Введение

Для повышения конкурентоспособности производители вынуждены одновременно ускорять вывод новой продукции на рынок и увеличивать прозрачность и гибкость производства, оптимизировать прогнозирование и планирование, снижать объемы отходов и складских запасов, уменьшать время простоев; вместе с тем гарантируя оптимальное качество и эффективность производства по всем подразделениям.

Достижение этих целей требует интегрированной информационной инфраструктуры, которая помогает координировать производство в глобальном масштабе, в том числе в режиме реального времени. Подобная координация затрагивает спецификации, оборудование, технологические процессы, организационные процедуры, тесты качества и персонал производства.

Сегодня на рынке присутствуют системы, служащие для сквозного повышения производительности производства, начиная с цехового уровня. Они называются Системами Оперативного Управления Производственными Процессами MES (Manufacturing Execution Systems), и обычно рассматриваются как уровень, объединяющий бизнессистемы (например, ERP) и системы управления технологическими процессами.

Международная ассоциация поставщиков решений для промышленных предприятий MESA (Manufacturing Enterprise

Solutions Association) дает следующее определение MES: “Система оперативного управления производственными процессами (MES) — это динамическая информационная система, обеспечивающая эффективное исполнение производственных операций. Используя точные и актуальные данные, MES регулирует, инициирует и протоколирует работу предприятия в соответствии с происходящими событиями. Набор функций MES позволяет управлять процессами производства с момента выдачи заказа на производство и до момента выпуска готового изделия. MES предоставляет наиболее важную информацию о производственной деятельности для предприятия и всей цепочки поставок посредством двустороннего взаимодействия“. Обычно MES-решение решает следующие задачи:

* распределение и контроль статуса ресурсов,
* диспетчеризация производственных заказов,
* сбор данных, управление качеством,
* управление техническим обслуживанием,
* анализ производительности,
* составление производственных расписаний,
* контроль документов,
* управление трудовыми ресурсами,
* координация технологических процессов и отслеживание готовой продукции.

Множество исследований, проводимых различными аналитиками рынка и консалтинговыми фирмами, показывают, что MES обеспечивает существенные преимущества в повышении доходности, продуктивности и производительности технологических процессов для предприятий, использующих MES, по сравнению с теми предприятиями, которые этого не делают. По данным статистики MES обеспечивает:

* снижение продолжительности цикла производства в среднем на 45%;
* сокращение времени ввода данных, обычно на 75% или более;
* сокращение количества незавершенной продукции в среднем на 24%;
* снижение объема бумажной отчетности между сменами в среднем на 61%;
* сокращение времени освоения новой продукции в среднем на 27%;
* сокращение ненужной бумажной документации в среднем на 56%;
* сокращение объема брака в среднем на 18%.

В исследовании, проведенном журналом Industry Week, была роанализирована производительность предприятий, использующих MES, и предприятий, не использующих подобные системы. С 1998 по 2002 год было опрошено 106 американских производителей с целью определения наиболее эффективного предприятия США. Более половины заводов-изготовителей с лучшей производительностью используют системы оперативного управления производством (MES). Этот анализ представляет надежные доказательства того, что MES-системы для предприятий обеспечивают значительные преимущества по производительности. Доходность предприятий с MES была в 4 раза выше по сравнению с предприятиями без MES. Было показано, что системы MES повышают:

* продуктивность;
* технологическую эффективность.

Кроме того, MES влияет на следующие факторы:

* увеличение производственной культуры предприятия;
* развитие инициативности служащих.

Цель разработки дипломного проекта: управление производственными процессами в реальном времени, т.е. осуществление непрерывного контроля состояния производственного процесса.

Реализуемые функции:

* контроль состояния и распределение ресурсов;
* сбор и хранение данных;
* мониторинг качества;
* отслеживание готовой продукции;
* анализ эффективности.

Предполагаемые инструментальные средства и технологии проектирования и разработки: языки программирования C#, JavaScript, ASP.NET MVC 5, MS SQL, IIS, Microsoft Visio.

1. Аналитический обзор и постановка задачи

1.1 Обзор аналогов

Рынок различных систем производственного управления очень широк и разнообразен. Все системы разные, сравнивать их большого смысла не имеет.

Рассмотрим несколько MES систем:

**MEScontrol**[9] - готовое решение для полной автоматизации рецептурных, дискретных и машиностроительных производственных процессов.

|  |  |  |
| --- | --- | --- |
| **Общие сведения о системе** | | |
| Название | MEScontrol | |
| Разработчик | BrightEye (Бельгия, г. Беверен) | |
| Сертификаты разработчика | нет данных | |
| Язык интерфейса | Русский (английский, французский, голландский) | |
| Язык документации | Русский (английский) | |
| Сайт разработчика | [www.brighteye.be](http://www.brighteye.be/) | |
| **Характеристики MES-системы** | | |
| Тип установки | Самостоятельное решение | |
| Типы (отрасли) производства | http://www.mescenter.ru/images/stories/yes.png Машиностроение http://www.mescenter.ru/images/stories/yes.png Дискретное производство http://www.mescenter.ru/images/stories/yes.png Рецептурное производство http://www.mescenter.ru/images/stories/no.png ~~Непрерывное производство~~ http://www.mescenter.ru/images/stories/no.png ~~Управление распределением~~ | |
| Поддерживаемые функции по модели MESA | http://www.mescenter.ru/images/stories/yes.png RAS Контроль состояния и распределение ресурсов http://www.mescenter.ru/images/stories/yes.png ODS Оперативное/Детальное планирование http://www.mescenter.ru/images/stories/yes.png DPU Диспетчеризация производства http://www.mescenter.ru/images/stories/yes.png DOC Управление документами http://www.mescenter.ru/images/stories/yes.png DCA Сбор и хранение данных http://www.mescenter.ru/images/stories/yes.png LM Управление персоналом http://www.mescenter.ru/images/stories/yes.png QM Управление качеством http://www.mescenter.ru/images/stories/yes.png PM Управление производственными процессами http://www.mescenter.ru/images/stories/yes.png MM Управление техобслуживанием и ремонтом http://www.mescenter.ru/images/stories/yes.png PTG Отслеживание и генеалогия продукции http://www.mescenter.ru/images/stories/yes.png PA Анализ производительности http://www.mescenter.ru/images/stories/yes.png PB Продуктовый баланс | |
| Наличие преднастроенных конфигураций | Конфигурация "Дискретные процессы", "Рецептурные процессы", "Машиностроение" | |
| **Технические характеристики MES-системы** | | |
| Архитектура | Клиент-сервер | |
| Операционная система серверной части | Microsoft Windows Server 2008, 2008r2, 2012 32/64b | |
| Тип стандартного клиентского приложения | Толстый клиент, Microsoft Windows (XP, Vista, 7, 8) 32/64b | |
| Поддержка производственных терминалов | http://www.mescenter.ru/images/stories/yes.png Тачскрин на базе Windows CE, XP/Embeddded | |
| Поддержка мобильных клиентов | http://www.mescenter.ru/images/stories/yes.png На базе тонкого клиента, решения для погрузчиков и носимых со считывателями | |
| Поддержка систем идентификации | http://www.mescenter.ru/images/stories/yes.png Штрих-кодирование http://www.mescenter.ru/images/stories/yes.png RFID-идентификация | |
| **Интеграционные возможности** | | |
| Интерфейсы с ERP-системами | SAP, Microsoft Dynamics AX, Microsoft Dynamics NAV, 1С | |
| Поддерживаемые протоколы обмена с оборудованием | OPC | |
| Совместимость с ISA-95 | http://www.mescenter.ru/images/stories/yes.png Построение модели данных на основе рекомендаций стандарта ISA-95 http://www.mescenter.ru/images/stories/yes.png Поддержка B2MML | |
| Средства горизонтальной интеграции | | Web-сервисы, MSMQ, обменные xml-файлы |

**MES Pharis**[9]

Современная система цехового управления для дискретного и рецептурного производства с широким спектром применения.

|  |  |
| --- | --- |
| **Общие сведения о системе** | |
| Название | Pharis |
| Разработчик | UNIS, a.s. (Чешская республика, г. Брно) |
| Сертификаты разработчика | ISO 9001:2008 (2012 г.) - система менеджмента качества ISO/IEC 20000-1:2005 (2011 г.) - информационная безопасность CAA-TI-026 - проектирование в сфере гражданской авиации JAR-21 - производство авиакомпонентов COS 051622 (AQAP 2110) - реализация военных проектов |
| Язык интерфейса | Русский (чешский, английский) |
| Язык документации | Русский (чешский, английский) |
| Сайт разработчика | [www.unis.cz](http://www.unis.cz/) [www.pharis.cz](http://www.pharis.cz/) |
| **Характеристики MES-системы** | |
| Тип установки | Самостоятельное решение |
| Типы (отрасли) производства | http://www.mescenter.ru/images/stories/yes.png Машиностроение http://www.mescenter.ru/images/stories/yes.png Дискретное производство http://www.mescenter.ru/images/stories/yes.png Рецептурное производство http://www.mescenter.ru/images/stories/no.png ~~Непрерывное производство~~ http://www.mescenter.ru/images/stories/no.png ~~Управление распределением~~ |
| Поддерживаемые функции по модели MESA | http://www.mescenter.ru/images/stories/yes.png RAS Контроль состояния и распределение ресурсов http://www.mescenter.ru/images/stories/yes.png ODS Оперативное/Детальное планирование http://www.mescenter.ru/images/stories/yes.png DPU Диспетчеризация производства http://www.mescenter.ru/images/stories/yes.png DOC Управление документами http://www.mescenter.ru/images/stories/yes.png DCA Сбор и хранение данных http://www.mescenter.ru/images/stories/yes.png LM Управление персоналом http://www.mescenter.ru/images/stories/yes.png QM Управление качеством http://www.mescenter.ru/images/stories/yes.png PM Управление производственными процессами http://www.mescenter.ru/images/stories/yes.png MM Управление техобслуживанием и ремонтом http://www.mescenter.ru/images/stories/yes.png PTG Отслеживание и генеалогия продукции http://www.mescenter.ru/images/stories/yes.png PA Анализ производительности http://www.mescenter.ru/images/stories/no.png ~~PB Продуктовый баланс~~ |
| Наличие преднастроенных конфигураций | Производство пластиковых изделий (литьё под давлением) Металлообрабока |
| **Технические характеристики MES-системы** | |
| Архитектура | Клиент-сервер |
| Операционная система серверной части | Microsoft Windows Server |
| Тип стандартного клиентского приложения | Тонкий клиент (IE, Firefox), требуется Flash и Microsoft Silverlight |
| Поддержка производственных терминалов | http://www.mescenter.ru/images/stories/yes.png На базе компьютера, Microsoft Windows XP Embedded (XP, Vista, 7) |
| Поддержка мобильных клиентов | http://www.mescenter.ru/images/stories/yes.png PDA на базе Microsoft Windows Mobile, требуется поддержка Windows .Net |
| Поддержка систем идентификации | http://www.mescenter.ru/images/stories/yes.png Штрих-кодирование http://www.mescenter.ru/images/stories/yes.png RFID-идентификация |
| **Интеграционные возможности** | |
| Интерфейсы с ERP-системами | 1С-Предприятие SAP/R3 Microsoft Dynamics NAV (Navision) Microsoft Dynamics AX (Axapta) Entry Helios QI  K2 |
| Поддерживаемые протоколы обмена с оборудованием | EUROMAP63 Arburg OPC DA |
| Совместимость с ISA-95 | http://www.mescenter.ru/images/stories/yes.png Построение модели данных на основе рекомендаций стандарта ISA-95 http://www.mescenter.ru/images/stories/no.png ~~Поддержка B2MML~~ |

**1С:MES Оперативное управление производством** [9]

Программный продукт "1С:Предприятие 8. MES Оперативное управление производством" предназначен для решения оперативных производственных задач, повышения эффективности управления и контроля процесса производства. Наибольший эффект от внедрения конфигурации "MES Оперативное управление производством" достигается при совместном использовании с ERP-решением "1С:Управление производственным предприятием", редакция 1.3, когда задачи объемно-календарного планирования и выполнения экономических расчетов решаются на уровне ERP системы, а задачи пооперационной оптимизации и управления на MES – уровне. Также конфигурация "MES Оперативное управление производством" может использоваться как самостоятельное решение для комплексного управления производственным процессом в целом на пооперационном уровне (за исключением финансово – экономических расчетов).

|  |  |
| --- | --- |
| **Общие сведения о системе** | |
| Название | 1С:MES Оперативное управление производством |
| Разработчик | Фирма "1С" (Россия, г. Москва) Совместно с "Лабораторией систем управления и интегрированных информационных технологий" Казанского национального исследовательского технического университета им. А.Н.Туполева (Россия, г. Казань) |
| Сертификаты разработчика | ISO 9001:2011, ГОСТ РВ 15.002-2003 и СРПП ВТ (2013 г.) |
| Язык интерфейса | Русский |
| Язык документации | Русский |
| Сайт разработчика | [1c.ru](http://1c.ru/) [solutions.1c.ru/catalog/mes/features](http://solutions.1c.ru/catalog/mes/features) |
| **Характеристики MES-системы** | |
| Тип установки | Самостоятельное решение в составе комплекса программ |
| Типы (отрасли) производства | http://www.mescenter.ru/images/stories/yes.png Машиностроение http://www.mescenter.ru/images/stories/yes.png Дискретное производство http://www.mescenter.ru/images/stories/no.png ~~Рецептурное производство~~ http://www.mescenter.ru/images/stories/no.png ~~Непрерывное производство~~ http://www.mescenter.ru/images/stories/no.png ~~Управление распределением~~ |
| Поддерживаемые функции по модели MESA | http://www.mescenter.ru/images/stories/yes.png RAS Контроль состояния и распределение ресурсов http://www.mescenter.ru/images/stories/yes.png ODS Оперативное/Детальное планирование http://www.mescenter.ru/images/stories/yes.png DPU Диспетчеризация производства http://www.mescenter.ru/images/stories/no.png ~~DOC Управление документами~~ http://www.mescenter.ru/images/stories/yes.png DCA Сбор и хранение данных http://www.mescenter.ru/images/stories/yes.png LM Управление персоналом http://www.mescenter.ru/images/stories/no.png ~~QM Управление качеством~~ http://www.mescenter.ru/images/stories/yes.png PM Управление производственными процессами http://www.mescenter.ru/images/stories/no.png ~~MM Управление техобслуживанием и ремонтом~~ http://www.mescenter.ru/images/stories/yes.png PTG Отслеживание и генеалогия продукции http://www.mescenter.ru/images/stories/yes.png PA Анализ производительности http://www.mescenter.ru/images/stories/no.png ~~PB Продуктовый баланс~~ |
| **Технические характеристики MES-системы** | |
| Архитектура | Клиент-сервер |
| Операционная система серверной части | Microsoft Windows, Linux |
| Тип стандартного клиентского приложения | Толстый клиент, тонкий клиент, веб-клиент (Windows Internet Explorer, Mozilla Firefox, Google Chrome, Safari) |
| Поддержка производственных терминалов | http://www.mescenter.ru/images/stories/no.png Не поддерживаются |
| Поддержка мобильных клиентов | http://www.mescenter.ru/images/stories/no.png Не поддерживаются |
| Поддержка систем идентификации | http://www.mescenter.ru/images/stories/yes.png Штрих-кодирование http://www.mescenter.ru/images/stories/no.png ~~RFID-идентификация~~ |
| **Интеграционные возможности** | |
| Интерфейсы с ERP-системами | 1С:Управление производственным предприятием, редакция 1.3 |
| Поддерживаемые протоколы обмена с оборудованием | Опция |
| Совместимость с ISA-95 | http://www.mescenter.ru/images/stories/no.png ~~Построение модели данных на основе рекомендаций стандарта ISA-95~~ http://www.mescenter.ru/images/stories/no.png ~~Поддержка B2MML~~ |
| Средства горизонтальной интеграции | Средства интеграции поддерживаемые платформой 1С:Предприятие 8, 1С:PDM Управление инженерными данными, 1С:ТОИР Управление ремонтами и обслуживанием оборудования, веб-сервисы. |

Причины, почему принято решение разрабатывать собственную систему управления производством, и почему вышеприведенные системы не подходят для маленького производства приведены ниже:

* Дороговизна;
* Излишняя функциональность;
* Сложность;
* Не охватывают узкоспециализированные задачи производства;
* Внесение изменений в ПО очень долгий и трудоемкий процесс;

**1.2. Постановка задачи и требования к системе**

Интернет сегодня – это то, без чего не может жить подавляющая часть населения. В 21 веке уже трудно представить себе современного человека без страницы в социальной сети или просто без наличия смартфона с выходом в Интернет.

Исходя из этого принято решение разрабатывать web-приложение, чтобы каждый пользователь мог с легкостью ввести или просмотреть информацию со своего смартфона или компьютера.

Для разработки программного продукта была выбрана технология ASP.NET MVC 5 и языки C# и JavaScript. Основные преимущества технологии ASP.NET MVC:

* Архитектура MVC
  + Взаимодействие пользователя с MVC приложением следует естественному циклу: пользователь совершает действие, в ответ на это приложение меняет свою модель данных и предоставляет пользователю обновленный вид. А затем цикл повторяется. Это очень удобно для веб-приложений, предоставляемых в виде серии HTTP запросов и ответов.
  + Необходимость веб приложению объединять несколько технологий (например, базы данных, HTML и исполняемый код), как правило, разбивается на множество уровней или слоев. Моделей, которые вытекают из этих комбинаций, естественны для концепции MVC.
* Расширяемость. Внутренние компоненты настольного ПК являются независимыми частями, которые взаимодействуют только через стандартные, публично документированные интерфейсы. Вы можете легко вынуть видеокарту или жесткий диск и заменить его другим от другого производителя и будете уверены, что он впишется в слот и будет работать. MVC Framework также построен как ряд независимых компонентов, удовлетворяющих .NET интерфейс или построенных на абстрактном базовом классе, так что вы можете легко заменить компоненты, такие как система маршрутизации, движок для просмотра и так далее другими. ASP.NET MVC дизайнеры построили его таким образом, чтобы дать вам три варианта выбора для каждого компонента MVC Framework:
  + Использовать реализацию по умолчанию компонента в его нынешнем виде (чего должно быть достаточно для большинства приложений).
  + Вывести подкласс реализации по умолчанию для настройки ее поведения.
  + Заменить компонент полностью при помощи новой реализации интерфейса или абстрактного базового класса.
* Жесткий контроль над HTML и HTTP. ASP.NET MVC признает важность получения чистой, соответствующей стандартам разметки. Его встроенные методы HTML помощника предоставляют соответствующие стандартам выходные данные, но есть и более значительные философские изменений по сравнению с Web Forms. Вместо того чтобы плодить огромные участки HTML, котором нам сложно управлять, MVC Framework рекомендует вам выработать простой, элегантный стиль разметки с помощью CSS. Конечно, если вы хотите использовать некоторые готовые виджеты для сложных элементов пользовательского интерфейса, такие как выбор даты или каскадное меню, то вам стоит знать, что подход ASP.NET MVC к разметке упрощает использование лучших в своем роде UI библиотек, таких как JQuery UI или библиотеки Yahoo YUI. Разработчики JavaScript будут рады узнать, что ASP.NET MVC так хорошо сработался с популярной библиотеки JQuery, что Microsoft сделал JQuery встроенной частью шаблона проектов ASP.NET MVC и даже позволяет напрямую ссылаться на .js файл jQuery на собственных CDN серверах Microsoft. Страницы, сгенерированные ASP.NET MVC, не содержат никаких данных View State, поэтому они могут быть в сотни килобайт меньше, чем обычные страницы, созданные при помощи ASP.NET Web Forms. Несмотря на современную широкополосную связь и быстрые подключения, эта экономия пропускной способности до сих пор чрезвычайно притягательна для конечных пользователей. Как Ruby on Rails, ASP.NET MVC работает в гармонии с HTTP. Вы полностью контролируете запросы, проходящие между браузером и сервером, поэтому вы можете подогнать настройки под себя, на сколько вам это нравится. AJAX сделан просто, и нет никакого автоматического обратного вмешательства в состояния на стороне клиента. Любой разработчик, который в первую очередь фокусируется на веб программировании, почти наверняка посчитает это освобождением и будет наслаждаться рабочим процессом.
* Тестируемость. Естественное разнесение различных задач приложения по разным, независимым друг от друга частям программного обеспечения, поддерживаемое архитектурой MVC. позволяет изначально строить легко сопровождаемые и тестируемые приложения. Однако проектировщики ASP.NET MVC на этом не остановились. Для каждого фрагмента компонентно-ориентированного дизайна платформы они обеспечили структурированность, необходимую для выполнения требований модульного тестирования и средств макетирования.
* Мощная система маршрутизации (роутинга). Существует ряд веских причин для того, чтобы заботиться о структуре URL-адресов.
  + Во-первых, поисковые механизмы придают ключевым словам, содержащимся в URL, больший вес.
  + Во-вторых, многие веб-пользователи достаточно сообразительны, чтобы понять URL, и ценят возможность осуществления навигации путем ввода запроса в адресной строке своего браузера.
  + В-третьих, когда кто-то понимает структуру URL-адреса, они с большей вероятностью пройдут по нему, поделятся им с другими или даже зачитают его вслух по телефону.
  + В-четвертых, при таком подходе в Интернете не раскрываются технические нюансы, структура каталогов и имен файлов приложения; следовательно, вы вольны изменять лежащую в основе сайта реализацию, не нарушая работоспособности всех входящих ссылок.
* Современный API-интерфейс. Платформа Microsoft .NET развивалась с каждым крупным выпуском, поддерживая - и даже определяя - многие передовые аспекты современного программирования. Версия ASP.NET MVC 5 построена для .NET Framework 4.5.1, поэтому ее API-интерфейс может в полной мере задействовать последние новшества языка и исполняющей среды, в том числе ключевое слово await, расширяющие методы, лямбда-выражения, анонимные и динамические типы, а также язык интегрированных запросов (Language Integrated Query - LINQ). Многие методы и шаблоны кодирования API-интерфейса MVC Framework следуют более четкой и выразительной композиции, чем это было возможно в ранних платформах.
* Инфраструктура ASP.NET MVC имеет открытый код. В отличие от предшествующих платформ веб-разработки производства Microsoft, первоначальный исходный код ASP.NET MVC доступен для свободной загрузки и даже для модификации и компиляции с целью получения собственной версии этой инфраструктуры. Это буквально неоценимо при отладке кода, обращающегося к системному компоненту, когда требуется пошагово выполнить его код (и даже ознакомиться с комментариями программистов, написавших этот код). Это также полезно, если вы создаете усовершенствованный компонент и хотите видеть, какие существуют возможности разработки, или узнать, как действительно работают встроенные компоненты. Вдобавок упомянутая возможность удобна и тогда, когда не устраивает работа того или иного компонента, когда требуется найти ошибку или когда необходимо получить доступ к тому, что недоступно с помощью других средств - интересующий компонент можно просто изменить самостоятельно. Однако при этом придется отслеживать свои изменения и повторно их применять при модернизации до более новой версии инфраструктуры.

В качестве IDE была выбрана Visual Studio 2017 – это интегрированная среда разработки Microsoft (IDE), являющейся самой функциональной IDE для разработки на платформе.NET.

В качестве DI контейнера был выбран Ninject. Он прост, изящен и легок в использовании. Существуют более сложные альтернативы, но Ninject функционирует при минимальном конфигурировании.

Для создания реализаций интерфейсов, предназначенных для использования в модульных тестах был выбран Moq, представляющий собой комплект инструментов имитации.

Для хранения данных был выбрана СУБД SQL Server 2016 и ORM Entitty Framework от Microsoft. Так как оба продукта от Microsoft у них есть наиболее тесная интеграция с другими продуктами.

Так же использовались такие технологии, библиотеки и фреймворки как: AJAX, JSON, Bootstrap, Chart.js и др.

Целью дипломного проектирования является разработка web-приложения, предназначенного для решения задач синхронизации, координации, анализа и оптимизации выпуска продукции в рамках конкретного производства.

Перечень подзадач, выполнение которых позволит разработать программный продукт:

* регистрация нового пользователя;
* контроль состояния и распределение ресурсов;
* сбор и хранение данных о продукте на каждом этапе производства;
* мониторинг качества;
* отслеживание готовой продукции;
* анализ эффективности.

2 Методы и модели, положенные в основу проекта

Использовать данное приложение можно только в авторизованном состоянии. Если пользователь не авторизован его автоматически перенаправляет на страницу авторизации. Если пользователь оставит поля логина или пароля пустыми, или неверно заполненными и попытается войти на сайт, система предупредит об этом, и подскажет в чем ошибка. Если данные введены корректно, система проверит, есть ли такой пользователь в базе данных. Если есть, то система переведет пользователя на главную страницу приложения, если нет, то снова отобразит страницу авторизации и сообщит, что пользователя с таким именем не существует либо имя и пароль не совпадают.

Т.к приложение предназначено только для внутреннего использования организации, регистрировать новых пользователей может только администратор сайта.

Для регистрации нового пользователя будет предложено заполнить форму с полями, где надо будет указать имя, пароль, повторно пароль и выбрать фото для аккаунта. Если, какие-нибудь данные введены не корректно или пользователь с таким именем уже есть в базе данных система сообщит об этом.



Рисунок 1 — Диаграмма деятельности «Вход на сайт»

Авторизовавшись пользователь может на главной странице узнать какое количество продукции находится на каком этапе производства. Также он может пройти по одному из пунктов меню, которые ведут к определенному этапу производства, информации о складе либо аналитической информации. Также пользователь может просмотреть информацию о своём профиле либо изменить его.

Перейдя на страницу, отвечающую за определенный этап производства, пользователь может ввести новые данные либо просмотреть историю ранее введенной информации. Любые внесенные данные помечаются информацией, кто и когда их внес.



Рисунок 2 — Диаграмма деятельности «Добавление информации о сборке»

Изменять или удалять данные может только администратор. При удалении, к примеру, данных о сборке продукции, сначала отобразится всплывающее окно, с просьбой подтвердить удаление. Если пользователь нажмет «Отмена» или закроет окно удаление не произойдет. Иначе будет вызвана функция удаления. При удалении сначала система проверит, есть ли такие данные в БД. Если нет, то выведет сообщение об ошибке. Если данные есть, то система проверит, не задействованы ли они в дальнейших этапах. Если задействованы, то придется сперва удалить их оттуда. Если нет, то система удалит данные и вернет используемые в сборке детали на склад.



Рисунок 3 — Диаграмма деятельности «Удаление информации о сборке»

Для добавления деталей в брак необходимо перейти на соответствующую страницу. Для этого нажать на пункт меню «Склад деталей» и в раскрывшемся меню выбрать пункт «Брак». Далее нажать кнопку «Добавить в брак». В появившемся модальном окне необходимо выбрать при помощи выпадающего списка название детали, далее указать количество и дату. Далее программа проверяет, есть ли указанное количество выбранной детали на складе. Если нет, то выведет сообщение об ошибке. Если есть, то добавит указанное количество в брак и отнимет такое же количество со склада.



Рисунок 4 — Диаграмма деятельности «Добавление деталей в брак»

Изменить имя пользователя, пароль либо изображение профиля можно нажав на иконку профиля и в появившемся окне нажать кнопку «Изменить профиль». Далее будет открыта форма, в которой будет предложено изменить имя пользователя, пароль и изображение профиля. Если попытаться сохранить изменения, оставив пустые поля или введя не валидные данные, система выведет сообщение об ошибке. Иначе изменения будут сохранены.



Рисунок 5 — Диаграмма деятельности «Редактирование профиля»

3 Разработка проекта программного обеспечения

3.1 Функциональный анализ предметной области

Основная цель создания любой программной системы - создание такого программного продукта, который помогает пользователю выполнять свои повседневные задачи. Для создания таких программ первым делом определяются требования, которым должна удовлетворять система. Однако если дать пользователям написать эти требования на бумаге, то часто можно получить список функций, по которому трудно судить будет ли будущая система выполнять свое назначение и сможет ли она облегчить пользователю выполнение его работы вообще. Непонятно какие из выполняемых функций более важны и для кого.

    Для того, чтобы более точно понять как должна работать система, все чаще используется описание функциональности системы через варианты использования (Use Case или прецеденты). Варианты использования это - описание последовательности действий, которые может осуществлять система в ответ на внешние воздействия пользователей или других программных систем. Варианты использования отражают функциональность системы с точки зрения получения значимого результата для пользователя, поэтому они точнее позволяют ранжировать функции по значимости получаемого результата.

 Варианты использования предназначены в первую очередь для определения функциональных требований к системе и  управляют всем процессом разработки. Все основные виды деятельности такие как анализ, проектирование, тестирование выполняются на основе вариантов использования. Во время анализа и проектирования варианты использования позволяют понять как результаты, которые хочет получить пользователь влияют на архитектуру системы и как должны себя вести компоненты системы, для того чтобы реализовать нужную для пользователя функциональность.

    В процессе тестирования, описанные ранее варианты использования позволяют проще оценить точность реализации требований пользователей и позволяют провести пошаговую проверку этих требований.

    Стратегия использования прецедентов при определении требований определяет необходимость дополнительно к вопросу "что пользователи ждут от системы?" задавать вопрос "что система должна сделать для конкретного пользователя?". Такой подход позволяет искать функции, которые нужны многим пользователям и исключать те возможности, которые не могут помочь пользователям выполнять свои повседневные задачи.

Диаграмма вариантов использования состоит из актеров, для которых система производит действие и собственно действия Use Case, которое описывает то, что актер хочет получить от системы. Актер обозначается значком человечка, а Use Case - овалом. Дополнительно в диаграммы могут быть добавлены комментарии.

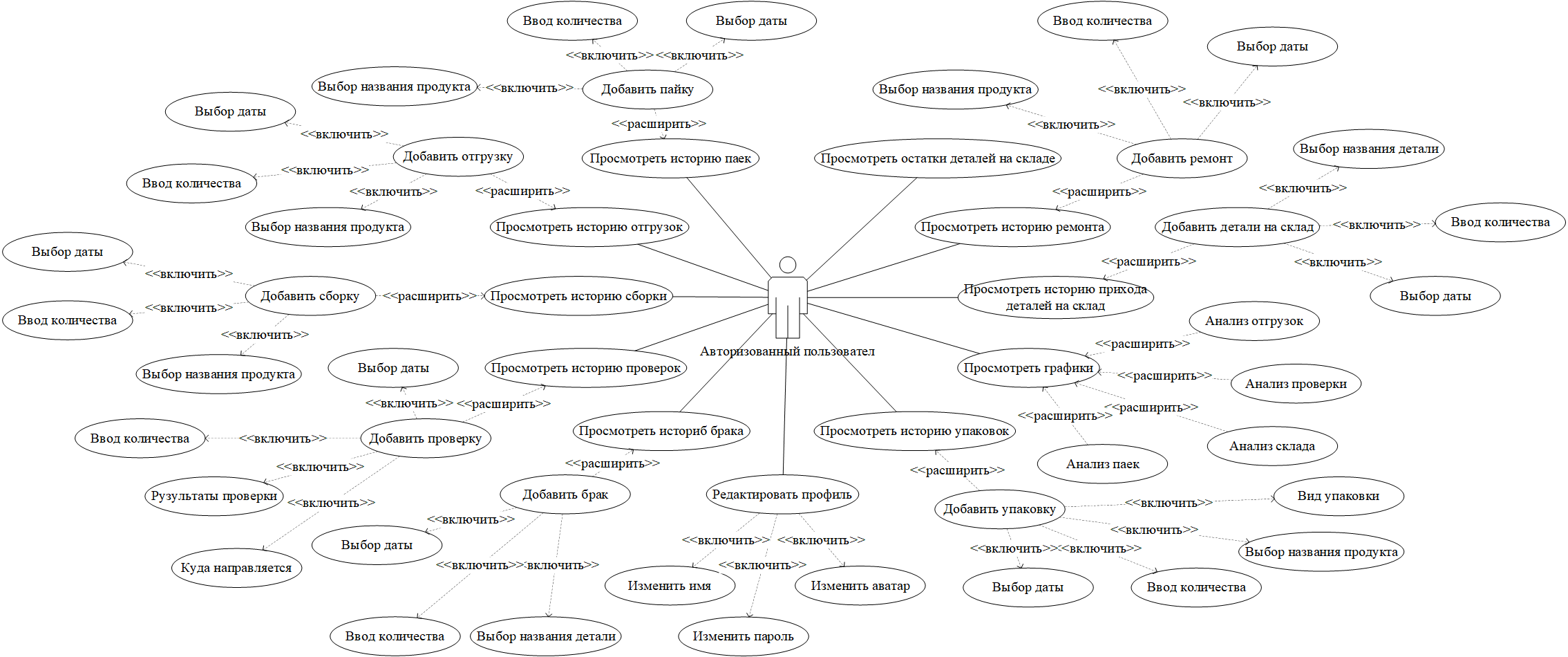
Ответы на следующие вопросы позволят определить актеров, взаимодействующих с системой:

* кто взаимодействует с системой или использует систему;
* кто передает или принимает информацию в/из системы;
* кто является внешним по отношению к системе.

Каждый вариант использования показывает, как конкретный актер использует систему и в дальнейшем расширяется диаграммами состояний и последовательности действий.

Между актерами и вариантами использования могут быть различные виды взаимодействия. Основные виды взаимодействия следующие:

* Простая ассоциация - отражается линией между актером и вариантом использования (без стрелки). Отражает связь актера и варианта использования. На рисунке между актером администратор и вариантом использования просматривать заказ.
* Направленная ассоциация - то же что и простая ассоциация, но показывает, что вариант использования инициализируется актером. Обозначается стрелкой.
* Наследование - показывает, что потомок наследует атрибуты и поведение своего прямого предка. Может применяться как для актеров, так для вариантов использования.
* Расширение (extend) - показывает, что вариант использования расширяет базовую последовательность действий и вставляет собственную последовательность. При этом в отличие от типа отношений "включение" расширенная последовательность может осуществляться в зависимости от определенных условий.
* Включение - показывает, что вариант использования включается в базовую последовательность и выполняется всегда (на рисунке не показан).



|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Пожеланиязаказчика (StakeholderNeeds, UserRequirements)** | **№ ф-ции** | **Функциональноетребование (Features, Functional Requirements)** |
| 1 | Регистрация нового пользователя в системе | 1.1  1.1  1.3 | Создание логина  пароля  имя пользователя |
| 2 | Вход зарегистрированного пользователя | 2.1  2.2  2.3  2.4  2.5 | Ввод логина.  Проверка логина.  Ввод пароля.  Проверка пароля.  Вход в систему. |
| 3 | Программа должна обеспечить просмотр профиля пользователя, его редактирование | 3.1  3.2  3.3 | Редактирование логина  Редактирование пароля Редактирование аватара |
| 4 | Программа должна обеспечить просмотр количества продукции на каждом этапе производства | 4.1 | Просмотр количества продукции на каждом этапе производства |
| 5 | Программа должна обеспечить добавление информации о сборках продукции | 5.1  5.2  5.3 | Выбор продукта  Ввод количества  Выбор даты |
| 6 | Программа должна обеспечить просмотр истории сборок продукции | 6.1  6.2  6.3  6.4  6.5 | Выбор диапазона дат  Отображение даты сборки  Отображение названия продукта  Отображение количества  Отображение имени пользователя, добавившего сборку |
| 7 | Программа должна обеспечить добавление информации о пайках продукции | 7.1  7.2  7.3 | Выбор продукта  Ввод количества  Выбор даты |
| 8 | Программа должна обеспечить просмотр истории паек продукции | 8.1  8.2  8.3  8.4  8.5 | Выбор диапазона дат  Отображение даты пайки  Отображение названия продукта  Отображение количества  Отображение имени пользователя, добавившего пайку |
| 9 | Программа должна обеспечить добавление информации о ремонтах продукции | 9.1  9.2  9.3  9.4 | Выбор продукта  Ввод количества  Выбор вида ремонта  Выбор даты |
| 10 | Программа должна обеспечить просмотр истории ремонтов продукции | 10.1  10.2  10.3  10.4  10.5  10.6 | Выбор диапазона дат  Отображение даты ремонта  Отображение названия продукта  Отображение вида ремонта  Отображение количества  Отображение имени пользователя, добавившего ремонт |
| 11 | Программа должна обеспечить добавление информации о проверках продукции | 11.1  11.2  11.3  11.4  11.5  11.6 | Выбор продукта  Ввод количества  Выбор вида проверки  Выбор даты  Ввод результатов проверки  Выбор дальнейшего направления |
| 12 | Программа должна обеспечить просмотр истории проверок продукции | 12.1  12.2  12.3  12.4  12.5  12.6  12.7  12.8 | Выбор диапазона дат  Отображение даты проверки  Отображение названия продукта  Отображение вида проверки  Отображение количества  Отображение количества целых  Отображение имени пользователя, добавившего проверку  Просмотр деталей проверки |
| 13 | Программа должна обеспечить добавление информации о упаковках продукции | 13.1  13.2  13.3  13.4 | Выбор продукта  Ввод количества  Выбор вида упаковки  Выбор даты |
| 14 | Программа должна обеспечить просмотр истории упаковок продукции | 14.1  14.2  14.3  14.4  14.5  14.6 | Выбор диапазона дат  Отображение даты упаковки  Отображение названия продукта  Отображение вида упаковки  Отображение количества  Отображение имени пользователя, добавившего упаковку |
| 15 | Программа должна обеспечить добавление информации о отгрузках продукции | 15.1  15.2  15.3  15.4 | Выбор продукта  Ввод количества  Выбор вида упаковки  Выбор даты |
| 16 | Программа должна обеспечить просмотр истории отгрузок продукции | 16.1  16.2  16.3  16.4  16.5  16.6 | Выбор диапазона дат  Отображение даты отгрузки  Отображение названия продукта  Отображение вида упаковки  Отображение количества  Отображение имени пользователя, добавившего отгрузку |
| 17 | Программа должна обеспечить добавление информации о приходе деталей на склад | 17.1  17.2  17.3 | Выбор детали  Ввод количества  Выбор даты |
| 18 | Программа должна обеспечить просмотр истории прихода деталей на склад | 18.1  18.2  18.3  18.4  18.5 | Выбор диапазона дат  Отображение даты прихода  Отображение названия детали  Отображение количества  Отображение имени пользователя, добавившего приход |
| 19 | Программа должна обеспечить добавление информации о браковке деталей | 19.1  19.2  19.3 | Выбор детали  Ввод количества  Выбор даты |
| 20 | Программа должна обеспечить просмотр истории браковки деталей | 20.1  20.2  20.3  20.4  20.5 | Выбор диапазона дат  Отображение даты браковки  Отображение названия детали  Отображение количества  Отображение имени пользователя, добавившего приход |
| 21 | Программа должна обеспечить просмотр остатков деталей на складе | 21.1  21.2  21.3 | Отображение названия детали  Отображение кода детали  Отображение количества |
| 22 | Программа должна обеспечить добавление новой детали | 22.1  22.2  22.3 | Ввод названия  Ввод кода  Количество 0 по умолчанию |
| 23 | Программа должна обеспечить редактирование детали | 23.1  23.2  23.3 | Редактирование названия  Редактирование кода  Редактирование количества |
| 24 | Программа должна обеспечить добавление нового продукта | 24.1 | Ввод названия |
| 25 | Программа должна обеспечить просмотр состава продукта | 25.1 | Просмотр состава продукта |
| 26 | Программа должна обеспечить изменение состава продукта | 26.1  26.2 | Добавление детали в состав  Удаление детали из состава |

3.2 Разработка схемы ресурсов системы

Существует множество различных видов и типов архитектур, которые успешно применяются. Одной их наиболее используемых является классическая трехуровневая система, которая подразумевает разделение приложения на три уровня.

Тут сразу надо сказать, что многоуровневой архитектурой часто обозначают два не совсем связанных понятия: n-layer и n-tier. И layer, и tier, как правило, обозначаются словом "уровень", иногда по отношению к "layer" еще употребляется слово "слой". Однако в обоих случаях уровни будут разного порядка.

Tier представляет физический уровень. То есть если мы говорим о трехуровневой архитектуре, то n-tier приложение могло быть разделено на такие уровни: сервер базы данных, веб-приложение на веб-сервере и браузер пользователя. То есть каждый уровень представлял бы особый отдельный физический процесс, даже если бы и сервер баз данных, и веб-сервер, и браузер пользователя находились бы на одном компьютере. Если бы в качестве клиента альтернативно использовалось мобильное приложение, то это был бы еще один физический уровень.

Особенность заключается в

том, что само Web приложение находится и выполняется на сервере, клиент при этом получает только результаты работы. Работа приложения основывается на получении запросов от пользователя (клиента), их обработке и выдачи результата. Передача запросов и результатов их обработки происходит через Интернет.

Отображением результатов запросов, а также приемом данных от

клиента и их передачей на сервер обычно занимается специальное

приложение – браузер (Internet Expolrer, Mozilla, Opera и т.д.). Как известно, одной из функций браузера является отображения данных, полученных из Интернета в виде страницы, описанной на языке HTML, следовательно, результат, передаваемый сервером клиенту должен быть представлен на этом языке.

На стороне сервера Web приложение выполняется специальным

программным обеспечением (Web сервером), который и принимает запросы клиентов, обрабатывает их, формирует ответ в виде страницы, описанной на языке HTML и передает его клиенту. Одним из таких Web серверов является Internet Information Services (IIS) компании Microsoft.

Layer представляет логический уровень. То есть у нас может быть уровень доступа к данным, уровень бизнес-логики, уровень представления, уровень сервисов и так далее. При этом логические уровни не совпадают с физическими. Так, обычно уровень предоставления в приложении ASP.NET содержит и контроллеры, которые обрабатывают ввод, и представления, которые отображаются в веб-браузере, то есть разделяется на два физических уровня.

Классическая трехуровневая система состоит из следующих уровней:

N-Layer Architecture in ASP.NET

Presentation layer (уровень представления): это тот уровень, с которым непосредственно взаимодействует пользователь. Этот уровень включает компоненты пользовательского интерфейса, механизм получения ввода от пользователя. Применительно к asp.net mvc на данном уровне расположены представления и все те компоненты, который составляют пользовательский интерфейс (стили, статичные страницы html, javascript), а также модели представлений, контроллеры, объекты контекста запроса.

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

Data Access layer (уровень доступа к данным): хранит модели, описывающие используемые сущности, также здесь размещаются специфичные классы для работы с разными технологиями доступа к данным, например, класс контекста данных Entity Framework. Здесь также хранятся репозитории, через которые уровень бизнес-логики взаимодействует с базой данных.

При этом надо отметить, что крайние уровни не могут взаимодействовать между собой, то есть уровень представления (применительно к ASP.NET MVC, контроллеры) не могут напрямую обращаться к базе данных и даже к уровню доступа к данным, а только через уровень бизнес-логики.

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

Компоненты, как правило, должны быть слабосвязанными (loose coupling), поэтому неотъемлемым звеном многоуровневых приложений является внедрение зависимостей.

При чем об ASP.NET MVC мы говорим прежде всего применительно к уровню представления, остальные же уровни могут быть реализованы независимо и могут использоваться в приложениях на других технологиях, как Windows Forms, WPF и т.д. И, как правило, все приложение в целом будет представлять решение (solution) в Visual Studio, а отдельные уровни - проекты. В то же время неверно полагать, что если уровень обязательно должен соответствовать отдельному проекту. При необходимости мы можем раздробить один уровень на несколько проектов, главное, чтобы его функционал представлял единое логическое звено.

Схема ресурсов системы представлена в графической части (РТДП 5033.046.01).

3.3 Разработка структуры базы данных

В качестве платформы для БД использовалась Microsoft SQL Server 2016. БД представлена таблицами: ArrivalOfDetails, Assemblies, Boxings, CheckJmts, DefectDetails, Details, GroupProducts, Products, ProductStates, Repairs, Roles, Shipments, Solderings, StructureOfTheProducts, Users.

Сущность ArrivalOfDetails хранит информацию о поступлении деталей на склад. Имеет атрибуты: идентификатор, идентификатор детали, дата поступления, количество поступивших деталей, идентификатор пользователя.

Сущность Assemblies хранит информацию о сборке продукции. Имеет атрибуты: идентификатор, идентификатор собираемой продукции, идентификатор пользователя, количество собираемой продукции, дата сборки.

Сущность Boxings хранит информацию о процессе упаковки продукции. Имеет атрибуты: идентификатор, идентификатор продукции, идентификатор пользователя, количество продукции, дата упаковки, вариант упаковки.

Сущность CheckJmts хранит информацию о процессе проверки продукции. Имеет атрибуты: идентификатор, идентификатор продукции, идентификатор пользователя, информацию от куда поступила продукция, количество продукции, дата проверки, атрибуты показывающие результаты проверки и на какие операции она будет перемещена.

Сущность DefectDetails хранит информацию о браке деталей. Имеет атрибуты: идентификатор, идентификатор детали, дата браковки, количество бракованных деталей, идентификатор пользователя.

Сущность Details хранит информацию о детали. Имеет атрибуты: идентификатор, название, уникальный код, количество на складе, к какой группе относится.

Сущность GroupProducts хранит информацию о группе выпускаемой продукции. Имеет атрибуты: идентификатор, название.

Сущность Products хранит информацию о выпускаемой продукции. Имеет атрибуты: идентификатор, название.

Сущность ProductStates хранит информацию о том, какое количество продукции находится на каком этапе производства. Имеет атрибуты: идентификатор, идентификатор продукта, идентификатор этапа производства, количество.

Сущность Repairs хранит информацию о процессе ремонта продукции. Имеет атрибуты: идентификатор, идентификатор продукции, идентификатор пользователя, количество продукции, дата ремонта, вариант ремонта.

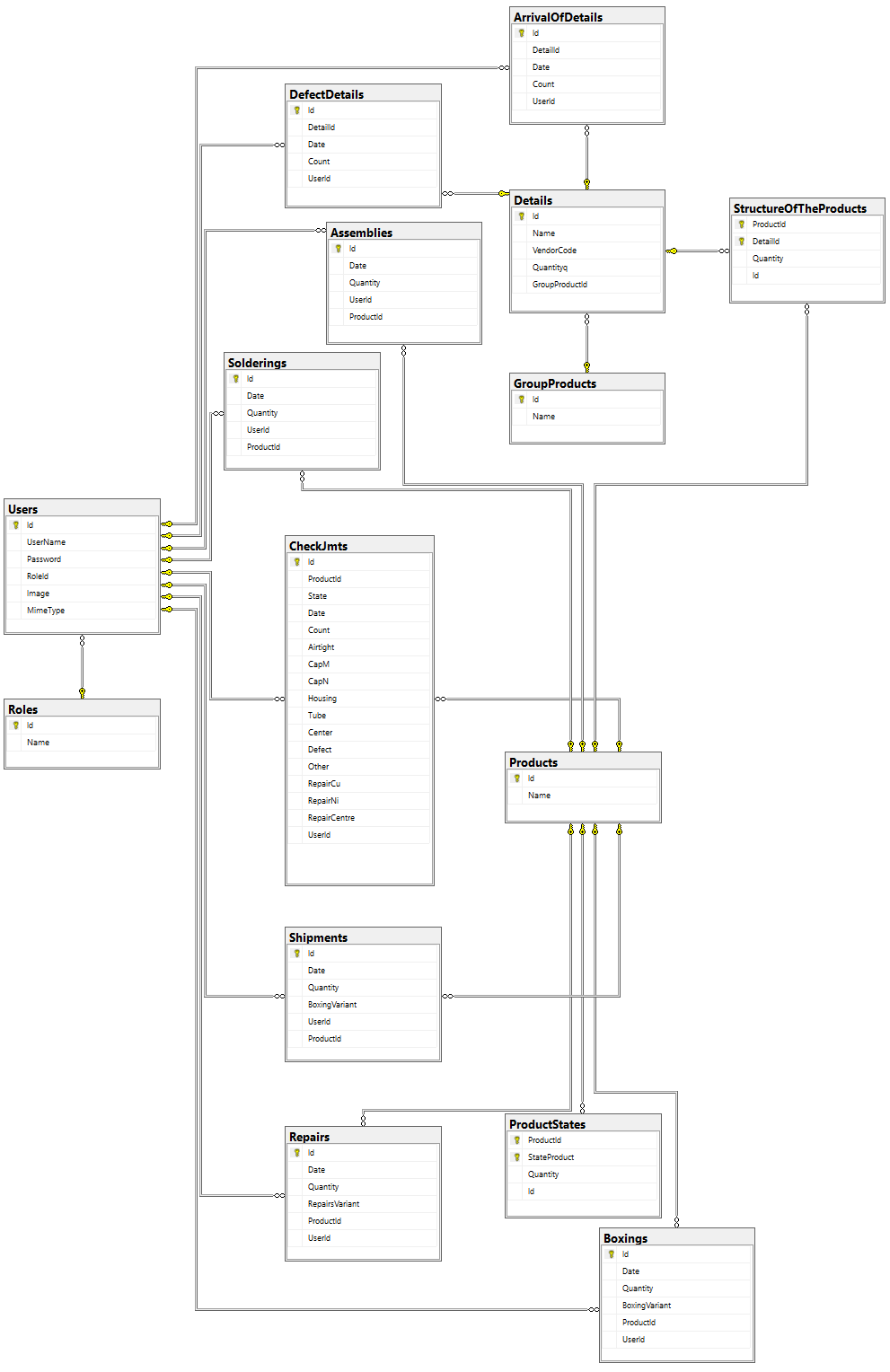
Сущность Roles хранит информацию о роли пользователя. Имеет атрибуты: идентификатор и название.

Сущность Shipments хранит информацию о процессе отгрузки готовой продукции. Имеет атрибуты: идентификатор, идентификатор продукции, идентификатор пользователя, количество продукции, дата отгрузки, вариант упаковки.

Сущность Solderings хранит информацию о процессе пайки. Имеет атрибуты: идентификатор, идентификатор продукции, идентификатор пользователя, количество продукции, дата пайки.

Сущность StructureOfTheProducts хранит информацию о составе продукта. Имеет атрибуты: идентификатор, идентификатор продукции, идентификатор детали, количество деталей.

Сущность Users хранит информацию о пользователях. Имеет атрибуты: идентификатор, идентификатор роли, имя пользователя, пароль, изображение.



3.4 Структура классов приложения

Краткие теоретические сведения о диаграммах классов

Диаграмма классов определяет типы классов системы и различного рода статические связи, которые существуют между ними. На диаграммах классов изображаются также атрибуты классов, операции классов и ограничения, которые накладываются на связи между классами. Вид и интерпретация диаграммы классов существенно зависит от точки зрения (уровня абстракции): классы могут представлять сущности предметной области (в процессе анализа) или элементы программной системы (в процессах проектирования и реализации).

Основные элементы диаграммы классов

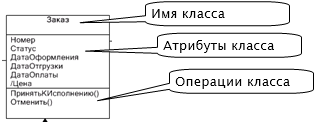


Таблица 3.2 - Описание классов приложения уровня DAL

|  |  |
| --- | --- |
| **Название класса** | **Описание класса** |
| IMesContext | Интерфейс контекста |
| MesContext | Класс контекста сущностной модели базы данных, содержащей сведения о всех сущностях бизнес-логики приложения. |
| IUnitOfWork | Объект UnitOfWork будет содержать ссылки на репозиторий пользователей. |
| UnitOfWork | Класс UnitOfWork наследуется от интерфейса IUnitOfWork и предоставляет доступ к репозиториям через отдельные свойства и определяет общий. Кроме того, данный класс содержит дополнительные методы Save() и Dispose(), которые в иной ситуации мы могли б определить в репозиториях. Но так как этот функционал будет общим для обоих репозиториев, то его лучше вынести в класс UnitOfWork. |
| IBaseRepository | Обобщенный интерфейс репозитория |
| BaseRepository | Обобщенный класс репозитория, позволяет абстрагироваться от конкретных подключений к источникам данных, с которыми работает программа, и является промежуточным звеном между классами, непосредственно взаимодействующими с данными, и остальной программой |
| BoxingVariant | Перечисление вариантов упаковки |
| RepairsVariant | Перечисление вариантов ремонта |
| StateFoTest | Перечисление вариантов проверки |
| VariantStateProduct | Перечисление этапов производства |
| IdProvider | Класс провайдера ID |
| ArrivalOfDetail, Assembly, Boxing, CheckJmt, DefectDetail, Detail, GroupProduct, Product, ProductState, Repair, Role, Shipment, Soldering, StructureOfTheProduct, User | Сущностные классы уровня доступа к данным. Наследуются от класса IdProvider |

Таблица 3.2 - Описание классов приложения уровня BLL

|  |  |
| --- | --- |
| **Название класса** | **Описание класса** |
| AutmapperConfigBLL | Класс для настройки конфигурации автомаппера |
| IdentityExtensions | Данный класс создает два метода расширения GetUserId и GetUserRole для получения из аутентификационного тикета id и роли пользователя. |
| OperationDetails | Данный класс будет хранить информацию об успешности операции. Свойство Succedeed указывает, успешна ли операция, свойство Message будет хранить сообщение об ошибке, Accessory для доп. информации |
| ServiceModule | Класс представляет специальный модуль Ninject, который служит для организации сопоставления зависимостей |
| IAdminService. IAnalyticsService, IArrivalService, IAssemblyService, IBoxingService, ICheckJmtService, IDefectService, IHomeService, IRepairService, IShipmentService, ISolderingService, IStockService, IUserService | Интерфейсы, определяют методы бизнес логики |
| AdminService. AnalyticsService, ArrivalService, AssemblyService, BoxingService, CheckJmtService, DefectService, HomeService, RepairService, ShipmentService, SolderingService, StockService, UserService | Классы сервисов бизнес логики, наследуются от соответствующих интерфейсов. В конструкторе принимают объект IUnitOfWork, через который идет взаимодействие с уровнем DAL. Реализуют методы бизнес логики |
| ArrivalOfDetailDTO, AssemblyDto, BoxingDto, CheckJmtDto, CheckJmtForListDto, ChekDetailsDto, DefectDetailDisplayDto, DefectDetailDto, DetailDTO, DetailInProductDto, DisplayArrivalOfDetailDto, IdProvider, LoginDto, ProductDTO, ProductStateDto, RepairDto, ShipmentChartDto, ShipmentDto, SolderingCountDto, SolderingDto, StructureOfTheProductDto, UserDto | Сущностные классы бизнес-уровня |

Таблица 3.2 - Описание классов приложения уровня представления

|  |  |
| --- | --- |
| **Название класса** | **Описание класса** |
| AutoMapperConfigWeb | Класс для настройки конфигурации автомаппера |
| BundleConfig | Класс регистрации бандлов. Метод RegisterBundles добавляет все создаваемые бандлы в коллекцию bundles |
| FilterConfig | Класс  глобальных фильтров. В методе **RegistreGlobalFilters** осуществляет регистрацию глобальных фильтров в коллекцию filters |
| NinjectRegistrations | Класс NinjectRegistrations наследуется от класса NinjectModule и фактически представляет модуль Ninject. Он переопределяет метод Load(), который вызывается при загрузке модуля. И с помощью вызова Bind<IT>().To<T>(); собственно устанавливается сопоставление между интерфейсом-зависимостью и конкретным классом этого интерфейса. |
| RouteConfig | Класс определения маршрутов. |
| Startup | Класс OWIN. С помощью метода расширения UseCookieAuthentication() здесь устанавливается, что при аутентификации пользователей будут использоваться куки: AuthenticationType = "ApplicationCookie". |
| AccountController, AdminController, AnalyticsController, ArrivalController, AssemblyController, BoxingController, CheckJmtController, DefectController, ErrorController, FinishedGoodsWarehouseController, HomeController, RepairController, SolderingController, StockController | Классы контроллеров, содержат методы,  которые обрабатывают запросы по определенному URL.  Отвечают за Авторизацию и аутентификацию, действия администратора, отображение графиков, приход комплектующих, сборку продукции, упаковку продукции, проверку продукции, браковку деталей, отображение страниц с ошибками, отгрузку готовой продукции, отображение главной страницы, ремонт продукции, пайку продукции, склад деталей соответствено. |
| ArrivalOfDetailVm, AssemblyVm, BoxingVm, CheckJmtForListVm, CheckJmtVm, ChekDetailsVm, DateVm, DefectDetailDisplayVm, DefectDetailVm, DetailInProductVm, DetailVm, DisplayArrivalOfDetailVm, IdProvider, LoginVm, ProductStateVm, ProductVm, RegisterVm, RepairVm, ShipmentChartVm, ShipmentVm, SolderingCountVm, SolderingVm | Сущностные классы уровня представления |

4. Разработка алгоритмов и их программная реализация

4.1 Алгоритм удаления сборки

Функция удаления сборки доступна только администратору. Для этого он кликает по соответствующей кнопке в таблице, отображающей список сборок. Id сборки которую хотим удалить передается в функцию и по этому id находим сборку в бд. Далее возвращаем детали, задействованные в сборке обратно на склад. Если эта сборка не задействована в дальнейших процессах, то отнимаем количество собранной в этой сборке продукции, от общего количества собранной продукции, удаляем данные о сборке и сохраняем изменения в бд. Иначе генерируем исключение и изменения в бд не вносятся.



4.2 Алгоритм добавления детали в состав продукта

Изменять состав продукции может только администратор. На странице конструктора, выбрав необходимый продукт нажимаем кнопку «Добавить в состав продукта». В появившемся модальном окне, при помощи выпадающего списка, выбираем название детали, которую хотим добавить в состав. Так же вводим, какое количество выбранной детали необходимо для сборки единицы продукции. После нажатия на кнопку добавления, система проверяет, не входит ли уже данная деталь в состав продукта. Если нет, то введенная информация добавляется в состав, если деталь уже присутствует в составе, то система выведет соответствующее сообщение.



4.3 Алгоритм просмотра истории паек

Просматривать историю раек может любой авторизованный пользователь. Для этого надо перейти на страницу «Пайки». На странице сразу же отобразится история паек за текущий месяц. Для просмотра истории за другой диапазон дат, необходимо выбрать в форме выбора диапазона дат, дату начала и дату конца.



4.4 Алгоритм редактирования детали

Для редактирования информации о детали, необходимо нажать соответствующую кнопку в списке деталей, после чего будет открыто модальное окно, с полями, заполненными текущей информацией о детали. После редактирования необходимых полей, если поля заполнены валидными данными, при нажатии на кнопку «Сохранить изменения», происходит изменение информации в бд и список деталей обновляется.



4.5 Алгоритм добавления упаковки

Добавить данные об упаковке готовой продукции может любой авторизованный пользователь. Для этого на странице упаковки нажав на кнопку «Добавить упаковку» в появившемся модальном окне указать какое количество, какой продукции куда направляется. Далее система проверяет, есть ли указанное количество продукции в наличии. Если есть, то переводит указанное количество из состояния «Проверено» в состояние, указанное при упаковке. Если нет, то выводит сообщение об ошибке.



5 Тестирование программного средства

Число основанных на web-технологиях программных продуктов с каждым днем увеличивается. При этом растет уровень сложности приложений, и для их разработки требуются все более длительные сроки и широкий спектр операций по тестированию, в том числе при их модернизации и переходе от одной версии к другой. Качество разработки программных продуктов во многом зависит от эффективности информационных систем, и любая ошибка может стать причиной значительных финансовых потерь для организации.

Тестирование веб-приложений имеет несколько основных направлений:

* Функциональное тестирование – предусматривает автоматизированную, полуавтоматизированную или ручную проверку программного продукта для подтверждения стабильности работы всех его компонентов и их соответствия бизнес-требованиям;
* Проверка пользовательского интерфейса и кроссбраузерное тестирование – подразумевают анализ взаимодействия приложения с пользователем и проверку его работы в разных браузерах;
* Анализ удобства пользования – выявляет в навигации и пользовательском интерфейсе места, которые могут быть непонятны для пользователя, недостаточно либо избыточно информативны;
* Нагрузочное тестирование – направлено на проверку стабильности функционирования программного продукта при предполагаемой рабочей и пиковых нагрузках.

Типы тестов по покрытию (по глубине)

Smoke test – тестирование системы для определения корректной работы базовых функций программы в целом, без углубления в детали. При проведении теста определяется пригодность сборки для дальнейшего тестирования.

Minimal Acceptance Test (MAT, Positive test): тестирование системы или ее части только на валидных данных (валидные данные – это данные, которые необходимо использовать для корректной работы модуля/функции). При тестировании проверяется правильной работы всех функций и модулей с валидными данными. Для крупных и сложных приложений используется ограниченный набор сценариев и функций.

Acceptance Test (AT): полное тестирование системы или ее части как на корректных, так и на некорректных данных/сценариях. Вид теста, направленный на подтверждение того, что приложение может использоваться по назначению при любых условиях. Тест на этом уровне покрывает все возможные сценарии тестирования: проверку работоспособности модулей при вводе корректных значений; проверку при вводе некорректных значений; использование форматов данных отличных от тех, которые указаны в требованиях; проверку исключительных ситуаций, сообщений об ошибках; тестирование на различных комбинациях входных параметров; проверку всех классов эквивалентности; тестирование граничных значений интервалов; сценарии не предусмотренные спецификацией и т.д.

В данном дипломном проекте были проведены smoke-тесты. В таблице 5.1 приведены smoke-тесты, которые были проведены в процессе тестирования программного средства.

Таблица 5.1 «Smoke-тестирование»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Модуль** | **Подмодуль** | **Описание теста** | **Ожидаемый результат** | **Прохождение теста** |
| MES\_T\_01 | Авторизация и аутентификация | Регистрация в системе нового пользователя: | Регистрация в системе нового пользователя:  1. Нажать на кнопку «Добавить пользователя».  2. Ввести в поле «Имя»: Иван.  3. Ввести в поле «Пароль»: 111111.  4. Ввести в поле «Подтвердите пароль»: 111111. 5. Нажать кнопку «Зарегистрировать». | 1. Отображается форма регистрации с полями для ввода имени, пароля и подтверждения пароля.  2. Появляется в поле «Имя» Иван.  3. Появляется в поле пароль символы, указывающие на ввод пароля.  4. Появляется в поле «Подтвердите пароль» символы, указывающие на ввод пароля.  5. Отображается сообщение с текстом «Пользователь Иван успешно зарегистрирован» | Тест пройден |
| MES\_T\_02 | Авторизация и аутентификация | Регистрация | Повторная регистрация в системе зарегистрированного ранее пользователя:  1. Нажать на кнопку «Добавить пользователя».  2. Ввести в поле «Имя»: Иван.  3. Ввести в поле «Пароль»: 111111.  4. Ввести в поле «Подтвердите пароль»: 111111. 5. Нажать кнопку «Зарегистрировать». | 1. Отображается форма регистрации с полями для ввода имени, пароля и подтверждения пароля.  2. Появляется в поле «Имя» Иван.  3. Появляется в поле пароль символы, указывающие на ввод пароля.  4. Появляется в поле «Подтвердите пароль» символы, указывающие на ввод пароля.  5. Отображается сообщение с текстом «Пользователь с таким именем уже зарегистрирован» | Тест пройден |
| MES\_T\_03 | Авторизация и аутентификация | Авторизация | Авторизация в системе пользователя с корректными данными:  1. Ввести Имя «Иван»  2. ввести пароль «111111»  3. нажать кнопку «Вход». | 1. В графе «Имя» появляется «Иван».  2. В графе пароль появляются символы, указывающие на ввод пароля.  3. Осуществляется переход на главную страницу. Вверху страницы отображается приветствие и имя пользователя | Тест пройден |
| MES\_T\_04 | Авторизация и аутентификация | Авторизация | Авторизация в системе пользователя с некорректными данными:  1. Ввести Имя «Иван2»  2. ввести пароль «111111»  3. нажать кнопку «Вход». | 1. В графе «Имя» появляется «Иван2».  2. В графе пароль появляются символы, указывающие на ввод пароля.  3. Пользователь получает сообщение «Неверный логин или пароль». | Тест пройден |
| MES\_T\_05 | Авторизация и аутентификация | Проверка прав администратора | 1.Авторизоваться в системе, как администратор  2. Перейти на страницу «Пайки» | 1. В меню слева есть пункты «Добавить пользователя» и «Конструктор».  2. В истории паек, напротив каждой записи есть кнопка «Удалить» | Тест пройден |
| MES\_T\_06 | Авторизация и аутентификация | Проверка прав пользователя | 1.Авторизоваться в системе, как пользователь  2. Перейти на страницу «Пайки» | 1. В меню слева нет пунктов «Добавить пользователя» и «Конструктор».  2. В истории паек, напротив каждой записи нет кнопки «Удалить» | Тест пройден |
| MES\_T\_07 | Авторизация и аутентификация | Авторизация | 1. Авторизоваться в системе.  2. В настройках браузера удалить cookie и обновить страницу | 1. Осуществляется переход на главную страницу. Вверху страницы отображается приветствие и имя пользователя  2. Осуществляется переход на страницу авторизации с полями для ввода имени и пароля | Тест пройден |
| MES\_T\_08 | Пайка | Добавление пайки | 1. Перейти на главную страницу.  Убедится, что количество собранных 5200-01 больше 10 (если нет добавить).  Запомнить количество собранных и количество спаянных.  2. Перейти на страницу «Пайки».  3. Нажать кнопку «Добавить пайку»  4. В поле «Название продукта» выбрать 5200-01  5. В поле количество ввести 10.  6. Нажать кнопку «Сохранить изменения»  7. Перейти на главную страницу и сравнить количество собранных  8. Сравнить количество спаянных | 1. Откроется страница с таблицей, показывающей, какое количество продукта находится на каждом этапе производства  2. Откроется страница, отображающая историю паек  3. Откроется модальное окно. В поле дата будет стоять сегодняшняя дата  4. В поле «Название продукта» отображается 5200-01  5. В поле «Количество» отображается 10.  6. Модальное окно закроется, отображается сообщение с текстом «Пайка успешно добавлена» и в истории паек появится запись с информацией о добавленной пайке  7. Количество собранных уменьшится на 10  8. Количество спаянных увеличится на 10 | Тест пройден |
| MES\_T\_09 | Пайка | Добавление пайки | 1. Перейти на главную страницу.  Убедится, что количество собранных 5200-01 меньше 10 (если нет добавить пайку).  Запомнить количество собранных и количество спаянных.  2. Перейти на страницу «Пайки».  3. Нажать кнопку «Добавить пайку»  4. В поле «Название продукта» выбрать 5200-01  5. В поле количество ввести 10.  6. Нажать кнопку «Сохранить изменения»  7. Перейти на главную страницу и сравнить количество собранных  8. Сравнить количество спаянных | 1. Откроется страница с таблицей, показывающей, какое количество продукта находится на каждом этапе производства  2. Откроется страница, отображающая историю паек  3. Откроется модальное окно. В поле дата будет стоять сегодняшняя дата  4. В поле «Название продукта» отображается 5200-01  5. В поле «Количество» отображается 10.  6. Модальное окно закроется, отображается сообщение с текстом «Пайка не добавлена, столько не собрали  7. Количество собранных не изменится  8. Количество спаянных не изменится | Тест пройден |
| MES\_T\_10 | Пайка | Удаление пайки | Перед началом теста войти на сайт как администратор и выполнить тест MES\_T\_08  1. Перейти на главную страницу.  Запомнить количество спаянных (если 0, то добавить пайку).  2. Перейти на страницу «Пайки».  3. Нажать кнопку «Удалить» напротив последней пайки  4. Перейти на главную страницу и сравнить количество спаянных | 1. Откроется страница с таблицей, показывающей, какое количество продукта находится на каждом этапе производства  2. Откроется страница, отображающая историю паек  3. Отображается сообщение с текстом «Пайка успешно удалена». Запись о последней пайке удалена из истории  4. Количество спаянных не уменьшится на 10 | Тест пройден |

В данной части было произведено функциональное тестирование приложения. Все тесты успешно пройдены, что позволяет сказать, что приложение спроектировано и разработано надлежащим образом.

Продолжение таблицы 5.1

Заключение

Результатом разработки стало клиент-серверное web-приложение с удобным и понятным интерфейсом, для работы с которым пользователю не требуется специальных знаний и навыков в области компьютерных технологий.

В результате выполнения дипломного проекта реализованы функции, которые были выявлены при функциональном анализе. Таким образом, пользователь Web-приложения пользователь получает функционал:

* контроль состояния и распределение ресурсов;
* сбор и хранение данных;
* мониторинг качества;
* отслеживание готовой продукции;
* анализ эффективности.

Данное приложение обеспечивает управление производственными процессами в реальном времени, т.е. осуществление непрерывного контроля состояния производственного процесса.

В процессе выполнения дипломного проекта в соответствии с поставленными задачами был проведен анализ программ предназначенных для управлением производством, на основании чего выявлены функции разрабатываемого программного средства.

В ходе выполнения работы были построены UML модели системы, разработана структура базы данных, произведен выбор языка и средств программирования.

Приложение реализовано на языке программирования C# (среда разработки Microsoft Visual Studio 2017 Community) с использованием фреймворка ASP.NET MVC 5.

В качестве IDE была выбрана Visual Studio 2017 – это интегрированная среда разработки Microsoft (IDE), являющейся самой функциональной IDE для разработки на платформе.NET.

В качестве DI контейнера был выбран Ninject. Он прост, изящен и легок в использовании. Существуют более сложные альтернативы, но Ninject функционирует при минимальном конфигурировании.

Для создания реализаций интерфейсов, предназначенных для использования в модульных тестах был выбран Moq, представляющий собой комплект инструментов имитации.

Для хранения данных был выбрана СУБД SQL Server 2016 и ORM Entitty Framework от Microsoft. Так как оба продукта от Microsoft у них есть наиболее тесная интеграция с другими продуктами.

Так же использовались такие технологии, библиотеки и фреймворки как: AJAX, JSON, Bootstrap, Chart.js и др.

Безусловно, планируется дальнейшее расширение заявленного функционала. В следующей версии приложения появится возможность постановки задачи различным пользователям и контроль их выполнения. Появятся внутренняя почта и чат. Также будут охвачены дополнительные этапы производства.

Данное приложение размещено на хостинге и внедрено в производство. И уже получило положительные отзывы от заказчика.

Список использованных источников

1. METANIT.COM. Сайт о программировании. [Электронный ресурс]. – Режим доступа: http://www.metanit.com/.
2. Фримен, А. ASP.NET MVC 4 с примерами на C# 5.0 для профессионалов. 4-е изд. – Россия : Вильямс, 2013. – 688 с.
3. Теория и практика UML. Диаграмма деятельности [Электронный ресурс] — Режим доступа: http://it-gost.ru/articles/view\_articles/96. — Дата доступа: 05.09.2017.
4. professorweb.ru. Сайт о программировании. [Электронный ресурс]. – Режим доступа: https://professorweb.ru/
5. learn.javascript.ru. Современный учебник Javascript. [Электронный ресурс]. – Режим доступа: https://learn.javascript.ru/
6. chartjs.org. Документация Chart.js. [Электронный ресурс]. – Режим доступа http://www.chartjs.org/docs/latest/
7. bootstrap-3.ru. Bootstrap 3 - Документация на русском языке. [Электронный ресурс]. – Режим доступа: http://www.bootstrap-3.ru/index.php
8. Microsoft. Developer Network. [Электронный ресурс]. – Режим доступа: http://www.msdn.microsoft.com/.
9. MES-центр. некоммерческое сообщество. [Электронный ресурс]. – Режим доступа: http://www.mescenter.ru/index.php

## **Приложение А (Обязательное) Листинг кода**

using System;

using System.Collections.Generic;

using System.Data.Entity;

using System.Linq;

using System.Threading.Tasks;

using AutoMapper;

using MES.BLL.DTO;

using MES.BLL.Infrastructure;

using MES.BLL.Interfaces;

using MES.DAL.Entities;

using MES.DAL.Interfaces;

namespace MES.BLL.Services

{

public class StockService : IStockService

{

private readonly IUnitOfWork \_uof;

public StockService(IUnitOfWork uof)

{

\_uof = uof ?? throw new ArgumentNullException(nameof(uof));

}

/// <summary>

/// Возвращает сколько деталей расходуется на 1 продукт

/// </summary>

/// <param name="name">Название продукта</param>

/// <returns>список деталей</returns>

public async Task<IEnumerable<DetailDTO>> GetDetailProduct(string name)

{

return await \_uof.StructureOfTheProducts.Entities.Where(w => w.Product.Name == name).Select(x => new DetailDTO

{

Name = x.Detail.Name,

GroupProductId = x.Detail.GroupProductId,

Quantityq = x.Detail.Quantityq / x.Quantity

}).ToListAsync();

}

public async Task<DetailDTO> GetDetail(int id) =>

Mapper.Map<DetailDTO>(await \_uof.Details.GetAsync(id));

public async Task<OperationDetails> CreateDetail(DetailDTO detailDto)

{

try

{

if (await \_uof.Details.Entities.AnyAsync(a=>a.Name==detailDto.Name || a.VendorCode==detailDto.VendorCode))

{

throw new Exception("Такая деталь уже есть в базе");

}

var detail = new Detail

{

Name = detailDto.Name,

VendorCode = detailDto.VendorCode,

Quantityq = 0,

GroupProductId = 1

};

\_uof.Details.Create(detail);

await \_uof.Commit();

return new OperationDetails(true, "Данные удалены", "/Stock/StockBalanceJmt");

}

catch (Exception e)

{

return new OperationDetails(false, e.Message, "");

}

}

public async Task<OperationDetails> EditDetail(DetailDTO detailDto)

{

try

{

var detail = await \_uof.Details.GetAsync(detailDto.Id);

detail.Name = detailDto.Name;

detail.Quantityq = detailDto.Quantityq;

detail.VendorCode = detailDto.VendorCode;

\_uof.Details.Update(detail);

await \_uof.Commit();

return new OperationDetails(true, "Данные изменены", "/Stock/StockBalanceJmt");

}

catch (Exception e)

{

return new OperationDetails(false, e.Message, "");

}

}

/// <summary>

/// Возвращает список деталей группы ЖМТ

/// </summary>

/// <returns>список деталей</returns>

public List<DetailDTO> GetDetailsJmt() => Mapper.Map<IEnumerable<Detail>, List<DetailDTO>>(\_uof.Details.Entities.Where(w => w.GroupProduct.Name == "JMT").ToList());

public async Task<OperationDetails> DeleteDetailAsync(int id)

{

try

{

var detail = await \_uof.Details.GetAsync(id);

if (detail == null) throw new Exception("Нет такой детали в базе");

\_uof.Details.Delete(id);

await \_uof.Commit();

return new OperationDetails(true, "Данные удалены", "");

}

catch (Exception)

{

\_uof.Rollback();

return new OperationDetails(false, "Данные удалены", "");

}

}

}

}