Министерство образования Республики Беларусь

Министерство науки и высшего образования Российской Федерации

МЕЖГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ   
ВЫСШЕГО ОБРАЗОВАНИЯ

«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра Металлорежущие станки и инструменты

**ДОПУЩЕН К ЗАЩИТЕ**

Зав. кафедрой «МРСиИ»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_С.Н. Хатетовский

«\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_2023 г.

**ДИПЛОМНЫЙ ПРОЕКТ**

Разработка информационной системы для мониторинга и контроля за процессами производства и качеством продукции

(ОАО «Могилевхимволокно»)

(тема дипломного проекта)

Пояснительная записка

(вид документа)

ДП.540/23.00.00.000 ПЗ

|  |  |
| --- | --- |
| Студент | ( Харкевич А.В. ) |
| Руководитель проекта | ( Галюжин Д.С. ) |
| Консультант по экономической части проекта | ( Пушкина Л. И. ) |
| Консультант по охране труда | ( Шилова И. В. ) |
| Нормоконтроль | ( Попов В. Б. ) |
|  |  |

Могилев 2023

Содержание

[Введение 5](#_Toc137602812)

[1 Анализ объекта автоматизации 8](#_Toc137602813)

[1.1 Структура предприятия 8](#_Toc137602814)

[1.2 Постановка задачи 13](#_Toc137602815)

[2 Проектирование системы 16](#_Toc137602816)

[2.1 Сущность задачи 16](#_Toc137602817)

[2.2 Структура системы 17](#_Toc137602818)

[3 Анализ текущего состояния системы мониторинга и контроля за процессами производства 20](#_Toc137602819)

[3.1 Описание процесса производства 20](#_Toc137602820)

[3.2 Анализ существующих проблем и недостатков 22](#_Toc137602821)

[3.3 Определение требований к системе мониторинга и контроля за процессами производства 23](#_Toc137602822)

[4 Реализация программного обеспечения 25](#_Toc137602823)

[4.1 Выбор и обоснование средств реализации для разработки программного обеспечения 25](#_Toc137602824)

[4.2 Описание таблиц базы данных 29](#_Toc137602825)

[4.3 Реализация отдельных функций 34](#_Toc137602826)

[4.3.1 Добавление новых записей в базу данных 34](#_Toc137602827)

[4.3.2 Поиск записей в базе данных 36](#_Toc137602828)

[4.3.3 Удаление записей в базе данных 37](#_Toc137602829)

[4.3.4 Редактирование записей в базе данных 38](#_Toc137602830)

[4.3.5 Печать Отчета 40](#_Toc137602831)

[5 Инструкция к эксплуатации 44](#_Toc137602832)

[6 Организационно-экономическая часть 48](#_Toc137602833)

[6.1 Организация технического контроля качества на предприятии 48](#_Toc137602834)

[6.2 Обоснование эффективности технических решений 55](#_Toc137602835)

[6.2.1 Определение объёма программного средства 55](#_Toc137602836)

[6.2.2 Расчёт нормативной трудоёмкости 57](#_Toc137602837)

[6.2.3 Расчёт плановой себестоимости ПО 60](#_Toc137602838)

[6.2.4 Расчет показателей экономической эффективности ПО 69](#_Toc137602839)

[7 Охрана труда 75](#_Toc137602840)

[7.1 Идентификация и анализ вредных и опасных факторов при работе на компьютере 75](#_Toc137602841)

[7.2 Технические, технологические, организационные решения по устранению опасных и вредных факторов, разработка защитных средств. 84](#_Toc137602842)

[7.3. Индивидуальное задание. 86](#_Toc137602843)

[7.4 Анализ выполнения требований охраны труда в структурном подразделении (на рабочем месте) при работе на ПЭВМ и другой офисной технике 86](#_Toc137602844)

[7.5 Заключение по разделу 89](#_Toc137602845)

[Заключение по проекту 90](#_Toc137602846)

[Список использованных источников 91](#_Toc137602847)

[Приложение А 94](#_Toc137602848)

[Приложение Б 102](#_Toc137602849)

# Введение

Между данными и информационными системами существует глубокая связь. Данные являются сырым материалом, состоящим из фактов, чисел, текстов и другой информации. Тем не менее, чтобы сделать данные полезными, их нужно обработать и организовать в структурированный вид, что позволит получить информацию, применимую для принятия решений и выполнения задач.

Информационные системы — это сложные системы, созданные для сбора, хранения, обработки, анализа и предоставления информации. Они предлагают структурированный и систематический подход к управлению данными и предоставлению информации для принятия решений и выполнения задач.

Для хранения и доступа к данным, содержащим информацию о конкретной предметной области, применяются базы данных. Система баз данных — это, в основном, компьютеризированная система хранения однородных записей. Сама база данных может рассматриваться как аналог электронного каталога, т. е. хранилище для определенного набора файлов данных, сохраненных на компьютере.

Базы данных являются неотъемлемым элементом современного информационного общества и находят применение в разных сферах, таких как бизнес, наука, образование, здравоохранение, правительство и другие отрасли. С помощью баз данных можно эффективно хранить и организовывать большие объемы информации. Они позволяют структурировать данные в таблицы, связи или коллекции, что упрощает доступ к ним и их управление.

Базы данных предоставляют инструменты для управления данными, такие как добавление, обновление, удаление и поиск информации. Используя MS SQL можно эффективно хранить и организовывать большие объемы данных. MS SQL поддерживает использование языка запросов SQL (Structured Query Language), что позволяет выполнять разные сложные операции с данными.

С помощью баз данных можно гарантировать целостность данных, устанавливая правила и ограничения, которым должны следовать при добавлении или изменении информации. Базы данных предлагают механизмы для обеспечения безопасности данных и помогают предотвращать несанкционированный доступ к конфиденциальной информации.

Базы данных позволяют нескольким пользователям одновременно работать с данными и совместно использовать информацию. Это особенно важно на рабочих местах, где большое количество пользователей должно иметь доступ к общей информации и вносить изменения. Таким образом, базы данных обеспечивают возможность совместной работы, позволяя пользователям делиться данными и синхронизировать свои действия, что способствует эффективности и согласованности работы в команде.

Взаимодействие между базой данных и приложением на языке программирования C# играет критическую роль в успешной реализации современных информационных систем. Использование C# в качестве основного языка для разработки приложений, взаимодействующих с базами данных, предоставляет ряд преимуществ, обеспечивает надежность, производительность и масштабируемость системы.

Один из основных аспектов взаимодействия с базами данных в C# заключается в использовании Entity Framework, специальной библиотеки, предназначенной для работы с различными источниками данных, включая MS SQL. Entity Framework обеспечивает мощный и гибкий интерфейс для работы с базами данных, что позволяет обрабатывать и управлять данными практически любой структуры и сложности.

В соответствии с изложенной выше информацией для дипломного проекта была выбрана тема: «Разработка информационной системы для мониторинга и контроля за процессами производства и качеством продукции (ОАО «Могилевхимволокно»)»

Объектом дипломного проектирования является мониторинг и контроль за процессами производства, хранение чертежей и другой необходимой информации.

Предметом дипломного проекта приложение для мониторинга и контроля за процессами производства.

Цель дипломного проекта – разработка системы управления для мониторинга и контроля за процессами производства на предприятии ОАО «Могилевхимволокно».

Практическая важность дипломной работы заключается в уменьшении количества бумажных документов, возможности отслеживания текущего этапа производства изделия, а также обеспечении хранения, извлечения и управления большими объемами информации.

# 1 Анализ объекта автоматизации

# 1.1 Структура предприятия

ОАО «Могилевхимволокно» является уникальным в Республике Беларусь крупным производителем таких продуктов, как диметилтерефталат, полиэфирный ПЭТ-гранулят, полиэфирные волокна и нити, синтетические пленки, и играет важную роль в поставке сырья для легкой промышленности.

Компания «Могилевхимволокно» является одним из основных градообразующих предприятий Могилева и обладает развитой инфраструктурой для поддержания успешной работы своего крупного производственного комплекса и подготовки высококвалифицированных кадров.

Достоинством этого предприятия можно считать интеграцию нескольких производств на одной территории, соединенных единым технологическим процессом. Это позволяет ему охватывать весь цикл производства - от добычи исходного сырья (диметилтерефталат и полиэтилентерефталат) до выхода готовой продукции в виде различных видов волокон, нитей и нетканых материалов.

Товары, выпускаемые компанией, соответствуют мировым качественным стандартам и пользуются большим спросом. Более 65% производимой продукции отправляется на международные рынки и поставляется в 30 стран по всему миру.

В данный момент компания включает в себя пять основных производственных подразделений:

1. Органический синтез: диметилтерефталат, полиэтилентерефталат, сополимеры полиэтилентерефталата, текстовые ПЭТ, ПЭТ-преформы, стандартные и «конжугейт» синтетические волокна;

2. Синтетическое волокно: полиэфирные волокна, композитные материалы, полиэфирные основы для кровельных материалов, геотекстили, нетканые полотна различного назначения, обвязочные ленты, пожарные рукава;

3. Технические нити: нити для технического использования;

4. Синтетические пленки: разнообразные полипропиленовые, полиолефиновые и полиэтиленовые пленки;

5. Ремонтно-механическое производство: оборудование и запчасти для химических производств.

ОАО «Могилевхимволокно» обладает разветвленной социальной инфраструктурой, включая санаторий «Сосны», гостиницы, сельскохозяйственный и общественный сектор, а также цех общественного питания и хозяйственный цех.

Основные приоритеты развития компании включают более эффективное использование имеющихся ресурсов, повышение конкурентоспособности продукции через улучшение качества и снижение затрат, модернизацию и разработку новых видов продукции.

Активно осуществляется реконструкция и внедрение новых производств, а также реализуются инвестиционные проекты, направленные на энергоэффективность и углубление переработки основного сырья.

Заслуживая усилия экспертов ОАО «Могилевхимволокно» и поддержку государства, компания успешно реализовала несколько проектов, ориентированных на передовые технологии и инновационные научные достижения, что позволило выпускать конкурентные товары, которые пользуются стабильным спросом на мировом рынке.

В рамках своей стратегии развития, ОАО «Могилевхимволокно» ставит перед собой главную задачу – переход на производство полиэфирной продукции на основе терефталевой кислоты. Это обусловлено как экономической выгодой такой технологии, так и глобальным трендом – мировое производство полиэтилентерефталата основывается на использовании терефталевой кислоты, в то время как производство диметилтерефталата почти исчезло.

Компания активно проводит масштабную модернизацию существующих производств. Была запущена первая стадия инвестиционного проекта «Комплекс по выпуску полиэфирной продукции», в рамках которой предусмотрены производство гранулята полиэтилентерефталата текстильного назначения и абсолютно новое производство полиэфирного волокна методом прямого формования на базе действующей установки непрерывной поликонденсации с мощностью 80 тыс. тонн в год.

В ходе выполнения данного проекта было начато производство новых видов полиэфирной продукции, таких как полиэфирное волокно «конжугейт», которое не производится в Республике Беларусь и странах СНГ. Это позволило расширить ассортимент полиэфирных технических нитей за счет производства «тонких» видов. Также были прекращены работы нерентабельных и устаревших производств.

Учитывая текущие обстоятельства экономики, стало необходимым осуществить инвестиционные проекты с минимальными капиталовложениями, которые приносят значительный результат. Приоритетными являются проекты, направленные на стабилизацию хозяйственной деятельности, выход на новые рынки и поддержание социальной стабильности.

Основываясь на поставленных целях, в Программе развития до 2030 года предусмотрели на реализацию следующих инвестиционных проектов:

1. «Комплекс по производству полиэфирной продукции в ОАО «Могилевхимволокно». Вторая очередь строительства: «Реконструкция корпуса ПОЦ №2 ЗСВ с размещением установки непрерывной твердофазной дополиконденсации ПЭТ и производством технических нитей по проспекту Шмидта, 45/148-1в г. Могилеве».

2. «Реконструкция главного корпуса ПСКН-1 для организации производства полиэфирных нетканых полотен гидроструйным методом по пр. Шмидта, 45 в г. Могилеве».

3. «Реконструкция главного корпуса ПСКН-1 для организации производства полиэфирных иглопробивных нетканых материалов по пр. Шмидта, 45 в г. Могилеве».

Преимущества ОАО «Могилевхимволокно» в производственной сфере включают:

- сильные конкурентные позиции на рынке стран СНГ;

- обширный ассортимент продукции;

- квалифицированные сотрудники;

- значительные производственные возможности.

В ОАО «Могилевхимволокно» функционирует интегрированная система управления, соответствующая международным стандартам ISO 9001, ISO 45001 и ISO 14001.

Основная цель политики в области качества, охраны труда и окружающей среды заключается в удовлетворении потребностей клиентов, получении оптимальной прибыли, обеспечении технической безопасности, безопасных условий труда и снижении негативного воздействия на окружающую среду.

Производимая продукция соответствует качеству зарубежных аналогов и является конкурентоспособной на международных рынках. В связи с достижениями в области качества, ОАО «Могилевхимволокно» стал лауреатом Премии Правительства Республики Беларусь в 2010 и 2015 годах.

По результатам социально-экономического развития, начиная с 2015 года, компания стабильно занимает лидирующие позиции в производственной сфере Октябрьского района города Могилев и каждый год включается в районную Доску Почета.

Товары предприятия регулярно становятся лауреатами конкурса «Лучшие товары Республики Беларусь», проводимого Государственным комитетом по стандартизации страны.

В 2020 году полиэфирное полое силиконизированное волокно «конжугейт» 0,78 текс, стандартное полиэфирное матированное волокно 0,156 текс 38 мм, техническая пневмосоединенная полиэфирная нить типа М 960 линейной плотности 113 текс, а также многослойная полимерная пленка были признаны лучшими товарами прошлого года.

Компания была награждена специальным призом «Лучший в регионе» и дипломом «Стабильное качество».

ОАО «Могилевхимволокно» предоставляет услуги промышленного и непромышленного характера, транспортно-экспедиционное обслуживание грузов, услуги в социальной сфере, а также услуги по ремонту и изготовлению запасных частей оборудования.

В составе ОАО «Могилевхимволокно» ремонтно-механическое производство было основано в 1969 году с целью производства деталей, узлов и оборудования для технологических процессов химических компаний, использующих сложное импортное оборудование. За свою историю это предприятие построило репутацию надежного производителя качественной продукции.

На сегодняшний день ремонтно-механическое производство представляет собой машиностроительное предприятие, которое включает

- цех сборки и сварки;

- центральный ремонтный цех;

- механический цех;

- ремонтно-монтажный цех;

- энергетико-механическую службу.

Современное ремонтно-механическое производство работает в таких ключевых направлениях:

1. Проектирование, изготовление, монтаж и ремонт нестандартного емкостного и теплообменного оборудования (в том числе по лицензии Госкомитета РБ по надзору за безопасным ведением работ в промышленности и атомной энергетике) с давлением до 25 кгс/кв. см (ремонт – до 50 кгс/кв. см) из различных металлов, включая коррозионностойкие.

2. Производство всех видов механической обработки металлов и сплавов (в том числе специальных видов зубчатых зацеплений) с возможностью разработки конструкторской документации.

3. Производство изделий из пластмасс методами литья под давлением и экструзии (полиэтилен, полипропилен, АБС-пластик) и изделий из резинотехнических смесей. Готовы рассмотреть возможность проектирования и изготовления технологической оснастки для освоения выпуска пластмассовых изделий по требованию заказчика.

4. Ремонт всех видов запорной, регулирующей и предохранительной арматуры Dу до 400 мм с регулировкой, испытанием и предоставлением необходимых документов по лицензии Государственного комитета Республики Беларусь по надзору за безопасным ведением работ в промышленности и атомной энергетике.

5. Ремонт воздуходувок, вакуум-насосов, центрифуг, вариаторов, редукторов всех видов с изготовлением запасных частей к ним.

6. Изготовление и испытание канатных строп грузоподъемностью от 0,32 до 8 тонн, цепных строп грузоподъемностью от 0,4 до 2,5 тонн, ремонт грузоподъемного оборудования (мостовые краны Q = 5...15 т, козловые краны Q = 8...32 т, электротельферы Q = 0,04...8 т, ручные тали Q = 1...8 т).

ОАО «Могилевхимволокно» активно привлекает инвестиции благодаря наличию производственных площадей на территории свободной экономической зоны «Могилев», работающей на основе специального таможенного и налогового режима, развитой инфраструктуры инженерных и транспортных коммуникаций, собственных энергетических мощностей, профессиональных кадров, а также больших производственных и складских помещений для создания новых производств.

## 1.2 Постановка задачи

Постановка задачи для разработки приложения на основе базы данных

Цель: Разработать приложение для управления процессами производства и контроля качества продукции на предприятии используя технологии баз данных, язык программирования C# и библиотеку React.

Основные функциональные требования:

1. Аутентификация и авторизация пользователей:

- Регистрация и вход пользователей с использованием ролей и разрешений (AspNetRoles, AspNetUsers, AspNetUserRoles, AspNetUserClaims).

2. Управление сотрудниками:

- Добавление, редактирование и удаление сотрудников (Employees).

- Просмотр списка сотрудников с возможностью фильтрации и сортировки.

3. Управление процессами производства:

- Добавление, редактирование и удаление процессов производства (ProductionProcesses).

- Просмотр списка процессов производства с возможностью фильтрации и сортировки.

4. Управление изделиями:

- Добавление, редактирование и удаление изделий (Products).

- Просмотр списка изделий с возможностью фильтрации и сортировки.

- Загрузка и хранение файлов, связанных с изделиями (ProductFile).

5. Управление заказами на производство:

- Добавление, редактирование и удаление заказов на производство (ProductionOrders).

- Просмотр списка заказов на производство с возможностью фильтрации и сортировки.

6. Управление выполнением процессов производства:

- Добавление, редактирование и удаление выполнения процессов производства (ProcessExecutions).

- Просмотр списка выполнения процессов производства с возможностью фильтрации и сортировки.

7. Управление контролем качества изделий:

- Добавление, редактирование и удаление контроля качества изделий (QualityControls).

- Просмотр списка контроля качества изделий с возможностью фильтрации и сортировки.

Технические требования:

- Использовать современные технологии и фреймворки для разработки приложения (например, React, Angular или Vue.js для фронтенда и ASP.NET Core или Node.js для бэкенда).

- Обеспечить безопасность данных и доступа к функциональным возможностям приложения.

- Реализовать адаптивный дизайн для корректного отображения на различных устройствах.

- Обеспечить интеграцию с предоставленной базой данных и корректную работу всех функциональных требований.

В результате выполнения задачи должно быть разработано приложение, обеспечивающее управление процессами производства и контроля качества изделий на предприятии, с интуитивно понятным интерфейсом.

Выполнение данного функционала значительно ускорит и упростит рабочий процесс, облегчит контроль за процессами производства. Произойдёт сокращение временных затрат за счёт автоматизации процессов.

# 2 Проектирование системы

## 2.1 Сущность задачи

В современном мире, где технологии развиваются стремительными темпами, эффективное управление процессами производства и контроль качества продукции становятся ключевыми факторами успеха любого производственного предприятия. Разработка информационной системы для мониторинга и контроля процессов производства и качества продукции позволит оптимизировать рабочие процессы, снизить издержки и повысить конкурентоспособность предприятия на рынке.

Основная цель разработки такой информационной системы заключается в автоматизации процесса сбора, обработки и анализа данных о производственных процессах и качестве выпускаемой продукции. Это позволит своевременно выявлять возможные проблемы и недостатки, предотвращать потери и ускорять процесс принятия решений на основе объективной и актуальной информации.

Проект разработки системы нацелен на обслуживание широкого спектра пользователей. Система предусмотрена для взаимодействия с различными сотрудниками предприятия, включая руководителя, инженера-конструктора, инженера-технолога, мастера производства, складского работника, бухгалтера, менеджера и других.

Исходя из сложности контроля на производстве из-за большого объема продукции, было решено создать систему, которая будет включать информацию обо всех деталях, такую как технологический процесс, чертежи и программы обработки. Кроме того, система будет обеспечивать хранение, формирование и печать внутренних накладных, а также передачу необходимой информации между сотрудниками.

Основным результатом работы системы является внутренний отчет о перемещении изделий на производственной площадке. В программе также будет доступна вся необходимая информация о деталях, заказах, отчетах, этапах производства изделий, сотрудниках и информации, передаваемой между отделами.

Таким образом, предлагаемая система будет служить универсальным инструментом для различных сотрудников предприятия, обеспечивая эффективный контроль и управление производственными процессами, а также обмен информацией между различными отделами и сотрудниками.

## 2.2 Структура системы

Система управления, разрабатываемая в данном дипломном проекте, представляет собой приложение и состоит из трех частей:

* клиентская часть;
* серверная часть;
* база данных.

Клиентская часть приложений, также известная как фронтенд, относится к тому разделу приложения, с которым взаимодействует пользователь. Она включает в себя пользовательский интерфейс (UI), дизайн, анимации и любые другие элементы, которые видит и с которыми взаимодействует пользователь на своем устройстве (компьютере, смартфоне и т. д.). Клиентская часть приложения обеспечивает представление данных, полученных от серверной части (бэкенда), и отправляет запросы на сервер для выполнения различных операций.

React — это популярная JavaScript-библиотека, разработанная Facebook, которая используется для создания пользовательских интерфейсов в клиентской части веб-приложений. React предоставляет эффективный и гибкий способ создания компонентов пользовательского интерфейса, управления состоянием приложения и обработки пользовательских событий.

React является одним из основных инструментов, используемых для разработки клиентской части веб-приложений, и позволяет разработчикам создавать масштабируемые, производительные и легко поддерживаемые приложения. Он позволяет быстро создавать сложные интерфейсы с использованием компонентного подхода, что облегчает разработку и тестирование.

В целом, React является важным инструментом для разработки клиентской части приложений, позволяя создавать динамические, интерактивные и производительные пользовательские интерфейсы.

Серверная часть приложений, также известная как бэкенд, относится к тому разделу приложения, который обрабатывает бизнес-логику, управление данными и взаимодействие с внешними сервисами. Она работает на сервере и отвечает за обработку запросов, полученных от клиентской части (фронтенда), выполнение необходимых операций и возврат данных обратно на клиентскую сторону. Серверная часть приложений включает в себя разработку API (Application Programming Interface), обработку данных, аутентификацию пользователей, авторизацию и интеграцию с базами данных и другими внешними сервисами.

C# (C-Sharp) — это объектно-ориентированный язык программирования, разработанный Microsoft, который может использоваться для разработки серверной части приложений. C# является частью платформы .NET, предоставляющей разработчикам множество инструментов и библиотек для создания мощных и надежных серверных приложений. С помощью C# можно создавать веб-приложения, работающие на сервере, веб-API, микросервисы, а также приложения для обработки данных и интеграции с базами данных.

Базы данных играют важную роль в серверной части приложений, поскольку они служат хранилищем информации, которую приложение обрабатывает и предоставляет клиентской части. Базы данных могут быть реляционными (например, MySQL, PostgreSQL, Microsoft SQL Server) или нереляционными (NoSQL, такими как MongoDB, Cassandra, Couchbase). Серверная часть приложения взаимодействует с базами данных для создания, чтения, обновления и удаления данных (CRUD-операции) в соответствии с запросами, полученными от клиентской части.

Таким образом, серверная часть приложений отвечает за обработку бизнес-логики, управление данными и интеграцию с внешними сервисами, а C# и базы данных являются ключевыми технологиями, используемыми для реализации этих функций.

На рисунке 2.1 изображена схема клиент-серверного приложения, которая отражает взаимодействие пользователя и блоков приложения между собой.

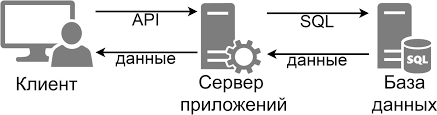


Рисунок 2.1 – Схемы клиент-серверного приложения

Пользователь взаимодействует с базой данных через интерфейс. Он совершает различные действия, такие как ввод данных в формы, нажатие на кнопки, выбор конкретных элементов и прочее. Интерфейс обрабатывает все пользовательские действия и отправляет соответствующий запрос на сервер. Сервер осуществляет доступ к базе данных с использованием языка SQL. Получив результат от базы данных в виде выборки определенных записей из таблиц, сервер отправляет ответ пользователю, который отображается с использованием интерфейса.

# 3 Анализ текущего состояния системы мониторинга и контроля за процессами производства

## 3.1 Описание процесса производства

Процессы производства на ОАО «Могилевхимволокно» можно разделить на несколько основных этапов:

1. Подготовка сырья: На первом этапе осуществляется приемка и подготовка сырья, такого как капролактам, полиэфир, полиамид и другие химические компоненты. Сырье тщательно проверяется на соответствие стандартам качества и хранится в специальных хранилищах до использования в производстве.

2. Полимеризация и экструзия: Второй этап включает в себя процесс полимеризации, в котором сырье превращается в полимеры, а затем в гранулы или порошок. Затем происходит процесс экструзии, в котором полимеры плавятся и превращаются в волокна или пленки с использованием специальных экструдеров.

3. Прядение и текстильная обработка: На третьем этапе синтетические волокна пропускаются через процесс прядения, в результате которого они превращаются в нити различной толщины и крутки. Затем нити подвергаются текстильной обработке, включая растяжение, кримпирование, закручивание и термообработку, чтобы придать им необходимые свойства и характеристики.

4. Отделка и контроль качества: На четвертом этапе готовые нити и пленки проходят процесс отделки, включающий окрашивание, покрытие специальными составами и другие технологические операции, придающие продукции конечный внешний вид и функциональные свойства. После отделки продукция проходит строгий контроль качества, включая испытания на прочность, устойчивость к воздействию внешних факторов и соответствие требованиям стандартов.

5. Упаковка и отгрузка: На заключительном этапе готовая продукция упаковывается в соответствии с требованиями клиентов и отправляется на склад для хранения до момента отгрузки.

Также можешь рассмотреть пример процессов в сборочно-сварочном цеху.

Процессы производства в сборочно-сварочном цеху на примере ремонта теплообменника включают несколько ключевых этапов, которые обеспечивают качественное и своевременное выполнение работ. Основные этапы ремонта теплообменника:

1. Диагностика и разборка: На первом этапе проводится диагностика теплообменника для выявления возможных дефектов и повреждений. Затем теплообменник разбирается на отдельные элементы, такие как трубы, корпус, фланцы и другие компоненты.

2. Очистка и подготовка деталей: После разборки теплообменника все его детали тщательно очищаются от грязи, накипи и других загрязнений. При необходимости проводится замена поврежденных деталей или их ремонт, например, проточка труб или восстановление поверхности фланцев.

3. Сборка и сварка: На этапе сборки и сварки специалисты сборочно-сварочного цеха осуществляют монтаж теплообменника, соединяя его компоненты с использованием сварки, заклепок или болтовых соединений. Важным аспектом этого этапа является контроль качества сварных швов и соединений, чтобы обеспечить надежность и долговечность теплообменника после ремонта.

4. Герметизация и проверка на герметичность: После сборки теплообменника проводится герметизация всех соединений и узлов, чтобы предотвратить возможные утечки рабочих сред. Затем теплообменник проверяется на герметичность с использованием специального оборудования, такого как вакуумные насосы, манометры и т. д.

5. Контроль качества и отгрузка: На заключительном этапе ремонта теплообменника проводится контроль качества выполненных работ, включая проверку сварных швов, соединений и герметичности. При положительном результате контроля качества теплообменник готов к отгрузке и дальнейшей эксплуатации на производстве.

Таким образом, процессы производства в сборочно-сварочном цеху на примере ремонта теплообменника включают в себя диагностику, разборку, очистку, сборку, сварку, герметизацию, проверку на герметичность и контроль качества, что обеспечивает качественный и надежный ремонт теплообменного оборудования.

## 3.2 Анализ существующих проблем и недостатков

Анализ существующих проблем и недостатков важен для определения областей, требующих улучшения и оптимизации. В контексте производственного предприятия, таких как сборочно-сварочный цех или ремонт теплообменников, возможные проблемы и недостатки могут включать следующие аспекты:

1. Неэффективность производственных процессов: Возможные проблемы с производственными процессами могут включать длительные простои, недостаточную автоматизацию, избыточные операции или несоответствие современным стандартам и технологиям.

2.Проблемы с качеством продукции: Недостатки в качестве выпускаемой продукции могут быть связаны с недостаточным контролем качества, использованием некачественного сырья или неправильными технологическими процессами.

3. Проблемы с оборудованием и инфраструктурой: Устаревшее или неисправное оборудование, недостаточное обслуживание и ремонт, а также несоответствие инфраструктуры требованиям производства могут привести к снижению производительности и возникновению проблем с качеством продукции.

4. Недостатки в управлении персоналом: Неэффективное управление персоналом может проявляться в недостаточной мотивации сотрудников, низкой производительности труда, проблемах с обучением и развитием персонала, а также недостаточной коммуникации между отделами и сотрудниками.

5. Проблемы с соблюдением норм безопасности и экологии: Несоблюдение требований по охране труда, промышленной безопасности и экологии может привести к возникновению аварийных ситуаций, ухудшению условий труда и негативному воздействию на окружающую среду.

Для выявления и устранения существующих проблем и недостатков необходимо провести комплексный анализ производственных процессов, качества продукции, состояния оборудования, управления персоналом и соблюдения норм безопасности и экологии. На основе полученных данных можно разработать систему улучшения и оптимизации производства, направленный на устранение выявленных проблем и повышение эффективности предприятия.

## 3.3 Определение требований к системе мониторинга и контроля за процессами производства

Определение требований к системе мониторинга и контроля за процессами производства и качеством продукции является важным этапом при разработке и внедрении такой системы. Основные требования к системе мониторинга и контроля могут включать следующие аспекты:

1. Охват всех производственных процессов: Система должна обеспечивать мониторинг и контроль всех ключевых этапов производства, начиная от подготовки сырья и заканчивая выпуском готовой продукции.

2. Реальное время: Система мониторинга и контроля должна работать в режиме реального времени, чтобы оперативно выявлять возможные проблемы и недостатки в процессе производства и обеспечивать своевременное принятие корректирующих мер.

3. Интеграция с оборудованием и инфраструктурой: Система должна быть интегрирована с производственным оборудованием и инфраструктурой, чтобы обеспечивать автоматический сбор и обработку данных о состоянии оборудования, параметрах процессов и качестве продукции.

4. Доступность и удобство интерфейса: Система мониторинга и контроля должна иметь простой и интуитивно понятный интерфейс, позволяющий сотрудникам легко получать информацию о процессах производства и качестве продукции, а также управлять системой и вносить необходимые изменения.

5. Гибкость и масштабируемость: Система должна быть гибкой и масштабируемой, чтобы обеспечивать возможность адаптации к изменениям в процессах производства, внедрению новых технологий и росту предприятия.

6. Аналитические возможности: Система мониторинга и контроля должна предоставлять возможности для анализа собранных данных, выявления тенденций и проблем, а также предоставления рекомендаций по оптимизации процессов и улучшению качества продукции.

7. Безопасность и защита данных: Система должна обеспечивать надежную защиту данных о производственных процессах и качестве продукции от несанкционированного доступа, а также предусматривать резервное копирование и восстановление данных в случае возникновения сбоев или аварий.

Выявление этих потребностей и целей позволяет разработать систему мониторинга и контроля за процессами производства и качеством продукции, которая будет эффективно соответствовать требованиям и достигать поставленных целей.

# 4 Реализация программного обеспечения

## 4.1 Выбор и обоснование средств реализации для разработки программного обеспечения

Выбор C# для реализации программного обеспечения системы мониторинга и контроля процессов производства и качества продукции может быть обоснован следующими причинами:

1. Мощная и гибкая платформа: C# является объектно-ориентированным языком программирования, разработанным Microsoft, и является частью платформы .NET. Эта платформа предоставляет разработчикам множество инструментов, библиотек и фреймворков для создания мощных и надежных приложений.

2. Производительность и эффективность: C# обеспечивает высокую производительность и эффективность благодаря компиляции в промежуточный язык (CIL), который затем выполняется в среде исполнения .NET с использованием Just-In-Time компиляции. Это позволяет программам на C# работать быстро и эффективно на различных платформах и архитектурах.

3. Поддержка кроссплатформенности: С использованием .NET Core или .NET 5 и выше, C# позволяет разрабатывать кросс-платформенные приложения, которые могут работать на различных операционных системах, таких как Windows, Linux и macOS. Это упрощает развертывание и поддержку системы мониторинга и контроля на различных платформах.

4. Богатые возможности для разработки веб-приложений и API: С использованием фреймворка ASP.NET Core, C# предоставляет мощные инструменты для разработки веб-приложений и API, которые могут быть использованы для создания пользовательского интерфейса и обеспечения взаимодействия с оборудованием и другими системами.

5. Интеграция с базами данных: C# предоставляет широкие возможности для работы с различными типами баз данных, такими как реляционные (SQL Server, PostgreSQL, MySQL) и нереляционные (MongoDB, Cassandra). Это позволяет легко интегрировать систему мониторинга и контроля с существующей инфраструктурой баз данных.

6. Обширное сообщество и поддержка: C# имеет большое и активное сообщество разработчиков, что обеспечивает доступ к множеству ресурсов, таких как документация, примеры кода, обучающие материалы и форумы. Это помогает ускорить разработку и решение возникающих проблем.

7. Язык с высоким уровнем абстракции: C# обладает высоким уровнем абстракции, что позволяет разработчикам сосредоточиться на решении конкретных задач, не заботясь о низкоуровневых деталях. Это упрощает разработку и сокращает время на создание системы мониторинга и контроля.

Исходя из вышеуказанных причин, C# представляется подходящим выбором для разработки программного обеспечения системы мониторинга и контроля процессов производства и качества продукции, обеспечивая мощные возможности, кроссплатформенность, интеграцию с базами данных и обширную поддержку со стороны сообщества разработчиков.

Выбор MS SQL Server для реализации программного обеспечения системы мониторинга и контроля процессов производства и качества продукции может быть обоснован следующими причинами:

1. Надежность и производительность: MS SQL Server является мощной и надежной системой управления реляционными базами данных (СУБД), разработанной Microsoft. Она обеспечивает высокую производительность и эффективность для обработки больших объемов данных и выполнения сложных запросов.

2. Интеграция с платформой .NET и C#: MS SQL Server тесно интегрирован с платформой .NET и языком программирования C#, что облегчает разработку и внедрение программного обеспечения, использующего эту СУБД. Это позволяет использовать единый стек технологий для разработки приложения и работы с данными.

3. Поддержка хранимых процедур и триггеров: MS SQL Server предоставляет возможности для создания хранимых процедур и триггеров, что позволяет реализовать сложную логику обработки данных на стороне сервера, улучшая производительность и обеспечивая централизованное управление бизнес-логикой.

4. Расширенные возможности анализа данных: MS SQL Server включает в себя инструменты для анализа данных, такие как интеграция с Reporting Services, Analysis Services и Power BI. Это позволяет создавать сложные отчеты и дашборды для мониторинга процессов производства и качества продукции.

5. Безопасность и контроль доступа: MS SQL Server предоставляет мощные средства для обеспечения безопасности данных и контроля доступа. Система поддерживает шифрование данных, аутентификацию пользователей, настройку прав доступа на уровне объектов баз данных и аудит операций с данными.

6. Масштабируемость и гибкость: MS SQL Server обеспечивает гибкость и масштабируемость, позволяя легко адаптировать систему мониторинга и контроля к изменениям в процессах производства, росту объемов данных и увеличению количества пользователей.

7. Поддержка кластеризации и репликации данных: MS SQL Server поддерживает кластеризацию и репликацию данных, что обеспечивает высокую доступность и надежность системы мониторинга и контроля, а также защиту от потери данных и сбоев оборудования.

Исходя из вышеуказанных причин, MS SQL Server представляется подходящим выбором для разработки программного обеспечения системы мониторинга и контроля процессов производства и качества продукции, обеспечивая надежность, производительность, интеграцию с платформой .NET и C#, а также расширенные возможности анализа данных и безопасности.

Выбор React для реализации программного обеспечения системы мониторинга и контроля процессов производства и качества продукции может быть обоснован следующими причинами:

1. Быстрая и эффективная разработка интерфейса: React является популярной и мощной библиотекой JavaScript для создания пользовательских интерфейсов, разработанной Facebook. React позволяет быстро и эффективно создавать сложные и динамичные интерфейсы с использованием компонентного подхода.

2. Оптимизация производительности: React использует виртуальный DOM для оптимизации работы с реальным DOM, что обеспечивает высокую производительность даже при работе с большими объемами данных и сложными интерфейсами.

3. Компонентный подход: React основан на идее разделения интерфейса на независимые компоненты, что облегчает разработку, тестирование и поддержку приложения. Компоненты могут быть легко переиспользованы и комбинированы для создания различных интерфейсов.

4. Широкое сообщество и поддержка: React имеет большое и активное сообщество разработчиков, что обеспечивает доступ к множеству ресурсов, таких как документация, примеры кода, обучающие материалы и форумы. Это помогает ускорить разработку и решение возникающих проблем.

5. Кроссплатформенность: React может быть использован для разработки веб-приложений, а также мобильных приложений с использованием React Native. Это позволяет создавать кросс-платформенные решения с единым кодом и стеком технологий.

6. Интеграция с другими технологиями: React легко интегрируется с другими технологиями и библиотеками, такими как Redux для управления состоянием приложения, React Router для реализации навигации, и Axios для работы с API и взаимодействия с сервером.

7. Современные методы разработки: React использует современные методы разработки, такие как JSX (синтаксис, объединяющий JavaScript и HTML), и поддерживает новейшие возможности ECMAScript, что упрощает разработку и повышает качество кода.

Исходя из вышеуказанных причин, React представляется подходящим выбором для разработки программного обеспечения системы мониторинга и контроля процессов производства и качества продукции, обеспечивая быструю и эффективную разработку интерфейса, высокую производительность, компонентный подход и кроссплатформенность.

## 4.2 Описание таблиц базы данных

Проектируемая реляционная база данных содержит в себе 18 таблиц. 7 таблиц основные. Остальные таблицы были созданы автоматически для обеспечения авторизации и аутентификации пользователя.

1. Таблица AspNetRoles:

- Id – идентификатор роли, строка

- Name – имя роли, строка

- NormalizedName – нормализованное имя роли, строка

- ConcurrencyStamp – метка согласованности, строка

2. Таблица Employees:

- Id – идентификатор сотрудника, целое число

- FirstName – имя сотрудника, строка

- LastName – фамилия сотрудника, строка

- Position – должность сотрудника, строка

- HireDate – дата приема на работу, дата и время

- Created – дата создания записи, дата и время

- CreatedBy – автор записи, строка

- LastModified – дата последнего изменения, дата и время

- LastModifiedBy – автор последнего изменения, строка

3. Таблица OpenIddictApplications:

- Id – идентификатор приложения, строка

- ClientId – идентификатор клиента, строка

- ClientSecret – секрет клиента, строка

- ConcurrencyToken – метка согласованности, строка

- ConsentType – тип согласия, строка

- DisplayName – отображаемое имя, строка

- DisplayNames – отображаемые имена, строка

- Permissions – разрешения, строка

- PostLogoutRedirectUris – URI перенаправления после выхода, строка

- Properties – свойства, строка

- RedirectUris – URI перенаправления, строка

- Requirements – требования, строка

- Type – тип, строка

4. Таблица OpenIddictScopes:

- Id – идентификатор области, строка

- ConcurrencyToken – метка согласованности, строка

- Description – описание, строка

- Descriptions – описания, строка

- DisplayName – отображаемое имя, строка

- DisplayNames – отображаемые имена, строка

- Name – имя, строка

- Properties – свойства, строка

- Resources – ресурсы, строка

5. Таблица ProductionProcesses:

- Id – идентификатор процесса производства, целое число

- Name – название процесса, строка

- Description – описание процесса, строка

- Duration – продолжительность процесса, время

- Created – дата создания записи, дата и время

- CreatedBy – автор записи, строка

- LastModified – дата последнего изменения, дата и время

- LastModifiedBy – автор последнего изменения, строка

6. Таблица Products:

- Id – идентификатор изделия, целое число

- Name – название изделия, строка

- Description – описание изделия, строка

- CreationDate – дата создания изделия, дата и время

- QualityStatus – статус качества, строка

- Created – дата создания записи, дата и время

- CreatedBy – автор записи, строка

- LastModified – дата последнего изменения, дата и время

- LastModifiedBy – автор последнего изменения, строка

7. Таблица AspNetRoleClaims:

- Id – идентификатор претензии роли, целое число

- RoleId – идентификатор роли, строка

- ClaimType – тип претензии, строка

- ClaimValue – значение претензии, строка

8. Таблица AspNetUsers:

- Id – идентификатор пользователя, строка

- EmployeeId – идентификатор сотрудника, целое число

- UserName – имя пользователя, строка

- NormalizedUserName – нормализованное имя пользователя, строка

- Email – электронная почта, строка

- NormalizedEmail – нормализованная электронная почта, строка

- EmailConfirmed – подтверждение электронной почты, булево значение

- PasswordHash – хеш пароля, строка

- SecurityStamp – метка безопасности, строка

- ConcurrencyStamp – метка согласованности, строка

- PhoneNumber – номер телефона, строка

- PhoneNumberConfirmed – подтверждение номера телефона, булево значение

- TwoFactorEnabled – двухфакторная аутентификация, булево значение

- LockoutEnd – окончание блокировки, дата и время смещения

- LockoutEnabled – блокировка включена, булево значение

- AccessFailedCount – количество неудачных попыток доступа, целое число

9. Таблица ProductionOrders:

- Id – идентификатор заказа на производство, целое число

- OrderDate – дата заказа, дата и время

- Quantity – количество, целое число

- Deadline – крайний срок, дата и время

- EmployeeId – идентификатор сотрудника, целое число

- Created – дата создания записи, дата и время

- CreatedBy – автор записи, строка

- LastModified – дата последнего изменения, дата и время

- LastModifiedBy – автор последнего изменения, строка

10. Таблица OpenIddictAuthorizations:

- Id – идентификатор авторизации, строка

- ApplicationId – идентификатор приложения, строка

- ConcurrencyToken – метка согласованности, строка

- CreationDate – дата создания, дата и время

- Properties – свойства, строка

- Scopes – области, строка

- Status – статус, строка

- Subject – предмет, строка

- Type – тип, строка

11. Таблица ProcessExecutions:

- Id – идентификатор выполнения процесса, целое число

- StartTime – время начала, дата и время

- EndTime – время окончания, дата и время

- EmployeeId – идентификатор сотрудника, целое число

- ProcessId – идентификатор процесса, целое число

- ProductId – идентификатор продукта, целое число

- LineId – идентификатор линии, целое число

- Created – дата создания записи, дата и время

- CreatedBy – автор записи, строка

- LastModified – дата последнего изменения, дата и время

- LastModifiedBy – автор последнего изменения, строка

12. Таблица ProductFile:

- Id – идентификатор файла изделия, целое число

- FileName – имя файла, строка

- FileType – тип файла, целое число

- FileContent – содержимое файла, двоичные данные

- ProductId – идентификатор изделия, целое число

- Created – дата создания записи, дата и время

- CreatedBy – автор записи, строка

- LastModified – дата последнего изменения, дата и время

- LastModifiedBy – автор последнего изменения, строка

13. Таблица QualityControls:

- Id – идентификатор контроля качества, целое число

- ControlDate – дата контроля, дата и время

- Result – результат, строка

- Comment – комментарий, строка

- EmployeeId – идентификатор сотрудника, целое число

- ProductId – идентификатор изделия, целое число

- Created – дата создания записи, дата и время

- CreatedBy – автор записи, строка

- LastModified – дата последнего изменения, дата и время

- LastModifiedBy – автор последнего изменения, строка

14. Таблица AspNetUserClaims:

- Id – идентификатор претензии пользователя, целое число

- UserId – идентификатор пользователя, строка

- ClaimType – тип претензии, строка

- ClaimValue – значение претензии, строка

15. Таблица AspNetUserLogins:

- LoginProvider – провайдер входа, строка

- ProviderKey – ключ провайдера, строка

- ProviderDisplayName – отображаемое имя провайдера, строка

- UserId – идентификатор пользователя, строка

16. Таблица AspNetUserRoles:

- UserId – идентификатор пользователя, строка

- RoleId – идентификатор роли, строка

17. Таблица AspNetUserTokens:

- UserId – идентификатор пользователя, строка

- LoginProvider – провайдер входа, строка

- Name – имя, строка

- Value – значение, строка

18. Таблица OpenIddictTokens:

- Id – идентификатор токена, строка

- ApplicationId – идентификатор приложения, строка

- AuthorizationId – идентификатор авторизации, строка

- ConcurrencyToken – метка согласованности, строка

- CreationDate – дата создания, дата и время

- ExpirationDate – дата истечения, дата и время

- Payload – полезная нагрузка, строка

- Properties – свойства, строка

- RedemptionDate – дата выкупа, дата и время

- ReferenceId – идентификатор ссылки, строка

- Status – статус, строка

- Subject – предмет, строка

- Type – тип, строка

## 4.3 Реализация отдельных функций

### 4.3.1 Добавление новых записей в базу данных

При нажатии на кнопку «Добавить изделие» на главной странице «Изделия» открывается форма для добавления данных.

Код добавления записи в базу данных:

public record CreateProductCommand : IRequest<int>

{

public string Name { get; set; }

public string Description { get; set; }

public string QualityStatus { get; set; }

}

public class CreateProductCommandHandler : IRequestHandler<CreateProductCommand, int>

{

private readonly IGenericRepository<Product> \_repository;

public CreateProductCommandHandler(IGenericRepository<Product> repository)

{

\_repository = repository;

}

public async Task<int> Handle(CreateProductCommand request, CancellationToken cancellationToken)

{

var entity = new Product

{

Name = request.Name,

Description = request.Description,

QualityStatus = request.QualityStatus,

CreationDate = DateTime.Now

};

await \_repository.AddAsync(entity, cancellationToken);

return entity.Id;

}

}

Также при добавлении данных производится проверка того верно ли пользователь ввел данные и не забыл ли он что-то заполнить.

Код проверки вводимых данных:

public class CreateProductCommandValidator : AbstractValidator<CreateProductCommand>

{

public CreateProductCommandValidator()

{

RuleFor(v => v.Name)

.MaximumLength(200)

.NotEmpty();

RuleFor(v => v.Description)

.MaximumLength(200)

.NotEmpty();

RuleFor(v => v.QualityStatus)

.MaximumLength(200)

.NotEmpty();

}

}

### 4.3.2 Поиск записей в базе данных

В базе данных должен производиться поиск информации по разным критериям.

Код поиска информации по изделию:

<TableBody>

{Array.isArray(products) &&

products

.filter((product) =>

product.name.toLowerCase().includes(search.toLowerCase())

)

.map((product) => (

<TableRow key={product.id}>

<TableCell>{product.name}</TableCell>

<TableCell>{product.description}</TableCell>

<TableCell>{product.qualityStatus}</TableCell>

<TableCell>{formatDate(product.creationDate)}</TableCell>

<TableCell style={{ whiteSpace: 'nowrap' }}>

<IconButton

onClick={() => {

setSelectedProduct(product);

setEditDialogOpen(true);

}}

>

<Edit />

</IconButton>

<IconButton

onClick={() => handleDeleteProduct(product.id)}

>

<Delete />

</IconButton>

<Button

variant="outlined"

onClick={() => navigate(`/processes/${product.id}`)}

>

Процессы

</Button>

</TableCell>

</TableRow>

))}

</TableBody>

public record GetProductsWithPaginationQuery : IRequest<PaginatedList<ProductVm>>

{

public int PageNumber { get; init; } = 1;

public int PageSize { get; init; } = 10;

}

public class GetProductsQueryHandler : IRequestHandler<GetProductsWithPaginationQuery, PaginatedList<ProductVm>>

{

private readonly IGenericRepository<Product> \_repository;

public GetProductsQueryHandler(IGenericRepository<Product> repository)

{

\_repository = repository;

}

public async Task<PaginatedList<ProductVm>> Handle(GetProductsWithPaginationQuery request, CancellationToken cancellationToken)

{

return await \_repository.GetAsync<Product, ProductVm>(request.PageNumber, request.PageSize, cancellationToken);

}

}

### 4.3.3 Удаление записей в базе данных

В базе данных производиться удаление записей по определенным критериям.

Код удаления записи об изделии из базы данных по уникальному идентификатору изделия:

[HttpDelete("{id}")]

public async Task<IActionResult> DeleteProduct(int id)

{

await \_mediator.Send(new DeleteProductCommand(id));

return NoContent();

}

public record DeleteProductCommand(int Id) : IRequest;

public class DeleteProductCommandHandler : IRequestHandler<DeleteProductCommand>

{

private readonly IGenericRepository<Product> \_repository;

public DeleteProductCommandHandler(IApplicationDbContext context, IGenericRepository<Product> repository)

{

\_repository = repository;

}

public async Task Handle(DeleteProductCommand request, CancellationToken cancellationToken)

{

var entity = await \_repository.GetByIdAsync(request.Id, cancellationToken);

await \_repository.DeleteAsync(entity, cancellationToken);

}

}

public async Task DeleteAsync(T entity, CancellationToken cancellationToken)

{

\_dbContext.Set<T>().Remove(entity);

await \_dbContext.SaveChangesAsync(cancellationToken);

}

### 4.3.4 Редактирование записей в базе данных

В базе данных производится редактирование записей по критерию идентификационный номер.

Код редактирования записей об изделиях:

[HttpPut("{id}")]

public async Task<IActionResult> UpdateProduct(int id, UpdateProductCommand command)

{

if (id != command.Id)

{

return BadRequest();

}

await \_mediator.Send(command);

return Ok();

}

public record UpdateProductCommand : IRequest

{

public int Id { get; set; }

public string Name { get; set; }

public string Description { get; set; }

public string QualityStatus { get; set; }

}

public class UpdateProductCommandHandler : IRequestHandler<UpdateProductCommand>

{

private readonly IGenericRepository<Product> \_repository;

public UpdateProductCommandHandler(IGenericRepository<Product> repository)

{

\_repository = repository;

}

public async Task Handle(UpdateProductCommand request, CancellationToken cancellationToken)

{

var entity = await \_repository.GetByIdAsync(request.Id, cancellationToken);

entity.Id = request.Id;

entity.Name = request.Name;

entity.Description = request.Description;

entity.QualityStatus = request.QualityStatus;

await \_repository.UpdateAsync(entity, cancellationToken);

}

}

public async Task UpdateAsync(T entity, CancellationToken cancellationToken)

{

\_dbContext.Set<T>().Update(entity);

await \_dbContext.SaveChangesAsync(cancellationToken);

}

### 4.3.5 Печать Отчета

В приложении есть возможность напечатать внутренний отчет. Код представлен ниже:

public async Task<GenerateReportVm> Handle(GenerateReportCommand request, CancellationToken cancellationToken)

{

var stream = new MemoryStream();

using (var document = SpreadsheetDocument.Create(stream, SpreadsheetDocumentType.Workbook))

{

var workbookPart = document.AddWorkbookPart();

workbookPart.Workbook = new Workbook();

var worksheetPart = workbookPart.AddNewPart<WorksheetPart>();

worksheetPart.Worksheet = new Worksheet(new SheetData());

var sheets = document.WorkbookPart.Workbook.AppendChild(new Sheets());

var sheet = new Sheet

{

Id = document.WorkbookPart.GetIdOfPart(worksheetPart),

SheetId = 1,

Name = "Sheet1"

};

sheets.Append(sheet);

var sheetData = worksheetPart.Worksheet.GetFirstChild<SheetData>();

Columns columns = new Columns();

uint columnCount = 6;

for (uint i = 1; i <= columnCount; i++)

{

Column column = new Column

{

Min = i,

Max = i,

BestFit = true,

CustomWidth = true,

Width = 30

};

columns.Append(column);

}

worksheetPart.Worksheet.InsertBefore(columns, sheetData);

var products = \_context.Products

.Include(x => x.ProcessExecutions)

.ThenInclude(x => x.ProductionProcess)

.Where(x => x.CreationDate > request.From && x.CreationDate < request.End);

uint index = 1;

var row1 = new Row { RowIndex = index++ };

row1.Append(new Cell

{

CellValue = new CellValue("Название"),

DataType = CellValues.String,

StyleIndex = 0,

});

row1.Append(new Cell

{

CellValue = new CellValue("Описание"),

DataType = CellValues.String,

});

row1.Append(new Cell

{

CellValue = new CellValue("Качество"),

DataType = CellValues.String,

});

row1.Append(new Cell

{

CellValue = new CellValue("Дата создания"),

DataType = CellValues.String,

});

row1.Append(new Cell

{

CellValue = new CellValue("Процессы"),

DataType = CellValues.String,

});

sheetData.Append(row1);

foreach (var product in products)

{

var row = new Row {RowIndex = index++};

row.Append(new Cell

{

CellValue = new CellValue(product.Name),

DataType = CellValues.String,

});

row.Append(new Cell

{

CellValue = new CellValue(product.Description),

DataType = CellValues.String,

});

row.Append(new Cell

{

CellValue = new CellValue(product.QualityStatus),

DataType = CellValues.String,

});

row.Append(new Cell

{

CellValue = new CellValue(product.CreationDate.ToString("MM-dd-yyyy")),

DataType = CellValues.String,

});

var processes = product.ProcessExecutions;

sheetData.Append(row);

foreach (var process in processes)

{

var nestedRow = new Row { RowIndex = index++ };

nestedRow.Append(new Cell());

nestedRow.Append(new Cell());

nestedRow.Append(new Cell());

nestedRow.Append(new Cell());

nestedRow.Append(new Cell

{

CellValue = new CellValue(process.ProductionProcess.Name),

DataType = CellValues.String,

});

sheetData.Append(nestedRow);

}

}

workbookPart.Workbook.Save();

}

stream.Position = 0;

string excelName = $"Отчет-({DateTime.Now.ToString("MM-dd-yyyy")}).xlsx";

return new GenerateReportVm

{

ExcelName = excelName,

Stream = stream

};

}

# 5 Инструкция к эксплуатации

При запуске приложения нужно пройти авторизацию введя логин и пароль (рисунок 5.1).

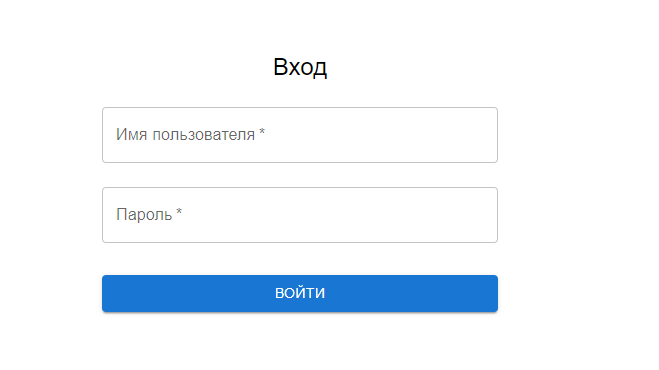


Рисунок 5.1 – Форма авторизации

Если пользователь ввел логин или пароль неправильно, то приложение выдаст ошибку (рисунок 5.2).

A screenshot of a login box

Description automatically generated with medium confidence

Рисунок 5.2 – Сообщение об ошибке

При верном вводе логина и пароля произойдет переход на главную форму (рисунок 5.3).

На странице «Изделия» можно добавить, отредактировать, удалить и посмотреть подробную информацию об изделиях.

При нажатии кнопки «Добавить» открывается форма добавления изделия (рисунок 5.4).

Чтобы добавить, например, процесс производства для изделия нажимаем кнопку «Процессы» и открывается страница «Процессы производства», где можем добавлять процессы производства для данного изделия (рисунок 5.5).

После выполнения процесса мы можем прикрепить фотографию (рисунок 5.6) и файл (рисунок 5.7) при необходимости.

Если ошибочно добавили процесс, то есть возможность его удалить. Для этого надо нажать на корзину и подтвердить удаление во всплывающем окне (рисунок 5.8).

A screenshot of a computer

Description automatically generated with medium confidence

Рисунок 5.3 – Главная форма

A screenshot of a computer

Description automatically generated with medium confidence

Рисунок 5.4 – Форма добавления

A screenshot of a computer

Description automatically generated with low confidence

Рисунок 5.5 – Страница процессов для изделия

A screenshot of a computer

Description automatically generated

Рисунок 5.6 – Выбор фотографии

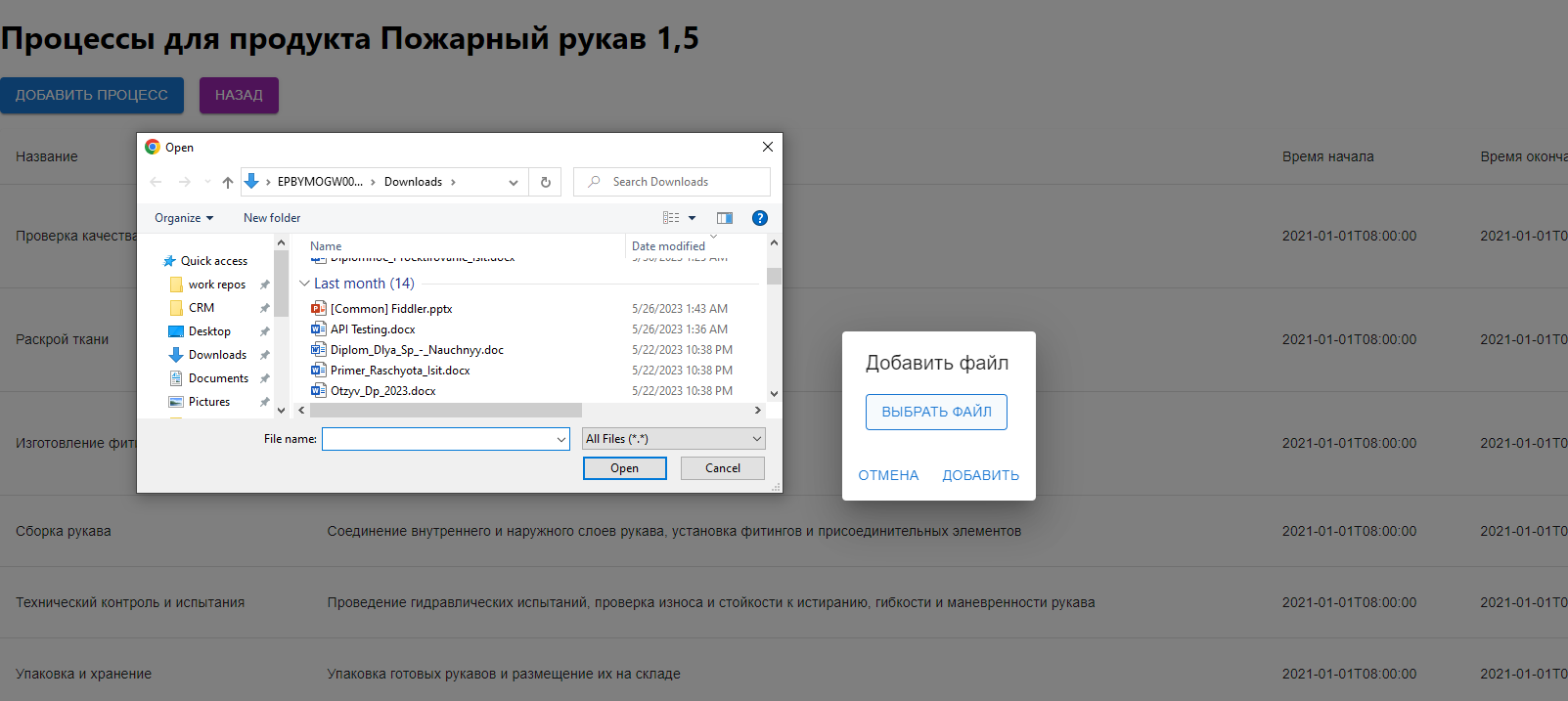


Рисунок 5.7 – Выбор файла

A screenshot of a computer

Description automatically generated with medium confidence

Рисунок 5.8 – Подтверждение удаления

A screenshot of a computer

Description automatically generated

Рисунок 5.9 – Генерация отчета

# 6 Организационно-экономическая часть

## 6.1 Организация технического контроля качества на предприятии

Одним из основных элементов системы управления качеством является организация технического контроля на предприятии.

Объектом контроля может быть продукция или процессы, связанные с ее созданием, хранением, транспортировкой, ремонтом, а также соответствующая техническая документация. Объект контроля имеет определенные характеристики, которые должны быть контролируемыми и могут иметь количественное или качественное значение [1].

Метод контроля определяет правила применения принципов и средств контроля. Он включает в себя технологию проведения контроля, контролируемые характеристики, средства контроля и точность выполнения контроля.

Организация и проведение технического контроля качества продукции являются важной частью системы управления качеством на всех этапах производства и реализации продукции.

Технический контроль — это проверка соответствия продукции или процесса, от которого зависит качество продукции, установленным стандартам или техническим требованиям [2].

Технический контроль является неотъемлемой частью производственного процесса и выполняется различными службами предприятия, в зависимости от объекта контроля. Например, служба нормоконтроля отвечает за контроль правильного использования стандартов, технических условий, руководящих материалов и другой нормативно-технической документации в процессе технической подготовки производства.

Отдел технического контроля (ОТК) осуществляет контроль качества готовой продукции и полуфабрикатов собственного производства. Однако исполнители и руководители производственных подразделений (цехов и участков) также несут ответственность за качество.

Основной задачей технического контроля на предприятии является своевременное получение полной и достоверной информации о качестве продукции, состоянии оборудования и технологического процесса. Это необходимо для предотвращения возможных неисправностей и отклонений, которые могут привести к нарушению стандартов и технических условий. Такой контроль помогает предупредить возникновение проблем и обеспечить соответствие продукции требованиям качества.

Функции технического контроля определяются во многом задачами и объектами производства. Это - контроль за качеством и комплектностью выпускаемых изделий, учет и анализ возвратов продукции, дефектов, брака, рекламаций и др.

Также к основным задачами отдела технического контроля (ОТК) относят предотвращение выпуска продукции, которая не соответствует требованиям стандартов, технических условий, эталонов, технической документации и договорных условий. Кроме того, ОТК также стремится укрепить производственную дисциплину и повысить ответственность всех звеньев производства за качество выпускаемой продукции.

Для того чтобы продукция предприятия могла быть реализована, она должна пройти приемку отделом технического контроля. Важно, чтобы приемка была подтверждена соответствующим документом, таким как сертификат, который удостоверяет качество продукции.

В соответствии с указанными задачами ОТК выполняет следующие функции: планирование и разработку методов обеспечения качества продукции, контроль и стимулирование качества.

Планирование и разработка методов обеспечения качества включают в себя: определение уровня качества продукции, планирование контроля качества и выбор необходимых технических средств для контроля; сбор информации о качестве, оценка затрат на обеспечение качества, обработка и анализ данных о качестве из производства и эксплуатации; управление качеством продукции со стороны поставщиков и управление качеством на предприятии; разработка методик контроля, которые обеспечивают сравнимость и надежность результатов контроля качества; разработка технических условий, условий и стандартов совместно с техническими подразделениями для обеспечения управления качеством продукции.

Контроль качества включает следующие аспекты [3]:

- Входной контроль качества: проверка качества сырья, основных и вспомогательных материалов, полуфабрикатов, комплектующих изделий, инструментов и других материалов, поступающих на склады предприятия;

- Производственный пооперационный контроль: осуществление контроля за соблюдением установленного технологического режима в процессе производства, а также в некоторых случаях проведение межоперационной приемки продукции;

- Систематический контроль: проведение проверок состояния оборудования, машин, режущих и измерительных инструментов, контрольно-измерительных приборов, прецизионных средств измерения, штампов, моделей испытательной аппаратуры, весового хозяйства, приспособлений, условий производства и транспортировки изделий и других элементов.

- Контроль моделей и опытных образцов: проверка качества готовой продукции, включая детали, мелкие сборочные единицы, подузлы, узлы, блоки, изделия, с использованием моделей и опытных образцов.

Стимулирование качества состоит из следующих компонентов:

- Разработка документации: создание документов, которые описывают методы и инструменты для мотивации в области обеспечения качества продукции. Эти документы могут включать политику качества, стандарты работы, процедуры и руководства, направленные на поощрение и поддержку достижения высокого уровня качества;

- Разработка положений о премировании: совместно с отделом организации труда и заработной платы разрабатываются положения, которые определяют систему вознаграждения и премирования работников предприятия за качественную работу. Эти положения могут включать критерии оценки качества работы, условия получения премий и механизмы их распределения;

- Обучение и повышение квалификации: предоставление обучения и программ повышения квалификации сотрудникам, чтобы развивать их навыки и знания в области обеспечения качества. Это может включать проведение тренингов, семинаров, курсов обучения и других форм образовательных мероприятий, которые помогут работникам повысить свою экспертизу и эффективность в области качества.

Отдел технического контроля (ОТК) на предприятии возглавляется начальником, который подчиняется напрямую директору предприятия. Начальник ОТК имеет полномочия прекратить приемочный контроль продукции, если она имеет повторяющиеся дефекты, до тех пор, пока не будут устранены причины, вызывающие эти дефекты. Также он имеет право запретить использование сырья, материалов, комплектующих изделий и инструментов, которые не соответствуют установленным требованиям для производства новой продукции.

В случае возникновения брака начальник ОТК предъявляет обязательные требования к подразделениям и должностным лицам предприятия по устранению причин дефектов продукции. Он также представляет руководству предприятия предложения о привлечении к ответственности должностных лиц и рабочих предприятия, которые виновны в производстве бракованной продукции. Начальник ОТК, наравне с директором и главным инженером предприятия, несет ответственность за выпуск недоброкачественной или несоответствующей стандартам и техническим условиям продукции.

Организация и структура отдела технического контроля (ОТК) на предприятии разрабатывается с учетом типовой структуры, а также учитываются задачи, функции и особенности производства. Обычно в составе отдела формируются следующие подразделения:

- Бюро технического контроля, которое размещается территориально в основных и вспомогательных цехах и отвечает за контроль качества в процессе производства;

- Бюро внешней приемки, которое осуществляет контроль качества поступающих материалов и комплектующих изделий на входе в предприятие;

- Бюро заключительного контроля и испытаний готовой продукции, которое осуществляет контроль и испытания готовых изделий перед их отгрузкой или реализацией;

- Бюро анализа и учета брака и рекламации, которое занимается анализом и учетом бракованной продукции и рассмотрением рекламаций от клиентов;

- Центрально-измерительная лаборатория и контрольно-поверочные пункты, которые осуществляют контроль состояния элементов и оснастки, используемых при контроле качества;

- Инспекторская группа, которая проводит проверочный контроль качества продукции и осуществляет целевые проверки соблюдения технологической дисциплины;

- Подразделения контроля экспортной продукции, которые занимаются контролем качества продукции, предназначенной для экспорта;

- Подразделение контроля качества лома и отходов цветных и благородных металлов, которое осуществляет контроль качества вторичного сырья.

Структурные подразделения ОТК определены на рисунке 6.1.1 [4].

A screenshot of a phone

Description automatically generated with medium confidence

Рисунок 6.1.1 – Структурные подразделения ОТК

Эти подразделения составляют структуру отдела технического контроля и выполняют свои функции в рамках обеспечения качества продукции на предприятии.

В зависимости от конкретных задач, стоящих перед техническим контролем, проводимом на промышленном предприятии используются различные виды и методы контроля (рис.6.1.2)

A picture containing text, receipt, font, screenshot

Description automatically generated

Рисунок 6.1.2 – Виды технического контроля

К настоящему времени сложились разнообразные методы контроля качества, которые можно разбить на две группы [5]:

- Самопроверка или самоконтроль - персональная проверка и контроль оператором с применением методов, установленных технологической картой на операцию, а также с использованием предусмотренных измерительных средств с соблюдением заданной периодичности проверки.

- Ревизия (проверка) - проверка, осуществляемая контролером, которая должна соответствовать содержанию карты контроля технологического процесса;

Усовершенствование работы отделов и управлений технического контроля на предприятиях должно включать следующие основные задачи:

- Разработка и внедрение мероприятий по предотвращению брака в производстве, снижению отклонений от утвержденных технологических процессов и предупреждению сбоев, которые могут повлиять на качество выпускаемой продукции;

- Внедрение прогрессивных методов и средств технического контроля, которые способствуют повышению производительности и эффективности работы контролеров ОТК;

- Объективный учет и комплексная дифференцированная оценка качества труда сотрудников контрольной службы, а также проверка достоверности результатов контроля;

- Подготовка необходимых данных для централизованной обработки информации о текущем состоянии и изменениях в основных условиях и предпосылках производства высококачественной продукции, а также информации о достигнутом уровне качества выпускаемой продукции;

Расширение внедрения самоконтроля основных производственных рабочих путем формирования перечня технологических операций, подлежащих самоконтролю, обеспечение необходимыми контрольно-измерительными приборами, инструментом, оснасткой и документацией, специальное обучение рабочих, про

- Проведение специальных исследований динамики качества изделий в процессе их эксплуатации, предполагающих организацию эффективной информационной взаимосвязи между поставщиками и потребителями по вопросам качества продукции;

- Планирование и технико-экономический анализ различных аспектов деятельности службы контроля качества продукции;

- Координация работы всех структурных подразделений отделов и управлений технического контроля предприятия;

- Периодическое определение абсолютной величины и динамики затрат на контроль качества продукции, влияния профилактичности, достоверности и экономичности технического контроля на качество изделий и основные показатели деятельности предприятия, оценка эффективности работы контрольной службы [6].

6.2 Обоснование эффективности технических решений

6.2.1 Определение объёма программного средства

Целью дипломного проектирования является разработка информационной системы мониторинга и контроля за процессами производства и качеством продукции, в ходе которой будет разработано приложение и база данных (ОАО «Могилевхиволокно»).

Разработанная база данных позволит ускорить документооборот, автоматизировать процессы вычисления, хранить большие объемы данных, среди которых можно легко найти необходимую информацию. Также к преимуществам этой системы можно отнести повышение скорости и качества работы сотрудников.

Разработка ПО предусматривает проведение всех стадий проектирования в соответствии с Постановлением Министерства труда и социальной защиты РБ №91 от 27.06.2007 г. «Об утверждении укрупненных норм затрат труда на разработку программного обеспечения».

Общий объем программного средства определяется на основе информации о функциях разрабатываемого программного обеспечения, исходя из количества и объема функций по формуле:

|  |  |  |
| --- | --- | --- |
|  |  | (6.1) |

где n − общее число функций,

− объем i-ой функции ПО (количество строк исходного кода (LOC).

С учетом условий разработки общий объем программного обеспечения уточняется в организации и определяется уточненный объем ПО по формуле:

|  |  |
| --- | --- |
|  | (6.2) |

где − уточненный объем i-й функции ПО (LОС).

Определение общего объема программного обеспечения в соответствии с содержанием функций представлено в таблице 6.1.

Таблица 6.1 – Перечень и объём функций программного средства

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № функции | Наименование (содержание) функции | | Объём функции строк исходного кода (LOC) | |
| по каталогу Vi | Уточненный Vyi |
| 1 | 2 | | 3 | 4 |
| Ввод, анализ входной информации, генерация кодов и процессор входного языка | | | | |
| 101 | Организация ввода информации | | 130 | 296 |
| 102 | Контроль, предварительная обработка и ввод информации | | 490 | 402 |
| 104 | Обработка входного заказа и формирование таблиц | | 1 040 | 356 |
| 109 | Упраление вводом/выводом | | 1 970 | 440 |
| Формирование, введение и обслуживание баз данных | | | | |
| 201 | | Генерация структуры базы данных | 3 500 | 326 |
| 206 | | Манипулирование данными | 7 860 | 292 |
| 207 | | Организация поиска и поиск в базе данных | 4 720 | 236 |
| 209 | | Загрузки базы данных | 2 360 | 371 |
| Формирование и обработка файлов | | | | |
| 305 | | Формирование файла | 2 130 | 292 |
| Расчетные задачи, формирование и вывод на внешние носители документов сложной формы и файлов | | | | |
| 703 | Расчет показателей | | 420 | 190 |
| Итого: | | | 24 620 | 3201 |

Расчет общего объёма программного средства:

V0 =1030+ 490+ 1040+ 1970+ 3500+ 7860+ 4720+2360+2130+420 = 24620 LOС.

Расчёт уточненного объёма программного средства:

Vу =296 + 402 + 356 + 440 +326 + 292 + 236+371+292+190 = 3 201 LOС.

В связи с использованием более совершенных средств автоматизации объемы функций были уменьшены и уточненный объем программного средства составил 3 201 LОС вместо 24 620 LОС.

6.2.2 Расчёт нормативной трудоёмкости

Уточненный объем программного средства составляет 3 501 LOC.

Всё программное обеспечение в зависимости от характеристик подразделяется на три категории сложности. Данное ПО относится к типу ПО функционального назначения и предназначено для повышения производительности труда пользователей, относится к 3-ей категории сложности (Приложение 2 Постановления Министерства труда и социальной защиты РБ №91 от 27.06.2007 г. «Об утверждении укрупненных норм затрат труда на разработку программного обеспечения»).

Согласно уточненного объема программного средства и категории сложности нормативная трудоемкость разработки ПО (Тн) составляет 140 чел./дн.

Дополнительный коэффициент сложности выбирается для ПО, обеспечивающего хранение, ведение и поиск в данных в сложных структурах, и принимается равным:

Кс=1+0,07 = 1,07

По степени новизны ПО относится к группе «В» с коэффициентом Кн = 0,63 и являющееся развитием определенного параметрического ряда ПО.

ПО разрабатывается для всех современных девайсов, охватывает реали-зуемые функции стандартными модулями от 60% и выше, поэтому коэффици-ент использования типовых программ принимается равным 0,55 (Кт = 0,55) (Приложение 6).

Новизне ПО категории «В» с применением CASE-технологий соответ-ствует следующее распределение трудоемкости по стадиям:

dТЗ + dЭП + dТП = 0,50; dРП = 0,35; dВН = 0,15,

где ТЗ – техническое задание,

ЭП – эскизный проект,

ТП – технический проект,

РП – рабочий проект,

ВН – ввод в действие.

Общая трудоемкость разработки ПО (То) определяется суммированием нормативной (скорректированной) трудоемкости разработки ПО по стадиям разработки:

|  |  |
| --- | --- |
|  | (6.3) |

где – нормативная (скорректированная) трудоемкость разработки ПО на *i-*й стадии (человеко-дней),

*n –* количество стадий разработки.

Нормативная трудоемкость разработки ПО рассчитывается по формуле:

|  |  |
| --- | --- |
| , | (6.4) |

где - коэффициент удельного веса трудоемкости *i-*й стадии разработки ПО,

*Тн* – нормативная трудоемкость, норма-час;

– коэффициент сложности ПО,

– коэффициент, учитывающий степень использования стандартных модулей,

– коэффициент, учитывающий новизну ПО.

Распределение нормативной трудоемкости ПО определяется по формуле:

,

где Тн - нормативная трудоемкость разработки ПО.

чел/дн.

чел/дн.

чел/дн.

Общий расчет показателей представлен в таблице 6.2.

Таблица 5.2 – Расчёт общей трудоёмкости и сроков разработки ПО с учётом стадий

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Показатели | Стадии | | | Итого |
| ТЗ + ЭП + ТП | РП | ВН |
| 1 | 2 | 3 | 4 | 5 |
| 1. Коэффициент удельных весов трудоёмкости стадии разработки ПО (d) | 0,5 | 0,35 | 0,15 | 1 |
| 2. Распределение нормативной трудоемкости ПО (Тн) по стадиям, чел/дн | 70 | 49 | 21 | 140 |
| 3. Коэффициент сложности ПО (Кс) | 1,07 | 1,07 | 1,07 | - |
| 4. Коэффициент, учитывающий использование стандартных модулей (Кт) | 0,55 | 0,55 | 0,55 | - |
| 5. Коэффициент, учитывающий новизну ПО (Кн) | 0,63 | 0,63 | 0,63 | - |
| 6. Общая трудоёмкость ПО (То), чел/дн | 26 | 18,2 | 7,8 | 52 |
| 7. Численность исполнителей, чел (Чи) | 1 | 1 | 1 | 1 |
| 8. Сроки разработки, лет | 0,071 | 0,05 | 0,021 | 0,142 |

Таким образом, согласно данным из приведённой выше таблицы, общая трудоёмкость ПО составит 52 чел./дн, а сроки разработки – 0,142 года.

6.2.3 Расчёт плановой себестоимости ПО

Материалы и комплектующие

В статье «Материалы и комплектующие» (М) отражаются затраты на материалы и принадлежности: носители, бумагу, красящие ленты и другие. Затраты определяются по действующим отпускным ценам путём перемножения количества материалов и цены за единицу.

|  |  |
| --- | --- |
|  | (6.5) |

где *Ki* – количество i-го материала,

*Ci* – стоимость единицы i-го материала, р;

*n* – количество разновидностей материалов.

Расчёты затрат на материалы и принадлежности приведены в таблице 6.3

р.

Таблица 6.3 – Расчёт затрат на материалы, необходимые для разработки ПО

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № п/п | Наименование материалов | Количество, шт | Цена за единицу, р. | Сумма, р. |
| 1 | Бумага А4 | 150 | 0,03 | 4,5 |
| 2 | Бумага А1 | 8 | 0,96 | 6,72 |
| 3 | Тонер Hi-Black универсальный для принтера Hp lj1005 | 0,3 | 40 | 12 |
| Итого: | | | | 23,22 |

Приложение было написано с помощью Visual Studio Community, который предоставляется разработчиком компанией Microsoft без лицензии.

Таким образом, затраты на материалы, необходимые для разработки программного обеспечения составляют 23,22 р.

Электроэнергия

В статье «Электроэнергия» (Э) отражаются затраты на электроэнергию, которые находятся исходя из продолжительности периода разработки изде-лия, количества кВт\*ч, затраченных на его проектирование и тарифа за 1 кВт\*ч по следующей формуле:

|  |  |
| --- | --- |
|  | (6.6) |

где *Kэ* – стоимость 1 кВт\*ч,

*Тэ* – количество затраченных на проектирование кВт\*ч,

Д – количество дней работы техники, дн.

Базовый тариф для прочих потребителей с 01.01.2023 г. составляет 0,43912 руб. за 1 кВт\*ч (согласно приказа Министерства антимонопольного регулирования и торговли Республики Беларусь от 08.02.2023 №34).

Потребление электричества ноутбуком в смешанном режиме работы процессора составит приблизительно 60 Вт\*ч или 0,060кВт\*ч. Теперь умножаем полученное значение на время работы. Компьютер включен и работает по 8 часов в день. Соответственно 0,060\*8=0,48 кВт\*ч. Умножив ежедневные энергозатраты на количество дней, необходимых для разработки проекта, и на стоимость 1 кВт\*ч по государственным тарифам оплаты за потребленное электричество получим:

Э = 0,43912  0,48 52 =10,94 р.

Основная заработная плата

Основная заработная плата – это вознаграждение за выполненную работу в соответствии с установленными нормами труда (нормы времени, выработки, обслуживания, должностные обязанности). Она устанавливается в виде тарифных ставок (окладов) и сдельных расценок для рабочих и должностных окладов – для служащих.

Статья «Основная заработная плата» (*Зосн*) рассчитывается по повременно-премиальной системе оплаты труда:

|  |  |
| --- | --- |
|  | (6.7) |

где *Tст1* – тарифная ставка 1-го разряда (согласно организации «» на 01.01.2023 г. составляет 580 р.),

*Ктар* – тарифный коэффициент (выбирается из единой тарифной сетки в зависимости от категории работника и присваиваемого ему разряда),

*Кр* –количество рабочих дней, дн;

*Д* – число фактически отработанных дней, дн.

Штатное расписание – организационно-распорядительный документ, определяющий структуру, штатный состав и штатную численность организации в соответствии с её Уставом (Положением), содержит перечень структур-ных подразделений, наименование должностей, специальностей, профессий с указанием квалификации, сведения о количестве штатных единиц.

Выписка из штатного расписания отображена в таблице 6.4.

Таблица 6.4 – Расчет фонда заработной платы

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Должность, профессия | Категория должности, профессии | Кол-во штатных единиц | Тарифные разряды по ТС | Тариф-е коэф-ты по ТС (кратные тариф. ставки 1-го разр.) | Коэф. повыш. Тарифных ставок по технолог.видам работ | Тарифный оклад в руб.,(тарифная ставка), в руб.согласно ТС с учётом коэф.повыш.по тенолог.вид. | Повышение тарифных ставок (окладов), предусмотренные коллективным договором | | | | | | | | Оклад (должн-ой оклад), (тарифная ставка) в руб., с учетом повыш. |
| за квали-фикационную кат-ю | | за сложн. работ и по др.основаниям | | за прим. произв-ой старший | | за исп-е дез-х ср-в | |
| % | руб. | % | руб. | % | руб. | % | руб. |
| Инженер-программист | спец. | 1 | 11 | 2,65 | – | 600 | – | – | – | – | – | – | – | – | 1590 |

Следовательно,



Дополнительная заработная плата

Статья «Дополнительная заработная плата» (*Здоп*) включает выплаты, предусмотренные законодательством о труде (оплата отпусков, льготных часов, времени выполнения государственных обязанностей и других выплат, не связанных с основной деятельностью исполнителей, сверхурочные, внезапные работы и др.), и определяется по нормативу в процентах к основной заработной плате:

|  |  |
| --- | --- |
|  | (6.8) |

где *Здоп* – дополнительная заработная плата исполнителей,

*Ндоп* – норматив дополнительной заработной платы (принимается в пределах 20% от основной заработной платы).



Отчисления от заработной платы

Уплата обязательных страховых взносов в ФСЗН производится коммерческими организациями единым платежом в размере 34% от фонда заработной платы и 1% из заработной платы работников:

|  |  |
| --- | --- |
|  | (6.9) |

где *Зсз* – отчисления в Фонд социальной защиты населения,

*Нсз* – норматив отчислений.



Порядок и условия обязательного страхования от несчастных случаев на производстве и профессиональных заболеваний определены Указом Президента Республики Беларусь «О страховой деятельности» от 25 августа 2006г. №530 (с изм. и доп. от 14 апреля 2014г. №165):

|  |  |
| --- | --- |
|  | (6.10) |

где *Знс* – отчисления на обязательное страхование от несчастных случаев на производстве и профессиональных заболеваний,

*Ннс* – норматив отчислений.



Амортизация основных средств и нематериальных активов

В статье «Амортизация основных средств и нематериальных активов» учитывается возмещение износа использованных в процессе проектирования средств труда. К основным средствам на данном этапе относится используемые электронные вычислительные машины и многофункциональные устройства, к нематериальным активам – программное обеспечение.

Существует три способа начисления амортизации согласно Инструкции о порядке начисления амортизации основных средств и нематериальных активов, утверждённой Постановлением Министерства экономики Республики Беларусь, Министерства финансов Республики Беларусь и Министерства архитектуры и строительства Республики Беларусь №37/18/6 от 27.02.2009 г.:

* линейный – в зависимости от полезного срока использования (пропорционально);
* нелинейный – метод суммы чисел лет и метод уменьшаемого остатка;
* производительный – в зависимости от объёма выпускаемой продукции (пропорционально).

Для расчёта амортизационных отчислений в дипломном проекте выбран линейный способ, который заключается в равномерном начислении амортизации.

Годовая норма амортизации – установленный размер амортизационных отчислений на полное восстановление, выраженное в %. Рассчитывается данный показатель по формуле:

|  |  |
| --- | --- |
|  | (6.11) |

где *На­* – норма амортизации, %;

*Тн* – нормативный срок службы, лет.

Годовая сумма амортизационных отчислений находится по формуле:

|  |  |
| --- | --- |
|  | (6.12) |

где *АОпер.ед.* – амортизационные отчисления на единицу на период проведения, р.;

*Фэ* – эффективный фонд рабочего времени (находится как календарный фонд времени за вычетом выходных и праздничных дней, а также простоев, в 2023 г. составляет 252 дня), дн;

*Д* – количество дней, затраченных на разработку изделия, дн.

Примем годовую норму амортизации 20%.

Таблица 6.5 – Расчёт амортизации основных средств

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование основных средств | Первоначальная стоимость, р. | Годовые амортизационные отчисления на единицу, р. | Период, дн. | Суммарные амортизационные отчисления на период проведения, р. |
| [Ноутбук Asus Vivobook 16X M1603QA-MB224](https://5element.by/products/779041-noutbuk-asus-vivobook-16x-m1603qa-mb224) | 1999 | 399,8 | 55 | 87,26 |
| [Принтер PANTUM P2500W](https://5element.by/products/545787-printer-pantum-p2500w) | 360 | 72 | 1 | 0,29 |
| Мышь LOGITECH M190 (L910-005906) | 37 | 7,4 | 55 | 1,60 |
| Всего | 2406 | 471,8 |  | 89,15 |

Примечание – Источник: каталог товаров компьютерного интернет-магазина 5element.by.

Таким образом суммарные амортизационные отчисления составляют 89,15 р.

Расходы на специальное оборудование

Расходы на спецоборудование включает затраты средств на приобретение вспомогательных технических и программных средств специального назначения, необходимых для разработки конкретного ПО, включая расходы на их проектирование, изготовление, отладку, установку и эксплуатацию. Затраты по этой статье определяются в соответствии со сметой расходов, которая составляется перед разработкой ПО.

Данная статья включается в смету расходов на разработку ПО в том случае, когда приобретаются специальное оборудование или специальные программы, предназначенные для разработки и создания только данного ПО.

При разработке ПО не использовались вспомогательные технические и платные программные средства.

Накладные расходы

Накладные расходы (*Рн*) связаны с необходимостью содержания аппарата управления, вспомогательных хозяйств и опытных производств, а также с расходами на общехозяйственные нужды (канцелярские расходы, командировочные расходы, расходы на телефонную связь и др.). Данные расходы берутся в процентном соотношении от основной заработной платы:

|  |  |
| --- | --- |
|  | (6.13) |

где *Рн* – накладные расходы на разрабатываемое изделие,

*Нр* – норматив накладных расходов (10%).



Прочие прямые расходы

Статья «Прочие прямые расходы» (*Рп*) на конкретное программное обеспечение включает затраты: на оплату услуг связи, Интернета, транспортные расходы, канцтовары, приобретение и подготовку специальной научно-технической информации и специальной литературы.

Данные расходы не предусмотрены при разработке ПО.

Общая сумма расходов по смете (плановая себестоимость)

Общая сумма расходов по смете (плановая себестоимость) (С) на ПО рассчитывается по формуле:

|  |  |
| --- | --- |
|  | (6.14) |

С = 23,22+ 10,94 +4 164,29+ 832,85+ 1 749+ 29,98+ 89,15+ 416,43= 7 315,87 р.

Общая сумма расходов по смете (плановая себестоимость) ожидается равной 723,35 руб. Сводка полученных результатов представлена в таблице 6.6.

Таблица 6.6 – Расчёт плановой себестоимости программного обеспечения

|  |  |
| --- | --- |
| Статья затрат | Затраты, р. |
| 1. Материалы и комплектующие *(М)* | 23,22 |
| 2. Электроэнергия *(Э)* | 10,94 |
| 3. Основная заработная плата исполнителей *(Зо)* | 4 164,29 |
| 4. Дополнительная заработная плата исполнителей *(Зд)* | 832,85 |
| 5. Отчисления: в ФСЗН 34%; от несчастных случаев 0,6% | 1 749 |
| 29,98 |
| 6. Амортизация *(А)* | 89,15 |
| 7. Расходы на спецоборудование *(Рс)* | – |
| 8. Накладные расходы *(Рн)* | 416,43 |
| 9. Прочие прямые расходы *(Рп)* | – |
| 10. Общая сумма расходов по смете (плановая себестоимость) *(С)* | 7 315,87 |

6.2.4 Расчет показателей экономической эффективности ПО

Эффективность – это отношение эффекта от внедрения автоматизированного рабочего места к затратам на его создание.

Эффект от использования программного продукта определяется скоростью обработки больших объемов информации, простотой освоения и эксплуатации персоналом различной квалификации. При этом изменяются условия труда работников, изменяется структура производственного персонала (меняется численность занятых на работах, требующих высшего или среднего специального образования; численность работников по разрядам работающих, подлежащих обучению, переобучению, повышению квалификации). В конечном счете, эффект выражается экономией материальных и трудовых ресурсов в стоимостном выражении за установленный период времени, обычно за год.

Расчет показателей экономической эффективности ПО для разработчика

У разработчика программного средства экономический эффект выступает в виде чистой прибыли, остающейся в распоряжении разработчика.

Для определения цены ПО необходимо рассчитать плановую прибыль. Прибыль рассчитывается по формуле:

|  |  |
| --- | --- |
|  | (6.15) |

где П – плановая прибыль от реализации ПО, р;

Сп – плановая себестоимость, р;

R – норматив рентабельности (30%), %.



Рентабельность и прибыль ПО, определяется исходя из результатов анализа рыночных условий, переговоров с заказчиком и согласования с ним отпускной цены.

После расчета прибыли от реализации определяется прогнозируемая цена ПО без налогов.

|  |  |
| --- | --- |
|  | (6.16) |

где *ЦП* – прогнозируемая цена без налогов.



Отпускная цена (цена реализации) ПО включает налог на добавленную стоимость (в настоящее время НДС = 20%).

|  |  |
| --- | --- |
|  | (6.17) |

НДС рассчитывается по формуле:

|  |  |
| --- | --- |
|  | (6.18) |

где *ЦП* – прогнозируемая цена без налогов, р;

*НДС* – налог на добавленную стоимость, р.





Прибыль от реализации ПО за вычетом налога на прибыль является чистой прибылью, остается организации разработчику и представляет собой экономический эффект от создания нового программного продукта.

|  |  |
| --- | --- |
|  | (6.19) |

где Нп – ставка налога на прибыль в настоящее время (Нп = 18%).



Сведём полученные данные в таблице 5.7.

Таблица 5.7 – Показатели экономической эффективности ПО

|  |  |
| --- | --- |
| Наименование статей затрат | Сумма, р. |
| 1.Полная себестоимость | 7 315,87 |
| 2.Прибыль | 2194,76 |
| 3.Цена без НДС | 9 510,63 |
| 4.НДС | 1 902,12 |
| 5.Отпускная цена | 11 412,76 |
| 6.Чистая прибыль | 1 799,70 |

Для создания ПО необходим начальный капитал, который инвестирован и его количество рассчитывается исходя из плановой себестоимости, увеличенной по нормативу на 20% (с учётом затрат по проценту банка).

|  |  |
| --- | --- |
|  | (6.20) |

где Кин – количество инвестиций, р;

С – планируемая себестоимость, р.



Рентабельность инвестиций рассчитывается по формуле:

|  |  |
| --- | --- |
|  | (6.21) |

где Р – рентабельность инвестиций, %;

Пч – чистая прибыль в месяц, р;

И – объём инвестиций, р.



Срок окупаемости инвестиций рассчитывается по формуле:

|  |  |
| --- | --- |
|  | (6.22) |

где Сокуп – срок окупаемости инвестиций,

И – объем инвестиций,

Пч – чистая прибыль,



Таким образом, разработчик программного обеспечения может продать заказчику программное обеспечение по цене, равной 11 412,76 руб., что покроет затраты, равные 7 315,87 рубля и обеспечит прибыль за разработку проекта равную 1 799,70 рублей, все инвестиции окупятся в течении 5 месяцев.

Расчёт экономического эффекта от использования нового программного обеспечения для заказчика

Главный экономический эффект от внедрения средств автоматизации заключается в улучшении экономических и хозяйственных показателей работы предприятия, в первую очередь за счет повышения оперативности управления и снижения трудозатрат на реализацию процесса управления, то есть сокращения расходов на управление.

Снижение же трудозатрат возможно за счет автоматизации работы с документами, снижения затрат на поиск информации.

Рассчитаем возможную экономию затрат на заработную плату.

Экономия за счёт затрат на заработную плату за месяц рассчитывается по формуле:

|  |  |
| --- | --- |
|  | (6.23) |

где *Эсн з/п* – сумма экономии за счёт затрат на заработную плату за месяц, р;

*Тчб* – количество часов работы в день в базовом варианте, ч;

*Тчн* – количество часов работы в день в новом варианте, ч;

*Зсм* – среднемесячная заработная плата одного работника, р;

К – количество работников, использующих ПО, чел.

Таблица 5.8 – Исходные данные для расчёта экономии за счет затрат на заработную плату в связи с применением нового программного обеспечения

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование показателей | Обозначения | Единицы измерения | Значение показателя | |
| в базовом варианте | в новом варианте |
|
| Количество сотрудников, использующих ПО | К | человек | 5 | 5 |
| Среднемесячная зарплата сотрудника, использующего ПО | Зсм | р. | 900 | 900 |
| Количество часов работы в день | Тч | ч | 8 | 6,5 |

Экономия за счет снижения затрат на заработную плату в месяц составит:



Таким образом экономический эффект в данном случае за счет снижения затрат на заработную плату, которая составляет 843,75 р..

Выводы: в результате технико-экономического обоснования применения программного продукта были получены следующие значения показателей:

для разработчика:

общая трудоемкость ПО – 52 человеко-дней;

общая сумма расходов – 7 315,87 р.;

отпускная цена с НДС – 11 412,76р.;

чистая прибыль – 1 799,70 р.

для пользователей – экономия за счет снижения затрат на заработную плату составит 843,75 р. в месяц.

Рассчитанные показатели заносятся в таблицу 6.9, после чего строятся диаграмма (Рисунок 6.1)

Таблица 6.9 – Итоговые экономические показатели проекта

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование статьи | Обозначение | Единица измерения | Значение |
| Общая трудоемкость ПО | То | Чел/дн | 52 |
| Плановая себестоимость ПО |  | р. | 7 315,87 |
| Цена проекта |  | р. | 11 412,76 |
| Экономия, в результате снижения трудоемкости |  | р. | 843,75 |

A picture containing screenshot, text, diagram

Description automatically generated

Рисунок 6.1 – Диаграмма плановой себестоимости ПО

# 7 Охрана труда

## 7.1 Идентификация и анализ вредных и опасных факторов при работе на компьютере

Идентификация опасных и вредных производственных факторов

производится в соответствии с ГОСТ 12.0.003-74 [7].

Вредные и опасные производственные факторы в отделе управления информационными технологиями связаны с работой на персональных компьютерах (ПК). При работе с ПК на работников могут оказывать неблагоприятное воздействие физические и психофизические опасные и вредные производственные фактор.

К физическим факторам относят:

повышенный уровень электромагнитных излучений;

повышенный уровень статического электричества;

повышенная напряженность электростатического поля;

повышенная или пониженная ионизация воздуха;

повышенная яркость света;

прямая и отраженная блесткость;

повышенное значение напряжения в электрической цепи, замыкание которой может произойти через тело человека;

К психофизиологическим факторам относят:

статические перегрузки костно-мышечного аппарата и динамические локальные перегрузки мышц кистей рук;

перенапряжение зрительного анализатора;

умственное перенапряжение;

эмоциональные перегрузки;

монотонность труда.

Каждый из факторов рассмотрим более подробно.

Повышенный уровень электромагнитных излучений.

В наши дни имеются данные, указывающие на то, что повышенное электромагнитное излучение может оказывать негативное воздействие на здоровье людей. Это излучение может приводить к разнообразным заболеваниям и нежелательным физиологическим эффектам.

Например, согласно информации от Всемирной организации здравоохранения (ВОЗ), продолжительное воздействие электромагнитных полей на человека может способствовать возникновению таких проблем со здоровьем, как рак, бесплодие, неврологические нарушения, снижение иммунитета и другие заболевания.

Кроме того, увеличенное электромагнитное излучение может вызывать головные боли, утомление, сонливость, раздражительность, нарушения сна, снижение работоспособности и другие проблемы, связанные со здоровьем.

Повышенный уровень статического электричества.

Влияние статического электричества на человека может проявляться в разнообразных формах, включая следующие проблемы со здоровьем:

1. Раздражение кожи: Длительное взаимодействие с поверхностями, накопившими статическое электричество, может вызвать раздражение кожи, проявляющееся в виде зуда, покраснения и шелушения.
2. Головные боли: Повышенное количество статического электричества на рабочем месте может приводить к головным болям у сотрудников, особенно при продолжительной работе с компьютерами или другими электронными устройствами.
3. Нарушение функций нервной системы: Частый контакт с заряженными статическим электричеством поверхностями может вызвать нарушения в нервной системе человека. Это может проявляться в виде разных симптомов, таких как головокружение, потеря сознания, нарушение координации движений.
4. Ухудшение общего состояния здоровья: Повышенное статическое электричество на рабочем месте может привести к ухудшению общего состояния здоровья работников, что проявляется в виде усталости, раздражительности, нарушения сна.
5. Нарушение работы сердечно-сосудистой системы: Длительное воздействие статического электричества на человека может вызвать нарушения в сердечно-сосудистой системе, что может привести к разным заболеваниям, таким как аритмия, гипертония и т. д.

Повышенная напряженность электростатического поля.

Повышенная интенсивность электростатического поля на рабочем месте, особенно около компьютеров, может негативно влиять на сотрудника, работающего за компьютером. Интенсивность электростатического поля измеряется в вольтах на метр (В/м) и может возникать из-за трения, контакта или разрядки электронных компонентов.

Ниже перечислены основные негативные последствия повышенной напряженности электростатического поля для работника за компьютером:

1. Неприятные ощущения и раздражение: Усиленное электростатическое поле может вызывать ощущение дискомфорта и раздражения у сотрудника, работающего за компьютером. Это может проявляться в виде жжения, покалывания, раздражения кожи и глаз.
2. Сбои в работе компьютера: Повышенная интенсивность электростатического поля может также вызвать сбои в работе компьютера, что может привести к потере данных или другим техническим проблемам.
3. Электромагнитные помехи: Усиленное электростатическое поле может также создавать электромагнитные помехи на экране монитора, что может ухудшить качество изображения и вызвать утомление глаз.
4. Здоровье глаз: Работа за компьютером в условиях усиленного электростатического поля может также увеличить риск развития проблем со здоровьем глаз, таких как синдром сухого глаза или ухудшение зрения.
5. Нарушение функций нервной системы: Продолжительное воздействие усиленного электростатического поля на сотрудника за компьютером может привести к нарушениям в нервной системе, вызывая усталость, головные боли и даже депрессию.

Повышенная или пониженная ионизация воздуха.

Неравномерная ионизация воздуха может негативно воздействовать на сотрудника, работающего за компьютером. Ионизация воздуха определяется присутствием ионов в атмосфере, которые могут образовываться в результате разнообразных физических процессов, таких как ультрафиолетовое излучение, электростатические взаимодействия и другие.

Перечислены основные негативные последствия для работника за компьютером от неравномерной ионизации воздуха:

1. Утомление и раздражительность: Неравномерная ионизация воздуха может вызывать утомление и раздражительность у сотрудника за компьютером. Это происходит из-за воздействия ионов на нервную систему человека, вызывая возбуждение или, наоборот, снижение активности.
2. Снижение внимания: Неравномерная ионизация воздуха может также уменьшать уровень внимания сотрудника за компьютером, что может привести к ошибкам в выполнении задач и снижению эффективности работы.
3. Здоровье глаз: Неравномерное количество ионов в воздухе также может негативно сказаться на здоровье глаз, вызывая утомляемость, сухость и раздражение.
4. Аллергические реакции: Неравномерная ионизация воздуха может вызывать у работника за компьютером аллергические реакции, такие как кашель, насморк и другие проявления аллергии.

Повышенная яркость света.

Воздействие сильной освещенности на рабочем месте с компьютером может негативно сказываться на здоровье и благополучии сотрудника. Интенсивность освещенности измеряется в люксах (lx) и может оказывать разнообразные эффекты на человеческий организм:

1. Риск развития проблем с глазами: Продолжительная работа с компьютером в условиях яркого освещения может увеличить вероятность возникновения заболеваний глаз, таких как катаракта, дегенерация сетчатки и другие проблемы со зрением.
2. Нарушение сна и циркадного ритма: Сильная освещенность может влиять на сон и биологические ритмы сотрудника, что может приводить к снижению продуктивности и увеличению риска заболеваний.
3. Зрительное перенапряжение: Работа с компьютером при сильной освещенности может вызывать утомление глаз, так как они вынуждены адаптироваться к яркому свету. Это может проявляться в виде напряжения в глазах, головных болях и снижении концентрации.
4. Ухудшение настроения и психологического состояния: Яркое освещение может вызывать раздражение, нарушение настроения и даже депрессию у работника, использующего компьютер.

Прямая и отраженная блесткость.

Прямое и отраженное свечение может негативно влиять на сотрудника, работающего за компьютером. Свечение происходит при отражении света от глянцевых поверхностей, таких как экран монитора, стекло окон, поверхность стола и другие предметы в комнате. Вот основные негативные последствия прямого и отраженного свечения для пользователя компьютера:

1. Раздражение глаз: Отраженное свечение от гладких поверхностей, таких как экран монитора, может вызывать раздражение глаз, поскольку глазам приходится постоянно фокусироваться на отраженных объектах. Это может приводить к дискомфорту, покалыванию, зуду и другим неприятным ощущениям в глазах.
2. Ухудшение зрения: Длительное воздействие прямого и отраженного свечения на глаза может вызвать ухудшение зрения. Это может быть связано с постоянным напряжением глаз для адаптации к яркому свету.
3. Утомление глаз: Прямое свечение от источника света, такого как окно или лампа, может привести к утомлению глаз. Если источник света расположен над глазами, это может переутомить глазные мышцы и вызвать головные боли.
4. Снижение производительности: Прямое и отраженное свечение может также уменьшить эффективность работы сотрудника за компьютером, поскольку они вызывают отвлечение, затрудняют чтение текста на экране и выполнение задач.
5. Стресс и раздражительность: Излишнее свечение на рабочем месте может вызывать стресс и раздражительность у работника, что в свою очередь может сказаться на общем благополучии и уровне сотрудничества с коллегами.

Повышенное значение напряжения в электрической цепи, замыкание которой может произойти через тело человека.

Повышенное напряжение в электрической цепи, замыкание которого может произойти через тело человека, представляет существенную опасность для здоровья и жизни работника за компьютером. В случае такого замыкания электрический ток проходит через тело человека и может вызвать серьезные травмы и даже смерть.

Основные негативные последствия повышенного напряжения в электрической цепи для работника за компьютером включают:

1. Электротравмы: Если электрическая цепь замыкается через тело сотрудника, ток проходит через его ткани, причиняя электротравмы. Это может вызвать ожоги, нарушение сердечного ритма, проблемы с дыханием и даже остановку сердца.
2. Потеря данных и повреждение оборудования: Если замыкание происходит в компьютерной системе, это может привести к потере данных или повреждению компьютерного оборудования, что, в свою очередь, вызовет проблемы с работой компьютера.
3. Затруднение работы: Рабочий за компьютером может испытывать проблемы с концентрацией и выполнением задач из-за повышенного напряжения в электрической цепи, поскольку он может быть отвлечен опасениями о безопасности своего рабочего места.
4. Стресс и тревога: Повышенное напряжение в электрической цепи может вызывать стресс и тревогу у работника за компьютером, что может снизить продуктивность и общее благополучие.
5. Безопасность рабочего места: Важно предпринимать все меры для предотвращения повышенного напряжения в электрической цепи, чтобы обеспечить безопасность рабочего места и снизить риск возникновения таких последствий.

Психофизиологические факторы рассмотрим ниже.

Статические перегрузки костно-мышечного аппарата и динамические локальные перегрузки мышц костей рук.

Статическое и динамическое напряжение костно-мышечной системы может негативно сказываться на сотрудниках, работающих за компьютером на протяжении длительного времени, оставаясь в одной позе.

Статическое напряжение костно-мышечной системы может возникнуть вследствие продолжительного сидения у компьютера в неподходящей позе. Например, если рабочее место не соответствует индивидуальным параметрам работника, это может вызывать усталость и мышечное напряжение в области шеи, плеч и спины. В результате могут появиться боли в этих зонах, а также головные боли.

Динамическое напряжение мышц и костей рук может возникать из-за многократного повторения однотипных движений, таких как работа с компьютерной мышью и клавиатурой. Такие нагрузки могут привести к развитию заболеваний, таких как синдром карпального туннеля, теннисный локоть или боли в мышцах и костях запястья, рук и предплечья.

Перенапряжение зрительного анализатора.

Перенапряжение зрительного анализатора возникает из-за высокой нагрузки на визуальную систему человека при продолжительной работе с компьютером или другими гаджетами. Оно может вызывать разнообразные неприятные симптомы и проблемы со здоровьем на длительной основе, негативно влияя на производительность и благополучие работника.

Симптомы перенапряжения зрительного анализатора могут варьироваться, однако чаще всего они включают усталость глаз, сухость, слезотечение или зуд, затрудненный фокус, головные боли, раздражительность, кольца перед глазами, боли в шее и плечах из-за постоянного фокусирования на экране. Работа в неподходящих условиях освещения, монитор с повышенной яркостью или контрастом, малое удаление от экрана и использование неправильной позы при сидении также могут вызывать перенапряжение зрительного анализатора.

Если перенапряжение зрительного анализатора не устранять, это может привести к ухудшению зрения, формированию заболеваний и снижению производительности. В краткосрочной перспективе сотрудник может столкнуться с временным снижением концентрации и монотонностью работы, что затрудняет выполнение задач. В долгосрочной перспективе перенапряжение зрительного анализатора может стать причиной более серьезных заболеваний, таких как катаракта, старческая дегенерация сетчатки и глаукома. Поэтому важно предпринимать меры по предотвращению перенапряжения зрительного анализатора, такие как контроль условий освещения, регулярные перерывы и упражнения для глаз.

Умственное перенапряжение.

Продолжительная работа за компьютером может вызывать умственное перенапряжение, что негативно сказывается на здоровье работника. Факторами, способствующими умственному перенапряжению, являются длительность рабочего времени, недостаток перерывов и неблагоприятная рабочая среда, включая низкую освещенность или высокий шум.

Умственное перенапряжение может приводить к усталости и снижению эффективности работы, особенно из-за постоянного фокусирования глаз на мониторе и чтении текста. Это может снизить концентрацию и замедлить реакцию. Кроме того, работа за компьютером может вызывать головные боли, особенно если рабочее место не соответствует антропометрическим параметрам работника и надлежащим нормам освещенности и уровня шума.

Долгосрочное умственное перенапряжение может привести к нарушению сна и повышенному стрессу, что может быть опасным для здоровья сотрудника. Бессонница, плохое настроение, раздражительность и снижение работоспособности – все это возможные последствия работнику. Длительный стресс может вызвать серьезные проблемы со здоровьем, такие как гипертония, сердечно-сосудистые заболевания и депрессия.

Эмоциональные перегрузки.

Эмоциональное напряжение может негативно сказываться на сотрудниках, работающих за компьютером, и связано с различными аспектами рабочей деятельности. Компьютерная работа часто связана с высокой ментальной нагрузкой, что может вызывать усталость и стресс. При этом такие факторы, как дедлайны, экстремальные загрузки или ограниченное количество доступных ресурсов, могут усугубить эти эффекты, приводя к эмоциональному перенапряжению.

Работа за компьютером может обусловливать изоляцию и снижение социальной активности, что может в итоге вызывать чувства одиночества, тревоги и депрессии. Кроме того, индивиды могут испытывать чувства некомпетентности, если задачи оказываются невыполнимыми, что еще больше усиливает эмоциональное напряжение.

Длительное сидение за компьютером может быть связано с низкой физической активностью и продолжительным пребыванием в неподвижной позе, что может ухудшить физическое состояние и усилить эмоциональные перегрузки. Непрерывное напряжение может снизить производительность и работоспособность, а также повысить вероятность возникновения заболеваний, связанных с эмоциональным и физическим здоровьем. Чтобы минимизировать эмоциональное напряжение при работе на компьютере, сотрудникам следует делать перерывы, выполнять физические упражнения, полноценно общаться с коллегами, а также практиковать методы релаксации и контроля стресса.

Монотонность труда.

Непрерывное выполнение похожих задач на протяжении длительного времени без разнообразия или стимуляции может вызвать состояние монотонности труда. Такая ситуация может привести к психическому и физическому истощению, негативно влияя на результаты рабочего процесса.

Для сотрудников, проводящих много времени за компьютером, такая однообразность может привести к ухудшению настроения, повышенной усталости, стрессу, уменьшению внимания и снижению производительности. Длительная работа в однотипном режиме, особенно без перерывов, может вызвать напряжение глаз, головные боли, боли в области шеи, спины и верхних конечностей.

## 7.2 Технические, технологические, организационные решения по устранению опасных и вредных факторов, разработка защитных средств.

Для обеспечения безопасности труда при работе в кабинете за компьютером необходимо применять ряд технологических и организационных решений. Вот некоторые из них:

1. Регулирование освещенности: Организация рабочих мест с использованием искусственного и естественного освещения, распределение рабочих мест в зависимости от источников света, регулярный контроль освещенности, установка светильников с регулировкой яркости и направления света в соответствии с ГОСТ 12.2.061-81 [8], СН 2.04.03-2020 [9].
2. Эргономичное оформление рабочего места: Разработка и внедрение стандартов оформления рабочих мест, обучение сотрудников правилам эргономики, контроль за соблюдением эргономических требований, регулирование высоты столов и стульев, использование подставок для ног и подлокотников в соответствии с ГОСТ 12.2.032-78 [10].
3. Регулярные перерывы в работе: Организация режима труда и отдыха сотрудников, контроль за соблюдением рекомендуемых перерывов в работе, внедрение коротких физкультурных пауз для разминки и снятия напряжения в соответствии с ГОСТ 12.2.003-91 [11].
4. Обеспечение комфортных условий труда: Регулирование температуры и влажности воздуха в помещении, проведение регулярных проверок микроклимата, использование кондиционеров и увлажнителей воздуха в соответствии с ГОСТ 12.1.005-88 [12], СанПиН №33 [13].
5. Защита от шума и вибрации: Организация рабочих мест с учетом уровня шума и вибрации, проведение регулярных измерений и контроля, использование звукоизоляционных материалов и антивибрационных подставок для оборудования в соответствии с ГОСТ 12.1.003-83 [14], СанПин №115 [15], ГОСТ 12.1.012-2004 [16].
6. Применение средств индивидуальной защиты: Обучение сотрудников правилам использования средств индивидуальной защиты, контроль за их использованием, регулярная замена и проверка средств индивидуальной защиты в соответствии с ГОСТ 12.4.253-2013 [17].
7. Обучение и инструктаж по безопасности труда: Проведение регулярных инструктажей и обучения работников, контроль за их проведением, организация курсов повышения квалификации по безопасности труда в соответствии с ГОСТ 12.0.004-90 [18].
8. Организация рабочего пространства: Рациональное использование рабочего пространства, обеспечение свободного доступа к рабочим местам и эвакуационным выходам, размещение оборудования и мебели с учетом требований безопасности согласно ГОСТ 12.2.032-78 [19].
9. Противопожарная безопасность: Организация системы пожарной безопасности, проведение регулярных проверок и инструктажей, обеспечение наличия огнетушителей и средств эвакуации в соответствии с ГОСТ 12.1.004-91 [20], Декрет Президента Республики Беларусь от 23.11.2017 № 7 [21], СТБ 11.16.10-2016 [22].
10. Электробезопасность: Обучение сотрудников правилам безопасной работы с электрооборудованием, контроль за соблюдением требований электробезопасности, использование защитных устройств и заземления в соответствии с ГОСТ 12.1.038-82 [23], ТКП 339-2022 [24], СТБ 1791-2008 [25], ГОСТ 12.1.030-81 [26].

## 7.3. Индивидуальное задание.

Разработаем инструкцию по охране труда при работе с компьютером на основе Типовой инструкции по охране труда при использовании в работе офисного оборудования, утвержденной Постановлением Министерства труда и социальной защиты Республики Беларусь от 14 апреля 2021 г. № 25 (приложение А).

## 7.4 Анализ выполнения требований охраны труда в структурном подразделении (на рабочем месте) при работе на ПЭВМ и другой офисной технике

Таблица 1 — Анализ соблюдения требований охраны труда при работе на ПЭВМ и другой офисной технике

|  |  |
| --- | --- |
| Требования охраны труда | Фактическое выполнение требования |
| 1 | 2 |
| *Соблюдение обязанностей нанимателя, использующим ПЭВМ и др. офисную технику* | |
| Допуск к работе:  - лица не моложе 18 лет  - женщины со времени установления беременности и в период кормления ребенка грудью работают за ПЭВМ до 3 часов за рабочий день | Выполняется  -Не выполняется |
| Направление на регулярные медицинские осмотры | Руководство не обращает внимания на этот вопрос. |
| Проведение инструктажей по охране труда и технике безопасности | Инструктажи по охране труда и технике безопасности проводятся при приеме на работу новых сотрудников. Регулярные инструктажи проводятся не реже одного раза в шесть месяцев. |
| Режим работы:  - наличие регламентированных пере­рывов  - разработка комплекса физических уп­ражнений | Выполняется  Комплексы физических упражнений отсутствуют |
| Наличие доплат:  - за работу во вредных условиях труда  - за совмещение работ и профессий  - дополнительные компенсации (бес­платные витаминные препараты, лечебно-профилактическое обеспечение и т. д.) | Наличие вредных условий труда отсутствует.  За совмещение работ и профессий предусмотрена доплата.  Дополнительные компенсации не предусмотрены. |
| Наличие государственной гигиенической регистрации ПЭВМ | В организации имеется государственная гигиеническая регистрация ПЭВМ. Это гарантирует, что использование компьютеров в организации не оказывает вредного воздействия на здоровье сотрудников. |
| Аттестация рабочих мест по условиям труда | Не выполняется |
| *Соблюдение санитарно-гигиенических и эргономических требований к рабочим местам* | |
| Требования к помещениям | |
| Параметры помещений:  - площадь на одно рабочее место не менее 4,5 м2  - высота помещения не менее 3 м | Площадь на одно рабочее место составляет 4 м².  Высота помещения составляет 3,5 м. |
| Интерьер помещений:  - материалы с коэффициентом отраже­ния для стен 0,5-0,6, потолка -0,7-0,8  - материалы с разрешением Государст­венного санитарного надзора  - пол ровный, нескользкий, с антиста­тическими свойствами  - оконные проемы со светозащитными устройствами (жалюзи, занавеси и т. п.) | Материалы для стен имеют коэффициент отражения в диапазоне 0,5-0,6. Коэффициент отражения для потолка не измерялся.  Материалы соответствуют разрешению Государственного санитарного надзора.  Пол нескользкий, но не имеет антистатических свойств.  Оконные проемы имеют жалюзи или занавеси для регулирования света. |
| Освещение:  - естественное освещение через окна, ориентированные на север и северо-восток  -КЕО не менее 1,5 % | Естественное освещение через окна ориентированы на север.  Коэффициент естественного освещения не был измерен. |
| - система искусственного освещения (общее равномерное, комбинированное)  - освещенность рабочей поверхности 300-500 лк | В помещении установлена система искусственного освещения, которая может быть, как общим равномерным, так и комбинированным типом.  Освещенность рабочей поверхности не был измерен, поэтому неизвестно, соответствует ли он требованиям. |
| Параметры микроклимата, физических и химических факторов | | |
| Температура воздуха в помещении 21-23ºС в холодный период | Температура воздуха в помещении поддерживается в диапазоне 21-23ºС в холодный период | |
| Относительная влажность воздуха 40-60 % | Относительная влажность воздуха поддерживается в диапазоне 40-60%. | |
| Рабочее место не граничит с помещениями с повышенными уровнями шума | Выполняется | |
| Уровень шума не выше 60 дБА | Уровень шума не измерялся | |
| Осуществление контроля за уровнями виб­рации и электромагнитных излучений | Не выполняется | |
| Осуществление контроля за уровнем со­держания химических веществ в воздухе помещения | Не выполняется | |
| Организация и оборудование рабочих мест | | |
| Наличие аптечки | Имеется | |
| Наличие огнетушителя | Имеется, порошковый | |
| Ежедневная влажная уборка | Выполняется каждое утро | |
| Помещение систематически проветривается после каждого часа работы с ПЭВМ | Выполняется | |
| Расстояние между боковыми поверхностя­ми мониторов не менее 1,2 м | Выполняется | |
| Высота стола 600-800 мм | Выполняется | |
| Стул подъемно-поворотный, полумягкий | Выполняется | |
| Подставка для ног | Не имеется | |
| Расстояние до экрана монитора – 600-700 мм | Выполняется | |
| Помещение оборудовано защитным заземлением (занулением) | Оборудовано | |

## 7.5 Заключение по разделу

В результате работы были выявлены и проанализированы негативные и опасные факторы, связанные с использованием компьютера. Были разработаны инженерные, технологические и организационные меры для устранения этих факторов. Кроме того, была составлена инструкция по соблюдению правил безопасности труда при работе с компьютерами и проведен анализ соблюдения норм охраны труда при использовании персональных электронно-вычислительных машин и другой офисной техникой.

# Заключение по проекту

В процессе выполнения дипломного проекта с использованием среды разработки Microsoft Visual Studio, языка программирования C#, базы данных MS SQL и React была создана программа для наблюдения и управления производственными процессами и качеством изделий. Программа позволяет добавлять, редактировать, удалять данные, создавать отчеты в формате Excel и отслеживать процессы изготовления продукции. Она также предоставляет возможность сотрудникам быстро обмениваться информацией и сохранять ее.

Помимо хранения информации об изделиях, отчетах и производственных процессах, приложение также позволяет хранить данные о сотрудниках компании. В приложении реализован функционал для отслеживания действий пользователей, таких как добавление или изменение изделий, а также удаление информации. Таким образом, программа обеспечивает полный контроль над процессами производства и качеством продукции, а также облегчает взаимодействие между сотрудниками. В ходе разработки базы данных были выделены основные объекты, их атрибуты и установлены связи между ними. Проектирование базы данных учитывало требования к хранению и обработке информации, а также обеспечивало эффективную работу приложения.

Приложение, созданное на основе фреймворка React, предлагает удобный и интуитивно понятный интерфейс для взаимодействия с информационной системой. В приложении реализованы функции добавления новых записей, редактирования существующих, просмотра данных и удаления лишних. Пользователь может взаимодействовать с базой данных, обновлять ее и получать актуальную информацию о производственных процессах и продуктах.

Тем не менее, стоит отметить, что проект имеет возможности для дополнительного развития и совершенствования. В будущем можно расширить функционал приложения, добавив дополнительные возможности для анализа данных, создания отчетов или интеграции с другими системами. В результате выполнения дипломного проекта были успешно решены все поставленные задачи.

# Список использованных источников

1. Азгальдов Г.Г., Карпова Н.Н. Оценка стоимости интеллектуальной собственности и нематериальных активов. М: МАОиК. 2006.
2. Азгальдов Г.Г., Костин А.В. Интеллектуальная собственность, инновации и квалиметрия//Экономические стратегии. 2008. №2.
3. Ильченко И.А. Интеллектуальная собственность в сфере технического регулирования // Стандарты и качество. 2008. № 6. С. 18-21.
4. Лопатин В.Н. Государство и интеллектуальная собственность: переход к инновационной экономике//Интеллектуальная собственность. М: Юрайт. 2008. Том 1.
5. Новицкий Н.И., Олексюк В.Н. Управление качеством продукции. Уч. пособие. Минск: Новое знание, 2001.
6. Ребрин Ю.И. Управление качеством: Учебное пособие. Таганрог: Изд-во ТРТУ, 2004.
7. ГОСТ 12.0.003–74. "ССБТ. Опасные и вредные производственные факторы. Классификация". – М.: Изд-во стандартов, 1980. – 4 с.
8. ГОСТ 12.2.061-81. "ССБТ. Освещение рабочих мест. Общие требования". - М.: Издательство стандартов, 1982. – 12 с.
9. СН 2.04.03-2020. "Естественное и искусственное освещение". Строительные нормы и правила. РУП "Стройтехнорм". Министерство архитектуры и строительства Республики Беларусь – 13 с.
10. ГОСТ 12.2.032-78. "Система стандартов безопасности труда. Рабочее место при выполнении работ сидя. Общие эргономические требования". - М.: Издательство стандартов, 1978. – 4 с.
11. ГОСТ 12.2.003-91. "ССБТ. Режимы труда и отдыха. Общие требования". - М.: Издательство стандартов, 1992. – 8 с.
12. ГОСТ 12.1.005-88. "ССБТ. Микроклимат производственных помещений. Общие требования". - М.: Издательство стандартов, 1989. – 16 с.
13. СанПиН №33. "Требования к микроклимату рабочих мест в производственных и офисных помещениях". Министерство здравоохранения Республики Беларусь, 2013. – 2 с.
14. ГОСТ 12.1.003-83. "ССБТ. Шум. Общие требования безопасности". - М.: Издательство стандартов, 1984. – 8 с.
15. СанПин №115 "Шум на рабочих местах, в транспортных средствах, в

помещениях жилых, общественных зданий и на территории жилой застройки". Министерство здравоохранения Республики Беларусь, 2011. – 3 с.

1. ГОСТ 12.1.012-2004. "Система стандартов безопасности труда. Вибрационная безопасность. Общие требования". - Москва: Стандартинформ, 2010. – 6 с.
2. ГОСТ 12.4.253-2013. "Система стандартов безопасности труда. Средства индивидуальной защиты глаз. Общие технические требования". Москва: Стандартинформ, 2019. Издание официальное. – 12 c.
3. ГОСТ 12.0.004-90. "ССБТ. Организация обучения безопасности труда". - М.: Издательство стандартов, 1991. – 8 с.
4. ГОСТ 12.2.032-78. "ССБТ. Рабочие места. Общие требования безопасности". - М.: Издательство стандартов, 1979. – 8 с.
5. ГОСТ 12.1.004-91. "ССБТ. Пожарная безопасность. Общие требования". - М.: Издательство стандартов, 1992. – 12 с.
6. Декрет Президента Республики Беларусь от 23.11.2017 № 7. "Общие требования пожарной безопасности к содержанию и эксплуатации капитальных строений (зданий, сооружений), изолированных помещений и иных объектов, принадлежащих субъектам хозяйствования". - Администрация Президента Республики Беларусь, 2017. – 13 c.
7. СТБ 11.16.10-2016. "Система стандартов пожарной безопасности. Системы передачи извещений о пожаре. Общие технические требования. Методы контроля". - Минск: Госстандарт, 2016. – 10 c.
8. ГОСТ 12.1.038-82. "ССБТ. Электробезопасность. Общие требования". - М.: Издательство стандартов, 1983. – 12 с.
9. ТКП 339-2022 (33240). "Электроустановки на напряжение до 750 кВ. Линии электропередачи воздушные и токопроводы, устройства распределительные и трансформаторные подстанции, установки электросиловые и аккумуляторные, электроустановки жилых и общественных зданий. Правила устройства и защитные меры электробезопасности. Учет электроэнергии. Нормы приемо-сдаточных испытаний". - Министерство энергетики Республики Беларусь, 2022. – 10 с.
10. СТБ 1791-2007. "Элементы заземления. Общие технические условия". – Минск: Госстандарт, 2007. – 16 c.
11. ГОСТ 12.1.030-81. "Система стандартов безопасности труда. Электробезопасность. Защитное заземление. Зануление". - М.: Издательство стандартов, 1981. – 2 с.

# Приложение А

(Обязательное)

Инструкция по охране труды при работе с компьютером

ГЛАВА 1

ОБЩИЕ ТРЕБОВАНИЯ ПО ОХРАНЕ ТРУДА

1. Настоящей инструкцией определяются требования по охране труда при использовании в работе виртуального испытательного стенда.

Для целей настоящей инструкции под компьютером понимается система, состоящая из технических средств (персональных электронных вычислительных машин (далее – ПЭВМ), копировально-множительной техники, сканирующих устройств, которые анализируя какой-либо объект (изображение, текст), создают цифровую копию изображения (далее – оборудование)), программного и методического обеспечения.

Иные термины и их определения применяются в значениях, установленных Законом Республики Беларусь «Об охране труда».

2. К выполнению работ с использованием компьютера допускаются работающие, прошедшие в установленном законодательством порядке инструктаж по охране труда (вводный, первичный на рабочем месте и повторный) (далее – работающие).

3. В процессе использования в работе компьютера на работающих возможно воздействие следующих вредных и (или) опасных производственных факторов:

повышенный уровень электромагнитных излучений;

повышенный уровень ионизирующих излучений;

повышенный уровень статического электричества;

повышенная напряженность электростатического поля;

повышенная или пониженная ионизация воздуха;

повышенная яркость света;

прямая и отраженная блесткость;

повышенное значение напряжения в электрической цепи, замыкание которой может произойти через тело человека;

статические перегрузки костно-мышечного аппарата и динамические локальные

перегрузки мышц кистей рук;

перенапряжение зрительного анализатора.

В зависимости от условий труда, в которых применяется компьютер, и характера работы на работающих могут воздействовать другие вредные и (или) опасные производственные факторы.

4. При использовании в работе компьютера по назначению работающие обязаны:

соблюдать требования по охране труда;

поддерживать свое рабочее место (место для выполнения работы (оказания услуги), оборудование в исправном состоянии, порядке и чистоте;

проходить в установленном законодательством порядке инструктаж по охране труда;

заботиться о личной безопасности и личном здоровье, а также о безопасности окружающих в процессе выполнения работ либо во время нахождения на территории организации;

немедленно сообщать работодателю о любой ситуации, угрожающей жизни или здоровью работающих и окружающих, несчастном случае, произошедшем на производстве, оказывать содействие работодателю в принятии мер по оказанию необходимой помощи потерпевшим и доставке их в организацию здравоохранения;

исполнять другие обязанности, предусмотренные законодательством об охране труда.

Работники помимо обязанностей, указанных в части первой настоящего пункта, обязаны:

выполнять нормы и обязательства по охране труда, предусмотренные коллективным договором, соглашением, трудовым договором, правилами внутреннего трудового распорядка, функциональными (должностными) обязанностями;

оказывать содействие и сотрудничать с нанимателем в деле обеспечения здоровых и безопасных условий труда, немедленно извещать своего непосредственного руководителя или иного уполномоченного должностного лица нанимателя о неисправности оборудования, об ухудшении состояния своего здоровья.

5. При использовании в работе компьютера работающим необходимо:

пользоваться исправными выключателями, розетками, штепсельными вилками и другой электроарматурой;

знать и соблюдать требования эксплуатационных документов организаций-изготовителей используемого в работе оборудования;

соблюдать правила личной гигиены;

производить чистку оборудования после отключения его от электрической сети.

6. Работник имеет право отказаться от выполнения порученной работы с использованием компьютера в случае возникновения непосредственной опасности для жизни и здоровья его и окружающих до устранения этой опасности. При отказе от выполнения порученной работы по указанному основанию работник обязан незамедлительно письменно сообщить нанимателю либо уполномоченному должностному лицу нанимателя о мотивах такого отказа, подчиняться правилам внутреннего трудового распорядка, за исключением выполнения вышеуказанной работы.

7. Не допускается появление работающих в состоянии алкогольного, наркотического или токсического опьянения, а также распитие спиртных напитков, употребление наркотических средств, психотропных веществ, их аналогов, токсических веществ в рабочее время и по месту работы.

ГЛАВА 2

ТРЕБОВАНИЯ ПО ОХРАНЕ ТРУДА ПЕРЕД НАЧАЛОМ РАБОТЫ

8. Перед началом работы с использованием компьютера работающему следует:

убедиться в достаточном освещении места, где расположен компьютер, при необходимости включить местное освещение;

проверить:

целостность питающих и соединительных кабелей, разъемов и штепсельных соединений, защитного заземления (зануления);

оснащенность рабочего места (места для выполнения работы (оказания услуги) и убедиться в устойчивости положения оборудования на рабочей поверхности;

отсутствие видимых повреждений оборудования;

протереть при необходимости поверхность экрана видеомонитора сухой мягкой тканевой салфеткой;

отрегулировать подъемно-поворотный стул (кресло) по высоте сиденья и углам наклона спинки, а также расстоянию спинки от переднего края сиденья, с учетом роста.

Регулировка каждого параметра подъемно-поворотного стула (кресла) должна быть

независимой, легко осуществляемой и иметь надежную фиксацию;

отрегулировать положение подставки для ног (в случае ее использования);

включить оборудование в электрическую сеть;

расположить клавиатуру ПЭВМ на поверхности рабочего стола на расстоянии 100–300 мм от края, обращенного к работающему, или на специальной, регулируемой по высоте поверхности, отделенной от основной столешницы;

разместить экран видеомонитора на расстоянии 600–700 мм от глаз, но не ближе 500 мм с учетом размеров алфавитно-цифровых знаков и символов (далее – оптимальное расстояние от экрана видеомонитора до глаз) так, чтобы уровень глаз при вертикально расположенном экране видеомонитора приходился на центр или 2/3 высоты экрана;

убедиться в отсутствии бликов (отражений) на экране видеомонитора, встречного светового потока. Возможные мешающие отражения и отблески на экране видеомонитора и другом оборудовании устраняются путем соответствующего их размещения, расположения светильников местного освещения. Для снижения яркости в поле зрения при естественном освещении необходимо применить регулируемые жалюзи, плотные шторы.

9. Перед началом работы с использованием компьютера не допускается:

включать охлажденное (принесенное с улицы в зимнее время) оборудование;

использовать для подключения оборудования розетки, удлинители, не оснащенные заземляющим контактом (шиной);

устанавливать системный блок в закрытых нишах мебели, непосредственно на полу;

располагать экраны видеомониторов навстречу друг другу при рядном размещении рабочих столов в целях исключения их взаимного отражения;

приступать к работе с ПЭВМ при:

мелькании изображения на экране видеомонитора;

обнаружении неисправности офисного оборудования, кабелей или проводов, разъемов, штепсельных соединений;

отсутствии или неисправности защитного заземления (зануления) оборудования.

ГЛАВА 3

ТРЕБОВАНИЯ ПО ОХРАНЕ ТРУДА ПРИ ВЫПОЛНЕНИИ РАБОТЫ

10. Работающему необходимо выполнять только ту работу, которая ему поручена.

11. При выполнении работы с использованием компьютера работающий должен:

поддерживать в порядке и чистоте оборудование и свое рабочее место (место для выполнения работы (оказания услуги);

использовать компьютер исключительно по назначению;

держать открытыми вентиляционные отверстия оборудования;

соблюдать оптимальное расстояние от экрана видеомонитора до глаз;

поддерживать рациональную рабочую позу и оптимальное размещение на рабочей поверхности используемого оборудования с учетом его количества и конструктивных особенностей, характера выполняемой работы;

вынимать застрявшую бумагу при отключенном оборудовании от электрической сети;

осуществлять проветривание помещения.

12. Работающий вправе периодически прерывать работу за экраном видеомонитора на регламентированные перерывы, в случае их установления, для обеспечения работоспособности и сохранения здоровья, или заменять другой работой с целью сокращения рабочей нагрузки у экрана.

13. Во время регламентированных перерывов, в случае их установления, с целью снижения нервно-эмоционального напряжения, утомления зрительного анализатора, устранения влияния гиподинамии и гипокинезии, предотвращения развития статического утомления работающему необходимо выполнять физические упражнения и упражнения для глаз.

14. При выполнении работы с использованием компьютера работающему не допускается:

работать мокрыми руками и способствовать попаданию влаги на поверхность оборудования;

качаться на стуле;

прикасаться к панелям с разъемами оборудования, разъемам питающих и соединительных кабелей, экрану видеомонитора при включенном питании;

ставить на кабель предметы, натягивать, перекручивать и перегибать его;

касаться кабеля (шнура) горячими предметами;

загромождать установленные для перемещения проходы;

загромождать рабочее место (место для выполнения работы (оказания услуги);

производить переключения и отключение питания во время выполнения активной задачи, а также частые переключения питания;

производить самостоятельно вскрытие и ремонт оборудования;

вытирать пыль на включенном оборудовании.

ГЛАВА 4

ТРЕБОВАНИЯ ПО ОХРАНЕ ТРУДА ПО ОКОНЧАНИИ РАБОТЫ

15. По окончании работы с использованием компьютера работающему следует:

корректно закрыть все активные задачи;

отключить оборудование от электрической сети;

осмотреть и привести в порядок рабочее место (место для выполнения работы (оказания услуги);

при необходимости протереть поверхности периферийных устройств (клавиатуру, манипулятор «мышь», принтер, сканер и другое) и вымыть с мылом руки. Протирание периферийных устройств производить мягкой ветошью с применением специальных чистящих средств.

ГЛАВА 5

ТРЕБОВАНИЯ ПО ОХРАНЕ ТРУДА В АВАРИЙНЫХ СИТУАЦИЯХ

16. При повреждении оборудования, проводов, кабелей, неисправности заземления (зануления), появлении запаха гари, возникновении необычного шума и других неисправностях работающему необходимо:

немедленно отключить оборудование от электрической сети;

сообщить о случившемся непосредственному руководителю или иному уполномоченному должностному лицу работодателя.

17. При несчастном случае на производстве работающему необходимо:

быстро принять меры по предотвращению воздействия на потерпевшего травмирующих факторов;

оказать потерпевшему первую помощь;

вызвать на место происшествия медицинских работников или доставить потерпевшего в организацию здравоохранения;

немедленно сообщить о несчастном случае работодателю.

18. При авариях и несчастных случаях на производстве работающему следует обеспечить до начала расследования сохранность обстановки, если это не представляет опасность для жизни и здоровья работающих.

# Приложение Б

(обязательное)

Код программы

Products.tsx

import React, { useState, useEffect } from 'react';

import {

Container,

Typography,

TableContainer,

Table,

TableHead,

TableRow,

TableCell,

TableBody,

IconButton,

Paper,

Box,

Button,

} from '@mui/material';

import { Add, Edit, Delete } from '@mui/icons-material';

import { TablePagination } from '@mui/material';

import { Product } from '../../common/types';

import { getProducts, addProduct, updateProduct, deleteProduct } from '../../common/apiService';

import AddProductDialog from '../Dialogs/AddProductDialog';

import EditProductDialog from '../Dialogs/EditProductDialog';

import { useNavigate } from 'react-router-dom';

import { TextField } from '@mui/material';

import MainMenu from '../Menu/MainMenu';

import { formatDate } from '../../common/dateUtils';

const Products: React.FC = () => {

const [totalCount, setTotalCount] = useState(0);

const [pageNumber, setPage] = useState(0);

const [pageSize, setPageSize] = useState(10);

const [products, setProducts] = useState<Product[]>([]);

const [addDialogOpen, setAddDialogOpen] = useState(false);

const [editDialogOpen, setEditDialogOpen] = useState(false);

const [selectedProduct, setSelectedProduct] = useState<Product | null>(null);

const [search, setSearch] = useState<string>('');

const navigate = useNavigate();

useEffect(() => {

console.log('Текущая страница:', pageNumber);

}, [pageNumber]);

useEffect(() => {

fetchProducts();

}, [pageNumber]);

useEffect(() => {

fetchProducts();

}, []);

const fetchProducts = async () => {

const data = await getProducts(pageNumber + 1, pageSize);

setProducts(data.items);

setTotalCount(data.totalCount);

};

const handleAddProduct = async (product: Product) => {

await addProduct(product);

fetchProducts();

setAddDialogOpen(false);

};

const handleUpdateProduct = async (product: Product) => {

await updateProduct(product);

fetchProducts();

setEditDialogOpen(false);

};

const handleDeleteProduct = async (id: number) => {

await deleteProduct(id);

fetchProducts();

};

const handleSearchChange = (event: React.ChangeEvent<HTMLInputElement>) => {

setSearch(event.target.value);

};

return (

<Container>

<MainMenu />

<Typography variant="h4" component="h1">

Список изделий

</Typography>

<Box mt={2}>

<Button

variant="contained"

startIcon={<Add />}

onClick={() => setAddDialogOpen(true)}

>

Добавить изделие

</Button>

<TableContainer component={Paper} sx={{ marginTop: 2 }}>

<TablePagination

component="div"

count={totalCount}

page={pageNumber}

onPageChange={(event, newPage) => setPage(newPage)}

rowsPerPage={pageSize}

onRowsPerPageChange={(event) => {

setPageSize(parseInt(event.target.value, 10));

setPage(0);

}}

/>

</TableContainer>

<TextField

label="Поиск"

value={search}

onChange={handleSearchChange}

variant="outlined"

size="small"

InputProps={{

style: {

borderRadius: '5px',

},

}}

style={{ marginBottom: '1rem', width: '100%' }}

/>

</Box>

<TableContainer component={Paper} sx={{ marginTop: 2 }}>

<Table>

<TableHead>

<TableRow>

<TableCell>Название</TableCell>

<TableCell>Описание</TableCell>

<TableCell>Статус качества</TableCell>

<TableCell>Дата добавления</TableCell>

</TableRow>

</TableHead>

<TableBody>

{Array.isArray(products) &&

products

.filter((product) =>

product.name.toLowerCase().includes(search.toLowerCase())

)

.map((product) => (

<TableRow key={product.id}>

<TableCell>{product.name}</TableCell>

<TableCell>{product.description}</TableCell>

<TableCell>{product.qualityStatus}</TableCell>

<TableCell>{formatDate(product.creationDate)}</TableCell>

<TableCell style={{ whiteSpace: 'nowrap' }}>

<IconButton

onClick={() => {

setSelectedProduct(product);

setEditDialogOpen(true);

}}

>

<Edit />

</IconButton>

<IconButton

onClick={() => handleDeleteProduct(product.id)}

>

<Delete />

</IconButton>

<Button

variant="outlined"

onClick={() => navigate(`/processes/${product.id}`)}

>

Процессы

</Button>

</TableCell>

</TableRow>

))}

</TableBody>

</Table>

</TableContainer>

<AddProductDialog

open={addDialogOpen}

onClose={() => setAddDialogOpen(false)}

onSubmit={handleAddProduct}

/>

<EditProductDialog

open={editDialogOpen}

onClose={() => setEditDialogOpen(false)}

onSubmit={handleUpdateProduct}

product={selectedProduct}

/>

</Container>

);

};

export default Products;

Processes.tsx

import { useNavigate } from 'react-router-dom';

import { Button, Container } from '@mui/material';

import React, { useState, useEffect } from 'react';

import { useParams } from 'react-router-dom';

import {

Table,

TableBody,

TableCell,

TableContainer,

TableHead,

TableRow,

Paper,

IconButton,

} from '@mui/material';

import { CloudUpload, Delete } from '@mui/icons-material';

import { AddAPhoto } from '@mui/icons-material';

import FileDialog from '../Dialogs/FileDialog';

import DeleteDialog from '../Dialogs/DeleteDialog';

import MainMenu from '../Menu/MainMenu';

import { Process } from '../../common/types';

import { addProcess, deleteProcess, getProcessPhotos, getProcesses, getProduct, uploadProcessPhoto } from '../../common/apiService';

import ExpandMore from '@mui/icons-material/ExpandMore';

import ExpandLess from '@mui/icons-material/ExpandLess';

import AddProcessDialog from '../Dialogs/AddProcessDialog';

import { formatDate } from '../../common/dateUtils';

const Processes: React.FC = () => {

const [fileDialogOpen, setFileDialogOpen] = useState<boolean>(false);

const openFileDialog = (processId: number) => {

setCurrentProcessId(processId);

setFileDialogOpen(true);

};

const [currentProcessId, setCurrentProcessId] = useState<number | null>(null);

const closeFileDialog = () => {

setFileDialogOpen(false);

};

const [deleteDialogOpen, setDeleteDialogOpen] = useState<boolean>(false);

const [processToDelete, setProcessToDelete] = useState<number | null>(null);

const openDeleteDialog = (processId: number) => {

setProcessToDelete(processId);

setDeleteDialogOpen(true);

};

const closeDeleteDialog = () => {

setDeleteDialogOpen(false);

};

const navigate = useNavigate();

const { id } = useParams<{ id: string }>();

const [processes, setProcesses] = useState<Process[]>([]);

const [totalCount, setTotalCount] = useState(0);

const [pageNumber, setPage] = useState(0);

const [pageSize, setPageSize] = useState(1000);

const [productName, setProductName] = useState<string>();

const [images, setImages] = useState([]);

useEffect(() => {

console.log('Загрузка продукта и процессов');

fetchProcesses();

fetchGetProduct();

}, []);

const fetchProcesses = async () => {

const data = await getProcesses(Number(id), pageNumber + 1, pageSize);

setTotalCount(data.totalCount);

setProcesses(data.items);

};

const fetchGetProduct = async () => {

const data = await getProduct(Number(id));

setProductName(data.name);

}

const [selectedProcess, setSelectedProcess] = useState<number | null>(null);

const handleDeleteProcess = async (processId: number) => {

await deleteProcess(processId);

fetchProcesses();

};

const [processImages, setProcessImages] = useState<{ [key: number]: any[] }>({});

const handleDetailsClick = async (processId: number) => {

if (selectedProcess !== processId) {

await fetchPhotos(processId);

}

setSelectedProcess(selectedProcess === processId ? null : processId);

};

const fetchPhotos = async (processId: number) => {

const fetchedImages = await getProcessPhotos(processId);

setProcessImages((prevImages) => ({

...prevImages,

[processId]: fetchedImages,

}));

};

const [addDialogOpen, setAddDialogOpen] = useState(false);

const handleAddProcess = async (process: Process) => {

await addProcess(process);

fetchProcesses();

setAddDialogOpen(false);

};

return (

<Container>

<MainMenu />

<h1>Процессы для изделия {productName}</h1>

<Button

variant="contained"

color="primary"

style={{ marginRight: '1rem', marginBottom: '1rem' }}

onClick={() => setAddDialogOpen(true)}

>

Добавить процесс

</Button>

<Button

variant="contained"

color="secondary"

style={{ marginBottom: '1rem' }}

onClick={() => navigate('/products')}

>

Назад

</Button>

<TableContainer component={Paper}>

<Table>

<TableHead>

<TableRow>

<TableCell>Название</TableCell>

<TableCell>Описание</TableCell>

<TableCell>Время начала</TableCell>

<TableCell>Время окончания</TableCell>

<TableCell>Действия</TableCell>

</TableRow>

</TableHead>

<TableBody>

{processes.map((process) => (

<TableRow key={process.id}>

<TableCell>

{process.name}

</TableCell>

<TableCell>

{process.description}

<div

onClick={() => handleDetailsClick(process.id)}

style={{

display: 'flex',

alignItems: 'center',

cursor: 'pointer',

marginTop: '0.5rem',

fontSize: '0.875rem',

}}

>

{selectedProcess === process.id ? <ExpandLess /> : <ExpandMore />}

<span style={{ marginLeft: '0.25rem' }}>Подробнее</span>

</div>

{selectedProcess === process.id && (

<div>

<h3>Расширенное описание процесса:</h3>

<p>{process.productionProcessDescription}</p>

<div>

{processImages[process.id]?.map((image, index) => (

<img

key={index}

src={`data:image/png;base64, ${image.imageData}`}

alt={`Процесс фото ${index + 1}`}

style={{ maxWidth: '100%', maxHeight: '300px', margin: '0.5rem' }}

/>

))}

</div>

</div>

)}

</TableCell>

<TableCell>{formatDate(process.startTime)}</TableCell>

<TableCell>{formatDate(process.endTime)}</TableCell>

<TableCell>

<IconButton onClick={() => openDeleteDialog(process.id)}>

<Delete />

</IconButton>

<IconButton onClick={() => openFileDialog(process.id)}>

<CloudUpload />

</IconButton>

<IconButton

onClick={() => {

const photoUploadInput = document.getElementById(

`photoUpload-${process.id}`

) as HTMLInputElement;

photoUploadInput?.click();

}}

>

<AddAPhoto />

</IconButton>

<input

type="file"

id={`photoUpload-${process.id}`}

accept="image/\*"

style={{ display: 'none' }}

onChange={async (e) => {

if (e.target.files && e.target.files.length > 0) {

const file = e.target.files[0];

try {

console.log('Добавлено фото:', file, 'для процесса с ID:', process.id);

await uploadProcessPhoto(process.id, file);

console.log('Фото успешно загружено');

} catch (error) {

console.error('Ошибка при загрузке фото:', error);

}

}

}}

/>

<FileDialog

open={fileDialogOpen}

onClose={closeFileDialog}

processId={currentProcessId}

onAdd={(file, processId) => {

console.log('Добавлен файл:', file, 'для процесса с ID:', processId);

}}

/>

</TableCell>

</TableRow>

))}

</TableBody>

</Table>

</TableContainer>

<AddProcessDialog

open={addDialogOpen}

onClose={() => setAddDialogOpen(false)}

onSubmit={handleAddProcess}

productId={Number(id)}

/>

<DeleteDialog

open={deleteDialogOpen}

onClose={closeDeleteDialog}

onDelete={() => {

if (processToDelete !== null) {

handleDeleteProcess(processToDelete);

}

closeDeleteDialog();

}}

/>

</Container>

);

}

export default Processes;

ApiService.ts

import axios from 'axios';

import { BASE\_URL } from './constants';

import { PaginatedResponse, Process, Product, ProductionProcess, User } from './types';

function getAccessToken() {

return localStorage.getItem('access\_token');

}

const api = axios.create({

baseURL: BASE\_URL,

headers: {

'Content-Type': 'application/json',

},

});

api.interceptors.request.use(

(config) => {

const token = getAccessToken();

if (token) {

config.headers.Authorization = `Bearer ${token}`;

}

return config;

},

(error) => {

return Promise.reject(error);

}

);

// Products

export const getProducts = async (pageNumber: number, pageSize: number): Promise<PaginatedResponse<Product>> => {

try {

const response = await api.get('/api/Products', {

params: {

pageNumber,

pageSize,

},

});

return response.data;

} catch (error) {

console.error('Ошибка при получении продуктов:', error);

return {

items: [],

pageNumber: 0,

totalPages: 0,

totalCount: 0,

hasPreviousPage: false,

hasNextPage: false,

};

}

};

export const getProduct = async (id: number): Promise<Product> => {

const response = await api.get(`/api/Products/${id}`);

return response.data;

};

export const addProduct = async (product: Product) => {

const response = await api.post<Product>('/api/Products', product);

return response.data;

};

export const updateProduct = async (product: Product) => {

const response = await api.put<Product>(`/api/Products/${product.id}`, product);

return response.data;

};

export const deleteProduct = async (id: number) => {

const response = await api.delete(`/api/Products/${id}`);

return response.data;

};

// Users

export const getUsers = async (pageNumber: number, pageSize: number): Promise<PaginatedResponse<User>> => {

try {

const response = await api.get('/api/Users', {

params: {

pageNumber,

pageSize,

},

});

return response.data;

} catch (error) {

console.error('Ошибка при получении пользователей:', error);

return {

items: [],

pageNumber: 0,

totalPages: 0,

totalCount: 0,

hasPreviousPage: false,

hasNextPage: false,

};

}

};

// Production processes

export const getProductionProcesses = async (pageNumber: number, pageSize: number): Promise<PaginatedResponse<ProductionProcess>> => {

try {

const response = await api.get('/api/ProductionProcesses', {

params: {

pageNumber,

pageSize,

},

});

return response.data;

} catch (error) {

console.error('Ошибка при получении процессов производства:', error);

return {

items: [],

pageNumber: 0,

totalPages: 0,

totalCount: 0,

hasPreviousPage: false,

hasNextPage: false,

};

}

};

export const addProductionProcess = async (productionProcess: ProductionProcess) => {

const response = await api.post<Product>('/api/ProductionProcesses', productionProcess);

return response.data;

};

export const updateProductionProcess = async (productionProcess: ProductionProcess) => {

const response = await api.put<Product>(`/api/ProductionProcesses/${productionProcess.id}`, productionProcess);

return response.data;

};

export const deleteProductionProcess = async (id: number) => {

const response = await api.delete(`/api/ProductionProcesses/${id}`);

return response.data;

};

// Processes

export const addProcess = async (productionProcess: Process) => {

const response = await api.post<Product>('/api/Processes', productionProcess);

return response.data;

};

export const getProcesses = async (productId: number, pageNumber: number, pageSize: number): Promise<PaginatedResponse<Process>> => {

try {

const response = await api.get('/api/Processes', {

params: {

productId,

pageNumber,

pageSize,

},

});

return response.data;

} catch (error) {

console.error('Ошибка при получении процессов изделия:', error);

return {

items: [],

pageNumber: 0,

totalPages: 0,

totalCount: 0,

hasPreviousPage: false,

hasNextPage: false,

};

}

};

export const deleteProcess = async (id: number) => {

const response = await api.delete(`/api/Processes/${id}`);

return response.data;

};

export const uploadProcessPhoto = async (id: number, file: File) => {

const formData = new FormData();

formData.append('processPhoto', file);

const response = await api.put(`/api/Processes/${id}/photo`, formData, {

headers: {

'Content-Type': 'multipart/form-data',

},

});

return response.data;

};

export const uploadProcessFile = async (id: number | null, file: File) => {

const formData = new FormData();

formData.append('processFile', file);

const response = await api.put(`/api/Processes/${id}/file`, formData, {

headers: {

'Content-Type': 'multipart/form-data',

},

});

return response.data;

};

export const getProcessPhotos = async (processId: number) => {

try {

const response = await api.get(`/api/processes/${processId}/photos`);

return response.data;

} catch (error) {

console.error('Ошибка при получении фотографий процесса:', error);

return [];

}

};

// report

export const downloadExcel = async (fromDate: Date, endDate: Date): Promise<Blob> => {

try {

const response = await api.get("/api/reports/generate", {

responseType: "blob",

params: {

from: fromDate.toISOString(),

end: endDate.toISOString(),

},

});

return response.data;

} catch (error) {

console.error("Ошибка при загрузке Excel-файла:", error);

throw error;

}

};

LoginForm.tsx

// LoginForm.tsx

import React, { useState, ChangeEvent, FormEvent } from 'react';

import axios from 'axios';

import {

Button,

TextField,

Typography,

Container,

Box,

Alert,

} from '@mui/material';

import { useNavigate } from 'react-router-dom';

import { BASE\_URL } from '../../common/constants';

const LoginForm: React.FC = () => {

const [username, setUsername] = useState<string>('');

const [password, setPassword] = useState<string>('');

const [error, setError] = useState<string | null>(null);

const navigate = useNavigate();

const handleSubmit = async (e: FormEvent) => {

e.preventDefault();

try {

const response = await axios.post(`${BASE\_URL}/connect/token`, new URLSearchParams({

grant\_type: 'password',

username,

password,

}), {

headers: {

'Content-Type': 'application/x-www-form-urlencoded',

},

});

localStorage.setItem('access\_token', response.data.access\_token);

localStorage.setItem('id\_token', response.data.id\_token);

setError(null);

navigate('/products');

} catch (err) {

setError('Ошибка входа');

}

};

const handleInputChange = (

e: ChangeEvent<HTMLInputElement | HTMLTextAreaElement>,

setter: React.Dispatch<React.SetStateAction<string>>

) => {

setter(e.target.value);

};

return (

<Container maxWidth="xs">

<Box

sx={{

marginTop: 8,

display: 'flex',

flexDirection: 'column',

alignItems: 'center',

}}

>

<Typography component="h1" variant="h5">

Вход

</Typography>

{error && (

<Box mt={2}>

<Alert severity="error">{error}</Alert>

</Box>

)}

<Box component="form" onSubmit={handleSubmit} sx={{ mt: 1, width: '100%' }}>

<TextField

variant="outlined"

margin="normal"

required

fullWidth

id="username"

label="Имя пользователя"

name="username"

autoFocus

value={username}

onChange={(e) => handleInputChange(e, setUsername)}

/>

<TextField

variant="outlined"

margin="normal"

required

fullWidth

name="password"

label="Пароль"

type="password"

id="password"

value={password}

onChange={(e) => handleInputChange(e, setPassword)}

/>

<Button

type="submit"

fullWidth

variant="contained"

sx={{ mt: 3, mb: 2 }}

>

Войти

</Button>

</Box>

</Box>

</Container>

);

};

export default LoginForm;

ProductionProcesses.tsx

import React, { useEffect, useState } from 'react';

import MainMenu from '../Menu/MainMenu';

import { Box, Button, Container, IconButton, Paper, Table, TableBody, TableCell, TableContainer, TableHead, TablePagination, TableRow, Typography } from '@mui/material';

import { ProductionProcess } from '../../common/types';

import { addProductionProcess, deleteProductionProcess, getProductionProcesses, updateProductionProcess } from '../../common/apiService';

import { useNavigate } from 'react-router-dom';

import { Add, Delete, Edit } from '@mui/icons-material';

import EditProductionProcessDialog from '../Dialogs/EditProductionProcessDialog';

import AddProductionProcessDialog from '../Dialogs/AddProductionProcessDialog';

const ProductionProcesses: React.FC = () => {

const [totalCount, setTotalCount] = useState(0);

const [pageNumber, setPage] = useState(0);

const [pageSize, setPageSize] = useState(10);

const [productionProcesses, setProductionProcesses] = useState<ProductionProcess[]>([]);

const [addDialogOpen, setAddDialogOpen] = useState(false);

const [editDialogOpen, setEditDialogOpen] = useState(false);

const [selectedProductionProcess, setSelectedProductionProcess] = useState<ProductionProcess | null>(null);

const navigate = useNavigate();

useEffect(() => {

console.log('Текущая страница:', pageNumber);

}, [pageNumber]);

useEffect(() => {

fetchProductionProcesses();

}, [pageNumber]);

useEffect(() => {

fetchProductionProcesses();

}, []);

const fetchProductionProcesses = async () => {

const data = await getProductionProcesses(pageNumber + 1, pageSize);

setProductionProcesses(data.items);

setTotalCount(data.totalCount);

};

const handleAddProductionProcess = async (productionProcess: ProductionProcess) => {

await addProductionProcess(productionProcess);

fetchProductionProcesses();

setAddDialogOpen(false);

};

const handleUpdateProductionProcess = async (productionProcess: ProductionProcess) => {

await updateProductionProcess(productionProcess);

fetchProductionProcesses();

setEditDialogOpen(false);

};

const handleDeleteProductionProcess = async (id: number) => {

await deleteProductionProcess(id);

fetchProductionProcesses();

};

return (

<Container>

<MainMenu />

<Typography variant="h4" component="h1">

Производственные процессы

</Typography>

<Box mt={2}>

<Button

variant="contained"

startIcon={<Add />}

onClick={() => setAddDialogOpen(true)}

>

Добавить производственный процесс

</Button>

<TableContainer component={Paper} sx={{ marginTop: 2 }}>

<TablePagination

component="div"

count={totalCount}

page={pageNumber}

onPageChange={(event, newPage) => setPage(newPage)}

rowsPerPage={pageSize}

onRowsPerPageChange={(event) => {

setPageSize(parseInt(event.target.value, 10));

setPage(0);

}}

/>

</TableContainer>

</Box>

<TableContainer component={Paper} sx={{ marginTop: 2 }}>

<Table>

<TableHead>

<TableRow>

<TableCell>Название</TableCell>

<TableCell>Описание</TableCell>

</TableRow>

</TableHead>

<TableBody>

{productionProcesses.map((productionProcess) => (

<TableRow key={productionProcess.id}>

<TableCell>{productionProcess.name}</TableCell>

<TableCell>{productionProcess.description}</TableCell>

<TableCell style={{ whiteSpace: 'nowrap' }}>

<IconButton

onClick={() => {

setSelectedProductionProcess(productionProcess);

setEditDialogOpen(true);

}}

>

<Edit />

</IconButton>

<IconButton

onClick={() => handleDeleteProductionProcess(productionProcess.id)}

>

<Delete />

</IconButton>

</TableCell>

</TableRow>

))}

</TableBody>

</Table>

</TableContainer>

<AddProductionProcessDialog

open={addDialogOpen}

onClose={() => setAddDialogOpen(false)}

onSubmit={handleAddProductionProcess}

/>

<EditProductionProcessDialog

open={editDialogOpen}

onClose={() => setEditDialogOpen(false)}

onSubmit={handleUpdateProductionProcess}

productionProcess={selectedProductionProcess}

/>

</Container>

);

}

export default ProductionProcesses;

Report.tsx

import { Button, Container, TextField, Typography } from '@mui/material';

import { downloadExcel } from '../../common/apiService';

import { useState } from 'react';

import MainMenu from '../Menu/MainMenu';

const Report: React.FC = () => {

const [fromDate, setFromDate] = useState<Date>(new Date());

const [toDate, setToDate] = useState<Date>(new Date());

return (

<Container>

<MainMenu />

<Typography variant="h4" gutterBottom>

Отчет

</Typography>

<TextField

margin="dense"

label="С"

type="datetime-local"

fullWidth

InputLabelProps={{

shrink: true,

}}

value={fromDate.toISOString().slice(0, 16)}

onChange={(e) => setFromDate(new Date(e.target.value))}

/>

<TextField

margin="dense"

label="По"

type="datetime-local"

fullWidth

InputLabelProps={{

shrink: true,

}}

value={toDate.toISOString().slice(0, 16)}

onChange={(e) => setToDate(new Date(e.target.value))}

/>

<Button

variant="contained"

color="primary"

onClick={async () => {

const excelBlob = await downloadExcel(fromDate, toDate);

const url = URL.createObjectURL(excelBlob);

const link = document.createElement('a');

link.href = url;

link.download = 'report.xlsx';

document.body.appendChild(link);

link.click();

document.body.removeChild(link);

}}

>

Скачать Excel-файл

</Button>

</Container>

);

};

export default Report;

AuthorizationCotroller.cs

using Manufacturing.Infrastructure.Identity;

using Microsoft.AspNetCore;

using Microsoft.AspNetCore.Authentication;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Identity;

using Microsoft.AspNetCore.Mvc;

using Microsoft.IdentityModel.Tokens;

using OpenIddict.Abstractions;

using OpenIddict.Server.AspNetCore;

using System.Collections.Immutable;

using System.Security.Claims;

using static OpenIddict.Abstractions.OpenIddictConstants;

namespace Manufacturing.Api.Controllers

{

[AllowAnonymous]

public class AuthorizationController : ControllerBase

{

private readonly SignInManager<ApplicationUser> \_signInManager;

private readonly UserManager<ApplicationUser> \_userManager;

public AuthorizationController(

SignInManager<ApplicationUser> signInManager,

UserManager<ApplicationUser> userManager)

{

\_signInManager = signInManager;

\_userManager = userManager;

}

[HttpPost("~/connect/token"), IgnoreAntiforgeryToken, Produces("application/json")]

public async Task<IActionResult> Exchange()

{

var request = HttpContext.GetOpenIddictServerRequest();

if (request.IsPasswordGrantType())

{

var user = await \_userManager.FindByNameAsync(request.Username);

if (user == null)

{

var properties = new AuthenticationProperties(new Dictionary<string, string>

{

[OpenIddictServerAspNetCoreConstants.Properties.Error] = Errors.InvalidGrant,

[OpenIddictServerAspNetCoreConstants.Properties.ErrorDescription] =

"The username/password couple is invalid."

});

return Forbid(properties, OpenIddictServerAspNetCoreDefaults.AuthenticationScheme);

}

// Validate the username/password parameters and ensure the account is not locked out.

var result = await \_signInManager.CheckPasswordSignInAsync(user, request.Password, lockoutOnFailure: true);

if (!result.Succeeded)

{

var properties = new AuthenticationProperties(new Dictionary<string, string>

{

[OpenIddictServerAspNetCoreConstants.Properties.Error] = Errors.InvalidGrant,

[OpenIddictServerAspNetCoreConstants.Properties.ErrorDescription] =

"The username/password couple is invalid."

});

return Forbid(properties, OpenIddictServerAspNetCoreDefaults.AuthenticationScheme);

}

// Create the claims-based identity that will be used by OpenIddict to generate tokens.

var identity = new ClaimsIdentity(

authenticationType: TokenValidationParameters.DefaultAuthenticationType,

nameType: Claims.Name,

roleType: Claims.Role);

// Add the claims that will be persisted in the tokens.

identity.SetClaim(Claims.Subject, await \_userManager.GetUserIdAsync(user))

.SetClaim(Claims.Email, await \_userManager.GetEmailAsync(user))

.SetClaim(Claims.Name, await \_userManager.GetUserNameAsync(user))

.SetClaims(Claims.Role, (await \_userManager.GetRolesAsync(user)).ToImmutableArray());

// Set the list of scopes granted to the client application.

identity.SetScopes(new[]

{

Scopes.OpenId,

Scopes.Email,

Scopes.Profile,

Scopes.Roles

}.Intersect(request.GetScopes()));

identity.SetDestinations(GetDestinations);

return SignIn(new ClaimsPrincipal(identity), OpenIddictServerAspNetCoreDefaults.AuthenticationScheme);

}

throw new NotImplementedException("The specified grant type is not implemented.");

}

private static IEnumerable<string> GetDestinations(Claim claim)

{

// Note: by default, claims are NOT automatically included in the access and identity tokens.

// To allow OpenIddict to serialize them, you must attach them a destination, that specifies

// whether they should be included in access tokens, in identity tokens or in both.

switch (claim.Type)

{

case Claims.Name:

yield return Destinations.AccessToken;

if (claim.Subject.HasScope(Scopes.Profile))

yield return Destinations.IdentityToken;

yield break;

case Claims.Email:

yield return Destinations.AccessToken;

if (claim.Subject.HasScope(Scopes.Email))

yield return Destinations.IdentityToken;

yield break;

case Claims.Role:

yield return Destinations.AccessToken;

if (claim.Subject.HasScope(Scopes.Roles))

yield return Destinations.IdentityToken;

yield break;

// Never include the security stamp in the access and identity tokens, as it's a secret value.

case "AspNet.Identity.SecurityStamp": yield break;

default:

yield return Destinations.AccessToken;

yield break;

}

}

}

}

ProccessesController.cs

using Manufacturing.Application.Common.Models;

using Manufacturing.Application.Processes.Commands.CreateProcess;

using Manufacturing.Application.Processes.Commands.DeleteProcess;

using Manufacturing.Application.Processes.Commands.UploadProcessFile;

using Manufacturing.Application.Processes.Commands.UploadProcessPhoto;

using Manufacturing.Application.Processes.Queries.GetProcessesByProductId;

using Manufacturing.Application.Processes.Queries.GetProcessPhoto;

using MediatR;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using OpenIddict.Validation.AspNetCore;

namespace Manufacturing.Api.Controllers

{

[Authorize(AuthenticationSchemes = OpenIddictValidationAspNetCoreDefaults.AuthenticationScheme)]

[Route("api/[controller]")]

[ApiController]

public class ProcessesController : ControllerBase

{

private readonly IMediator \_mediator;

public ProcessesController(IMediator mediator)

{

\_mediator = mediator;

}

[HttpGet]

public async Task<PaginatedList<ProcessVm>> GetProcesses([FromQuery] GetProcessesByProductIdQuery query)

{

return await \_mediator.Send(query);

}

[HttpPost]

public async Task<ActionResult<int>> CreateProcess(CreateProcessCommand command)

{

return await \_mediator.Send(command);

}

[HttpPut("{id}/photo")]

public async Task<IActionResult> UploadProcessPhoto(int id, IFormFile processPhoto)

{

byte[] processPhotoBytes = null;

if (processPhoto != null)

{

using (var memoryStream = new MemoryStream())

{

await processPhoto.CopyToAsync(memoryStream);

processPhotoBytes = memoryStream.ToArray();

}

}

var command = new UploadProcessPhotoCommand()

{

Id = id,

ProcessPhoto = processPhotoBytes,

FileName = processPhoto?.FileName,

};

await \_mediator.Send(command);

return Ok();

}

[HttpPut("{id}/file")]

public async Task<IActionResult> UploadProcessFile(int id, IFormFile processFile)

{

byte[] processFileBytes = null;

if (processFile != null)

{

using (var memoryStream = new MemoryStream())

{

await processFile.CopyToAsync(memoryStream);

processFileBytes = memoryStream.ToArray();

}

}

var command = new UploadProcessFileCommand()

{

Id = id,

ProcessFile = processFileBytes,

FileName = processFile?.FileName,

};

await \_mediator.Send(command);

return Ok();

}

[HttpDelete("{id}")]

public async Task<IActionResult> DeleteProcess(int id)

{

await \_mediator.Send(new DeleteProcessCommand(id));

return NoContent();

}

[HttpGet("{id}/photos")]

public async Task<ActionResult<List<ProcessPhotoVm>>> GetPhotos(int id)

{

return await \_mediator.Send(new GetProcessPhotoQuery(id));

}

}

}

ProductsController.cs

using Manufacturing.Application.Common.Models;

using Manufacturing.Application.Products.Commands.CreateProduct;

using Manufacturing.Application.Products.Commands.DeleteProduct;

using Manufacturing.Application.Products.Commands.UpdateProduct;

using Manufacturing.Application.Products.Queries.GetProductById;

using Manufacturing.Application.Products.Queries.GetProductsWithPagination;

using Manufacturing.Application.Products.Queries.ViewModels;

using MediatR;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using OpenIddict.Validation.AspNetCore;

namespace Manufacturing.Api.Controllers

{

[Authorize(AuthenticationSchemes = OpenIddictValidationAspNetCoreDefaults.AuthenticationScheme)]

[Route("api/[controller]")]

[ApiController]

public class ProductsController : ControllerBase

{

private readonly IMediator \_mediator;

public ProductsController(IMediator mediator)

{

\_mediator = mediator;

}

[HttpGet]

public async Task<PaginatedList<ProductVm>> GetProducts([FromQuery] GetProductsWithPaginationQuery query)

{

return await \_mediator.Send(query);

}

[HttpGet("{id}")]

public async Task<IActionResult> GetProduct(int id)

{

var product = await \_mediator.Send(new GetProductQuery(id));

return Ok(product);

}

[HttpPost]

public async Task<ActionResult<int>> CreateProduct(CreateProductCommand command)

{

return await \_mediator.Send(command);

}

[HttpPut("{id}")]

public async Task<IActionResult> UpdateProduct(int id, UpdateProductCommand command)

{

if (id != command.Id)

{

return BadRequest();

}

await \_mediator.Send(command);

return Ok();

}

[HttpDelete("{id}")]

public async Task<IActionResult> DeleteProduct(int id)

{

await \_mediator.Send(new DeleteProductCommand(id));

return NoContent();

}

}

}

ReportsController.cs

using Manufacturing.Application.Reports.Commands.GenerateReport;

using MediatR;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using OpenIddict.Validation.AspNetCore;

namespace Manufacturing.Api.Controllers

{

[Authorize(AuthenticationSchemes = OpenIddictValidationAspNetCoreDefaults.AuthenticationScheme)]

[Route("api/[controller]")]

[ApiController]

public class ReportsController : ControllerBase

{

private readonly IMediator \_mediator;

public ReportsController(IMediator mediator)

{

\_mediator = mediator;

}

[HttpGet("generate")]

public async Task<IActionResult> Generate(

[FromQuery] DateTime from,

[FromQuery] DateTime end)

{

var generateReportDto = await \_mediator.Send(new GenerateReportCommand

{

From = from,

End = end

});

return File(

generateReportDto.Stream,

"application/vnd.openxmlformats-officedocument.spreadsheetml.sheet",

generateReportDto.ExcelName);

}

}

}

ApplicationDbContext.cs

using Manufacturing.Application.Common.Interfaces;

using Manufacturing.Domain.Entities;

using Manufacturing.Infrastructure.Common;

using Manufacturing.Infrastructure.Identity;

using Manufacturing.Infrastructure.Persistence.Interceptors;

using MediatR;

using Microsoft.AspNetCore.Identity.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore;

using System.Reflection;

namespace Manufacturing.Infrastructure.Persistence

{

public class ApplicationDbContext : IdentityDbContext<ApplicationUser>, IApplicationDbContext

{

private readonly IMediator \_mediator;

private readonly AuditableEntitySaveChangesInterceptor \_auditableEntitySaveChangesInterceptor;

public ApplicationDbContext(

DbContextOptions options,

AuditableEntitySaveChangesInterceptor auditableEntitySaveChangesInterceptor,

IMediator mediator)

: base(options)

{

\_auditableEntitySaveChangesInterceptor = auditableEntitySaveChangesInterceptor;

\_mediator = mediator;

}

public DbSet<Employee> Employees => Set<Employee>();

public DbSet<ProductionProcess> ProductionProcesses => Set<ProductionProcess>();

public DbSet<Product> Products => Set<Product>();

public DbSet<ProductionOrder> ProductionOrders => Set<ProductionOrder>();

public DbSet<ProcessExecution> ProcessExecutions => Set<ProcessExecution>();

public DbSet<QualityControl> QualityControls => Set<QualityControl>();

public DbSet<ProductFile> ProductFiles => Set<ProductFile>();

protected override void OnModelCreating(ModelBuilder builder)

{

builder.ApplyConfigurationsFromAssembly(Assembly.GetExecutingAssembly());

base.OnModelCreating(builder);

builder.Entity<ProcessExecution>()

.HasOne(pe => pe.ProcessFile)

.WithMany()

.HasForeignKey(pe => pe.ProcessFileId)

.OnDelete(DeleteBehavior.Restrict);

builder.Entity<ProcessExecution>()

.HasOne(pe => pe.ProcessPhoto)

.WithMany()

.HasForeignKey(pe => pe.ProcessPhotoId)

.OnDelete(DeleteBehavior.Restrict);

}

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

optionsBuilder.AddInterceptors(\_auditableEntitySaveChangesInterceptor);

}

public override async Task<int> SaveChangesAsync(CancellationToken cancellationToken = default)

{

await \_mediator.DispatchDomainEvents(this);

return await base.SaveChangesAsync(cancellationToken);

}

}

}

CreateProcessCommand.cs

using Manufacturing.Application.Common.Interfaces;

using Manufacturing.Domain.Entities;

using MediatR;

namespace Manufacturing.Application.Processes.Commands.CreateProcess

{

public record CreateProcessCommand : IRequest<int>

{

public DateTime StartTime { get; set; }

public DateTime EndTime { get; set; }

public int ProductionProcessId { get; set; }

public int ProductId { get; set; }

public string Description { get; set; }

}

public class CreateProcessCommandHandler : IRequestHandler<CreateProcessCommand, int>

{

private readonly IGenericRepository<ProcessExecution> \_repository;

private readonly ICurrentUserService \_currentUserService;

public CreateProcessCommandHandler(

IGenericRepository<ProcessExecution> repository,

ICurrentUserService currentUserService)

{

\_repository = repository;

\_currentUserService = currentUserService;

}

public async Task<int> Handle(CreateProcessCommand request, CancellationToken cancellationToken)

{

var entity = new ProcessExecution()

{

StartTime = request.StartTime,

EndTime = request.EndTime,

ProductId = request.ProductId,

ProcessId = request.ProductionProcessId,

Description = request.Description,

};

await \_repository.AddAsync(entity, cancellationToken);

return entity.Id;

}

}

}

CreateProductCommand.cs

using Manufacturing.Application.Common.Interfaces;

using Manufacturing.Domain.Entities;

using MediatR;

namespace Manufacturing.Application.Products.Commands.CreateProduct

{

public record CreateProductCommand : IRequest<int>

{

public string Name { get; set; }

public string Description { get; set; }

public string QualityStatus { get; set; }

}

public class CreateProductCommandHandler : IRequestHandler<CreateProductCommand, int>

{

private readonly IGenericRepository<Product> \_repository;

public CreateProductCommandHandler(IGenericRepository<Product> repository)

{

\_repository = repository;

}

public async Task<int> Handle(CreateProductCommand request, CancellationToken cancellationToken)

{

var entity = new Product

{

Name = request.Name,

Description = request.Description,

QualityStatus = request.QualityStatus,

CreationDate = DateTime.Now

};

await \_repository.AddAsync(entity, cancellationToken);

return entity.Id;

}

}

}

GenerateReportCommand.cs

using DocumentFormat.OpenXml;

using DocumentFormat.OpenXml.Packaging;

using DocumentFormat.OpenXml.Spreadsheet;

using Manufacturing.Application.Common.Interfaces;

using MediatR;

using Microsoft.EntityFrameworkCore;

namespace Manufacturing.Application.Reports.Commands.GenerateReport

{

public record GenerateReportCommand : IRequest<GenerateReportVm>

{

public DateTime From { get; set; }

public DateTime End { get; set; }

}

public class GenerateReportCommandHandler : IRequestHandler<GenerateReportCommand, GenerateReportVm>

{

private readonly IApplicationDbContext \_context;

public GenerateReportCommandHandler(IApplicationDbContext context)

{

\_context = context;

}

public async Task<GenerateReportVm> Handle(GenerateReportCommand request, CancellationToken cancellationToken)

{

var stream = new MemoryStream();

using (var document = SpreadsheetDocument.Create(stream, SpreadsheetDocumentType.Workbook))

{

var workbookPart = document.AddWorkbookPart();

workbookPart.Workbook = new Workbook();

var worksheetPart = workbookPart.AddNewPart<WorksheetPart>();

worksheetPart.Worksheet = new Worksheet(new SheetData());

var sheets = document.WorkbookPart.Workbook.AppendChild(new Sheets());

var sheet = new Sheet

{

Id = document.WorkbookPart.GetIdOfPart(worksheetPart),

SheetId = 1,

Name = "Sheet1"

};

sheets.Append(sheet);

var sheetData = worksheetPart.Worksheet.GetFirstChild<SheetData>();

Columns columns = new Columns();

uint columnCount = 6;

for (uint i = 1; i <= columnCount; i++)

{

Column column = new Column

{

Min = i,

Max = i,

BestFit = true,

CustomWidth = true,

Width = 30

};

columns.Append(column);

}

worksheetPart.Worksheet.InsertBefore(columns, sheetData);

// Fill sheet

var products = \_context.Products

.Include(x => x.ProcessExecutions)

.ThenInclude(x => x.ProductionProcess)

.Where(x => x.CreationDate > request.From && x.CreationDate < request.End);

uint index = 1;

var row1 = new Row { RowIndex = index++ };

row1.Append(new Cell

{

CellValue = new CellValue("Название"),

DataType = CellValues.String,

StyleIndex = 0,

});

row1.Append(new Cell

{

CellValue = new CellValue("Описание"),

DataType = CellValues.String,

});

row1.Append(new Cell

{

CellValue = new CellValue("Качество"),

DataType = CellValues.String,

});

row1.Append(new Cell

{

CellValue = new CellValue("Дата создания"),

DataType = CellValues.String,

});

row1.Append(new Cell

{

CellValue = new CellValue("Процессы"),

DataType = CellValues.String,

});

sheetData.Append(row1);

foreach (var product in products)

{

var row = new Row {RowIndex = index++};

row.Append(new Cell

{

CellValue = new CellValue(product.Name),

DataType = CellValues.String,

});

row.Append(new Cell

{

CellValue = new CellValue(product.Description),

DataType = CellValues.String,

});

row.Append(new Cell

{

CellValue = new CellValue(product.QualityStatus),

DataType = CellValues.String,

});

row.Append(new Cell

{

CellValue = new CellValue(product.CreationDate.ToString("MM-dd-yyyy")),

DataType = CellValues.String,

});

var processes = product.ProcessExecutions;

sheetData.Append(row);

foreach (var process in processes)

{

var nestedRow = new Row { RowIndex = index++ };

nestedRow.Append(new Cell());

nestedRow.Append(new Cell());

nestedRow.Append(new Cell());

nestedRow.Append(new Cell());

nestedRow.Append(new Cell

{

CellValue = new CellValue(process.ProductionProcess.Name),

DataType = CellValues.String,

});

sheetData.Append(nestedRow);

}

}

workbookPart.Workbook.Save();

}

stream.Position = 0;

string excelName = $"Отчет-({DateTime.Now.ToString("MM-dd-yyyy")}).xlsx";

return new GenerateReportVm

{

ExcelName = excelName,

Stream = stream

};

}

}

}

UploadProcessPhotoCommand.cs

using Manufacturing.Application.Common.Interfaces;

using Manufacturing.Domain.Entities;

using MediatR;

namespace Manufacturing.Application.Processes.Commands.UploadProcessPhoto

{

public record UploadProcessPhotoCommand : IRequest

{

public int Id { get; set; }

public byte[] ProcessPhoto { get; set; }

public string FileName { get; set; }

}

public class UploadProcessPhotoCommandHandler : IRequestHandler<UploadProcessPhotoCommand>

{

private readonly IGenericRepository<ProcessExecution> \_processRepository;

private readonly IGenericRepository<ProductFile> \_fileRepository;

private readonly IApplicationDbContext \_context;

public UploadProcessPhotoCommandHandler(

IGenericRepository<ProcessExecution> processRepository,

IGenericRepository<ProductFile> fileRepository,

IApplicationDbContext context)

{

\_processRepository = processRepository;

\_fileRepository = fileRepository;

\_context = context;

}

public async Task Handle(UploadProcessPhotoCommand request, CancellationToken cancellationToken)

{

var entity = await \_processRepository.GetByIdAsync(request.Id, cancellationToken);

if (request.ProcessPhoto != null)

{

var file = new ProductFile

{

FileName = request.FileName,

FileContent = request.ProcessPhoto,

FileType = FileType.Photo,

ProcessExecutionId = request.Id,

ProductId = entity.ProductId

};

await \_fileRepository.AddAsync(file, cancellationToken);

}

var fileId = \_context.ProductFiles

.First(f => f.FileName == request.FileName && f.ProcessExecutionId == request.Id);

entity.Id = request.Id;

entity.ProcessPhotoId = fileId.Id;

await \_processRepository.UpdateAsync(entity, cancellationToken);

}

}

}

GetProcessesByProductIdQuery.cs

using AutoMapper;

using Manufacturing.Application.Common.Interfaces;

using Manufacturing.Application.Common.Models;

using MediatR;

using Microsoft.EntityFrameworkCore;

namespace Manufacturing.Application.Processes.Queries.GetProcessesByProductId

{

public record GetProcessesByProductIdQuery : IRequest<PaginatedList<ProcessVm>>

{

public int ProductId { get; set; }

public int PageSize { get; set; }

public int PageNumber { get; set; }

}

public class GetProcessesByProductIdQueryHandler : IRequestHandler<GetProcessesByProductIdQuery, PaginatedList<ProcessVm>>

{

private readonly IApplicationDbContext \_context;

private readonly IMapper \_mapper;

public GetProcessesByProductIdQueryHandler(IApplicationDbContext context, IMapper mapper)

{

\_context = context;

\_mapper = mapper;

}

public async Task<PaginatedList<ProcessVm>> Handle(GetProcessesByProductIdQuery request,

CancellationToken cancellationToken)

{

var processes = \_context.ProcessExecutions

.Where(x => x.ProductId == request.ProductId)

.Include(x => x.ProductionProcess)

.Select(x => new ProcessVm()

{

Id = x.Id,

Name = x.ProductionProcess.Name,

ProductionProcessDescription = x.ProductionProcess.Description,

Description = x.Description,

StartTime = x.StartTime,

EndTime = x.EndTime,

ProductionProcessId = x.ProductionProcess.Id,

});

return await PaginatedList<ProcessVm>.CreateAsync(processes, request.PageNumber, request.PageSize);

}

}

}