ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ»
(Н И У « Б е л Г У »)

ИНСТИТУТ ИНЖЕНЕРНЫХ ТЕХНОЛОГИЙ И ЕСТЕСТВЕННЫХ НАУК

КАФЕДРА МАТЕМАТИЧЕСКОГО И ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ

АВТОМАТИЗИРОВАННАЯ СИСТЕМА УЧЁТА ТРУДОВЫХ РЕСУРСОВ И ЗАРАБОТНОЙ ПЛАТЫ ДЛЯ ИД ООО «МОНТАЖАВТОМАТИКА»

Выпускная квалификационная работа
обучающегося по направлению подготовки 02.03.02
Фундаментальная информатика и информационные технологии
очной формы обучения, группы 07001401
Жолумского Кирилла Викторовича

Научный руководитель
к.т.н., доцент
Муромцев В.В.

БЕЛГОРОД 2018

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Наличие средств автоматизации производственных процессов является обязательным критерием успешного функционирования предприятия в условиях рыночной экономики. Внедрение системы автоматизированного управления производственной деятельностью приводит к значительному росту конкурентоспособности предприятия, увеличивает его рентабельность. Таким образом, в настоящее время проблематика повышения уровня автоматизации производства представляется особенно актуальной.

В общем виде автоматизированная система управления предприятием представляет собой интегрированный пакет прикладного программного обеспечения, позволяющий осуществлять централизованное управление производством, трудовыми и материальными ресурсами, финансовыми потоками и активами. Специфика процессов, подлежащих автоматизации, определяется сферой деятельности конкретного предприятия. Следует также подчеркнуть, что эффективная автоматизация производства подразумевает оптимизацию отдельных этапов производственного цикла. Данный критерий, в свою очередь, оказывает непосредственное влияние на снижение издержек производства, увеличение общей результативности труда и повышение эффективности управления.

Полная или частичная автоматизация производственной деятельности предприятия приводит к значительному сокращению временных затрат, необходимых для выполнения производственных задач, что позволяет увеличить количество произведённой продукции. Преимуществом высокого уровня автоматизации производственных процессов также следует назвать общее снижение количества материальных и финансовых затрат на всех этапах производства, которое непосредственно связано с сокращением времени, необходимого персоналу организации для качественного выполнения профессиональных обязанностей.

Автоматическое функционирование производства снижает потребность предприятия в работниках высокой квалификации, что приводит к существенному уменьшению зависимости производственной деятельности от трудовых ресурсов предприятия, и таким образом повышает устойчивость процесса производства в целом.

Комплексная информационная система автоматизации производства предоставляет широкие возможности для долгосрочного хранения данных определённого характера, описывающих качественные и количественные показатели производственной деятельности предприятия. В перспективе накопленная информация может послужить материалом для проведения различных статистических исследований, позволяющих проанализировать эффективность функционирования производственных, управленческих и других структурных подразделений организации. Полученная статистическая информация также может использоваться для организации процесса планирования потребностей производства, что позволит существенно снизить материальные издержки путём более рационального распределения финансовых затрат.

Целью выпускной квалификационной работы является повышение уровня автоматизации процессов управления трудовыми ресурсами ИД ООО «Монтажавтоматика». Автоматизированная система учёта трудовых ресурсов и заработной платы представляет собой совокупность средств, позволяющих планировать потребности предприятия в трудовых ресурсах, проводить анализ кадрового потенциала и рынка труда, давать оценку эффективности труда, производить начисление заработной платы, управлять наймом и учётом персонала. Оплата трудовой деятельности составляет значительную часть общих финансовых расходов предприятия. Таким образом, проблема эффективного использования трудовых ресурсов является особенно актуальной.

Первая глава данной работы содержит общее описание программных модулей, входящих в состав информационной системы предприятия, а также

комплексный анализ подсистемы учёта трудовых ресурсов и заработной платы, позволяющий определить ключевые направления автоматизации. Вторая глава посвящена вопросам проектирования архитектуры приложения и описанию средств разработки. В третьей главе представлены процесс реализации основных структурных компонентов программного обеспечения и результат тестовых испытаний разработанного приложения.

Данная выпускная квалификационная работа содержит 49 страниц, 25 рисунков и 1 приложение. При написании работы было использовано 20 литературных источников.

1. АНАЛИЗ ИНФОРМАЦИОННОЙ СИСТЕМЫ ИД ООО «МОНТАЖАВТОМАТИКА»

1.1 Обзор текущего состояния информационной системы предприятия

Информационная система предприятия ИД ООО «Монтажавтоматика» представляет собой интегрированный пакет автономных программных модулей, позволяющих выполнять широкий спектр задач по управлению различными элементами производственной деятельности. Модульная схема автоматизированной системы управления производством, представленная на рисунке 1.1 и рисунке 1.2.

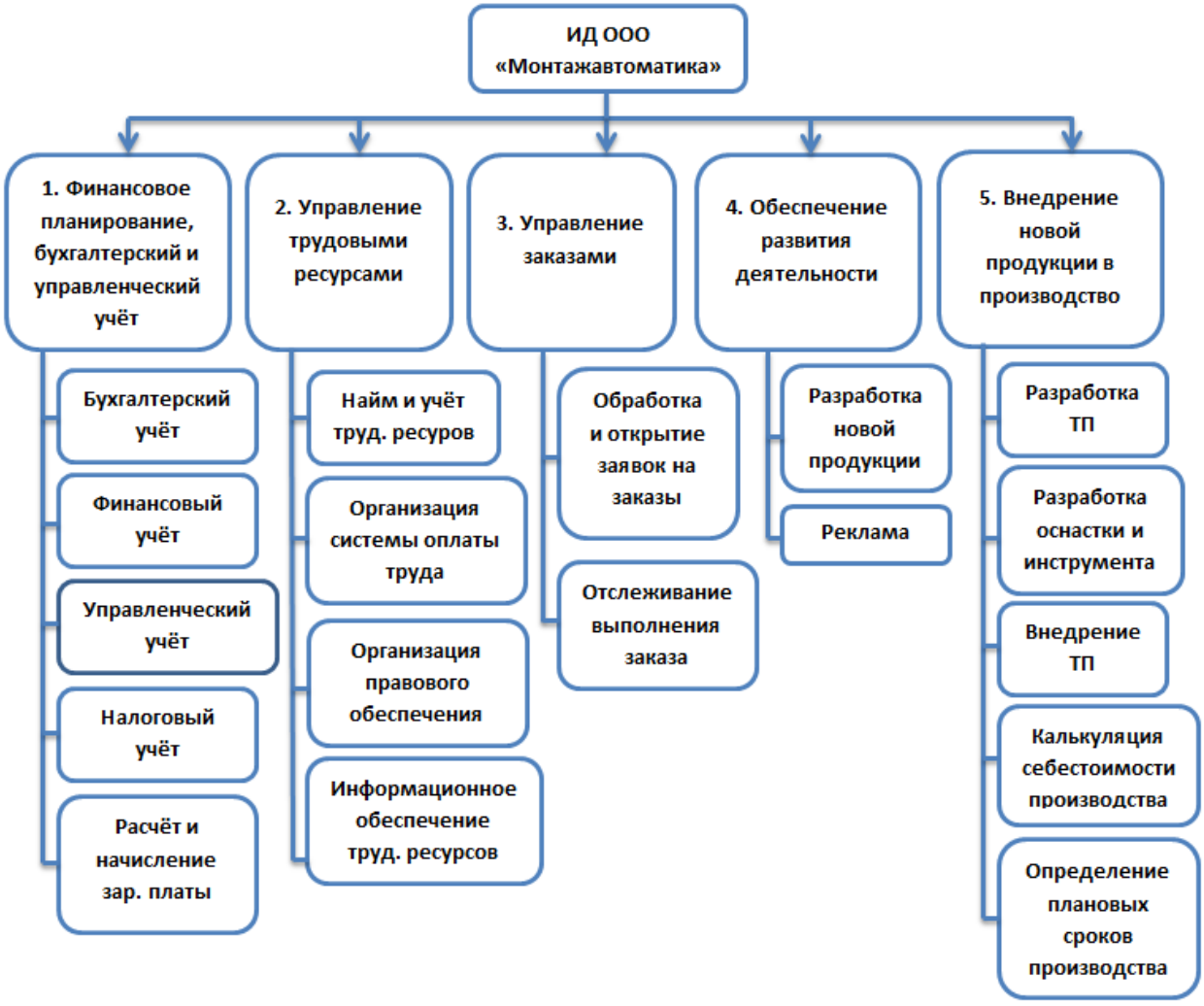


Рис. 1.1. Структурная схема автоматизированной системы управления производством (1-я часть)

На рисунке 1.1 представлено описание первых пяти компонентов информационной системы предприятия: модуль финансового планирования, бухгалтерского и управленческого учёта; модуль управления трудовыми ресурсами; модуль управления заказами; модуль обеспечения развития деятельности; модуль внедрения новой продукции в производство.

Модуль финансового планирования, бухгалтерского и управленческого учёта содержит подсистемы, отвечающие за реализацию финансового, налогового, управленческого и бухгалтерского учёта, а также подсистему расчёта и начисления заработной платы. Данный программный компонент предоставляет управленческому аппарату предприятия необходимую информацию для осуществления планирования, управления и контроля над производственной деятельностью и является своего рода центральным ядром, концентрирующим учётную информацию предприятия. Процесс ведения учётной деятельности включает сбор, анализ, форматирование, передачу и приём информации.

Модуль управления трудовыми ресурсами включает подсистемы найма и учёта трудовых ресурсов, организации оплаты труда, а также правового и информационного обеспечения трудовых ресурсов. Назначением данного структурного компонента является реализация механизмов управленческой деятельности, направленной на повышение производительности труда и эффективности производства. Организация подсистемы учёта трудовых ресурсов напрямую затрагивает решение таких задач, как оперативно-производственное планирование, экономический анализ, учёт и контроль производственной деятельности. [4]

Модуль управления заказами включает подсистему обработки и открытия заявок на заказы, а также подсистему отслеживания степени их выполнения. Данный программный компонент выполняет автоматизацию процесса обработки заказов и обеспечивает оптимизацию отдельных компонент управления заказами. Анализ информации о заказах на тот или иной вид продукции позволяет оценить уровень потребительского спроса,

осуществить планирование поставок необходимых материалов, провести прогнозирование потребности производства в трудовых ресурсах.

Модуль по обеспечению развития производственной деятельности предприятия отвечает за управление процессом разработки новой продукции, и за осуществление маркетинговой деятельности. Подсистема, включающая организацию разработки новой продукции, предназначена для решения различных аналитических задач, связанных с определением стратегии производства и сбыта определённых видов продукции. Маркетинговые службы включают анализ коммерческих показателей внедряемых инноваций, которые выражаются оценкой рентабельности продукции, краткосрочным и среднесрочным прогнозированием объёмов продаж, оценкой финансовых, производственных возможностей предприятия и другими статистическими показателями. [5]

Модуль внедрения новой продукции в производство, в свою очередь, содержит подсистемы, отвечающие за разработку технического плана, разработку оснастки и инструмента, внедрение технического плана, калькуляцию себестоимости производства, определение плановых сроков производства. Назначением данного программного компонента является осуществление контроля над этапами разработки и внедрения в производство новых видов продукции.

Рисунок 1.2 содержит описание второй части автоматизированной системы управления производством ИД ООО «Монтажавтоматика». Вторая часть структурной схемы включает такие разделы производственной деятельности, как управление основным производством (оперативное планирование и контроль выполнения оперативного плана), управление вспомогательным производством (изготовление комплектующих изделий), обеспечение сбыта готовой продукции и снабжение производства необходимыми материальными ресурсами, информационное, техническое и юридическое обеспечение производственного процесса, управление административной и хозяйственной деятельностью.

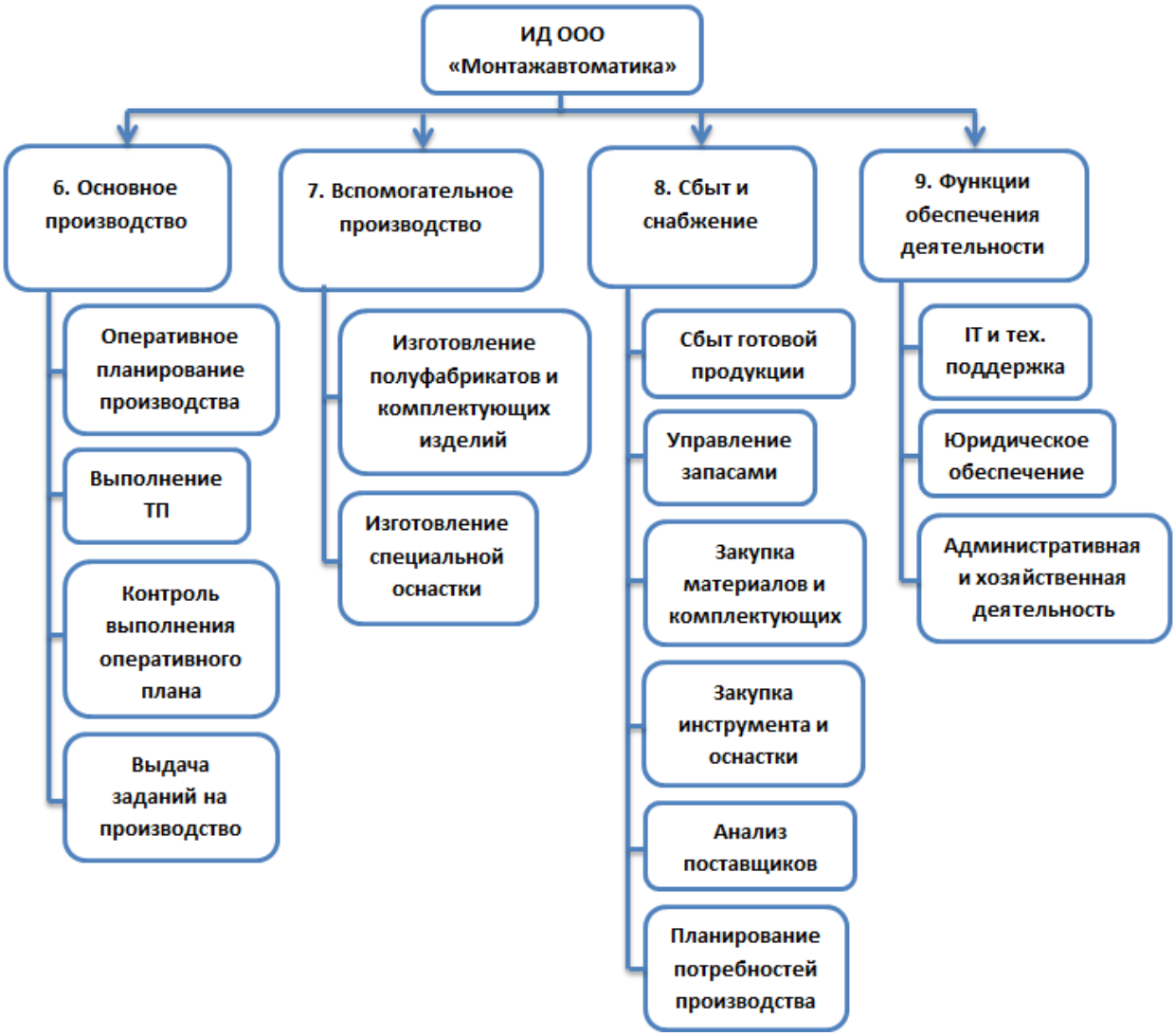


Рис. 1.2. Структурная схема автоматизированной системы управления производством (2-я часть)

Раздел, реализующий функции управления основным производством, содержит системы оперативного планирования, выполнения технического плана, контроля выполнения оперативного плана, выдачи заданий на производство. Назначением модуля управления основным производством являются оперативное планирование производственной деятельности, учёт и анализ хода производственного процесса, обеспечение оперативного регулирования в случае возникновения внештатных ситуаций. [2]

Модуль управления вспомогательным производством реализует подсистемы, отвечающие за изготовление полуфабрикатов и комплектующих

изделий и изготовление специальной оснастки. К функциям вспомогательной производственной деятельности относятся оснащение инструментальными средствами, энергетическими, складскими и транспортно-логистическими ресурсами, обеспечивающими нормальное выполнение производственного цикла.

Модуль сбыта и снабжения охватывает процессы сбыта готовой продукции, управления запасами, закупки материалов и комплектующих, закупки инструмента и оснастки, анализа поставщиков и планирования потребностей производства. Подсистема сбыта и снабжения отвечает за удовлетворение потребностей производства в материальных ресурсах, а также за обеспечение процесса реализации произведённой продукции. Функциональное обеспечение деятельности включает в себя техническую поддержку, юридическое обеспечение и организацию административной и хозяйственной деятельности. Основными задачами технического отдела являются обеспечение бесперебойного функционирования оборудования, проведение ремонтных работ и осуществление технической модернизации производственных линий. Цель юридического обеспечения – осуществление контроля над соблюдением трудового законодательства и решение правовых вопросов. Подсистема IT выполняет функцию обеспечения сотрудников предприятия информационными сервисами, которые необходимы для выполнения ими служебных обязанностей.

1.2 Анализ модуля управления трудовыми ресурсами

Представленный в предыдущем разделе обзор текущего состояния информационной системы предприятия ИД ООО «Монтажавтоматика» наглядно продемонстрировал высокий уровень автоматизации основных элементов производства. Проведённое исследование позволило составить общее представление о назначении отдельных структурных подразделений

предприятия и выявить закономерности в их функционировании. Однако в соответствии с целями, сформулированными во введении, основной задачей данной работы является увеличение эффективности управления трудовыми ресурсами путём автоматизации отдельных аспектов управленческой деятельности. Таким образом, необходимо провести детальный анализ модуля по учёту трудовых ресурсов и выявить в его реализации фундаментальные упущения, возмещение которых позволит добиться повышения эффективности управления.

Система управления трудовыми ресурсами содержит следующие программные компоненты: найм и учёт трудовых ресурсов, организация оплаты труда, правовое и информационное обеспечение трудовых ресурсов предприятия, которые представлены на рисунке 1.3.

Рис. 1.3. Модуль управления трудовыми ресурсами предприятия

Подсистема найма и учёта трудовых ресурсов предназначена для управления информацией о найме, приёме, перемещениях и увольнениях персонала, а также для организации рационального использования трудовых ресурсов и ведения учётной документации. Организация системы оплаты труда выполняет функцию объединения учётных данных, формирующих сводную информацию о величине заработной платы каждого работника, осуществляющего тот или иной вид трудовой деятельности. Правовое обеспечение играет роль регулятора при решении правовых вопросов

производственной и хозяйственной деятельности, а также предоставляет возможности консультирования по различным юридическим вопросам. [3] Подсистема информационного обеспечения предоставляет технические средства для ведения учётной деятельности, а также выполняет интеграцию модуля управления трудовыми ресурсами с другими программными модулями информационной системы.

Исследование схемы, представленной на рисунке 1.3, позволяет утверждать, что средства управления трудовыми ресурсами, реализованные в информационной системе предприятия, автоматизируют ведение только тех аспектов учётной деятельности, которые связаны с обеспечением контроля над ходом кадровых процессов. При этом фиксируется отсутствие какой-либо аналитической работы с накопленной информацией, изучение которой могло бы послужить повышению эффективности управления посредством организации процесса планирования трудовых ресурсов.

В условиях рыночной экономики планирование и анализ показателей по труду приобретают серьёзное значение, поскольку таким образом можно более полно выявить пути минимизации издержек и задействовать имеющиеся резервы с целью повышения эффективности производства. Следует также подчеркнуть, что в данном случае целесообразным является планирование именно количественных потребностей, поскольку оценка необходимого уровня квалификации сотрудников и анализ требований к их профессиональным качествам не могут быть проведены посредством исследования имеющихся статистических показателей.

Итак, в процессе анализа модуля управления трудовыми ресурсами была выявлена возможность качественного усовершенствования данного программного компонента за счёт включения подсистемы планирования трудовых ресурсов.

Окончательный вариант информационной системы предприятия представлен на рисунке 1.4 (внесённые изменения выделены красным цветом).

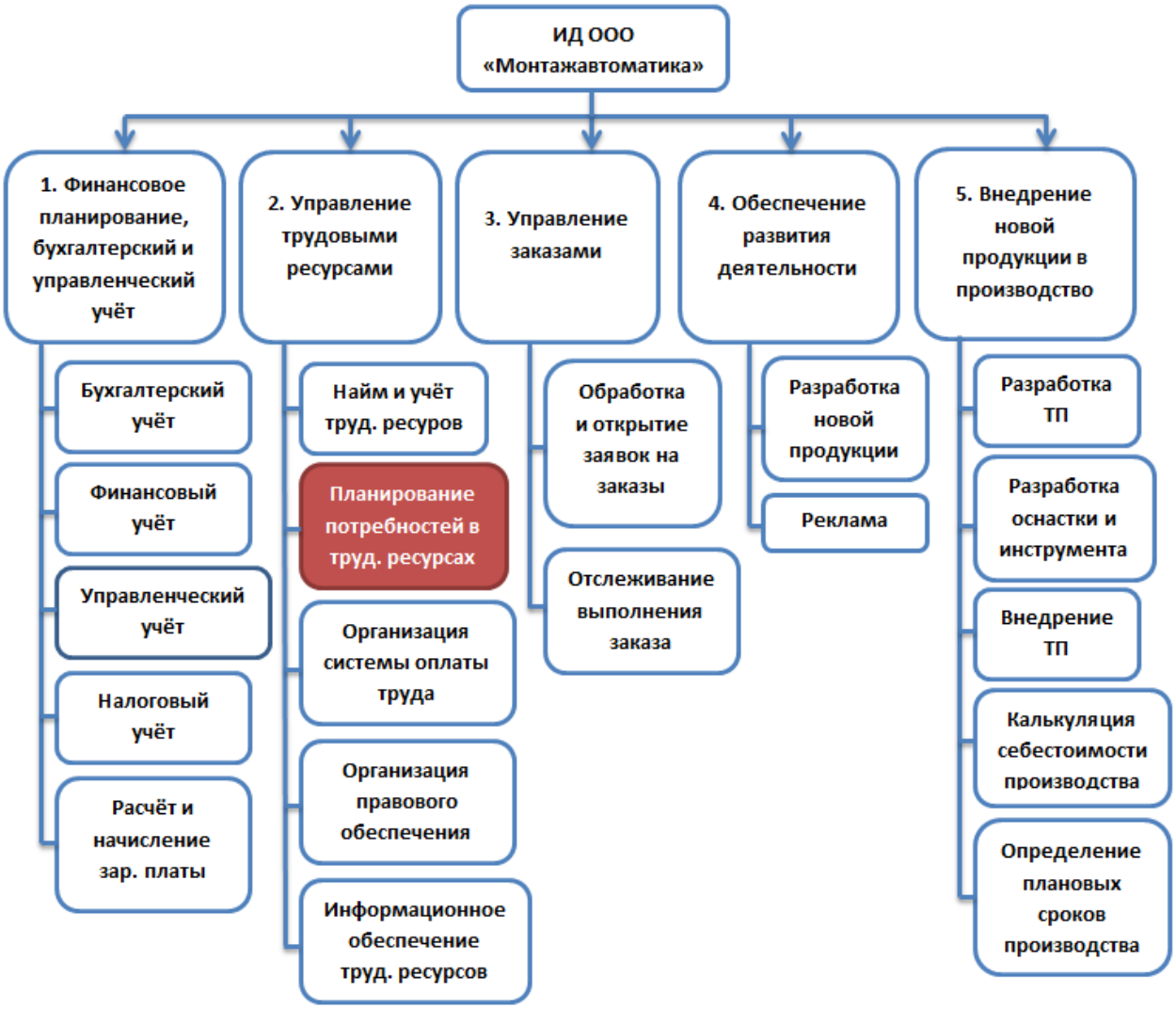


Рис. 1.4. Информационная система предприятия после изменений

1.3 Постановка и анализ задачи планирования трудовых ресурсов

Опираясь на результаты проведённых исследований, в соответствии с целями, сформулированными во введении, можно выделить ряд шагов, направленных на решение задачи планирования трудовых ресурсов:

- обзор текущего состояния информационной системы ИД ООО «Монтажавтоматика»;
- анализ модуля управления трудовыми ресурсами предприятия;

- постановка и анализ задачи планирования трудовых ресурсов;
- функциональный анализ приложения;
- проектирование архитектуры приложения;
- описание средств программирования;
- реализация уровня доступа к данным;
- реализация уровня бизнес-логики;
- реализация уровня представления;
- тестирование и анализ результатов.

Процесс планирования трудовых ресурсов состоит из двух этапов: оценка трудовых ресурсов, имеющихся в наличии, и прогнозирование будущих потребностей производства. [9] Оценка численности необходимой рабочей силы производится посредством прогнозирования соответствующих объёмов производства и показателей производительности труда. Прогнозная информация может быть получена с применением различных подходов к прогнозированию, которые отличаются друг от друга принципами функционирования. Наиболее распространёнными среди них являются методы экспертных оценок, методы экстраполяции тенденций, метод компонент, различные статистические методы. Однако для разработки программного решения по прогнозированию наиболее подходящими являются методы экстраполяции тенденций, поскольку применение данных методов базируется на экстраполяции временных рядов и не требует построения сложных прогнозных моделей.

Экстраполяция – это метод эмпирического исследования, основанный на определении функциональных закономерностей в развитии объекта и последующей проекции полученной аналитической зависимости на определённый временной отрезок в будущем с целью получения прогнозного значения. К наиболее надёжным методам экстраполяции относятся метод скользящей средней, метод экспоненциального сглаживания и метод наименьших квадратов.

Сущность метода наименьших квадратов состоит в минимизации суммы квадратических отклонений между расчетными и фактическими значениями. Расчетные значения находятся с помощью специально подобранного уравнения, называемого уравнением регрессии. Чем меньше разница между действительными и расчетными значениями, тем более точен прогноз, построенный по уравнению регрессии. Тип кривой, наиболее точно отражающий поведение исследуемого явления, определяется посредством наблюдений. Например, если изменение фактических значений происходит в арифметической прогрессии, то сглаживание целесообразно производить по прямой. Если же изменение происходит в геометрической прогрессии, сглаживание необходимо производить по показательной функции. [7]

Результат процесса минимизации расстояний между расчётными и фактическими значениями при использовании метода наименьших квадратов наглядно продемонстрирован на рис. 1.5 (фактические значения отмечены синим цветом, расчётные – красным).

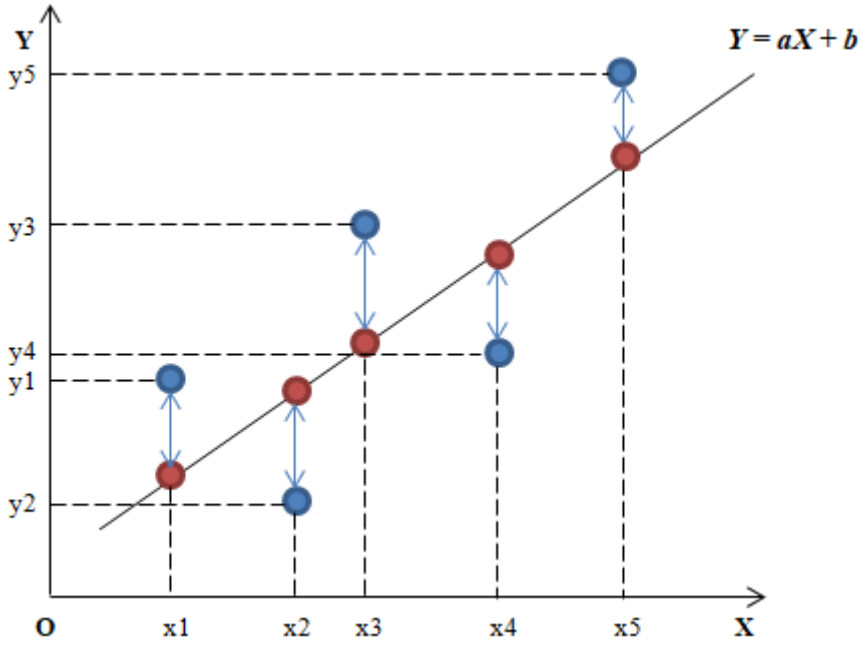


Рисунок 1.5. Метод наименьших квадратов

Расчётная формула для прогнозирования методом наименьших квадратов выглядит следующим образом:

$$Y_{t+1} = a * X + b, \quad (1.1)$$

где $t + 1$ – период прогнозирования;

Y_{t+1} – прогнозное значение;

X – обозначение времени:

a, b – коэффициенты.

Для расчёта значений коэффициентов a и b используются следующие формулы:

$$a = \left(\sum(Y_ф X) - \left(\sum X \sum Y_ф\right)/n\right)/(\sum X^2 - (\sum X)^2/n), \quad (1.2)$$

$$b = \left(\sum Y_ф /n\right)/(a \sum X /n), \quad (1.3)$$

где $Y_ф$ – фактические значения уровней временного ряда;

n – число уровней временного ряда.

Таким образом, с помощью метода наименьших квадратов достигается необходимый уровень сглаживания временного ряда, что позволяет отразить общие закономерности в поведении изучаемого процесса. При этом время выступает в уравнении регрессии в качестве переменной величины, а уровни ряда являются функцией этой переменной.

С помощью расчёта среднеквадратической ошибки можно опытным путём определить характер функции, наиболее точно описывающей тренд. Формула для расчёта среднеквадратической ошибки выглядит следующим образом:

$$S = \sqrt{\sum(Y_ф - Y_р)^2/n - p - 1}, \quad (1.4)$$

где $Y_ф$ – фактические значения ряда динамики;

Y_p – расчётные значения ряда динамики;

n – число уровней временного ряда;

p – число параметров в формулах, описывающих тренд.

Существенными недостатками метода наименьших квадратов являются уменьшение точности прогноза при увеличении прогнозного периода и неоднозначность в вопросе выбора подходящего уравнения регрессии.

Другим эффективным методом сглаживания временных рядов является метод скользящих средних. Использование данного метода позволяет почти полностью исключить случайные колебания расчётных значений. Эффект погашения (сглаживания) случайных колебаний достигается за счёт усреднения первоначальных уровней временного ряда внутри выбранного временного интервала. [6] Операция по вычислению среднего значения выполняется для каждого выделенного периода времени и затрагивает все уровни ряда. При этом период для усреднения одинаков на каждом шаге. Таким образом, для каждого интервала скользящая средняя центрируется относительно своей окрестности.

Плавность результирующего тренда увеличивается при расширении интервала сглаживания, значительно сокращая количество наблюдений, что может привести к определённым трудностям при прогнозировании. Вычислительная формула при экстраполяции методом скользящих средних выглядит следующим образом:

$$Y_{t+1} = m_{t-1} + (1/n) * (Y_t - Y_{t-1}), \quad (1.5)$$

где $t + 1$ – период прогнозирования;

t – период, предшествующий прогнозному;

Y_{t+1} – прогнозное значение;

m_{t-1} – скользящая средняя за два периода до прогнозного;

n – число уровней в интервале сглаживания;

Y_t – фактическое значение ряда динамики за предшествующий период;

Y_{t-1} – фактическое значение ряда динамики за два периода до прогнозного.

Результат сглаживания временного ряда методом скользящих средних представлен на рис. 1.6 (фактические значения отмечены синим цветом, сглаженные – красным):

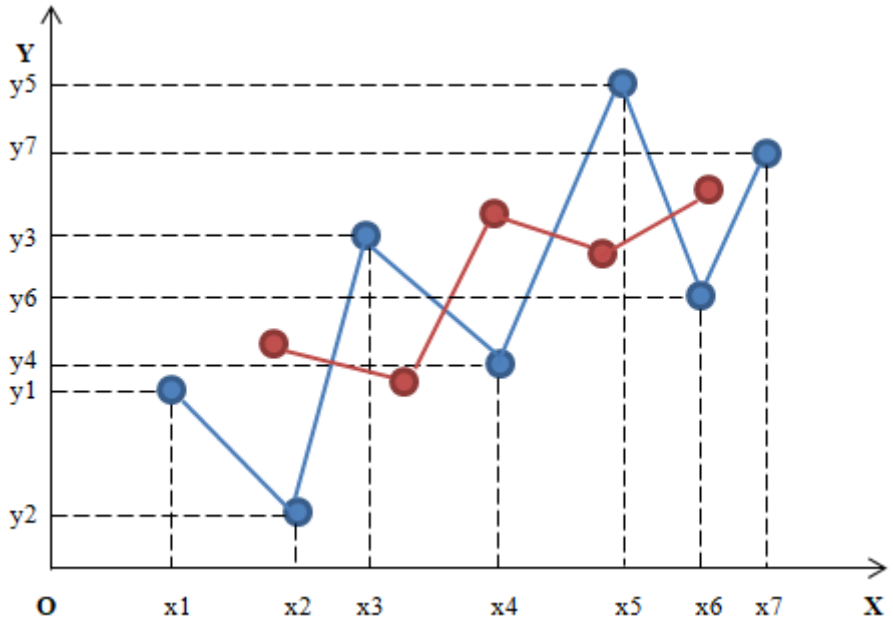


Рис. 1.6. Метод скользящих средних

С целью проверки уровня достоверности прогноза производится расчёт средней относительной ошибки ε по следующей формуле:

$$\varepsilon = (1/n) * \sum \left| Y_{\text{ф}} - Y_{\text{р}} \right| * 100\% / Y_{\text{ф}}, \qquad (1.6)$$

где n – число уровней временного ряда;

$Y_{\text{ф}}$ – фактические значения ряда динамики;

$Y_{\text{р}}$ – расчётные значения ряда динамики.

При необходимости среднесрочного прогнозирования целесообразно применение метода экспоненциального сглаживания. Однако в данном случае экстраполяция возможна лишь на один прогнозный период.

Основными достоинствами метода экспоненциального сглаживания являются простота вычислительных операций и возможность учёта весовых значений исходных данных. [6] Расчётная формула при прогнозировании методом экспоненциального сглаживания выглядит следующим образом:

$$U_{t+1} = \alpha Y_t + (1 - \alpha)U_t, \quad (1.7)$$

где $t + 1$ – период прогнозирования;

t – период, предшествующий прогнозному;

U_{t+1} – прогнозируемый показатель;

α – параметр сглаживания;

Y_t – фактическое значение явления за период до прогнозного;

U_t – экспоненциально взвешенная средняя для периода, предшествующего прогнозному.

Значение параметра сглаживания α вычисляется по формуле 1.8:

$$\alpha = 2/(n + 1), \quad (1.8)$$

где n – число наблюдений, входящих в интервал сглаживания.

Значением параметра сглаживания определяется степень влияния предыдущих наблюдений на результат экстраполяции. При постепенном увеличении значения параметра α снижается зависимость прогнозного значения от прошлых значений временного ряда. В свою очередь, в случае постепенного уменьшения параметра сглаживания учитывается влияние усреднённых величин, полученных путём использования предыдущих фактических значений временного ряда. [8]

За начальное значение U_0 принимается либо средняя арифметическая имеющихся фактических значений, либо исходное фактическое значение.

Процесс сглаживания фактических значений временного ряда наглядно продемонстрирован на рисунке 1.7.

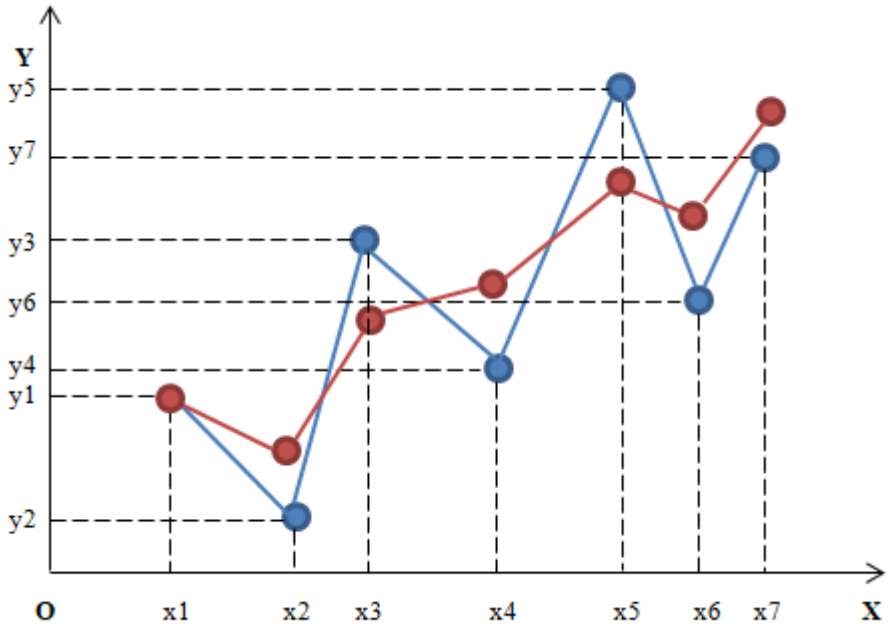


Рис. 1.7. Метод экспоненциального сглаживания

2. ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

2.1 Функциональный анализ приложения

Функциональная схема приложения представляет собой описание принципов взаимодействия основных структурных модулей программного обеспечения с указанием движения потоков данных между отдельными компонентами. [14]

Функциональная схема программного обеспечения по планированию трудовых ресурсов представлена на рис. 2.1:

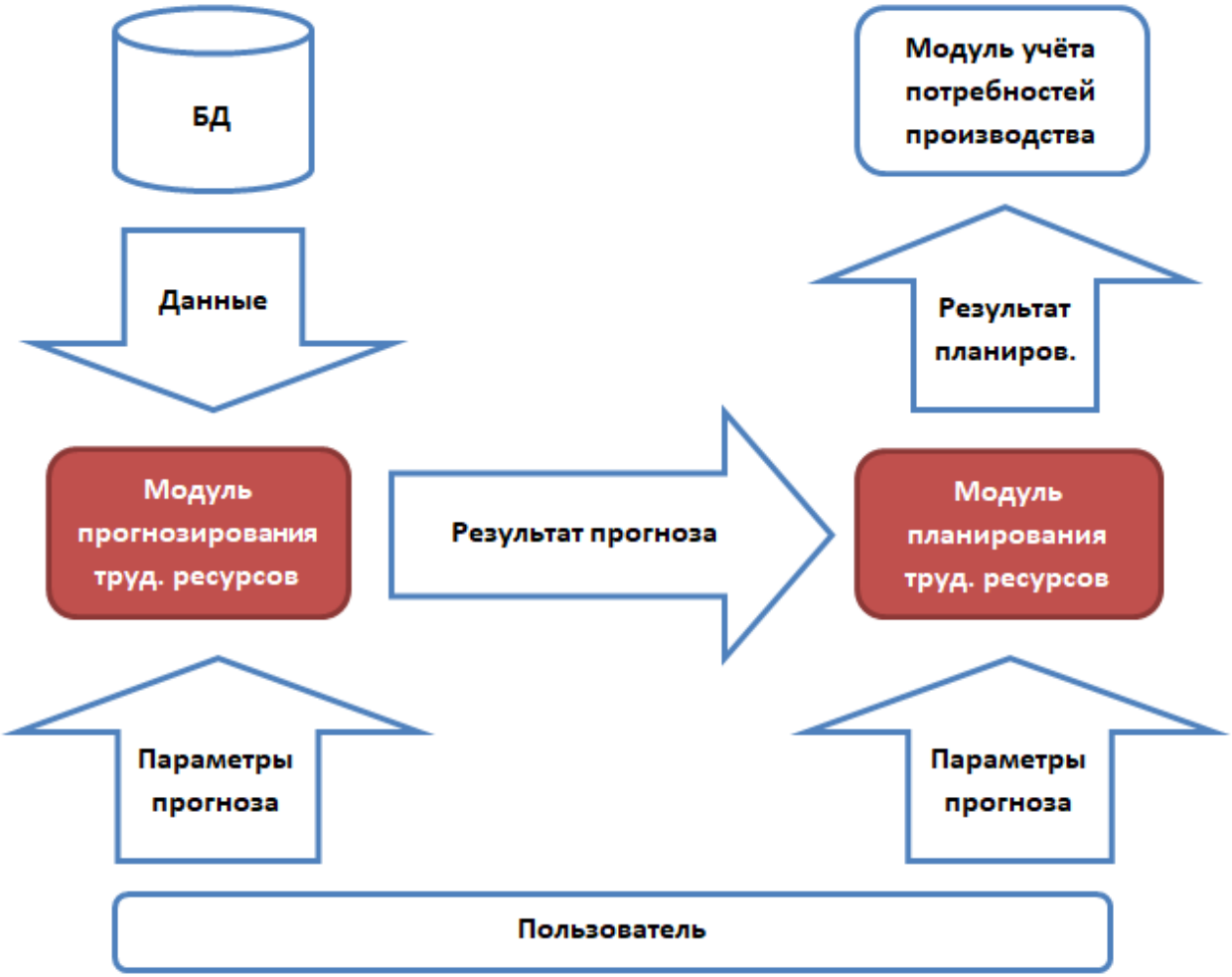


Рис. 2.1. Функциональная схема приложения

Расчётные данные, которые необходимы для экстраполяции значений временного ряда, поступают в модуль прогнозирования трудовых ресурсов из базы данных предприятия. Решение задачи прогнозирования потребностей производства в трудовых ресурсах подразумевает анализ информации о производительности труда и объёмах производства за определённый период времени. Таким образом, экстраполяции подвергаются данные о количестве произведённой продукции и коэффициенты производительности труда. Полученные прогнозные значения позволяют определить количество рабочей силы, необходимое для производства того или иного вида продукции в установленных объёмах.

Параметры, характеризующие процесс прогнозирования, задаются пользователем. К параметрам прогноза непосредственно относятся диапазон анализа фактических значений исследуемого явления, метод экстраполяции, глубина прогноза. Максимально допустимый диапазон анализа значений задаётся по умолчанию. Пользователь при необходимости способен изменить его по собственному усмотрению. Конкретный метод экстраполяции также задан по умолчанию и может быть изменён пользователем. Выбор метода экстраполяции зависит, прежде всего, от целей прогнозирования. В случае необходимости краткосрочного прогноза целесообразно использование метода наименьших квадратов, либо метода скользящих средних. Если же требуется прогноз на среднесрочный период, необходимо применение метода экспоненциального сглаживания.

Модуль прогнозирования трудовых ресурсов также содержит полный список номенклатурных позиций, по каждой из которых прогнозируется потребность в рабочей силе. При необходимости пользователь может указать конкретные виды продукции, по которым требуется прогноз. Временной ряд, описывающий количество произведённой продукции данного вида за определённый период, а также динамика потребности в рабочей силе выводятся на экран. Анализ количественных показателей временных рядов позволяет выявить общую тенденцию в колебаниях уровня производства,

которые оказывают непосредственное влияние на количество необходимой рабочей силы.

Результат процесса прогнозирования используется при планировании трудовых ресурсов. В частности, в зависимости от прогнозных значений формируются показатели нехватки рабочей силы по тем или иным видам производимой продукции. В модуле планирования трудовых ресурсов также отображаются коэффициенты точности прогноза и прогнозные значения уровня производства. В том случае, если величина ошибки прогноза не устраивает пользователя, процесс прогнозирования можно повторить с применением другого метода.

Показатели, полученные в результате процесса планирования трудовых ресурсов, непосредственно необходимы для учёта общих потребностей производства в материальных ресурсах различного характера. Однако функциональная составляющая данной подсистемы не затрагивается при выполнении дипломной работы.

2.2 Проектирование архитектуры приложения

Процесс проектирования архитектуры приложения состоит в установлении взаимоотношений между различными компонентами информационной системы. Правильно подобранный тип архитектуры позволит разработать программное решение, соответствующее критериям безопасности, масштабируемости и производительности. [11]

В качестве наиболее подходящей для данного программного обеспечения была выбрана классическая трёхуровневая клиент-серверная архитектура. Выбор указанного архитектурного шаблона обусловлен тем фактом, что в процессе разработки информационной системы предприятия использовался многоуровневый подход к проектированию. Таким образом, разбиение программного обеспечения на отдельные архитектурные уровни

значительно облегчит процесс его интеграции в информационную систему предприятия.

Многоуровневая архитектура представляет собой тип архитектуры программного обеспечения, позволяющий разделить механизм доступа к данным, выполнение бизнес-процессов и управление системой ввода/вывода представлений на отдельные логические уровни. Уровень представления обеспечивает взаимодействие между информационной системой и конечным пользователем. На уровне представления реализуются, главным образом, компоненты пользовательского интерфейса, механизмы получения входных и формирования выходных данных. Уровень бизнес-логики предназначен для обработки данных, полученных на уровне представления. Данный логический слой содержит компоненты для взаимодействия с базами данных и передаёт уровню представления результат проведённых вычислений. Уровень доступа к данным реализует классы, необходимые для использования разнообразных технологий доступа к данным, хранит модели данных и обеспечивает средства, позволяющие уровню бизнес-логики получать необходимые данные. [12]

Следует также подчеркнуть, что уровень представления и уровень доступа к данным не могут напрямую взаимодействовать между собой, посредником между ними выступает уровень бизнес-логики. При этом уровень доступа к данным является полностью автономным, то есть никак не зависит от других уровней. Уровень бизнес-логики, в свою очередь, зависит от уровня доступа к данным, а уровень представления непосредственно от уровня бизнес-логики. Структурные компоненты архитектуры приложения должны сохранять слабую связность, что влечёт за собой необходимость внедрения зависимостей.

Структурная схема архитектуры программного обеспечения по планированию потребностей производства в трудовых ресурсах (с указанием взаимосвязи между отдельными архитектурными слоями) представлена на рисунке 2.2.

Рис. 2.2. Архитектура программного обеспечения

Данные, содержащиеся в запросе клиента, поступают на уровень представления, где с помощью специальных средств происходит их форматирование для обработки с помощью модулей бизнес-логики. Компоненты пользовательского интерфейса являются стандартными html-страницами, содержащими поля ввода/вывода для получения входных данных и презентации результатов вычислений. Обработка различных клиентских запросов, формирование представлений и управление их выводом также происходят на данном логическом уровне.

Применительно к предметной области программного обеспечения по планированию потребностей производства в трудовых ресурсах компоненты пользовательского интерфейса будут содержать поля для указания диапазона анализа имеющейся выборки (даты начала и окончания анализа), метода

прогнозирования и глубины прогноза. После воздействия пользователя на компонент отправки формы введённые данные отправляются на сервер приложений для их последующей обработки. Компоненты бизнес-логики, реализованные на следующем логическом уровне, содержат алгоритмы обработки информации, введённой пользователем, либо содержащейся в базе данных.

Бизнес-логика приложения по планированию трудовых ресурсов заключается в экстраполяции значений, характеризующих производственные потребности в трудовых ресурсах, на определённый период времени. Так как в данной дипломной работе прогнозирование осуществляется посредством обработки значений временных рядов, компоненты, описывающие бизнес-логику, будут содержать функционал экстраполяции с помощью следующих методов: метод наименьших квадратов, метод скользящей средней, метод экспоненциального сглаживания.

Обращение к разработанным бизнес-моделям осуществляется с помощью специального сервиса, содержащего реализации соответствующих интерфейсов доступа. Модули уровня бизнес-логики не могут напрямую использовать структуры данных, описанные на более низком уровне, для этого применяются специальные промежуточные сущности (объекты передачи данных). Введение подобных ограничений связано с необходимостью ослабления структурной зависимости между объектами разных логических уровней. С этой же целью на уровне бизнес-логики реализуется модуль внедрения зависимости между интерфейсом доступа к сущностям базы данных и его конкретной реализацией, исключая таким образом необходимость работы с механизмами подключения.

Слой доступа к данным содержит описание структуры сущностей, хранящихся в базе данных. В нашем случае корпоративная база данных ИД ООО «Монтажавтоматика» содержит коллекции объектов, описывающих производственные процессы. Решение задачи прогнозирования потребностей производства в трудовых ресурсах подразумевает использование тех

объектов данных, которые содержат сведения о заказах на производство определённых видов продукции, а также количественную информацию о сотрудниках предприятия, принимающих непосредственное участие в том или ином производственном процессе.

Компоненты доступа к данным представляют собой реализацию репозиториев, позволяющих выполнять основные операции по обращению к таблицам базы данных (запись, редактирование, удаление, копирование). Сервисные компоненты содержат описание интерфейсов доступа к службам, реализованным на уровне доступа к данным.

2.3 Описание средств разработки

Технологии разработки программного обеспечения, используемые в данной дипломной работе, активно применяются на предприятии ИД ООО «Монтажавтоматика» при программировании информационной системы. Таким образом, выбор средств разработки приложения обусловлен, в частности, требованиями технологической совместимости, позволяющими беспрепятственно интегрировать приложение в основное программное обеспечение предприятия.

В качестве архитектурной платформы используется инфраструктура Microsoft ASP.NET MVC 5, которая является усовершенствованной версией платформы для разработки web-приложений ASP.NET. Преимуществами данной технологии по сравнению с предыдущими версиями можно назвать использование MVC в качестве архитектурного шаблона для разработки приложения, широкие возможности в области модификации существующих компонент, контроль за соблюдением стандартов разметки web-страниц, поддержка модульного и автоматизированного тестирования, использование усовершенствованной системы маршрутизации, создание улучшенного API-интерфейса. [17]

Платформа ASP.NET MVC 5 реализует существенно улучшенный вариант архитектурного шаблона MVC, который гарантирует рациональное разбиение приложения на различные функциональные уровни. Реализация MVC в качестве основного архитектурного шаблона позволяет ASP.NET MVC 5 конкурировать с другими популярными web-платформами (например, Ruby on Rails).

Возможности модификации существующих компонент, реализованные в данной web-платформе, предусматривают создание дочерних классов, которые наследуют функциональность стандартных реализаций. Полученная дочерняя структура позволяет корректировать поведение наследуемых методов, или при необходимости полностью изменять его. [18]

В платформу ASP.NET MVC 5 встроены средства, генерирующие html-представления в полном соответствии со стандартами разметки web-страниц. Отсутствие каких-либо ограничивающих требований в области html-разметки позволяет активно применять различные готовые решения при проектировании пользовательского интерфейса (например, Bootstrap, jQuery). В частности, возможность использования таких известных библиотек, как Bootstrap или jQuery, встроена в данную платформу по умолчанию. Также в инфраструктуре платформы реализована функция контроля за выполнением http-запросов, отправленных на сервер клиентской стороной.

Использование архитектурного шаблона MVC, подразумевающего разбиение функциональности приложения на отдельные логические уровни, облегчает процесс тестирования программного обеспечения. В частности, структурные компоненты приложения реализованы в web-платформе таким образом, чтобы обеспечить возможности для выполнения модульного и автоматизированного тестирования. [19]

В качестве системы управления базами данных в данном проекте используется Microsoft SQL Server, являющаяся наиболее удачным выбором при использовании операционных систем Windows по причине её глубокой интегрированности с данным классом операционных систем. Платформа

SQL Server позволяет работать с массивами данных различной степени структурированности, обеспечивая при этом необходимый уровень надёжности.

Для работы с базами данных применяется технология ADO.NET Entity Framework, которая представляет собой ORM-решение, реализованное для платформы .NET Framework. Данная технология доступа к данным позволяет установить соответствие между компонентами, хранящимися в базе данных, и объектно-ориентированными сущностями. Таким образом, технология ADO.NET Entity Framework предоставляет возможности по обращению к объектам баз данных без использования SQL-запросов.

Технологии, применяемые в процессе разработки программного обеспечения, представлены на рисунке 2.3.

Рис. 2.3. Средства разработки приложения

3. РЕАЛИЗАЦИЯ И ТЕСТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

3.1 Реализация уровня доступа к данным

Уровень доступа к данным содержит модели сущностей, хранящихся в базе данных, а также реализует объекты, позволяющие взаимодействовать с базой данных. Логические слои приложения реализованы в виде отдельных проектов, входящих в состав программного решения под названием MTA. Структура программного решения проиллюстрирована на рисунке 3.1.

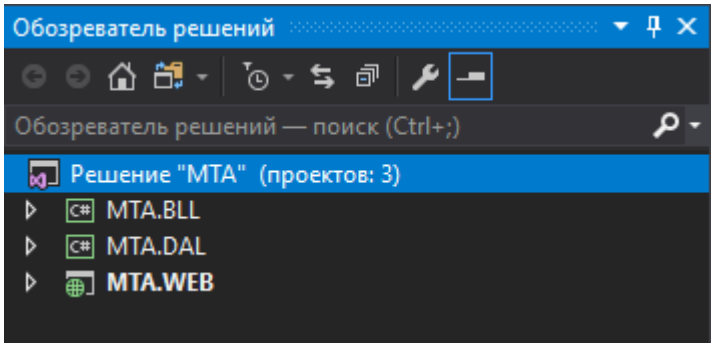


Рис. 3.1. Структура программного решения

Уровень доступа к данным реализован в проекте MTA.DAL, структура которого представлена на рисунке 3.2.

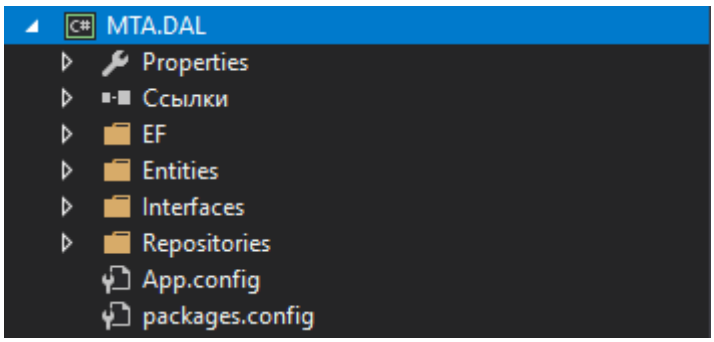


Рис. 3.2. Структура проекта MTA.DAL

Модели данных, объекты которых хранятся в базе данных, определены в папке Entities. Для решения задачи прогнозирования нам необходимы следующие структуры данных: заказы на производство, спецификации заказов, производимая продукция, производственные работники. Сущность заказов на производство представлена классом Order.cs, который состоит из следующих полей, описанных в листинге 3.1.

Листинг 3.1. Модель заказа на производство.

```
public class Order
{
   public int Id { get; set; }
   public DateTime Date { get; set; }
   public string Client { get; set; }
   public double Price { get; set; }
}
```

Поле Id хранит идентификатор объекта, в свойстве Date содержится дата получения заказа на производство; в свойстве Client, в свою очередь, хранится название организации, оформившей заказ, в свойстве Price хранится общая стоимость заказа. С помощью даты того или иного заказа будут формироваться значения временного ряда, которые содержат количество заказанной и произведённой продукции в течение одного периода времени.

Спецификации заказов хранятся в модели OrderProduct.cs, которая содержит информацию о количестве заказанной продукции, цену реализации, номер заказа и номер продукции. В поле Id данной сущности указан идентификатор спецификации, в поле Number хранится количество заказанной продукции, в поле Sum содержится цена реализации; свойство OrderId содержит номер заказа, к которому относится данная спецификация, свойство ProductId хранит идентификатор заказанной продукции. Свойства OrderId и ProductId являются навигационными свойствами, по которым

производится загрузка экземпляров соответствующих объектов. С помощью данной сущности будет производиться расчёт количества произведённой продукции в рамках отдельного периода времени. Свойства модели OrderProdcut.cs представлены в листинге 3.2.

Листинг 3.2. Модель спецификации заказа

```
public class OrderProduct
{
    public int Id { get; set; }
    public int Number { get; set; }
    public double Sum { get; set; }

    public int OrderId { get; set; }
    public Order Order { get; set; }

    public int ProductId { get; set; }
    public Product Product { get; set; }
}
```

Модель производимой продукции описана в листинге 3.3. Класс Product.cs содержит информацию об отдельной номенклатурной позиции.

Листинг 3.3. Модель производимой продукции

```
public class Product
{
    public int Id { get; set; }
    public string Name { get; set; }
    public string Unit { get; set; }
    public double Price { get; set; }
}
```

Свойство Id хранит идентификатор объекта, свойство Name предназначено для хранения названия номенклатурной позиции; поле Unit содержит единицу измерения, поле Price – стоимость единицы продукции данного вида.

Модель, представляющая производственного работника, реализована в классе Employee.cs и представлена в листинге 3.4:

Листинг 3.4. Модель работника

```
public class Employee
{
   public int Id { get; set; }
   public string Surname { get; set; }
   public string Name { get; set; }
}
```

В поле Id содержится идентификатор объекта, поля Surname и Name предназначены для хранения фамилии и имени работника соответственно.

Для хранения взаимосвязи между работником и типом продукции, в изготовлении которой непосредственно задействован данный работник, предназначен класс Production.cs, описанный в листинге 3.5.

Листинг 3.5. Модель производства

```
public class Production
{
   public int Id { get; set; }
   public int ProductId { get; set; }
   public Product Product { get; set; }
   public int EmployeeId { get; set; }
   public Employee Employee { get; set; }
}
```

Соответствие между значениями, хранящимися в базе данных, и описанными сущностями устанавливается с помощью технологии доступа к данным EntityFramework. С этой целью в папке EF определён класс контекста данных EnterpriseContext.cs, который представлен в листинге 3.6:

Листинг 3.6. Класс контекста данных

```
public class EnterpriseContext : DbContext
{
   public DbSet<Order> Orders { get; set; }
   public DbSet<Product> Products { get; set; }
   public DbSet<OrderProduct> OrderProducts { get; set; }
   public DbSet<Production> Productions { get; set; }
   public DbSet<Employee> Employees { get; set; }
   public EnterpriseContext(string connectionString) : base(connectionString) { }
}
```

Свойства класса, определяющего контекст данных, возвращают коллекции объектов определённого типа, которые содержат сущности соответствующих таблиц базы данных. Конструктор класса контекста данных в качестве параметра принимает строку подключения к базе данных для установления соединения.

Папка Repositories содержит различные реализации репозиториев, инкапсулирующих возможности обращения к таблицам базы данных. Каждый класс репозитория позволяет выполнять стандартный набор операций по добавлению, извлечению, удалению и редактированию информации, хранящейся в базе данных.

В папке Interfaces содержатся классы IRepository.cs и IUnitOfWork.cs, представляющие собой интерфейс доступа к репозиториям и интерфейс, организующий унифицированное подключение к базе данных для различных

репозиториев, соответственно. Интерфейс IRepository.cs представлен в листинге 3.7.

Листниг 3.7. Интерфейс доступа к репозиторию

```
public interface IRepository<T> where T : class
{
    IEnumerable<T> GetAll();
    T Get(int id);
    IEnumerable<T> Find(Func<T, Boolean> predicate);
    void Create(T item);
    void Update(T item);
    void Delete(int id);
}
```

Метод GetAll возвращает коллекцию объектов определённого типа, хранящихся в базе данных. Метод Get позволяет получить объект по его идентификатору. Методы Create и Update, в свою очередь, выполняют удаление и редактирование объектов соответственно. С помощью метода Find можно осуществить поиск объекта в базе данных.

Интерфейс IUnitOfWork.cs позволяет централизованно подключаться к базе данных и обращаться к различным её сущностям посредством создания экземпляров соответствующих репозиториев.

Таким образом, в данном разделе наглядно описана структура уровня доступа к данным, включающая реализацию сущностей базы данных, контекста данных, а также репозиториев доступа к соответствующим таблицам базы данных. Использование интерфейсных ссылок позволяет абстрагироваться от конкретной технологии доступа к данным, предоставляя возможности модификации, расширения и оптимизации существующих реализаций.

3.2 Реализация уровня бизнес-логики

Уровень бизнес-логики служит своеобразным посредником между слоем доступа к данным и слоем представления. Объекты, полученные на уровне доступа к данным и передаваемые уровнем представления, доступны только уровню бизнес-логики. Также данный логический слой содержит реализацию бизнес-моделей, инкапсулирующих всю вычислительную логику приложения.

Уровень бизнес-логики реализован в проекте MTA.BLL, структура которого представлена на рис. 3.3.

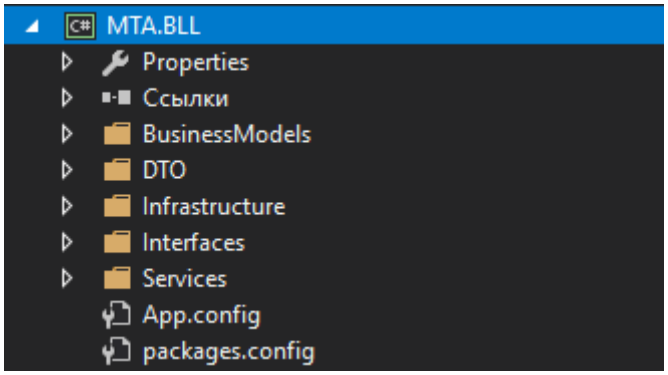


Рис. 3.3. Структура проекта MTA.BLL

Уровень представления не может использовать функциональность уровня доступа к данным, поэтому для передачи данных на уровень представления необходимо создание альтернативных сущностей – объектов передачи данных. Соответствие между объектами уровня доступа к данным и объектами передачи данных устанавливается при помощи технологии AutoMapper. Промежуточные структуры данных, необходимые для передачи на уровень представления, хранятся в папке DTO. В качестве примера в листинге 3.8 приведена реализация объекта передачи данных OrderDTO.cs, свойства которого идентичны свойствам объекта Order.cs, представленного

на уровне доступа к данным и описывающего данные о заказах на производство.

Листинг 3.8. Объект передачи данных OrderDTO.cs

```
public class OrderDTO
{
   public int Id { get; set; }
   public DateTime Date { get; set; }
   public string Client { get; set; }
   public double Price { get; set; }
}
```

Бизнес-модели приложения хранятся в папке BusinessModels и представляют собой реализацию следующих методов прогнозирования: метод наименьших квадратов, метод экспоненциального сглаживания, метод скользящей средней.

Реализация метода наименьших квадратов основана на определении значений расчётных коэффициентов, которые необходимы для вычисления прогнозного значения. В данном случае экстраполяция осуществляется по прямой, наклон которой должен выбираться таким образом, чтобы минимизировать среднеквадратическое расстояние между расчётными и фактическими значениями. Получив корректное уравнение регрессии, можно экстраполировать значения на один или несколько временных периодов.

Вариант реализации метода наименьших квадратов, включающий процедуру вычисления коэффициентов уравнения регрессии путём расчёта значений соответствующих сумм (сумма отрезков временного ряда, сумма фактических значений временного ряда, сумма произведений отрезков и значений временного ряда, сумма квадратов отрезков временного ряда) и непосредственно расчёт прогнозного значения представлены в листинге 3.9.

Листинг 3.9. Реализация метода наименьших квадратов

```
public double[] GetForecast()
{
        double sumX = 0, sumY = 0, sumXY = 0, sumXX = 0;
        for (int i = 0; i < samples.Length; i++)
        {
           sumX += i + 1;
           sumY += samples[i];
           sumXY += samples[i] * (i + 1);
           sumXX += Math.Pow(i + 1, 2);
        }
        double a = sumXY - (sumX * sumY) / samples.Length;
        a = a / (sumXX - Math.Pow(sumX, 2) / samples.Length);
        double b = sumY / samples.Length - a * sumX / samples.Length;
        double[] forecast = new double[depth];

        for (int i = 0; i < depth; i++)
           forecast[i] = a * (samples.Length + i + 1) + b;
        return forecast;
}
```

В переменных sumX и sumY содержатся значения суммы фактических и расчётных значений временного ряда соответственно. Переменная sumXX хранит сумму возведённых в квадрат фактических значений, переменная sumXY – сумму произведений расчётных и фактических значений. После расчёта значений переменных sumX, sumY, sumXX и sumXY производится вычисление коэффициентов уравнения регрессии. Значения коэффициентов уравнения регрессии хранятся в переменных a и b. Результат экстраполяции содержится в массиве forecast.

Реализация метода экспоненциального сглаживания содержит, главным образом, процедуру усреднения фактических значений исследуемого явления с учётом параметра сглаживания. В листинге 3.10 представлен пример реализации метода экспоненциального сглаживания.

Листинг 3.10. Реализация метода экспоненциального сглаживания

```
public double[] GetForecast()
{
        double parameter = 2 / ((double) samples.Length + 1);
        double[] weightedValues = new double[samples.Length];
        weightedValues[0] = 0;

        for (int i = 0; i < samples.Length; i++)
            weightedValues[0] += samples[i];
          weightedValues[0] /= samples.Length;

          for (int i = 1; i < samples.Length; i++)
          {
            weightedValues[i] = parameter * samples[i - 1] + (1 - parameter) *
            weightedValues[i - 1];
          }
          double[] forecast = { parameter * samples[samples.Length - 1] + (1 –
          parameter) * weightedValues[samples.Length - 1] };
          return forecast;
}
```

Значение параметра сглаживания содержится в переменной parameter. Сглаженные значения сохраняются в массив weightedValue. Результат экстраполяции помещается в массив forecast.

Доступ к функциональности слоя бизнес-логики осуществляется с помощью специального сервиса, интерфейс которого описан в листинге 3.11.

Листинг 3.11. Интерфейс доступа к слою бизнес-логики

```
public interface IForecastService
{
    IEnumerable<OrderProductDTO> GetOrderProducts();
    IEnumerable<ProductDTO> GetProducts();
    double[] GetProductSample(ProductDTO product, DateTime begin,
    DateTime end);
    double[] GetEmployeeSample(ProductDTO product, DateTime begin,
    DateTime end);
    double[,] Forecasting(DateTime begin, DateTime end, int depth);
}
```

Метод GetOrderProducts предназначен для копирования коллекции объектов OrderProduct, которые предоставляют информацию о продукции, содержащейся в заказах на производство. Метод GetProducts возвращает коллекцию объектов Product, описывающих производимую продукцию.

Методы GetProductSample и GetEmployeeSample принимают в качестве аргументов объект продукции, по которой необходим прогноз, и даты начала и окончания анализа имеющихся заказов на производство данного вида продукции. При этом метод GetProductSample возвращает временной ряд, содержащий информацию о количестве произведённой продукции в пределах указанного диапазона времени, а метод GetEmployeeSample – временной ряд, предоставляющий информацию о количестве трудовых ресурсов, задействованных в процессе производства данного вида продукции. С помощью метода Forecasting осуществляется непосредственно процесс экстраполяции значений временного ряда.

3.3 Реализация уровня представления

Уровень представления содержит механизмы, обеспечивающие возможность взаимодействия между пользователем и информационной системой. Процессы, связанные с управлением системой ввода/вывода, обработкой входных данных и формированием представлений, выполняются на данном логическом уровне. Функциональность уровня представления непосредственно реализована в проекте MTA.WEB, который представляет собой стандартное MVC-приложение. Структура проекта MTA.WEB описана на рисунке 3.4.

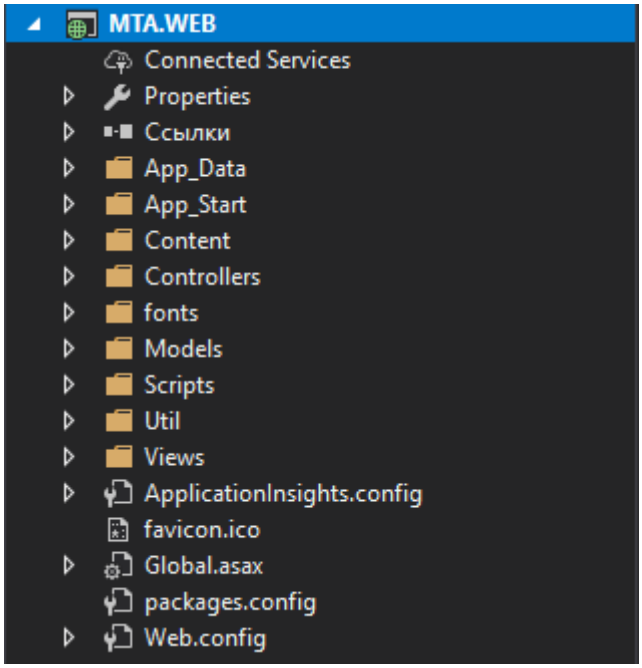


Рис. 3.4. Структура проекта MTA.WEB

Модели данных, применяемые на уровне представления, хранятся в папке Models. В ходе реализации приложения по планированию трудовых ресурсов использовалась модель ProductViewModel.cs, свойства которой описывают производимую продукцию. Данная модель представлена в листинге 3.12 и состоит из полей Id, Name, Unit, Price. В поле Id хранится идентификатор объекта, в поле Name – название продукции. Свойство Unit

содержит название единицы измерения, свойство Price – стоимость данного вида продукции.

Листинг 3.12. Модель ProductViewModel.cs

```
public class ProductViewModel
{
    public int Id { get; set; }
    public string Name { get; set; }
    public string Unit { get; set; }
    public double Price { get; set; }
}
```

Представления, возвращаемые пользователю при обращении к методам контроллера, хранятся в папке Views. Начальное представление реализовано в файле Index.cshtml и возвращается методом Index контроллера. Компоненты представления Index.cshtml представлены на рисунке 3.5.

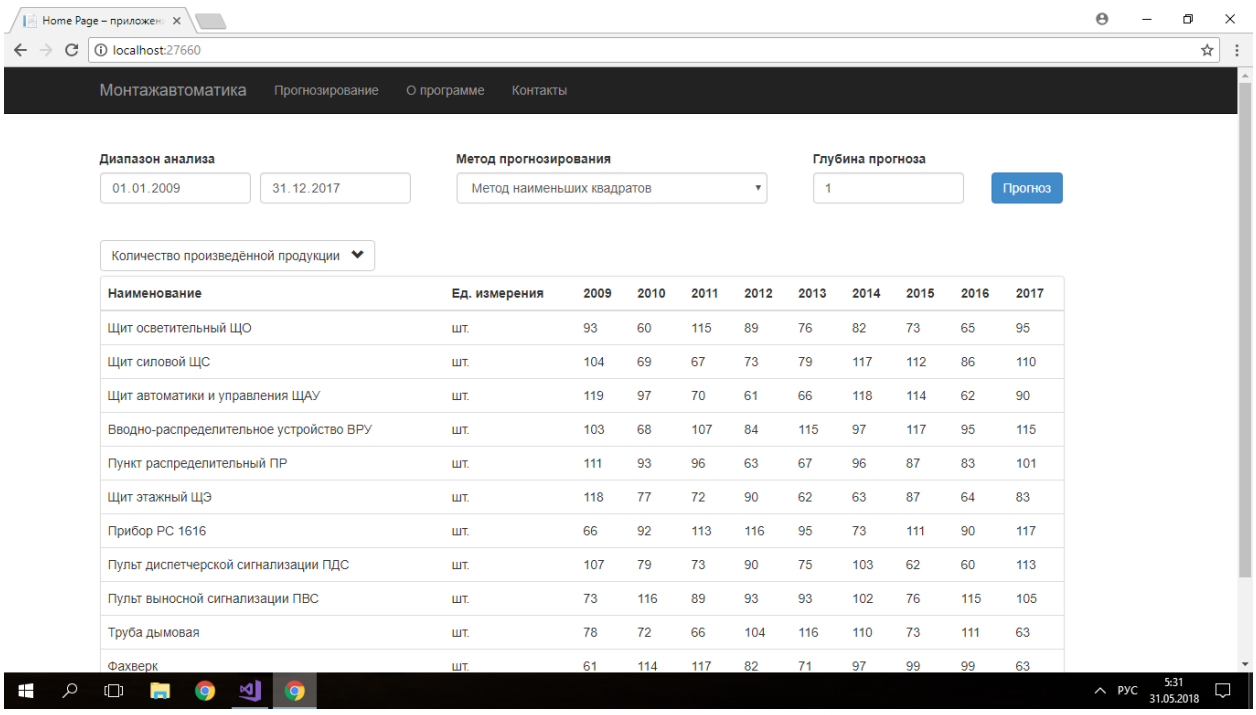


Рис. 3.5. Компоненты представления Index.cshtml

В поле Диапазон анализа необходимо указать временной отрезок (начальную и конечную даты), в пределах которого будут анализироваться фактические значения временного ряда. Максимально допустимый диапазон анализа задаётся по умолчанию. На рисунке 3.6 продемонстрирован процесс установки значений временного отрезка.

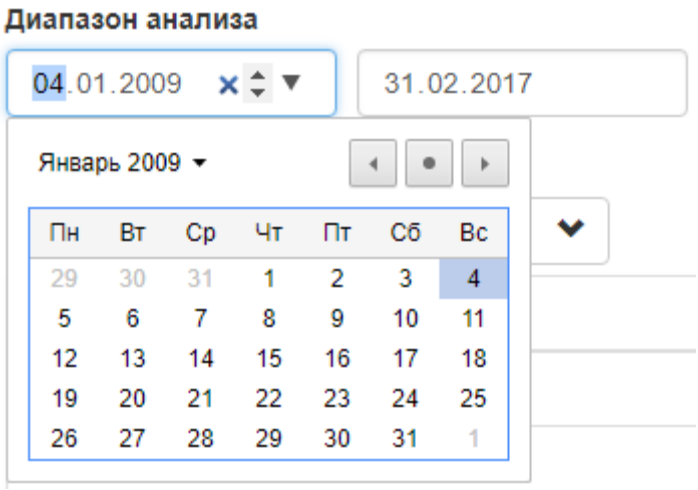


Рис. 3.6. Установка значений диапазона

Поле Метод прогнозирования предназначено для выбора метода, с помощью которого будет осуществляться экстраполяция. Выпадающий список содержит следующие методы прогнозирования: метод наименьших квадратов (задаётся по умолчанию), метод экспоненциального сглаживания, метод скользящей средней. Возможность выбора метода прогнозирования демонстрируется на рисунке 3.7.

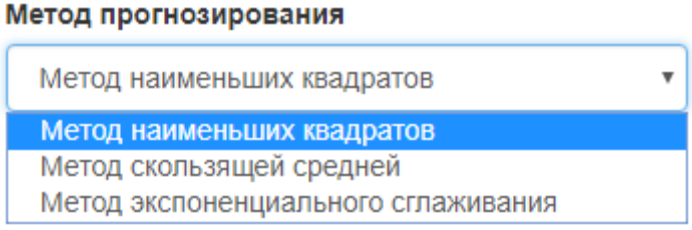


Рис. 3.7. Установка метода прогнозирования

В поле Глубина прогноза необходимо указать количество единичных временных отрезков, на которые будет осуществляться прогнозирование. По умолчанию глубина прогноза равна единице. В случае необходимости пользователь может увеличить прогнозный период до максимально допустимого. На рисунке 3.8 показана возможность изменения глубины прогноза.

Рис. 3.8. Установка глубины прогноза

Таблица, содержащая данные о количестве единиц продукции, произведённых за весь период времени, также выводится на экран при загрузке начального представления. Показатели количества произведённой продукции позволяют в целом оценить динамику производства в пределах установленного отрезка времени. Таблица, демонстрирующая показатели производства по каждому виду номенклатурной позиции представлена на рисунке 3.9.

Количество произведённой продукции

Наименование	Ед. измерения	2009	2010	2011	2012	2013	2014	2015	2016	2017
Щит осветительный ЩО	шт.	85	112	101	85	74	101	96	91	62
Щит силовой ЩС	шт.	94	73	80	98	115	108	98	100	60
Щит автоматики и управления ЩАУ	шт.	98	119	76	60	93	84	114	101	116
Вводно-распределительное устройство ВРУ	шт.	107	76	64	87	62	60	117	107	85
Пункт распределительный ПР	шт.	62	112	119	89	61	104	81	115	95
Щит этажный ЩЭ	шт.	71	116	82	94	79	106	75	117	87
Прибор РС 1616	шт.	93	84	79	76	90	93	104	108	75
Пульт диспетчерской сигнализации ПДС	шт.	118	67	70	80	101	68	82	96	107

Рис. 3.9. Данные о количестве произведённой продукции

При необходимости пользователь также может получить информацию о количестве трудовых ресурсов, задействованных в процессе производства того или иного вида продукции в пределах установленного отрезка времени. Демонстрация количественных показателей применяемой рабочей силы приведена на рисунке 3.10.

Количество трудовых ресурсов

Наименование	2009	2010	2011	2012	2013	2014	2015	2016	2017
Щит осветительный ЩО	64	33	58	90	75	96	78	76	42
Щит силовой ЩС	91	46	82	36	74	38	63	83	89
Щит автоматики и управления ЩАУ	60	32	71	39	90	43	67	54	76
Вводно-распределительное устройство ВРУ	74	38	48	82	89	50	59	92	65
Пункт распределительный ПР	54	66	74	70	51	47	83	96	62
Щит этажный ЩЭ	68	72	51	43	50	54	75	65	50
Прибор РС 1616	95	86	74	59	50	66	60	72	97
Пульт диспетчерской сигнализации ПДС	64	71	64	62	65	72	64	43	84

Рис. 3.10. Данные о количестве трудовых ресурсов

После указания всех необходимых параметров можно непосредственно приступать к процессу экстраполяции. При воздействии пользователя на соответствующий компонент (клавиша Прогноз) осуществляется вычисление прогнозных значений. Результат процесса прогнозирования отправляется в модуль планирования потребностей производства в трудовых ресурсах, за реализацию которого отвечает представление Forecast.cshtml. В данном представлении на экран выводятся результаты процесса прогнозирования, описанные следующими категориями: прогноз количества произведённой продукции; непосредственно прогноз количества необходимых трудовых ресурсов; показатели дефицита рабочей силы, которые формируются в зависимости от полученных прогнозных значений; средняя относительная ошибка прогноза, позволяющая оценить точность экстраполяции. В том случае, если величина ошибки прогноза не устраивает пользователя, можно

повторить процесс экстраполяции с применением других методов. Компоненты представления Forecast.cshtml представлены на рисунке 3.11.

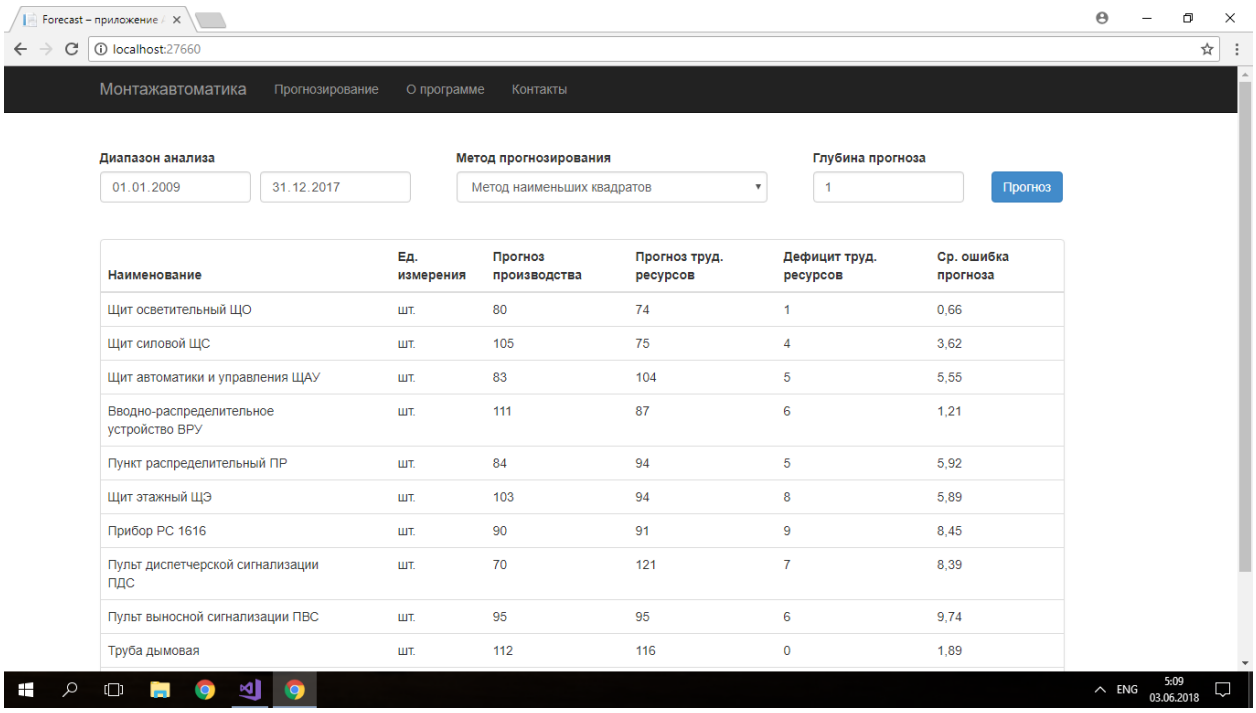


Рис. 3.11. Компоненты представления Forecast.cshtml

3.4 Тестирование и анализ результатов

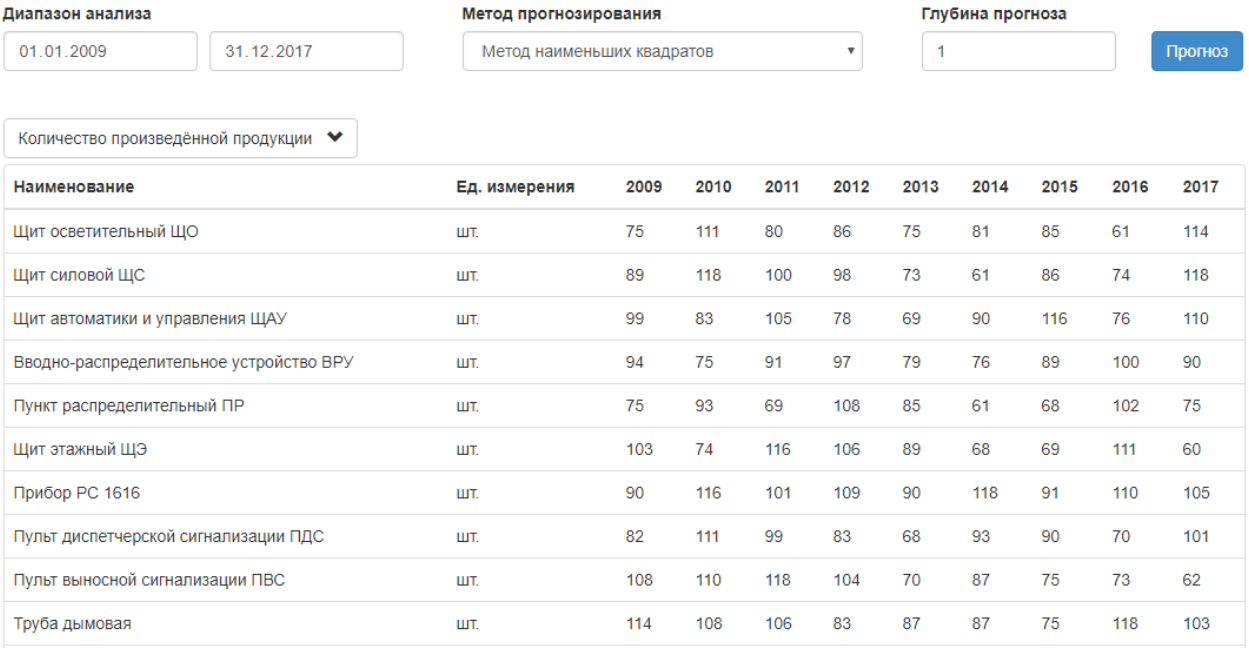
Данные, необходимые для проведения тестовых испытаний, хранятся в корпоративной базе данных предприятия. Процесс экстраполяции значений, описывающих потребность в рабочей силе, осуществим лишь при наличии данных о количестве произведённой продукции и о производительности труда в пределах установленного периода времени. В свою очередь, значения временного ряда, описывающего показатели производительности труда, равны отношению количества произведённой продукции к числу работников, задействованных в процессе производства.

Таким образом, временная выборка, непосредственно применяемая при тестировании программного обеспечения, представляет собой описание

количественных характеристик производственного процесса предприятия ИД ООО «Монтажавтоматика».

Тестовые испытания непосредственно включают оценку корректности функционирования разработанного программного обеспечения. В связи с этим, процесс тестирования разделён на три этапа, каждый из которых демонстрирует результат прогнозирования при использовании одного из реализованных методов. Диапазон анализа при этом принимает максимально допустимые значения, глубина прогноза равна одному временному периоду.

Тестовые данные, описывающие количество произведённой продукции в установленных временных пределах, представлены на рисунке 3.12.



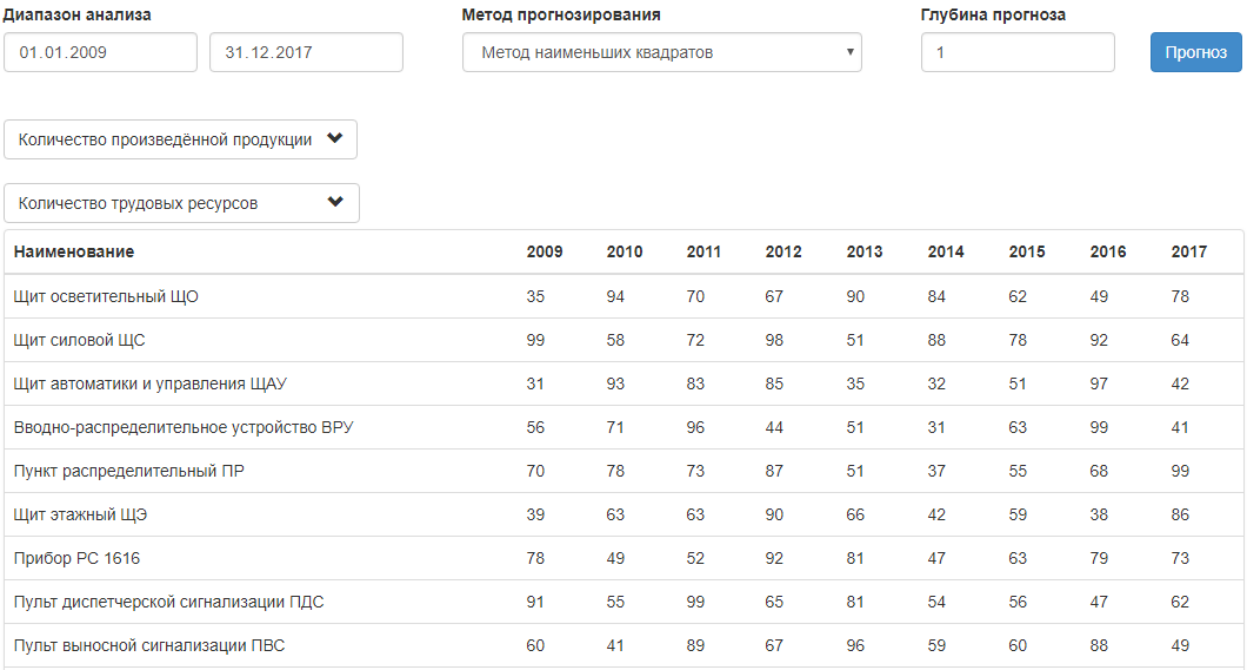
Наименование	Ед. измерения	2009	2010	2011	2012	2013	2014	2015	2016	2017
Щит осветительный ЩО	шт.	75	111	80	86	75	81	85	61	114
Щит силовой ЩС	шт.	89	118	100	98	73	61	86	74	118
Щит автоматики и управления ЩАУ	шт.	99	83	105	78	69	90	116	76	110
Вводно-распределительное устройство ВРУ	шт.	94	75	91	97	79	76	89	100	90
Пункт распределительный ПР	шт.	75	93	69	108	85	61	68	102	75
Щит этажный ЩЭ	шт.	103	74	116	106	89	68	69	111	60
Прибор РС 1616	шт.	90	116	101	109	90	118	91	110	105
Пульт диспетчерской сигнализации ПДС	шт.	82	111	99	83	68	93	90	70	101
Пульт выносной сигнализации ПВС	шт.	108	110	118	104	70	87	75	73	62
Труба дымовая	шт.	114	108	106	83	87	87	75	118	103

Рис. 3.12. Информация о количестве произведённой продукции

Данные, представленные на рисунке 3.12, охватывают первые десять номенклатурных позиций. Количественные данные описывают уровень производства в период с 2009 до 2017 года. По умолчанию в качестве наиболее подходящего метода прогнозирования задан метод наименьших квадратов, глубина прогноза равна единице. Для обеспечения равных условий в ходе тестовых испытаний значение глубины прогноза не будет

изменяться, поскольку прогнозирование методом экспоненциального сглаживания возможно лишь на один временной период, в отличие от двух других методов прогнозирования.

Тестовые данные, описывающие количество трудовых ресурсов, задействованных в процессе производства в установленный период времени, описаны на рисунке 3.13.



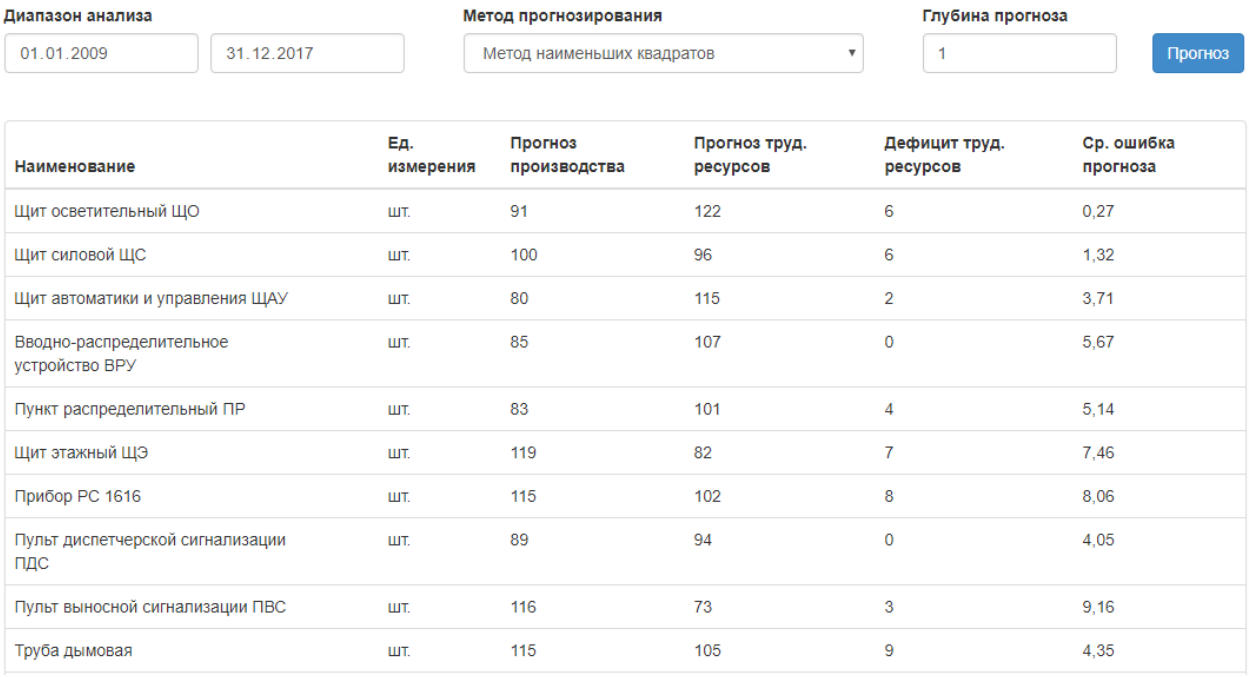
Наименование	2009	2010	2011	2012	2013	2014	2015	2016	2017
Щит осветительный ЩО	35	94	70	67	90	84	62	49	78
Щит силовой ЩС	99	58	72	98	51	88	78	92	64
Щит автоматики и управления ЩАУ	31	93	83	85	35	32	51	97	42
Вводно-распределительное устройство ВРУ	56	71	96	44	51	31	63	99	41
Пункт распределительный ПР	70	78	73	87	51	37	55	68	99
Щит этажный ЩЭ	39	63	63	90	66	42	59	38	86
Прибор РС 1616	78	49	52	92	81	47	63	79	73
Пульт диспетчерской сигнализации ПДС	91	55	99	65	81	54	56	47	62
Пульт выносной сигнализации ПВС	60	41	89	67	96	59	60	88	49

Рис. 3.13. Информация о количестве трудовых ресурсов

Таким образом, данные о количестве произведённой продукции и о количестве задействованных в процессе производства трудовых ресурсах будут использованы в процессе тестирования с целью анализа показателей динамики производства за определённый период времени, а также для оценки уровня волатильности количественных показателей используемой рабочей силы. Результат проведённого анализа, в свою очередь, позволит осуществить экстраполяцию значений временного ряда.

Сравнение результатов экстраполяции при использовании различных методов будет проводиться по первой номенклатурной позиции.

Результат прогнозирования потребности в трудовых ресурсах методом наименьших квадратов представлен на рисунке 3.14.



Диапазон анализа: 01.01.2009 — 31.12.2017

Метод прогнозирования: Метод наименьших квадратов

Глубина прогноза: 1

Прогноз

Наименование	Ед. измерения	Прогноз производства	Прогноз труд. ресурсов	Дефицит труд. ресурсов	Ср. ошибка прогноза
Щит осветительный ЩО	шт.	91	122	6	0,27
Щит силовой ЩС	шт.	100	96	6	1,32
Щит автоматики и управления ЩАУ	шт.	80	115	2	3,71
Вводно-распределительное устройство ВРУ	шт.	85	107	0	5,67
Пункт распределительный ПР	шт.	83	101	4	5,14
Щит этажный ЩЭ	шт.	119	82	7	7,46
Прибор РС 1616	шт.	115	102	8	8,06
Пульт диспетчерской сигнализации ПДС	шт.	89	94	0	4,05
Пульт выносной сигнализации ПВС	шт.	116	73	3	9,16
Труба дымовая	шт.	115	105	9	4,35

Рис. 3.14. Результат экстраполяции методом наименьших квадратов

Результат экстраполяции методом скользящей средней представлен на рисунке 3.15.

Наименование	Ед. измерения	Прогноз производства	Прогноз труд. ресурсов	Дефицит труд. ресурсов	Ср. ошибка прогноза
Щит осветительный ЩО	шт.	104	126	3	6,47
Щит силовой ЩС	шт.	73	125	5	0,95
Щит автоматики и управления ЩАУ	шт.	86	93	0	0,43
Вводно-распределительное устройство ВРУ	шт.	92	112	9	5,05
Пункт распределительный ПР	шт.	111	86	2	7,47
Щит этажный ЩЭ	шт.	91	105	4	9,36
Прибор РС 1616	шт.	85	100	1	5,98
Пульт диспетчерской сигнализации ПДС	шт.	81	102	7	8,88
Пульт выносной сигнализации ПВС	шт.	115	78	4	4,78
Труба дымовая	шт.	84	82	2	9,49

Рис. 3.15. Результат экстраполяции методом скользящей средней

Результат прогнозирования методом экспоненциального сглаживания представлен на рисунке 3.16.

Наименование	Ед. измерения	Прогноз производства	Прогноз труд. ресурсов	Дефицит труд. ресурсов	Ср. ошибка прогноза
Щит осветительный ЩО	шт.	70	105	1	2,72
Щит силовой ЩС	шт.	98	87	8	0,87
Щит автоматики и управления ЩАУ	шт.	96	111	0	9,22
Вводно-распределительное устройство ВРУ	шт.	117	77	8	2,14
Пункт распределительный ПР	шт.	95	110	7	5,53
Щит этажный ЩЭ	шт.	109	83	6	9,07
Прибор РС 1616	шт.	93	88	2	6,92
Пульт диспетчерской сигнализации ПДС	шт.	106	85	7	4,70
Пульт выносной сигнализации ПВС	шт.	80	129	7	6,21
Труба дымовая	шт.	77	90	2	7,33

Рис. 3.16. Результат экстраполяции методом экспоненциального сглаживания

Сравнение значений среднеквадратической ошибки прогноза для первой номенклатурной позиции (Щит осветительный ЩО), полученных в результате экстраполяции различными методами, позволяет однозначно утверждать, что в данном случае при прогнозировании трудовых ресурсов целесообразно применение метода наименьших квадратов, поскольку величина ошибки прогноза при этом будет минимальной.

ЗАКЛЮЧЕНИЕ

В результате комплексного анализа подсистемы учёта трудовых ресурсов и заработной платы ИД ООО «Монтажавтоматика» была выявлена возможность качественного усовершенствования данного программного компонента путём добавления модуля планирования трудовых ресурсов. Поскольку необходимым элементом процесса планирования является прогнозирование различных количественных показателей, вопрос выбора оптимального метода экстраполяции приобретает первостепенное значение.

Исследование существующих методов прогнозирования позволило сделать вывод, что в случае экстраполяции фактических значений временного ряда целесообразно применение следующих трёх методов: метод наименьших квадратов, метод экспоненциального сглаживания, метод скользящей средней. Реализация перечисленных методов непосредственно включена в структуру программного обеспечения по планированию потребностей производства в трудовых ресурсах.

Данные, полученные в ходе тестовых испытаний разработанного приложения, позволяют утверждать, что в случае применения того или иного метода экстраполяции точность прогнозирования при различных расчётных значениях может сильно варьироваться. Результат тестирования, таким образом, указывает на тот факт, что метод экстраполяции, обеспечивающий минимальную погрешность прогнозного значения, необходимо выбирать эмпирическим путём.

Итак, решение задачи планирования трудовых ресурсов основано на выполнении следующих шагов: обзор текущего состояния информационной системы, анализ модуля управления трудовыми ресурсами, постановка и анализ задачи планирования трудовых ресурсов; функциональный анализ приложения, проектирование архитектуры приложения и описание средств программирования; реализация уровня доступа к данным, уровня бизнес-логики и уровня представления, тестирование и анализ результатов.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. *Ерохина, Р.И. Анализ и моделирование трудовых показателей на предприятии / Р.И. Ерохина. – Минск: Новое издание, 2009. – 322 с.*
2. *Шишкин, А.К. Учет, анализ, аудит на предприятии: учеб. пособие / А.К. Шишкин, В.А. Микрюков, И.Д. Дышкант. – Москва: ЮНИТИ, 2010. – 415 с.*
3. *Савицкая, Г.В. Анализ хозяйственной деятельности предприятия: учебник / Г.В. Савицкая. – Москва: ИНФРА-М, 2011. – 595 с.*
4. *Козлов, А.А. Управление трудовыми ресурсами промышленных предприятий при переходе к рынку / А.А. Козлов. – Минск, 2010. – 512с.*
5. *Бузырев, В.В. Планирование на предприятии / В.В. Бузырев. – Москва: Академия, 2009. – 451 с.*
6. *Наумов, В.Н. Математическая модель сглаживания временного ряда при решении задач прогнозирования / В.Н. Наумов, С.В. Наумов. – Управленческое консультирование, 2011. –168 с.*
7. *Прокуратова, О.Н. Метод наименьших квадратов / О.Н. Прокуратова, С.А. Неклюдова. – Наука и современность, 2014. – 207 с.*
8. *Шевченко, И.В. Некоторые модели анализа и прогнозирования временных рядов / И.В. Шевченко. – Системная информатика, 2013. – 40 с.*
9. *Прогнозирование и планирование в условиях рынка / В.И. Бархатов, А.А. Горшков, Ю.Ш. Капкаев, М.А. Усачев. – Челябинск: ЮУрГУ, 2012.– 140 с.*
10. *Тихонов, Э.Е. Методы прогнозирования в условиях рынка: учеб. пособие / Э.Е. Тихонов – Невинномысск: Северо-Кавказский ГТУ, 2012.– 221 с.*
11. *Емельянова, Н.З. Проектирование информационных систем / Н.З. Емельянова, Т.Л. Партыка, И.И. Попов. – Москва: Форум, 2009.– 432 с.*

12. *Мещеряков, С.В. Технологии создания информационных систем / С.В. Мещеряков, В.М Иванов. – Москва: Политехника, 2005. – 312 с.*
13. *Иванова, Г.С. Технологии программирования / Г.С. Иванова. – Москва: Изд-во МГТУ им. Баумана, 2002. – 446 с.*
14. *Коберн, А. Современные функциональные требования к системам / А. Коберн; пер. с англ. – Москва: ЛОРИ, 2002. – 350 с.*
15. *Леффингуэлл Д. Принципы работы с требованиями к программному обеспечению. Унифицированный подход / Д. Леффингуэлл, Д. Уидриг; пер. с англ. – Москва: Вильямс, 2002. – 574 с.*
16. *Иванова, Г.С. Объектно-ориентированное программирование / Г.С. Иванова, Т. Н. Ничушкина, Е. К. Пугачев. – Москва: Изд-во МГТУ им. Баумана, 2001. – 741 с.*
17. *Магдануров Г. ASP.NET MVC Framework / Г. Магдануров, В. Юнев. – Москва: БХВ-Петербург, 2010. – 320 с.*
18. *Эспозито Д. Программирование на основе Microsoft ASP.NET MVC / Д. Эспозито. – Москва: Русская Редакция, 2012. - 464 с.*
19. *Фримен А. Платформа ASP.NET MVC 5 для профессионалов. А. Фримен. – Москва: Вильямс, 2013. – 688 с.*
20. *Котляров, В.П. Основы тестирования программного обеспечения / Котляров В.П., Т.В. Коликова. – Москва: Бином, 2006. – 346 с.*

ПРИЛОЖЕНИЕ

```
public class LeastSquareMethod : IForecast
{
    private double[] samples;
    private int depth;

    public LeastSquareMethod(double[] samples, int depth)
    {
        this.samples = samples;
        this.depth = depth;
    }

    public double[] GetForecast()
    {
        double sumX = 0, sumY = 0, sumXY = 0, sumXX = 0;
        for (int i = 0; i < samples.Length; i++)
        {
            sumX += i + 1;
            sumY += samples[i];
            sumXY += samples[i] * (i + 1);
            sumXX += Math.Pow(i + 1, 2);
        }

        double a = sumXY - (sumX * sumY) / samples.Length;
        a = a / (sumXX - Math.Pow(sumX, 2) / samples.Length);
        double b = sumY / samples.Length - a * sumX / samples.Length;

        double[] forecast = new double[depth];
        for (int i = 0; i < depth; i++)
            forecast[i] = a * (samples.Length + i + 1) + b;

        return forecast;
    }
}
```

```
public class ExponentialSmoothingMethod : IForecast
{
    private double[] samples;

    public ExponentialSmoothingMethod(double[] samples)
    {
        this.samples = samples;
    }

    public double[] GetForecast()
    {
        double parameter = 2 / ((double) samples.Length + 1);
        double[] weightedValues = new double[samples.Length];
        weightedValues[0] = 0;

        for (int i = 0; i < samples.Length; i++)
            weightedValues[0] += samples[i];
        weightedValues[0] /= samples.Length;

        for (int i = 1; i < samples.Length; i++)
        weightedValues[i] = parameter * samples[i - 1] + (1 - parameter) * weightedValues[i - 1];

        double[] forecast = { parameter * samples[samples.Length - 1] + (1 - parameter) *
        weightedValues[samples.Length - 1] };

        return forecast;
    }
}

public class MovingAverageMethod : IForecast
{
    private double[] samples;
    private int depth;
```

```
public MovingAverageMethod(double[] samples, int depth)
{
   this.samples = samples;
   this.depth = depth;
}

public double[] GetForecast()
{
   const int interval = 3;
   double[] smoothedSamples = new double[samples.Length];

   int index = (interval - 1) / 2;
   for (int i = index; i < samples.Length - index; i++)
      smoothedSamples[i] = (double)(samples[i - 1] + samples[i] + samples[i + 1]) / interval;

   double[] newSamples = new double[samples.Length + depth];
   double[] newSmoothedSamples = new double[samples.Length + depth];

   for (int i = 0; i < samples.Length; i++)
   {
      newSamples[i] = samples[i];
      newSmoothedSamples[i] = smoothedSamples[i];
   }

   double[] forecast = new double[depth];
   for (int i = 0; i < forecast.Length; i++)
   {
      forecast[i] = newSmoothedSamples[samples.Length + i - 2];
      forecast[i] += (newSamples[samples.Length + i - 1] –
      newSamples[samples.Length + i - 2]) / interval;

newSamples[samples.Length + i] = forecast[i];
newSmoothedSamples[samples.Length + i - 1] = newSamples[samples.Length + i - 2];
newSmoothedSamples[samples.Length + i - 1] += newSamples[samples.Length + i - 1];
newSmoothedSamples[samples.Length + i - 1] += newSamples[samples.Length + i];
```

```
        newSmoothedSamples[samples.Length + i - 1] /= interval;
            }

            return forecast;
        }
    }

public abstract class ForecastService : IForecastService
{
        protected IUnitOfWork Database { get; set; }

        public ForecastService(IUnitOfWork database)
        {
            Database = database;
        }

        protected int GetSampleLength(DateTime begin, DateTime end)
        {
            int length = 0;
            for (int i = begin.Year; i <= end.Year; i++)
                length++;

            return length;
        }

        public IEnumerable<OrderProductDTO> GetOrderProducts()
        {
            var mapper = new MapperConfiguration(config => config.CreateMap<OrderProduct,
             OrderProductDTO>()).CreateMapper();

           IEnumerable<OrderProductDTO> products = mapper.Map<IEnumerable<OrderProduct>,
            List<OrderProductDTO>>(Database.OrderProducts.GetAll());

            return products;
        }
```

```
public IEnumerable<ProductDTO> GetProducts()
{
    var mapper = new MapperConfiguration(config => config.CreateMap<Product,
    ProductDTO>()).CreateMapper();

     IEnumerable<ProductDTO> products = mapper.Map<IEnumerable<Product>,
     List<ProductDTO>>(Database.Products.GetAll());

  return products;
}

public double[] GetProductSample(ProductDTO product, DateTime begin, DateTime end)
{
  IEnumerable<OrderProductDTO> orderProdcuts = GetOrderProducts();
  int length = GetSampleLength(begin, end);
  double[] samples = new double[length];

  for (int i = 0; i < samples.Length; i++)
     samples[i] = 0;

  int counter = 0;
  for (int i = begin.Year; i <= end.Year; i++)
  {
     foreach (OrderProductDTO orderProduct in orderProdcuts)
     {
        if ((orderProduct.Order.Date.Year == i) && (orderProduct.ProductId ==
         product.Id))
           samples[counter] += orderProduct.Number;
     }
     counter++;
  }

  return samples;
}
```

```
        public double[] GetEmployeeSample(ProductDTO product, DateTime begin, DateTime end)
        {
            int length = GetSampleLength(begin, end);
            double[] samples = new double[length];

            Random random = new Random();
            for (int i = 0; i < samples.Length; i++)
                samples[i] = random.Next(30, 100);

            return samples;
        }

        public abstract double[,] Forecasting(DateTime begin, DateTime end, int depth);
    }

public class LeastSquareForecastService : ForecastService
{
        public LeastSquareForecastService(IUnitOfWork database) : base(database) { }

        public override double[,] Forecasting(DateTime begin, DateTime end, int depth)
        {
            IEnumerable<ProductDTO> products = GetProducts();
            int length = GetSampleLength(begin, end);
            double[,] result = new double[3, products.Count()];

            IForecastType forecast = null;
            double[] productSamples = null;
            double[] employeeSamples = null;
            double[] forecastProductResult = null;
            double[] forecastEmployeeResult = null;
            double[] forecastProductivityResult = null;
            double[] forecastEmployeeRequirements = null;

            int counter = 0;
            foreach (ProductDTO product in products)
```

```
{
   productSamples = GetProductSample(product, begin, end);
   forecast = new ForecastType(productSamples, depth);
   forecastProductResult = forecast.LeastSqueraForecast.GetForecast();

   employeeSamples = GetEmployeeSample(product, begin, end);
   double[] productivitySamples = new double[employeeSamples.Length];

   for (int i = 0; i < productivitySamples.Length; i++)
   {
      if (employeeSamples[i] == 0) productivitySamples[i] = 0;
      else productivitySamples[i] = productSamples[i] / employeeSamples[i];
   }

   forecast = new ForecastType(productivitySamples, depth);
   forecastProductivityResult = forecast.LeastSqueraForecast.GetForecast();

   forecastEmployeeResult = new double[forecastProductivityResult.Length];
   forecastEmployeeRequirements = new double[forecastEmployeeResult.Length];

   for (int i = 0; i < forecastEmployeeResult.Length; i++)
   {
      if (forecastProductivityResult[i] == 0)
      {
         forecastEmployeeResult[i] = 0;
         forecastEmployeeRequirements[i] = 0;
      }
      else
      {
           forecastEmployeeResult[i] = forecastProductResult[i] /
           forecastProductivityResult[i];
         if (forecastEmployeeResult[i] > employeeSamples[employeeSamples.Length - 1])
           forecastEmployeeRequirements[i] = forecastEmployeeResult[i] –
            employeeSamples[employeeSamples.Length - 1];
         else forecastEmployeeRequirements[i] = 0;
```

```
            }
        }

        result[0, counter] = forecastProductResult[0];
        result[1, counter] = forecastEmployeeResult[0];
        result[2, counter] = forecastEmployeeRequirements[0];
        counter++;
    }

    return result;
  }
}

public class ExponentialSmoothingForecastService : ForecastService
{
    public ExponentialSmoothingForecastService(IUnitOfWork database): base(database) { }

    public override double[,] Forecasting(DateTime begin, DateTime end, int depth)
    {
        IEnumerable<ProductDTO> products = GetProducts();
        double[,] result = new double[products.Count(), depth];

        IForecastType forecast = null;
        double[] samples = null;
        double[] temp = null;
        int counter = 0;

        foreach (ProductDTO product in products)
        {
            samples = GetProductSample(product, begin, end);
            forecast = new ForecastType(samples);
            temp = forecast.ExponentialSmoothingForecast.GetForecast();

            for (int j = 0; j < depth; j++)
                result[counter, j] = temp[j];
```

```
            counter++;
        }

        return result;
    }
}

class MovingAverageForecasstService : ForecastService
{
    public MovingAverageForecasstService(IUnitOfWork database) : base(database) { }

    public override double[,] Forecasting(DateTime begin, DateTime end, int depth)
    {
        IEnumerable<ProductDTO> products = GetProducts();
        double[,] result = new double[products.Count(), depth];

        IForecastType forecast = null;
        double[] samples = null;
        double[] temp = null;
        int counter = 0;

        foreach (ProductDTO product in products)
        {
            samples = GetProductSample(product, begin, end);
            forecast = new ForecastType(samples, depth);
            temp = forecast.MovingAverageForecast.GetForecast();

            for (int j = 0; j < depth; j++)
                result[counter, j] = temp[j];
            counter++;
        }

        return result;
    }
}
```

```
public class HomeController : Controller
{
    private IForecastService forecastService;

    public HomeController(IForecastService forecastService)
    {
        this.forecastService = forecastService;
    }

    public ActionResult Index()
    {
        IEnumerable<ProductDTO> productsDTO = forecastService.GetProducts();

        var mapper = new MapperConfiguration(config => config.CreateMap<ProductDTO,
        ProductViewModel>()).CreateMapper();
       var products = mapper.Map<IEnumerable<ProductDTO>,
        List<ProductViewModel>>(productsDTO);

        int length = end.Date.Year - begin.Date.Year + 1;

        double[] productSamples = new double[length];
        double[] employeeSamples = new double[length];

        double[,] productSamplesResult = new double[productsDTO.Count(),
         productSamples.Length];
        double[,] employeeSamplesResult = new double[productsDTO.Count(),
        employeeSamples.Length];

        foreach (ProductDTO product in productsDTO)
        {
            productSamples = forecastService.GetProductSample(product, begin, end);
            employeeSamples = forecastService.GetEmployeeSample(product, begin, end);

             ViewBag.productSamplesResult = productSamplesResult;
```

```
            ViewBag.employeeSamplesResult = employeeSamplesResult;
            ViewBag.begin = begin;
             ViewBag.end = end;

        return View(products);
    }

    [HttpPost]
    public ActionResult Index(DateTime begin, DateTime end, int method, int depth)
    {
        IEnumerable<ProductDTO> productsDTO = forecastService.GetProducts();
        var mapper = new MapperConfiguration(config => config.CreateMap<ProductDTO,
         ProductViewModel>()).CreateMapper();
        var products = mapper.Map<IEnumerable<ProductDTO>,
         List<ProductViewModel>>(productsDTO);

        return View("~/Views/Home/Forecast.cshtml", products);
    }

    public ActionResult Forecast(IEnumerable<ProductViewModel> products)
    {
        return View(products);
    }
}
```

Выпускная квалификационная работа выполнена мной совершенно самостоятельно. Все использованные в работе материалы и концепции из опубликованной научной литературы и других источников имеют ссылки на них.

«___» _____ _____ г.

_____ _____

(подпись) (Ф.И.О.)