СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc169117145)

[1 СВЕДЕНИЯ О ПРЕДПРИЯТИИ 5](#_Toc169117146)

[1.1 История компании 5](#_Toc169117147)

[1.2 Охрана труда и техника безопасности на рабочем месте 7](#_Toc169117148)

[1.3 Технологии используемые на предприятии, программное 10](#_Toc169117149)

[Обеспечение 10](#_Toc169117150)

[2 ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ 12](#_Toc169117151)

[2.1 Технологии решения поставленной задачи 12](#_Toc169117152)

# ВВЕДЕНИЕ

В современном мире одним из самых важных ресурсов является информация. С каждым днём её становится всё больше и больше, поэтому возникает проблема хранения и её эффективной обработки. Для решения данной проблемы были созданы информационные системы. Информационная система представляет собой совокупность методов и средств, которые используются для хранения и обработки информации. Существует множество типов и вариаций информационных систем.

Наиболее используемым типом информационной системы является клиент-серверная система. Данный тип системы представляет собой взаимодействие структурных компонентов, где структурными компонентами являются сервер и узлы-поставщики определённых сервисов, а также клиенты, которые пользуются данным сервисом. Данный тип системы наиболее часто используется в создании корпоративных баз данных, в которой база данных является главным элементом, а все необходимые операции с базой выполняются сервером. Запросы на получение и изменение информации из базы данных отправляют клиенты. Сервер обрабатывает запросы и возвращает ответ клиенту. Преимуществом такой системы является её достаточно высокий уровень производительности за счёт распределения вычислительной нагрузки между клиентом и сервером, а также непротиворечивость данных за счёт централизованной обработки.

Целью данного задания по практике является проектирование и создание базы данных в выбранной СУБД и разработка веб-приложения, которое обеспечивает отображение, редактирование и обработку информации из разработанной базы данных. Структура базы данных должна быть нормализована – таблицы базы данных должны удовлетворять требованиям третьей нормальной формы. База данных должна содержать тестовый набор данных (не менее 100 записей у таблицы на стороне отношения «один» и не менее 10000 записей у таблицы на стороне отношения «многие»).

Для решения поставленной задачи в качестве СУБД используется *MS Sql Server*. Данная СУБД обеспечивает поддержку баз данных очень большого объёма и обработку сложных запросов, а также имеет эффективные алгоритмы для работы с памятью и автоматизированным контролем размера файлов баз данных.

В качестве технологии для разработки приложения используется платформа *ASP.NET Core MVC*. Данная платформа является многофункциональной платформой для создания веб-приложений с помощью структуры проектирования *Model-View-Controller* (модель-контроллер-представление). Структура архитектуры *MVC* предполагает разделение приложения на три основных компонента: модель, представление и контроллер. Каждый компонент решает свои задачи и взаимодействует с другими компонентами. Т.е. данная структура позволяет реализовать принципы разделения задач. Благодаря такой схеме связей и распределения обязанностей между компонентами процесс масштабирования приложения становится проще, т.к. облегчается процесс написания кода, выполнения отладки и тестирования компонентов.

# 1 СВЕДЕНИЯ О ПРЕДПРИЯТИИ

Строительство завода [сельскохозяйственных машин](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D0%BB%D1%8C%D1%81%D0%BA%D0%BE%D1%85%D0%BE%D0%B7%D1%8F%D0%B9%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D1%8B%D0%B5_%D0%BC%D0%B0%D1%88%D0%B8%D0%BD%D1%8B) в [Гомеле](https://ru.wikipedia.org/wiki/%D0%93%D0%BE%D0%BC%D0%B5%D0%BB%D1%8C) начато в [1928 году](https://ru.wikipedia.org/wiki/1928_%D0%B3%D0%BE%D0%B4). Днём рождения завода «Гомсельмаш» считается [15 октября](https://ru.wikipedia.org/wiki/15_%D0%BE%D0%BA%D1%82%D1%8F%D0%B1%D1%80%D1%8F) [1930 года](https://ru.wikipedia.org/wiki/1930_%D0%B3%D0%BE%D0%B4), когда выдал первую плавку литейный цех.

Выход завода на проектную мощность и его успешная работа в 30-е годы позволили [СССР](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D1%8E%D0%B7_%D0%A1%D0%BE%D0%B2%D0%B5%D1%82%D1%81%D0%BA%D0%B8%D1%85_%D0%A1%D0%BE%D1%86%D0%B8%D0%B0%D0%BB%D0%B8%D1%81%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D1%85_%D0%A0%D0%B5%D1%81%D0%BF%D1%83%D0%B1%D0%BB%D0%B8%D0%BA) полностью отказаться от импорта десятков наименований машин для кормопроизводства, зернового хозяйства, льно- и коноплеводства, первичной переработки [лубяных культур](https://ru.wikipedia.org/wiki/%D0%9B%D1%83%D0%B1%D1%8F%D0%BD%D1%8B%D0%B5_%D0%BA%D1%83%D0%BB%D1%8C%D1%82%D1%83%D1%80%D1%8B) — теперь такими машинами обеспечивал сельхозпроизводителей «Гомсельмаш». В 1940 году 18 из 26 наименований продукции «Гомсельмаша» были включены в советский экспорт.

## **1.1 История компании**

Гомсельмаш ведёт отсчёт своей истории с 1930 года. От производства простых сельхозмашин – до создания и массового производства зерно-, кормоуборочных комбайнов, комплексов машин на базе универсальных энергосредств, косилок, свеклоуборочной, картофелеуборочной и другой сельскохозяйственной техники - такой путь прошёл Гомсельмаш в XX веке.

Перелистывая страницы истории Гомсельмаша, нередко можно встретить упоминания о событиях, применительно к которым использовались слова «первый», «впервые в стране», «впервые в мире», «на первом месте». Выстроенные в хронологической последовательности без каких-либо комментариев, эти фрагменты истории создают масштабную картину постоянного стремления к лидерству, движения вперед.

**1930** Выпуск первой партии приводных дисковых силосорезок на строящемся «Гомсельмаше» стал ПЕРВЫМ шагом к созданию в стране ранее не существовавшей отрасли – машиностроения для кормопроизводства.

**1932** Наряду с производством лицензионных силосорезок и соломорезок, на «Гомсельмаше» ВПЕРВЫЕ в мире создана универсальная кормоприготовительная машина «Универсалка». Благодаря круглогодичному использованию, в то время она не имела равных по экономической эффективности.

**1940** «Гомсельмаш» стал ПЕРВЫМ предприятием в сельхозмашиностроении страны, которое производило одновременно 26 моделей машин для кормопроизводства, зернового хозяйства, семеноводства, первичной переработки льна и конопли.

**1956** «Гомсельмаш» ВПЕРВЫЕ в стране начал массовое производство нового вида техники – прицепного силосоуборочного комбайна СК-2,6. За ним последовали модели СК-2,6А; СКБ-2,6; УКСК-2,6; КС-2,6…

**1966** ВПЕРВЫЕ в практике машиностроения на заводе внедрена технология радиальной штамповки звездочек цепных передач, позволившая в 25-30 раз повысить производительность труда на обработке зубьев звездочек, и в полтора раза повысить их износостойкость. Технология впоследствии была запатентована в США, Канаде, Великобритании, Франции, Бельгии и Швейцарии.

**1971** Конструкторами «Гомсельмаша» создан и передан в производство КС-1,8 «Вихрь» – ПЕРВЫЙ прицепной кормоуборочный комбайн, который мог убирать не только кукурузу, но и травы.

**1977** Началось производство ПЕРВОГО в СССР самоходного кормоуборочного комбайна КСК-100. Комбайн также стал ПЕРВОЙ в стране самоходной сельхозмашиной с гидроприводом ходовой части. В его конструкции ВПЕРВЫЕ в практике отечественного кормоуборочного машиностроения были применены сменные навесные адаптеры. Эта востребованная машина и ее модификации производились в течение 32 лет.

**1984** По объемам годового производства самоходных кормоуборочных комбайнов «Гомсельмаш» вышел на ПЕРВОЕ место в мире.

**1988** ВПЕРВЫЕ в практике отечественного сельскохозяйственного машиностроения «Гомсельмаш» начал производство сельхозмашины нового типа – универсального энергосредства УЭС-250 и агрегатируемого с УЭС полунавесного кормоуборочного комбайна.

**1992** Создан КДП-3000 – ПЕРВЫЙ прицепной кормоуборочный комбайн, предназначенный для работы с тракторами от 2 до 5 тягового класса.

**1999** На Гомсельмаше ВПЕРВЫЕ в мире создан блочно-модульный зерноуборочный комплекс на базе универсального энергосредства.

**2001** ПЕРВЫМ из производителей стран СНГ «Гомсельмаш» приступил к серийному производству зерноуборочного комбайна, имеющего современную компоновочную схему с центральным расположением кабины (КЗС-7).

**2003** Универсальное энергосредство УЭС-2-250А стало ПЕРВОЙ в мире самоходной машиной, которая использовалась на кошении трав, заготовке измельченных кормов, уборке зерновых и сахарной свеклы, а также на предпосевной обработке почвы с одновременным посевом и внесением удобрений.

**2004** Гомсельмаш приступил к производству самоходных кормоуборочных комбайнов [FS80 (КВК-800)](https://gomselmash.by/produktsiya/kormouborochnye-kombainy/kvk-800-palesse-fs80/) с двигателем мощностью 360 л.с. Для оснащения комбайна более мощным двигателем была проведена доработка конструкции комплекса FS80 (КВК-800), и с 2008 года выпускаются его модификации с двигателями мощностью 450 л.с.

**2005** Изготовлены первые картофелеуборочные комбайны.

**2006** На Гомсельмаше ВПЕРВЫЕ в СНГ создан зерноуборочный комбайн 6-го класса [GS12 (КЗС-1218)](https://gomselmash.by/produktsiya/zernouborochnye-kombainy/kzs-1218-palesse-gs12/). В дальнейшем комбайн стал одним из самых востребованных в своем классе.

**2008** Гомсельмаш стал ПЕРВЫМ в мире производителем, который в течение 10 лет со дня создания опытного образца своего первого зерноуборочного комбайна смог разработать и поставить на производство 5 базовых моделей этих сложных сельскохозяйственных машин.

**2009** ПЕРВАЯ партия комбайнов собрана на совместном предприятии в Китае. В дальнейшем объемы совместного производства достигли тысяч кормоуборочных и початкоуборочных комбайнов.

**2010** Комбайн совместного белорусско-казахстанского производства ESSIL-760 (GS12) стал ПЕРВЫМ зерноуборочным комбайном 6-го класса, собранным в Казахстане.

**2012** Гомсельмаш приступил к производству кормоуборочных комбайнов FS8060 с двигателем мощностью 632 л.с. Такая высокопроизводительная машина создана ВПЕРВЫЕ в СНГ.

**2013** Успешно завершены приемочные испытания ПЕРВЫХ в СНГ высокопроизводительных зерноуборочных комбайнов с шириной молотилки 1700 мм (GS14 и GS16).

**2014** Разработка и испытания хлопкоуборочной машины [ХМП-1,8](https://gomselmash.by/produktsiya/khlopkouborochnaya-tekhnika/hmp-1-8/).

**2013-2015** Модельный ряд зерноуборочных комбайнов ПАЛЕССЕ расширен с трех до шести базовых моделей. Он включает модели GS575, [GS812](https://gomselmash.by/produktsiya/zernouborochnye-kombainy/kzs-812-palesse-gs812/), [GS10](https://gomselmash.by/produktsiya/zernouborochnye-kombainy/kzs-10k-palesse-gs10/), [GS12](https://gomselmash.by/produktsiya/zernouborochnye-kombainy/kzs-1218-palesse-gs12/), GS14 и GS16.

**2015** Создано второе совместное предприятие в Китае.

**2016** Выпущен в серию модернизированный зерноуборочный комбайн [GS12A1.](https://www.gomselmash.by/produktsiya/zernouborochnye-kombainy/kzs-1218-palesse-gs12/)

**2017** Инженеры НТЦК ОАО «Гомсельмаш» разработали, изготовили и отправили на испытания пять новых зерноуборочных комбайнов, не имеющих аналогов в модельном ряду предприятия: трёхбарабанный КЗС-2221К, роторный GS19R и два двухбарабанных комбайна гибридного типа GS13 и [GS3219](https://gomselmash.by/produktsiya/zernouborochnye-kombainy/kzs-3219kr-palesse-gs3219/), а также ПЕРВЫЙ В МИРЕ зерноуборочный комбайн GS4118К на газовом топливе.

## **1.2 Охрана труда и техника безопасности на рабочем месте**

Охрана труда – система обеспечения безопасности жизни и здоровья работающих в процессе трудовой деятельности, включающая правовые, социально-экономические, организационные, технические, психофизиологические, санитарно-гигиенические, лечебно-профилактические, реабилитационные и иные мероприятия и средства.

Руководством ОАО «Гомсельмаш» уделяется большое внимание улучшению эргономики рабочих мест, обеспечению гигиены и совершенствованию организации труда, регламентации режимов труда и отдыха.

Вся деятельность в области охраны труда на ОАО «Гомсельмаш» регламентирована действующим законодательством Республики Беларусь, санитарными нормами и правилами, гигиеническими нормативами, предписаниями надзорных органов.

Важным фактором создания безопасных условий труда является оптимизация организации рабочих мест. Рабочее место, хорошо приспособленное к трудовой деятельности инженера, правильно и целесообразно организованное в отношении пространства, формы, размера, обеспечивает ему удобное положение при работе и высокую производительность труда при наименьшем физическом и психическом напряжении.

При правильной организации рабочего места производительность труда инженера возрастает с 8 до 20 процентов.

Рабочее место для выполнения работ в положении сидя организуется в соответствии с ГОСТ 12.2.032-78 «Система стандартов безопасности труда. Рабочее место при выполнении работ сидя. Общие эргономические требования».

Требования к организации работы при использовании персонального компьютера и организационных средств определяются СанПиН 9-131 РБ 2000 «Гигиенические требования к видео-дисплейным терминалам, электронно-вычислительным машинам и организации работы».

Согласно ГОСТ 12.2.032-78, конструкция рабочего места и взаимное расположение всех его элементов должно соответствовать антропометрическим, физическим и психологическим требованиям. Большое значение имеет также характер работы. В частности, при организации рабочего места программиста должны быть соблюдены следующие основные условия:

– оптимальное размещение оборудования, входящего и состав рабочего места;

– достаточное рабочее пространство, позволяющее осуществлять все необходимые движения и перемещения;

– уровень акустического шума не должен превышать допустимого значения.

Рабочая поза сидя вызывает минимальное утомление программиста. Рациональная планировка рабочего места предусматривает четкий порядок и постоянство размещения предметов, средств труда и документации. То, что требуется для выполнения работ чаще, расположено в зоне легкой досягаемости рабочего пространства.

Помещения для работы программиста должны иметь естественное и искусственное освещение.

Искусственное освещение в помещениях эксплуатации ВДТ и ПЭВМ должно осуществляться системой общего равномерного освещения. В административно-общественных помещениях, в случаях преимущественной работы с документами, допускается применение системы комбинированного освещения (к общему освещению дополнительно устанавливаются светильники местного освещения, предназначенные для освещения зоны расположения документов).

Освещенность на поверхности стола в тоне размещения рабочего документа должна быть 300-500 лк. Местное освещение не должно создавать бликов на поверхности экрана и увеличивать освещенность экрана более 300 лк.

В качестве источников света при искусственном освещении должны применяться преимущественно люминесцентные лампы.

Конструкция рабочего стола должна обеспечивать оптимальное размещение на рабочей поверхности используемого оборудования с учетом его количества и конструктивных особенностей (размер ВДТ и ПЭВМ, клавиатуры и др.), характера выполняемой работы. При этом допускается использование рабочих столов различных конструкций, отвечающих современным требованиям эргономики.

Конструкция рабочего стула (кресла) должна обеспечивать поддержание рациональной рабочей позы при работе, позволять изменять позу с целью снижения статического напряжения мышц шейно-плечевой области и спины для предупреждения развития утомления.

Тип рабочего стула (кресла) должен выбираться в зависимости от характера и продолжительности работы.

Рабочий стул (кресло) должен быть подъемно-поворотным и регулируемым по высоте и углам наклона сиденья и спинки, а также расстоянию спинки от переднего края сиденья, при этом регулировка каждого параметра должна быть независимой, легко осуществляемой и иметь надежную фиксацию.

Экран видеомонитора должен находиться от глаз пользователя на оптимальном расстоянии 600-700 мм, но не ближе 500 мм с учетом размеров алфавитно-цифровых знаков и символов.

Высота рабочей поверхности стола должна регулироваться в пределах 680-800 мм, при отсутствии такой возможности высота рабочей поверхности стола должна составлять 725 мм.

Рабочий стол должен иметь пространство для ног высотой не менее 600 мм, шириной – не менее 500 мм, глубиной на уровне колен – не менее 450 мм и на уровне вытянутых ног – не менее 650 мм.

Клавиатуру следует располагать на поверхности стола на расстоянии не менее чем 300 мм от края, обращенного к пользователю или на специальной, регулируемой по высоте рабочей поверхности, отделенной от основной столешницы.

Помимо требований к организации рабочего места СанПиН 9-131 РБ 2000 устанавливает требования к микроклимату рабочей зоны: влажности, температуре, скорости потока воздуха и пр.

Техника безопасности – это система организационных мероприятий и технических средств, предотвращающих воздействие на работающих опасных производственных факторов.

## **1.3 Технологии, используемые на предприятии, программное**

## **обеспечение**

Для обеспечения полноценной совместной работы специалистов крупного машиностроительного предприятия в Windhill и Creo, требуется создание полноценной библиотеки “Стандартных изделий”, “Прочих изделий” и “Материалов”, используемых на предприятии. Требуется поддержание библиотек – добавление новых компонентов и удаление или запрет на использование старых компонентов.

Первоначально задача наполнения библиотек встает на этапе внедрения, поскольку невозможно продуктивно работать, не имея библиотечных элементов. Эта задача возникает и при внедрении новых модулей (например, гидравлика, электрика).

В Windchill имеются возможности по импорту и экспорту библиотек, но они не полностью обеспечивают требования заказчика. А если требуются связанные данные между CAD системой и Windchill – такие инструменты отсутствуют.

К тому же на каждом предприятии используется свой ограничительный перечень, поэтому заполняются не все объекты (например, целиком весь ГОСТ), а только те, которые нужны на предприятии. В связи с этим заполнение библиотек для каждого предприятия выполняется индивидуально.

На крупных предприятиях библиотеки очень объемные (по несколько тысяч объектов для каждого типа), для их создания требуется много ручного труда, если их заполнять вручную. Автоматизация этого процесса очень актуальна для всех предприятий.

Поскольку в Windchill отсутствуют полноценные инструменты, для автоматизации процесса наполнения библиотек, разрабатывается собственный программный код, специалистами компаний, выполняющих внедрение Windchill в организации.

В зависимости от типа, библиотеки могут содержать:

– Только составные части, не имеющие связи с CAD объектами. Например, материалы;

– Объекты Creo, связанные с составной частью. Например, крепежные элементы, подшипники и др;

– Информационные объекты Creo, связанные с составными частями. Это объекты, добавляемые в структуру сборки Creo, не имеющие Секционные доклады 192 конкретной формы - например, масло моторное, краски и др. жидкие материалы, а также гидравлические рукава высокого и низкого давления, провода и др;

В связи с необходимостью добавления в библиотеку большого колва объектов, автором разработаны программные средства для управления библиотекой “Стандартных изделий”, “Прочих изделий” и “Материалов”. Используя настраиваемый конфигурационный файл, программные средства могут быть настроены для работы с различными данными и для выполнения различных операций с объектами библиотеки – создание, удаление, контроль обозначения, наименования и имени файла, контроль атрибутов, назначение новых атрибутов, изменение расположения, изменение состояния.

Первоначальная цель стояла в добавлении в Windchill 2936 рукавов высокого давления, применяемых на предприятии.

При этом, каждый объект должен состоять из части и связанного с ним связью “Владелец” информационного объекта Creo. Каждый объект имеет свои атрибуты. Часть атрибутов должны контролироваться на уникальность, например: “Наименование”, “код САП”, “Номер”. Для каждого компонента должен быть назначен статус “Закупка” и указан поставщик. Все объекты по завершению должны быть переведены в состояние “Выпущено” и перемещены в определенную папку в библиотеке.

При этом по каждому объекту должно вестись подробное логирование, для оценки результата выполнения операций создания объектов и их связывания. Должны указываться причины ошибок, если они возникали при выполнении. Анализируя данную информацию можно откорректировать объекты, которые программа не смогла создать или исправить существующие объекты.

Использование программных средств позволило быстро и без ошибок добавить новые данные в базу Windchill.

# 2 ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

## **2.1 Технологии решения поставленной задачи**

Веб-приложение – это прикладное программное обеспечение, которое работает на веб-сервере, в отличие от компьютерных программ, которые запускаются локально в операционной системе устройства. Доступ к веб-приложениям осуществляется пользователем через веб-браузер с активным сетевым подключением. Эти приложения программируются с использованием клиент-серверной моделируемой структуры – пользователю или можно сказать клиенту предоставляются услуги через внешний сервер, размещенный третьей стороной.

В более ранних вычислительных моделях, таких как клиент-сервер, вычислительная нагрузка для приложения распределялась между кодом на сервере и кодом, установленным на каждом клиенте локально. Другими словами, приложение имело свою собственную предварительно скомпилированную клиентскую программу, которая служила его пользовательским интерфейсом и должна была быть отдельно установлена на персональном компьютере каждого пользователя. Обновление серверного кода приложения, как правило, также требует обновления клиентского кода, установленного на каждой рабочей станции пользователя, что увеличивает затраты на поддержку и снижает производительность. Кроме того, как клиентские, так и серверные компоненты приложения обычно были тесно связаны с определенной компьютерной архитектурой и операционной системой, и перенос их на другие часто был непомерно дорогим для всех, кроме самых крупных приложений. В настоящее время, нативные приложения для мобильных устройств также связаны некоторыми или всеми вышеперечисленными проблемами.

В отличие от этого, веб-приложения используют веб-документы, написанные в стандартном формате, таком как *HTML* и *JavaScript*, которые поддерживаются различными веб-браузерами. Веб-приложения можно рассматривать как специфический вариант клиент-серверного программного обеспечения, где клиентское программное обеспечение загружается на клиентскую машину при посещении соответствующий веб-страницы с использованием стандартных процедур, таких как *HTTP*. Обновления клиентского веб-программного обеспечения могут происходить каждый раз при посещении веб-страницы. Во время сеанса веб-браузер интерпретирует и отображает страницы и действует как универсальный клиент для любого веб-приложения.

С помощью *Java*, *JavaScript*, *CSS*, *Flash*, *Silverlight* и других технологий возможны такие специфические для приложений методы, как рисование на экране, воспроизведение звука и доступ к клавиатуре и мыши. Многие службы работали над тем, чтобы объединить все это в более привычный интерфейс, который принимает вид операционной системы. Универсальные методы, такие как перетаскивание также поддерживаются эти технологии. Веб-разработчики часто используют сценарии на стороне клиента для добавления функциональности, особенно для создания интерактивного интерфейса, который не требует перезагрузки страницы. В последнее время были разработаны технологии для координации клиентских сценариев с серверными технологиями, такими как *ASP.NET*, *J2EE*, *Perl/Plack* и *PHP*. *Ajax*, метод веб-разработки, использующий комбинацию различных технологий, является примером технологии, которая создает более интерактивный опыт.

Написание веб-приложений часто упрощается за счет использования фреймворка веб-приложений. Эти фреймворки облегчают быструю разработку приложений, позволяя команде разработчиков сосредоточиться на тех частях своего приложения, которые являются уникальными для их целей, без необходимости решать общие проблемы разработки, такие как управление пользователями.

Использование фреймворков веб-приложений часто может уменьшить количество ошибок в программе, как за счет упрощения кода, так и за счет того, что одна команда может сосредоточиться на фреймворке, а другая, на конкретном случае использования. В приложениях, которые подвергаются постоянным попыткам взлома в интернете, проблемы безопасности могут быть вызваны ошибками в программе.

*ASP.NET MVC* является многофункциональной платформой для создания веб-приложений и *API*-интерфейсов с помощью структуры проектирования *Model-View-Controller*.

*ASP.NET Core MVC* представляет собой упрощенную, эффективно тестируемую платформу с открытым исходным кодом, оптимизированную для использования с *ASP.NET Core*.

*ASP.NET Core MVC* предоставляет основанный на шаблонах способ создания динамических веб-сайтов с четким разделением задач. Она обеспечивает полный контроль разметки, поддерживает согласованную с *TDD* разработку и использует новейшие веб-стандарты.

Важно различать архитектурный паттерн *MVC* и *ASP.NET MVC Framework*. *MVC* паттерн не является новым, но он завоевал огромную популярность сегодня в качестве паттерна для веб-приложений по следующим причинам:

– 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*.

*ASP.NET MVC Framework* реализует *MVC* паттерн и, тем самым, обеспечивает значительно улучшенное разделение концепций. На самом деле *ASP*.*NET* *MVC* реализует современный вариант *MVC* паттерна, который особенно хорошо подходит для веб-приложений.

## **2.2 Цель задачи**

Необходимо реализовать *web*-приложение на тему «Цепочка производства велосипедов»:

Основные страницы приложения:

– «Типы поставщиков»

– «Поставщики»;

– «Детали»;

– «Велосипеды»;

– «Детали велосипедов»;

– «Заказы на детали».

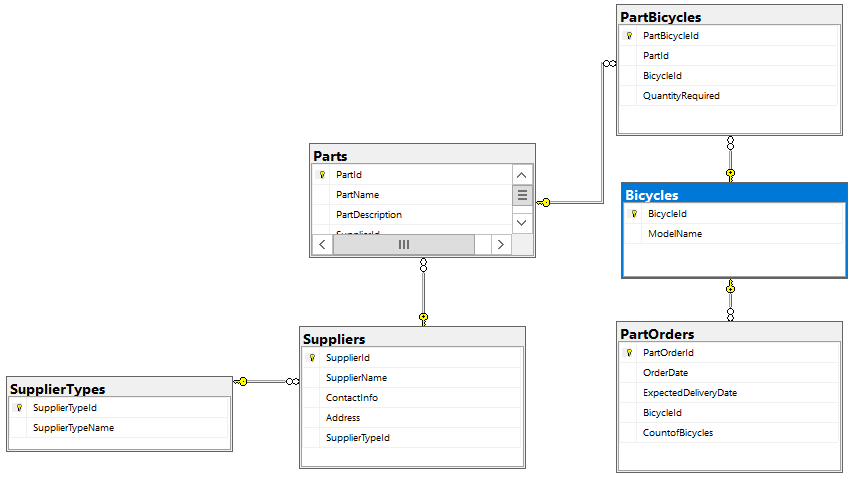


Рисунок 2.1 – Диаграмма базы данных

## **2.3 Сущности базы данных**

В результате изучения исходных данных выделены следующие сущности:

– «Тип поставщика»

– «Поставщик»;

– «Деталь»;

– «Велосипед»;

– «Деталь велосипеда»;

– «Заказ на детали».

В сущности «Тип поставщика» можно выявить следующий атрибуты: «SupplierTypeId», «SupplierTypeName налога». Подровбное описание атрибутов приведено в таблице 2.1.

Таблица 2.1 – Описание атрибутов таблицы «Типы поставщиков»

|  |  |  |
| --- | --- | --- |
| Атрибуты | Описание домена | Тип данных |
| 1 | 2 | 3 |
| SupplierTypeId | Уникальный инкрементируемый идентификатор. Является первичным ключом | Целочисленный |
| SupplierTypeName | Название типа поставщика | Строка |

В сущности «Поставщик» можно выявить следующие атрибуты: «SupplierId», «SupplierName», «ContactInfo», «Address», «SupplierTypeId». Описание атрибутов более подробно приведено в таблице 2.2.

Таблица 2.2 – Описание атрибутов таблицы «Поставщики»

|  |  |  |
| --- | --- | --- |
| Атрибуты | Описание домена | Тип данных |
| 1 | 2 | 3 |
| SupplierId | Уникальный инкрементируемый идентификатор. Является первичным ключом | Целочисленный |
| SupplierName | Название поставщика | Строка |
| ContactInfo | Номер телефона поставщика | Строка |
| Address | Адресс поставщика | Строка |
| SupplierTypeId | Внешний ключ к таблице типы «Типы поставщиков» | Целочисленный |

В сущности «Деталь» можно выявить следующие атрибуты: «PartId», «PartName», «PartDescription», «SupplierId». Описание атрибутов более подробно приведено в таблице 2.3.

Таблица 2.3 – Описание атрибутов таблицы «Детали»

|  |  |  |
| --- | --- | --- |
| Атрибуты | Описание домена | Тип данных |
| 1 | 2 | 3 |
| SupplierId | Уникальный инкрементируемый идентификатор. Является первичным ключом | Целочисленный |
| PartName | Название | Строка |
| PartDescription | Описание детали | Строка |
| SupplierId | Внешний ключ к таблице типы «Поставщики» | Целочисленный |

В сущности «Велосипед» можно выявить следующие атрибуты: «BicycleId», «ModelName». Описание атрибутов более подробно приведено в таблице 2.4.

Таблица 2.4 – Описание атрибутов таблицы «Велосипед»

|  |  |  |
| --- | --- | --- |
| Атрибуты | Описание домена | Тип данных |
| 1 | 2 | 3 |
| BicycleId | Уникальный инкрементируемый идентификатор. Является первичным ключом | Целочисленный |
| ModelName | Название модели велосипеда | Строка |

В сущности «Деталь велосипеда» можно выявить следующие атрибуты: «PartBicycleId», «PartId», «BicycleId», «QuantityRequired». Описание атрибутов более подробно приведено в таблице 2.5.

Таблица 2.5 – Описание атрибутов таблицы «Деталь велосипеда»

|  |  |  |
| --- | --- | --- |
| Атрибуты | Описание домена | Тип данных |
| 1 | 2 | 3 |
| PartBicycleId | Уникальный инкрементируемый идентификатор. Является первичным ключом | Целочисленный |
| Продолжение таблицы 2.5 | | |
| 1 | 2 | 3 |
| PartId | Внешний ключ к таблице типы «Детали» | Целочисленный |
| BicycleId | Внешний ключ к таблице типы «Велосипеды» | Целочисленный |
| QuantityRequired | Количество заказываемых деталей | Целочисленный |

В сущности «Заказ на детали» можно выявить следующие атрибуты: «PartOrderId», «OrderDate», «ExpectedDeliveryDate», «BicycleId», «CountofBicycles». Описание атрибутов более подробно приведено в таблице 2.6.

Таблица 2.6 – Описание атрибутов таблицы «Заказы на детали»

|  |  |  |
| --- | --- | --- |
| Атрибуты | Описание домена | Тип данных |
| 1 | 2 | 3 |
| PartOrderId | Уникальный инкрементируемый идентификатор. Является первичным ключом | Целочисленный |
| OrderDate | Дата заказа деталей | Дата |
| ExpectedDeliveryDate | Дата, к которой детали должны быть доставлены | Дата |
| BicycleId | Внешний ключ к таблице типы «Велосипеды» | Целочисленный |
| CountofBicycles | Кол-во велосипедов, которое должно быть произведено из этих деталей | Целочисленный |

## **2.4 Структура веб-приложения**

В состав данного веб-приложения входят три основных компонента: модель, представление и контроллер.

Модель представляет состояние приложения и бизнес-логику, непосредственно связанную с данными. Как правило, объекты моделей хранятся в базе данных. В архитектуре *MVC* модели представлены двумя основными типами: модели представлений, которые используются представлениями для отображения данных на веб*-*странице, и модели домена, описывающие логику управления данными. Модель содержит данные и хранит логику обработки этих данных, но не содержит логику взаимодействия с пользователем, т.е. с представлением.

Представление является графическим веб*-*интерфейсом, через который пользователь может взаимодействовать с приложением напрямую. Данный компонент содержит минимальную логику, которая связана с представлением данных.

Контроллер представляет центральный компонент архитектуры *MVC* для управления взаимодействием с пользователем, работы с моделью и выбора представления для отображения. Контроллер обеспечивает связь между пользователем и приложением, представлением и хранилищем данных. Он содержит логику обработки введённых пользователем данных и логику формирования ответа пользователю. Контроллер является начальной отправной точкой в приложении и отвечает за выбор рабочих типов моделей и отображаемых представлений.

## **2.5 Классы для доступа к данным**

Для работы с таблицами базы данных в приложении необходимы классы, которые описывают каждую таблицу. В данных классах описываются поля таблиц в виде свойств и связи между таблицами в виде связей между классами. Классы, описывающие сущности, необходимо разработать самостоятельно.

Классы *SupplierType, Supplier, PartOrder, PartBicycle, Part, Bicycle* описывают соответствующие таблицы.

Свойства в каждом классе описывают столбцы соответствующей таблицы. В классах, описывающих таблицы, которые находятся на стороне отношения «многие», содержат ссылку на объект класса, моделирующего таблицу, связанную внешним ключом.

Также для операций с данными существуют отдельные классы, они хранятся в папке *Repositories.* Описание классов, содержащихся в данной папке приведено в таблице 2.7.

Таблица 2.7 – Описание классов в папке *Repositories.*

|  |  |
| --- | --- |
| Наименование класса | Описание класса |
| *BicycleRepository* | Класс, для работы с велосипедами |
| *PartBicycleRepository* | Класс, для работы с деталями велосипедов |
| *IRepository* | Интерфейс, который содержит общую логику репозиториев |
| *PartOrderRepository* | Класс, для с заказами на детали |
| *PartRepository* | Класс, для работы с деталями |
| *SupplierRepository* | Класс, для работы с поставщиками |
| *SupplierTypeRepository* | Класс, для работы с типами поставщиков |

## **2.6 Контроллеры**

Контроллер представляет обычный класс, который наследуется от абстрактного базового класса *Microsoft.AspNetCore.Mvc.Controller*. Именование контроллеров строго предопределено, т.е. имя контроллера обязательно должно иметь суффикс «*Controller*», а остальная часть считается названием контроллера.

Адрес, который обрабатывается контроллерами, представлен в виде паттерна *{controller=[ControllerName]}/{action=[MethodName]}*, где [*ControllerName*] – название контроллера, [*MethodName*] – название метода контроллера.

Для работы с созданными моделями разработаны следующие контроллеры:

– *BicyclesController*– отвечает за работу с таблицей *Bicycles*;

*– PartBicyclesController–* отвечает за работу с таблицей *PartBicycles;*

*– PartOrdersController–* отвечает за работу с таблицей *PartOrders;*

*– PartsController –* отвечает за работу с таблицей *Parts;*

*– SuppliersController–* отвечает за работу с таблицей *Suppliers;*

*– SuppliersTypesController–* отвечает за работу с таблицей *SuppliersTypes;*

Код данных контроллеров приведён в приложении В.

Контроллеры, отвечающие за работу с таблицами, имеют следующие методы:

– *Index[GET];*

*– Create[GET];*

*– Create[POST];*

*– Edit[GET];*

*– Edit[POST];*

*– Delete[GET];*

*– Delete[POST].*

Метод *Index[Get]* в качестве входных параметров поступает список какой либо из сущнойстей

Метод *Create[GET]* возвращает одноимённое представление с полями для добавления записи в таблицу базы данных. Для таблиц, стоящих на стороне «многие» данный метод формирует списки, в которые добавляются необходимые данные из таблиц, стоящих на стороне отношения «один».

Метод *Create[POST]* вызывается при отправке результата формы создания записи. Данный метод принимает объект, таблицу которого он моделирует и содержит данные, которые необходимо записать в базу данных. Перед записью производится валидация данных. Если данные неверны, то запись в базу данных не происходит. Если данные верны, то происходит запись данных в базу и переход в метод *Index* текущего контроллера.

Метод *Edit[GET]* принимает идентификатор записи и производит выборку нужной записи из определённой таблицы базы данных. Если запись найдена, то происходит добавление необходимых данных из других таблиц в списки и возврат представления с формой редактирования записи. Если запись не найдена, то метод возвращает стандартное сообщение об ошибке.

Метод *Edit[POST]* вызывается при отправке результата формы редактирования записи. Данный метод в качестве входных параметров принимает идентификатор записи и объект, содержащий данные об этой записи. Если входной идентификатор и идентификатор объекта не совпадают, то метод возвращает стандартное сообщение об ошибке. Иначе метод выполняет валидацию входных данных и если данные верны, то производится обновление данных в базе. Если операция обновления прошла успешно, то происходит переход в метод *Index* текущего контроллера. В случае возникновения ошибки метод возвращает стандартное сообщение об ошибке.

Метод *Delete[GET]* принимает идентификатор записи и производит выборку нужной записи из определённой таблицы базы данных. Если запись найдена, то метод возвращает представление с формой подтверждения удаления записи. Если запись не удалось найти, то метод возвращает стандартное сообщение об ошибке.

Метод *Delete[POST]* вызывается при подтверждении удаления данных и производит удаление заданной записи из базы данных. После произведения операции удаления происходит переход в метод *Index* текущего контроллера.

## **2.7 Представления**

Представления – это файлы в формате *cshtml*, в которых используется язык разметки *HTML* и язык программирования *C#* в разметке *Razor*. Все представления объединяются в папки с именами, соответствующими названиям контроллеров. Все эти папки находятся в папке *Views* в корне приложения.

Для существующих контроллеров разработаны представления, которые содержатся следующих в папках:

– *Bicycles –* содержит представления для работы с таблицей «Велосипеды»;

– *PartBicycles* – содержит представления для работы с таблицей «Детали велосипедов»;

– *PartOrders –* содержит представления для работы с таблицей «Заказы на детали»;

– *Parts –* содержит представления для работы с таблицей «Детали»;

– *Supliers –* содержит представления для работы с таблицей «Поставщики»;

– *SuplierTypes –* содержит представления для работы с таблицей «Типов поставщиков».

Для каждого справочника имеется 4 представления.

Все эти представления будет приведены на примере сущности «Поставщик»

Представление для отображения информации приведено на рисунке 2.2

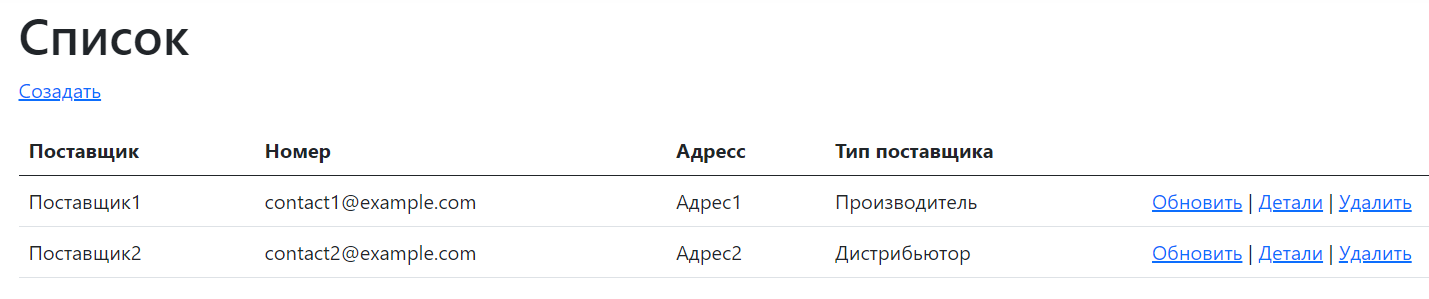


Рисунок 2.2 – Отображение информации

Для добавления информации необходимо нажать кнопку «Создать», после чего ввести соответствующие данные и нажать на кнопку «Создать». Если же нет необходимости добавлять информацию, то необходимо нажать на кнопку «Вернуться».

Представление добавления информации приведено на рисунке 2.4

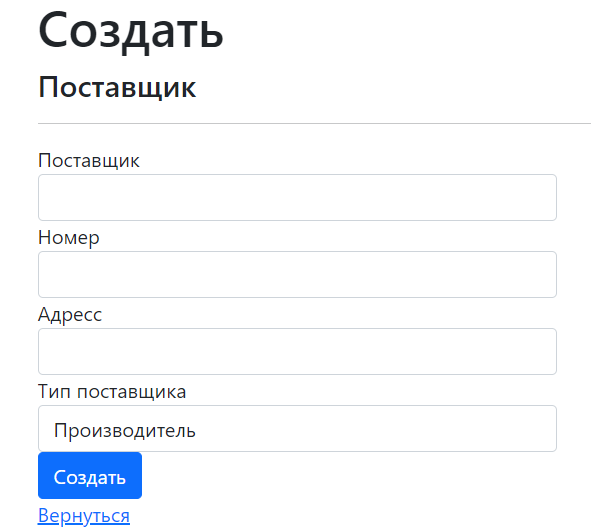


Рисунок 2.3 – Добавление информации

Для изменения информации необходимо нажать на кнопку «Обновить» и ввести соответствующие данные, после чего нажать на кнопку «Сохранить». Если же нет необходимости изменять информацию, то необходимо нажать на кнопку «Вернуться».

Представление изменения информации приведено на рисунке 2.5

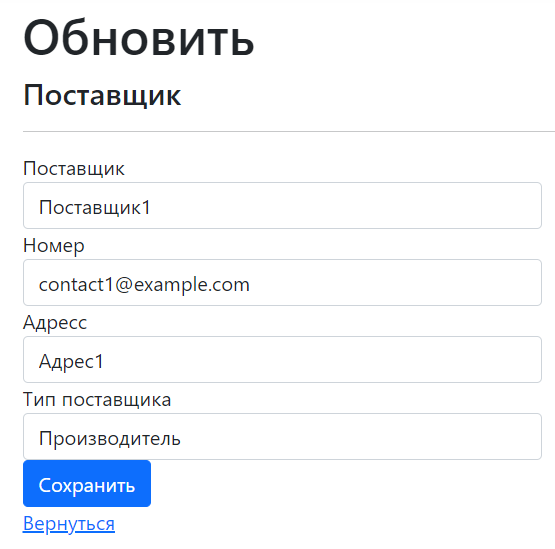


Рисунок 2.4 – Изменение информации

Для удаления информации необходимо нажать кнопку «Удалить». После чего откроется представление для удаления информации. Если есть необходимость удалить информацию, то в открывшемся представлении необходимо нажать кнопку «Удалить», если же нет необходимости, то нажать кнопку «Назад».

Представления для удаления информации приведено на рисунке 2.6

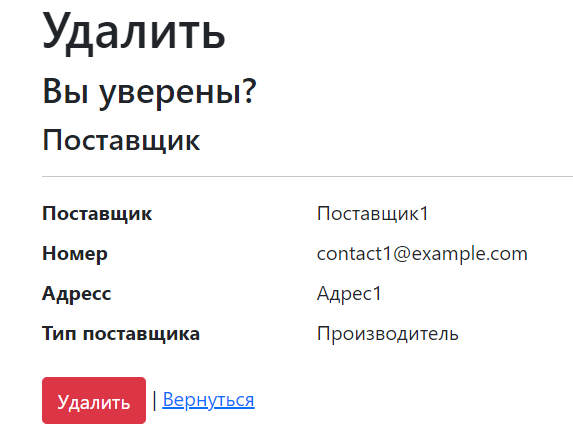


Рисунок 2.5 – Удаление информации

Все остальные представления работают так же, как и представления для справочника «Вид деятельности».