РЕФЕРАТ

Отчет 84 страниц, 43 рисунка, 9 таблиц, 26 источников.

ТЕХНИКО-ЭКОНОМИЧЕСКИЙ АНАЛИЗ, ИНФОРМАЦИОННАЯ СИСТЕМА, JAVASCRIPT, ANGULAR, TYPESSCRIPT, MONGODB, ВЕБ-ПРИЛОЖЕНИЕ

Целью выпускной квалификационной работы является проектирование информационной системы учёта работ по госзаказам для Рубцовского филиала АО «НПК Уралвагонзавод».

Объектом выпускной квалификационной работы является Рубцовский филиал АО «НПК Уралвагонзавод».

Предметом выпускной квалификационной работы является процесс учёта деятельности научно-производственного предприятия по исполнению государственных заказов на выполнение различных видов работ.

Методы и технологии решения поставленных задач: опрос исполнителей на рабочих местах, системный анализ, функционально-ориентированная и объектно-ориентированная методология описания систем, оригинальное проектирование, web-технологии, RAD-технологии.

Результатом работы является проект информационной системы, которая позволяет автоматизировать учётные функции в деятельности научно-производственного предприятия по исполнению госзаказов.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Для эффективного управления деятельностью предприятия необходимо учитывать множество факторов и показателей. Учётные процессы являются базовыми для осуществления всех последующих действий по управлению – анализу, оперативному реагированию, планированию на новый период. Особые требования предъявляются при деятельности предприятий по исполнению государственных заказов. Они связаны с обеспечением сроков выполнения, качеством продукции и услуг, оперативностью взаимосвязи с заказчиком и другими важными составляющими.

Актуальность выпускной квалификационной работы обусловлена необходимостью создания и использования информационной системы, призванной снизить трудозатраты на документооборот и принятие управленческих решений за счёт всестороннего учёта деятельности научно-производственного предприятия при выполнении государственных заказов.

Объектом выпускной квалификационной работы является Рубцовский филиал АО «НПК Уралвагонзавод».

Предмет исследования – процесс учёта деятельности научно-производственного предприятия по исполнению государственных заказов на выполнение различных видов работ.

Целью выпускной квалификационной работы является проектирование информационной системы учёта работ по госзаказам для Рубцовского филиала АО «НПК Уралвагонзавод».

Для достижения поставленной цели были обозначены следующие задачи:

– проанализировать управленческие процессы в деятельности предприятия при исполнении госзаказов;

– выявить недостатки в существующих реализациях информационных процессов;

– разработать функциональную архитектуру проектируемой системы;

– выработать и реализовать проектные решения по обеспечивающим подсистемам;

– оценить экономическую эффективность от внедрения информационной системы.

При написании выпускной квалификационной работы использовались методы системного функционального анализа, оригинального проектирования, быстрой разработки приложений (RAD).

В качестве средств реализации ИС использовались: инструмент графического описания систем MS Visio, инструмент визуального моделирования структуры базы данных DbSchema, интегрированная среда разработки WebStorm.

Ресурсами информации, используемой в данной работе, является современная учебная и научная литература и интернет-источники по проектированию систем и разработке программных приложений, нормативные и нормативно-справочные документы организации, федеральные нормативные документы, стандарты.

Результатом работы является проект информационной системы, которая позволяет автоматизировать учётные функции в деятельности научно-производственного предприятия по исполнению госзаказов.

1 Аналитическая часть

1.1 Технико-экономическая характеристика предметной области

Рубцовский филиал АО Научно-производственная корпорация «Уралвагонзавод» (сокращенно – Рубцовский филиал АО «НПК Уралвагонзавод») – российское предприятие, проектирующее и выпускающее технику военного и гражданского назначения. Адрес (место нахождения) Учреждения: 658225, Россия, Алтайский край, г. Рубцовск, пр-т Ленина 204. Входит в состав Научно-производственной корпорации «Уралвагонзавод». В настоящее время предприятием определены следующие основные направления деятельности: проведение НИОКР в интересах обороны и безопасности страны, разработка, мелкосерийное производство специальных военных гусеничных машин (в интересах МО РФ: ГАБТУ, ГРАУ, УНИВ, УРЭБ ГШ), утилизация военных гусеничных машин, проведение научно-исследовательских, опытно-конструкторских работ с целью создания и производства образцов новой техники и технологий, проведение НИОКР по гусеничным машинам народно-хозяйственного назначения, автотракторной, сельскохозяйственной технике и ТНП широкой номенклатуры, а также их производство и сбыт, гарантийное, послегарантийное обслуживание продукции, выпущенной предприятием. Рубцовский филиал является единственным предприятием корпорации, производящим боевые разведывательные машины для сухопутных войск. Одним из таких изделий является боевая разведывательная машина БРМ-3К, которая предназначена для ведения войсковой разведки в любое время года и суток в условиях ограниченной оптической видимости в составе разведывательных подразделений.

Компьютеры на предприятии оснащены операционной системой

Windows и соответствующим программным обеспечением для работы каждого сотрудника.

Организационно-штатная структура Рубцовского филиала АО «НПК Уралвагонзавод» представлена на рисунке 1.1.

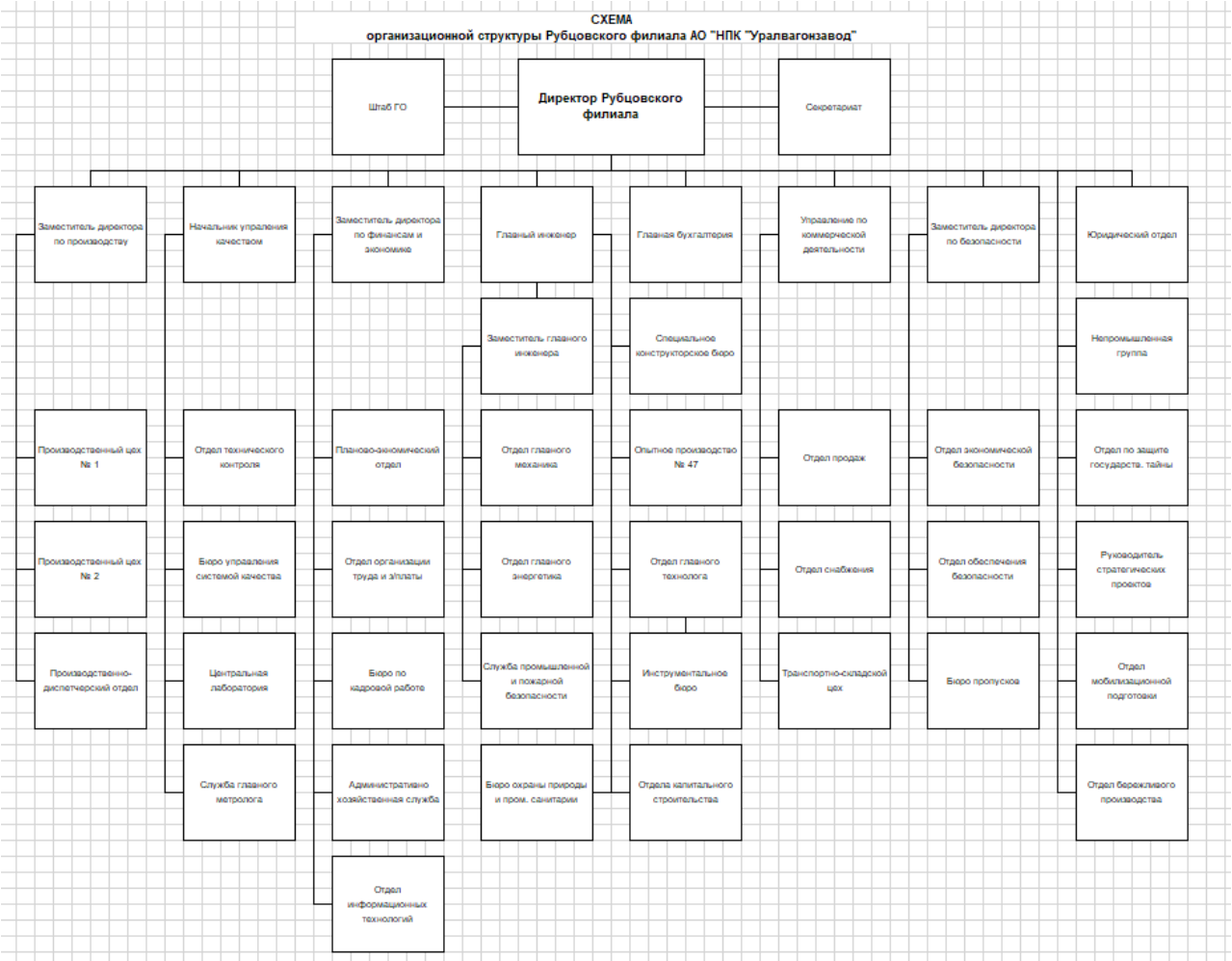


Рисунок 1.1 – Организационная структура Рубцовского филиала АО «НПК Уралвагонзавод»

Производственный отдел насчитывает 800 человек. Отдел вовлечён на прямую в утилизацию, производство, ремонт техники.

Бухгалтерский отдел ведёт: финансовую деятельность предприятия, осуществляет начисление заработной платы, заключает контракты на работу.

Отдел кадров занимается: учётом персонала, принятием на работу, увольнениями.

Отдел информационных технологий занимается: сопровождением ИС на предприятии, устранением возникших неполадок.

Хозяйственный отдел обеспечивает сохранность материально-технического обеспечения предприятия.

1.2 Анализ функционирования объекта исследования

Для описания и анализа деятельности объекта исследования воспользуемся структурно-функциональным подходом на основе методологии SADT [1]. Данный подход реализуется через три методологии моделирования: функциональное моделирование (IDEF0), описание бизнес-процессов (IDEF3), диаграммы потоков данных (DFD).

IDEF0 (функциональное моделирование) – это методология функционального моделирования и графическая нотация, предназначенная для формализации и описания бизнес-процессов. Отличительной особенностью IDEF0 является ее акцент на соподчиненность объектов. IDEF0 рассматриваются логические отношения между работами, а не их временная последовательность.

Функциональное моделирование базируется на четырех основных понятия: функциональный блок, интерфейсная дуга, декомпозиция, глоссарий.

DFD (Data Flow Diagramming) – диаграммы потоков данных представляют собой иерархию процессов, связанных потоками данных. На диаграммах показано, как каждый процесс обрабатывает информацию, как процессы связаны друг с другом, как работает сама система и как она обрабатывает входящие данные.

IDEF3 (Integrated DEFinition for process Description Capture Method) – предлагает структурный метод описания процессов. Модель описывается как

упорядоченная последовательность событий. Метод IDEF3 хорошо подходит для сбора данных. Данная методика не имеет строгих синтаксических и семантических ограничений. Очень часто IDEF3 используется как метод, дополняющий IDEF0. Каждый функциональный блок IDEF0 может быть представлен как отдельный процесс. Объектом анализа является процесс исполнения государственных заказов на АО «НПК Уралвагонзавод», по которой построена диаграмма IDEF0 «Как есть», представленная на рисунке 1.2.

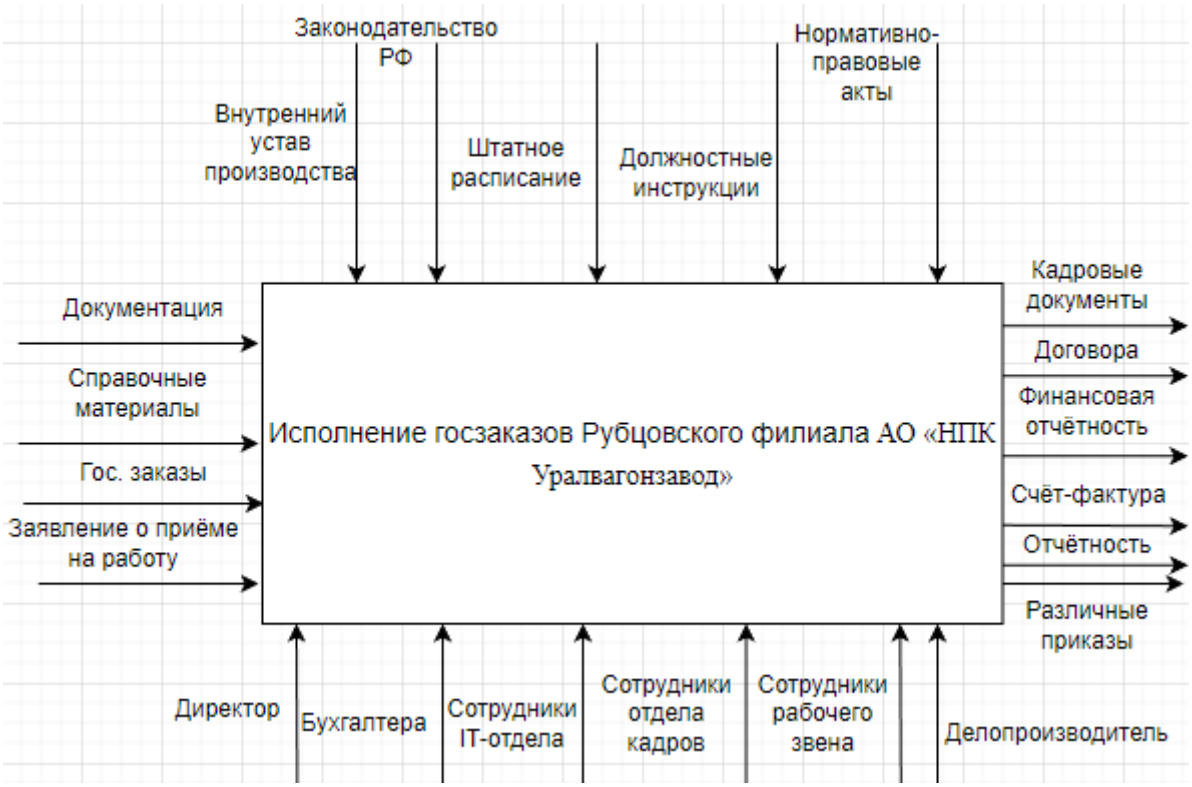


Рисунок 1.2 – Диаграмма IDEF0 AS–IS «Исполнение госзаказов Рубцовского филиала АО «НПК Уралвагонзавод»»

Входными данными являются:

- документация;
- справочные материалы;
- госзаказы;
- заявление о приёме на работу.

Выходными данными модели являются:

- кадровые документы;
- договора;
- финансовая отчётность;
- счёт-фактура;
- отчётность;
- различные приказы.

Управлением данной модели являются:

- внутренний устав производства;
- законодательство РФ;
- штатное расписание;
- должностные инструкции;
- нормативно-правовые акты.

Механизм:

- директор;
- бухгалтеры;
- сотрудники IT-отдела;
- сотрудник рабочего звена;
- делопроизводитель;
- сотрудники отдела кадров.

На рисунке 1.3 представлена декомпозиция диаграммы IDEF0 AS-IS «Исполнение госзаказов АО «НПК Уралвагонзавод»».

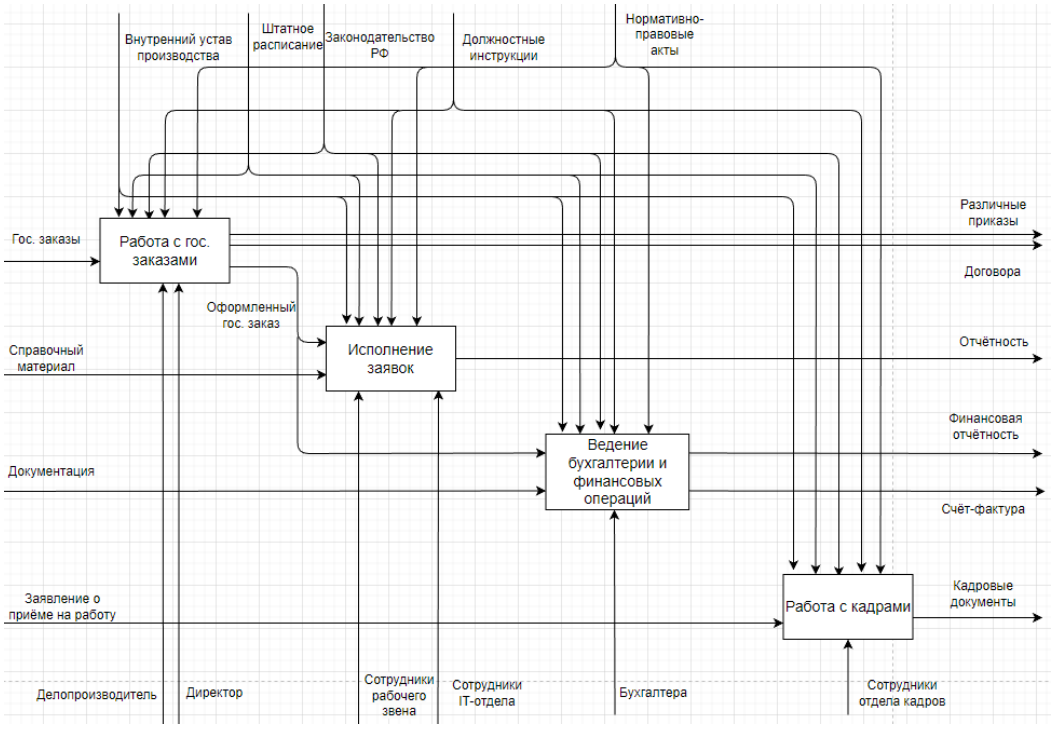


Рисунок 1.3 – Декомпозиция диаграммы IDEF0 AS-IS «Исполнение госзаказов Рубцовского филиала АО «НПК Уралвагонзавод»»

Данная декомпозиция состоит из четырех блоков:

– работа с госзаказами;

– исполнение заявок;

– ведение бухгалтерии и финансовых операций;

– работа с кадрами.

Из диаграмм видно, что абсолютно вся информация фиксируется и обрабатывается вручную, а также весь персонал предприятия работает с огромным количеством бумажных документов, что, особенно хорошо можно разглядеть на декомпозиции блока «Учёт информации о госзаказах», представленной на рисунке 1.4.

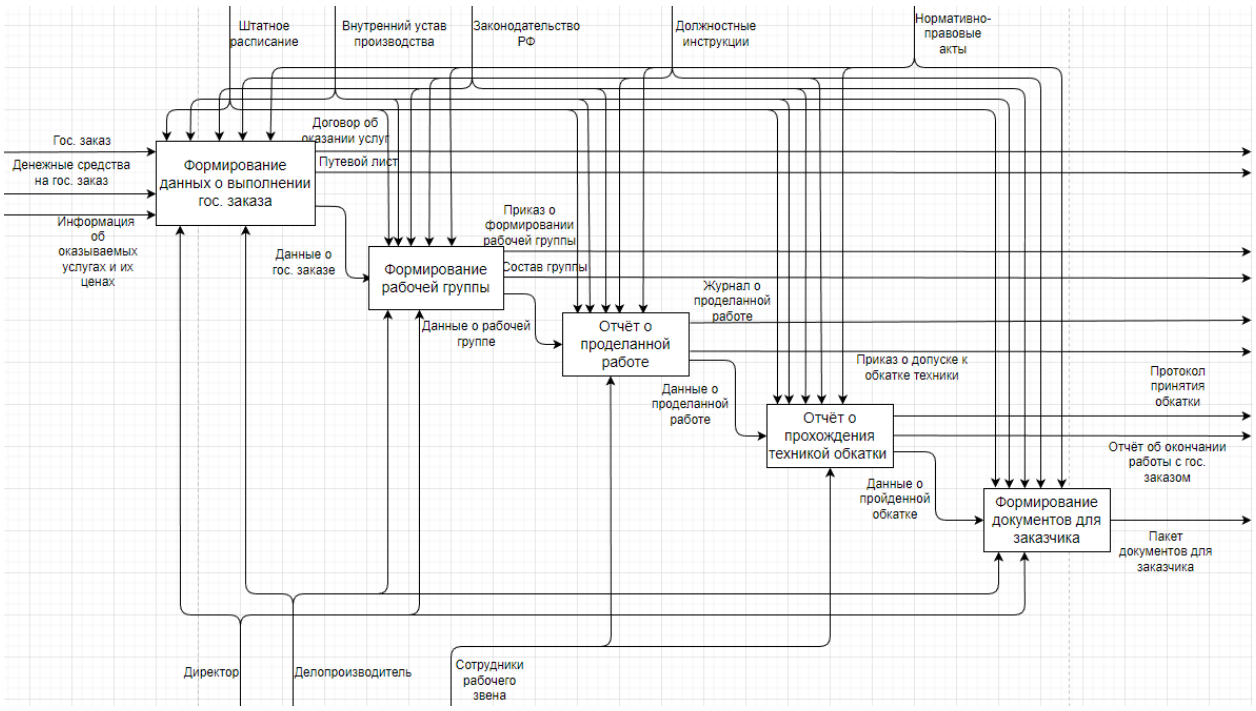


Рисунок 1.4 – Детализированная диаграмма IDEF0 AS-IS блока «Учёт информации о госзаказах»

На данной декомпозиции отчетливо видно, что директор и делопроизводитель тратят огромное количество времени на контроль всего процесса работы с госзаказами и на подготовку значительного числа документации.

1.3 Определение цели и задач проектирования

Динамичный рост работы производства и, как следствие, рост поступления новых госзаказов, не подкрепленная структурными изменениями и достаточным количество персонала, может привести к аварийному режиму работы производства, и, как следствие, все вышеперечисленные недостатки приводят к значительному снижению качества услуг, и некоторый отток персонала. В данное время в организации наблюдается проблема в сфере информатизации, которая заключается в следующем:

– ведение документооборота осуществляется вручную, путём ведения большого количества рабочих журналов и реестров в бумажном виде;

– отсутствие отдельного места для хранения документов, поэтому они располагаются в различных удаленных местах офиса, что усложняет поиск необходимых документов;

– отсутствие возможности мобильно и оперативно устранять нарушения в документах;

– при подачи необходимых документов начальнику рабочей группы, делопроизводителю приходится заполнять и распечатывать документы вручную на каждого работника;

– отсутствие возможности работникам производства просматривать график и расписание работ на рабочем месте.

Целью создания информационной системы является автоматизация процесса учёта всех составляющих деятельности по исполнению на предприятии госзаказов.

Разработанная ИС позволит:

– повысить эффективность работы сотрудников, за счет перевода работы с документами в систему электронного оборота;

– вести более полный учёт деятельности производства;

– сократить время на обработку и получение оперативных данных, а также получения первичной информации в электронном виде;

– повысить степень достоверности обрабатываемой информации, ее защищенности;

– своевременно производить необходимые отчеты;

– устранить необходимость ручного заполнения необходимой документации для подачи начальнику рабочей группы;

– осуществить автоматизацию процесса внесения и изменения данных в определенных журналах и реестрах;

– исключить появление ошибок.

Назначение информационной системы:

– предоставление клиенту полной информации о деятельности производства;

– ведение карт каждого госзаказа;

– учёт и печать документов, связанных с приемом, работой и формированием отчётной ведомости по госзаказу;

– формирование и печать комплекта документов для предоставления начальнику рабочей группы;

– отслеживание графика работника;

– формирование полной карты работы и печать журналов учёта работы;

– формирование различной документации и отчетности.

1.4 Обзор и анализ существующих разработок, выбор технологии проектирования

1.4.1 Обзор и анализ существующих разработок

Огромный объем документооборота производств является одним из немногих факторов, который влияет на стабильную работу производства.

Самым популярным решением данной проблемы является программа 1С-ЭДО, созданный в виде конфигурации к программе 1С: Предприятие версии 8, и приобретается на сайте программы. Данная программа представлена на рисунке 1.5.

К возможностям программы можно отнести:

– формирование и печать документов, связанных с приемом, работой и формированием отчётной ведомости по госзаказу;

– печать образцов документов;

– обмен юридически значимыми документами между организациями;

– ведение личных карточек работников;

– контролирование работы рабочих групп;

– отслеживание работы над госзаказом;

– формирование полной карты работы над госзаказом и печать журналов проделанной работы;

– ведение учёта оплаты за работу;

– формирование графиков путевых листов;

– создание собственных шаблонов документов и использование их в дальнейшей работе в программе;

– создание и пополнения своего архива;

– формирование отчетов.

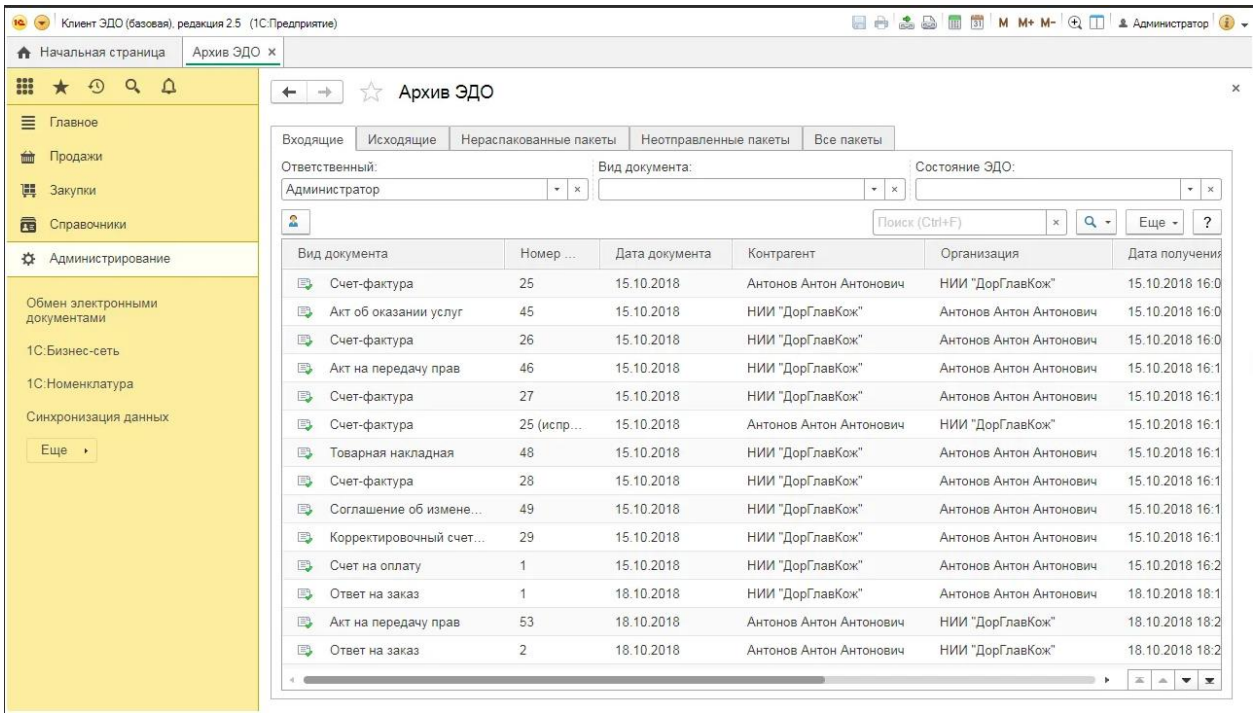


Рисунок 1.5 – Программа 1С-ЭДО

Как видно из возможностей программы, она ведет полный учёт работы предприятия. Всегда можно посмотреть всю историю работы предприятия,

сформировать в автоматическом режиме различные документы, а также сформировать список незаконченных госзаказов.

Но, она не может решить одну из главных проблем предприятия, а именно отсутствие возможности ведения учёта деятельности предприятия, не находясь непосредственно в офисе и полной независимости от компьютера с установленной программой.

Еще одна программа для ведения документооборота: «Электронный документооборот КонтурДиадок», представленная на рисунке 1.6.

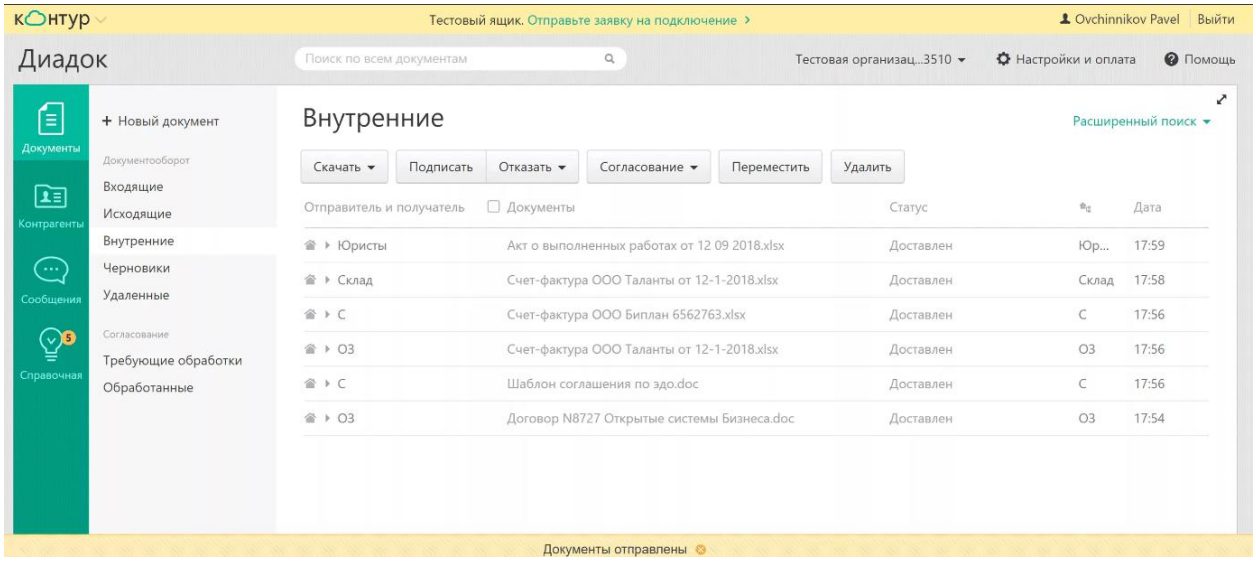


Рисунок 1.6 – Программа «Электронный документооборот КонтурДиадок»

К возможностям программы можно отнести:

– формирование и печать документов, связанных с приемом, процессом и окончанием работы над госзаказом;

– печать образцов документов;

– формирование и печать комплекта документов для предоставления заказчику;

– отслеживание выполненных работ на каждый госзаказ;

– формирование полного расписания и печать журналов произведённых работ;

– создание собственных шаблонов документов и использование их в дальнейшей работе в программе;

– создание и пополнения своего архива;

– формирование отчетов;

– быстрый доступ к нормативным документам производства;

– сохранение баз данных.

В преимущества программы так же можно добавить поддержку кроссплатформенности, но платная лицензия и невозможность вести точный график проделанных работ не подходят для производства.

Одной из последних разработок в области документооборота производства является система EOS for SharePoint построенная на базе платформы Microsoft SharePoint, разработанная компанией «Электронные офисные системы», представленная на рисунке 1.7.

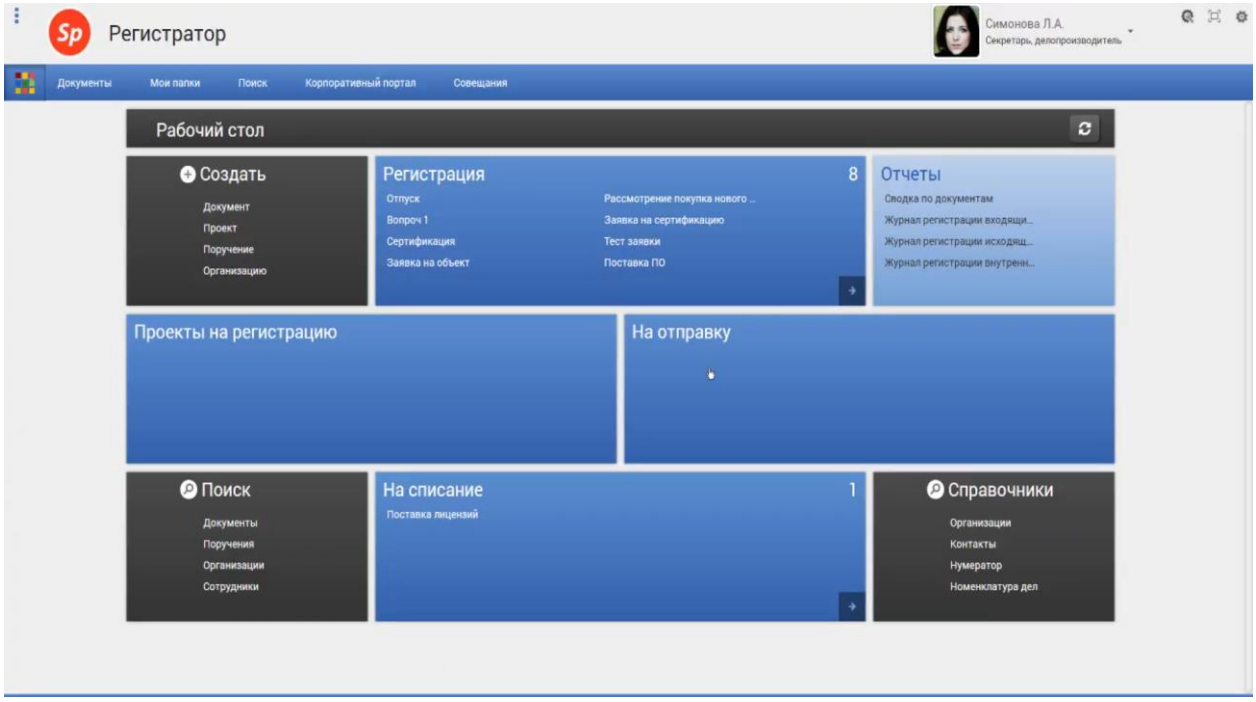


Рисунок 1.7 – Система хранения и обработки данных EOS for SharePoint

Ключевыми возможностями данной системы является:

– гибкий и мгновенный поиск по базе производства;

– полная интеграция с системой справочников;

− возможность отправлять документы на регистрацию и одобрение директора в электронном виде.

Данная система является самой новой и продвинутой среди всех представленных, но в данный момент приложение не поддерживает кроссплатформенность, а значит работники вне своего офиса не смогут работать и просматривать с различной документацией.

Данный фактор дает возможность поставить главную цель, а именно использовать технологию создания бесплатного веб-приложения.

1.4.2 Выбор технологии проектирования

На данный момент есть много фреймворков с помощью которых можно реализовать одностраничное приложение. У каждого есть свои плюсы и недостатки.

React – разработанная и используемая компаниями Facebook и Instagram.

JavaScript – библиотека с открытым исходным кодом для создания пользовательских интерфейсов. Большинство пользователей React считают его представлением в идеологии MVC. React был разработан для того, чтобы решить большую проблему: написание больших приложений с данными, которые меняются с течением времени.

С помощью этой технологии вы можете безопасно создавать интерактивные пользовательские интерфейсы. Разработчик может спроектировать простые представления для каждого состояния будущего веб-приложения, а React сможет эффективно и рационально обновлять, и перерисовывать только те компоненты, на которые повлияло изменение данных. Декларативные представления делают код более предсказуемым при выполнении и более удобным при отладке.

Несмотря на то, что это библиотека, а не платформа, она стоит за

пользовательским интерфейсом таких крупных порталов, как Facebook и Instagram, демонстрируя свою эффективность в динамических приложениях с высоким трафиком (пропускной способностью).

Vue.js – это прогрессивная структура для создания пользовательских интерфейсов. В отличие от монолитных фреймворков, Vue.js предназначен для постепенной реализации. Его ядро в первую очередь решает задачи уровня представления, что упрощает интеграцию с другими библиотеками и существующими проектами. С другой стороны, Vue.js также полностью подходит для создания сложных одностраничных приложений, если он используется в сочетании с современными инструментами и дополнительными библиотеками. Это может быть прочной основой для высокопроизводительных одностраничных приложений и выгодным решением для случаев, когда производительность важнее, чем хорошая организация кода или структура приложения.

Кроме того, различные разработки, ориентированные на среду Vue.js, довольно просто объединить в более сложные решения при создании новых проектов.

Разработчик фреймворка утверждает, что Vue.js является одной из самых быстрых фреймворков в целом. Существует солидное сообщество энтузиастов и сторонних проектов.

Angular – это открытая и свободная платформа для разработки веб-приложений, написанная на языке TypeScript, разрабатываемая командой из компании Google, а также сообществом разработчиков из различных компаний [2].

Еще один замечательный инструмент, предоставляемый Angular, – это то, как фреймворк управляет DOM (Document Objerct Model) или, другими словами, тегами HTML. Если во время разработки с использованием jQuery вам постоянно приходится вручную обновлять все данные на экране пользователя, то Angular выполняет всю работу за разработчика. Достаточно

указать, какие элементы прикреплены к каким объектам данных, и обновленная информация будет отображаться на экране при каждом изменении состояния. Angular использует объектный подход к дизайну. Это архитектура MVC (Model-View-Controller). Следует отметить, что одновременно с классическим подходом эта модель не совсем соответствует тому, как Angular работает с данными. [3].

Существует более точное определение его архитектуры – MVVM (Model-View-ViewModel).

Однако стоит отметить, что в обеих архитектурах обработка данных и их вывод пользователю разделены. Таким образом, выходные данные представляют собой более структурированный код, который легче поддерживать и легче управлять, а затем управлять им. В конце концов, разработка не останавливается после передачи продукта клиенту, она часто продолжается долго после этого в форме поддержки, добавления функциональности или исправления ошибок.

Следующий важный вопрос – какую систему управления базами данных лучше всего (наиболее удобно) использовать для разработки нашей информационной системы.

Существует множество моделей работы с данными, нас больше всего интересует реляционная модель системы управления баз данных (РСУБД) и как пример не реляционные базы, в нашем случае чаще всего встречающиеся под термином noSQL – или документно-ориентированная модель, так как SQL – (Structured Query Language) структурированный язык запросов. Значения данных типизированы, это могут быть числа, строки, даты, неструктурированные двоичные объекты (BLOB) и т.п. Тип данных контролируется системой, существенно, что благодаря математическим основаниям реляционной модели (теории множеств) исходные таблицы можно соединять и трансформировать в новые, более сложные.

В данный момент две основные документно-ориентированные базы

данных являются MongoDB и CouchDB [4].

CouchDB – документо-ориентированная система управления базами данных с открытым исходным кодом, не требующая описания схемы данных. Данный продукт написан на языке Erlang и распространяется свободно. Первый релиз состоялся в 2005 году.

Из ее особенностей можно выделить:

- поддержка ACID-семантики;
- автономный режим работы;
- распределенная архитектура с репликацией;
- не требуется схема описания данных, а каждый документ имеет уникальный идентификатор.

MongoDB – кроссплатформенная документо-ориентированная система управления базами данных. MongoDB отходит от традиционных основ реляционной структуры базы данных в пользу JSON – подобных документов с динамическими схемами. Впервые разработанный софтверной компанией MongoDB Inc. в октябре 2007 года в качестве компонента запланированной площадке в качестве продукта обслуживания, компания смещается к модели развития с открытым исходным кодом в 2009 году.

Из ее особенностей можно выделить: поддержка поиска по области, запросов по диапазону и поиск регулярного выражения; индексация; репликации; пакетная обработка данных и операции агрегирования; использование JavaScript в запросах.

Использование единого языка программирования на сервере и на клиенте – давняя мечта веб-разработчиков. Своими корнями желание иметь универсальный язык уходит в период становления Java, когда апплеты представлялись клиентским интерфейсом к написанным на Java серверным приложениям, а JavaScript первоначально виделся как облегченный скриптовый язык взаимодействия с апплетами. Но сложилось иначе, и в результате не Java, а JavaScript стал основным языком на стороне клиента-

браузера.[5] С появлением Node – мечта реализовалась, сделать JavaScript языком, используемым по обе стороны веб – на стороне клиента и сервера.

У единого языка есть несколько потенциальных плюсов:

- одни и те же программисты могут работать над обеими сторонами приложения;
- код проще переносить с сервера на клиент и в обратном направлении;
- общий для клиента и сервера формат данных (JSON);
- общий программный инструментарий (как пример IDE);
- общие для клиента и сервера средства тестирования и контроля качества;
- на обеих сторонах веб-приложения можно использовать общие шаблоны представлений;
- общий язык общения между группами разработчиков, работающими над клиентской и серверной частью.

Node упрощает реализацию этих (и других) достоинств, предлагая основательную платформу и активное сообщество разработчиков [6].

Node.js – программная платформа, основанная на движке V8 (транслирующем JavaScript в машинный код), превращающая JavaScript из узкоспециализированного языка в язык общего назначения. Node.js добавляет возможность JavaScript взаимодействовать с устройствами ввода-вывода через свой API (написанный на C++), подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызовы к ним из JavaScript-кода. Node.js применяется преимущественно на сервере.

1.5 Выбор и обоснование проектных решений

Разработка веб-приложений, адаптированных как для настольных, так и для мобильных устройств, развивается очень активно. Многие

разработчики поняли, что использование REST-архитектуры имеет ряд преимуществ и помогает избежать многих проблем в будущем, таких как перенос приложения на другую платформу.

Технически это интернет-приложение с архитектурой «клиент-сервер». Клиентом служит браузер, сервером – веб-сервис.

Технические аспекты:

- для веб-приложений на стороне сервера можно применять различные технологии и любые языки программирования. Для клиента-браузера не важно, какая ОС настроена у человека, в этом плане интернет-приложения можно считать универсальными кроссплатформенными сервисами;
- стандарты в целом общие для любых продуктов web-разработки. Функциональные основаны на реализации функций для решения задач пользователей.

Single page Application, сокращенно SPA, представляет собой веб-приложение, размещенное на одной веб-странице, которое загружает весь необходимый код вместе с самой страницей для обеспечения работы [7]. Приложение такого типа появились сравнительно недавно, с началом эры HTML5. К преимуществам использования веб-приложений относят:

– работа на большом количестве устройств. SPA отлично работают на устройствах как стационарных, так и мобильных. ПК, планшеты, смартфоны, и даже простые телефоны могут беспрепятственно работать с сайтами построенных по принципу SPA. А значит, создав одно приложение, мы получаем гораздо большую аудиторию пользователей, нежели при использовании стандартного подхода;

– богатый пользовательский интерфейс (User Experience). Поскольку существует только одна веб-страница, создание пользовательского интерфейса намного проще. Проще хранить информацию о сеансе, управлять состояниями представления и анимациями;

– сокращение загрузки одного и того же контента снова и снова.

Если портал использует шаблон, то наряду с основным контентом любой страницы посетитель сайта обязательно загружает макет шаблона. Кэширование данных на данном этапе разработки WWW достигло наивысших результатов, но если кэшировать нечего, то и время, и ресурсы на это не тратятся.

Кроме того, при обновлении веб-приложения все пользователи могут получить к нему доступ немедленно, и нет необходимости обновлять программу на компьютере каждого пользователя. Принципы любого фреймворка, который реализует парадигму SPA должны придерживаться следующих понятий и определений:

– поддерживает навигацию клиента. Все перемещения пользователя по страницам модуля четко фиксируются в истории навигации, а навигация «глубокая», то есть, если пользователь скопирует и откроет ссылку на внутренний модуль-страницу в другом браузере или окне, он попадет на соответствующую страницу;

– размещается на одной веб-странице, поэтому все скрипты и стили, необходимые для работы портала, должны быть определены в одном месте проекта – на одной веб-странице;

– хранит постоянно состояние работы клиента в кэше браузера или в Web Storage. SPA загружает все скрипты требующиеся для старта приложения при инициализации веб-страницы;

– постепенно подгружает модули по требованию.

SPA можно представить, как набор некоторых технологий и компонентов.

1.5.1 Клиентское приложение

Приложение, которое выполняется в браузере, изменяя или расширяя его возможности и организуя уровень абстракции для работы с бизнес-

логикой, логикой представления, сервисом, предоставляющим данные.

В данном случае, в качестве средства реализации клиентского приложения, было принято решение выбрать язык JavaScript [8].

Это связано с тем, что JavaScript:

– интегрирован в браузер;

– взаимодействует с элементами и стилями браузера;

– возможность взаимодействия осуществляется даже через текстовые редакторы.

Современный JavaScript – это «безопасный» язык программирования. Он не предоставляет низкоуровневый доступ к памяти или процессору, потому что изначально был создан для браузеров, не требующих этого.

Возможности JavaScript сильно зависят от окружения, в котором он работает. Например, Node.JS поддерживает функции чтения/записи произвольных файлов, выполнения сетевых запросов и т.д.

В браузере для JavaScript доступно всё, что связано с манипулированием веб-страницами, взаимодействием с пользователем и веб-сервером.

Например, в браузере JavaScript может:

1. Добавлять новый HTML-код на страницу, изменять существующее содержимое, модифицировать стили.

2. Реагировать на действия пользователя, щелчки мыши, перемещения указателя, нажатия клавиш.

3. Отправлять сетевые запросы на удалённые сервера, скачивать и загружать файлы.

4. Получать и устанавливать куки, задавать вопросы посетителю, показывать сообщения.

5. Запоминать данные на стороне клиента («local storage»).

Возможности JavaScript в браузере ограничены ради безопасности пользователя. Цель заключается в предотвращении доступа

недобросовестной веб-страницы к личной информации или нанесения ущерба данным пользователя.

Примеры таких ограничений включают в себя:

1. JavaScript на веб-странице не может читать/записывать произвольные файлы на жёстком диске, копировать их или запускать программы. Он не имеет прямого доступа к системным функциям ОС.

2. Современные браузеры позволяют ему работать с файлами, но с ограниченным доступом, и предоставляют его, только если пользователь выполняет определённые действия, такие как «перетаскивание» файла в окно браузера.

3. Существуют способы взаимодействия с камерой/микрофоном и другими устройствами, но они требуют явного разрешения пользователя. Таким образом, страница с поддержкой JavaScript не может незаметно включить веб-камеру, наблюдать за происходящим и отправлять информацию в руки злоумышленникам.

4. Различные окна и вкладки не знают друг о друге. Иногда одно окно, используя JavaScript, открывает другое окно. Но даже в этом случае JavaScript с одной страницы не имеет доступа к другой, если они пришли с разных сайтов (с другого домена, протокола или порта). Это называется «Политика одинакового источника» (Same Origin Policy). Чтобы обойти это ограничение, обе страницы должны согласиться с этим и содержать JavaScript-код, который специальным образом обменивается данными. Это ограничение необходимо, опять же, для безопасности пользователя. Страница, которую открыл пользователь, не должна иметь доступ к другой вкладке браузера и воровать информацию оттуда.

5. JavaScript может легко взаимодействовать с сервером, с которого пришла текущая страница. Но его способность получать данные с других сайтов и доменов ограничена. Хотя это возможно в принципе, для чего требуется явное согласие (выраженное в заголовках HTTP) с удалённой

стороной. Опять же, это ограничение безопасности.

В качестве фреймворка было решено выбрать один из самых популярных – Angular.

Angular – это MVVM-фреймворк от компании Google для создания клиентских приложений. Прежде всего, он нацелен на разработку SPA-решений (Single Page Application), то есть одностраничных приложений и в этом плане он является наследником другого фреймворка – AngularJS, правда, в отличие от него, он поддерживается на разных платформах (веб, мобильные устройства, нативный десктоп).

Преимущества создания клиентской части веб-приложения на Angular являются:

– большое сообщество. Она включает в себя как членов постоянной команды разработчиков и тех, кто хочет внести свой вклад в развитие с открытым исходным кодом. В Angular есть много конференций, рассказывающих о нем и спорящих в различных ИТ-сообществах. Есть много книг и интернет-ресурсов по Angular для разработчиков;

– декларативный стиль кода. При создании шаблонов в Angular используется декларативная парадигма программирования. Это делает код легче для чтения и обслуживания, потому что он описывает желаемый конечный результат, а не все шаги для его достижения;

– использование директив. Как шаблон языка в Angular использует HTML. Он расширяется директивами, которые добавляют информацию о требуемом поведении в код. Директивы позволяют сосредоточиться на логике и работать более продуктивно. Они могут быть использованы повторно, что также улучшает читаемость кода. Для Angular существует огромное количество готовых решений, которые позволяют решать достаточно разнообразные задачи с помощью готовых модулей;

– двусторонняя привязка данных. Angular использует двустороннюю привязку: любые изменения в пользовательском интерфейсе немедленно

отражаются в объектах приложения и наоборот. Фреймворк сам отслеживает события браузера, изменения модели и действия пользователя на странице, чтобы сразу обновить необходимые шаблоны.

Последняя версия Angular 7 вышла в октябре 2018 года.

Кроме того, одной из ключевых особенностей этой системы является использование машинопись язык программирования – это строго типизированный и компилируемый язык, который, в свою очередь, опирается на ES6 и является надмножеством JavaScript, который означает, что любая JavaScript-программа представляет собой машинописный текст программы. В TypeScript можно использовать все те конструкции, которые используются в JavaScript – те же операторы, условные, циклические конструкции. Кроме того, код TypeScript компилируется в JavaScript.

Также для более легких, удобных и быстрых настроек стиля приложения был выбран препроцессор SASS, метаязык на основе CSS, который предназначен для повышения уровня абстракции CSS кода и упрощения каскадирования файлов таблиц стилей. SAS поможет сэкономить огромные стилей и малых стилей, чтобы быстро работать.

Язык SASS имеет два синтаксиса: sass и scss. В данной работе будет использоваться синтаксис sass, так как он отличается отсутствием фигурных скобок, в нём вложенные элементы реализованы с помощью отступов, что позволит писать более понятный и структурированный код.

Основными преимуществами препроцессора является:

– поддерживает работу с переменными;

– позволяет писать чистый CSS-код в стиле языков программирования;

– можно использовать синтаксис вложенности;

– совместим со всеми версиями CSS, использовать можно любые доступные CSS-библиотеки.

1.5.2 База данных

Документы хранятся в документ-ориентированных или просто базах данных документов. В двух словах, документ является аналогом кассового аппарата, в котором есть поле уникального идентификатора, а значением может быть любой тип данных, в том числе и другие хэши.

Различные базы данных документов используют различные подходы к индексированию, формулированию произвольных запросов, репликации, согласованности и другим аспектам.

Для создания базы данных была выбрана СУБД MongoDB, с тех пор СУБД как драйвер для веб-сервера пишут, используя технологию NodeJS, большую популярность, открытую и информативную документацию.

MongoDB – это система управления базами данных (СУБД) с открытым исходным кодом, ориентированная на документы и не требующая описания схемы таблицы. Написана на C++. Эта база данных используется гигантскими сайтами, такими как Foursquare и bit.ly, и в Европейском центре ядерных исследований (CERN), он используется для хранения данных с Большого адронного коллайдера. Данные в MongoDB хранятся в формате BSON, что представляет собой бинарную форму представления JSON, который соответствует формату данных, использующимся в JavaScript, уменьшающий и оптимизирующий количество используемых технологий в данном проекте.

MongoDB имеет интерфейс для выполнения операций на JavaScript, возможности асинхронной репликации, поддерживает необходимые типы данных, которые могут понадобиться при проектировании базы данных по предметной области, например, массив, дата, объект и другие. Рационально соответствующе для хранить иерархические данные.

Поскольку MongoDB не имеет схемы описания хранилища данных, описание предметной области выполняется средством объектного

моделирования Object data Manager (ODM) в коде приложения [10].

В качестве такого инструмента был выбран Mongoose представляющий специальную ODM-библиотеку для работы с MongoDB, позволяет сопоставлять объекты классов и документы коллекций из базы данных и имеет следующие особенности:

– имеет поддержку транзакций;
– может перезапрашивать соединение с СУБД в случае потери связи;
– высокая степень доступности данных;
– проверяет на корректность тип сохраняемых данных;
– автоматически создает идентификаторы вложенных сущностей;
– работает под NodeJS.

1.5.3 Серверная часть

Для разработки серверной части веб-приложения была выбрана технология NodeJS.

NodeJS – это платформа для создания серверных приложений, которые ориентированы на высокую интенсивность ввода-вывода и невысокую интенсивность вычислений, а также позволяют ускорить процесс внедрения, используя тот же язык программирования, что и в веб-интерфейсе (JavaScript). Среда основана на механизме JavaScript chrome V8, который позволяет переводить вызовы JavaScript в собственный код.

Эта платформа использует разработанный переводчик от Google, который позволяет напрямую преобразовывать код JavaScript в ассемблер целевого процессора, что обеспечивает высокую производительность.

Для создания гибкой конфигурации и быстрой разработки серверной части веб-приложения была выбрана одна из самых популярных платформ lightweight Express.js которое имеет отличительные черты [11]:

– высокая скорость обработки запросов, что повысит скорость работы ВП в целом на серверной части;

– максимальное покрытие тестами;

– встроена мощная система маршрутизации;

– популярность и документация;

– готовый набор функций для разработки SPA.

Также для надежной аутентификации приложения был выбран популярный middleware Passport.

Passport – это открытая стандартная библиотека для распределенной аутентификации и авторизации узла. Библиотека поддерживает авторизацию через огромное количество сервисов, в том числе «Вконтакте», «Твиттер» и другие. В частности, это промежуточное ПО, которое легко интегрируется с приложениями Express. Для каждого поставщика услуг OAuth оптимизировано более 140 плагинов (так называемых стратегий), что позволяет различать другие элементы веб-приложений – это аспекты аутентификации, позволяющие библиотеке легко настраивать любое веб-приложение на основе Express.

В данной работе авторизация будет осуществляться с помощью плагина passport-jwt (strategy), плагина библиотек на основе JSON Web Token (JWT) [12], который представляет собой стандартизированный, в некоторых случаях, подписанный и / или зашифрованный формат упаковки данных, который используется для безопасной передачи информации между двумя сторонами, что позволит хранить хэш паролей в MongoDb.

1.5.4 Среда разработки

В качестве основной среды разработки системы выступило решение от компании JetBrains – WebStorm.

WebStorm – это среда разработки на основе JavaScript, которая

подходит как для разработки на стороне клиента, так и для разработки приложений на основе узлов. Основным преимуществом является удобный и умный редактор для JavaScript, HTML и CSS, с подсветкой кода, расширенной конфигурацией форматирования кода, проверкой ошибок на лету и умным автозаполнением. Также поддерживает TypeScript, препроцессор Sass и фреймворк, например Angular [13].

Основные возможности данной среды:

– удалённое развёртывание по протоколам FTP, SFTP, на монтированных сетевых дисках и т.д. с возможностью автоматической синхронизации;

– интеграция с системами управления версиями: Subversion, Git, GitHub, Perforce, Mercurial, CVS поддерживаются из коробки с возможностью построения списка изменений и отложенных изменений;

– интеграция с системами отслеживания ошибок;

– возможности Zen Coding и Emmet;

– live Edit, в некоторых источниках эта функциональность называется «редактирование файлов на лету» или «в реальном времени» или «без перезагрузки страницы».

Данное программное решение является коммерческим решением и предоставляет несколько способов лицензирования данного продукта:

– персональная лицензия (платная, для индивидуальных разработчиков);

– коммерческая лицензия (платная, для компаний и организаций);

– академическая лицензия (бесплатная, для студентов и преподавателей).

Для данного проекта была выбрана академическая лицензия, которая поставляется для студентов бесплатно, при этом, не вводя ограничения функционала.

2 Проектная часть

2.1 Разработка функционального обеспечения

Построенная функциональная модель «как есть» (AS-IS) в подразделе «анализ функционирования объекта исследования» аналитической части, и ее декомпозиция приводят к необходимости построения модели «как должно быть» (TO-BE).

Задача описания будущей модели – найти меры по блокированию негативного влияния неблагоприятных бизнес-факторов, полученных при анализе данных. Полученная модель представлена на рисунке 2.1.

Рисунок 2.1 – Контекстная диаграмма IDEF0 TO-BE «Исполнение госзаказов Рубцовского филиала АО «НПК Уралвагонзавод»»

Декомпозиция данной контекстной диаграммы представлена на рисунке 2.2.

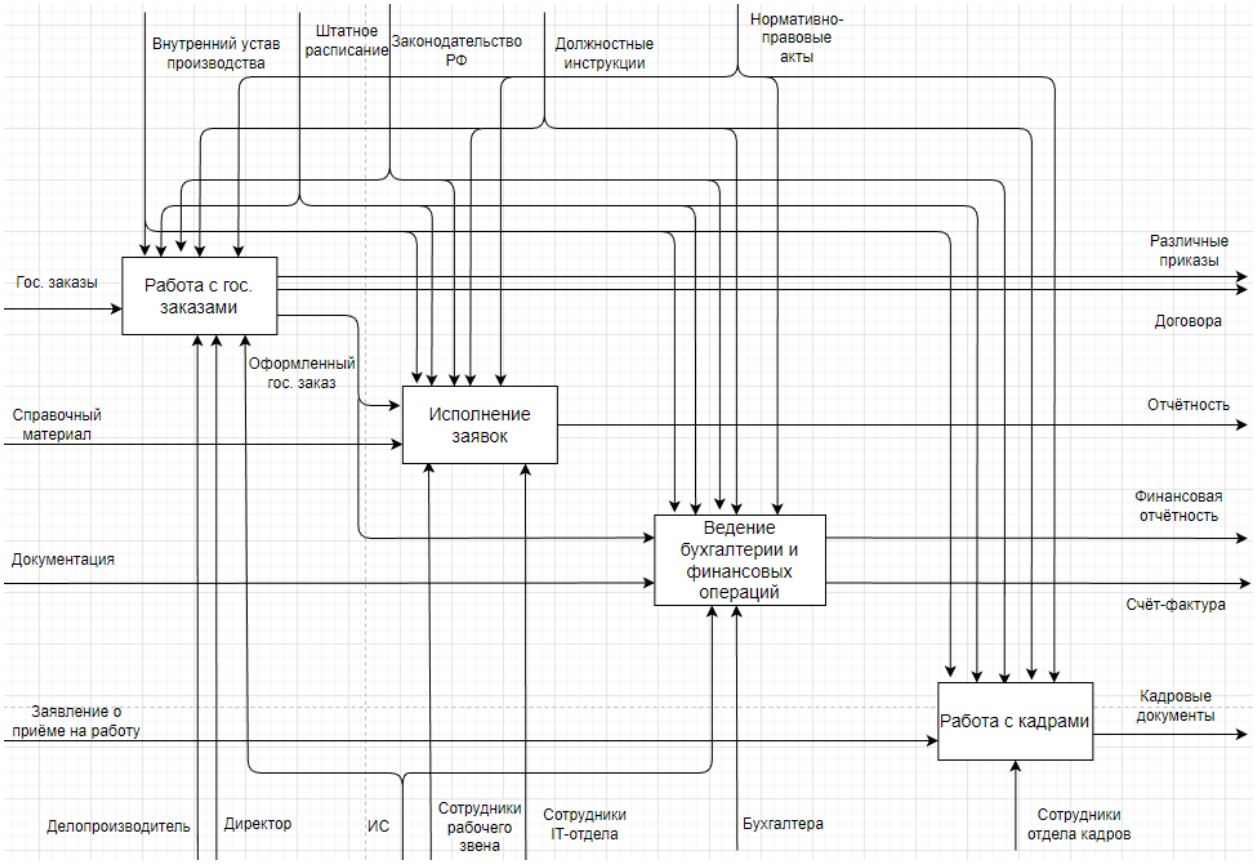


Рисунок 2.2 – Декомпозиция контекстной диаграммы IDEF0 TO-BE «Исполнение госзаказов Рубцовского филиала АО «НПК Уралвагонзавод»»

Из полученных диаграмм видно, что ИС будет полезна при работе с госзаказами и бухгалтерией и больше не будет требоваться привязка к определённому рабочему месту с компьютером так как веб-приложение будет предоставлять доступ к информации об госзаказах, работниках, бухгалтерии и отчётности о проделанной работе и позволит производить работу вне зависимости от рабочего места.

На рисунке 2.3 представлено изображение DFD-диаграммы «Как должно быть».

Рисунок 2.3 – DFD-диаграмма «Как должно быть»

Кроме того, веб-приложение позволит получать отчетные данные о госзаказах, персонале производства, самостоятельно формировать необходимые внутренние отчеты и пакет отчётной документации для заказчика.

2.2 Разработка информационного обеспечения

2.2.1 Используемые классификаторы и системы кодирования

Классификатор – систематизированный свод однородных наименований и их кодовых обозначений. Виды классификаторов:

– общегосударственные – разрабатываемые в централизованном порядке и являющиеся едиными для всей страны;

– отраслевые – единые для какой-либо отрасли деятельности;

– локальные – составляются на номенклатуры, характерные для

конкретного предприятия, организации.

Для разработанного ИС используется порядковая система кодирования. При построении системы последовательность всех записей в таблице присваиваются порядковые номера без пропуска номеров.

Таблицы состоят из автоинкрементного поля-идентификатора.

Это позволяет не управлять процессами генерации и присвоения уникальных кодов для записей в программе, а полностью обеспечить этот процесс под управлением СУБД.

Все даты представляются в стандартном для России формате «ДД.ММ.ГГГГ», где ДД – день месяца от 1 до 31, ММ – месяц от 1 до 12, ГГГГ – год в четырехзначном представлении.

Вся информация в базе данных представлена в кодировке UTF-8 – распространённой кодировке, реализующей представление Юникода, совместимое с 8-битным кодированием текста, позволяющая представить знаки практически всех письменных языков.

Она совместима с 8-битным кодированием текста, позволяющая представить знаки практически всех письменных языков.

2.2.2 Характеристика нормативно-справочной и входной оперативной информации

В базах данных MongoDB данные объединяются в коллекции.

Коллекция – это группа документов в MongoDB. Является эквивалентом простой таблицы в реляционной базе данных. Коллекция помещается в одну базу данных.

Документ-набор пар «ключ-значение», что означает, что в данной СУБД во всех документах будет присутствовать своего рода системное поле «id» – идентификатор, обязательный для индивидуализации записи в документе. Документ в коллекции может иметь различные поля. Чаще всего

все документы в коллекции создаются для одной или смежных целей. Смысл таких коллекций в том, что они позволяют объединять данные на любой основе [14]. Также при создании этой СУБД есть возможность хранить данные в облачном хранилище Yandex. В данной работе используется объектно-документный отобразитель Mongoose.

Mongoose – это библиотека JavaScript, позволяющая определять схемы со строго-типизированными данными [15].

Данная библиотека предоставляет огромный набор функциональных возможностей для создания и работы со схемами. На данный момент Mongoose содержит восемь SchemaTypes (типы данных схемы), которые может иметь свойство, сохраняемое в MongoDB, а именно: String, Number, Date, Buffer, Boolean, Mixed, ObjectId (уникальный идентификатор объекта, первичный ключ, _id), Array.

Для каждого типа данных можно:

– задать значение по умолчанию;

– задать пользовательскую функцию проверки данных;

– указать, что поле необходимо заполнить;

– задать get-функцию (геттер), которая позволяет вам проводить манипуляции с данными до их возвращения в виде объекта;

– задать set-функцию (сеттер), которая позволяет вам проводить манипуляции с данными до их сохранения в базу данных;

– определить индексы для более быстрого получения данных.

Mongoose позволяет создать модель на основе определенной схемы. Затем модель синхронизируется с документом MongoDB с использованием определения схемы модели, что исключает ручное управление коллекциями, позволяя библиотеке делать это самостоятельно на основе готовой схемы.

В данной работе будут использованы следующие схемы Mongoose: пользователь, работник, заказчик, начальник рабочей группы, рабочая группа, станок, госзаказ, план работ, ведомость фактически-отработанного

рабочего времени, обкатка техники, документы для заказчика.

В довольно простой схеме «Пользователь», показанной на рисунке 2.4, изображается структура хранения пароля и рабочей почты для входа в систему.

```
const userSchema = new Schema({
    email: {
        type: String,
        requnique: true
    },
    password: {
        type:String,
        required: true
    }
});
```

Рисунок 2.4 – Схема «Пользователь»

В схеме «Работник», показанной на рисунке 2.5, изображается структура хранения данных о работниках, такие как: имя, фамилия, отчество, дата рождения, телефон, паспортные данные, профессия работника, а также данные пользователя.

```
nst workerSchema = new Schema({
  name: {
      type: String,
      requnique: true
  },
  lastName: {
      type:String,
      required: true
  }
  patronymic: {
      type: String,
      requnique: true
  },
  birthday: {
      type: Date,
      required: true
  }
  phone: {
      type: Number,
      required: true
  }
  profession: {
      type: String,
      required: true
  }
  passport: [
      {
          serial: {
              type: Number,
              required: true
          },
          number: {
              type: Number,
              required: true
          },
          date: {
              type: Date,
              required: true
          },
          issued: {
              type: String,
              required: true
          }
      }
  ],
      user: {
          ref: 'users'
          type: Schema.Types.ObjectId
      }
});
```

Рисунок 2.5 – Схема «Работник»

В схеме «Заказчик», показанной на рисунке 2.6, изображается структура хранения данных о заказчиках, такие как: имя, фамилия, отчество,

дата рождения, телефон, а также данные пользователя, который создал данную позицию.

```
const customerSchema = new Schema({
    name: {
        type: String,
        requnique: true
    },
    lastName: {
        type:String,
        required: true
    }
    patronymic: {
        type: String,
        requnique: true
    },
    birthday: {
        type: Date,
        required: true
    }
    phone: {
        type: Number,
        required: true
    }
        user: {
            ref: 'users'
            type: Schema.Types.ObjectId
        }
});
```

Рисунок 2.6 – Схема «Заказчик»

В схеме «Начальник рабочей группы», показанной на рисунке 2.7, изображается структура хранения данных о начальниках рабочих групп, такие как: имя, фамилия, отчество, дата рождения, телефон, данные о госзаказах, в которых он участвует, данные о закрепленных к нему работников, а также данные пользователя, который создал данную позицию.

В схеме «Рабочая группа», показанной на рисунке 2.8, изображается структура хранения данных об работниках рабочей группы, такие как: номер группы, дата создания, данные о плане работ для выбранной группы, данные о закрепленном к ним госзаказе, а также данные пользователя, который создал данную позицию.

```
const chiefSchema = new Schema({
    name: {
        type: String,
        requnique: true
    },
    lastName: {
        type:String,
        required: true
    }
    patronymic: {
        type: String,
        requnique: true
    },
    birthday: {
        type: Date,
        required: true
    }
    phone: {
        type: Number,
        required: true
    }
    order: {
        type: String,
        required: true
    }
    workers: {
        type: String,
        required: true
    }
        user: {
            ref: 'users'
            type: Schema.Types.ObjectId
        }
});
```

Рисунок 2.7 – Схема «Начальник рабочей группы»

```
const groupSchema = new Schema({
    number: {
        type: Number,
        requnique: true
    },
    date: {
        tupe: Date,
        requnique: true
    }
    plan: {
        type: String,
        requnique: true
    }
    information: {
        type: String,
        requnique: true
    }
        user: {
            ref: 'users'
            type: Schema.Types.ObjectId
        }
});
```

Рисунок 2.8 – Схема «Рабочая группа»

В схеме «Станок», показанной на рисунке 2.9, изображается структура хранения данных о станках, такие как: название станка, серийный номер, категория станка, закрепленный за ним работник, а также данные пользователя, который создал данную позицию.

```
const machineSchema = new Schema({
    title: {
        type: String,
        requnique: true
    },
    number: {
        type:Number,
        required: true
    }
    category: {
        type: String,
        requnique: true
    },
    worker: {
        type: String,
        required: true
    }
        user: {
            ref: 'users'
            type: Schema.Types.ObjectId
        }
});
```

Рисунок 2.9 – Схема «Станок»

В схеме «Госзаказ», показанной на рисунке 2.10, изображается структура хранения данных о госзаказах, такие как: номер рабочей группы, дата заключения договора, серии/номера договора, статусе заказа, платы за заказ, начальник рабочей группы, плане работы и данные пользователя, который создал данную позицию.

В схеме «План работы», показанной на рисунке 2.11, изображается структура хранения данных доступных планах работ на категорию, такие как: план работ, описание работ.

```
const orderSchema = new Schema({
    number: {
        type: Number,
        requnique: true
    },
    date: {
        type:Date,
        required: true
    }
    series: {
        type: Number,
        requnique: true
    },
    status: {
        type: String,
        required: true
    }
    payment: {
        type: Number,
        required: true
    }
    master: {
        type: String,
        required: true
    }
    plan: {
        type: String,
        required: true
    }
        user: {
            ref: 'users'
            type: Schema.Types.ObjectId
        }
});
```

Рисунок 2.10 – Схема «Госзаказ»

```
const planSchema = new Schema({
    plan: {
        type: String,
        requnique: true
    },
    description: {
        type:String,
        required: true
    }
});
```

Рисунок 2.11 – Схема «План работ»

В схеме «Ведомость фактически-отработанного рабочего времени», показанной на рисунке 2.12, изображается структура хранения данных о посещаемости производства работником, такие как: дата работы, факт посещения, причина отсутствия, сведение о работнике и данные пользователя, который создал данную позицию.

```
const timeSchema = new Schema({
    date: {
        type:Date,
        required: true
    }
    visiting: {
        type: String,
        requnique: true
    },
    absence: {
        type: String,
        requnique: true
    },
    worker: {
        type: String,
        required: true
    }
        user: {
            ref: 'users'
            type: Schema.Types.ObjectId
        }
});
```

Рисунок 2.12 – Схема «Ведомость фактически-отработанного рабочего времени»

В схеме «Обкатка техники», показанной на рисунке 2.13, изображается структура хранения данных о результате обкатки техники, такие как: дата проведения обкатки, результат прохождения первого этапа обкатки, результат прохождения второго этапа обкатки, сведение о технике, прошедшей обкатку, данные пользователя, который создал данную позицию.

В схеме «Документы для заказчика», показанной на рисунке 2.14, изображается структура хранения данных о документах для заказчика, такие как: дата работы с заказом, список проведённых работ, результат прохождения первого этапа обкатки, результат прохождения второго этапа

обкатки, сведение о технике, данные пользователя, который создал данную позицию.

```
const technicSchema = new Schema({
    date: {
        type:Date,
        required: true
    }
    result_1: {
        type: String,
        requnique: true
    },
    result_2: {
        type: String,
        requnique: true
    },
    technic: {
        type: String,
        required: true
    }
        user: {
            ref: 'users'
            type: Schema.Types.ObjectId
        }
});
```

Рисунок 2.13 – Схема «Обкатка техники»

```
const documentsSchema = new Schema({
    date: {
        type:Date,
        required: true
    }
    plan: {
        type: String,
        requnique: true
    },
    result_1: {
        type: String,
        requnique: true
    },
    result_2: {
        type: String,
        required: true
    }
    technic: {
        type: String,
        required: true
    }
        user: {
            ref: 'users'
            type: Schema.Types.ObjectId
        }
});
```

Рисунок 2.14 – Схема «Документы для заказчика»

2.2.3 Характеристика результатной информации

К результатной информации в веб-приложении относится:

– информация об аутентификации пользователя;

– сведения обо всех работниках;

– сведения обо всех рабочих группах и выполненных работах;

– сведения о всех мастерах рабочих групп;

– сведения о станках;

– результаты обкатки;

– ведомость фактически-отработанного рабочего времени работником;

– различные отчеты.

Вся перечисленная в результате информация будет передана в веб-приложение для дальнейшей обработки и представления в форме, необходимой пользователю, и будет передана только в формате JSON [16].

JSON является простой текстовый формат обмена данными, легко читать и писать, о человеке и компьютере. Он основан на подмножестве языка программирования JavaScript, и поскольку в этой информационной системе абсолютно все компоненты основаны на этом языке, это делает его единственным правильным решением для использования.

Также, данный формат основан на двух структурах данных:

Объект – это неупорядоченное множество пар «ключ: значение», заключённое в фигурные скобки «{» и «}». Ключ описывается строкой, между ним и значением стоит символ «:». Пары ключ-значение отделяются друг от друга запятыми;

Массив – упорядоченная коллекция значений.

Строка – коллекция нуля или больше символов Unicode, заключенная в двойные кавычки, используя в качестве символа экранирования. Для представления числа используется только десятичная система счисления.

Это универсальные структуры данных. Почти все современные языки программирования поддерживают их в какой-либо форме.

Веб-приложение, посредством AJAX-запросов, будет передавать необходимые параметры серверу, а в качестве результатной информации будет получать ответ в формате JSON [17].

Структура результативной информации имеет относительный формат, потому что сам формат данных может выбирать клиентское приложение, и структура ответа может меняться в зависимости от версии API.

Базовыми коллекциями являются:

– работники;

– начальник рабочей группы;

– заказчик.

На рисунке 2.15 представлена структура коллекции «Работника» в формате JSON.


```
{
  "_id": {
    "$oid": "5gh64vhvu53hf"
  },
  "name": "Иван",
  "lastName": "Иванов",
  "patronymic": "Иванович",
  "phone": 89991235544,
  "birthday": "1970-10-13",
  "passport": [
    {
      "_id": {
        "$oid": "5hg5hhx76vb9"
      },
      "serial": 2020,
      "number": 123456,
      "date": "2015-12-27",
      "issued": "ОУФМС РФ ПО НСО"
    }
  ],
  "user": {
    "$oid": "gh42hvhh3ah"
  },
  "__v": 0
}
```


Рисунок 2.15 – Структура коллекции «Работник» в формате JSON

На рисунке 2.16 представлена структура коллекции «Мастера рабочей

группы» в формате JSON.

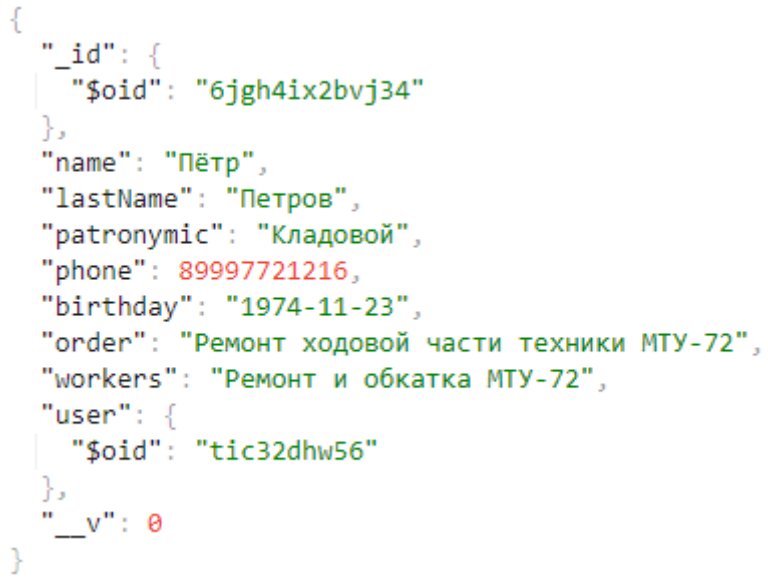


Рисунок 2.16 – Структура коллекции «Начальник рабочей группы» в формате JSON

На рисунке 2.17 представлена структура коллекции «Заказчик» в формате JSON.

Рисунок 2.17 – Структура коллекции «Заказчик» в формате JSON

2.2.4 Информационная модель и ее описание

Начальной стадией проектирования системы баз данных является

построение семантической модели предметной области, которая базируется на анализе параметров объектов предметной области и информационных потребностей будущих пользователей разрабатываемой системы. Этот этап называется концептуальным проектированием системы, а ее результат – концептуальной моделью предметной области [18].

Объектом моделирования здесь является предметная область будущей системы. Этот этап также соответствует терминам «концептуальный дизайн» и «концептуальная модель».

Как правило, такие модели представляют информационные потребности пользователей создаваемой системы с точки зрения использования хранимых данных и являются по сути средством коммуникации, как разработчиков, так и пользователей на разных этапах жизненного цикла базы данных. К инфологическим моделям относятся различные компоненты, по-разному и разными способами отражающие предметную область. Помимо наиболее известного описания объектов и отношений между ними (модель «сущность-связь»), к мифологическому уровню описания предметной области можно отнести следующие компоненты:

– систему атрибутов и средств описания предметной области. Например, логические (алгоритмические) связи между показателями или лингвистические свойства языка (синонимию, синтаксис и т. д.), используемого для вербального представления объектов;

– ограничения целостности, определяющие допустимость значения отдельных полей и взаимосвязей как на уровне семантики содержимого БД, так и ее физической структуры (отдельных файлов данных и взаимосвязей между ними);

– описание информационных потребностей пользователей, например, в виде типовых запросов, отражающих процедурные особенности обращения к данным.

Разработка информационной модели является последним этапом проектирования информационной системы. Этот процесс представляет собой построение двух типов логических и физических моделей.

Но, информационная система основана на сборниках различных документов. Количество полей, содержание и размер этих документов могут отличаться. То есть, разные сущности не обязательно должны быть идентичными по структуре, из-за этого они не могут иметь схему, подобную реляционным базам данных. Все модели, создаваемые мангустом с помощью метода Schema, автоматически создают коллекции в самой базе данных MongoDb, управляя серверной частью приложения с помощью входной библиотеки [19].

На рисунке 2.18 представлена логическая схема базы данных, а на рисунке 2.19 – физическая схема базы данных.

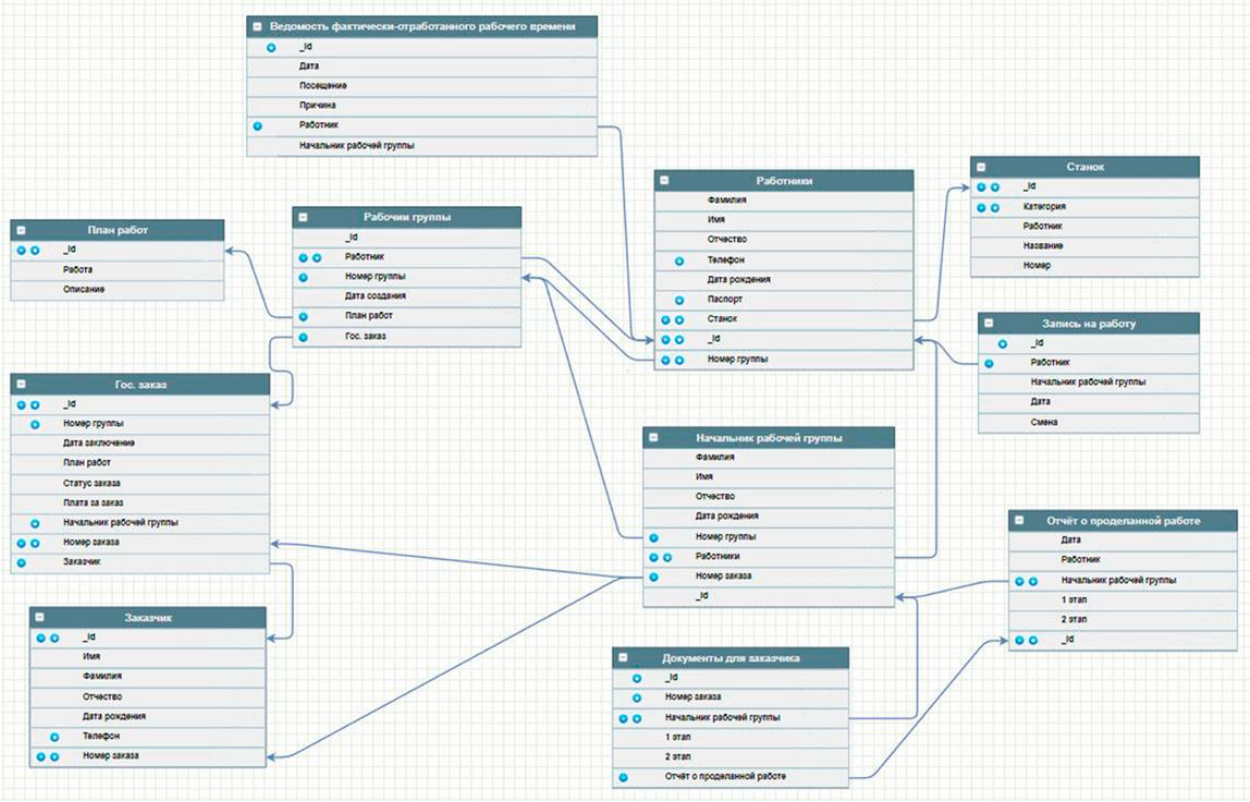


Рисунок 2.18 – Логическая схема базы данных

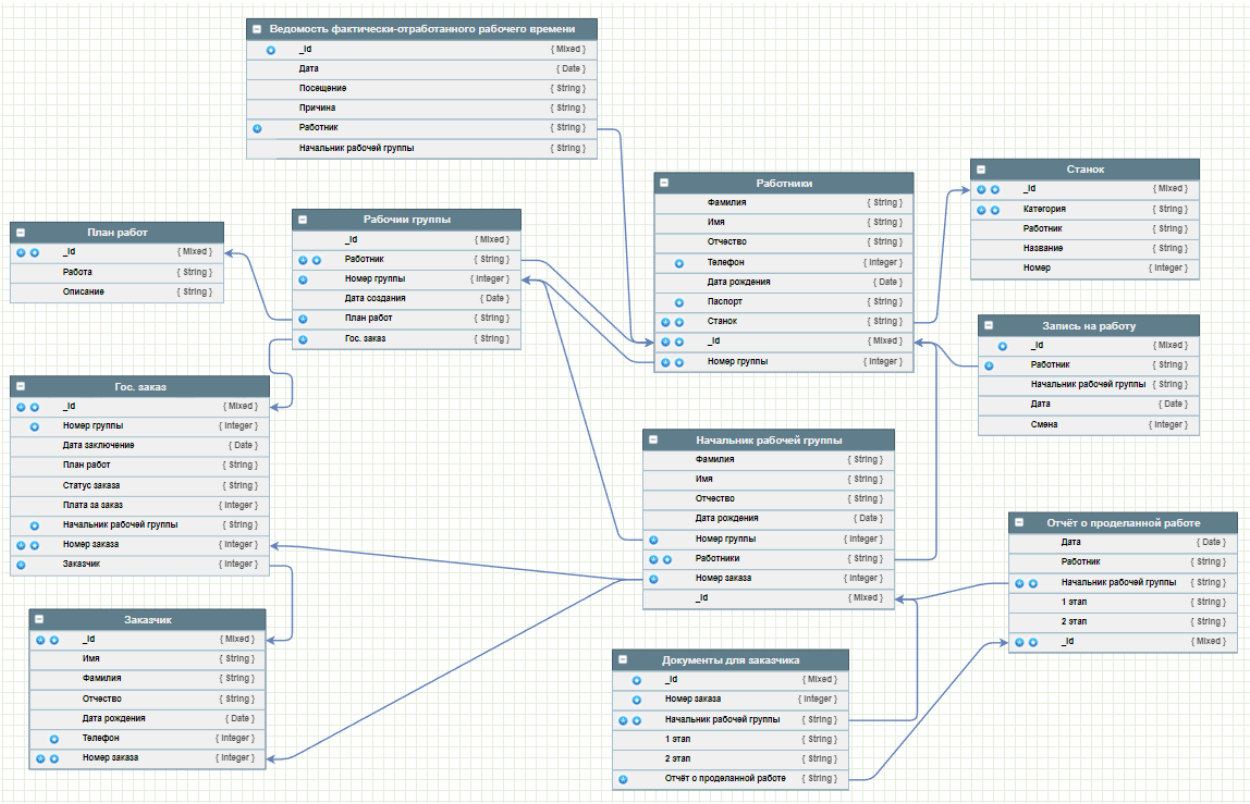


Рисунок 2.19 – Физическая схема базы данных

2.3 Разработка программного обеспечения

Разработанная информационная система позволит:

– повысить степень защищенности обрабатываемой информации;

– своевременно производить необходимые отчеты;

– устранить необходимость ручного заполнения необходимой документации для заказчика;

– вести личные карточки работников и начальников рабочих групп;

– вести учёт и печать документов, связанных с приемом, процессом работ по заказу и подготовку отчётной документации;

– формировать и печатать комплекты документов для предоставления заказчику.

2.3.1 Описание серверных программных модулей

API (интерфейс программирования приложений, интерфейс прикладного программирования) – набор готовых классов, процедур, функций, структур и констант, предоставляемых приложением (библиотекой, сервисом) или операционной системой для использования во внешних программных продуктах.

Функциональность определяется приложением (модуль, библиотеку), в то время как API позволяет абстрагироваться от того, как эта функциональность реализована.

API функции и библиотеки классов включает описание сигнатур и семантики таких функций, как – REST (Representative State Transfer – «просмотр передачи состояния») – архитектурный стиль взаимодействия компонентов распределенного приложения в сети [20].

Остальное – это согласованный набор ограничений, которые учитываются при проектировании распределенной системы гипермедиа. В некоторых случаях (на основе данных) это приводит к повышению производительности и упрощению архитектуры. В широком смысле компоненты REST взаимодействуют как клиенты и серверы во всемирной паутине. REST является альтернативой RPC.

REST API подразумевает под собой простые правила:

– каждый URL является ресурсом;

– при обращении к ресурсу методом GET возвращается описание этого ресурса;

– метод POST добавляет новый ресурс;

– метод PUT изменяет ресурс;

– метод DELETE удаляет ресурс.

Эти правила предоставляют простой CRUD интерфейс для других приложений, взаимодействие с которым происходит через протокол HTTP.

Express framework – это набор модулей узлов.js которое можно использовать индивидуально или совместно. Приложение взаимодействует с источником через модель API Express, доступную локально в узле.js, удаленно через REST и через собственные клиентские API для iOS, Android и HTML5. С помощью этого API приложения могут запрашивать данные из баз данных, записывать данные, загружать файлы, отправлять электронную почту, создавать push-уведомления, регистрировать пользователей и выполнять другие действия, предоставляемые хранилищем данных и службой. Клиенты могут получить доступ к API Express напрямую, используя сильное удаленное взаимодействие, встроенные протоколы связи, которые позволят вам отправить через API REST.

На рисунке 2.20 представлены основные серверные модули API их зависимости.

Каждый серверный модуль API предполагает взаимодействие с другими модулями, и выполняют общую работу.

Всего в структуре состоят следующие модули:

− client – клиентский модуль, используется для разработки клиентского приложения;

− config – модуль, в котором хранятся все конфигурации, присущие проекту, например ключ доступа к базе данных;

− controllers – модуль, полностью реализующий обработку данных информационной системы;

− middleware – модуль, в котором хранятся вспомогательные инструменты обработки данных;

− models – модуль, в котором модели mongoose, использующиеся для быстрого и динамического прототипирования и использования в API не заботясь о ее реализации;

− routes – модуль, в котором хранятся файлы обработки запросов клиентов;

− config – модуль, в котором хранятся функции или классы, выполняющие вспомогательные действия обработки кода.

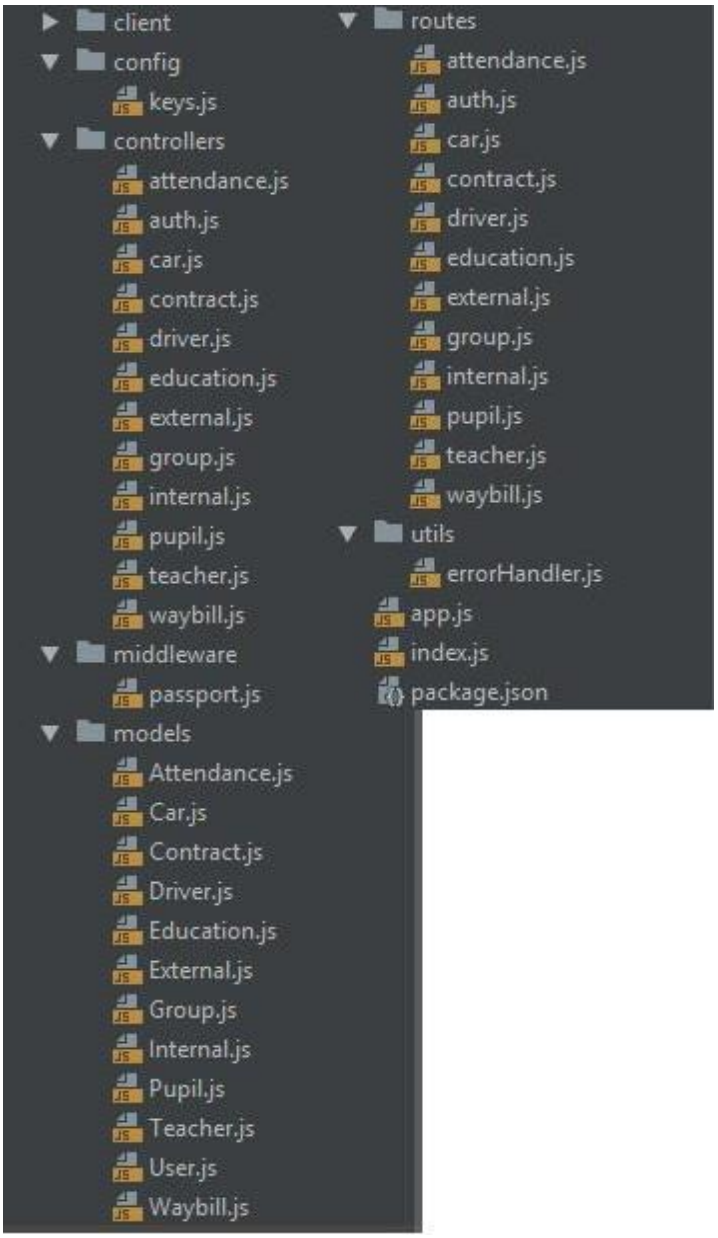


Рисунок 2.20 – Серверные модули API

2.3.2 Описание клиентских программных модулей

В качестве клиента выступает модуль «Информационная система «Производство»», разработанная на основе языка программирования JavaScript.

Как правило, веб-приложение состоит из нескольких модулей, обычно

один из модулей обозначается как «основной» модуль, который предлагается при запуске приложения, и соединяет другие модули, что позволяет сопоставлять запросы к приложению с ресурсами внутри приложения.

Описание основных пользовательских модулей содержится в таблице 2.1. Описание основных административных модулей содержится в таблице 2.2.

Таблица 2.1 – Описание пользовательских программных модулей

Наименование модуля	Описание модуля
info-page.ts	Модуль просмотра основной информации о работе, прогресса выполнения и подачи документов по госзаказу
calendar-page.ts	Модуль просмотра графика работ
pay-page.ts	Модуль с данными о заработной плате

Таблица 2.2 – Описание административных программных модулей

Наименование модуля	Описание модуля
overview-page.ts	Модуль просмотра начальников и самих рабочих групп, история выполненных работ, а также анализ учёта выполненной работы по госзаказу
order-page.ts	Модуль просмотра и редактирования информации о заказе
order-exams.ts	Подмодуль просмотра и редактирования информации о плане работ для госзаказа и их выполнение
order-page.ts	Подмодуль просмотра и редактирования информации о прохождение обкатки техникой
office-page.ts	Модуль просмотра и редактирования внутренних данных производства
office-master.ts	Подмодуль просмотра и редактирования данных о мастерах рабочих групп
office-worker.ts	Подмодуль просмотра и редактирования данных о работниках
office-work-plan.ts	Подмодуль просмотра и редактирования планов работ
office-manufacture.ts	Подмодуль просмотра и редактирования данных производства

Схема взаимосвязи программных модулей изображена на рисунке 2.21.

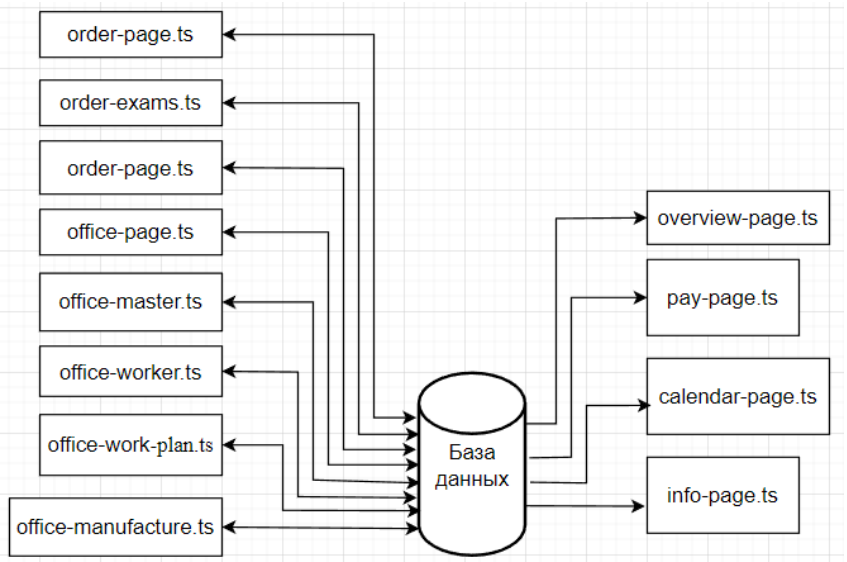


Рисунок 2.21 – Схема взаимосвязи программных модулей

Дерево функций изображено на рисунке 2.22.

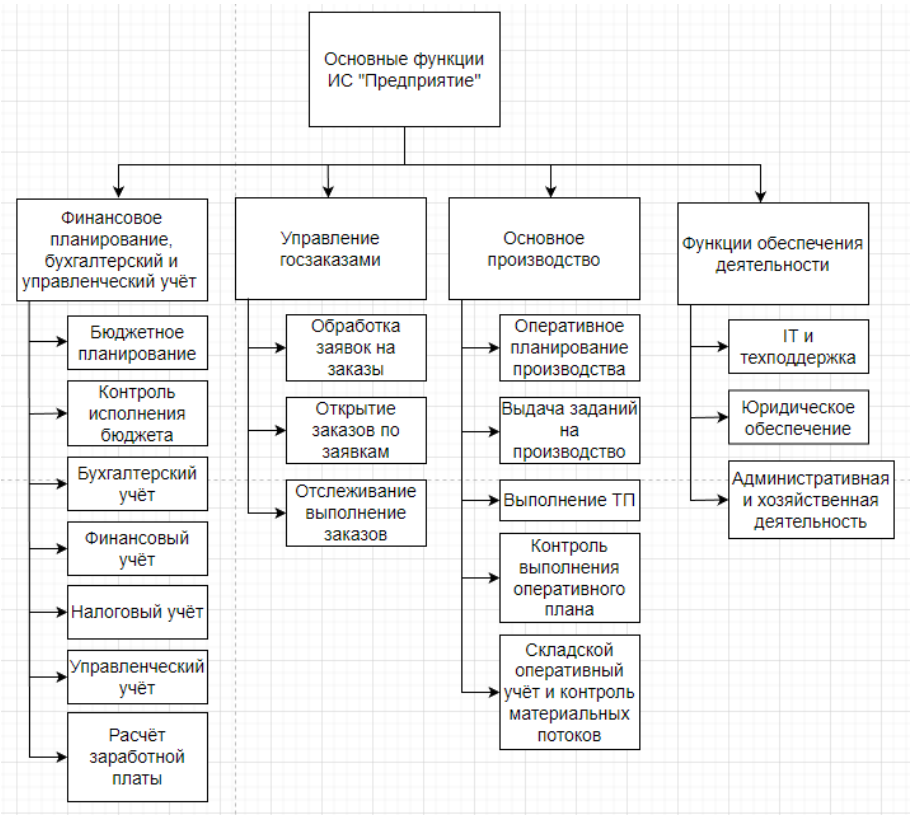


Рисунок 2.22 – Дерево функций

2.3.3 Компоненты пользовательского интерфейса

Графический интерфейс пользователя – среда взаимодействия пользователя с компьютерной системой. Графический интерфейс позволяет управлять поведением вычислительной системы через визуальные элементы управления: окна, списки, кнопки, гиперссылки и т.д.

Стандартный графический интерфейс пользователя должен отвечать ряду требований:

– поддерживать информационную технологию работы пользователя с программным продуктом – содержать привычные и понятные пользователю пункты меню, соответствующие функциям обработки, расположенные в естественной последовательности использования;

– ориентироваться на конечного пользователя, который общается с программой на внешнем уровне взаимодействия;

– удовлетворять правилу «шести» – в одну линейку меню включать не более шести понятий, каждое из которых содержит не более шести операций;

– графические объекты сохраняют свое стандартизованное назначение и, по возможности, местоположение на экране.

Разработка пользовательского интерфейса – одна из самых сложных и ответственных задач проектирования.

Для разработки информационной системы использовались принципы Material design.

Material design – это концепция, которая опирается на простую мысль, что современные технологии, в частности, современное железо и программное обеспечение, способны давать очень отзывчивый, живой и удобный интерфейс, не требуя от разработчика сильных дизайнерских навыков [21].

Информационная система состоит из двух компонентов:

– пользовательский модуль;

– административный модуль.

Вход в систему осуществляется через нажатие на кнопку «Войти в систему» на главную страницу сайта производства, представленной на рисунке 2.23.

Рисунок 2.23 – Главная страница ИС Предприятие

После чего открывается окно авторизации в систему, показанное на рисунке 2.24.

Вход в учётную запись пользователя создается в административном модуле.

После успешной авторизации пользователь перенаправляется в личный кабинет «Администратор» или «Работник».

При входе в систему под ролью «Администратор», пользователю открывается основное окно ИС (рисунок 2.25).

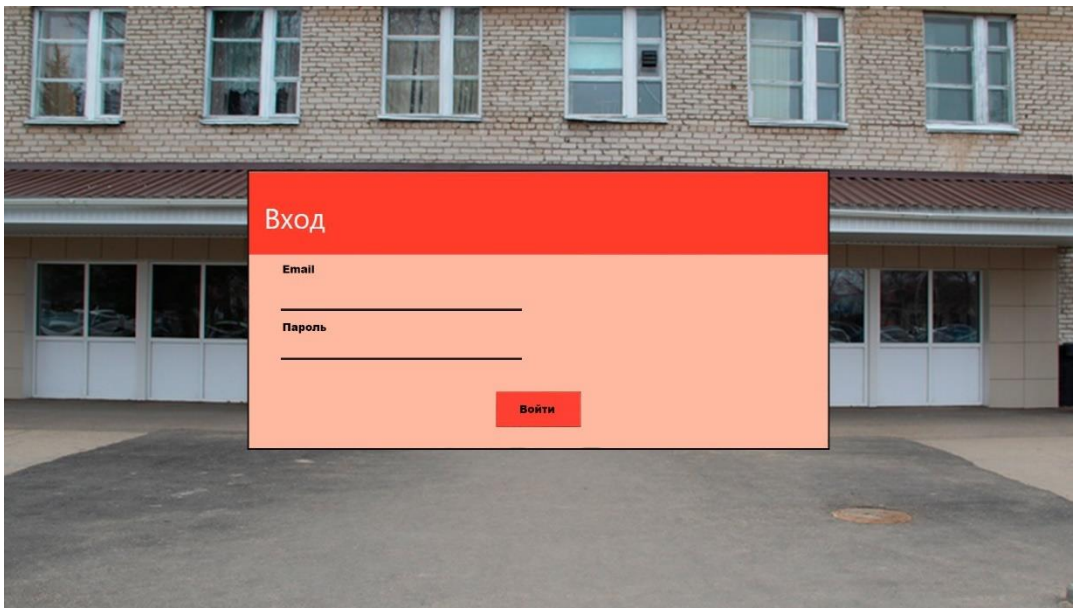


Рисунок 2.24 – Окно входа в систему

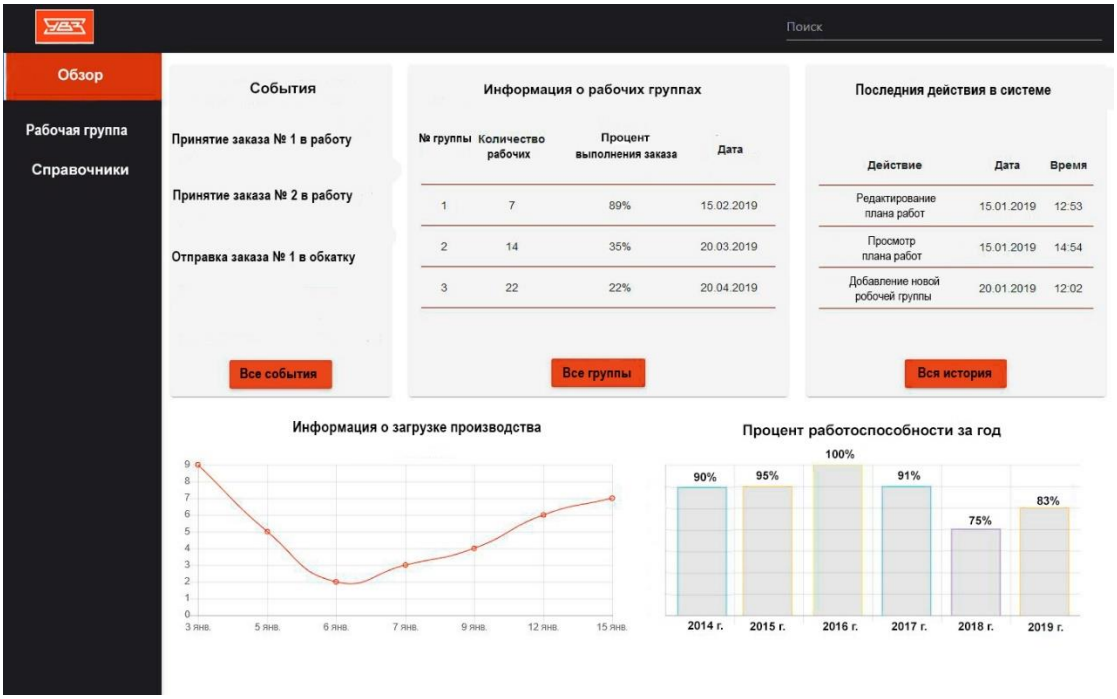


Рисунок 2.25 – Главное окно роли «Администратор» ИС

Основное меню навигации расположено слева и имеет следующие пункты:

– обзор;
– рабочая группа;

– справочники.

В разделе «Обзор» пользователю доступен следующий набор возможностей:

- информация о заказах в работе;
- информация о рабочих группах;
- история действий.

Внешний вид раздела «Рабочая группа» предоставлен на рисунке 2.26.

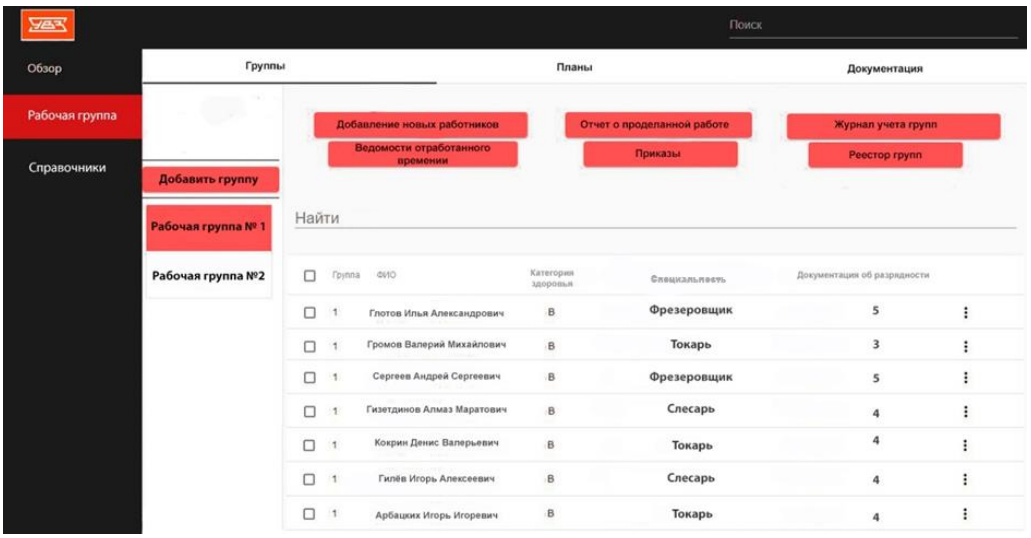


Рисунок 2.26 – Раздел «Рабочая группа»

В данном разделе пользователь получает полную информацию о рабочих группах и возможностях полностью взаимодействовать с ними.

Для удобства, данный раздел имеет три вкладки:

- группы;
- планы;
- документация.

Во вкладке «Планы» пользователю доступен следующий набор возможностей:

- создание нового плана работ;
- закрепление работы за рабочими группами;
- просмотр выполнение рабочего плана;

– просмотр ведомости затраченного времени на работу с заказом;

– перенаправление техники на обкатку;

– отчёт о прохождении обкатки.

Внешний вид вкладки «Планы» предоставлен на рисунке 2.27.

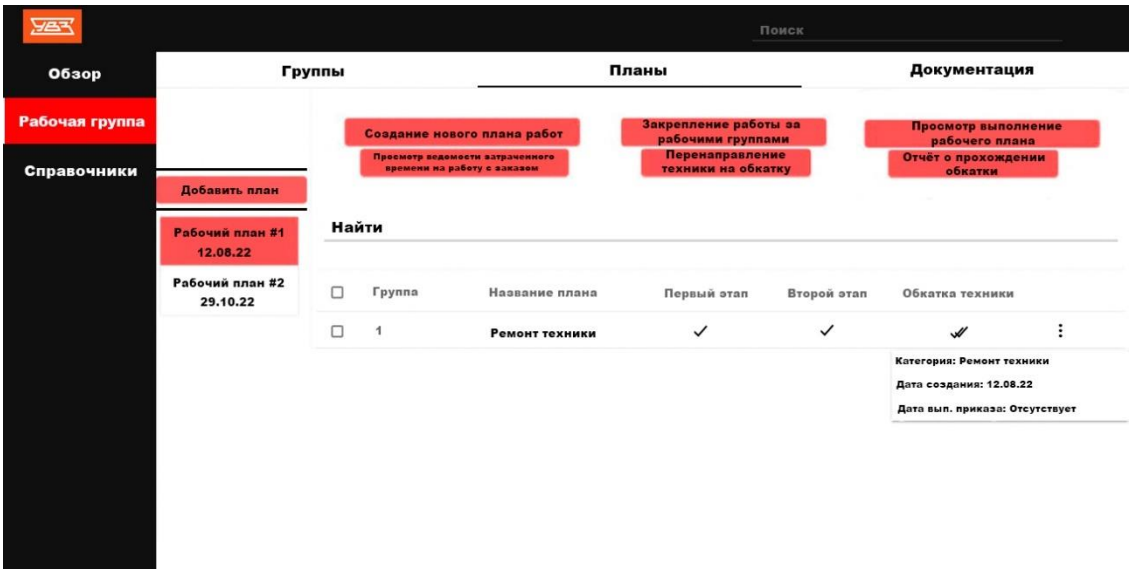


Рисунок 2.27 – Вкладка «Планы»

Внешний вид вкладки «Документация» предоставлен на рисунке 2.28.

В данной вкладке пользователю доступен следующий набор данных:

– документация о заказах;

– печать необходимой документации для рабочих группы;

– выгрузка данных о рабочей группе в формате xls для начальника группы;

– отчёты о проделанной работе;

– отчёт о прохождении обкатки техникой;

– архивация данных о выполненных заказах.

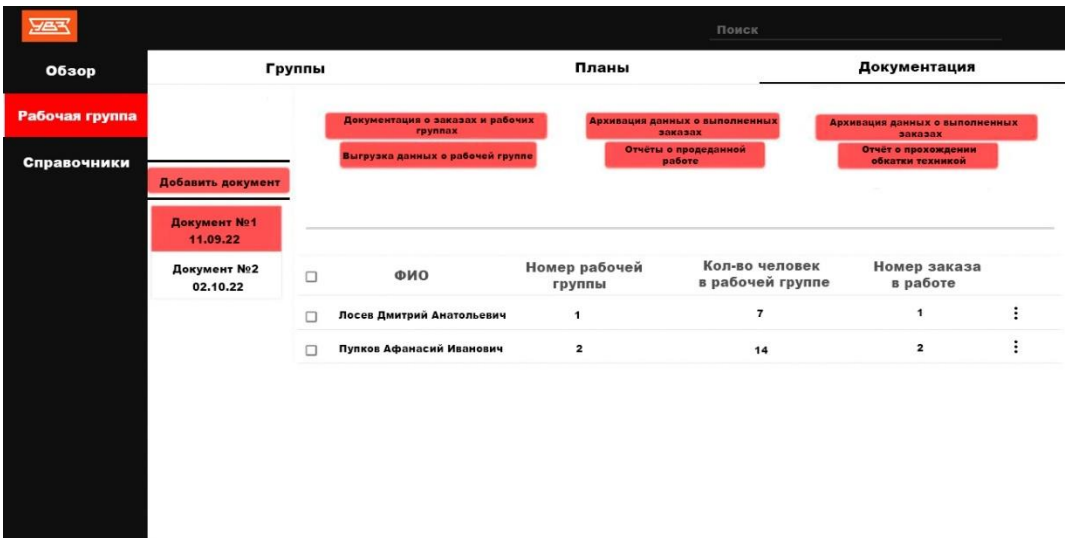


Рисунок 2.28 – Вкладка «Документация»

Внешний вид раздела «Справочники» предоставлен на рисунке 2.29. Данный раздел имеет 4 вкладки:

– начальники рабочих групп;
– работники;
– планы работ;
– предприятие.

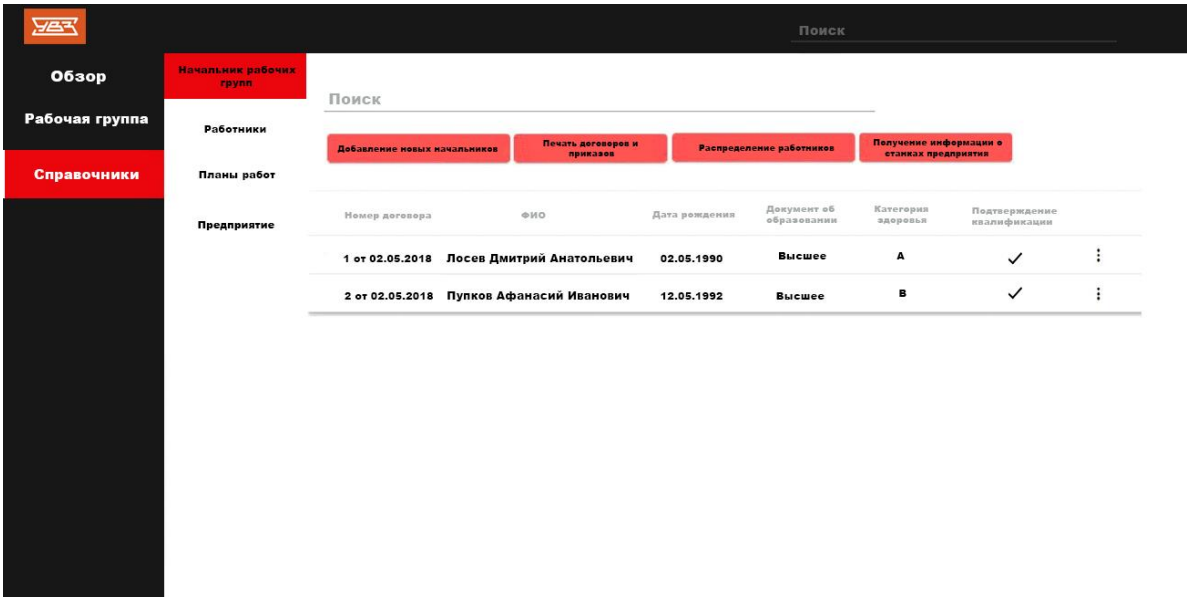


Рисунок 2.29 – Раздел «Справочники»

Первая вкладка «Начальники рабочих групп» изображена при входе в

раздел «Справочники» и предоставляет пользователю следующие функции:

– добавление новых начальников;

– печать необходимых договоров и приказов;

– распределение работников по рабочим группам;

– получение информации о станках предприятия и их закрепление за работниками.

Внешний вид вкладки «Работники» предоставлен на рисунке 2.30.

В данной вкладке пользователю доступен следующий набор функций:

– добавление новых работников;

– печать договора о принятии на работу;

– печать необходимых приказов;

– ведение учёта посещение работы работником;

– просмотр и редактирование рабочего графика.

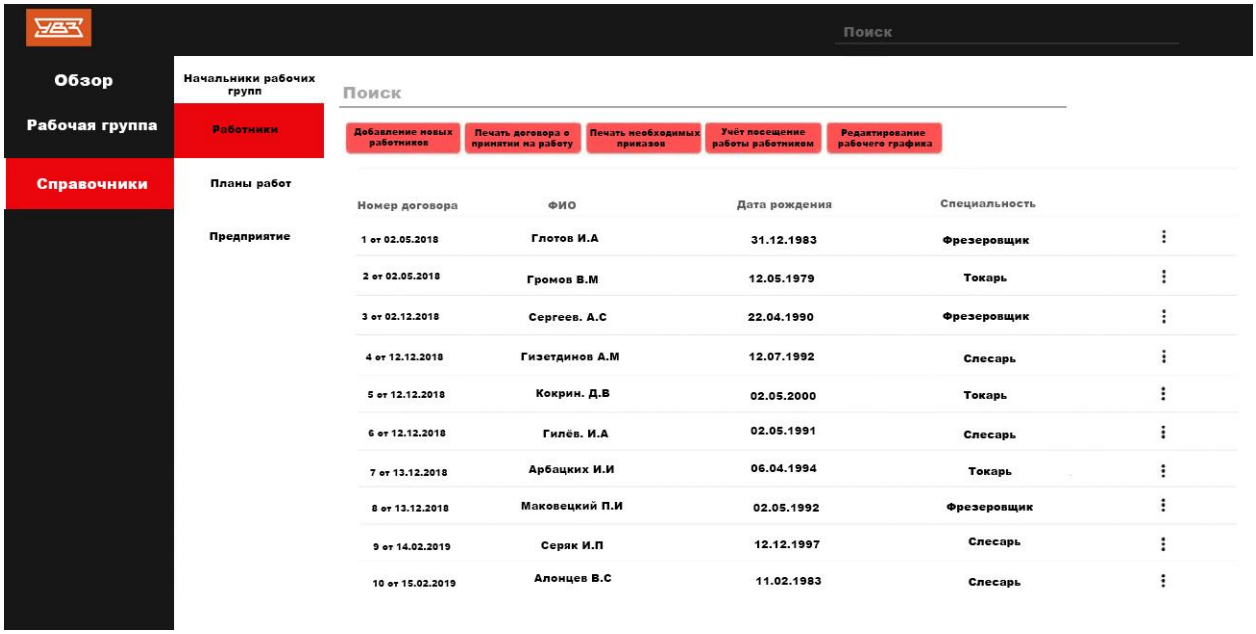


Рисунок 2.30 – Вкладка «Работники»

Внешний вид вкладки «Планы работ» предоставлен на рисунке 2.31.

В данной вкладке пользователю предоставляется возможность добавление и редактирования планов работ на какого-либо работника, количество часов, необходимых для выполнения плана работ, а также расчёт

заработной платы. Изначально количество часов выставляется согласно трудовому кодексу Российской Федерации № 588н «Порядок исчисление нормы рабочего времени»

Внешний вид вкладки «Предприятие» предоставлен на рисунке 2.32.

В данной вкладке пользователю предоставляется возможность добавление и редактирования информации о предприятии и о списке работ. Эти данные входят в отчеты, договоры и приказы, которые распечатываются из данной системы.

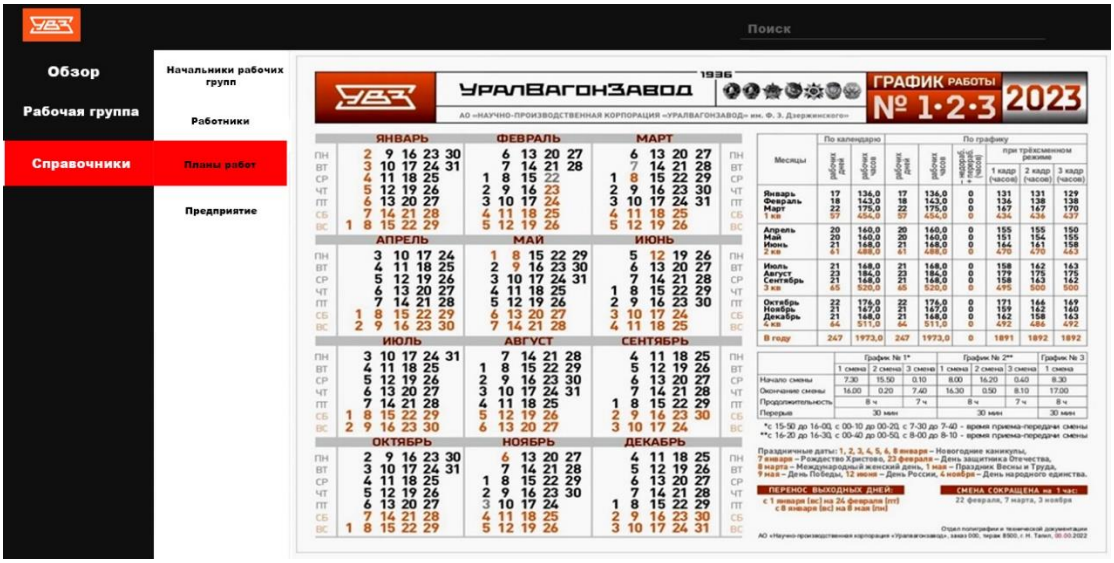


Рисунок 2.31 – Вкладка «Планы работ»

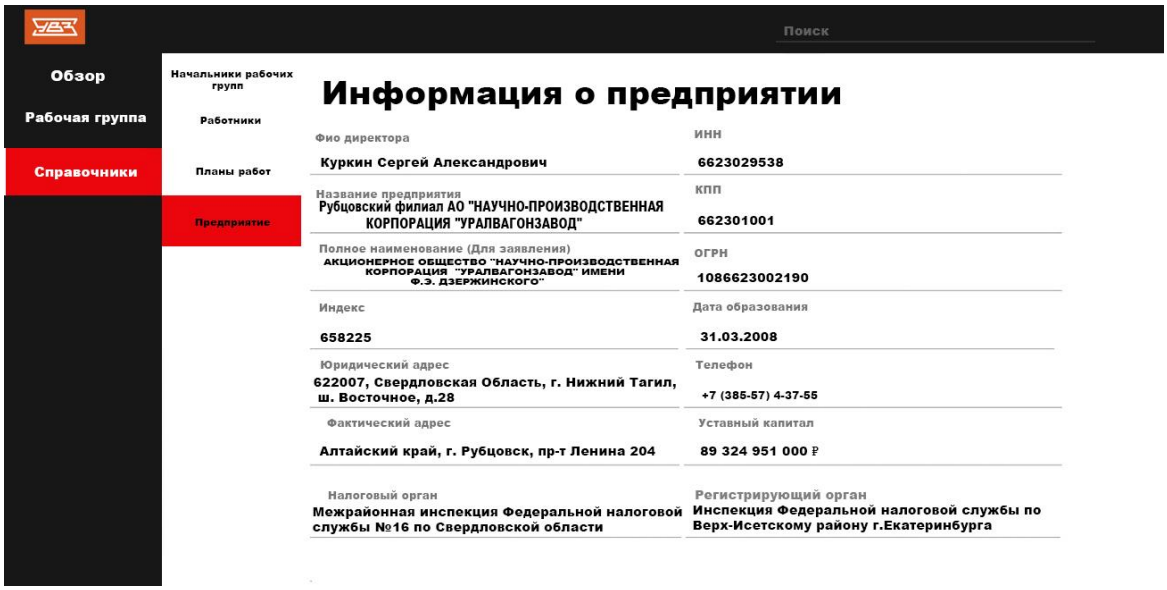


Рисунок 2.32 – Вкладка «Предприятие»

При авторизации под ролью «Работник» открывается пользовательский модуль.

В разделе «Информация», представленный на рисунке 2.33, содержится основная информация о данном пользователе, такая как, например: информация о его рабочей группе, о профессии, сведения о закрепленном за ним начальнике и станке.

Рисунок 2.33 – Раздел «Информация»

Внешний вид раздела «Рабочий график» предоставлен на рисунке 2.34. В данном разделе пользователю предоставлен рабочий план, где он может увидеть дни, на которые он записан работать.

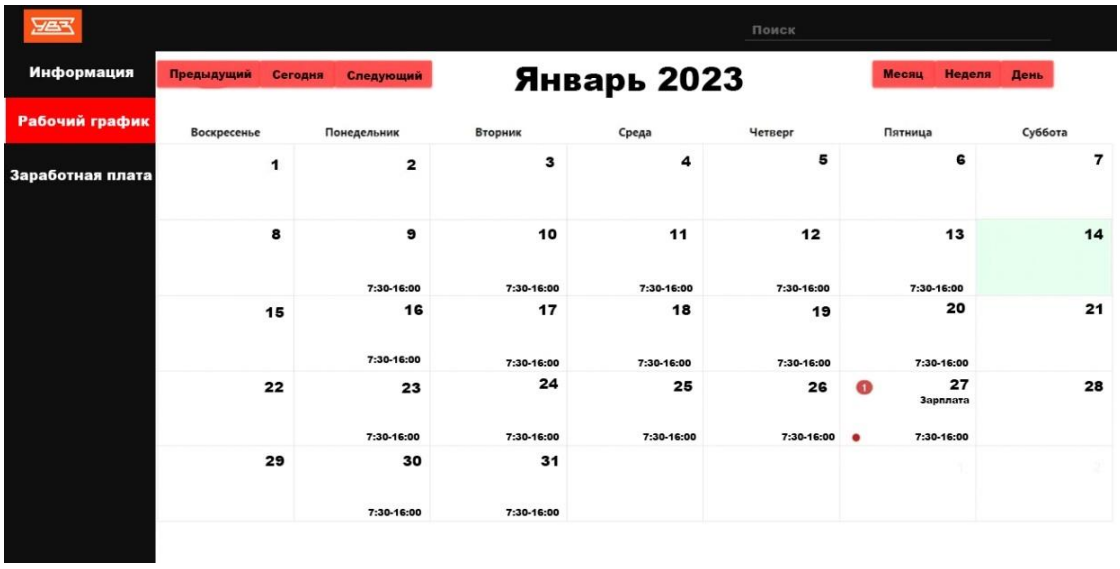
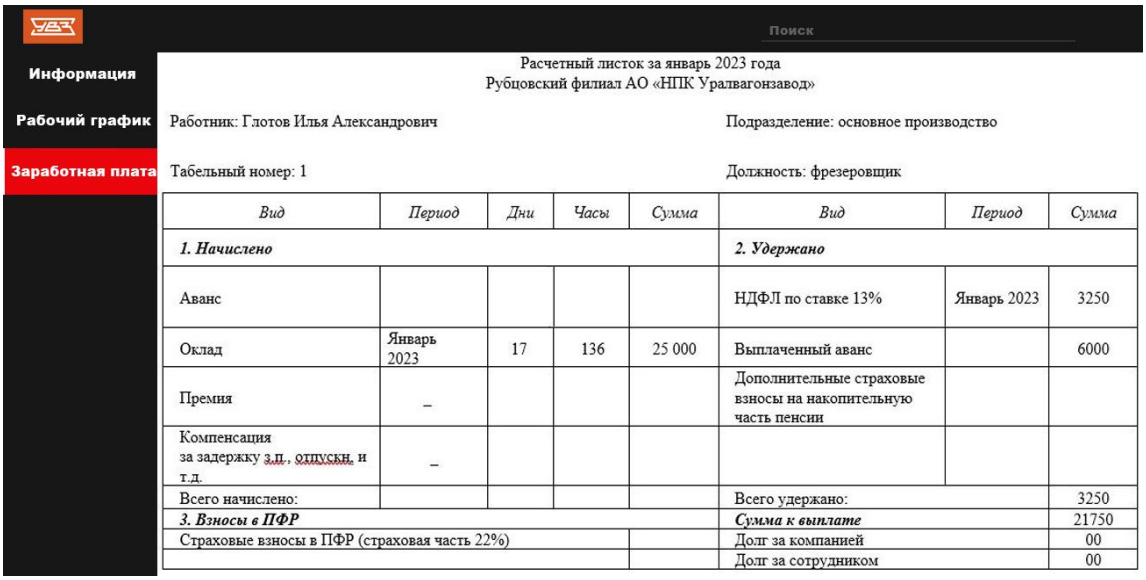


Рисунок 2.34 – Раздел «Рабочий график»

Внешний вид раздела «Заработная плата» предоставлен на рисунке 2.35.



Расчетный листок за январь 2023 года
Рубцовский филиал АО «НПК Уралвагонзавод»

Работник: Глотов Илья Александрович — Подразделение: основное производство

Табельный номер: 1 — Должность: фрезеровщик

Вид	*Период*	*Дни*	*Часы*	*Сумма*	*Вид*	*Период*	*Сумма*
1. Начислено					***2. Удержано***		
Аванс					НДФЛ по ставке 13%	Январь 2023	3250
Оклад	Январь 2023	17	136	25 000	Выплаченный аванс		6000
Премия	–				Дополнительные страховые взносы на накопительную часть пенсии		
Компенсация за задержку з.п., отпускн. и т.д.	–						
Всего начислено:					Всего удержано:		3250
3. Взносы в ПФР					***Сумма к выплате***		21750
Страховые взносы в ПФР (страховая часть 22%)					Долг за компанией		00
					Долг за сотрудником		00

Рисунок 2.35 – Раздел «Заработная плата»

В данном разделе, пользователю показана информация о расчёте его заработной платы с учётом переработок, вредности производства, графика работы и сдельной заработной платы если такая есть.

2.4 Компьютерно-сетевое обеспечение

Разрабатываемая информационная система в составе веб-приложения имеет архитектуру клиент-сервер. Всё веб-приложение размещается на облачной PaaS-платформе «Heroku», имеющую полную поддержку NodeJs, а также предоставляет поддержку СУБД MangoDb. Характеристики тарифа «Standart» полностью удовлетворяют требованиям системы.

2.5 Обеспечение информационной безопасности

При разработке проекта ИС большое внимание уделяется созданию системы защиты данных, которая подразумевает разработку системы мер безопасности, направленных на предотвращение несанкционированного получения информации, физического уничтожения или модификации охраняемой информации [22].

В зависимости от типа, разрабатываемой ИС, обрабатываемой и хранимой информации, а также уровня конфиденциальности данных, при проектировании необходимо определить ряд требований к методам защиты.

Выделяют следующие способы защиты:

– физические, законодательные;
– управление доступом;
– криптографическое закрытие.

Физические способы защиты основаны на создании физических препятствий для злоумышленника, преградив ему путь к защищаемой информации. При этом физической защитой являются: система видеонаблюдения, физические препятствия в помещении (замки, решетки на окнах, сигнализация).

Также необходимо обеспечить требования охраны труда и надежности системы на аппаратном уровне, использование источников бесперебойного питания.

Управление доступом – способ защиты информации путем регулирования доступа ко всем ресурсам системы [23].

В этом проекте, ИС для обеспечения информационной безопасности должны регулироваться по распоряжению пользователей и персонала, право доступа к отдельным файлам в базах данных. Управление доступом включает следующие функции защиты:

– идентификацию и аутентификацию пользователей;

– авторизацию;

– проверку полномочий;

– разрешение и создание условий работы в пределах установленного регламента;

– протоколирование обращений к защищаемым ресурсам;

– реагирование (сигнализация, отключение, отказ в запросе) при попытках несанкционированных действий.

Для защиты данных от потери и искажения в случае возникновения аварийной ситуации целесообразно обеспечить резервное копирование и восстановление информации [24].

Самый распространенный метод проверки подлинности, используемый в разработке, является пароль способ, но с помощью passportJs библиотека стратегии, основанной на JSON веб-маркера, который представляет собой стандартизированный, в некоторых случаях, подписанных и/или зашифрованных данных формат упаковки, который используется для безопасной передачи информации между двумя сторонами, с хранить хэши паролей в базе данных MongoDB.

При запуске IP перед пользователем появляется окно авторизации пользователя, в котором пользователь должен ввести свой email и пароль, а затем, в зависимости от введенных данных, будет перенаправлен на главный экран ИС для роли «Администратор» или «Работник».

Полный доступ на изменение информации имеет пользователь с ролью «Администратор», доступ к которому имеется только у директора заместителя директора (делопроизводителя).

3 Оценка эффективности внедрения ИС

3.1 Общие положения

При выполнении проекта по информатизации для любой организации принципиально важен вопрос об эффективности выполняемых работ.

Эффективность ИС – это свойство системы, которое выполняет поставленную цель в заданных условиях использования и с определенным качеством [25]. Такая характеристика отражает:

- эффективность системы, то есть степень соответствия ее назначению;
- техническое совершенство ИС;
- простота и технологичность разработки и создания системы;
- простота использования и обслуживания системы;
- улучшение и облегчение условий труда, изменение его содержания, развитие творческих функций, способностей и потребностей людей, преодоление существенных различий в труде;
- экономическая целесообразность внедрения ИС.

Понятие эффективности связано с получением некоторого полезного результата – эффекта использования.

Основными задачами при создании ИС являются минимизация затрат и обеспечение требуемого качества ИС.

Качество – совокупность свойств системы, что указывает на возможность ее использования для удовлетворения определенных потребностей пользователей в соответствии с ее назначением.

Основные показатели качества и ухода: надежность, надежность, безопасность.

Надежность – это свойство системы сохранять во времени в установленных пределах значения всех параметров, характеризующих

способность выполнять требуемые функции в определенных условиях использования.

Надежность IP – это средство предоставления актуальной и достоверной информации на выходе системы.

Надежность функционирования – это свойство системы, определяющее достоверность информации, ее трансформацию.

Надежность ИС полностью определяется и измеряется достоверностью его результатов.

Безопасность – это способность системы обеспечивать конфиденциальность и целостность информации, то есть защиту информации от несанкционированного доступа.

В любой сфере человеческой деятельности оценка эффективности внедрения любых новых технологий, технологий и ИС осуществляется с использованием различных показателей. К ним относятся показатели прагматической, технической, эксплуатационной, социальной и экономической эффективности.

3.2 Показатели эффективности

С помощью показателей эффективности определяется, готова ли разработанная система к поставленным перед ней задачам, и дается краткое описание показателей эффективности.

Радикально обобщенные показатели являются показателями экономической эффективности системы, характеризующими целесообразность затрат, понесенных при создании и эксплуатации системы. Наряду с экономической эффективностью, мы можем говорить о прагматической, технической, операционной и технологической эффективности. Показатели прагматической эффективности могут быть показателями:

– достоверность преобразования информации;

– безопасность информационных систем;

– точность вычислений и преобразования информации, характеризующие степень близости результирующей (выходной) информации к истинной информации, отображающей реальный процесс;

– полноты формирования системой результирующей информации, характеризующие достаточность этой информации для правильного выполнения пользователем запланированных действий;

– эффективность, показывающая, насколько быстро полученная информация формируется в системе, устарела ли она; Показатели эффективности тесно связаны с актуальностью этой информации;

– степенью сохранения ее значения во времени, что, в свою очередь, зависит от динамических характеристик выходной информации и временных интервалов ее преобразования в системе;

– своевременность, учитывающая соответствие заданного и реального момента поступления результирующей информации пользователю.

Показатели технической эффективности необходимы для оценки технического совершенства информационной системы как эргономичной технической системы с ее различными режимами и для оценки научно-технического уровня организации и функционирования системы.

К показателям социальной эффективности относятся уровень и качество жизни населения, продолжительность его жизни, благосостояние, дифференциация доходов. К таким показателям относится:

– годовой экономический эффект;

– коэффициент экономической эффективности капитальных вложений;

– срок окупаемости (в годах) капитальных вложений;

– трудоемкость обработки информации;

– эксплуатационная стоимость затрат;

– расчет текущих затрат пользователя;

– экономия текущих затрат при автоматизации;

– годовая экономия затрат на материалы.

Экономический эффект – это результат внедрения какого-либо мероприятия, выраженный в стоимостной форме, в виде экономии от его осуществления. Основными источниками экономии являются:

– улучшение показателей их основной деятельности, происходящее в результате использования программного изделия;

– повышение технического уровня, качества и объёмов вычислительных работ;

– увеличение объёмов и сокращение сроков переработки информации;

– повышение коэффициента использования вычислительных ресурсов, средств подготовки и передачи информации;

– уменьшение численности персонала, занятого обработкой исходных данных, переработкой и получением необходимой информации;

– снижение затрат на эксплуатационные материалы.

Предварительный экономический эффект представляет собой расчетный показатель, показывающий прочность до разработки на основе этих технических предложений и прогноза использования [26].

Потенциальный экономический эффект рассчитывается по завершении разработки на основе достигнутых технико-экономических характеристик и прогнозных данных о максимальных объемах использования программного продукта. Срок окупаемости является показателем эффективности использования капитальных вложений. Это период времени, в течение которого затраты на программные продукты окупаются полученным эффектом. Чтобы оценить экономическую эффективность внедрения ИС, вы можете получить прибыль от привлечения новых клиентов.

3.3 Расчет затрат на разработку ИС

График выполнения работ предоставлен в таблице 3.1.

Таким образом, на проектирование системы учёта и анализа выполнения госзаказов на производстве было затрачено 64 дней, или 448 человеко-часов.

Таблица 3.1 – График выполнения работ по разработке ИС

№ п/п	Наименование работ	Длительность работы	
		в днях	в часах
1	Разработка технического задания	2	14
2	Планирование ИС	8	56
3	Рабочее проектирование ИС	39	273
4	Отладка и тестирование ИС	12	84
5	Обобщение и оценка результатов	3	21
6	Итого	64	448

При расчете стоимости (составлении сметы затрат) разработки ИС учитываются следующие виды расходов:

- стоимость материалов и покупных изделий;
- основная заработная плата;
- дополнительная заработная плата;
- страховые взносы;
- накладные расходы;
- затраты на машинное время (затраты на электроэнергию).

Перечень затрат на материалы и покупные изделия приведен в таблице 3.2.

Таблица 3.2 – Затраты на материалы и покупные изделия

№ п/п	Наименование	Единица измерения	Кол-во	Цена за единицу, руб.	Стоимость, руб.
1	Доступ в Internet	шт.	1	400	400
2	Канцтовары	шт.	2	50	100
3	Бумага формата А4	упаковка	1	250	250
4	Аренда сервера на хостинге	шт.	1	1300	1300
4	Итого				2050

Ниже приведен расчет фонда заработной платы (основной и дополнительной заработной платы).

К этой статье относится основная и дополнительная заработная плата разработчика (программиста). Результаты расчета фонда заработной платы представлены в таблице 3.3.

Таблица 3.3 – Расчет фонда заработной платы

№ п/п	Должность: техник	Кол-во рабочих дней	Кол-во проработанных дней	Размер дневной оплаты	Заработная плата, руб.
1	Основная заработная плата	46	46	568,75	26162,5
2	Дополнительная заработная плата (10% от основной заработной платы)				2616
3	Итого фонд заработной платы				28778

В статью «Дополнительная заработная плата» входят выплаты, предусмотренные трудовым договором. Размер дополнительной заработной платы разработчика определяется в размере 10 процентов от основной заработной платы:

$$Здоп = Зосн * 10/100 = 26162,5*10/100 = 2616,2. \quad (3.1)$$

Следовательно, разработчику всего начислено:

$$Знач = (Зосн + Здоп) * = 34125+ 3412,5 = 28778 \text{ руб.} \quad (3.2)$$

Таким образом, фонд заработной платы разработчика составляет 28778 руб.

К отчислениям на социальные нужды относят страховые взносы в ПФР, ФСС, ФФОМС и взносы на страхование от несчастных случаев на производстве и профзаболеваний.

Страховые взносы рассчитываются в размере 32,2 процентов от фонда заработной платы, что составит:

$$СВ = Знач * 32,2/100 = 28778,75* 32,2/100 = 9266 \text{ руб.} \quad (3.3)$$

Тарифы страховых взносов приведены в таблице 3.4.

Отчисления в пенсионный фонд ЗПФ составляют 24 процента от фонда заработной платы и равны:

$$ЗПФ = Знач * 24/100 = 28778,75 * 24/100= 6906 \text{ руб.} \quad (3.4)$$

Отчисления в фонд обязательного медицинского страхования Змс равны:

$$Змс = Знач * 5,1/100 = 28778,75 * 5,1/100 = 1467 \text{ руб.} \quad (3.5)$$

Отчисления на социальное страхование Зсс равны:

$$Зсс = Знач * 2,9/100 = 28778,75 * 2,9/100 = 834 \text{ руб.} \quad (3.6)$$

Отчисления на обязательное социальное страхование от несчастных случаев на производстве и профессиональных заболеваний равны:

$$Знс = Знач * 0,2/100 = 28778,75 * 0,2/100 = 57,5 \text{ руб.} \quad (3.7)$$

Численные значения отчислений на социальные нужды (страховые взносы), представлены в таблице 3.4.

Таблица 3.4 – Расчет отчислений на социальные нужды (страховые взносы)

№ п/п	Отчисления на социальные взносы (страховые нужды)	Тарифы страховых взносов, в %	Суммы страховых взносов, руб.
1	Отчисления в ПФР	24,00	6906
2	Отчисления В ФОМС	5,10	1467
3	Отчисления в ФСС	2,90	834
4	Отчисления на обязательное социальное страхование от несчастных случаев на производстве и профессиональных заболеваний	0,20	57
5	Итого	32,20	9266

Размеры страховых премий устанавливаются федеральными законами. На момент разработки проекта необходимо соблюдать действующее законодательство.

Далее приведены накладные расходы.

Накладные расходы, косвенные затраты – затраты, затраты, сопутствующие, сопровождающие основное производство, но не связанные непосредственно с ним, не включенные в затраты на оплату труда и материалы, – это дополнительные расходы к основным затратам на процессы производства и обращения. Накладные расходы Зн фирмы составляют 20

процентов (условно) от суммы основной и дополнительной заработной платы:

$$Зн = (Зосн + Здоп) * 20/100 = 28778,75 * 20/100 = 5755 \text{ руб.} \quad (3.8)$$

Рассчитаем затраты на машинное время.

Как следует из данных таблицы 3.1, на разработку и тестирование веб-приложения потребовалось 46 рабочих дней (Дн).

В среднем с учётом перерывов программист работает за компьютером 7 часов в день. Себестоимость одного кВт/ч электроэнергии (С1квт/ч) для организаций составляет 6 рублей 81 копеек.

При проведении расчетов в проекте необходимо в расчеты брать существующие на дату расчета тарифы.

Суммарная мощность энергопотребителей для программиста складывается из мощности, потребляемой системным блоком персонального компьютера, монитором, принтером и другим периферийным оборудованием, которая составляет 1,2 кВт. Следовательно, за 7 часов работы программиста суммарное энергопотребление за день составит:

$$Р = 1,2 * 7 = 8.4 \text{ кВт/ч.} \quad (3.9)$$

Таким образом, стоимость машинного времени Змаш, необходимого для разработки веб-приложения составит:

$$Змаш = Р*Дн* С1квт/ч = 8,4 \text{ кВт/ч} *46 *6,81 \text{ руб./кВт/ч} = 2631 \text{ руб.} \quad (3.10)$$

Затраты на машинное время учитываются как затраты на электроэнергию.

В результате выше произведенных расчетов были получены итоговые

затраты на разработку (таблице 3.5).

В результате цена программного продукта определяется итоговыми затратами и прибылью, которая, в свою очередь, составляет 30 процентов (условно) от фонда заработной платы:

$$Ц = 52013,16+ 28778,75 * 30/100 = 60646 \text{ руб.} \quad (3.11)$$

Таблица 3.5 – Итоговая смета затрат

№ п/п	Наименование статей расхода	Сумма, руб.
1	Стоимость материалов и покупных изделий	2050
2	Основная заработная плата	28778
3	Дополнительная заработная плата	2877
4	Отчисления за социальные нужды (32, 2 % от п. 2 и п. 3)	9266
5	Накладные расходы (20 % от п. 2 и п. 3)	5755
6	Затраты на машинное время (затраты на электроэнергию)	2631
7	Итого	51360

3.4 Оценка экономической эффективности

Чтобы оценить эффективность внедрения информационной системы, необходимо сравнить время, затрачиваемое на учёт и анализ деятельности с использованием информационной системы и без использования системы.

В ходе анализа деятельности производства был выявлен сотрудник, занимающийся этой работой – заместитель директора по производственной части (делопроизводитель).

Показатель величины трудоемкости обработки информации по базовому T_0 (без использования ИС) и предлагаемому варианту T_j (с использованием ИС).

Для проектируемого бизнес-процесса следует рассмотреть уже

оптимизированный бизнес-процесс, что даст время (таблица 3.6).

Таблица 3.6 – Показатели величины трудоемкости обработки информации

№ п/п	Наименование операции	Базовый вариант (ручная технология по обработке информации) ()		Проектный вариант (проектируемая ИС) ()	
		Минут за сутки	Часов за год	Минут за сутки	Часов за год
1	Управление рабочими	60	246	20	49,2
2	Управление начальниками рабочих групп	20	49,2	10	24,6
3	Ведение журналов	30	73,8	5	12,3
4	Подготовка пакета документов для заказчика		166,8		25
5	Составление и распечатка приказов	15	36,9	5	12,3
6	Анализ и составление отчетности		98.4		12,3
5	Всего	195	671,1	55	135,7

Показатель снижения трудовых затрат () рассчитывается по формуле:

$$\Delta T = T_0 - T_j = 671,1 - 135,7 = 536,4 \text{ чел./час.} \quad (3.12)$$

Индекс снижения трудовых затрат или повышения производительности труда () рассчитывается по формуле:

$$Y_T = T_0 \div T_j = 671{,}1 \div 135{,}7 = 4{,}95. \quad (3.13)$$

Коэффициент относительного снижения трудовых затрат по учёту деятельности производства по всем сотрудникам, оценивающих их (КТ) по следующей формуле:

$$К_Т = \Delta T / T_0 * 100\% = 536{,}4/671{,}1*100 = 80\%. \quad (3.14)$$

Таким образом, на 80% снижаются трудовые затраты. Это происходит за счёт того, что теперь вся необходимая информация для деятельности производства хранится в самой системе.

Далее рассчитаем стоимостные показатели.

Абсолютное снижение стоимостных затрат (ΔС) вычисляется по формуле:

$$\Delta C = C_0 - C_j = 491 - 99 = 392 \text{ р/день}, \quad (3.15)$$

где C_0 = (671.1 / 246) * 180 = 491р – стоимость затрат на обработку информации по базовому варианту в час;

C_1= (135,7 / 246) *180 = 99р – стоимость затрат на обработку информации при внедрении ИС в час.

Коэффициент относительного снижения стоимостных затрат рассчитывается по формуле:

$$К_С = \Delta C / C_0 = 392 / 491 * 100\% = 79{,}8\%. \quad (3.16)$$

Индекс снижения стоимостных затрат, аналогично рассчитывается по формуле:

$$Y_C = C_0 / C_1 = 491/99 = 4{,}96 \quad (3.17)$$

Помимо рассмотренных показателей целесообразно также рассчитать срок окупаемости затрат на внедрение ИС (Ток) по формуле:

$$Ток = Ц / \Delta C = 60646{,}79 / 392 = 154{,}7 \quad (3.18)$$

По проведенным расчетам можно сделать следующий вывод, что проект окупится приблизительно через 5 месяцев.

ЗАКЛЮЧЕНИЕ

Целью выпускной квалификационной работы являлось проектирование информационной системы учёта работ по госзаказам для Рубцовского филиала АО «НПК Уралвагонзавод».

Для достижения поставленной цели были решены следующие задачи:

– проанализированы управленческие процессы в деятельности предприятия при исполнении госзаказов;

– выявлены недостатки в существующих реализациях информационных процессов;

– разработана функциональная архитектура проектируемой системы;

– выработаны и реализованы проектные решения по обеспечивающим подсистемам;

– оценена экономическая эффективность от внедрения информационной системы.

Результатом работы является проект информационной системы, которая позволяет автоматизировать учётные функции в деятельности научно-производственного предприятия по исполнению госзаказов.

На основе расчета экономической эффективности был сделан вывод об эффективности ИС, что выражается в сокращении времени получения первичной информации и обработки трудоемких операций, повышении достоверности и точности информации, позволит быстро и своевременно формировать необходимые отчеты.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Программная инженерия. Парадигмы, технологии и CASE-средства/ Е. М. Лаврищева – Москва: Издательство Юрайт, 2023. – 280 с. – (Бакалавриат). – Текст: электронный // Образовательная платформа Юрайт [сайт]. – URL: https://urait.ru/bcode/513086. – Загл. с экрана.

2. Веб-компоненты в действии/ Б. Фаррелл – Москва: Издательство ДМК Пресс, 2020. – 462 с. – (Бакалавриат). – Текст: электронный // Образовательная платформа Лань [сайт]. – URL: https://e.lanbook.com/book/179480. – Загл. с экрана.

3. Разработка веб-приложений/ Н. Р. Полуэктова – Москва: Издательство Юрайт, 2023. – 121 с. – (Бакалавриат). – Текст: электронный // Образовательная платформа Юрайт [сайт]. – URL: https://urait.ru/bcode/514724. – Загл. с экрана.

4. Пост реляционные хранилища данных/ Ю. П. Парфёнов – Москва: Издательство Лань, 2020. – 462 с. – (Бакалавриат). – Текст: электронный // Образовательная платформа Лань [сайт]. – URL: https://e.lanbook.com/book/179480. – Загл. с экрана.

5. Введение в веб-разработку на языке JavaScript/ И. Б. Государев – Москва: Издательство ДМК Пресс, 2019. – 144 с. – (Бакалавриат). – Текст: электронный // Образовательная платформа Лань [сайт]. – URL: https://e.lanbook.com/book/118648. – Загл. с экрана.

6. Node.js. Путеводитель по технологии/ К. К. Сухов – Москва: Издательство ДМК Пресс, 2020. – 416 с. – (Бакалавриат). – Текст: электронный // Образовательная платформа Лань [сайт]. – URL: https://e.lanbook.com/book/69954. – Загл. с экрана.

7. Искусство WebAssembly/ Р. Баттальини – Москва: Издательство ДМК Пресс, 2022. – 310 с. – (Бакалавриат). – Текст: электронный //

Образовательная платформа Лань [сайт]. – URL: https://e.lanbook.com/book/241181. – Загл. с экрана.

8. JavaScript в разработке клиентской части веб-страниц/ О. И. Красыльникова – Москва: Издательство Санкт-Петербургский государственный университет аэрокосмического приборостроения, 2022. – 87 с. – (Бакалавриат). – Текст: электронный // Образовательная платформа Лань [сайт]. – URL: https://e.lanbook.com/book/263951. – Загл. с экрана.

9. Введение в гибридные технологии разработки мобильных приложений/ Н. П. Васильев, А. М. Заяц – Москва: Издательство Лань, 2022. – 160 с. – (Бакалавриат). – Текст: электронный // Образовательная платформа Лань [сайт]. – URL: https://e.lanbook.com/book/230387. – Загл. с экрана.

10. Mongo DB Полное руководство/ Ш. Брэдшоу, Й. Брэзил, К. Ходоров – Москва: Издательство ДМК Пресс, 2020. – 540 с. – (Бакалавриат). – Текст: электронный // Образовательная платформа Лань [сайт]. – URL: https://e.lanbook.com/book/179483. – Загл. с экрана.

11. Без серверные приложения на JavaScript/ С. Стоянович, А. Симович – Москва: Издательство ДМК Пресс, 2020. – 394 с. – (Бакалавриат). – Текст: электронный // Образовательная платформа Лань [сайт]. – URL: https://e.lanbook.com/book/140588. – Загл. с экрана.

12. Основы технологий баз данных/ Б. А. Новиков, Е. А. Горшков, Н. Г. Графеева – Москва: Издательство ДМК Пресс, 2020. – 582 с. – (Бакалавриат). – Текст: электронный // Образовательная платформа Лань [сайт]. – URL: URL: https://e.lanbook.com/book/179477. – Загл. с экрана.

13. Современный JavaScript для нетерпеливых/ К. Хорстманн – Москва: Издательство ДМК Пресс, 2021. – 288 с. – (Бакалавриат). – Текст: электронный // Образовательная платформа Лань [сайт]. – URL: https://e.lanbook.com/book/190715. – Загл. с экрана.

14. Семь баз данных за семь недель. Введение в современные базы данных и идеологию NoSQL/ Э. Рэдмонд, Д. Р. Уилсон – Москва: Издательство ДМК Пресс, 2018. – 384 с. – (Бакалавриат). – Текст: электронный // Образовательная платформа Лань [сайт]. – URL: https://e.lanbook.com/book/58690. – Загл. с экрана.

15. Искуственный интеллект. Пределы возможного/ М. Бруссард – Москва: Издательство Альпина Паблишер, 2020. – 362 с. – (Бакалавриат). – Текст: электронный // Образовательная платформа Лань [сайт]. – URL: https://e.lanbook.com/book/140446. – Загл. с экрана.

16. Основы хранения данных/ Д. П. Бураков – Москва: Издательство Лань, 2022. – 60 с. – (Бакалавриат). – Текст: электронный // Образовательная платформа Лань [сайт]. – URL: https://e.lanbook.com/book/264668. – Загл. с экрана.

17. Twisted из первых рук/ М. Загадка, М. Уильямс, Т. Мост – Москва: Издательство ДМК Пресс, 2020. – 338 с. – (Бакалавриат). – Текст: электронный // Образовательная платформа Лань [сайт]. – URL: https://e.lanbook.com/book/123702. – Загл. с экрана.

18. Проектирование Информационных систем/ Т. В. Гвоздева, Б. А. Баллод – Москва: Издательство Лань, 2023. – 148 с. – (Бакалавриат). – Текст: электронный // Образовательная платформа Лань [сайт]. – URL: https://e.lanbook.com/book/152623. – Загл. с экрана.

19. Данные, информатика, знания: методология, теория, технологии: Монография/ А. А. Москвитин – Москва: Издательство Лань, 2023. – 236 с. – (Бакалавриат). – Текст: электронный // Образовательная платформа Лань [сайт]. – URL: URL: https://e.lanbook.com/book/206267. – Загл. с экрана.

20. Material Design: на Луну и обратно [Электронный ресурс]. – Режим доступа: https://habr.com/company/redmadrobot/blog/252773/. – Загл. с экрана.

21. Принципы Material Design [Электронный ресурс]. – Режим доступа: https://material.io/design/– Загл. с экрана.

22. Методы защиты информации / Ю. М. Краковский – Москва: Издательство Лань, 2021. – 6236 с. – (Бакалавриат). – Текст: электронный // Образовательная платформа Лань [сайт]. – URL: https://e.lanbook.com/book/156401. – Загл. с экрана.

23. Защита персональных данных в информационных системах / В. И. Петренко, И. В. Мандрица – Москва: Издательство Лань, 2022. – 108 с. – (Бакалавриат). – Текст: электронный // Образовательная платформа Лань [сайт]. – URL: https://e.lanbook.com/book/183744. – Загл. с экрана.

24. Комплексное обеспечение информационной безопасности на предприятии/ М. В. Тумбинская – Москва: Издательство Лань, 2022. – 344 с. – (Бакалавриат). – Текст: электронный // Образовательная платформа Лань [сайт]. – URL: https://e.lanbook.com/book/207095. – Загл. с экрана.

25. Основы хранения данных/ Д. П. Бураков – Москва: Издательство Лань, 2022. – 60 с. – (Бакалавриат). – Текст: электронный // Образовательная платформа Лань [сайт]. – URL: https://e.lanbook.com/book/264668. – Загл. с экрана.

26. Правовые основы цифровой безопасности бизнеса/ В. В. Беспалова – Москва: Издательство Санкт-Петербургский государственный лесотехнический университет имени С. М. Кирова, 2023. – 60 с. – (Бакалавриат). – Текст: электронный // Образовательная платформа Лань [сайт]. – URL: https://e.lanbook.com/book/326372. – Загл. с экрана.