Содержание

[1 ОAО "Могилевхимволокно" – база практики 2](#_Toc138205766)

[2 Охрана труда и окружающей среды 3](#_Toc138205767)

[2 Технические вопросы 7](#_Toc138205768)

[3 Правовые вопросы 8](#_Toc138205769)

[4. Индивидуальное задание 9](#_Toc138205770)

[Заключение 20](#_Toc138205771)

[Список использованной литературы 21](#_Toc138205772)

[Приложение А 22](#_Toc138205773)

# 1 ОAО "Могилевхимволокно" – база практики

Открытое акционерное общество «Могилевхимволокно» является единственным в Республике Беларусь крупным производителем диметилтерефталата, полиэфирного гранулята ПЭТ, полиэфирных волокон и нитей, синтетических пленок и основным поставщиком сырья для легкой промышленности.

ОАО «Могилевхимволокно» относится к разряду градообразующих предприятий г. Могилева, обладает инфраструктурой обеспечения функционирования крупного производственного комплекса при наличии квалифицированных кадров и системы их подготовки.

Особенностью предприятия является объединение на одной площадке в единый комплекс производств, связанных технологическим циклом, – от получения исходного сырья (диметилтерефталата, полиэтилентерефталата) до выпуска готовой продукции в виде волокон, нитей, нетканых материалов различного ассортимента и назначения.

Выпускаемая продукция соответствует мировым стандартам качества, пользуется широким спросом: более 65% производимой продукции поставляется на внешние рынки. Поставки осуществляются более чем в 30 стран мира.

# 2 Охрана труда и окружающей среды

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 | |

# 2 Технические вопросы

Выбор СУБД. В ходе работы необходимо было выбрать подходящую СУБД для создания базы данных. Для этого были рассмотрены такие варианты, как MySQL, PostgreSQL, Microsoft SQL Server. Было решено использовать MySQL, так как она является бесплатной и достаточно быстрой и надежной для нашей задачи.

Проектирование базы данных. Для создания эффективной базы данных необходимо было разработать ее структуру и связи между таблицами. Для этого был использован подход с использование Entity Framerwork. В результате были созданы таблицы для учета информации об изделиях, поставщиках, складах и перемещении продукции между ними, формирование отчетов.

Написание запросов. Для получения нужной информации из базы данных были написаны различные запросы на языке C#. Они включали в себя запросы на выборку данных, обновление, удаление, вставку данных и другие.

# 3 Правовые вопросы

Защита персональных данных. При работе с базой данных необходимо было учитывать требования по защите персональных данных. Была разработана политика конфиденциальности, которая определяла, какие данные могут быть собраны, как они будут использоваться и кто имеет доступ к ним.

Авторские права. При разработке базы данных и написании кода необходимо было учитывать авторские права на программное обеспечение. Было принято решение использовать открытые и свободные библиотеки и фреймворки для минимизации рисков.

# 4. Индивидуальное задание

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 разреше-ний (AspNetRoles, AspNetUsers, AspNetUserRoles, AspNetUserClaims).

2. Управление сотрудниками:

- Добавление, редактирование и удаление сотрудников (Employees).

- Просмотр списка сотрудников с возможностью фильтрации и сортиров-ки.

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 произ-водства (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 фильтра-ции 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 возмож-ностям приложения.

- Реализовать адаптивный дизайн для корректного отображения на раз-личных 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 из-делий на предприятии, с интуитивно понятным интерфейсом.

Реализация отдельных функций

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 : IRequest-Handler<CreateProductCommand, int>

{

private readonly IGenericRepository<Product> \_repository;

public CreateProductCommandHandler(IGenericRepository<Product> re-pository)

{

\_repository = repository;

}

public async Task<int> Handle(CreateProductCommand request, Cancel-lationToken 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 : AbstractValida-tor<CreateProductCommand>

{

public CreateProductCommandValidator()

{

RuleFor(v => v.Name)

.MaximumLength(200)

.NotEmpty();

RuleFor(v => v.Description)

.MaximumLength(200)

.NotEmpty();

RuleFor(v => v.QualityStatus)

.MaximumLength(200)

.NotEmpty();

}

}

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);

}

}

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 : IRequest-Handler<DeleteProductCommand>

{

private readonly IGenericRepository<Product> \_repository;

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

{

\_repository = repository;

}

public async Task Handle(DeleteProductCommand request, Cancella-tionToken cancellationToken)

{

var entity = await \_repository.GetByIdAsync(request.Id, cancel-lationToken);

await \_repository.DeleteAsync(entity, cancellationToken);

}

}

public async Task DeleteAsync(T entity, CancellationToken cancella-tionToken)

{

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

await \_dbContext.SaveChangesAsync(cancellationToken);

}

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, UpdatePro-ductCommand 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 : IRequest-Handler<UpdateProductCommand>

{

private readonly IGenericRepository<Product> \_repository;

public UpdateProductCommandHandler(IGenericRepository<Product> re-pository)

{

\_repository = repository;

}

public async Task Handle(UpdateProductCommand request, Cancella-tionToken cancellationToken)

{

var entity = await \_repository.GetByIdAsync(request.Id, cancel-lationToken);

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 cancella-tionToken)

{

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

await \_dbContext.SaveChangesAsync(cancellationToken);

}

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

};

}

# Заключение

В процессе прохождения практики, были приобретены необходимые практические умения и навыки работы, путём непосредственного участия в деятельности управления информационных технологий ОАО "Могилевхимволокно". Во время прохождения практики были выполнены все задачи, которые были поставлены. Достигнута цель практики, а именно, я овладел необходимыми компетенциями, систематизировал, обобщил и углубил знания, полученные в ходе теоретического обучения в вузе.

Данная практика является хорошим практическим опытом для дальнейшей самостоятельной деятельности. За время пройденной практики я изучил структуру предприятия, изучил программное обеспечение, используемое в управлении информационных технологий. Изучил новые паттерны проектирования, новые технологии в создании приложений.

# Список использованной литературы

1. Справочный материал MSDN. Microsoft [Электронный ресурс] — 2021. —Режим доступа: <https://msdn.microsoft.com/library>.
2. Сайт о программировании Habrahabr.ru [Электронный ресурс] — 2021 — Режим доступа: [http://habrahabr.ru](http://habrahabr.ru/).
3. Сайт о программировании Stackoverflow.com [Электронный ресурс] — 2021. — Режим доступа: [https://stackoverflow.com](https://stackoverflow.com/).
4. Сайт о программировании Metanit.com [Электронный ресурс] — 2021. — Режим доступа: <https://metanit.com>.
5. Шилдт, Г. C# 4.0. Полное руководство : пер. с англ. / Г. Шилдт. – М. : Вильямс, 2011. – 1056 с.
6. Дейт К., Коулс А. Базы данных: проектирование, реализация, использование. - 3-е изд. - СПб.: Питер, 2017. - 832 с

Приложение А

Полученная база данных

A screenshot of a computer

Description automatically generated with low confidence