МЕЖГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Программное обеспечение информационных технологий»

РАЗРАБОТКА АСОИ С ИСПОЛЬЗОВАНИЕМ ТЕХНОЛОГИЙ C#, ASP.NET, SQL, HTML, CSS, ANGULAR ДЛЯ УЧЁТА ТОВАРНО-МАТЕРИАЛЬНЫХ ЦЕННОСТЕЙ НА ОАО «МАГИЛЁВЛИФТМАШ»

Курсовое проектирование

по дисциплине «Проектирование автоматизированных систем»

КР.1-53 01 02.10028326.17.81-01

Исполнитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Тищенко Кристина Геннадьевна рррррррррррррррррррррррррррррррррррррррррррррррррррррррАСОИ-181

(подпись)

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Крутолевич Сергей Константинович

(подпись)

Дата допуска к защите \_\_\_\_\_\_\_\_\_\_\_\_\_

Дата защиты \_\_\_\_\_\_\_\_\_\_\_\_\_

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_

Могилев 2021

Содержание

|  |  |
| --- | --- |
| Введение…………………………………………………………………………….. | 3 |
| 1 Анализ бизнес процессов…………………………………………………………. | 4 |
| 1.1 Обоснование разработки АСОИ……………………………………………….. | 4 |
| 1.2 Функциональные требования к АСОИ………………………………………… | 5 |
| 1.3 Прочие требования к АСОИ……………………………………………………. | 9 |
| 2 Проектирование структуры базы данных……………………………………….. | 11 |
| 3 Проектирование архитектуры проекта…………………………………………... | 15 |
| 3.1 Разработка диаграмм взаимодействия…………………………………………. | 15 |
| 3.2 Структура классов АСОИ………………………………………………………. | 15 |
| 3.3 Диаграмма состояний …………………………………………………………... | 19 |
| 4 Управление процессом разработки программного обеспечения………………. | 20 |
| 4.1 Определение трудоёмкости разработки……………………………………….. | 20 |
| 4.2 Отчёт о разработке программных компонентов……………………………… | 20 |
| 5 Тестирование разработанного ПО ………………………………………………. | 31 |
| 6 Руководство пользователя……………………………………………………….. | 36 |
| Заключение…………………………………………………………………………. | 39 |
| Список использованных источников……………………………………………… | 40 |

Введение

Целью данной работы является разработка АСОИ, позволяющей оптимизировать процесс учёта товарно-материальных ценностей в рамках как одного цеха, так и предприятия в целом. Пояснительная записка к курсовому проектированию содержит 6 разделов:

* Анализ бизнес процессов – приводится структура формируемых документов в виде таблиц.
* Проектирование структуры базы данных – приводится структура БД в виде таблиц.
* Проектирование архитектуры проекта – приводится структура интерфейсов, запросов и процедур в виде таблиц.
* Управление процессом разработки программного обеспечения – таблицы трудоёмкости и календарный график.
* Тестирование разработанного ПО – раздел включает в себя информацию по тестированию ПО.
* Руководство пользователя – приводится описание процесса формирования документов с формами.
* Заключение – содержит обобщение выполненной работы.

1 Анализ бизнес процессов

* 1. Обоснование начала разработки АСОИ

В качестве объекта автоматизации выбран стандарт ОАО «Могилёвлифтмаш» - порядок учёта товарно-материальных ценностей в механосборочном цеху.

Система создаётся на основе следующих документов:

* Основные положения;
* Матрица распределения ответственности;
* Порядок получения товарно-материальных ценностей;
* Отчётность о расходе товарно-материальных ценностей;
* Лимитно-заборная карта;
* Акт-требование на замену (дополнительный отпуск) материалов;
* Карта складского учёта;

Настоящий стандарт устанавливает требования по обеспечению своевременного и правильного документированного оформления операций по учёту товарно-материальных ценностей, используемых цехами для изготовления продукции и на ремонтные нужды.

Действие стандарта распространяется на цеха, использующие товарно-материальные ценности для изготовления продукции и ремонтные нужды, службы завода, связанные с контролем получения и использования товарно-материальных ценностей в цехах.

Одним из таких цехов является механосборочный цех (МСЦ). В нём осуществляется большая часть технологических операций.

Учёт товарно-материальных ценностей призван обеспечить правильное и своевременное документированное отражение всех операций по поступлению, отпуску материалов, систематический контроль использования материалов в производстве на базе обоснованных норм их расходования, своевременное получение информации о величине экономии или перерасхода материальных ресурсов по сравнению с установленными нормами.

Настоящий стандарт устанавливает порядок оформления документов на получение товарно-материальных ценностей со складов, отчетность цехов за фактический расход, учёт остатков, неиспользованных товарно-материальных ценностей.

На ОАО «Могилёвский завод лифтового машиностроения» работают служащие с высшим и средне специальным образованием.

Систему предполагается использовать на рабочих местах начальника МСЦ, мастеров МСЦ и заместителя генерального директора по производству.

1.2 Функциональные требования к АСОИ

Назначением проекта является проектирование многопользовательской системы. Систему предполагается использовать на рабочих местах начальника механосборочного цеха (далее – МСЦ), мастеров МСЦ, а так же заместителя генерального директора по производству. Система позволит облегчить порядок учёта товарно-материальных ценностей в цеху, организовав единую систем, позволяющую оформлять операции по учёту товарно-материальных ценностей, используемых цехами для изготовления продукции и на ремонтные нужды. Процесс учёта товарно-материальных ценностей будет ускорен, так как это можно будет проводить непосредственно с рабочего места и с меньшим количеством оформляемых бумаг.

Систему предполагается создать для оптимизации работы с документацией, касающейся учёта товарно-материальных ценностей в МСЦ.

Критерий оценки достижений системы – увеличение скорости обработки информации, которая поступает в формате актов и накладных, для создания различного типа отчётов.

Критерием оценки достижения целей системы будет уменьшение вероятности потерять информацию по перемещению, списанию и отпуску материала и деталей в МСЦ.

Система должна удовлетворять следующим требованиям:

1. надёжность;
2. безопасность;
3. требования к защите информации от несанкционированного доступа. Должна быть предусмотрена защита от несанкционированного доступа к данным, ввода данных, их удаления;
4. данные должны хранить в соответствии с имеющимися документами;
5. система должна быть доступна с любого компьютера в границах завода;
6. информация, хранящаяся в системе должна быть защищена от аварийных ситуаций, влияния внешних воздействий (радиоэлектронная защита);
7. квалификация персонала. Персонал должен быть обучен правилам работы с системой, при этом не обязательно наличие специального технического образования.

Функции, выполняемые подсистемами объектов автоматизации:

1. внесение информации о поступление на склад новых товарно-материальных ценностей и дальнейшее изменение информации о них в ходе производственного процесса. При использовании единой базы данных для учёта всех товарно-материальных ценностей, упрощается работа со всей информации, которая будет поступать по материалам и деталям, которые были получены в ходе производства. Так же проще проводить мониторинг потерь при производстве товарно-материальных ценностей;
2. внесение данных в лимитно-заборную карту по определённому материалу (детали) и вывод на печать. Данный отчёт будет содержать в себе дату, количество отпущенного и возвращённого материала, а так же номер склада, с которого производился отпуск товарно-материальных ценностей;
3. составление и вывод на печать формы №М-10п (акт-требование на замену (дополнительный отпуск) материалов). Данный акт-требование будет содержать информацию о виде операции, отправителе, получателе, корреспондирующем счёте, номенклатурном номере, подразделении, инициировавшем замену (дополнительный отпуск) (таблица 1). Таблица 2 будет содержать следующую информацию: материал (наименование, сорт, размер, марка), единица измерения (код, наименование), количество (затребовано, отпущено), цена, сумма, номер паспорта и порядковый номер карточки складского учёта. Так же в документе будет внесена причина замены (дополнительного отпуска), с кем согласована замена, кто отпустил и получил материал;
4. составление и вывод на печать формы №М-12п (карточки складского учёта). Данная карточка складского учёта будет содержать информацию о номере склада, месте хранения (стеллаж, ячейка), марке, сорте, профиле, размере, номенклатурном номере, единице измерения (код, наименование), цена и номер запаса (таблица 1). Таблица 2 включает в себя следующую информацию: дата записи, номер записи, порядковый номер записи, от кого получено или кому отпущено, приход, расход, остаток;
5. составление и вывод на печать формы №М-13п (накладная на внутреннее перемещение материалов). Данная накладная содержит информацию по корреспондирующему счёту (счёт, субсчёт, код аналитического учёта), материалам (наименование, сорт, размер, марка, номенклатурный номер), единице измерения (код, наименование), количеству (отправлено, принято), цене, стоимости, порядковому номеру карточки складского учёта. Так же форма содержит информацию о тех должностных лицах, которые разрешили внутреннее перемещение материалов, приняли и сдали их;
6. составление и вывод на печать формы №М-11п (требование-накладная на отпуск материалов). Таблица 1 будет содержать: код вида операции, номер документа, дата составления, отправитель (структурное подразделение, вид деятельности), получатель (структурное подразделение, вид деятельности), учётная единица выпуска продукции (работ, услуг). Таблица 2 содержит следующую информацию: корреспондирующий счёт (счёт, субсчёт, код аналитического учёта), материалы (наименование, сорт, размер, марка, номенклатурный номер), единица измерения (код, наименование), количество (затребовано, отпущено), цена, сумма, порядковый номер карточки складского учёта. Так же форма будет содержать информацию о должностных лицах, которые затребовали, разрешили отпуск материалов, отпустили, получили материалы, а так же через кого проводится отпуск материалов;
7. составление и вывод на печать отчёта об использовании материалов в производстве за определённый месяц в МСЦ. Отчёт содержит следующую информацию: номенклатурный номер, наименование материала, единица измерения, потребность по лимиту (графа 5), из графы 5 по карте замены, из графы 5 по требованию, фактически получено, фактический расход, остаток на конец месяца, отклонение фактический расход – лимит. Отчёт должен иметь так же 5 подписей на выходе: начальник цеха, начальник производственно-диспетчерское бюро, начальник производственно-диспетчерского отдела, начальник техбюро, экономиста;
8. составление и вывод на печать отчёта об использовании товарно-материальных ценностей за определённый период времени в МСЦ. Таблица содержит следующие столбцы: порядковый номер, номенклатурный номер, наименование товарно-материальных ценностей, единица измерения, цена, количество, сумма, цифр производственных затрат. Отчёт должен иметь 3 подписи: начальник производственно-диспетчерского бюро, старший кладовщик, экономист;
9. составление и вывод на печать акта на списание материалов, использованных на ремонтные нужды за определённую дату. Таблица акта включает в себя: номенклатурный номер, наименование материала, единицу измерения, количество, цену, сумму, причины списания. Так же в акте содержится информация по материалам, которые подлежат к оприходованию от списания материалов отходы. Акт утверждается начальником цеха, а так же подписывается председателем и членами комиссии;
10. создание на базе всей вышеперечисленной документации материального отчёта за месяц по цеху. Материальный отчёт должен включать в себя следующие приложения:
    * 1. Реестр приемки – сдачи документов по приходу;
      2. Реестр приёмки – сдачи документов по расходу;
      3. Отчёт о движении материалов.

В реестрах должны указываться: номер документа, дата составления, номер первичного документа, дата первичного документа, сколько всего принято документов. Реестры подписываются должностными лицами, которые сдают и принимают документы. В отчёте о движении материала по цеху за месяц составляется таблица со следующими столбцами: наименование, номенклатурный номер, средневзвешенная цена на начало месяца, единица измерения, остаток на начало месяца (количество/сумма), приход с начала месяца (количество/сумма), остаток на конец месяца (количество/сумма), средневзвешенная цена на конец месяца. Подсчитывается итог по таблице. Отчёт подписывается старшим кладовщиком цеха и бухгалтером. Материальный отчёт утверждается начальником цеха.

Таблица 1.1 – Поля документа «Товарная накладная»

|  |  |
| --- | --- |
| Поле в документе | Обозначение |
| Id материала | MaterialID |
| Id единицы измерения | UnitID |
| Запрошенное количество | RequestCount |
| Отданное количество | ReleasedCount |
| Id работника | EmployeeID |
| Номер документа | DocumentNumber |

Таблица 1.2 – Поля документа «Отчет об использовании материалов в производстве»

|  |  |
| --- | --- |
| Поле в документе | Обозначение |
| Id материала | MaterialID |
| Id единицы измерения | UnitID |
| Id работника | EmployeeID |
| Остаток материалов | Remains |
| Запрошенное количество | RequestCount |
| Фактический приход | ActualRise |

Таблица 1.3 – Поля документа «Акт на списание материалов, использованных на ремонтные нужды»

|  |  |
| --- | --- |
| Поле в документе | Обозначение |
| Id материала | MaterialID |
| Id единицы измерения | UnitID |
| Id работника | EmployeeID |
| Количество материала | MaterialCount |

Таблица 1.4 – Поля документа «Карточка складского учёта»

|  |  |
| --- | --- |
| Поле в документе | Обозначение |
| Id материала | MaterialID |
| Дата | Data |
| Номер документа | DocumentNumber |
| Откуда | FromWhom |
| Приход | Income |
| Уход | Outcome |

Таблица 1.5 – Поля документа «Требование-накладная на отпуск материалов»

|  |  |
| --- | --- |
| Поле в документе | Обозначение |
| Id материала | MaterialID |
| Id единицы измерения | UnitID |
| Id работника | EmployeeID |
| Дата | Data |
| Номер документа | DocumentNumber |
| Запрошенное количество | RequestCount |
| Отпущенное количество | ReleasedCount |

Таблица 1.6 – Поля документа «Акт-требование на замену (дополнительный отпуск) материалов»

|  |  |
| --- | --- |
| Поле в документе | Обозначение |
| Id материала | MaterialID |
| Id единицы измерения | UnitID |
| Id работника | EmployeeID |
| Запрошенное количество | RequestCount |
| Отпущенное количество | ReleasedCount |

Таблица 1.7 – Поля документа «Отчёт об использовании ТМЦ»

|  |  |
| --- | --- |
| Поле в документе | Обозначение |
| Id материала | MaterialID |
| Id единицы измерения | UnitID |
| Id работника | EmployeeID |
| Id производственных затрат | ManufacturingCostID |
| Количество материала | MaterialCount |

1.3 Прочие требования к АСОИ

Система должна быть адаптивной к изменениям и простой в использовании, для большего ее распространения.

Минимальные системные требования:

1. Процессор: двухъядерный Intel с тактовой частотой 2,2 ГГц или двухъядерный AMD с тактовой частотой 2,5 ГГц
2. Оперативная память: 1,5 ГБ (для XP), 2 ГБ (для Vista/7)
3. Видеокарта: GeForce 8800 (с 512 МБ видеопамяти) или Radeon HD3850 (с 512 МБ видеопамяти)
4. Жесткий диск: 1.2 ГБ свободного места

Рекомендуемые:

1. Процессор: Intel Core i7 9700KF 3.6GHz
2. Оперативная память: 16 ГБ (для Windows 10)
3. Видеокарта: GeForce RTX 3080 (с 8 ГБ видеопамяти)
4. Жесткий диск: 1.2 ГБ свободного места

2 Проектирование структуры базы данных

С использованием всей предоставленной документации была разработана структура базы данных. Все таблицы были приведены к третьей нормальной форме.

Для каждой таблицы, которая есть в базе данных, ниже приведена структура.

Таблица 2.1 – Структура таблицы «Материалы»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| PK | Name | Type | Not Null | Unique | Length |
| + | ID | Bigint | + | + |  |
|  | Name | Varchar | Allows Null | - | 150 |
|  | Number | Bigint | + | - |  |
|  | Price | Money | + | - |  |

Таблица 2.2 – Структура таблицы «Единицы измерения»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Key | Name | Type | Not Null | Unique | Length |
| PK | ID | Bigint | + | + |  |
|  | Name | Varchar | Allows Null | + | 50 |

Таблица 2.3 – Структура таблицы «Работники»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Key | Name | Type | Not Null | Unique | Length |
| PK | ID | Bigint | + | + |  |
|  | Name | Varchar | Allows Null | + | 50 |
|  | Position | Varchar | Allows Null | - | 100 |

Таблица 2.4 – Структура таблицы «Производственные затраты»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Key | Name | Type | Not Null | Unique | Length |
| PK | ID | Bigint | + | + |  |
|  | Name | Varchar | Allows Null | + | 60 |

Таблица 2.5 – Структура таблицы «Товарная накладная»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Key | Name | Type | Not Null | Unique | Length |
| PK | ID | Bigint | + | + |  |
| FK | MaterialID | Bigint | + | - |  |
| FK | UnitID | Bigint | + | - |  |
|  | RequestCount | Bigint | + | - |  |
|  | ReleasedCount | Bigint | + | - |  |
| FK | EmployeeID | Bigint | - | - |  |
|  | DocumentNumber | Bigint | + | - |  |

Таблица 2.6 – Структура таблицы «Отчет об использовании материалов в производстве»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Key | Name | Type | Not Null | Unique | Length |
| PK | ID | Bigint | + | + |  |
| FK | MaterialID | Bigint | + | - |  |
| FK | UnitID | Bigint | + | - |  |
| FK | EmployeeID | Bigint | - | - |  |
|  | Remains | Bigint | + | - |  |
|  | RequestCount | Bigint | + | - |  |
|  | ActualRise | Bigint | + | - |  |

Таблица 2.7 – Структура таблицы «Карточка складского учёта»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Key | Name | Type | Not Null | Unique | Length |
| PK | ID | Bigint | + | + |  |
| FK | MaterialID | Bigint | + | - |  |
|  | Data | Datetime | + | - |  |
|  | DocumentNumber | Bigint | + | - |  |
|  | FromWhom | Varchar | + | - | 50 |
|  | Income | Bigint | + | - |  |
|  | Outcome | Bigint | + | - |  |

Таблица 2.8 – Структура таблицы «Требование-накладная на отпуск материалов»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Key | Name | Type | Not Null | Unique | Length |
| PK | ID | Bigint | + | + |  |
| FK | MaterialID | Bigint | + | - |  |
| FK | UnitID | Bigint | + | - |  |
| FK | EmployeeID | Bigint | - | - |  |
|  | Data | Datetime | + | - |  |
|  | DocumentNumber | Bigint | + | - |  |
|  | RequestCount | Bigint | + | - |  |
|  | ReleasedCount | Bigint | + | - |  |

Таблица 2.9 – Структура таблицы «Акт-требование на замену (дополнительный отпуск) материалов»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Key | Name | Type | Not Null | Unique | Length |
| PK | ID | Bigint | + | + |  |
| FK | MaterialID | Bigint | + | - |  |
| FK | UnitID | Bigint | + | - |  |
| FK | EmployeeID | Bigint | - | - |  |
|  | RequestCount | Bigint | + | - |  |
|  | ReleasedCount | Bigint | + | - |  |

Таблица 2.10 – Структура таблицы «Отчёт об использовании ТМЦ»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Key | Name | Type | Not Null | Unique | Length |
| PK | ID | Bigint | + | + |  |
| FK | MaterialID | Bigint | + | - |  |
| FK | UnitID | Bigint | + | - |  |
| FK | EmployeeID | Bigint | - | - |  |
| FK | ManufacturingCostID | Bigint | + | - |  |
|  | MaterialCount | Bigint | + | - |  |

Для отображения информационной модели рассматриваемого процесса используются следующие сущности:

* «Материалы». Данная сущность хранит в себе всю информацию, которая относится к материалам, используемых в производстве.
* «Работники». Данная сущность содержит в себе всю информацию о работниках предприятия: их ФИО и занимаемые должности.
* «Единицы измерения». Данная сущность содержит информацию об единицах измерения.
* «Производственные затраты». Данная сущность содержит информацию о затратах на производство.
* «Карточка складского учёта». Данная сущность представляет собой карточку складского учёта для ТМЦ. В ней содержатся все необходимые поля, необходимые для оформления данного документа.
* «Акт-требование на замену (дополнительный отпуск) материалов». Данная сущность представляет собой акт-требование, который составляется на предприятии для дополнительного отпуска или замены материала. В ней содержатся все необходимые поля, чтобы составить данный документ.
* «Накладная на внутреннее перемещение материалов». Данная сущность собой представляет собой накладную на внутренне перемещение материалов. В ней содержится вся необходимая информация, которая необходима для составления данной накладной.
* «Акт на списание материалов, использованных на ремонтные нужды». Данная сущность представляет собой акт, составляемый для тех материалов, которые были использованы на ремонтные нужды. В ней содержится вся информация по материалам, которые подлежат списанию за счёт того, что они были использованы на ремонтные нужды.
* «Отчёт об использовании ТМЦ». Данная сущность представляет собой отчёт, составляемый впоследствии после использования ТМЦ. В ней содержится вся необходимая информация, которая потребуется для составления данного отчёта.
* «Отчёт об использовании материалов в производстве». Данная сущность представляет собой отчёт, который составляется по тем материалам, которые были использованы в производстве. В ней содержится вся необходимая информация по тем материалам, которые использовались в производстве.
* «Требование-накладная на отпуск материалов». Данная сущность представляет собой требование-накладную, которая составляется на те материалы, которые необходимо задействовать в производстве. В ней содержится информация, необходимая для составления данного документа.

3 Проектирование архитектуры проекта

3.1 Разработка диаграммы взаимодействия

Для описания процессов, происходящих в клиентской части web-приложения для учёта ТМЦ, была разработана диаграмма взаимодействия. Она описывает все реализованные на данный момент реализованные варианты использования приложения, которые доступны пользователю. Диаграмма взаимодействия представлена в графической части.

В таблице 3.1 перечислены классы, реализованные в ходе разработки приложения.

Таблица 3.1 – Классы АСОИ

|  |  |
| --- | --- |
| Имя класса | Стереотип класса |
| DebtAct | boundary |
| Employee | boundary |
| ManufacturingCost | boundary |
| Material | boundary |
| RequirementAct | boundary |
| RequirementInvoice | boundary |
| Timecard | boundary |
| Unit | boundary |
| UseInProduction | boundary |
| UseOfMatireals | boundary |
| Waybill | boundary |

3.2 Структура классов АСОИ

Диаграмма классов – это диаграмма, которая демонстрирует общую структуру классов, их атрибутов и взаимосвязей между ними.

Атрибуты класса определяют состав и структуру данных, которые хранятся в объектах этого класса. Каждый атрибут имеет имя и тип, определяющий, какие данные он представляет.

Таблица 3.2 – Интерфейс класса DebtActController

|  |  |
| --- | --- |
| Класс DebtActController | |
| Поля | |
| Имя | Тип данных |
| debtActRepository | IDebtActRepository |
| Методы | |
| Имя | Тип данных |
| 1 | 2 |

Продолжение таблицы 3.2

|  |  |
| --- | --- |
| 1 | 2 |
| GetAll | IList<DebtAct> |
| GetById | ActionResult |
| Post | ActionResult |
| Delete | ActionResult |

Таблица 3.3 – Интерфейс класса EmployeeController

|  |  |
| --- | --- |
| Класс EmployeeController | |
| Поля | |
| Имя | Тип данных |
| employeeRepository | IEmployeeRepository |
| Методы | |
| Имя | Тип данных |
| GetAll | IList<Employee> |
| GetById | ActionResult |
| Put | ActionResult |
| Post | ActionResult |
| Delete | ActionResult |

Таблица 3.4 – Интерфейс класса ManufacturingCostController

|  |  |
| --- | --- |
| Класс ManufacturingCostController | |
| Поля | |
| Имя | Тип данных |
| manufacturingCostRepository | IManufacturingCostRepository |
| Методы | |
| Имя | Тип данных |
| GetAll | IList<ManufacturingCost> |
| GetById | ActionResult |
| Post | ActionResult |
| Delete | ActionResult |

Таблица 3.5 – Интерфейс класса MaterialController

|  |  |
| --- | --- |
| Класс MaterialController | |
| Поля | |
| Имя | Тип данных |
| materialRepository | IMterialRepository |
| Методы | |
| Имя | Тип данных |
| 1 | 2 |
| GetAll | IList<Material> |
| GetById | ActionResult |

Продолжение таблицы 3.5

|  |  |
| --- | --- |
| 1 | 2 |
| Post | ActionResult |
| Put | ActionResult |
| Delete | ActionResult |

Таблица 3.6 – Интерфейс класса RequimentActController

|  |  |
| --- | --- |
| Класс RequimentActController | |
| Поля | |
| Имя | Тип данных |
| requimentActRepository | IRequimentActRepository |
| Методы | |
| Имя | Тип данных |
| GetAll | IList<RequimentAct> |
| GetById | ActionResult |
| Post | ActionResult |
| Delete | ActionResult |

Таблица 3.7 – Интерфейс класса RequirementInvoiceController

|  |  |
| --- | --- |
| Класс RequimentInvoiceController | |
| Поля | |
| Имя | Тип данных |
| requimentInvoiceRepository | IRequimentInvoiceRepository |
| Методы | |
| Имя | Тип данных |
| GetAll | IList<RequimentInvoice> |
| GetById | ActionResult |
| Post | ActionResult |
| Delete | ActionResult |

Таблица 3.8 – Интерфейс класса TimecardController

|  |  |
| --- | --- |
| Класс TimecardController | |
| Поля | |
| Имя | Тип данных |
| timecardRepository | ITimecardRepository |
| Методы | |
| Имя | Тип данных |
| 1 | 2 |
| GetAll | IList<Timecard> |
| GetById | ActionResult |
| Post | ActionResult |
| Delete | ActionResult |

Таблица 3.9 – Интерфейс класса UnitController

|  |  |
| --- | --- |
| Класс UnitController | |
| Поля | |
| Имя | Тип данных |
| unitRepository | IUnitRepository |
| Методы | |
| Имя | Тип данных |
| GetAll | IList<Unit> |
| GetById | ActionResult |
| Post | ActionResult |
| Delete | ActionResult |

Таблица 3.10 – Интерфейс класса UseInProductionController

|  |  |
| --- | --- |
| Класс UseInProductionController | |
| Поля | |
| Имя | Тип данных |
| useInProductionRepository | IUseInProductionRepository |
| Методы | |
| Имя | Тип данных |
| GetAll | IList<UseInProduction> |
| GetById | ActionResult |
| Post | ActionResult |
| Delete | ActionResult |

Таблица 3.11 – Интерфейс класса UseOfMaterialsController

|  |  |
| --- | --- |
| Класс UseOfMaterialsController | |
| Поля | |
| Имя | Тип данных |
| useOfMaterialsRepository | IUseOfMaterialsRepository |
| Методы | |
| Имя | Тип данных |
| GetAll | IList<UseOfMaterials> |
| GetById | ActionResult |
| Post | ActionResult |
| Delete | ActionResult |

Таблица 3.12 – Интерфейс класса WaybillController

|  |  |
| --- | --- |
| Класс WaybillController | |
| Поля | |
| Имя | Тип данных |
| waybillRepository | IWaybillRepository |
| Методы | |

Продолжение таблицы 3.12

|  |  |
| --- | --- |
| Имя | Тип данных |
| GetAll | IList<Waybill> |
| GetById | ActionResult |
| Post | ActionResult |
| Delete | ActionResult |

Перечисленные выше классы, которые представляют собой контроллеры, содержат всю основную бизнес логику данного приложения. Они позволяют просматривать, изменять, добавлять и удалять информацию.

3.3 Диаграмма состояний

Диаграмма состояний используется для описания взаимодействия между программой и пользователем. Диаграмма моделирует переходы между диалоговыми формами и список внутренних действий в форме. На переходах между формами отображается имя кнопки, вызвавшей событие перехода.

Основными элементами диаграммы состояний являются «Состояние» и «Переход». Диаграмма состояний похожа на диаграмму деятельности, но деятельность в случае диаграммы состояний заменена состоянием, переходы символизируют действия.

Состояние содержит имя или имя и список внутренних действий. Список внутренних действий содержит перечень действий или деятельностей, которые выполняются во время нахождения объекта в данном состоянии. Данный список фиксированный.

Список основных действий включает в себя:

* Open\_web\_application – действие, которое выполняется при открытии web-страницы
* Change\_employee\_info – действие, которое выполняется при изменении информации о сотруднике
* Change\_material\_info – действие, которое выполняется при изменении информации о материале
* Open\_admin\_page – действие, которое выполняется при переходе на админскую сторону клиентского приложения.

4 Управление процессом разработки программного обеспечения

4.1 Определение трудоёмкости разработки

В таблице 4.1 указаны примерные трудозатраты разработки элементов АСОИ: количество часов, потраченное на создание того или иного элемента АСОИ.

Таблица 4.1 – Трудоёмкость разработки программного обеспечения

|  |  |
| --- | --- |
| Компонент | Трудоёмкость разработки, ч |
| WebAPI.Controllers | 15 |
| WebAPI.ClientApp | 30 |
| WebAPI.Startup | 5 |
| Infrastructure.Data | 15 |
| Domain.Entites | 15 |
| Domain.Abstraction.Repositories | 10 |
| Итого | 90 |

Таблица 4.2 – Календарный план разработки

|  |  |
| --- | --- |
| Компонент | Даты |
| WebAPI.Controllers | 08.09.2021-13.09.2021 |
| WebAPI.ClientApp | 14.09.2021-24.09.2021 |
| WebAPI.Startup | 25.09.2021-27.09.2021 |
| Infrastructure.Data | 01.10.2021-06.10.2021 |
| Domain.Entites | 10.10.2021-17.10.2021 |
| Domain.Abstraction.Repositories | 21.10.2021-30.10.2021 |

4.2 Отчёт о разработке программных компонентов

В разработке использовался обширный стек технологий. В него входят:

* платформа .NET 5.0 и её основной язык C#
* web-фреймворк ASP.NET
* ORM Entity Framework Core
* фреймворк для клиентской части приложения Angular его основной язык TS
* использование web-служб RESTful

В разработке приложения использовались ORM Entity Framework Core и подход Code first, которые позволяю непосредственно в коде C# создавать сущности базы данных, а так же и контекст этой базы данных. Фрагмент кода, в котором происходит создание сущностей базы данных и контекста базы данных, приведён ниже:

public class DataContext : DbContext

{

public DataContext(DbContextOptions<DataContext> options)

: base(options)

{

Database.EnsureCreated();

}

public DbSet<Materials> Materials { get; set; }

public DbSet<Employee> Employees { get; set; }

public DbSet<Unit> Units { get; set; }

public DbSet<ManufacturingCost> ManufacturingCosts { get; set; }

public DbSet<DebtAct> DebtActs { get; set; }

public DbSet<RequiementAct> RequiementActs { get; set; }

public DbSet<Timecards> Timecards { get; set; }

public DbSet<Waybill> Waybills { get; set; }

public DbSet<UseInProduction> UseInProductions { get; set; }

public DbSet<UseOfMatrials> UseOfMatrials { get; set; }

public DbSet<RequirementInvoice> RequirementInvoices { get; set; }

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

modelBuilder.Entity<Materials>(e => e.HasIndex(p => p.Name));

modelBuilder.Entity<Employee>(e => e.HasIndex(p => p.Name));

modelBuilder.Entity<Unit>(e => e.HasIndex(p => p.Name));

modelBuilder.Entity<ManufacturingCost>(e => e.HasIndex(p => p.Name));

}

}

public class DebtAct

{

public long Id { get; set; }

[ForeignKey("MaterialID")]

public Materials Material { get; set; }

[ForeignKey("UnitID")]

public Unit Unit { get; set; }

public long MaterialCount { get; set; }

[ForeignKey("EmployeeID")]

public Employee Employee { get; set; }

}

public class Employee

{

public long Id { get; set; }

[StringLength(50)]

public string Name { get; set; }

[StringLength(150)]

public string Position { get; set; }

}public class ManufacturingCost

{

public long Id { get; set; }

[StringLength(60)]

public string Name { get; set; }

}

public class Materials

{

public long Id { get; set; }

[StringLength(150)]

public string Name { get; set; }

public long Number { get; set; }

[Column(TypeName="money")]

public decimal Price { get; set; }

}

public class RequiementAct

{

public long Id { get; set; }

[ForeignKey("MaterialID")]

public Materials Material { get; set; }

[ForeignKey("UnitID")]

public Unit Unit { get; set; }

public long RequiredCount { get; set; }

public long ReleasedCount { get; set; }

[ForeignKey("EmployeeID")]

public Employee Employee { get; set; }

}

public class RequirementInvoice

{

public long Id { get; set; }

[ForeignKey("MaterialID")]

public Materials Material { get; set; }

[ForeignKey("UnitID")]

public Unit Unit { get; set; }

public long RequiredCount { get; set; }

public long ReleasedCount { get; set; }

[ForeignKey("EmployeeID")]

public Employee Employee { get; set; }

public long DocumentNumber { get; set; }

}

public class Timecards

{

public long Id { get; set; }

[ForeignKey("MaterialID")]

public Materials Material { get; set; }

public DateTime Date { get; set; }

public long DocumentNumber { get; set; }

[StringLength(50)]

public string FromWhom { get; set; }

public int Income { get; set; }

public int Outcome { get; set; }

}

public class Unit

{

public long Id { get; set; }

[StringLength(50)]

public string Name { get; set; }

}

public class UseInProduction

{

public long Id { get; set; }

[ForeignKey("MaterialID")]

public Materials Material { get; set; }

[ForeignKey("UnitID")]

public Unit Unit { get; set; }

public long Remains { get; set; }

public long RequiredCount { get; set; }

public long ActualRise { get; set; }

[ForeignKey("EmployeeID")]

public Employee Employee { get; set; }

}

public class UseOfMatrials

{

public long Id { get; set; }

[ForeignKey("MaterialID")]

public Materials Material { get; set; }

[ForeignKey("UnitID")]

public Unit Unit { get; set; }

public long MaterialCount { get; set; }

[ForeignKey("ManufacturingCostID")]

public ManufacturingCost ManufacturingCost { get; set; }

[ForeignKey("EmployeeID")]

public Employee Employee { get; set; }

}

public class Waybill

{

public long Id { get; set; }

[ForeignKey("MaterialID")]

public Materials Material { get; set; }

[ForeignKey("UnitID")]

public Unit Unit { get; set; }

public long RequiredCount { get; set; }

public long ReleasedCount { get; set; }

[ForeignKey("EmployeeID")]

public Employee Employee { get; set; }

public long DocumentNumber { get; set; }

}

public interface IDebtActRepository

{

public Task<IList<DebtAct>> GetAll();

public Task<DebtAct> GetById(long id);

public Task Add(DebtAct debtAct);

public Task Delete(long id);

}

public interface IEmployeeRepository

{

public Task<IList<Employee>> GetAll();

public Task<Employee> GetById(long id);

public Task Add(Employee employee);

public Task Update(Employee employee);

public Task Delete(long id);

}

public interface IManufacturingCostsRepository

{

public Task<IList<ManufacturingCost>> GetAll();

public Task<ManufacturingCost> GetById(long id);

public Task Add(ManufacturingCost manufacturingCost);

public Task Delete(long id);

}

public interface IMaterialsRepository

{

public Task<IList<Materials>> GetAll();

public Task<Materials> GetById(long id);

public Task Add(Materials materials);

public Task Update(Materials materials);

public Task Delete(long id);

}

public interface IRequimentActRepository

{

public Task<IList<RequiementAct>> GetAll();

public Task<RequiementAct> GetById(long id);

public Task Add(RequiementAct requiementAct);

public Task Delete(long id);

}

public interface IRequirementInvoiceRepository

{

public Task<IList<RequirementInvoice>> GetAll();

public Task<RequirementInvoice> GetById(long id);

public Task Add(RequirementInvoice requirementInvoice);

public Task Delete(long id);

} public interface ITimecardRepository

{

public Task<IList<Timecards>> GetAll();

public Task<Timecards> GetById(long id);

public Task Add(Timecards timecards);

public Task Delete(long id);

}

public interface IUnitRepository

{

public Task<IList<Unit>> GetAll();

public Task<Unit> GetById(long id);

public Task Add(Unit unit);

public Task Delete(long id);

}

public interface IUseInProductionRepository

{

public Task<IList<UseInProduction>> GetAll();

public Task<UseInProduction> GetById(long id);

public Task Add(UseInProduction useInProduction);

public Task Delete(long id);

}

public interface IUseOfMaterialsRepository

{

public Task<IList<UseOfMatrials>> GetAll();

public Task<UseOfMatrials> GetById(long id);

public Task Add(UseOfMatrials useOfMatrials);

public Task Delete(long id);

}

public interface IWaybillRepository

{

public Task<IList<Waybill>> GetAll();

public Task<Waybill> GetById(long id);

public Task Add(Waybill waybill);

public Task Delete(long id);

}

В ходе разработки, тестирования и использования приложения так же использовались web-служба RESTful и Swagger. Исходя из этого, был изменён код класса Startup. Изменённый код приведён ниже:

public class Startup

{

public Startup(IConfiguration configuration)

{

Configuration = configuration;

}

public IConfiguration Configuration { get; }

// This method gets called by the runtime. Use this method to add services to the container.

public void ConfigureServices(IServiceCollection services)

{

services.AddCors();

services.AddControllersWithViews();

services.AddScoped<IDebtActRepository, DebtActRepository>();

services.AddScoped<IEmployeeRepository, EmployeeRepository>();

services.AddScoped<IManufacturingCostsRepository, ManufacturingCostsRepository>();

services.AddScoped<IMaterialsRepository, MaterialsRepository>();

services.AddScoped<IRequimentActRepository, RequimentActRepository>();

services.AddScoped<IRequirementInvoiceRepository, RequirementInvoiceRepository>();

services.AddScoped<ITimecardRepository, TimecardRepository>();

services.AddScoped<IUnitRepository, UnitRepository>();

services.AddScoped<IUseInProductionRepository, UseInProductionRepository>();

services.AddScoped<IUseOfMaterialsRepository, UseOfMaterialsRepository>();

services.AddScoped<IWaybillRepository, WaybillRepository>();

services.AddDbContext<DataContext>(options => options.UseSqlServer(Configuration.GetConnectionString("DefaultConnection")));

services.AddSwaggerDocument(config =>

{

config.PostProcess = document =>

{

document.Info.Version = "v1";

document.Info.Title = "ToDo API";

document.Info.Description = "A simple ASP.NET Core web API";

document.Info.TermsOfService = "None";

document.Info.Contact = new NSwag.OpenApiContact

{

Name = "Shayne Boyer",

Email = string.Empty,

Url = "https://twitter.com/spboyer"

};

document.Info.License = new NSwag.OpenApiLicense

{

Name = "Use under LICX",

Url = "https://example.com/license"

};

};

});

}

// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

else

{

app.UseExceptionHandler("/Error");

// The default HSTS value is 30 days. You may want to change this for production scenarios, see https://aka.ms/aspnetcore-hsts.

app.UseHsts();

}

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseOpenApi();

app.UseSwaggerUi3();

app.UseRouting();

app.UseCors(builder => builder

.AllowAnyOrigin()

.AllowAnyHeader()

.AllowAnyMethod()

);

app.UseHttpsRedirection();

app.UseEndpoints(endpoints =>

{

endpoints.MapControllerRoute(

name: "default",

pattern: "{controller}/{action=Index}/{id?}");

});

}

}

5 Тестирование разработанного ПО

Для нормальной работы системы необходимо 512 Мбайт оперативной памяти, Windows 7 и выше, MS SQLServer 2018 и выше.

Результатом тестирования в данном случае может послужить безошибочное добавление нового сотрудника, материала и т.д.

После тестирования получили результаты:

* Информационная система поддерживает многопользовательский режим.
* Тестирование запросов прошло успешно.

В курсовом проекте разработана база данных с клиентскими приложениями, которая хранится на SQL сервере. Составлены запросы и формы. Формы составлены на основе запросов и таблиц, и используются для занесения и модификации информации в базе данных.

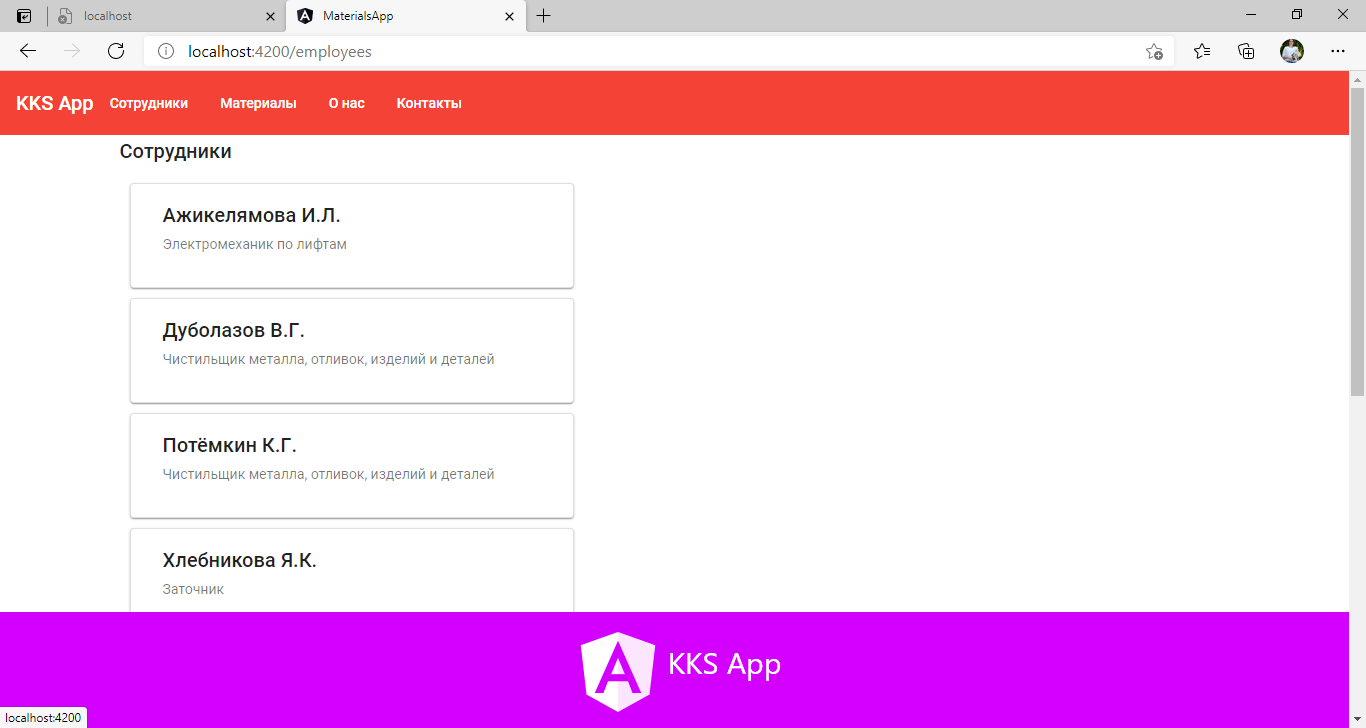


Рисунок 5.1 – Вывод списка работников

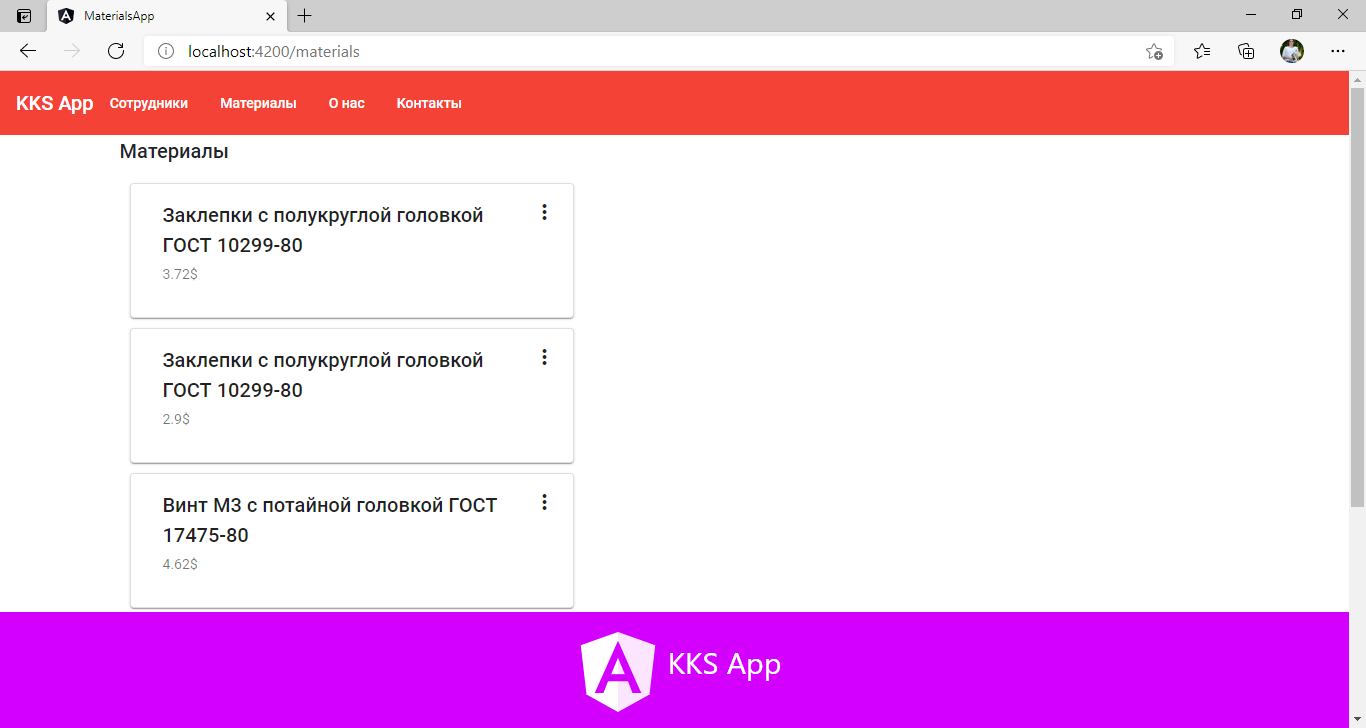


Рисунок 5.2 – Вывод списка материалов

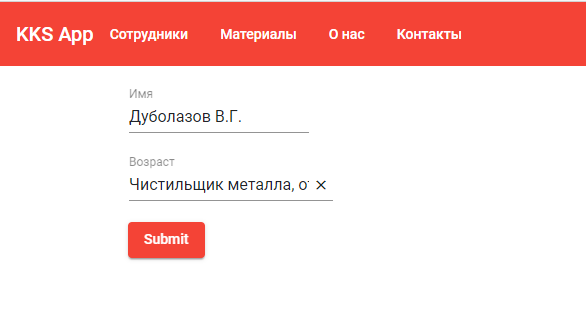


Рисунок 5.3 – Изменение записи работника в базе данных

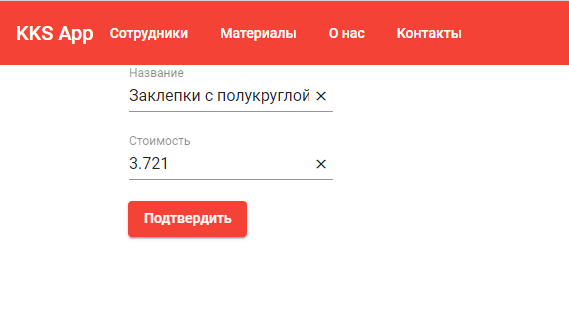


Рисунок 5.4 – Изменение стоимости материала

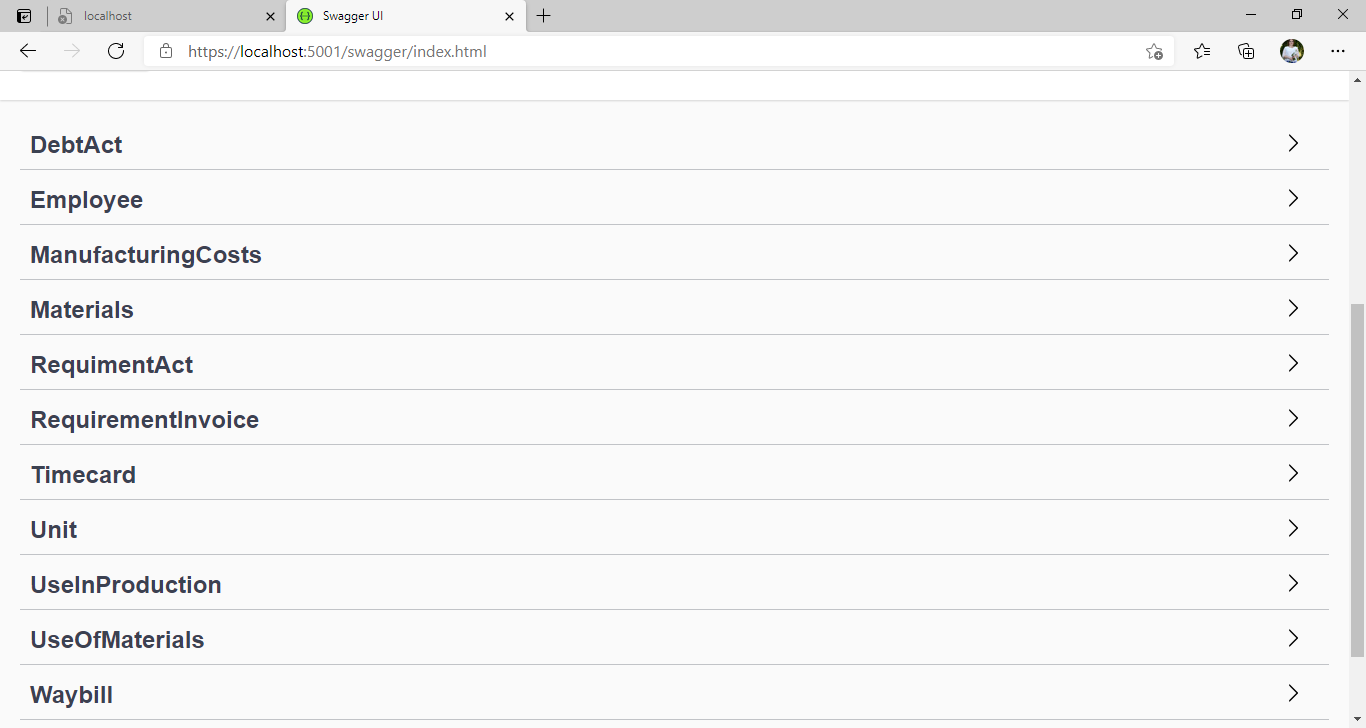


Рисунок 5.5 - Вывод всей информации (админская сторона клиента)

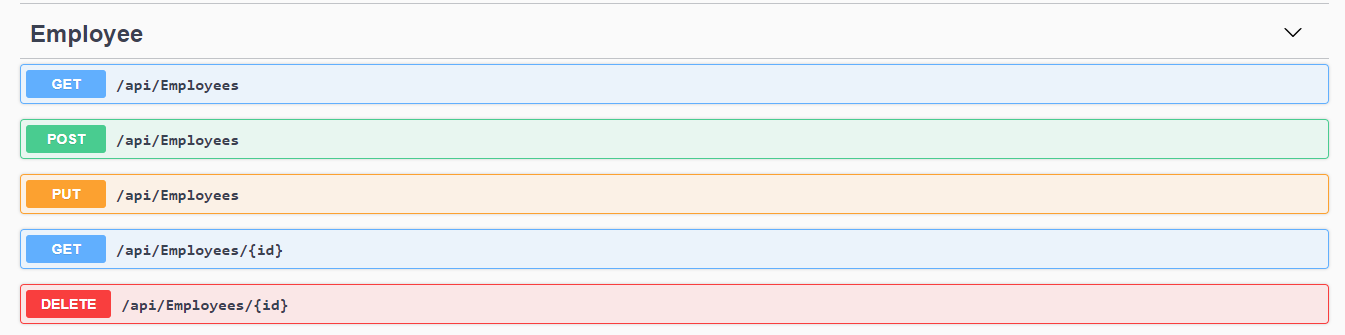


Рисунок 5.6 – CRUD запросы для работы с Сотрудниками

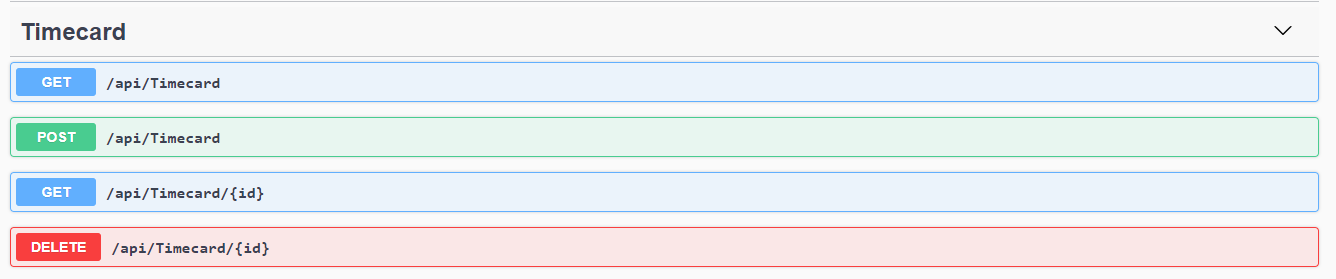


Рисунок 5.7 - CRUD запросы для работы с Карточкой складского учёта

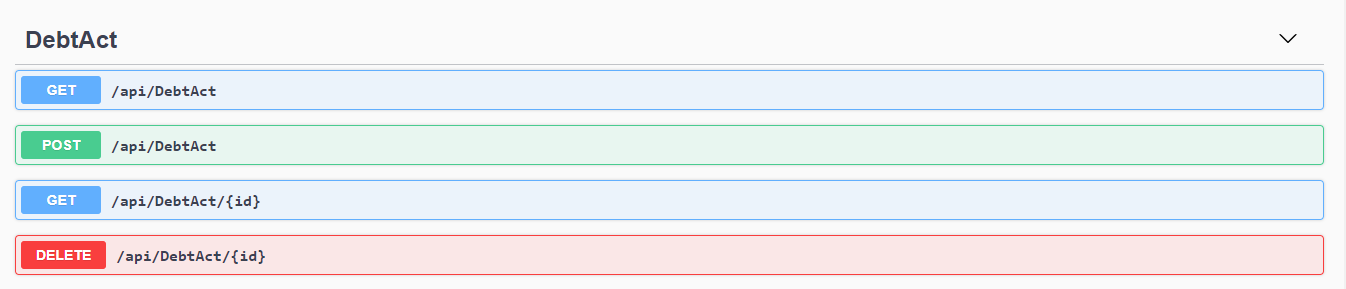


Рисунок 5.7 - CRUD запросы для работы с Актом на списание материалов, использованных на ремонтные нужды

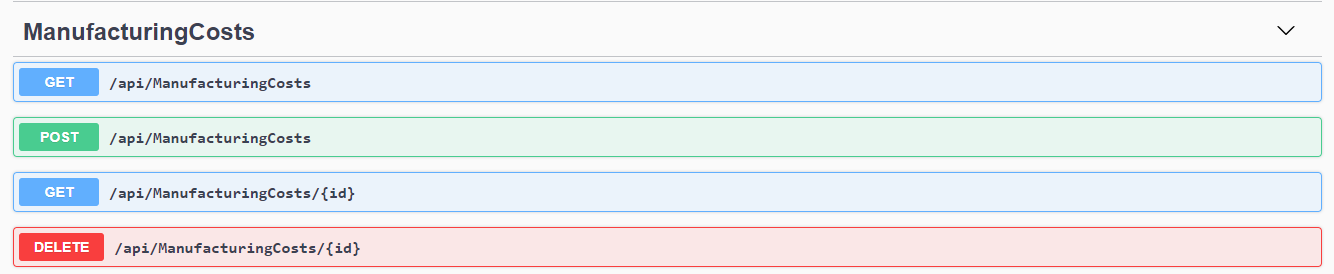


Рисунок 5.8 - CRUD запросы для работы с Производственными затратами

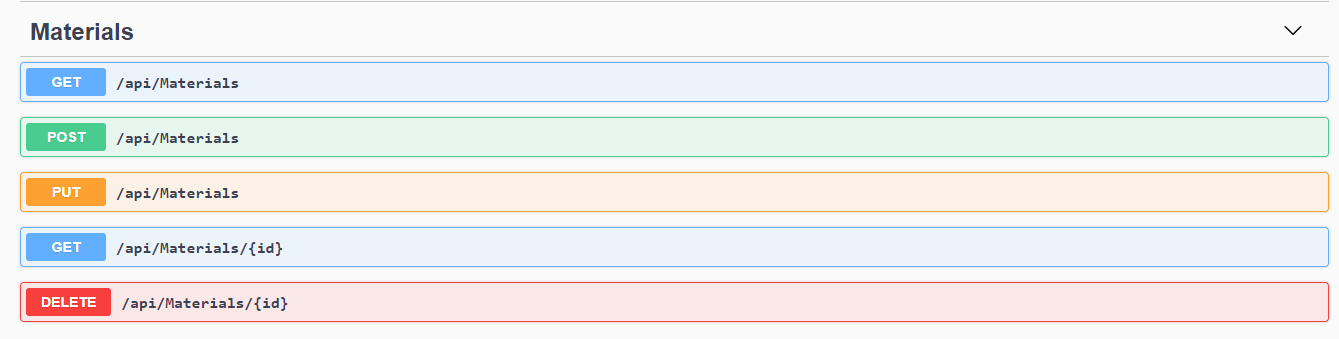


Рисунок 5.9 - CRUD запросы для работы с Материалами

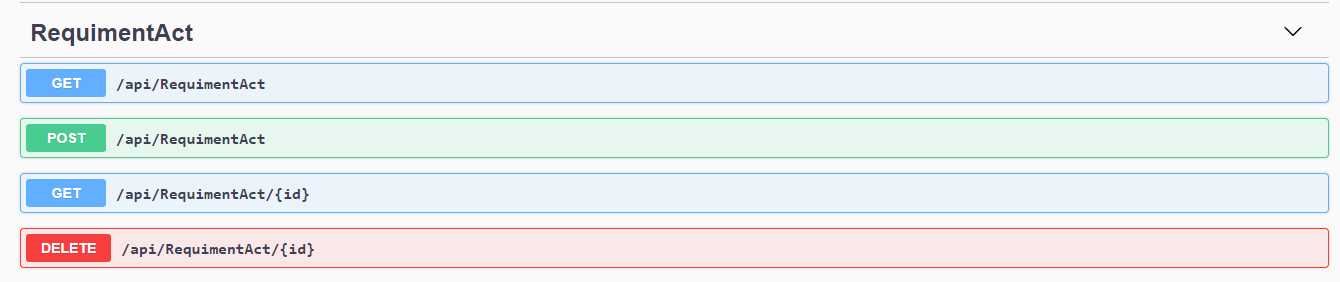


Рисунок 5.10 - CRUD запросы для работы с Актом-требованием на замену (дополнительный отпуск) материалов

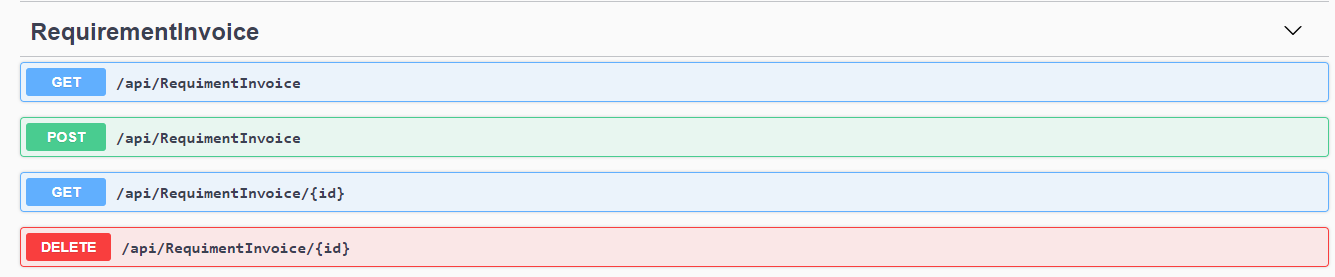


Рисунок 5.11 - CRUD запросы для работы с Требованием-накладной на отпуск материалов

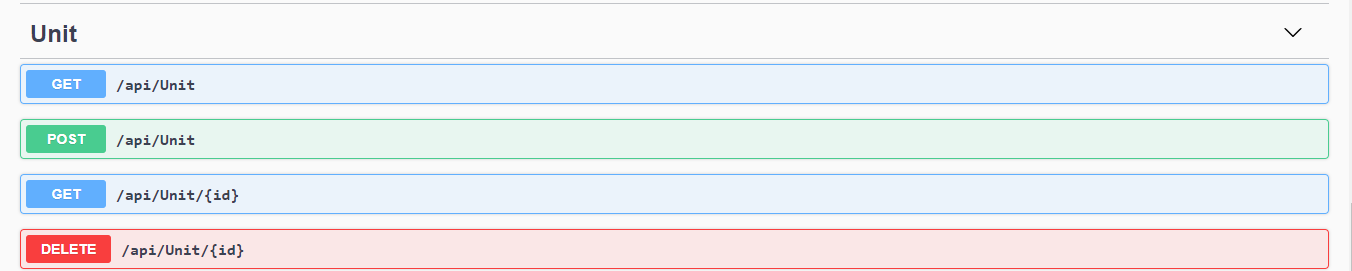


Рисунок 5.12 - CRUD запросы для работы с Единицами измерения

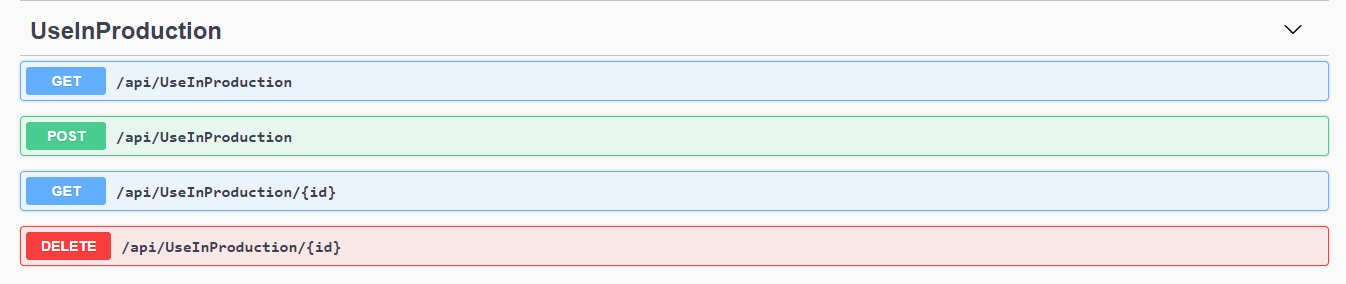


Рисунок 5.13 - CRUD запросы для работы с Отчётом об использовании материалов в производстве

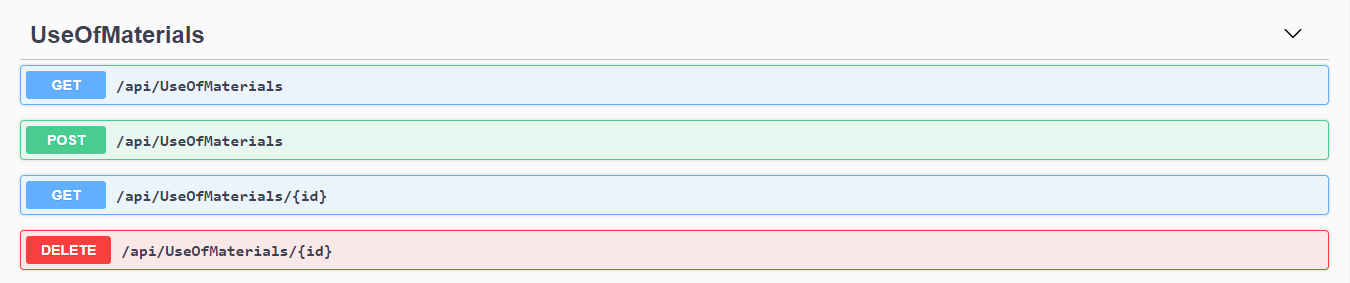


Рисунок 5.14 - CRUD запросы для работы с Отчётом об использовании ТМЦ

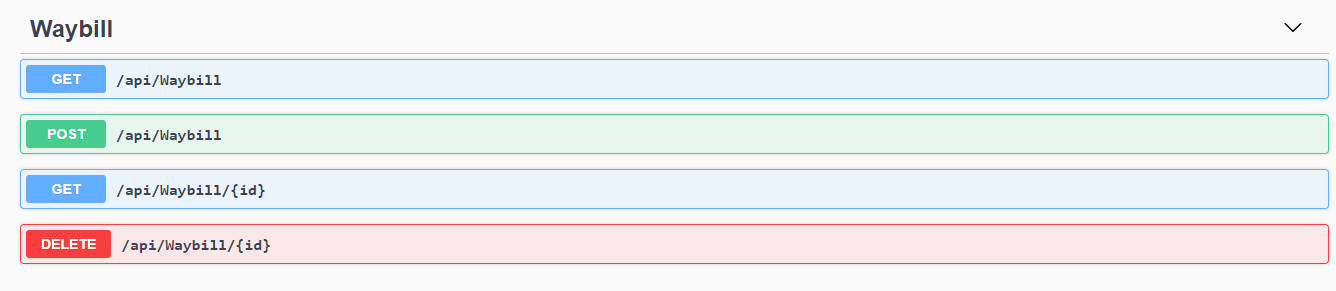


Рисунок 5.15 - CRUD запросы для работы с Накладной на внутреннее перемещение материалов



Рисунок 5.16 – Модель данных

6 Руководство пользователя

В рамках данного курсового проекта демонстрируется две стороны работы клиентского приложения:

* сторона, которой может пользоваться любой пользователь, которая демонстрирует информацию о сотрудниках предприятия и материалах, используемых на производстве
* админская сторона клиентского приложения, которая представляет собой web-службы RESTful.

Так же, учитывая то, что web-приложение было написано с использованием технологий .NET Core, Angular, то оно может использоваться на различных платформах, а именно Windows, macOS, IOS, Linux.

Так как приложение является кроссплатформенным, то его внешний вид не должен меняться в зависимости от устройства. Дизайн, как и у большинства современных приложений, простой, минималистичный, интуитивно понятный, цепляет выигрышно подобранной цветовой схемой.

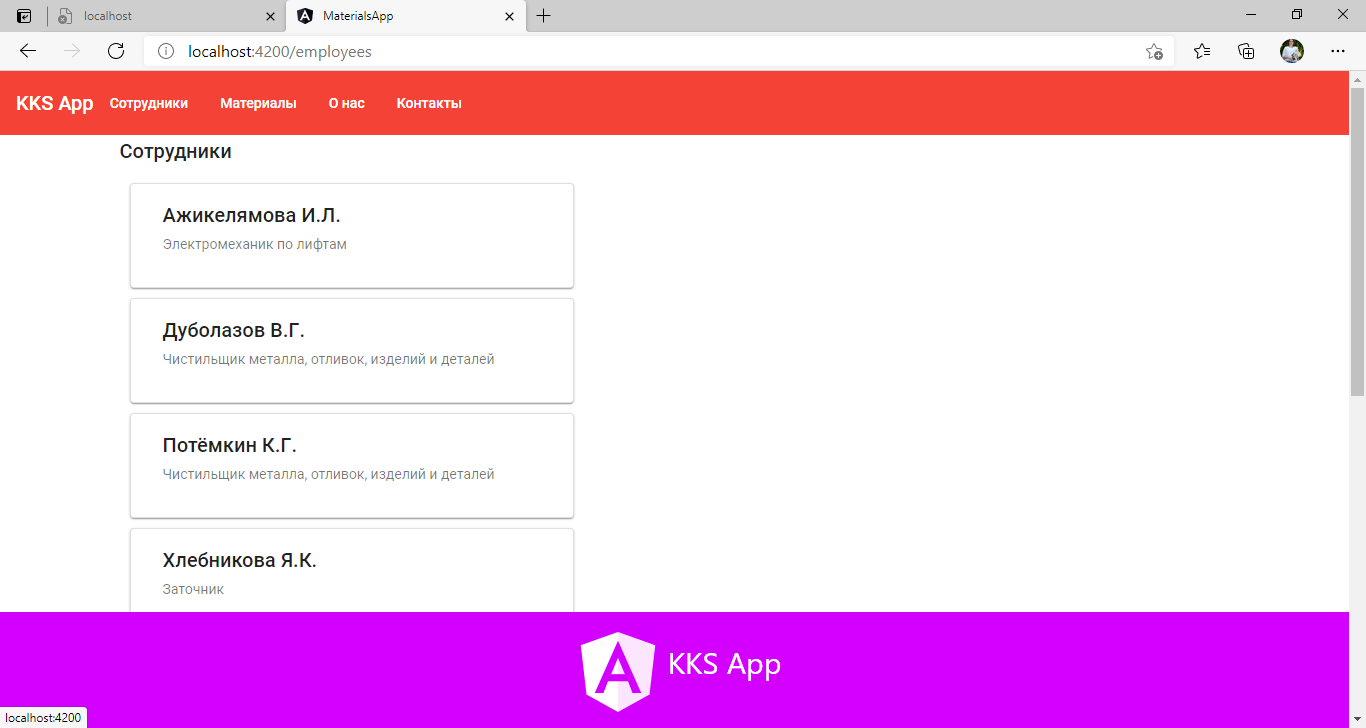


Рисунок 6.1 – Список сотрудников предприятия

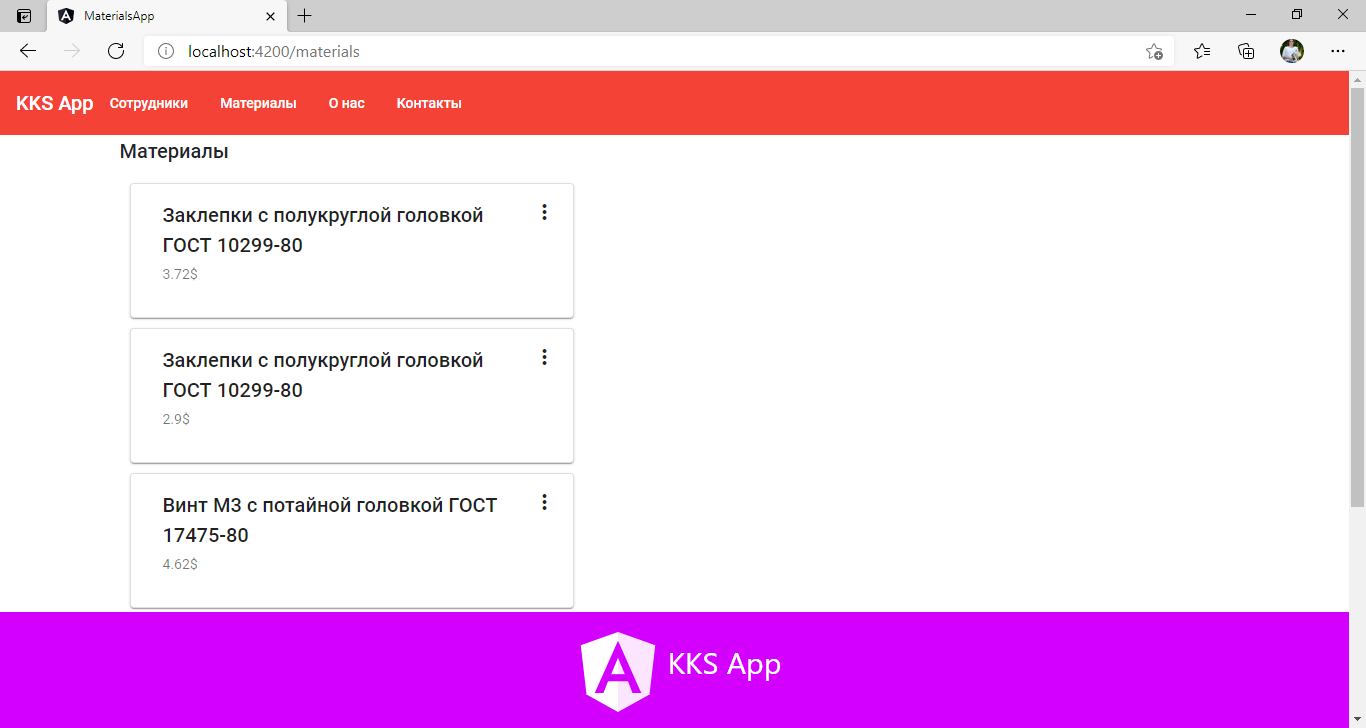


Рисунок 6.2 – Список материалов, используемых на производстве

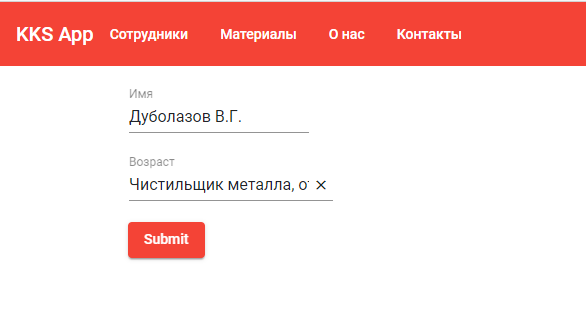


Рисунок 6.3 – Окно изменения информации о занимаемой должности работником

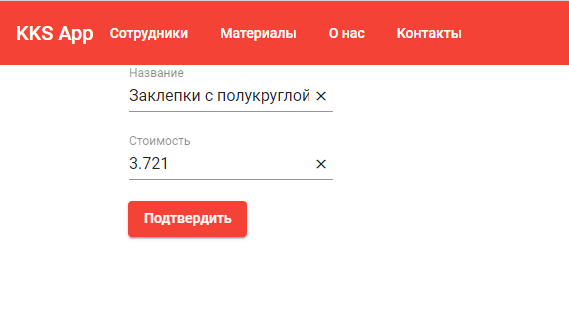


Рисунок 6.4 – Окно изменения стоимости материала, используемого на производстве

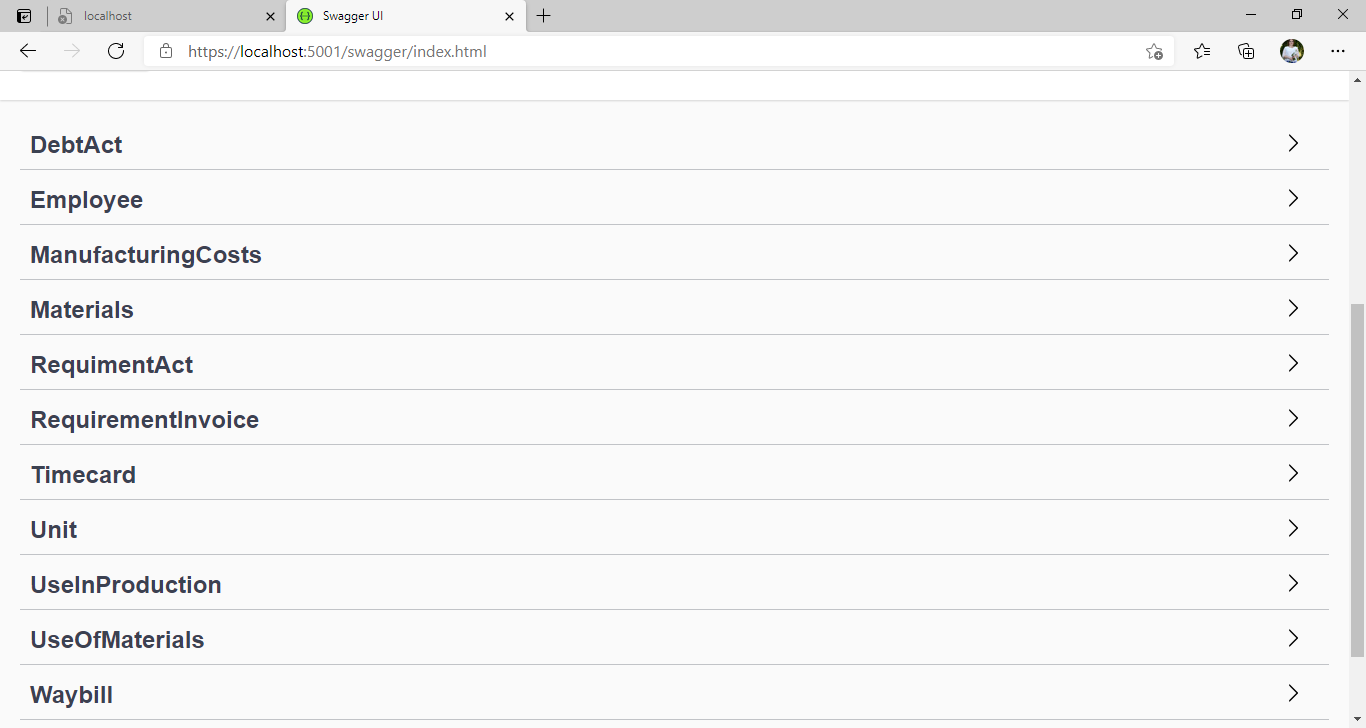


Рисунок 6.5 – Окно вывода всей необходимой информации, которая используется для оформления документации

Заключение

В настоящее время становится все труднее и труднее манипулировать окружающими нас вещами. Для этого сегодня и существуют технологии автоматизации. Мало осталось вещей, которые нельзя было автоматизировать. Людям все время облегчается работа с информацией. Раньше некоторые могли сидеть за одним делом часами, а то и днями. Но настали времен, когда стоит приложить маленькие усилие и все получится даже лучше, чем планировалось.

Во время выполнения курсовой работы было разработано многопользовательская информационная система учета товарно-материальных ценностей на ОАО «Могилёвлифтмаш».

При создании автоматизированной системы обработки информации использовался следующий стек технологий

* платформа .NET 5.0 и её основной язык C#
* web-фреймворк ASP.NET
* ORM Entity Framework Core
* фреймворк для клиентской части приложения Angular его основной язык TS
* использование web-служб RESTful
* инструмент визуального моделирования и проектирования на основе OMG UML Enterprise Architect, язык программирования C.

Были разработаны следующие элементы проектирования систем: диаграмма бизнес процессов, диаграмма классов АСОИ, диаграмма классов БД, диаграмма вариантов использования, диаграмма последовательности, диаграмма состояний.

Проведено тестирование реализованных систем. Проведенное тестирование показало целостность и правильность составленного кода взаимодействия с созданной базой данных

К достоинствам программного продукта можно отнести понятный для пользователя интерфейс.

Список использованных источников

* 1. Литвин, П. Разработка корпоративных приложений в SQL 2008. Для профессионалов / П.Литвин – СПб.: Питер; Киев: BHV, 2009. – 1008 с.
  2. [Бхамидипати](http://oz.by/books/more106510.html?id_search=127378#writer), К. SQL. Справочник программиста/ К.Бхамидипати–М.: Эком, 2003. – 304 с.
  3. Бьюли, А. Изучаем SQL / А.Бьюли, Э. Оппель. – М.: Символ, 2007. – 312 с.
  4. Жилинский, А. Самоучитель Microsoft SQL Server 2005 / А.Жилинский – СПб: BHV, 2004. – 224 с.
  5. Васильев, А. Н. C#. Объектно-ориентированное программирование. Учебный курс / А. Н. Васильев. – СПб. : Питер, 2012. – 320 с.: ил.
  6. Павловская, Т. А. С#. Программирование на языке высокого уровня: учебник для вузов / Т. А. Павловская. – СПб. : Питер, 2007. – 432 с.: ил.
  7. Подбельский, В. В. Язык С#. Решение задач / В. В. Подбельский. – М.: Инфра-М, 2014. – 296 с.
  8. Троелсен, Э. Язык программирования С# 4.0 и платформа .NET 4.0 : пер. с англ. / Э. Троелсен. – М. : Вильямс, 2011. – 1392 с.
  9. Фленов, М. Библия C# / М. Фленов. – СПб.: БХВ-Петербург, 2013. — 560 с.
  10. Еремин И.И., Астафьев Н.Н. Введение в теорию линейного и выпуклого программирования -М.: Наука, 1976 г. – 239 с.
  11. Общий курс высшей математики для экономистов. Учебник / под ред В.И. Ермакова.- М.: ИНФА - М. - 656 с. - (серия «высшее образование»).
  12. И.Г. Семакин Основы программирования: учебник для сред. проф. Образования / И.Г. Семакин, А.П.Шестаков. - 2-е изд., стер,- М.: Издательский центр «Академия», 2003.-432 с.