), FIO (ФИО).

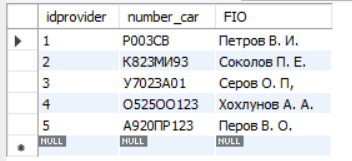


Рисунок 15 - таблица Поставщики

Таблица Сырье состоит из 5 полей: idraw (id сырья) PK, idindicators (id показателя) FK1, date\_of\_receipt (дата поступления), time\_of\_receipt (время поступления), volume\_weight (объем\масса).

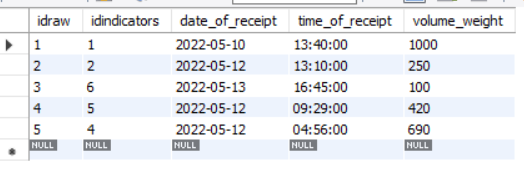


Рисунок 16 - таблица Сырьё

Таблица Стадии Технологического процесса состоит из 5 полей: idstages\_TP (id стадии ТП) PK, idTP (id ТП) FK1, iddevisions (id подразделения) FK2, name\_TP (наименование ТП), duration\_min ( продолжительность в минутах).

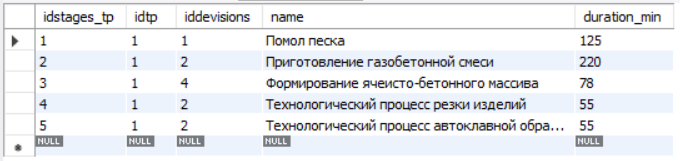


Рисунок 17 - таблица Стадии ТП

Таблица ТП (технологический процесс) состоит из 4 полей: idtp (id ТП) PK, date (дата), time (время), duration\_min (продолжительность в минутах).

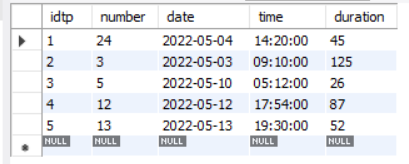


Рисунок 18 - таблица ТП

Таблица Типы сырья состоит из 4 полей: idtypes\_of\_raw (id типа сырья) PK, idraw (id сырья) FK1, idprovider (id поставщика) FK2, name\_tr (наименование сырья).

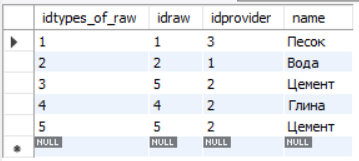


Рисунок 19 - таблица Виды сырья

Создав базу данных в СУБД MySQL Workbench 8.0 CE можем приступить к созданию интерфейса.

# **Реализация приложения**

Приложение по управлению базой данных для прогнозирования % брака имеет следующий алгоритм решения:

1. Разработка графического макета приложения. Состоит из следующих элементов:

* Форма: Подразделения, Поставщики, ТП (технологический процесс), Этапы ТП, Прогноз % брака, Сырьё, Виды сырья, Показатели.
* Кнопка выхода

1. Создание функций для работы с БД: Загрузка информации из БД.
2. Программирование кнопок, работающих с таблицей( добавление, удаление).
3. Создание диалоговых окон для добавления, удаления (данных во все таблицы), а также выхода из приложения.
4. Создание полей для внесения и удаления соответствующих данных во всех таблицах.
5. Создание функций для экспорта данных из таблиц в файл-книгу MS Excel.
   1. Описание интерфейса приложения

Интерфейс включает в себя 8 форм с таблицами, формы для добавления и удаления записей, всего в интерфейсе 25 форм, включая главную (стартовую) страницу. Рассмотрим таблицу ТП и действия с ней. Т.к. эта таблица имеет особые типы данных – время, дату, которые необходимо отдельно проверять.

### **Форма «Главная страница»**

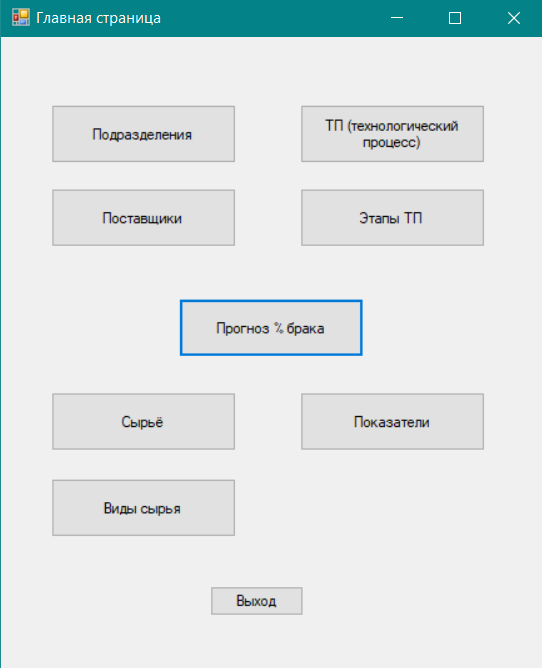


Рисунок 20 - Главная страница и ее компоненты

1. Кнопка перехода к форме с таблицей Подразделения. Имя компонента – button\_devisions. Класс компонента – Button.
2. Кнопка перехода к форме с таблицей Поставщики. Имя компонента – button\_providers. Класс компонента – Button.
3. Кнопка перехода к форме с таблицей ТП (технологический процесс). Имя компонента – button\_tp. Класс компонента – Button.
4. Кнопка перехода к форме с таблицей Этапы ТП. Имя компонента – button\_stages\_tp. Класс компонента – Button.
5. Кнопка перехода к форме с таблицей Прогноз % брака. Имя компонента – button\_prediction\_percent\_defect. Класс компонента – Button.
6. Кнопка перехода к форме с таблицей Сырьё. Имя компонента – button\_raw. Класс компонента – Button.
7. Кнопка перехода к форме с таблицей Виды сырья. Имя компонента – button\_types\_of\_raw. Класс компонента – Button.
8. Кнопка перехода к форме с таблицей Показатели. Имя компонента – button\_indicators. Класс компонента – Button.
9. Кнопка выхода из приложения с диалоговым окном. Имя компонента – button\_exit. Класс компонента – Button.

Т.к. все формы с таблицами имеют одинаковые возможности (ИДЕНТИЧНЫЕ) рассмотрим в качестве примера несколько форм. Листинги будут представлены в Приложении 1.

### **Форма «ТП»**

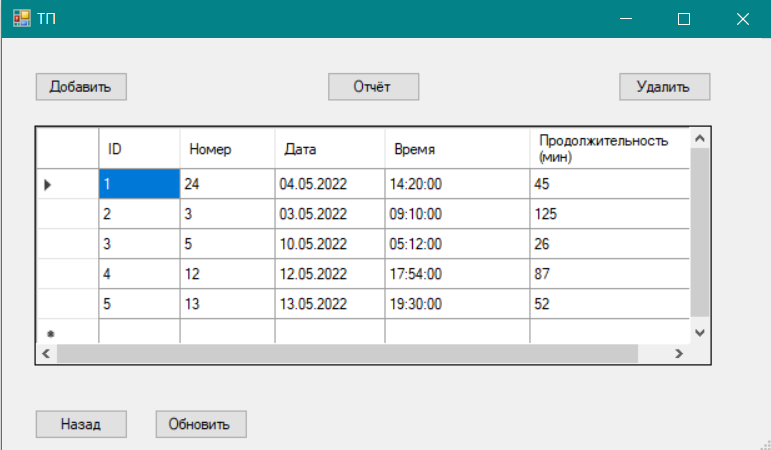


Рисунок 21 - форма ТП и ее компоненты

1. Таблица с данными о ТП. Имя компонента - dataGridView\_tp. Класс компонента – dataGridView
2. Кнопка перехода к форме добавления данных в таблицу. Имя компонента – button\_add. Класс компонента – Button.
3. Кнопка экспорта данных в Excel. Имя компонента – button\_export. Класс компонента – Button. (см. рис. 22-23)
4. Кнопка перехода к форме удаления данных из таблицы. Имя компонента – button\_delete. Класс компонента – Button.
5. Кнопка отмены и выхода из окна ТП. Имя компонента – button\_cancel. Класс компонента – Button.
6. Кнопка обновления формы. Имя компонента – button\_refresh. Класс компонента – Button.

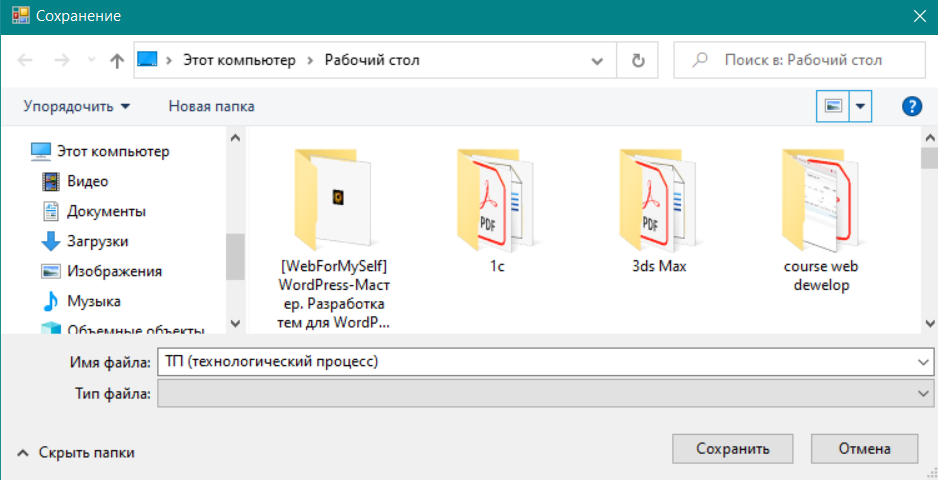


Рисунок 22 - окно сохранения данных

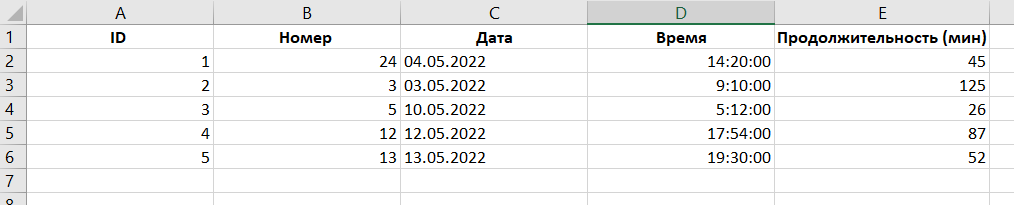


Рисунок 23 - содержимое отчета

**Функция экспорта:**

private void button\_export\_Click(object sender, EventArgs e)

{

Microsoft.Office.Interop.Excel.\_Application app = new Microsoft.Office.Interop.Excel.Application();

Microsoft.Office.Interop.Excel.\_Workbook workbook = app.Workbooks.Add(Type.Missing);

Microsoft.Office.Interop.Excel.\_Worksheet worksheet = null;

worksheet = workbook.Sheets["Лист1"];

worksheet = workbook.ActiveSheet;

worksheet.Name = "ТП";

app.Columns.ColumnWidth = 20;

for (int i = 1; i < dataGridView1.Columns.Count + 1; i++)

{

worksheet.Cells[1, i] = dataGridView1.Columns[i - 1].HeaderText;

}

for (int i = 0; i < dataGridView1.Rows.Count; i++)

{

for (int j = 0; j < dataGridView1.Columns.Count; j++)

{

worksheet.Cells[i + 2, j + 1] = dataGridView1.Rows[i].Cells[j].Value;

}

}

var saveFileDialoge = new SaveFileDialog();

saveFileDialoge.FileName = "ТП (технологический процесс)";

saveFileDialoge.DefaultExt = ".xlsx";

if (saveFileDialoge.ShowDialog() == DialogResult.OK)

{

workbook.SaveAs(saveFileDialoge.FileName, Type.Missing, Type.Missing, Type.Missing, Type.Missing, Type.Missing, Microsoft.Office.Interop.Excel.XlSaveAsAccessMode.xlExclusive, Type.Missing, Type.Missing, Type.Missing, Type.Missing, Type.Missing);

}

app.Quit();

}

### **Форма «Добавление ТП»**

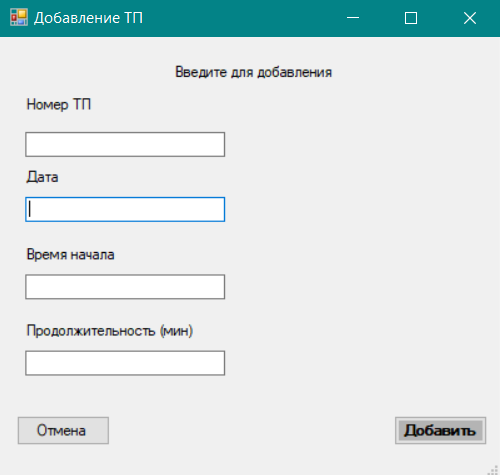


Рисунок 24 - Добавление ТП и ее компоненты

1. Поле ввода для номера ТП. Имя компонента – number\_Box. Класс компонента – TextBox.
2. Поле ввода для даты. Имя компонента – date\_Box. Класс компонента – TextBox.
3. Поле для ввода времени начала. Имя компонента – time\_Box. Класс компонента – TextBox.
4. Поле для ввода продолжительности в минутах. Имя компонента – duration\_Box. Класс компонента – TextBox.
5. Кнопка отмены и закрытия формы добавления ТП. Имя компонента – cancel\_button. класс компонента - Button.
6. Кнопка добавления данных в базу. Имя компонента – add\_button. класс компонента - Button.

Для всех полей ввода прописана валидация, что позволяет корректно вводить данные в БД. (см. рис. 25-26).

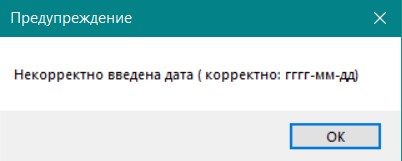


Рисунок 25 - окно предупреждения ввода даты

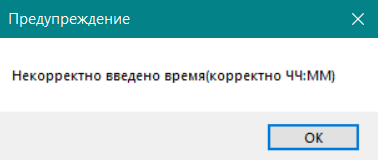


Рисунок 26 - окно предупреждения ввода времени

**Проверка на заполненность полей:**

private void button\_add\_Click(object sender, EventArgs e)

{

if (Get\_Correct\_Date() == "")

{

MessageBox.Show("Некорректно введена дата ( корректно: гггг-мм-дд) ", "Предупреждение"); //предупреждение о некорректно введенем формате даты

return;

}

if (Get\_Correct\_Time() == "")

{

MessageBox.Show("Некорректно введено время(корректно ЧЧ:ММ) ", "Предупреждение"); //предупреждение о незаполненном поле

return;

}

if (string.IsNullOrWhiteSpace(number\_Box.Text))

{

MessageBox.Show("Поле\"Номер ТП\" не заполнено", "Предупреждение"); //предупреждение о незаполненном поле

return;

}

if (string.IsNullOrWhiteSpace(duration\_Box.Text))

{

MessageBox.Show("Поле\"Продолжительность\" не заполнено", "Предупреждение"); //предупреждение о незаполненном поле

return;

}

**Регулярные выражения для проверки корректности даты и времени:**

private string Get\_Correct\_Date()

{

string result = "";

if (Regex.IsMatch(date\_Box.Text, @"[0-9]{4}\-[0-9]{2}\-[0-9]{2}", RegexOptions.IgnoreCase)) //условие для корректного ввода даты

{

result = date\_Box.Text;

}

return result;

}

private string Get\_Correct\_Time()

{

string result = "";

if (Regex.IsMatch(time\_Box.Text, @"([01][0-9]|2[0-3]):([0-5][0-9])", RegexOptions.IgnoreCase))

{

result = time\_Box.Text;

}

return result;

}

### **Форма «Удаление ТП»**

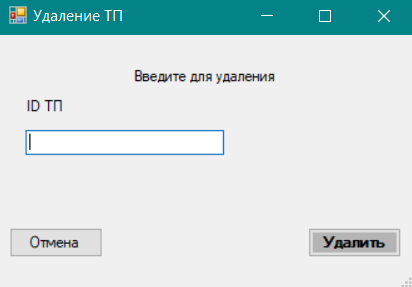


Рисунок 27 - Удаление ТП и ее компоненты

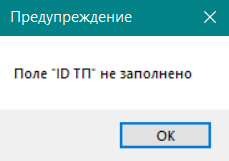


Рисунок 28 - окно предупреждения при удалении

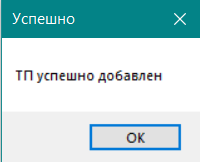


Рисунок 29 - окно успешного добавления данных

### **Ключевая форма «Прогнозирование % брака»**

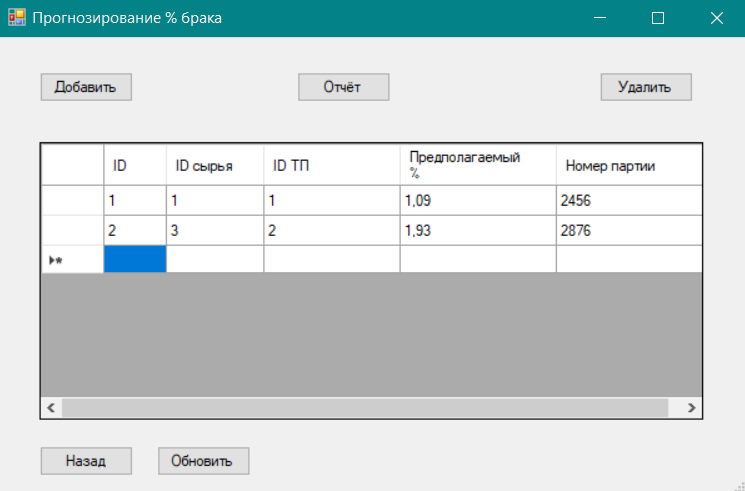


Рисунок 30 - ключевая форма и ее компоненты

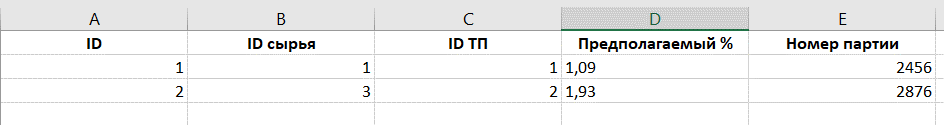


Рисунок 31 - отчет по % брака

* 1. Работа с MySQL

Для начала необходимо подключить ссылку MySql.Data. Теперь для удобства использования классов, через директиву using подключаем пространство имен MySql.Data.MySqlClient; Далее строки, которые будут подключаться к среде и к БД. Они имеют адрес сервер, данные пользователя, название БД. (все используемые данные описаны перед этими строками)

String connString = "Server=" + host + ";" + ";port=" + port + ";User Id=" + username + ";password=" + password;

String connString\_with\_DB = "Server=" + host + ";" + ";database=" + database + ";port=" + port + ";User Id=" + username + ";password=" + password;

Создадим экземпляр класса, который будет устанавливать соединение с нашей БД: MySqlConnection conn = new MySqlConnection(connString);

В качестве аргумента передаем нашу строку подключения. Объект создан. Необходимо установить соединение, а по окончании работы, соответственно, закрыть его. Для этого вызываем методы: conn.Open(); //открытие подключения conn.Close(); //закрытие(разрыв) подключения

Используем команду на создание БД (если ранее она не была создана), далее команда на создание таблиц в ней (при том же условии). Console.WriteLine(ex.ToString()); - выводим запрошенные данные.

* 1. Результат

После запуска приложения мы попадаем на Главную форму, на которой находятся основные кнопки для работы с таблицами БД:

* Подразделения;
* Поставщики;
* ТП (технологический процесс);
* Этапы ТП;
* Сырьё;
* Виды сырья;
* Показатели;
* Кнопка «Выход»

Приложение защищает базу данных от некорректных данных, которые пытается ввести пользователь, и предотвращает наличие незаполненных полей в базе.

# **Заключение**

Перед выполнением работы ставилась задача, о создании приложения для управления БД. Посредством изучения предметной области, были выбраны дизайн и структура приложения. Зная, насколько важна точность в подобной сфере деятельности, приложение защищает БД от некорректных и неточных данных.

Результатом проделанной работы является готовый продукт - приложение для ведения БД «Прогнозирование % брака», который может служить полноценной основой для проектирование наиболее масштабного и детального, а главное безопасного приложения.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Проектирование информационных систем: лабораторный практикум / Е. Б. Тюнин, В. Ю. Кондратьев. – Краснодар.
2. Г.А Черноморов «Базы данных в среде промышленных СУБД» - Новочеркасск : ЮРГТУ, 2006;
3. Построение баз данных в среде Oracle/Алексей Наместников,2010г;
4. Кумратова А.М. Методы хранения и анализа данных: учеб. пособие / А. М. Кумратова, И. И. Василенко. – Краснодар: КубГАУ, 2021. – 183 с.
5. Кузин А.В., Левонисова С.В. Базы данных / А.В. Кузин, С.В. Левонисова. – М. : Academia, 2012. – 320 с.

# **ПРИЛОЖЕНИЕ 1. ЛИСТИНГ ПРОГРАММЫ**

**Файл Form1.cs (файл главной страницы):**

using MySql.Data.MySqlClient;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace CourseWork

{

public partial class Form1 : Form

{

SQL\_connector connector;

public Form1()

{

InitializeComponent();

connector = new SQL\_connector();

connector.Get\_Connection\_First\_Time();

}

private void button\_exit\_Click(object sender, EventArgs e)

{

MessageBoxButtons msb = MessageBoxButtons.YesNo;

String message = "Вы действительно хотите выйти?";

String caption = "Выход";

if (MessageBox.Show(message, caption, msb) == DialogResult.Yes)

this.Close();

}

private void button\_devisions\_Click(object sender, EventArgs e)

{

DialogResult dr = new DialogResult();

devisions frm2 = new devisions();

dr = frm2.ShowDialog();

}

private void button\_indicators\_Click(object sender, EventArgs e)

{

DialogResult dr = new DialogResult();

indicators frm3 = new indicators();

dr = frm3.ShowDialog();

}

private void button\_percent\_Click(object sender, EventArgs e)

{

DialogResult dr = new DialogResult();

prediction\_percent\_defect frm4 = new prediction\_percent\_defect();

dr = frm4.ShowDialog();

}

private void button\_providers\_Click(object sender, EventArgs e)

{

DialogResult dr = new DialogResult();

provider frm5 = new provider();

dr = frm5.ShowDialog();

}

private void button\_raw\_Click(object sender, EventArgs e)

{

DialogResult dr = new DialogResult();

raw frm6 = new raw();

dr = frm6.ShowDialog();

}

private void button\_stages\_tp\_Click(object sender, EventArgs e)

{

DialogResult dr = new DialogResult();

stages\_tp frm7 = new stages\_tp();

dr = frm7.ShowDialog();

}

private void button\_tp\_Click(object sender, EventArgs e)

{

DialogResult dr = new DialogResult();

tp frm8 = new tp();

dr = frm8.ShowDialog();

}

private void button\_types\_raw\_Click(object sender, EventArgs e)

{

DialogResult dr = new DialogResult();

types\_of\_raw frm9 = new types\_of\_raw();

dr = frm9.ShowDialog();

}

}

}

**Файл SQL\_connector.cs (Файл подключения к БД):**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using MySql.Data.MySqlClient;

namespace CourseWork

{

class SQL\_connector

{

static string host = "127.0.0.1";

static int port = 3306;

static string database = "PREDICTION\_PERCENT";

static string username = "admin";

static string password = "password";

String connString = "Server=" + host + ";" + ";port=" + port + ";User Id=" + username + ";password=" + password;

String connString\_with\_DB = "Server=" + host + ";" + ";database=" + database + ";port=" + port + ";User Id=" + username + ";password=" + password;

public SQL\_connector()

{

}

public void Get\_Connection\_First\_Time()

{

MySqlConnection conn = new MySqlConnection(connString);

try

{

conn.Open();

MySqlCommand cmd = new MySqlCommand();

cmd.Connection = conn;

cmd.CommandText = "CREATE DATABASE IF NOT EXISTS `PREDICTION\_PERCENT`;";

cmd.ExecuteNonQuery();

conn.Close();

}

catch (Exception ex)

{

Console.WriteLine(ex.ToString());

}

conn = new MySqlConnection(connString\_with\_DB);

try

{

conn.Open();

MySqlCommand cmd = new MySqlCommand();

cmd.Connection = conn;

cmd.CommandText = "CREATE TABLE IF NOT EXISTS `devisions`(iddevisions INTEGER PRIMARY KEY AUTO\_INCREMENT, name TEXT); ";

cmd.ExecuteNonQuery();

cmd.CommandText = "CREATE TABLE IF NOT EXISTS `indicators`(idindicators INTEGER PRIMARY KEY AUTO\_INCREMENT, name TEXT, normal BOOLEAN); ";

cmd.ExecuteNonQuery();

cmd.CommandText = "CREATE TABLE IF NOT EXISTS `provider`(idprovider INTEGER PRIMARY KEY AUTO\_INCREMENT, number\_car TEXT, FIO TEXT);";

cmd.ExecuteNonQuery();

cmd.CommandText = "CREATE TABLE IF NOT EXISTS `tp`(idtp INTEGER PRIMARY KEY AUTO\_INCREMENT, number INTEGER, date DATE, time TIME, duration INTEGER);";

cmd.ExecuteNonQuery();

cmd.CommandText = "CREATE TABLE IF NOT EXISTS `PREDICTION\_PERCENT`.`prediction\_percent\_defect` (" +

"`idprediction\_percent\_defect` INT NOT NULL AUTO\_INCREMENT," +

"`idraw` INT NOT NULL," +

"`idtp` INT NOT NULL," +

"`assumed\_percent` FLOAT NOT NULL," +

"`batch\_number` INT NOT NULL," +

"PRIMARY KEY(`idprediction\_percent\_defect`)," +

"INDEX `tp\_idx` (`idtp` ASC) VISIBLE," +

"INDEX `raw\_idx` (`idraw` ASC) VISIBLE," +

"CONSTRAINT `tp\_prediction`" +

" FOREIGN KEY(`idtp`)" +

" REFERENCES `PREDICTION\_PERCENT`.`tp` (`idtp`)" +

" ON DELETE CASCADE" +

" ON UPDATE CASCADE," +

"CONSTRAINT `raw`" +

" FOREIGN KEY(`idraw`)" +

" REFERENCES `PREDICTION\_PERCENT`.`raw` (`idraw`)" +

" ON DELETE CASCADE" +

" ON UPDATE CASCADE)";

cmd.ExecuteNonQuery();

cmd.CommandText = "CREATE TABLE IF NOT EXISTS `PREDICTION\_PERCENT`.`raw` (" +

"`idraw` INT NOT NULL AUTO\_INCREMENT," +

"`idindicators` INT NOT NULL," +

"`date\_of\_receipt` DATE NOT NULL," +

"`time\_of\_receipt` TIME NOT NULL," +

"`volume\_weight` INT NOT NULL," +

"PRIMARY KEY(`idraw`)," +

"INDEX `indic\_idx` (`idindicators` ASC) VISIBLE," +

"CONSTRAINT `indic`" +

" FOREIGN KEY(`idindicators`)" +

" REFERENCES `PREDICTION\_PERCENT`.`indicators` (`idindicators`)" +

" ON DELETE CASCADE" +

" ON UPDATE CASCADE)";

cmd.ExecuteNonQuery();

cmd.CommandText = "CREATE TABLE IF NOT EXISTS `PREDICTION\_PERCENT`.`stages\_tp` (" +

"`idstages\_tp` INT NOT NULL AUTO\_INCREMENT," +

"`idtp` INT NOT NULL," +

"`iddevisions` INT NOT NULL," +

"`name` TEXT NOT NULL," +

"`duration\_min` INT NOT NULL," +

"PRIMARY KEY(`idstages\_tp`)," +

"INDEX `dev\_idx` (`iddevisions` ASC) VISIBLE," +

"INDEX `tp\_idx` (`idtp` ASC) VISIBLE," +

"CONSTRAINT `dev`" +

" FOREIGN KEY(`iddevisions`)" +

" REFERENCES `PREDICTION\_PERCENT`.`devisions` (`iddevisions`)" +

" ON DELETE CASCADE" +

" ON UPDATE CASCADE," +

"CONSTRAINT `tp`" +

" FOREIGN KEY(`idtp`)" +

" REFERENCES `PREDICTION\_PERCENT`.`tp` (`idtp`)" +

" ON DELETE CASCADE" +

" ON UPDATE CASCADE)";

cmd.ExecuteNonQuery();

cmd.CommandText = "CREATE TABLE IF NOT EXISTS `PREDICTION\_PERCENT`.`types\_of\_raw` (" +

"`idtypes\_of\_raw` INT NOT NULL AUTO\_INCREMENT," +

"`idraw` INT NOT NULL," +

"`idprovider` INT NOT NULL," +

"`name` TEXT NOT NULL," +

"PRIMARY KEY(`idtypes\_of\_raw`)," +

"INDEX `prov\_idx` (`idprovider` ASC) VISIBLE," +

"INDEX `raw\_idx` (`idraw` ASC) VISIBLE," +

"CONSTRAINT `prov`" +

" FOREIGN KEY(`idprovider`)" +

" REFERENCES `PREDICTION\_PERCENT`.`provider` (`idprovider`)" +

" ON DELETE CASCADE" +

" ON UPDATE CASCADE," +

"CONSTRAINT `raw\_types`" +

" FOREIGN KEY(`idraw`)" +

" REFERENCES `PREDICTION\_PERCENT`.`raw` (`idraw`)" +

" ON DELETE CASCADE" +

" ON UPDATE CASCADE)";

cmd.ExecuteNonQuery();

conn.Close();

}

catch (Exception ex)

{

Console.WriteLine(ex.ToString());

}

}

public MySqlConnection Get\_Connection\_For\_Operations()

{

return new MySqlConnection(connString\_with\_DB);

}

}

}

**Файл tp.cs (файл формы ТП, которая рассмотрена в главе 3. Файлы форм: Подразделения, показатели, Стадии ТП, Прогнозирование % брака, Виды сырья, Сырьё и Показатели идентичны по содержанию):**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using MySql.Data.MySqlClient;

using System.Text.RegularExpressions;

using System.Globalization;

namespace CourseWork

{

public partial class tp : Form

{

SQL\_connector connector;

public tp()

{

InitializeComponent();

connector = new SQL\_connector();

connector.Get\_Connection\_First\_Time();

LoadData();

}

private void button\_cancel\_Click(object sender, EventArgs e)

{

this.Close();

}

private void LoadData()

{

MySqlConnection conn = connector.Get\_Connection\_For\_Operations();

try

{

conn.Open();

string query = "SELECT \* FROM tp ORDER BY idtp";

MySqlCommand cmd = new MySqlCommand(query);

cmd.Connection = conn;

MySqlDataReader rdr = cmd.ExecuteReader();

List<string[]> data = new List<string[]>();

while (rdr.Read())

{

data.Add(new string[5]);

data[data.Count - 1][0] = rdr[0].ToString(); //прочитываем 0 столбец ID

data[data.Count - 1][1] = rdr[1].ToString(); //прочитываем 1 столбец number

data[data.Count - 1][2] = rdr[2].ToString().Split(' ')[0]; //прочитываем 2 столбец data

data[data.Count - 1][3] = rdr[3].ToString(); //прочитываем 3 столбец time

data[data.Count - 1][4] = rdr[4].ToString(); //прочитываем 4 столбец duration

}

rdr.Close();

foreach (string[] s in data)

dataGridView1.Rows.Add(s);

conn.Close();

}

catch (Exception ex)

{

MessageBox.Show("Ошибка вывода данных", "Ошибка");

}

}

private void button\_export\_Click(object sender, EventArgs e)

{

Microsoft.Office.Interop.Excel.\_Application app = new Microsoft.Office.Interop.Excel.Application();

Microsoft.Office.Interop.Excel.\_Workbook workbook = app.Workbooks.Add(Type.Missing);

Microsoft.Office.Interop.Excel.\_Worksheet worksheet = null;

worksheet = workbook.Sheets["Лист1"];

worksheet = workbook.ActiveSheet;

worksheet.Name = "ТП";

app.Columns.ColumnWidth = 20;

for (int i = 1; i < dataGridView1.Columns.Count + 1; i++)

{

worksheet.Cells[1, i] = dataGridView1.Columns[i - 1].HeaderText;

}

for (int i = 0; i < dataGridView1.Rows.Count; i++)

{

for (int j = 0; j < dataGridView1.Columns.Count; j++)

{

worksheet.Cells[i + 2, j + 1] = dataGridView1.Rows[i].Cells[j].Value;

}

}

var saveFileDialoge = new SaveFileDialog();

saveFileDialoge.FileName = "ТП (технологический процесс)";

saveFileDialoge.DefaultExt = ".xlsx";

if (saveFileDialoge.ShowDialog() == DialogResult.OK)

{

workbook.SaveAs(saveFileDialoge.FileName, Type.Missing, Type.Missing, Type.Missing, Type.Missing, Type.Missing, Microsoft.Office.Interop.Excel.XlSaveAsAccessMode.xlExclusive, Type.Missing, Type.Missing, Type.Missing, Type.Missing, Type.Missing);

}

app.Quit();

}

private void button\_add\_Click(object sender, EventArgs e)

{

DialogResult dr = new DialogResult();

add\_tp frm13 = new add\_tp();

dr = frm13.ShowDialog();

refresh\_window();

}

private void button\_delete\_Click(object sender, EventArgs e)

{

DialogResult dr = new DialogResult();

del\_tp frm15 = new del\_tp();

dr = frm15.ShowDialog();

refresh\_window();

}

private void button\_refresh\_Click(object sender, EventArgs e)

{

refresh\_window();

}

private void refresh\_window()

{

this.Hide(); //закрывает предыдущую форму

tp tp = new tp(); // показывает заново форму

tp.ShowDialog();

}

}

}

**Файл add\_tp.cs (файл формы Добавления ТП, которая рассмотрена в главе 3. Файлы форм по добавлению идентичны):**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Text.RegularExpressions;

using System.Windows.Forms;

using MySql.Data.MySqlClient;

using MySqlX.XDevAPI.Common;

// здесь не отображается последний столбец в форме??

namespace CourseWork

{

public partial class add\_tp : Form

{

SQL\_connector connector;

public add\_tp()

{

InitializeComponent();

connector = new SQL\_connector();

}

private void button\_cancel\_Click(object sender, EventArgs e)

{

this.Close();

}

private void button\_add\_Click(object sender, EventArgs e)

{

if (Get\_Correct\_Date() == "")

{

MessageBox.Show("Некорректно введена дата ( корректно: гггг-мм-дд) ", "Предупреждение"); //предупреждение о некорректно введенем формате даты

return;

}

if (Get\_Correct\_Time() == "")

{

MessageBox.Show("Некорректно введено время(корректно ЧЧ:ММ) ", "Предупреждение"); //предупреждение о незаполненном поле

return;

}

if (string.IsNullOrWhiteSpace(number\_Box.Text))

{

MessageBox.Show("Поле\"Номер ТП\" не заполнено", "Предупреждение"); //предупреждение о незаполненном поле

return;

}

if (string.IsNullOrWhiteSpace(duration\_Box.Text))

{

MessageBox.Show("Поле\"Продолжительность\" не заполнено", "Предупреждение"); //предупреждение о незаполненном поле

return;

}

MySqlConnection conn = connector.Get\_Connection\_For\_Operations();

try

{

conn.Open(); //открываем соединение

MySqlCommand cmd = new MySqlCommand(); //подключаемся к таблице

cmd.Connection = conn;

cmd.CommandText = "INSERT INTO tp(number, date, time, duration) VALUES(@number, @date, @time, @duration)"; //если таблица отсутствует, создает

cmd.Parameters.AddWithValue("@number", number\_Box.Text);

cmd.Parameters.AddWithValue("@date", date\_Box.Text);

cmd.Parameters.AddWithValue("@time", time\_Box.Text);

cmd.Parameters.AddWithValue("@duration", duration\_Box.Text);

cmd.ExecuteNonQuery();

conn.Close(); //передаем данные и закрываем соединение

}

catch (Exception ex)

{

MessageBox.Show("ТП не добавлен", " Ошибка "); //сообщение о результате

}

MessageBox.Show("ТП успешно добавлен", "Успешно");

this.Close();

}

private string Get\_Correct\_Date()

{

string result = "";

if (Regex.IsMatch(date\_Box.Text, @"[0-9]{4}\-[0-9]{2}\-[0-9]{2}", RegexOptions.IgnoreCase)) //условие для корректного ввода даты

{

result = date\_Box.Text;

}

return result;

}

private string Get\_Correct\_Time()

{

string result = "";

if (Regex.IsMatch(time\_Box.Text, @"([01][0-9]|2[0-3]):([0-5][0-9])", RegexOptions.IgnoreCase))

{

result = time\_Box.Text;

}

return result;

}

}

}

**Файл del\_tp.cs (файл формы Удаление ТП, которая рассмотрена в главе 3. Файлы форм по удалению идентичны):**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using MySql.Data.MySqlClient;

namespace CourseWork

{

public partial class del\_tp : Form

{

SQL\_connector connector;

public del\_tp()

{

InitializeComponent();

connector = new SQL\_connector();

}

private void button\_cancel\_Click(object sender, EventArgs e)

{

this.Close();

}

private void button\_delete\_Click(object sender, EventArgs e)

{

if (string.IsNullOrEmpty(id\_Box.Text))

{

string message = "Поле \"ID ТП\" не заполнено";

string title = "Предупреждение";

MessageBox.Show(message, title);

return;

}

else

{

MySqlConnection conn = connector.Get\_Connection\_For\_Operations();

try

{

conn.Open(); //открываем соединение

MySqlCommand cmd = new MySqlCommand(); //подключаемся к таблице

cmd.Connection = conn;

cmd.CommandText = "DELETE FROM tp WHERE idtp = @id;"; //если таблица отсутствует, создает

cmd.Parameters.AddWithValue("@id", int.Parse(id\_Box.Text));

cmd.ExecuteNonQuery();

MessageBox.Show("Успешно удалено по ID", "Успешно");

conn.Close(); //передаем данные и закрываем соединение

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, " Ошибка "); //сообщение о результате

}

this.Close();

}

}

}

}