МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ
имени ГУМАРБЕКА ДАУКЕЕВА»
Кафедра IT – инжиниринг

«ДОПУЩЕН К ЗАЩИТЕ»
Зав. кафедрой PhD, доцент Досжанова А.А
_____ «____» _____2020 г.

ДИПЛОМНЫЙ ПРОЕКТ

На тему: Разработка web-портала «Документооборот IT-организации»
Специальность 5В060200 – Информатика
Выполнил: Абдимижитов Э.М. Группа Инф-16-2
Научный руководитель: к.т.н., доцент Балгабаева Л.Ш.

Консультанты:
по экономической части: к.э.н., профессор Габелашвили К.Р
(учѐная степень, звание, Ф.И.О.)
_____ «____» _____2020 г.
по безопасности жизнедеятельности: ассистент Тыщенко Е.М
(учѐная степень, звание, Ф.И.О.)
_____ «____» _____2020 г.
по программному обеспечению: ст.преп. Майкотов М.Н
(учѐная степень, звание, Ф.И.О.)
_____ «____» _____2020 г.
Нормоконтролер: ст.преп. Абсатарова Б.Р
(учѐная степень, звание, Ф.И.О.)
_____ «____» _____2020 г.
Рецензент:_____
(учѐная степень, звание, Ф.И.О.)
_____ «____» _____2020 г.

Алматы 2020

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ
имени ГУМАРБЕКА ДАУКЕЕВА»

Институт систем управления и информационных технологий

Специальность 5В060200 – «Информатика»

Кафедра IT- инжиниринг

ЗАДАНИЕ
на выполнение дипломного проекта

Студенту Абдимижитову Эльдару Муратжановичу

Тема проекта: Разработка web-портала «Документооборот IT-организации»

Утверждена приказом по университету № ___ от «___» _____2020 г.

Срок сдачи законченного проекта «___» _____ 2020 г.

Исходные данные к проекту (требуемые параметры результатов исследования (проектирования) и исходные данные объекта): MongoDB - среда разработки БД, Node.js – среда программирования, Angular.js – среда исполнения, Visual Studio Code – интегрированная среда разработки.

Перечень вопросов, подлежащих разработке в дипломном проекте, или краткое содержание дипломного проекта:

а) аналитическая часть;
б) технологии разработки приложения;
в) функционал программного продукта;
г) экономическая эффективность проекта;
д) вопросы безопасности жизнедеятельности и охраны труда.

Перечень графического материала (с точным указанием обязательных чертежей): представлены 10 таблиц, 42 иллюстрации.

Основная рекомендуемая литература:

1 Бэнкер, К. MongoDB в действии / К. Бэнкер. - Москва: Высшая школа, 2016. - 287 с.

2 Мерсер Drupal 6. Создание надежных и полнофункциональных веб-сайтов,блогов, форумов, порталов и сайтов-сообществ / Мерсер, Дэвид. - М.: Вильямс, 2016. - 272 с.

3 Прамодкумар Дж. Садаладж, Фаулер Мартин NoSQL. Новая методология разработки нереляционных баз данных; Вильямс - М., 2015. - 192 с.

Консультация по проекту с указанием относящихся к ним разделов проекта

Раздел	Консультант	Сроки	Подпись
Экономическая часть	Габелашвили К.Р.	24.04.2020	
Безопасности жизнедеятельности	Приходько Н.Г.	23.04.2020	
Программная часть	Майкотов М.Н.	15.05.2020	
Нормоконтролер	Абсатарова Б.Р.		

ГРАФИК
подготовки дипломного проекта

Наименование разделов, перечень разрабатываемых вопросов	Сроки представления научному руководителю	Примечания
Анализ предметной области	13.01.2020 - 20.01.2020	
Проектная часть	21.01.2020 - 15.02.2020	
Экспериментальная часть	16.02.2020 - 11.03.2020	
Экономическое обоснование	24.04.2020	
Безопасность жизнедеятельности	23.04.2020	

Дата выдачи задания «____» _____2020г.

Заведующий кафедрой _____А.А. Досжанова

Научный руководитель проекта _____Л.Ш. Балгабаева

Задание принял к исполнению студент _____ Э.М. Абдимижитов

АҢДАТПА

Дипломдық жобаның тақырыбы: «IT-ұйымның құжат айналымы» web-порталын құру.

Дипломдық жобаның мақсаты - тұтынушыларға клиент жұмысының бөлігі ретінде жұмыс тапсырыстарын жасауға, құжаттармен алмасуға мүмкіндік беретін бағдарламалық өнімді әзірлеу.

Осы мақсатқа жету үшін келесі технологиялар пайдаланылды:

- клиент интерфейсін дамыту: Angular.js, HTML, CSS;
- сервер жағын дамыту: Node.js, Express.js;
- дерекқордың дизайны: MongoDB ДББЖ.

Дипломдық жұмыс кіріспеден, 5 бөлімнен және қорытындыдан тұрады.

Кіріспе жобаның негізгі мақсаттарын сипаттайды, өзектілігін көрсетеді. Келесі 3 тарауда веб-порталды құруға қатысты қадамдар сипатталған. 4-тарауда әзірленіп жатқан жобаның техникалық-экономикалық негіздемесі жасалды Бесінші тарауда еңбектің ауырлығын интегралдық-балдық бағалауды есептеу жүргізіледі.

АННОТАЦИЯ

Тема дипломного проекта: Разработка web-портала «Документооборот IT-организации».

Цель дипломного проекта – разработка программного продукта предоставляющего пользователям возможность создания нарядов на выполнение работ, обмена документами в рамках выполнения работ у клиента.

Для выполнения поставленной цели использовались технологии:

- разработка клиентского интерфейса: Angular.js, HTML, CSS;
- разработка серверной части: Node.js, Express.js;
- проектирование базы данных: СУБД MongoDB.

Дипломный проект состоит из введения, 5-и глав и заключения.

Введение описывает основные цели проекта, раскрывает его актуальность. В последующих 3-х главах следует описание этапов разработки web-портала. В 4 главе было произведено технико-экономическое обоснования целесообразности разрабатываемого проекта. В пятой главе производится расчет интегрально-балльной оценки тяжести труда.

ANNOTATION

Theme of the graduation project: Development of a web-portal "Document flow of IT-enterprise."

The purpose of the thesis project is to develop a software product that provides users with the opportunity to create work orders, exchange documents as part of the client's work.

To achieve this goal, the following technologies were used:

- development of the client interface: Angular.js, HTML, CSS;
- development of the server side: Node.js, Express.js;
- database design: DBMS MongoDB.

The graduation project consists of introduction, 5 chapters and conclusion.

Introduction describes the main objectives of the project, reveals its relevance. The following 3 chapters describe the steps involved in developing a web portal. In chapter 4, a feasibility study was made on the feasibility of the project being developed. In the fifth chapter, the calculation of the integral-point assessment of the severity of labor is performed.

Содержание

Введение

Темой дипломного проекта является Разработка web-портала «Документооборот IT-организации». В современном бизнесе, документы являются фундаментальной основой для осуществления внутренней и внешней деятельности организации. В условиях нынешней тенденции цифровизации процессов предприятий, для повышения их эффективности, система электронного документооборота может сыграть значительную роль в развитии предприятия.

В работе необходимо создать web-сайт для предприятия ТОО «Первый БИТ», который будет автоматизировать процесс создания заявок на выезд специалиста и обмен документами между заказчиком и исполнителем, создать панель авторизации, вывести форму подачи заявки и просмотра календаря специалиста, форму обратной связи, реализовать понятный и удобный интерфейс для навигации по сайту.

Предметная область это среда подверженная подробному исследованию при написании дипломного проекта. Предметной областью данной дипломной является процесс обмена документами и контроля выполненных работ у заказчиков в ТОО «Первый БИТ», а именно создание нарядов и их согласование в отделе внедрения, фиксация проведенных работ и контроль трудозатрат, процесс приема работ клиентом и подписание подтверждающих документов.

В первой главе рассматривается предметная область предприятия, ставятся задачи дипломного проекта. Во второй главе производится проектирование будущей системы, построение UML-диаграмм, разработка блок-схем. В третьей главе дипломного проекта описываются этапы разработки самого web-портала. В четвертой и пятой главе, производится расчет показателей экономической эффективности в рамках раздела экономики и расчет интегральной-балльной оценки условий труда в рамках раздела безопасности жизнедеятельности.

1 Теоретическая часть

1.1 Понятие электронного документооборота в составе управления корпоративным контентом

Традиционно Enterprise Content Management (ECM) ограничивалось только бэк-офисом, оставаясь неструктурированным, хотя и централизованным способом управления контентом. Однако за последние несколько лет в ECM произошел сдвиг в сторону более интерактивной роли в бизнесе.

Бумага близится к исчезновению, поскольку цифровизация становится новой нормой. Машинное обучение, облачные технологии и мобильные приложения открывают новые возможности для бизнеса. Появляются новые типы контента - видео, аудио, социальные - которые стирают черты традиционного ECM. Получившийся сдвиг заставил многих лидеров отрасли задуматься о переосмыслении понятия ECM.

В современной бизнес-среде существует множество определений того, что такое ECM и какую роль оно играетв современных ИС. В общем смысле - управление корпоративным контентом (ECM) определяется как совместный процесс цифрового управления и применения информации компании для поддержки ее процессов и бизнеса. Цели ECM - обеспечение того, чтобы информация была доступна и максимально полезна в любой точке жизненного цикла процесса.

Управление документами, часто называемое системой электронного документооборота (СЭД), представляет собой использование компьютерной системы и программного обеспечения для хранения, управления и отслеживания электронных документов и электронных изображений полученных путем сканирования информации с бумажного носителя либо непосредственного ввода в программу.

Согласно ISO 12651-2 документ представляет собой «записанную информацию или объект, который можно рассматривать как единое целое». Хотя это звучит немного сложно, довольно просто расшифровать, что документ это цельный объект, который можно использовать для хранения, обработки и передачи информации.

Таким образом, можно определить систему управления документами как программное обеспечение, которое контролирует и организует поток документов по всей организации. Оно включает в себя прием документов и контента, отслеживание рабочего процесса, хранилище документов, системы COLD / ERM, а также системы поиска информации. Некоторые из ключевых функций системы управления документами:

- регистрация и блокировка, чтобы координировать одновременное редактирование документа так, чтобы изменения одного человека не перезаписывали изменения другого;
- контроль версий, чтобы можно было следить за тем, как появился текущий документ, и чем он отличается от версий, которые были до;

- откат, для возвращения к предыдущей версии в случае ввода некорректных данных;

- контрольный журнал, позволяющий восстановить хронологию изменений документа.

Управление документами в конечном итоге было отнесено к управлению контентом в немалой степени потому, что сегодня нам доступно больше информации, чем когда-либо прежде, и большая ее часть не создается нами. Благодаря возникновению целого ряда источников информации, таких как интернет, флэш-накопители, смартфоны и т.д., возросла потребность в обработке всех видов информации не только с точки зрения большего количества типов мультимедиа, таких как текст, изображения и аудио, но также с точки зрения того, как эти файлы структурированы, и следовательно, насколько сложно их обрабатывать.

В данном дипломном проекте будет разработана информационная система, позволяющая хранить данные о программных продуктах и услугах, вести учет заявок и выводить сравнительный анализ в разрезе дат. Информационная система будет разработана для дальнейшей адаптации и внедрения в компанию «Первый БИТ», внедрение данной системы будет проведено без значительных изменений в структуру системы.

1.2 Аналоги разрабатываемой программы

Вопрос о переходе на систему электронного документооборота уже давно остро стоит в некоторых компаниях. Развитие системы электронного обмена документами также является вопросом, рассматриваемым на государственном уровне. В данный момент существует множество разработанных систем обмена документами, каждая из них имеет свои функции и проработанную структуру, также каждая из систем имеет свой ряд преимуществ и недостатков. Требования к системе электронного документооборота значительно разнятся в зависимости от структуры организаций и бизнес-процессов происходящих в них. Для всех систем электронного документооборота имеются как общие негласные требования, так и специфические.

Таблица 1.1 - Требования к системе электронного документооборота

Общие	Специфические
Возможность создания электронных документов (Ввод, Загрузка, Сканирование)	Возможность массовой загрузки документов в систему
Обеспечение связи между документами	Комментирование содержания документов
Возможность согласования документов и контроль их исполнения	Возможность интеграции в системы ERP\CAD\CRM\OCR

Продолжение таблицы 1.1

Общие	Специфические
Хранение и классификация по видам документов	Возможность сравнения содержимого различных документов
Контроль авторства для всех действий в системе, ограничение доступа и протоколирование несанкционированного доступа	

Рассматривая аналогичные системы в сравнении с системой, разработанной в рамках проекта можно вывести его актуальность. Для этого необходимо провести анализ рынка систем электронного документооборота.

Одной из самых популярных и распространенных систем является 1С:Документооборот. 1С: Документооборот — программа, разработанная российской фирмой «1С», предназначенная для автоматизации документооборота, для работы с данным программным продуктом требуется установленная технологическая платформа «1С: Предприятие 8». Данный программный продукт также локализован для Казахстана фирмой «1С-Рейтинг», основным функционалом данной программы является автоматизация делопроизводства, общий документооборот, управление договорной деятельностью, электронный архив документов и управление взаимодействием. Недостатком данной системы является ее дороговизна, обязательное наличие платформы «1С: Предприятие 8», каждая лицензия на каждого пользователя приобретается отдельно. При существенных экономических затратах на внедрение, такую систему будет необходимо обновлять, поддерживать работоспособность, обновлять и дорабатывать в случае нехватки типового функционала, при необходимости расширения количества рабочих мест потребуется установка дорогостоящего сервера либо аренды хостинга, что также влечет за собой дополнительные затраты.

DIRECTUM – система управления корпоративным контентом (Enterprise Content Management), включающая функционал системы электронного документооборота с возможностью взаимодействия пользователей на уровне прав доступа или должностей. Основным преимуществом данной системы является гибкий функционал, широкий спектр возможностей и возможность использования на различных платформах. Недостатком данной системы является стоимость и сложность внедрения данной системы, для настройки функционала потребуется внешняя экспертиза всех процессов организации.

Если взглянуть на рынок программных продуктов в сфере электронного документооборота в Республике Казахстан, можно выделить несколько разработок, составляющих конкуренцию зарубежным аналогам – это Documentolog, AlmexECM, ЕСЭДО и т.д.

Documentolog разработана в Казахстане, является адаптируемой и интегрируемой системой электронного документооборота с возможностью

интеграции со внешними информационными системами (Oracle, 1С, IBM и др.). Преимуществом данной разработки является возможность конфигурирования системы без навыков программирования, доступный графический интерфейс и возможность работы на различных устройствах.

Основные возможности программы:

1) Общий документооборот. ведение и обработка основных документов, принимающих участие в бизнес-процессе организаций. возможность их создания, отслеживания и удаления.

2) Договорные отношения. отслеживание движения и автоматический процесс обработки документов, упрощение и улучшение прозрачности договорной деятельности организации.

3) Управление кадрами. модуль, позволяющий оптимизировать управленческий процесс, автоматизировать работу с персоналом и документами кадровой службы.

При проведении сравнительного анализа, можно подвести итог, что у каждой разработанной системы есть свои преимущества и недостатки. Внедрение той или иной системы электронного документооборота в компанию требует анализа и выявления процессов, требующих мер которые позволят их ускорить. Для организации ТОО «Первый БИТ» данные системы не подойдут, так как, будет необходим долгий процесс внедрения с большим количеством затрат, для некоторых систем будет обязательно наличие мощного сервера, в котором будут храниться и обрабатываться данные, также данные системы обладают большим количеством встроенных функций, которые впоследствии не будут использоваться вследствие ненадобности. Также одним из условий для разработки являлась – «возможность учета заявок и выписка нарядов на выполнение работ» что оказывается специфичным для вида деятельности предприятия, вышеперечисленные системы данными возможностями в типовом функционале не обладают.

Информационная система электронного документооборота для организации ТОО «Первый БИТ» будет реализована в виде web-сайта, так как данный метод реализации позволит использовать систему с помощью различных устройств.

1.3 Постановка задачи

Предметной областью дипломного проекта является автоматизация процесса учета заявок и документов при выполнении проектов и сопровождении клиентов для сотрудников Отдела внедрения, Отдела сопровождения и Отдела продаж компании ТОО «Первый БИТ».

ТОО «Первый БИТ» международная компания, занимающаяся автоматизацией деятельности предприятий на платформе «1С: Предприятие», консалтинг, аналитика бизнес-процессов, финансовый анализ, продажа и установка компьютерной техники, монтаж компьютерных сетей, услуги по защите информации, услуги по разработке web-порталов, продвижению

сайтов, разработке сайтов. На данный момент компания является одной из крупнейших 1С-Франчайзи на рынке постсоветского пространства, имеет филиалы зарубежом (ОАЭ, Испания, Канада), основная цель компании на 2020 год – войти в топ-100 IT компаний мира. Стратегией компании является непрерывное обучение сотрудников, предоставление полного спектра IT-услуг, индивидуальный подход и долгосрочное сотрудничество с каждым из клиентов.

Основным функционалом отдела внедрения является сопровождение, внедрение и доработки по программным продуктам «1С: Предприятие 8», все конфигурации с которыми работает отдел внедрения являются доработаными что существенно увеличивает количество трудозатрат на обновление и поддержку данных систем. При подключении сопровождения клиент получает следующий спектр услуг:

1) Закрепленного за собой персонального менеджера и специалиста отдела внедрения;

2) Бесплатные консультации специалистов линии консультации по методическим вопросам, ведению учета, работе в программе;

3) Выезды технических специалистов вслучае возникновения сбоев в системе;

4) Согласованный график обслуживания и ежемесячный выезд специалиста по графикую.

При подключении данного вида сопровождения клиентам дается гарантия качественного непрерывного обслуживания, система всегда находится в актуальном состоянии, готовая к непрерывной работе.

Также процесс внедрения одного из видов системы проходит в несколько этапов, которые также проводятся специалистами отдела внедрения, это:

1) Обследование. Изучение текущих бизнес-процессов предприятия, проведение собеседования с пользователями системы и учет их требований к будущей системе. На основании требований подбирается программный продукт;

2) Моделирование. При моделировании определяются цели проекта, все сложные процессы организации заранее моделируются в типовом функционале программного продукта, если типового функционала недостаточно вводятся листы требований для доработки функционала программы.

3) Проектирование. На этапе проектирования пишется техническое задание, указываются четкие результаты которые будут получены после внедрения программы.

4) Внедрение. На данном этапе происходит само внедрение программного продукта с обучением пользователей.

Штат офиса в городе Алматы состоит из 35 человек, что является небольшим показателем для офисов компании по региону, но офис проводит непрерывный набор новых сотрудников и расширяется. Офис состоит из 5

отделов, каждый отдел состоит из специалистов различных категорий и руководителя. Общая структура предприятия представлена на рисунке 1.3.1.

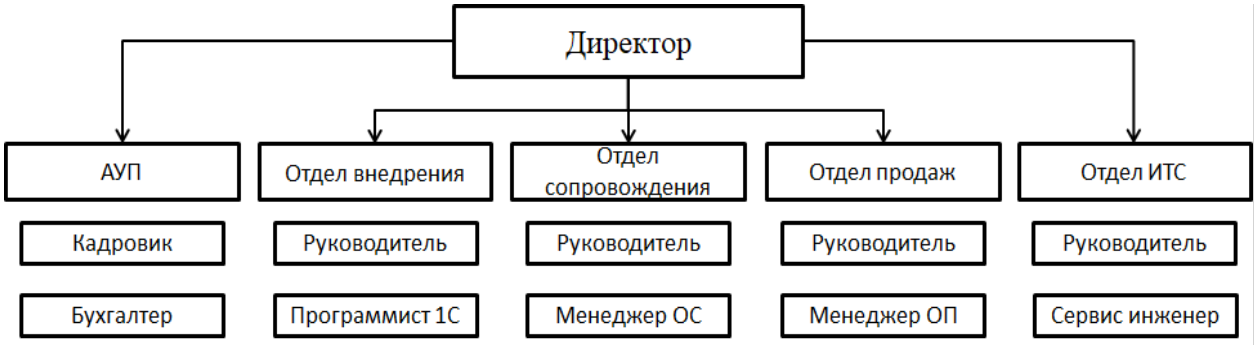

Рисунок 1.3.1 - Организационная структура предприятия

Сотрудники отдела сопровождения осуществляют связь с клиентами, проводят сбор и подготовку документов, определяют необходимость подключения клиентов к тем или иным специалистам по запросам пользователей, отвечают за согласование приема и оплаты работ. Сотрудники отдела продаж отвечают за первичную консультацию по выбору программных продуктов, проводят демонстрации их функционала, определяют необходимость и трудоемкость внедрения тех или иных видов программ.

Следует выделить следующие задачи дипломного проекта:

- подробный анализ предметной области;
- создание макета web-сайта;
- разработка серверной части сайта;
- разработка клиентской части сайта;
- наполнение веб-сайта данными.

2 Проектная часть

2.1 Инструментальные средства разработки

Для разработки сайта будет использоваться технология разработки MEAN (MongoDB, Express.js, Angular.js, Node.js). MEAN - это сквозной стек JavaScript, широко используемый для приложений, готовых к работе в облаке. Преимуществом данного типа разработки является то, что серверная и клиентская часть пишутся на одном языке программирования JavaScript.

Для хранения и обработки данных будет использоваться СУБД MongoDB. MongoDB - это кроссплатформенная, ориентированная на документы система управления базами данных, которая обеспечивает высокую производительность, высокую доступность и простоту масштабирования. MongoDB - это своего рода база данных NoSQL. Как база данных NoSQL, MongoDB избегает основанной на таблицах структуры реляционной базы данных для адаптации JSON-подобных документов, имеющих динамические схемы, которые она называет BSON. Это делает интеграцию данных для определенных типов приложений быстрее и проще.

Для проекта будет использоваться облачная база данных MongoDB Atlas. MongoDB Atlas – управляемая база данных MongoDB в виде сервиса. Это означает, что Atlas берет на себя ответственность за размещение, исправление, управление и защиту кластера MongoDB и предоставляет мощности своих серверов для эффективного использования базы данных.

Atlas позволяет установить MongoDB в несколько действий, никаких дополнительных настроек и администрирования не понадобится. Создать пользователей и распределить ограниченные права можно с помощью удобного пользовательского интерфейса.

Atlas также управляет увеличением/уменьшением кластера при необходимости и исправлением/обновлением кластера MongoDB при выпуске новой версии. Если вам нужны метрики, то есть хорошие метрики и оповещения, чтобы быть в курсе того, что происходит в базе данных.

MongoDB Atlas предлагает множество вариантов ценообразования и размеров, которые помогают снизить затраты при запуске, но позволяют быстро и легко увеличить кластер до рабочего размера при запуске приложений. Существует три уровня кластера MongoDB Atlas.

Общие кластеры - это самые дешевые и самые маленькие кластеры, которые идеально подходят для начала работы с MongoDB. Это кластеры экземпляров MongoDB, которые используются другими разработчиками MongoDB, но вы получаете свой зарезервированный фрагмент. Недостатком этого размера является то, что есть несколько функций, которые недоступны (например, пиринг VPC).

Выделенные Кластеры Развития. Этот размер кластера для новых приложений, выполняющих серьезную разработку. Это самый дешевый автономный размер кластеров MongoDB Atlas, что означает, что он отлично подходит для приложений, которые начинают получать большое количество

трафика. В них также включены все основные функции, что важно для разработки.

Выделенные производственные кластеры. Это автономные кластеры с большим количеством ресурсов, чтобы обеспечить бесперебойную работу компонента MongoDB в приложении.

Для разработки серверной части приложения будут использоваться Node.js и Express.js. Node.js - это кроссплатформенная среда выполнения, а также библиотека для запуска приложений JavaScript вне браузера. Узел используется для создания серверных и сетевых веб-приложений. При использовании даннного стека разработки Node.js является основной платформой исполнения приложения. Основным преимуществом Node.js является его высокая скорость работы, при эффективном использовании ресурсов скорость реакции сервера будет быстрой. Node.js не требует больших системных ресурсов, потому что он однопоточный, тогда как традиционные веб-серверы многопоточные.

Поскольку Node.js является платформой, он не предписывает то, как он должен быть настроен или использован. Это одна из его сильных сторон. Но при создании веб-сайтов и веб-приложений возникает довольно много общих задач, которые нужно выполнять каждый раз. Express.js- это инфраструктура для Node.js, со своими библитотеками, позволяющими ускорить процесс разработки. Одной из примечательных особенностей Express является то, что он предоставляет действительно простой интерфейс для направления входящего URL-адреса на определенный фрагмент кода. Express обеспечивает поддержку ряда различных шаблонизаторов, которые упрощают интеллектуальное создание HTML-страниц, используя встроенные компоненты, а также данные из приложения. Express компилирует их вместе и передает их браузеру в виде HTML.

За разработку интерфейса и вывода данных в браузер отвечает Angular.js. Angular.js это opensource JavaScript-фреймворк, использующий шаблон MVC. Использование данного шаблона позволяет разделять данные приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента: модель, представление и контроллер — таким образом, что модификация каждого компонента может осуществляться независимо.

Средой разработки был выбран IDE Microsoft Visual Studio Code. Visual Studio Code - это легкий, но мощный редактор исходного кода, доступен для Windows, macOS и Linux. Он поставляется со встроенной поддержкой JavaScript, TypeScript и Node.js и имеет богатую экосистему расширений для других языков (таких как C ++, C #, Java, Python, PHP, Go) и сред выполнения (таких как .NET и Unity). Встроенный функционал VS IntelliSense значительно упрощает разработку, всплывающие подсказки и автозамена по коду ускоряют процесс программирования. Также он поддерживает встроенный интегрированный терминал, в корне открытого проекта. Visual Studio Code имеет встроенную интеграцию с Git, что позволяет мгновенно увидеть

изменения, которые были внесены проект. Контроль изменений и версионирование позволяют вносить изменения в программу без боязни потери данных. Встроенный компилятор позволяет запускать отладку приложений без использования дополнительных программ, при внесении изменений отладка автоматически перезапускается что также удобно при разработке.

Таким образом, при разработке web-портала средой разработки будет выступать IDE Microsoft Visual Studio Code, Angular будет использоваться для отображения данных пользователям, на Node.js будут обрабатываться клиент-серверные запросы, express.js будет отвечать за запросы в базу данных и их быстродействие, MongoDB будет хранить и выдавать данные по запросам (рисунок 2.1.1).

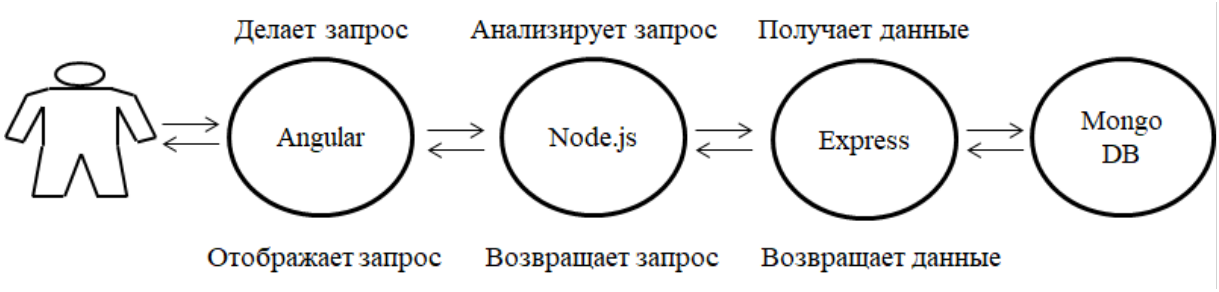

Рисунок 2.1.1 - Пример обработки запроса

2.2 Создание UML-диаграммы

UML, сокращенно от Unified Modeling Language (Унифицированный язык моделирования), является стандартизированным языком моделирования, состоящим из интегрированного набора диаграмм, разработанных, чтобы помочь разработчикам систем и программного обеспечения в определении, визуализации, конструировании и документировании частей программных систем, а также для создания бизнес-моделей.

Первое, что следует отметить в UML, это то, что есть много различных видов диаграмм (моделей). Это обусловлено тем, что систему необходимо рассматривать с разных точек зрения, так как в разработке программного обеспечения будут участвовать несколько заинтересованных сторон, представлены на рисунке 2.2.1.

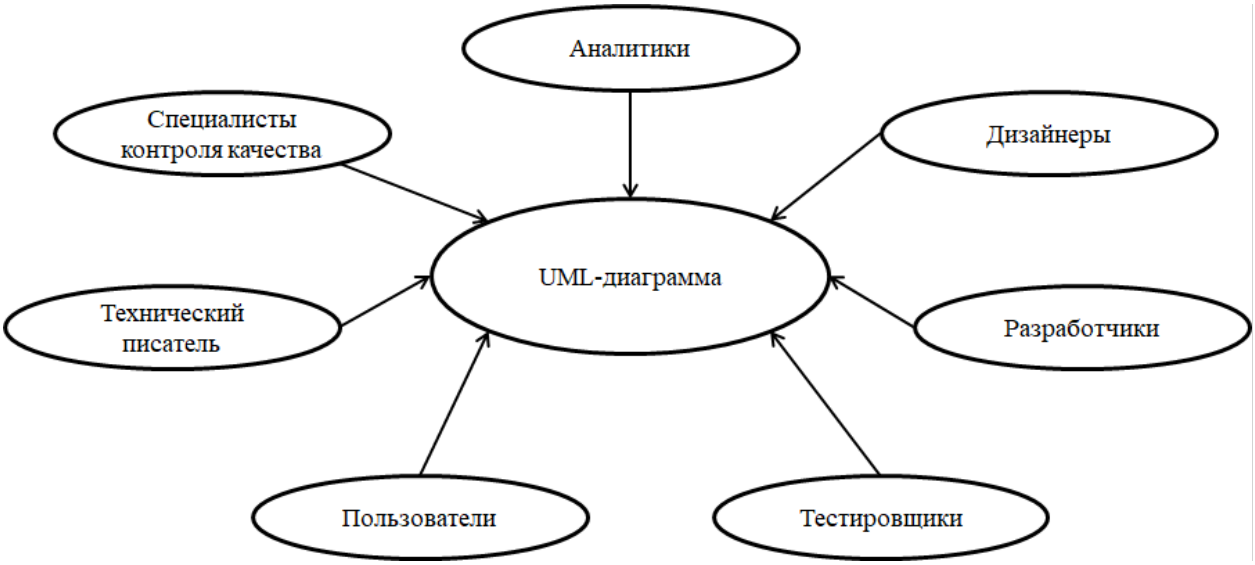

Рисунок 2.2.1 - Пользователи, использующие UML-диаграмму

Все эти люди заинтересованы в различных аспектах системы, и каждый из них требует разного уровня детализации. Например, разработчик должен понимать структуру системы для того чтобы можно было создать блок-схему работы системы. Напротив, технический писатель интересуется поведением системы в целом, и должен понимать как функционирует продукт. UML пытается предоставить язык, настолько универсальный, чтобы все заинтересованные стороны смогли найти исчерпывающую для себя информацию из нескольких диаграмм UML, заменяя огромное количество документации.

Структуру UML-диаграммы можно поделить на 2 основных составляющих это диаграммы структуры и диаграммы поведения. Структурные диаграммы показывают статическую структуру системы и ее частей на разных уровнях абстракции и реализации, а также их взаимосвязь. Диаграммы поведения показывают динамическое поведение объектов в системе, которое можно описать как серию изменений в системе с течением времени. Данные диаграммы делятся на свои подвиды, в общей сложности существует 13 видов диаграмм, все они представлены на рисунке 2.2.2.

Рисунок 2.2.2 - Виды UML-диаграмм

Для начала рассмотрим диаграмму вариантов использования (прецедентов). Диаграмма вариантов использования описывает функциональные требования системы с точки зрения вариантов использования. Это модель предполагаемой функциональности системы (варианты использования) и ее среды (субъекты). Варианты использования позволяют связать то, что необходимо от системы, с тем, как система удовлетворяет эту потребность.

Субъект взаимодействующий с системой также называется Actor (эктор, от слова действующий). Вариантами использования являются все действия, результат которых получает эктор. В разрабатываемой системе экторами являются менеджер и внедренец – пользователи, непосредственно взаимодействующие с программой (Рисунок 2.2.3).

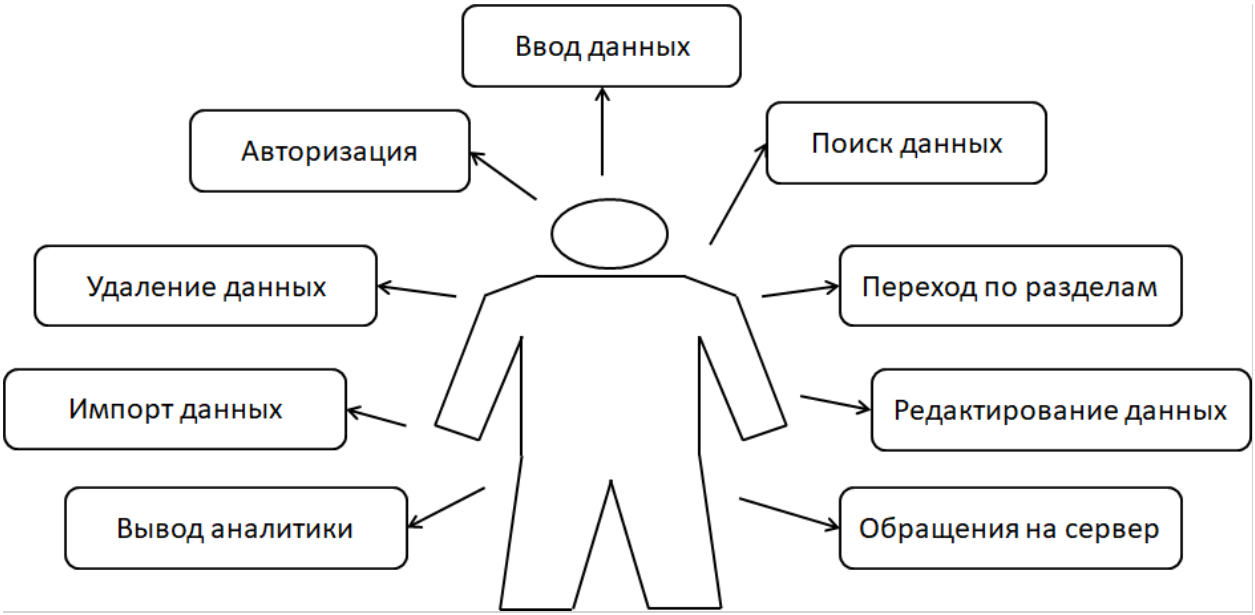

Рисунок 2.2.3 - Взаимодействие эктора с системой

После построения UML диаграммы вариантов использования, можно определить первоначальные требования к разрабатываемой системе.

Во-первых, нужно разработать страницу авторизации, с контролем доступа по ролям. Также при разработке страницы авторизации следует учесть моменты: под одним логином может зарегистрироваться только один пользователь, пароль должен содержать минимально допустимое количество символов, нужно обеспечить безопасную передачу данных пользователей, для этого необходимо предусмотреть возможность передачи и хранения паролей пользователей в зашифрованном виде.

Во-вторых следует разработать удобный функционал для работы с системой. Так-как система будет предусматривать несколько разделов, следует разработать удобное меню по которому пользователь сможет перемещаться между вкладками для быстрой навигации также можно добавить горячие клавиши.

Как видно в диаграмме, пользователь будет работать с некоторыми данными в системе, необходимо создать таблицы базы данных, реализовать возможность поиска данных в базе, продумать роуты между таблицами, наладить связь между серверной и клиентской частью приложения. Для работы с редактированием, поиском данных необходимо разработать присвоение уникального id для каждого элемента таблицы базы данных, также требуется установить возможность отслеживания авторства документа для контроля несанкционированных действий со стороны пользователей администратором.

Для дальнейшего удобства моделирования таблиц базы данных и настройки доступа ним в информационной системе, требуется определить основные составляющие системы и разработать словарь предметной области. Для данных целей подходит UML-диаграмма класса (Рисунок 2.2.4).

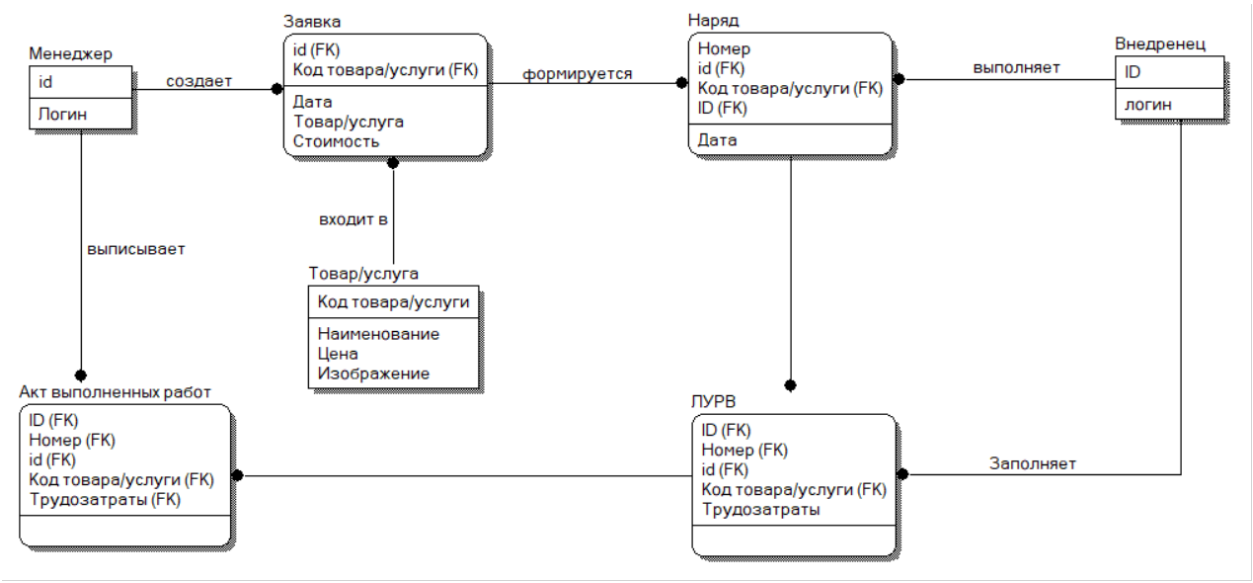

Рисунок 2.2.4 - Диаграмма классов для информационной системы

Основным функционалом разрабатываемой системы является электронный документооборот и учет заявок. При разработке электронного документооборота обязательным является моделирование взаимодействия объектов системы, для данной цели подойдет UML-диаграмма последовательностей (Рисунок 2.2.5).

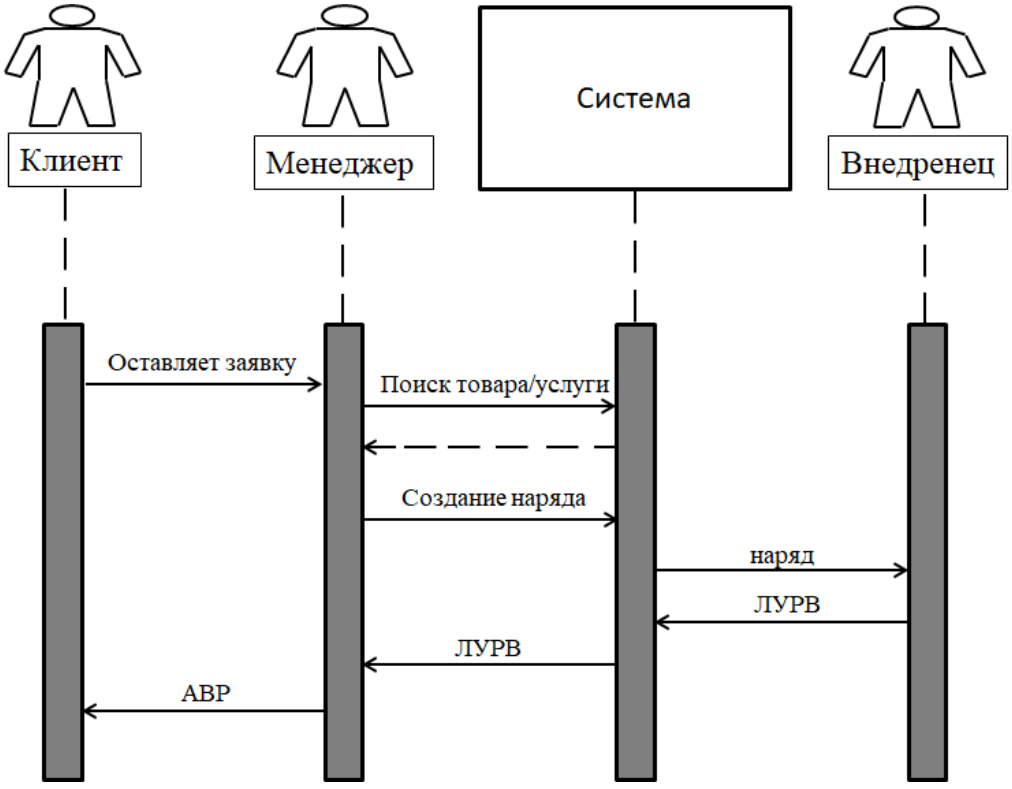

Рисунок 2.2.5 - Диаграмма последовательностей для предприятия

Структура процесса создания заявок и формирования документов происходит следующим образом, клиент оставляет заявку на приобретение услуг либо программных продуктов, менеджер обрабатывает данный запрос, ищет в базе данных клиента, программный продукт, далее создает наряд на выезд специалиста отдела внедрения (внедренец), специалист при завершении работы выписывает документ лист учета рабочего времени в котором описываются все выполненные работы и количество затраченного времени, менеджер находит данный документ в системе и создает на его основании документ акт выполненных работ для взаиморасчетов с клиентом.

2.3 Проектирование структуры web-портала

Информационная структура сайта позволяет разработать маршруты по которым сможет следовать пользователь, объединить весь функционал сайта в одну схему для наглядного представления проекта, это поможет найти и избавиться от лишнего функционала или добавить недостающие разделы для полной логической целостности сайта.

При грамотной настройке структуры сайта будет большая степень доверия к нему со стороны браузера, что посодействует его продвижению в поисковом списке. Моделирование структуры начинается с первого уровня, на первом уровне расположена главная страница сайта, далее следует расположить остальные уровни, полагаясь на логическое представление. При иерархической структуре пользователю будет легче находить необходимую информацию. Структуры сайтов бывают 3-х видов, это линейные, блочные и иерархические. При линейной структуре каждая страница сайта содержит ссылки на следующую, пользователь при работе с таким сайтом проходит весь путь от начала до конца по одному жестко заданному маршруту. Блочные структуры сайта разработаны таким образом что каждая страница сайта является одним блоком для главной страницы, данная структура удобна для сайтов презентующих какой-либо продукт, где каждый блок будет описывать его характеристику. Древовидные структуры являются одними из самых распространенных и наиболее универсальных видов структур, при использовании данного вида структур, для каждой страницы определено свое место в общей иерархии и ссылки на следующие страницы заданы согласно занимаемым уровням. Для проекта была выбрана древовидная структура сайта, на верхнем уровне будет находиться страница авторизации, так как в информационной системе будет два типа пользователей, для каждого будет разработана своя структура, на следующем уровне будет находиться главная страница, следующие страницы будут отвечать за вывод и ввод информации, на нижнем уровне находятся страницы элементов для выбора (рисунок 2.3.1).

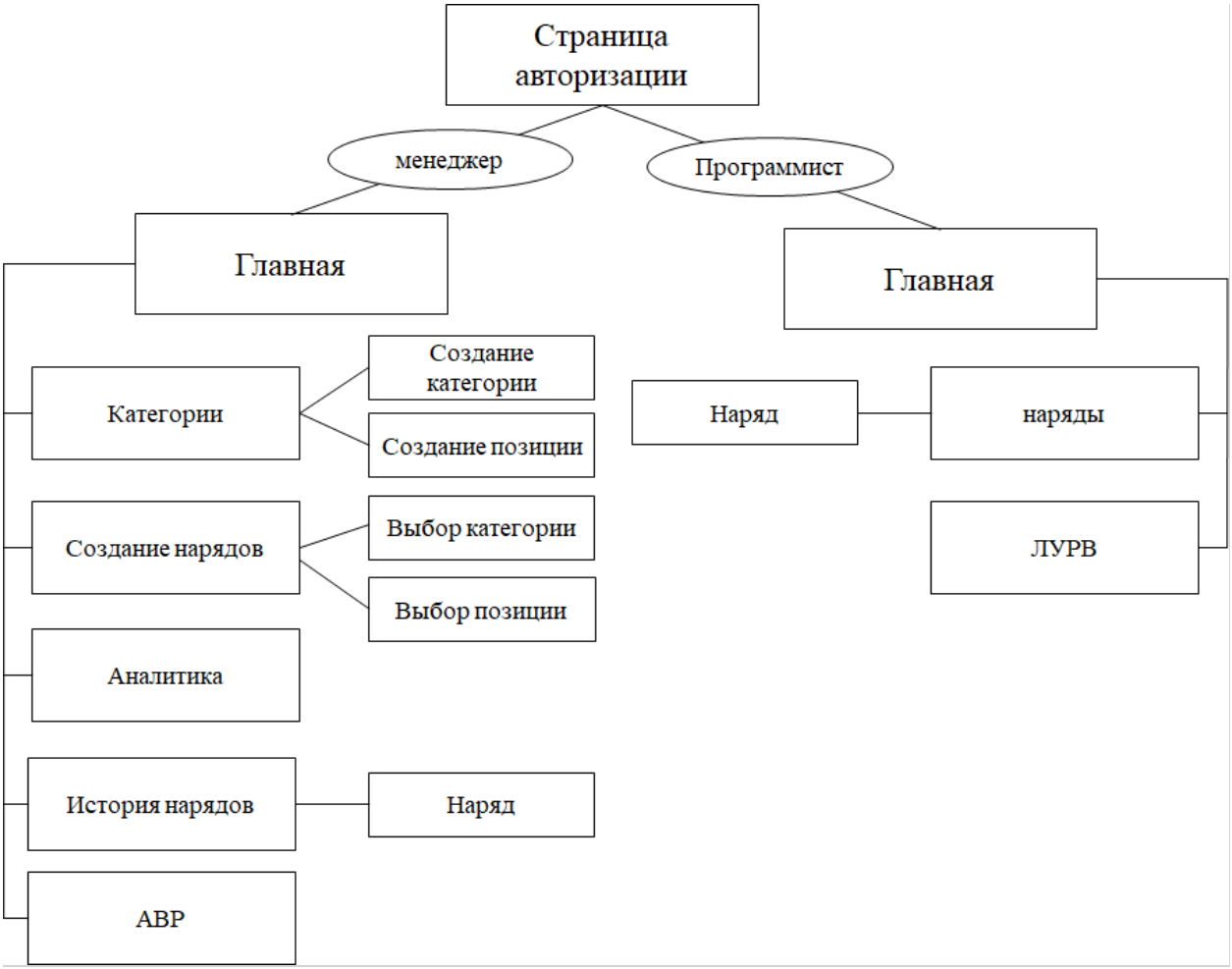

Рисунок 2.3.1 - Информационная структура web-портала

2.4 Визуализация алгоритмов и программного кода

Визуализация алгоритмов программного кода позволяет наглядно увидеть принцип работы заложенного алгоритма, тем самым происходит полное понимание процессов программы. Визуализация алгоритмов программного кода позволяет перейти от абстрактного представления работы приложения, к написанию правильно структурированного программного кода.

Для получения доступа к функционалу приложения пользователю необходимо авторизоваться в системе, данный шаг является первым в алгоритме работы приложения и представлен на рисунке 2.4.1.

.

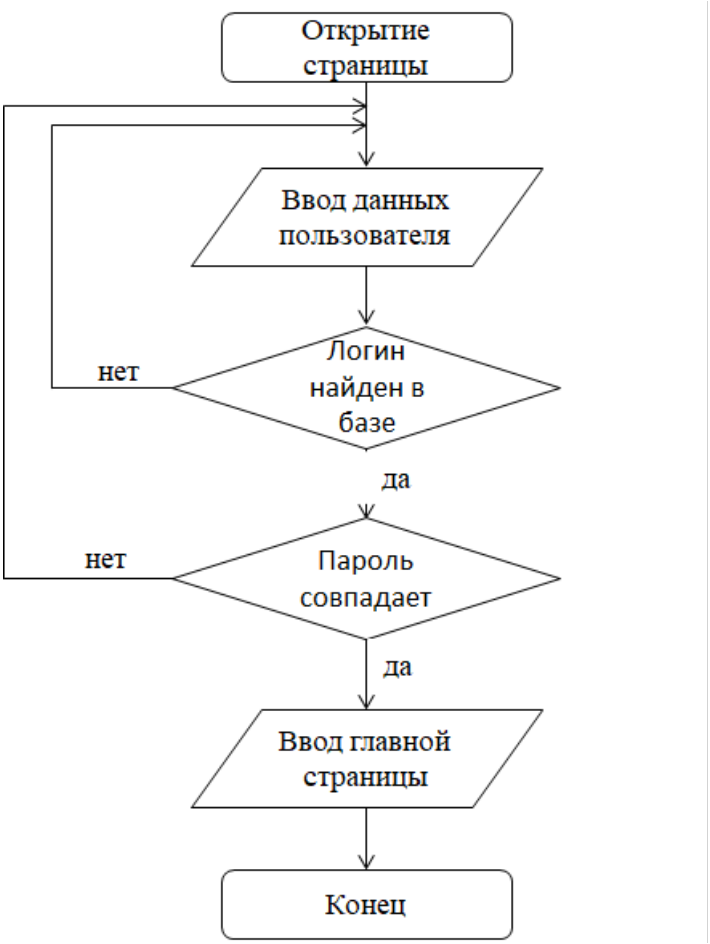

Рисунок 2.4.1 - Блок-схема алгоритма авторизации в системе

В данной блок-схеме изображены начало и остановка процесса, ввод-вывод данных, и проверка условий. Блок-схема имеет простую структуру, но позволяет наглядно увидеть процесс авторизации пользователя.

3 Экспериментальная часть

Model-View-Controller (MVC) - это архитектурный шаблон, который разделяет приложение на три основных логических компонента: модель, представление и контроллер. Каждый из этих компонентов создан для обработки определенных аспектов разработки приложения (рисунок 3.1). MVC является одной из наиболее часто используемых отраслевых сред разработки веб-приложений для создания масштабируемых и расширяемых проектов.

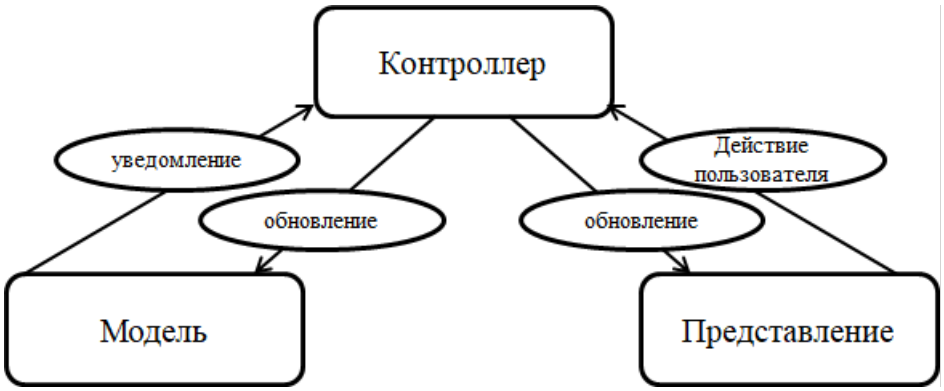

Рисунок 3.1 - Компоненты MVC–архитектуры

Модель - это место, где хранятся объекты данных приложения. Модель ничего не знает о представлениях и контроллерах. Когда модель изменяется, обычно она уведомляет своих наблюдателей о том, что произошло изменение.

Представление - это то, что видят пользователи и как пользователи взаимодействуют с приложением. Представление состоит из HTML, CSS, JavaScript шаблонов.

Контроллер является связующим звеном между моделью и представлением. Контроллер обновляет представление при изменении модели. Он также добавляет прослушиватели событий в представление и обновляет модель, когда пользователь управляет представлением.

Существует несколько вариантов шаблона проектирования MVC, таких как MVP (модель – вид – презентатор) и MVVP (модель – вид – видМодель). Даже с так называемым шаблоном проектирования MVC, существует некоторая разница между традиционным шаблоном MVC и современной интерпретацией в различных языках программирования. Например, некоторые основанные на MVC фреймворки будут иметь вид, наблюдающий изменения в моделях, в то время как другие позволят контроллеру изменять представление.

Подводя итог, можно сказать, что шаблон MVC обеспечивает модульность для разработчиков приложений и позволяет создавать многоразовый и расширяемый код и отделять логику представления от бизнес-логики.

3.1 Инициализация приложения

Для начала необходимо инициализировать приложение, это делается с помощью команды npm init (рисунок 3.1.1). Происходит инициализация проекта и создание файла package.json в данном файле будет храниться вся информация о проекте, также вся информация об установленных пакетах также будет находиться в данном файле. При вводе команды, система предложит заполнить несколько ключевых полей, которые будут инициализировать проект:

- название проекта;
- первоначальная версия;
- описание;
- точка входа;
- тестовые команды;
- репозиторий Git;
- ключевые слова;
- автор.

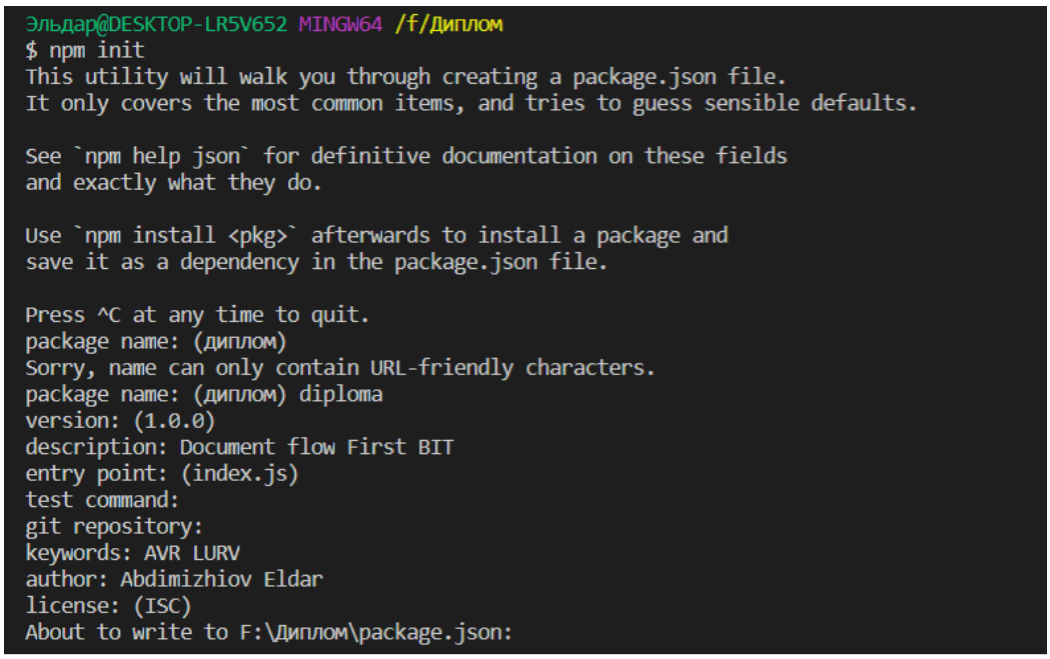

Рисунок 3.1.1 - Пример выполнения команды npm init

Далее при инициализации главным файлом был указан index.js, необходимо будет его создать. В данном файле будет вестись вся разработка. Также так как было решено использовать Express.js, необходимо будет подключить данный модуль к проекту, это делается командой npm install express (рисунок 3.1.2).

```
Эльдар@DESKTOP-LR5V652 MINGW64 /f/Диплом
$ npm i express
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN diploma@1.0.0 No repository field.

+ express@4.17.1
added 50 packages from 37 contributors and audited 50 packages in 16.642s
found 0 vulnerabilities
```

Рисунок 3.1.2 - Пример выполнения команды npm install express

После установки данного модуля, в корне проекта создается папка node_modules, в которой хранятся все пакеты для работы с express.js. В файле package.json появляется новое свойство Dependences, в нем отображаются все зависимости и модули, которые использует проект. После установки модулей можно подключить их к главному файлу index.js с помощью команды require. Команда require используется для загрузки модулей, с присвоением результата ее выполнения в какую-либо переменную (рисунок 3.1.3).

```
const express = require('express')
const app = express()
```

Рисунок 3.1.3 - Пример подключения пакета express

После инициализации проекта и подключения основного модуля производится тестовый запуск сервера для проверки корректности работы проекта. Команда для запуска app.listen, свойством которой является порт на котором будет запускаться сервер и console.log для просмотра журнала отладки в командной строке, при возникновении ошибок все ошибки можно будет отследить в данном журнале. Также необходимо включить обработку роута с помощью get запроса в корень приложения, параметром необходимо указать коллбэк функцию с параметрами request и response, request это запрос который отправляет клиент на сервер, response соответственно ответ от сервера клиенту (рисунок 3.1.4).

```
app.get('/', (req, res)=> {
    res.status(200).json({
        message: 'Working'
    })
})

app.listen(5000, ()=> {
    console.log('Server has been started')
})
```

Рисунок 3.1.4 - Пример настройки запуска сервера

В результате успешного выполнения команды, со статусом 200, будет выведено сообщение (рисунок 3.1.5).

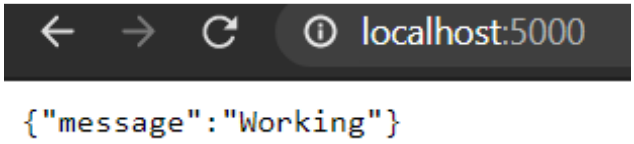

Рисунок 3.1.5 - Результат успешного запуска сервера

Для автоматической перезагрузки экспресс-сервера, при изменении какого либо файла, также необходимо установить пакет nodemon, в противном случае необходимо будет при внесении изменений останавливать и перезапускать сервер вручную (рисунок 3.1.6).

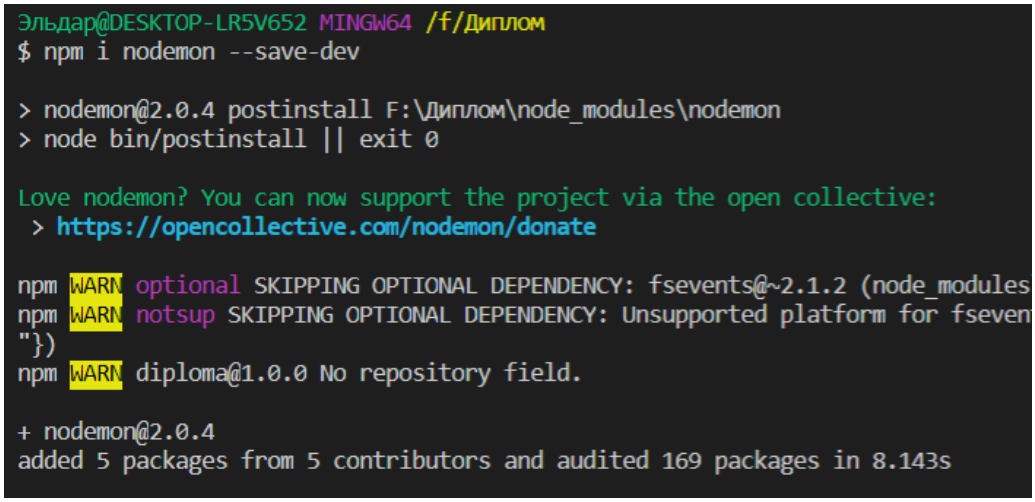

Рисунок 3.1.6 - Установка компонента nodemon

3.2 Создание роутов

Маршруты – одна из функций Express, обеспечивающая связь HTTP действия (GET, POST, PUT, DELETE, etc.), URL пути (шаблона), и функцию, которая обрабатывает этот шаблон[14].

Для начала будет создан роут авторизации auth.js, для начала подключается express, далее для переменной Router указывается пакет Express.Router который по сути является конструктором. Затем с помощью метода get задается сам маршрут.

Для регистрации роута в главном файле используется свойство app.use в котором указывается маршрут до роута на котором будет происходить определение функции обратного вызова при обнаружении HTTP- GET запроса определенного вида (рисунок 3.2.1).

Рисунок 3.2.1 - Пример создания роутов

Следующая сущность – контроллеры, в контроллерах будут описаны функции, которые будут выполняться в зависимости от запроса. Основной принцип работы в том, что при переходе на определенный адрес указанный в роуте, будет происходить выполнение функции, это является наглядным примером MVC архитектуры (рисунок 3.2.2).

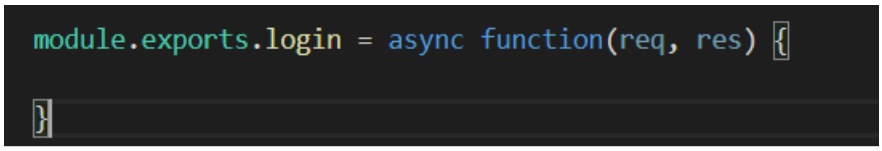

Рисунок 3.2.2 - Пример создания контроллера

На данном этапе будут добавлены все роуты которые требуются для работы web-приложения.

Таблица 3.1 – Перечень роутов, необходимых для работы базы

Наименование	HTTP-запрос	Путь	Действие
login	post	/auth	Вход
register	post	/auth	Регистрация
overview	get	/analytics	Обзор
analytics	get	/analytics	Аналитика
getAll	get	/order	Список всех
create	post	/order	Создание новой
getByCategoryId	get	/position	Просмотр позиции
create	post	/position	Создание позиции
update	patch	/position	Изменение позции
remove	delete	/position	Удаление позиции
getAll	get	/LURV	Список всех
create	post	/LURV	Создание нового
getAll	get	/AVR	Список всех

Продолжение таблицы 3.1

Наименование	HTTP-запрос	Путь	Действие
create	post	/AVR	Создание нового
getAll	get	/category	Список всех
getById	get	/category	Просмотр одного
remove	delete	/category	Удаление категории
create	post	/category	Создание новой
update	patch	/category	Изменение категории

3.3 Создание моделей

Для создания моделей потребуется установить фреймворк Mongoose. При установке Mongoose будут установлены зависимости для моделей базы данных и сами драйвера MongoDB, база данных подключается отдельно, либо с помощью установки на локальный компьютер или сервер, либо с помощью облачного хранилища (Рисунок 3.3.1).

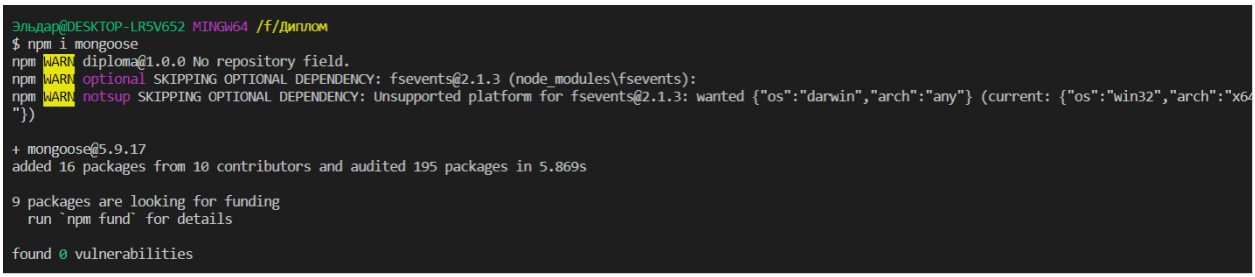

Рисунок 3.3.1 - Установка фреймворка Mongoose

Mongoose - инструмент объектного моделирования MongoDB, разработанный для работы в асинхронной среде. Он позволяет сопоставить объектную модель и базу данных документов.

Mongoose предоставляет невероятное количество функций для создания и работы со схемами. В настоящее время Mongoose содержит нескротко типов SchemaType, которые присваиваются объектам при сохранении в базе данных: строка, число, дата, смешанный, Buffer, ObjectId, массив и т.д.

Каждый тип данных позволяет указать:

1) значение по умолчанию;
2) пользовательская функция проверки;
3) указать поле, обязательное для заполнения;
4) функция get, которая позволяет вам манипулировать данными, прежде чем они будут возвращены как объект;
5) функция set, которая позволяет вам манипулировать данными до их сохранения в базе данных;
6) создавать индексы, позволяющие быстрее извлекать данные.

Свойства Number и Date поддерживают указание минимального и максимального значения, допустимого для этого поля.

Тип данных Buffer позволяет сохранять двоичные данные. Типичным примером двоичных данных может быть изображение или закодированный файл, такой как документ PDF.

Тип данных ObjectId обычно указывает ссылку на другой документ в вашей базе данных. Например, если у вас есть коллекция книг и авторов, документ книги может содержать свойство ObjectId, которое ссылается на конкретного автора документа.

Тип данных Array позволяет хранить JavaScript-подобные массивы. С типом данных Array можно выполнять общие операции с массивами JavaScript, такие как push, pop, shift, slice и т.д.

На примере представлена модель category, в первую очередь происходит подключение mongoose, с помощью выбора пакета из библиотеки. Затем создается схема, описывающая модель, с помощью переменной пакета mongoose - schema. Далее можно создать саму модель, в ней с помошью команды mongoose.model добавляем таблицу и передаем в нее ранее созданную схему. В схеме описываются поля, которые имеются в объекте category это name. imageSrc, user. У каждого поля имеется свой тип и различные модификаторы. У выбранного объекта поле user является ссылкой на коллекцию users, данное свойство можно указать с помощью ключевого слова ref (Рисунок 3.3.2).

```
const mongoose = require('mongoose')
const Schema = mongoose.Schema

const categorySchema = new Schema({
  name: {
    type: String,
    required: true
  },
  imageSrc: {
    type: String,
    default: ''
  },
  user: {
    ref: 'users',
    type: Schema.Types.ObjectId
  }
})

module.exports = mongoose.model('categories', categorySchema)
```

Рисунок 3.3.2 - Пример модели category

3.4 Создание и подключение базы данных

Для создания базы данных MognoDB в сервисе MongoDB Atlas нет необходимости устанавливать СУБД на локальный сервер, достаточно

зарегистрироваться на официальном сайте [8]. Для этого необходимо заполнить форму регистрации с указанием своих данных, организации и направление деятельности организации. Далее на главной странице сайта для добавления проекта необходимо воспользоваться командой New Project (Рисунок 3.4.1).

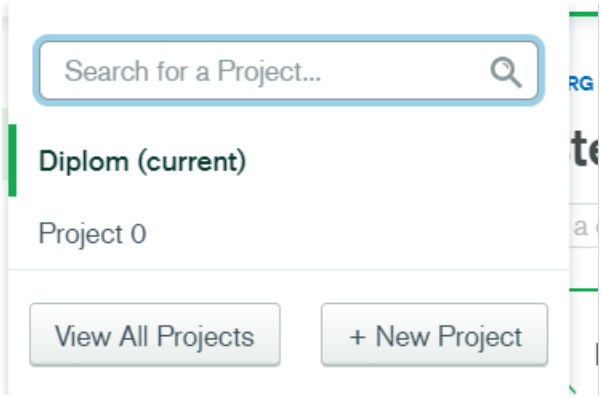

Рисунок 3.4.1 - Создание проекта MongoDB Atlas

В форме создания проекта необходимо дать название проекту и в случае использования совместного доступа к проекту можно подключить пользователей к проекту путем отправки приглашения им на почту. На основной панели формы появляется команда создания нового кластера, воспользовавшись ей будет создан новый кластер, далее указав пакет на выбор, необходимо указать облачного провайдера и наименование самого кластера. После проведения всех настроек будет создан новый кластер в течение 1–3 минут.

Для подключения к кластеру можно воспользоваться командой connect. В параметре подключения можно настроить IP адрес с которого можно будет подключаться к кластеру, это сделано в целях безопасности для защиты от несанкционированного подключения. В той же форме также нужно ввести параметры пользователя который будет иметь доступ к кластеру, логин и пароль (Рисунок 3.4.2).

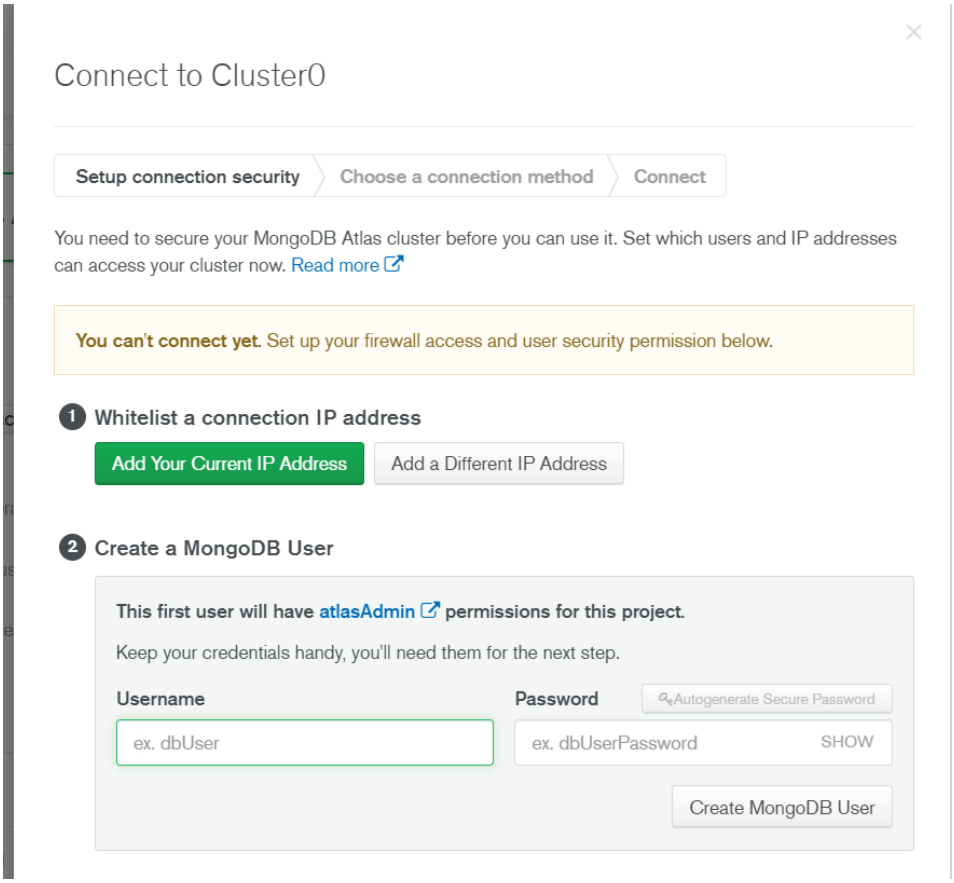

Рисунок 3.4.2 - Пример настройки безопасности кластера

После настройки подключения к кластеру, можно указать метод подключения к кластеру. Системой предусмотрено три вида подключения, это Mongo shell, application, MongoDB Compass.

Mongo shell - это интерактивный интерфейс JavaScript для MongoDB. Можно использовать оболочку mongo для запроса и обновления данных, а также для выполнения административных операций.

MongoDB Compass - приложение с графическим интерфейсом, которое устраняет необходимость подключения к оболочке Mongo или изучения синтаксиса запросов MongoDB. Он поставляется набором функций: есть возможность редактировать добавлять удалять свои данные, создавать конвейеры агрегации в интерактивном режиме, осуществлять проверку данных и оценивать производительность конкретной коллекции или базы данных.

При использовании MEAN стека указывается тип настройки connect your application. Указывается тип и версия драйвера. Далее появляется возможность использовать ссылку для подключения облачной базы данных к разрабатываемому приложению (Рисунок 3.4.3).

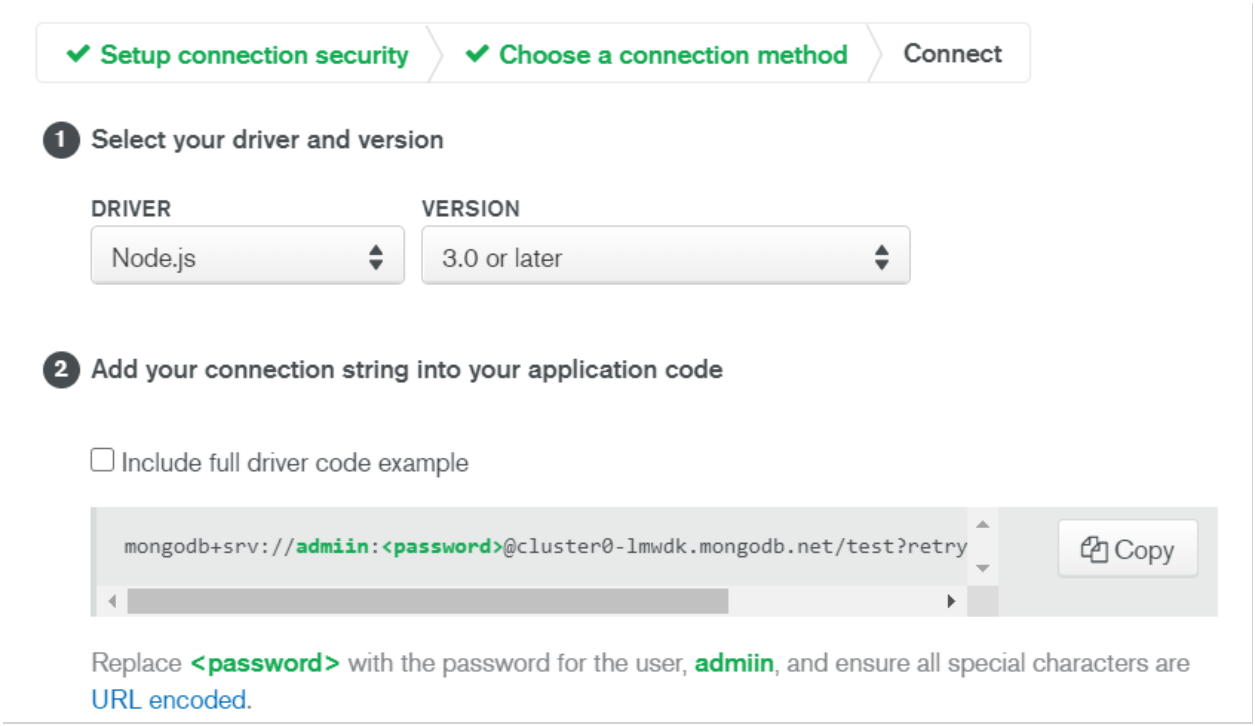

Рисунок 3.4.3 - Пример настройки подключения к кластеру

Для подключения базы данных необходимо воспользоваться инструментом Mongoose и в главном файле app.js указать данный инструмент (Рисунок 3.4.4).

```
app.js > mongoose
1   const express = require('express')
2   const bodyParser = require('body-parser')
3   const mongoose = require('mongoose')

mongoose.connect(keys.mongoURI, {
    useNewUrlParser: true,
    useUnifiedTopology: true,
    useCreateIndex: true
    })
    .then(() => console.log('MongoDB connected'))
    .catch(error => console.log(error))
```

Рисунок 3.4.4 - Пример настройки подключения базы данных

После подключения Mongoose можно начать пользоваться данным функционалом, для этого в корне проекта необходимо создать файл keys.js в котором будет храниться настройка подключения к базе данных MongoDB Atlas. В параметре MongoURI будет указан адрес подключения к базе данных, логин и пароль администратора базы данных в зашифрованном виде (Рисунок 3.4.4).

Рисунок 3.4.4 - Пример настройки URI базы данных

3.5 Создание контроллеров

После подключения базы данных и создания моделей можно переходить к описанию функций контроллеров. Для контроллера авторизации предусмотрено 2 функции, login и register. Как было рассмотрено в блок–схеме при визуализации алгоритмов программного кода, при авторизации пользователь вводит логин и система проверяет его наличие в базе данных, далее проверяется пароль при успешной попытке идет переадресация на главную страницу сайта.

Для поиска в базе данных используется ключевое слово await для ожидания результата запроса, в запросе обращение идет к объекту User с помощью метода findOne поиск одного объекта в базе. В запросе передается введенный пользователем логин. Если по запросу найдено совпадение произойдет проверка пароля с помощью метода bcrypt, в случае если результат пустой будет вызван статус 404 – Not Found. При вводе корректного пароля произойдет генерация токена, при вводе ошибочного, будет вызван статус 401 – Unauthorized (Рисунок 3.4.4).

```
const candidate = await User.findOne({email: req.body.email})

if (candidate) {
  // Проверка пароля, пользователь существует
  const passwordResult = bcrypt.compareSync(req.body.password, candidate.password)
  if (passwordResult) {
    // Генерация токена, пароли совпали
    const token = jwt.sign({
      email: candidate.email,
      userId: candidate._id
    }, keys.jwt, {expiresIn: 60 * 60})

    res.status(200).json({
      token: `Bearer ${token}`
    })
  } else {
    // Пароли не совпали
    res.status(401).json({
      message: 'Неверный пароль'
    })
  }
} else {
  // Пользователя нет, ошибка
  res.status(404).json({
    message: 'Пользователь с данным email не найден.'
  })
}
```

Рисунок 3.5.1 - Пример создания контроллера login

3.6 Разработка пользовательского интерфейса

Для подключения angular к приложению необходимо воспользоваться командой npm install –g @angular/cli. Далее необходимо создать angular проект, для этого необходимо воспользоваться командой ng new client. При выполнении данной команды в корне проекта появляется директроия Client, для разработки клиентской части создаются свои файлы package.json с пакетами для angular.

Для разработки пользовательского интерфейса был использован Materialize CSS. Materialize - это библиотека компонентов пользовательского интерфейса, созданная с использованием CSS, JavaScript и HTML. Компоненты пользовательского интерфейса Materialize помогают создавать согласованные и функциональные веб-страницы и веб-приложения, придерживаясь современных принципов веб-дизайна, таких как портативность браузера и независимость от устройства.

Установка Materialize CSS возможна с помощью команды npm install materialize–css@next. После установки в файле package.json клиентской части станет доступен данный пакет. В клиентской части приложения в файле style.css для всего проекта необходимо подключить materialize чтобы можно было его вызвать для каждой страницы web–приложения (Рисунок 3.6.1).

Рисунок 3.6.1 - Подключение Materialize CSS

Перед тем как приступить к созданию пользовательского интерфейса необходимо настроить прокси, данная настройка нужна для того, чтобы не приходилось прописывать полные URL адреса серверной части приложения для клиентской части. При формировании запросов на клиентской части будут прописываться только API сервера, прокси позволит перехватывать данные запросы и для всех запросов указывать соответствующий URL (Рисунок 3.6.2).

Рисунок 3.6.2 - Настройка прокси

Для каждого компонента клиентской части необходимо настроить роутинг. Все маршруты настраиваются в файле app-routing module. В данный файл подключается декоратор @NgModule, который принимает объект метаданных, описывающий то, как скомпилировать шаблон компонента и как создать инжектор во время выполнения. Он идентифицирует собственные компоненты, директивы и каналы модуля, делая некоторые из них общедоступными, через свойство export, чтобы их могли использовать внешние компоненты. Все роуты подключаются в одном массиве Routes (Рисунок 3.6.3).

```
@NgModule({
  imports: [
    RouterModule.forRoot(routes)
  ],
  exports: [
    RouterModule
  ]
})
export class AppRoutingModule {
}
const routes: Routes = [
  {
    path: '', component: AuthLayoutComponent, children: [
      {path: '', redirectTo: '/login', pathMatch: 'full'},
      {path: 'login', component: LoginPageComponent},
      {path: 'register', component: RegistrPageComponent}
    ]
  },
```

Риснунок 3.6.3 - Настройка роутинга

Для создания новой страницы применяется команда ng generate component login-page --skipTests. При выполнении данной команды будет сгенерировано 3 компонента для страницы сайта (Рисунок 3.6.4).

```
CREATE src/app/login-page/login-page.component.html (29 bytes)
CREATE src/app/login-page/login-page.component.ts (284 bytes)
CREATE src/app/login-page/login-page.component.css (0 bytes)
UPDATE src/app/app.module.ts (410 bytes)
```

Рисунок 3.6.4 - Пример создания нового компонента

Каждый компонент отвечает за свою область применения в компоненте формата html, описывается сама разметка веб-страницы, в файле формата typescript выполняются основные функции angular.

TypeScript - это расширенный набор JavaScript, что означает, что он содержит все функции JavaScript, и некоторые дополнительные. Следовательно, любая программа, написанная на допустимом JavaScript, также будет работать так, как ожидается в TypeScript. TypeScript предлагает больше контроля над кодом через аннотации типов, интерфейсы и классы.

3.7 Условие и порядок работы web-портала

В текущем разделе будет описан порядок работы пользователя на web-портале, также будет проведен обзор дополнительных особенностей.

При входе на сайт первым делом пользователю будет необходимо авторизоваться в системе. Откроется начальная страница сайта с формой авторизации либо регистрации в системе (рисунок 3.7.1)

Для входа в программу пользователю необходимо ввести логин и пароль. Первым делом система проверит, наличие логина в базе, далее соответствие паролей.

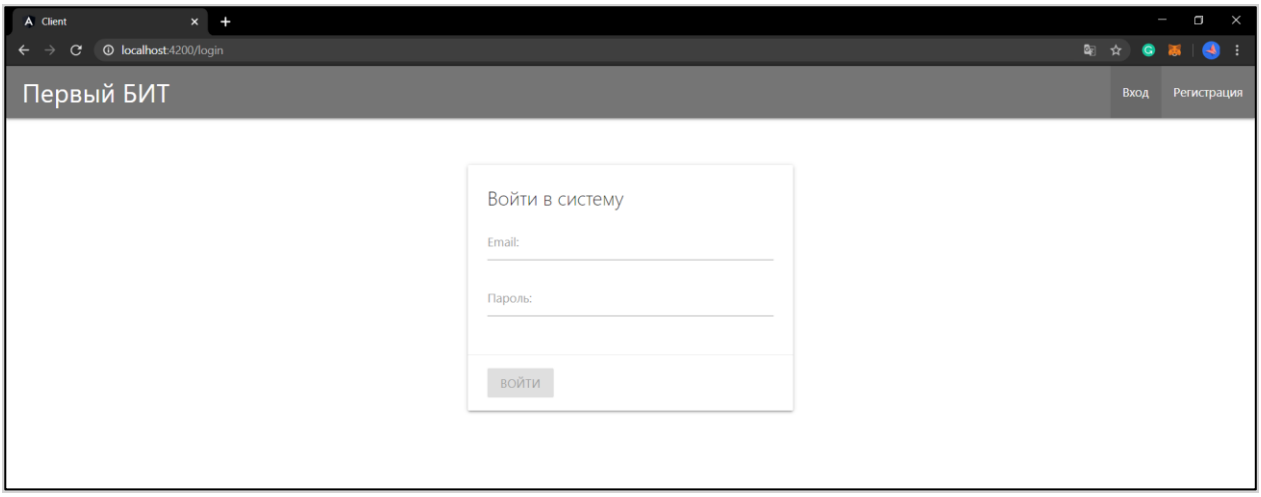

Рисунок 3.7.1 - Страница авторизации на сайте

В системе предусмотрен контроль строки, при вводе данных также происходит проверка длины пароля и формата ввода логина, в качестве логина выступает email (Рисунок 3.7.2).

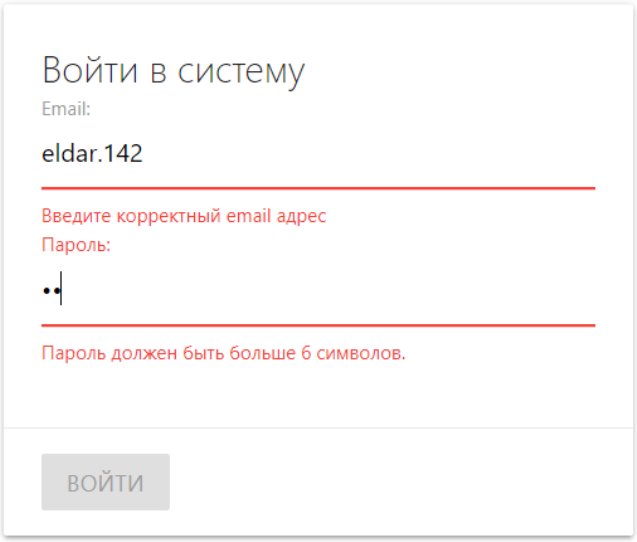

Рисунок 3.7.2 - Пример неудачной попытки авторизации

Также в функционал были добавлены токены, при входе на страницу пользователю выдается токен, действующий в течение часа, если проходит час работы пользователя, выводится предупреждающее сообщение и сеанс завершается (Рисунок 3.7.3).

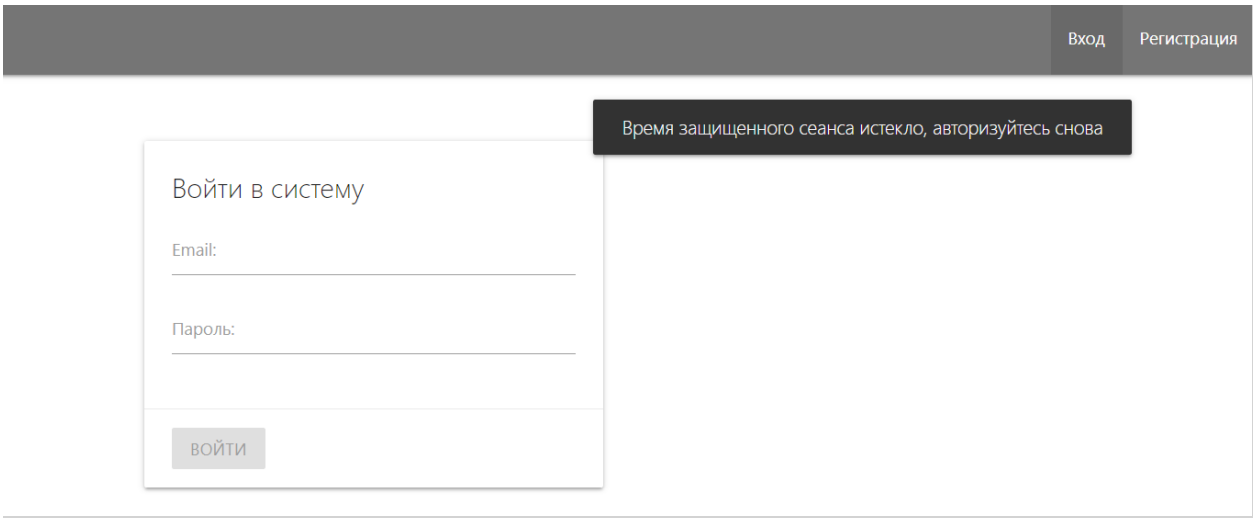

Рисунок 3.7.3 - Пример завершения действия сеанса

Если пользователь ранее не был зарегистрирован в системе, ему будет необходимо перейти в раздел регистрации и заполнить регистрационную форму. В форме пользователь вводит свой email адрес и пароль. В системе происходит проверка на уникальность введенного email адреса, в случае если

данный email был найден в базе, выйдет соответствующее информационное сообщение. Также пользователь при регистрации указывает свою роль в предприятии, является ли он менеджером, либо специалистом отдела внедрения (Рисунок 3.7.4)

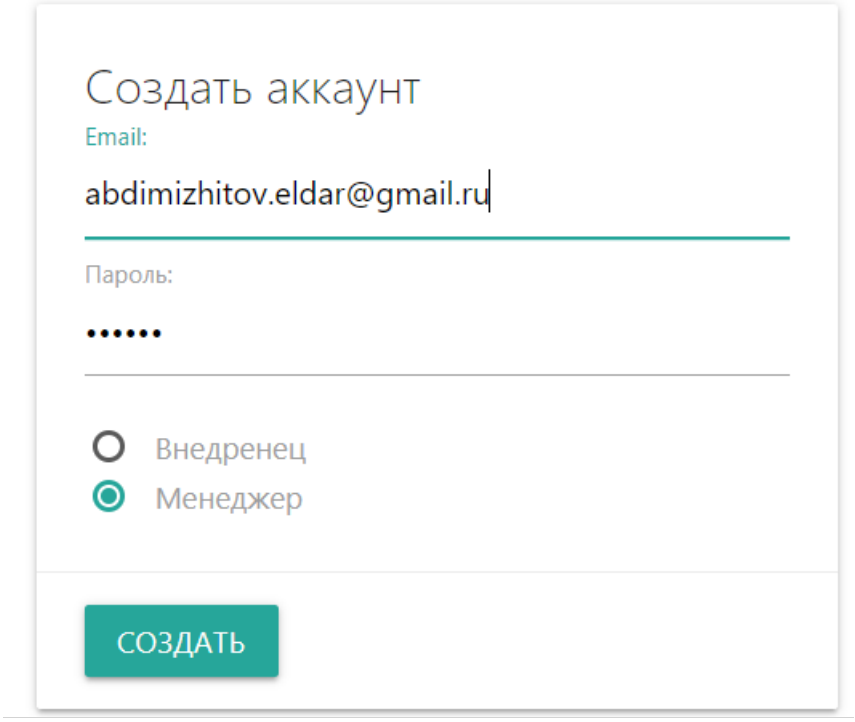

Рисунок 3.7.4 - Форма регистрации пользователя

Если пользователь зашел на сайт с правами менеджера, у него выйдет соответствующее меню разделов. Главной страницей сайта является панель обзора. На данной панели отображена статистика по количеству поступивших заявок от клиентов, с суммами реализаций клиентам за вчерашний день. Также при нажатии на кнопку получения дополнительной информации на экране отобразится всплывающее сообщение. Данная функция повышает лояльность пользователей системы, не неся первоочередную важность (Рисунок 3.7.4)

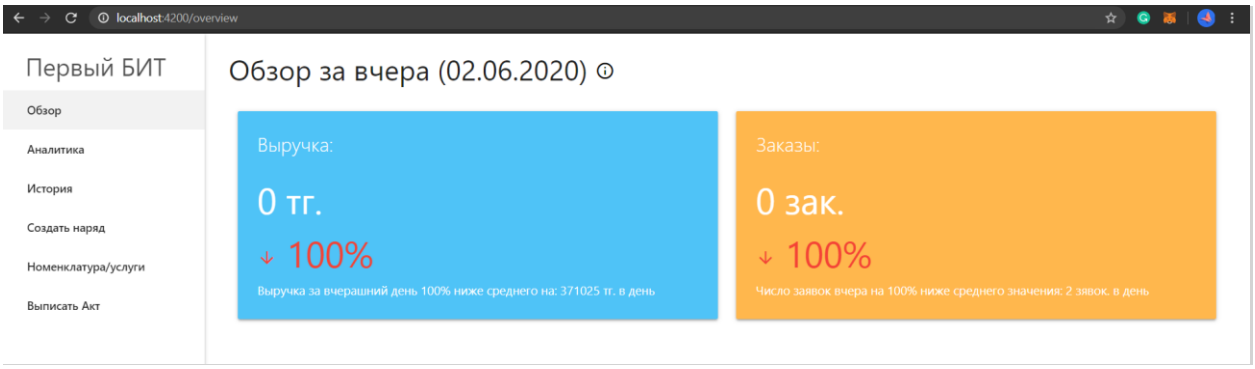

Рисунок 3.7.5 - Страница обзора за вчерашний день

В правом нижнем углу страницы отображается кнопка быстрой навигации в системе. С ее помощью можно быстро переместиться к странице создания наряда и добавления категории. Данная функция также была добавлена в целях дружественного интерфейса (Рисунок 3.7.6)

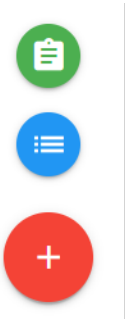

Рисунок 3.7.6 - Форма быстрой навигации

Для перехода по странице также можно воспользоваться меню, для этого необходимо в меню выбрать нужный раздел и кликнуть по нему левой кнопкой мыши. После выполнения данного действия откроется необходимая страница с соответствующими данными. На странице аналитики пользователь сможет увидеть графическую аналитику по нарядам и суммам в разрезе дней с выводом суммы среднего чека, данные показатели позволяют проанализировать эффективность работы менеджеров и платежеспособность клиентов за определенный период (Рисунок 3.7.7).

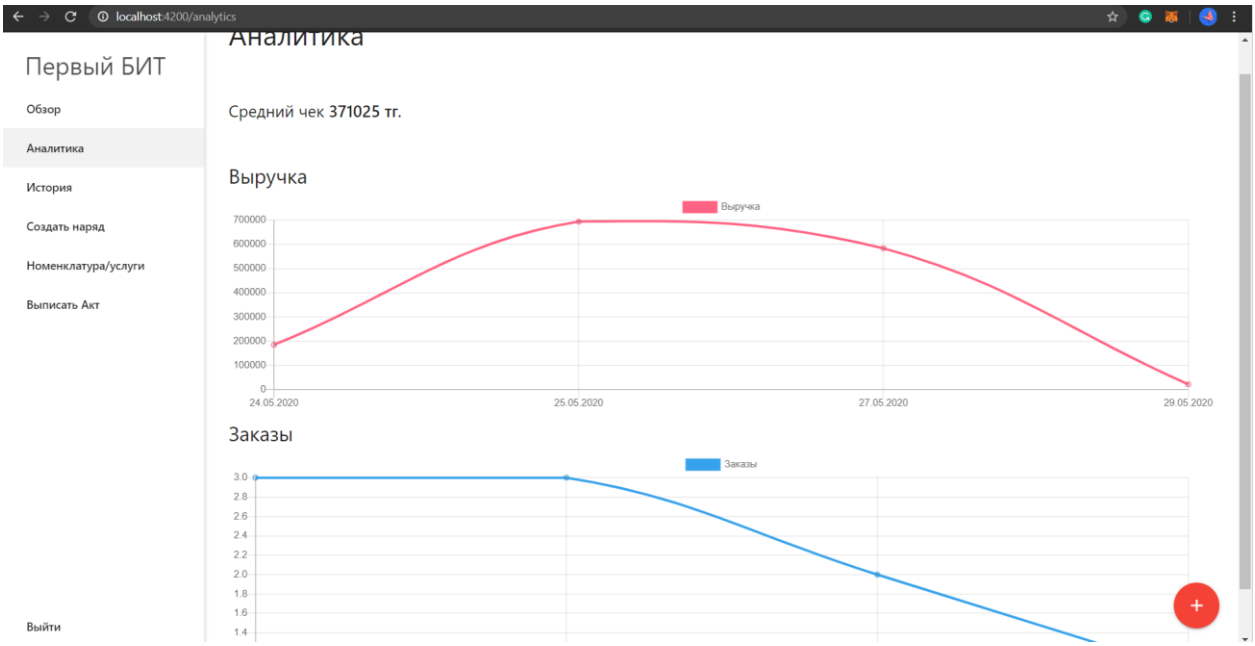

Рисунок 3.7.7 - Страница аналитики

На странице номенклатура/услуги пользователь может создать категории номенклатуры либо услуг указать их стоимость, при желании можно также загрузить изображение номенклатуры (Рисунок 3.7.8)

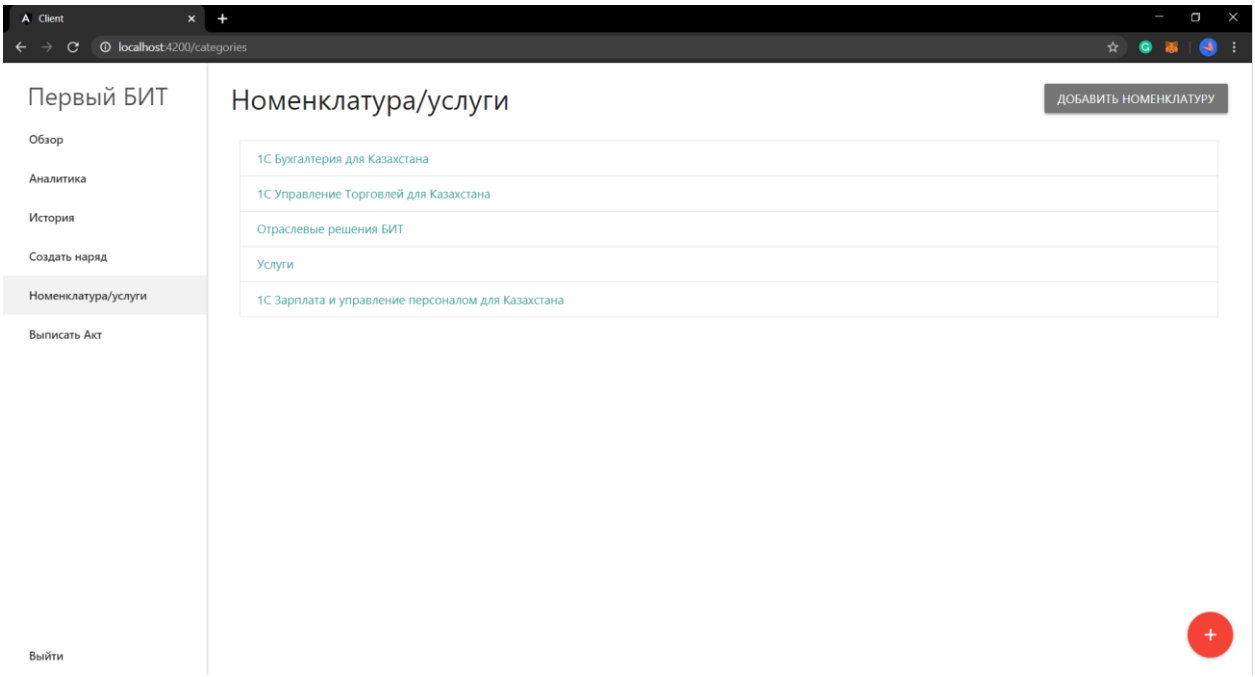

Рисунок 3.7.8 - Страница номенклатуры/услуг

На странице редактирования категории пользователь может внести изменения в категорию, создать позицию для данной категории, либо редактировать или удалить позицию для данной категории номенклатуры (Рисунок 3.7.10). При редактировании категории имеется возможность редактирования изображения категории (Рисунок 3.7.9).

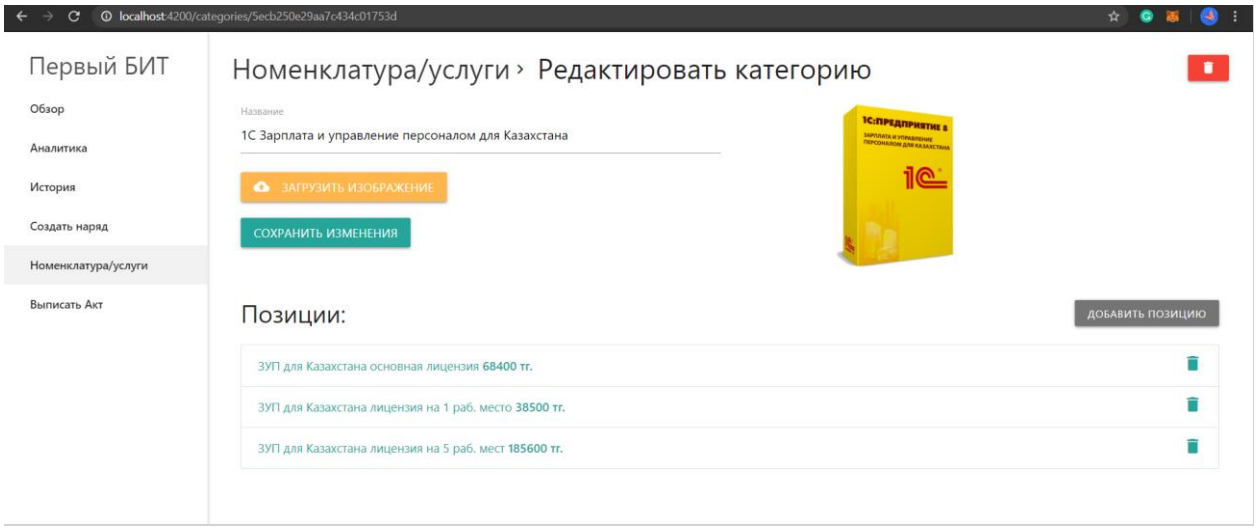

Рисунок 3.7.9 - Страница редактирования категории

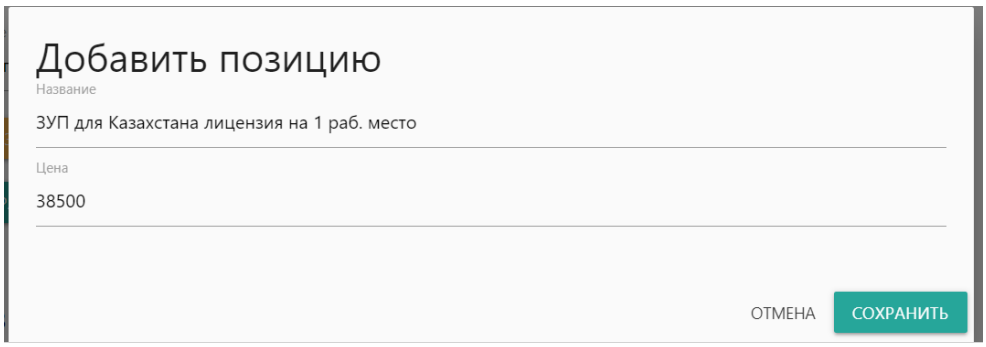

Рисунок 3.7.10 - Редактирование позиции

В разделе создание наряда пользователь создает наряд на специалиста отдела внедрения в котором указывает позицию, либо услугу (Рисунок 3.7.11).

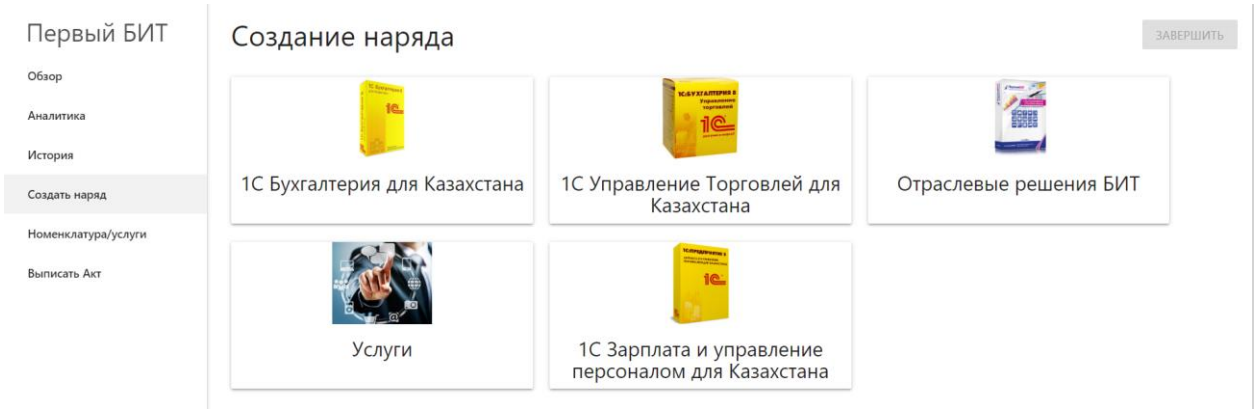

Рисунок 3.7.11 - Создание наряда на выполнение работ

Стоит отметить что даже при выборе позиции в стоимость каждого товара входит установка, так что наряд на сотрудника ОВ создается в любом случае.

4 Технико-экономическое обоснование

Темой дипломного проекта является Разработка web-портала «Документооборот IT-организации». Цель проекта – автоматизация процесса обмена документами Лист учета рабочего времени и Акт выполненных работ между сотрудниками ТОО «Первый БИТ». Использование данного web-портала позволит решить проблему подписи подтверждающих документов при выполнении работ на стороне заказчика, также решит проблему создания заявок на работу специалистов менеджерами компании.

В данном разделе будет произведен расчет затрат на реализацию продукта, расчет его себестоимости и определение экономической эффективности данного проекта. При расчете затрат будут учтены финансовые, трудовые и временные затраты на создание web-портала.

Для расчета себестоимости программы необходимо расчитать прямые и косвенные затраты на разработку ПП. Прямые затраты – затраты на разработку еденицы продукции, к ним можно отнести:

1) трудоемкость;
2) материальные затраты;
3) затраты на оплату труда;
4) социальный налог.

Косвенные затраты это затраты на выпуск партии или линейки продукции, к косвенным затратам отнесена амортизация основных фондов.

4.1 Трудоемкость разработки ПП

Таблица 4.1.1 – Распределение работ по этапам и видам и оценка их трудоемкости

Этапы разработки ПП	Вид работы на данном этапе	Трудоемкость разработки ПП	
		Чел. х час	Час х день
Анализ требований	Анализ предметной области, выявление целей и задач	1 х 16	8 х 2
Анализ рынка	Анализ конкурентов, подобных продуктов, потребность в данном продукте, нахождение преимуществ перед альтернативными товарами	1 х 24	8 х 2
Проектирование	Формирование требований к проекту, к интерфейсу, разработка и получение технического задания, индивидуальные требования руководства к продукту	1 х 32	8 х 4

Продолжение таблицы 4.2.1

Реализация	Построение диаграмм, реализация интерфейса, разработка ПО	1 х 32	8 х 4
Тестирование продукта	Тестирование, исправление ошибок и неполадок продукта	1 х 40	8 х 5
Подготовка инструкции	Подготовка подробной и понятной инструкции и руководства по работе с программой	1 х 16	8 х 2
Внедрение и поддержка	Установка программного обеспечения, исправление выявленных ошибок, сопровождение программного продукта.	1 х 32	8 х 4
Итого трудоемкость выполнения проекта		1 х 192	8 х 24

4.2 Расчет затрат на разработку ПП

Нужно рассчитать следующие затраты для составления:
- материальные затраты;
- затраты на оплату труда;
- социальный налог;
- амортизация основных фондов.

4.3 Материальные затраты

Затраты по материалам, необходимым для программного продукта, рассчитываются с помощью таблицы

Таблица 4.3.1 – Стоимость оборудования и ПО

№	Наименование	Описание	Цена за единицу, тг	Сумма, тг
1	Ноутбук	Lenovo Ideapad 110 80T7005URK	86 000	86 000
2	Операционная система	Microsoft Windows 10 Профессиональная	85 000	85 000
4	Антивирус	Kaspersky Anti-Virus 2020	6 200	6 200
5	Принтер	HP LaserJet M15a	41 000	41 000
Итого:				218 200

Также необходимо учесть затраты машинного времени на разработку информационной системы. Затраты машинного времени за время разработки ИС определяются по формуле:

$$З_м = К \times q \quad (4.1)$$

где К – количество часов использования ПК за время разработки ИС;
q – стоимость часа машинного времени (146 тенге /час) [9].

На разработку, внедрение и поддержку суммарно затрачено 192 часа. Исходя из этого получится:

$$З_м = 192 \times 146 = 28\,032 \text{ тг.}$$

4.4 Затраты на электроэнергию

Эта глава аудитории собой группа включает затраты на электроэнергию. полунина кабинетах соответствуют Общая сумма затрат одном разделением улучшены рассчитывается по формуле (4.1).

$$З_э = \sum_{i=1}^{n} М_i * K_i * T_i * Ц, \quad (4.2)$$

С 1 января 2020 года цена на электроэнергию по тарифу ТОО «АлматыЭнергоСбыт» составляет 19,17 тенге за 1 кВтч с НДС.

Таблица 4.4.1 – акустических портативной лк Затраты на технологические нужды

Наименование оборудования	Паспортная мощность, кВт	Коэффициент использования мощности	Время работы оборудования для разработки ПП, ч	Цена электроэнергиитг/кВт*ч	Сумма, тг
Ноутбук	0,5	0,8	192	19,17	1472,26
Принтер	0,4	0,8	8	19,17	49,08
Итого затраты на электроэнергию					1521,33

4.5 Затраты на оплату труда

В этой статье идет расчет оплаты труда всех работников, которые были задействованы в разработке программного продукта. Общая сумма затрат на оплату труда рассчитывается по формуле (4.3).

$$З_{тр} = \sum_{i=1}^{n} ЧС_i * T_i, \quad (4.3)$$

Часовая требующие кто полный ставка работника равняется – 650 тг тг/час.

Таблица 4.5.1 – Затраты на оплату труда

Категория работника	Трудоемкость разработки ПП, чел. х ч	Часовая ставка, тг/ч	Сумма, тг
Разработчик - программист	1 х 192	650	124 800
Дополнительная зарплата	192	650×10%=65	12 480
ИТОГО затрат на оплату труда:			137 280

4.6 Социальный налог

Отчисления на социальные нужды 10,46 % от затрат которых баннеры ограждающих на оплату труда всех работников, но пенсионные отчисления (10% от затрат) не облагаются данным видом налога.

Таблица 4.6.1 – Налоговые отчисления

Уплаченные налоги юридическим лицом	10,46	ФОТ	137 280
СО (Социальные отчисления)	3,5	(ЗП - ОПВ)×3,5%	4324,32
ВОСМСЮ (Отчисления на ВОСМСЮ)	2,0	ЗП×2%	2745,6
СН (Социальный налог)	9,5	(ЗП - ОПВ - ВОСМСФ) ×9,5%-СО	7282,704
Всего уплаченные налоги			14352,624

Амортизационные отчисления определяются согласно Таблице 4.7. Сумма амортизационных отчислений вычисляется по формуле (4.4).

$$З_{ам} = \frac{С_{обор} * Н_а * N}{100 * 12 * t} \quad (4.4)$$

где $Н_а$– норма амортизации (%);

Собор – первоначальная стоимость оборудования;

N – время использования оборудования;

t – количество рабочих дней в месяце.

Норма амортизации для линейного способа вычисляется пор по формуле (4.4).

Использование ОФ варьируется от 3 до 10 лет. Все используется в течении 7 лет. Программное обеспечение – 3 года. Используя формулу (4.4), заполним Таблицу 4.6 для отображения амортизации основных фондов.

$Н_{А1}$= 100/7 = 14,29%.
$Н_{А3}$= 100/3 = 33,33%.

Расчеты контактная решить фактического амортизации:
Зам= (86 000 × 0,1429 × 24)/(1 × 12 × 21) = 1170,42 тг;
Зам= (85 000 × 0,3333 × 24)/(1 × 12 × 21) = 2698,14 тг;
Зам= (6 200 × 0,3333 × 24)/(1 × 12 × 21) = 196,81 тг;
Зам= (41 000 × 0,1429 × 1)/(1 × 12 × 21) = 23,25 тг.

Таблица 4.6.2 – Затраты на оборудование

Наименование оборудования и ПО	Стоимость оборудования и ПО, тг	Годовая норма амортизации, %	Время работы оборудования и ПО для разработки ПП, д	Сумма, тг
Lenovo Ideapad 110 80T7005URK	86 000	14,29	24	1170,42
Microsoft Windows 10 Профессиональная	85 000	33,33	24	2698,14
Kaspersky Anti-Virus 2020	6 200	33,33	24	196,81
HP LaserJet M15a	41 000	14,29	1	23,25
			Итого:	4162,95

Величина затрат на материалы на основании исходных данных определяется по формуле:

$$Mi = \frac{З_{осн} \cdot Н_{мз}}{100} \quad (4.5)$$

где $Н_{мз}$ - норма расхода материалов от основной заработной платы (3-5%).

Величина затрат на материалы при норме расхода – 3%:

$$M_i = \frac{137\ 280 \cdot 3}{100} = 4\ 118{,}4 \text{ тенге}$$

Научные командировки не предусмотрены.

Расходы по статье «Прочие затраты» (Пзi) на конкретное ПО включают затраты на приобретение и подготовку специальной научно-технической информации и специальной литературы. Определяются по нормативу,

разрабатываемому в целом по организации, в процентах к основной заработной плате:

$$П_{зi} = З_{оi} \cdot \frac{Н_{пз}}{100} \quad (4.6)$$

где $Н_{пз}$ - норматив прочих затрат в целом по организации в (%), в дипломной работе нужно брать 20%.

$$П_{з} = 137\,280 \cdot 0{,}2 = 27\,456 \text{ тенге}$$

Затраты по статье «Накладные расходы» ($Р_{нi}$), связанные с необходимостью содержания аппарата управления, вспомогательных хозяйств и опытных (экспериментальных) производств, а также с расходами на общехозяйственные нужды ($Р_{нi}$), относятся на конкретное ПО по нормативу ($Ннр$) в процентном отношении к основной заработной плате исполнителей. Норматив устанавливается в целом по организации:

$$Р_{нi} = З_{оi} \cdot \frac{Н_{нр}}{100} \quad (4.7)$$

где $Рнi$ - накладные расходы на конкретную ПО (тыс. тенге);

$Ннр$ - норматив накладных расходов в целом по организации в (%), в дипломной работе нужно брать 70%.

$$Р_{нi} = 137\,280 \cdot 0{,}7 = 96\,096 \text{ тенге}$$

4.7 Смета затрат на разработку ПП

Таблица 4.7.1 – Смета затрат на разработку ПП

Статья затрат	Сумма, тг
Стоимость оборудования и ПО	218 200
Оплата труда	137 280
Социальный налог	14 352,62
Электроэнергия	1 521,33
Амортизация основных фондов	4 088,62
Затраты на машинное время	28 032
Накладные расходы	96 096
Прочие расходы	27 456
Материальные затраты	4 118,40
ИТОГО по смете	531 145

Рентабельность и прибыль по создаваемому ПО (Псi) определяются исходя из результатов анализа рыночных условий, переговоров с заказчиком (потребителем) и согласования с ним отпускной цены, включающей дополнительно налог на добавленную стоимость. В случае разработки ПО для использования внутри организации оценка программного продукта производится по действующим правилам и показателям внутреннего хозрасчета (по ценам, устанавливаемым для расчета за услуги между подразделениями). Прибыль рассчитывается по формуле:

$$П_{Oi} = C_{ni} \cdot \frac{У_{рni}}{100} \quad (4.8)$$

где ПОi - прибыль от реализации ПО заказчику (тыс. тенге);

Урni - уровень рентабельности ПО (%), в дипломной работе брать 40-60%;

Cni - себестоимость ПО (тыс. тенге).

$$П_{Oi} = 531\,145 \cdot 0.5 = 265572{,}5 \text{ тенге}$$

Прогнозируемая цена ПО без налогов (Цпi):

$$Ц_{пi} = C_{ni} + П_{Oi} = 531\,145 + 265572{,}5 = 796717{,}5 \text{ тенге}$$

Прогнозируемая отпускная цена *($Ц_{оl}$):*

$$Ц_{О1} = Ц_{ni} + НДС \quad (4.9)$$

Ставка налога на добавленную стоимость НДС в РК на 2020 год составляет 12% от отпускной цены ПО.

$$Ц_{О1} = 796717{,}5 + \frac{796717{,}5 \cdot 12}{100} = 892323{,}6 \text{ тенге}$$

Организация-разработчик участвует в освоении ПО и несет соответствующие затраты, на которые составляется смета, оплачиваемая заказчиком по договору. Затраты на освоение определяются по нормативу ($Н_О$=*10%)* от себестоимости ПО в расчете на 3 месяца и рассчитываются по формуле:

$$P_{Oi} = C_{ni} \cdot \frac{Н_О}{100} = 531\,145 \cdot 0{,}1 = 53114{,}5 \text{ тенге}$$

Затраты на сопровождение ПО ($Р_{Ci}$). Организация-разработчик осуществляет сопровождение ПО и несет соответствующие расходы, которые

оплачиваются заказчиком в соответствии с договором и сметой на сопровождение. Затраты на сопровождение определяются по установленному нормативу (НС=*20%)* от себестоимости ПО (в расчете на год) и рассчитываются по формуле:

$$P_{Ci} = C_{ni} \cdot \frac{\text{Н}_\text{С}}{100} = 531\,145 \cdot 0{,}2 = 106229 \text{ тенге}$$

Капиталовложения программного обеспечения с учетом затрат на освоение и сопровождение будет:

$$K = 892323{,}6 + 53114{,}5 + 106229 = 1051668 \text{ тенге.}$$

4.8 Оценка эффективности внедрения программных средств

Экономическая эффективность была рассчитана программистом-разработчиком. Затраты на решение задачи без использования программного средства рассчитываются по формуле (4.5):

$$\text{З}_{\text{тр}} = \text{ФЗП}_\text{р} + \text{ОТ}_{\text{з/п}} \quad , \qquad (4.10)$$

где $\text{ФЗП}_\text{р}$ – фонд заработной платы работников, решающих данную задачу;

$\text{ОТ}_{\text{з/п}}$ – отчисления на социальные нужды (10,46%) [1].

Фонд заработной платы работников рассчитывается по формуле (4.6):

$$\text{ФЗП}_\text{р} = \text{ЗП}_\text{р} * N * 12 \quad , \qquad (4.11)$$

где $\text{ЗП}_\text{р}$ – оклад работника, тенге/месяц;

N – количество работников.

Оклад работника составляет 135 000 тенге в месяц.

Исходя из этого, фонд заработной платы сотрудников за год трафика составляет:

$$\text{ФЗП}_\text{р} = 135\,000 * 12 = 1\,620\,000 \text{ тг.}$$

$$\text{uml постоянное ассоциируют ся } \text{ОТ}_{\text{з/п}} = 1\,620\,000 * 10{,}46\% = 152\,506{,}8 \text{ тг.}$$

Далее рассчитываются затраты на решение задач без использования программного продукта:

$$\text{З}_{\text{тр}} = 1\,620\,000 + 152\,506{,}8 = 1\,772\,506{,}8 \text{ тг.}$$

Годовые затраты машинного времени определяются по формуле (4.12):

$$З_м = K * q * 12 \quad , \qquad (4.13)$$

где К количество часов использования ПК в месяц;

q – стоимость часа аренды сервера (146 тенге/час) [9].

С учетом 8 часового рабочего дня, а также 21 рабочего дня в месяц, получаем часы использования ПК в месяц К=168 часов. Исходя из этого получится:

$$З_м = 168 * 146 * 12 = 294\,336 \text{ тг.}$$

Суммарные затраты после внедрения программного продукта ($З_{ом}$) составят 294 336 тг.

Экономия затрат от внедрения программного продукта определяется по формуле:

$$Э = З_{тр} - З_{ом} \quad , \qquad (4.14)$$

где $З_{тр}$ – затраты до внедрения системы;

$З_{ом}$ – затраты после внедрения системы [9].

Подставив значения получим следующее:

$$Э = 1\,772\,506{,}8 - 294\,336 = 1\,478\,170{,}8 \text{ тг.}$$

Так как разработанная информационная система несет экономический эффект, целесообразно оценивать его эффективность за счет экономии в сравнении с предыдущим периодом работы без использования ИС.

Величина ожидаемого годового экономического эффекта от внедрения ИС рассчитывается по формуле:

$$Э_г = Э_{уг} - К \cdot E_Н \qquad (4.15)$$

где $Э_г$ - ожидаемый годовой экономический эффект, тенге:

$Э_{уг}$ — ожидаемая условная годовая экономия, тенге;

К — капитальные вложения, тенге;

$E_Н$ - нормативный коэффициент экономической эффективности капитальных вложений.

Нормативный коэффициент экономической эффективности капитальных вложений определяется по формуле:

$$E_Н = \frac{1}{T_Н} \qquad (4.16)$$

где $Т_н$ — нормативный срок окупаемости капитальных вложений, лет.

Нормативный срок окупаемости капитальных вложений принимается исходя из срока морального старения -технических средств и проектных решений ИС (Тн=1,2,3...n), для программных продуктов срок окупаемости принимаем равным 4 года.

$$E_H = \frac{1}{4} = 0{,}25$$

$$Э_г = 1\,478\,170{,}8 - 1051668 \cdot 0{,}25 = 1215254 \text{ тенге}$$

Расчетный коэффициент экономической эффективности капитальных вложений составляет:

$$E_р = \frac{Э_{уг}}{К} \qquad (4.17)$$

где Ер - расчетный коэффициент экономической эффективности капитальных вложений;

Эуг — ожидаемая условная годовая экономия, тенге;

К — капитальные вложения на создание системы, тенге.

$$E_р = \frac{1\,478\,170{,}8}{1051668} = 1{,}41$$

Расчетный срок окупаемости капитальных вложений составляет:

$$Т_Р = \frac{1}{E_Р} \qquad (4.18)$$

где Ер - коэффициент экономической эффективности капитальных вложений.

$$Т_Р = \frac{1}{1{,}41} = 0{,}72 \text{ года} \approx 8{,}6 \text{ месяцев}$$

Таблица 4.8.1 – Показатели экономической эффективности внедрения программного продукта

Наименование показателей	Значение
Условная годовая экономия затрат, тенге	1215254
Коэффициент экономической эффективности капитальных вложений (Ер)	1,41
Срок окупаемости капитальных вложении (Тр)	0,72

4.9 Вывод по технико-экономической части

Таким образом, разработанная информационная система позволила не только упросить процесс управления работой организации и процесс принятия решений руководством, но и значительно сэкономить на различных факторах. Во-первых, автоматизация процесса управления позволит больше не использовать услуги дополнительных кураторов, так как теперь отсутствует необходимость осуществлять контроль непосредственно на месте филиала, вся работа организации сосредоточена в ИС. Во-вторых, значительно снизится расход материалов, за счет того, что назначение на задания и все изменения по заданиям можно сделать прямиком в системе, не прибегая к формированию документов по назначению и их распечатке.

Ожидаемый годовой экономический эффект составил 1 478 170,8 тенге. Приложение окупится в первые 8,6 месяцев использования.

5 Безопасность жизнедеятельности

Темой дипломного проекта является Разработка web-портала «Документооборот IT-организации». В современном бизнесе, документы являются фундаментальной основой для осуществления внутренней и внешней деятельности организации. В условиях нынешней тенденции цифровизации процессов предприятий, для повышения их эффективности, система электронного документооборота может сыграть значительную роль в развитии предприятия. В работе необходимо создать web-сайт для предприятия ТОО «Первый БИТ», который будет автоматизировать процесс создания заявок на выезд специалиста и обмен документами между заказчиком и исполнителем, создать панель авторизации, вывести форму подачи заявки на работы специалиста, форму обратной связи, реализовать понятный и удобный интерфейс для навигации по сайту.

Особенность использования персональных компьютеров заключается в том, что в процессе взаимодействия человек-машина пользователь воспринимает компьютер как собеседника. Таким образом, возникает много новых проблем в психологии и психофизиологии, которые необходимо учитывать при проектировании рабочего процесса. Другой фактор – большая информационная нагрузка. Увеличение нагрузки на зрение и нервную систему вызывает гипертонию в нервной системе, что в свою очередь, оказывает негативное воздействие на сердечно-сосудистую систему. Важным аспектом физического состояния пользователя персонального компьютера является влияние на него комплекса факторов окружающей среды, таких как электромагнитные волны на разных частотах, электричество, шумовой мусор микроклимат и др. Также стоит отметить, что при неправильной разработки эргономики рабочего места сотрудника, может также возникнуть ряд факторов негативно влияющих на здоровье человека, что скажется на его работоспособности и эффективности. Правильное проектирование рабочего места сотрудника может решить все эти проблемы.

К проектированию рабочего пространства относятся только вопросы конструирования рабочего места пользователя, не затрагивая само изменение технических характеристик техники, с которой пользователь будет работать. Также следует отметить, что зачастую работа с ЭВМ происходит при постоянном взаимодействии с другими людьми. Поэтому возникают факторы межличностного взаимодействия, которые включают как психологические так и социальные аспекты. Таким образом, четыре типа факторов в рабочей среде влияют на пользователей компьютеров это: физические, эргономические, информационные и социально-психологические.

5.1 Расчётная часть

Для определения тяжести труда пользователей информационной системы необходимо каждому фактору, который характеризует условия труда, соответствующий балл. Исходные данные представлены в таблице 5.1.

Таблица 5.1.1 – Оценка условий труда на рабочем месте

№	Факторы	Значение	Баллы
1	Температура воздуха на РМ в теплый период года, t °С	23	3
2	Температура воздуха на рабочем месте в холодный период года, t °С	21	1
3	Рабочее место, поза и перемещение в пространстве	РМ стационарное, поза свободная, масса перемещаемого груза до 5 кг	1
4	Относительная влажность воздуха, φ, %	55	1
5	Скорость движения воздуха,V, мс.	0,6	3
6	Освещенность, Е, лк.	260	2
7	Шум, L, дБ.	58	3
8	Число важных объектов наблюдения	1	1
9	Темп (число движений пальцев в час)	610	2
10	Длительность сосредоточенного наблюдения, % от рабочего времени	82	4
11	Нервно-эмоциональная нагрузка	Простые действия по индивидуальному плану	1

Расчет интегральной бальной оценки тяжести труда:

$$И_{т1} = 10 \times (X_{оп} + X_{ср} \times \frac{6 - X_{оп}}{6}) \quad (5.1)$$

где $И_{т1}$ - интегральная бальная оценка тяжести труда;

$X_{оп}$ – элемент условий труда, который получил наибольшую оценку;

$X_{ср}$ – средний балл всех активных элементов условий труда кроме определяющего $X_{оп}$.

Средний балл всех активных элементов условий труда, кроме определяющего $X_{оп}$, рассчитывается по формуле:

$$X_{ср} = \frac{\sum_{i=1}^{n} X_i}{n - 1} \quad (5.2)$$

где $\sum_{i=1}^{n} X_i$ – сумма всех элементов кроме определяющего $X_{оп}$;

n – количество учтенных элементов условий труда.

В соответствии с имеющимися данными, $X_{оп} = 3$, а количество учетных элементов условий труда равняется 10.

Средний балл всех активных элементов условий труда:

$$X_{ср} = (3 + 1 + 1 + 1 + 3 + 2 + 3 + 1 + 2 + 1) / (11 – 1) = 18 / 10 = 1,8 \text{ балла.}$$

Далее производится интегрально-балльная оценка тяжести труда:

$$И_т = 10 \times (4 + 1,8 \times ((6 – 4)/6) = 46 \text{ баллов.}$$

Для определения категории тяжести труда необходимо воспользоваться таблицей 5.1.1

Таблица 5.1.1 - Категории тяжести труда

Категория тяжести труда	I	II	III	IV	V	VI
Интегральная оценка элементов условий труда, $И_т$, баллы	до 18	18,1- 33	33,1- 45	45,1- 53	53,1- 59	59,1- 60

Интегральная оценка тяжести труда составляет 46 баллов, что соответствует IV категории тяжести труда.

Повышение тяжести труда будет влиять на работоспособность человека. Снижение работоспособности непосредственно связано с состоянием утомления, которое количественно можно оценить при помощи показателя утомления, выраженного в условных единицах.

Интегральная бальная оценка тяжести труда позволяет определить влияние условий труда на работоспособность человека. Степень утомления определяется в условных единицах по формуле:

$$У = (И_т – 15,6) / 0,64 \qquad (5.3)$$

где У – степень утомления;
$И_т$ – интегральная бальная оценка тяжести труда;
15,6 и 0,64 – коэффициенты регрессии.
Таким образом, степень утомления составит:

$$У = (46 – 15,6) / 0,64 = 47,5$$

Зная степень утомления можно рассчитать уровень работоспособности. Уровень работоспособности является величиной противоположной утомлению и рассчитывается по формуле:

$$R = 100 – У \quad (5.4)$$

где R - работоспособность человека;
У - степень утомления.
Исходя из данной формулы, работоспособность составит:

$$R = 100 – 47,5= 52,5$$

Тяжесть и напряженность труда оказывает влияние на рост производственного травматизма. Так как интегральная балльная оценка дает возможность определить категорию тяжести труда, то величину производственного травматизма можно рассчитать по формуле:

$$K = 1 / (1,3 – 0,0185 \times И_т) \quad (5.5)$$

Таким образом, производственный травматизм составит:

$$K = 1 / (1,3 – 0,0185 \times 46) = 2,23$$

5.2 Вывод по разделу безопасности жизнедеятельности

В данной части дипломного проекта был произведен расчет интегральной бальной оценки тяжести и напряженности труда, выполнена оценка уровней утомляемости и работоспособности на рабочем месте. В результате расчета была определена категория тяжести труда на рабочем месте, произведен расчет работоспособности и степень утомления сотрудников. Исходя из полученных результатов, можно сделать следующие выводы:

1) Коэффициент работоспособности (52,5) превышает коэффициент степени утомления (47,5);
2) Коэффициент производственного травматизма составил 2,22.

В заключении можно отметить, что уровень тяжести и работоспособности труда, играет значительную роль для предприятия, при высоких значениях показателей тяжести труда снижается работоспособность сотрудников, при достижении высоких значений предприятие может понести убытки. При высоком травматизме, предприятие должно компенсировать тяжелые условия труда работникам. По результатам показателей рабочее место удовлетворяет нормам и внесения каких-либо изменений не требует.

Заключение

В ходе выполнения дипломного проекта был разработан web-портал для электронного документооборота в компании ТОО «Первый БИТ». Разработанная система предоставляет возможность создания заявок менеджерами компании, учета рабочего времени по созданным заявкам. Целью дипломного проекта был перевод документов листов учета рабочего времени и актов выполненных работ в электронный формат.

Для достижения поставленных целей были выполнены следующие пункты:

1) произведен анализ предметной области компании «Первый БИТ» и его строения;

2) проведено проектирование структуры web-портала, определены этапы а также выбор инструментальных средств разработки;

3) была реализована серверная часть web-приложения с использованием тенологий Node.js и Express.js;

4) была разработана модель базы данных MongoDB, был выбран облачный сервис MongoDB Atlas;

5) был реализован основной функционал web-портала, для тестирования серверной части был использован Postman API;

6) была разработана клиентская часть web-приложения с использованием технологий Angular.js и Materialize CSS;

7) произведено технико-экономическое обоснование целесообразности разрабатываемого проекта в результате которого было выяснено что, ожидаемый годовой экономический эффект составит 1 478 170,8 тенге. Приложение окупится в первые 8,6 месяцев использования.

Был произведен расчет интегральной оценки тяжести труда, были разработаны меры по улучшению показателей тяжести труда работников.

Результат выполнения дипломного проекта позволил автоматизировать процессы компании, увеличил скорость взаимодействия сотрудников, помог перевести ключевые документы для отдела внедрения в электронный формат.

Список использованной литературы

1 Г. Боканова Методические указания по выполнению экономической части дипломных работ Алматы, АУЭС, 2020 – 35с.

2 Технико-экономическое обоснование дипломных проектов Брест, БГТу, 2014 – 15с.

3 Методические указания к выполнению расчетно-графической работы для студентов специальности 080801 «Прикладная информатика (по областям)» Уфа, 2010 – 12с.

4 Методические указания по выполнению экономической части дипломных работ Москва, Московский университет им. С.Ю. Витте, 2016 – 21с.

5 Симионов Ю.Ф., Боромотов В.В. Информационный менеджмент. — Ростов н.Д: Феникс, 2013, 250с.

6 Simom Holmes, Getting MEAN with Mongo, Express, Angular, and Node, ISBN 2015, 440с.

7 URL: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/

8 URL: https://account.mongodb.com/

9 Кантелон, М. Node.js в действии / М. Кантелон. - М.: Питер, 2015. - 810 с.

10 Хэррон, Дэвид Node.js Разработка серверных веб-приложений на JavaScript / Дэвид Хэррон. - М.: ДМК Пресс, 2014. - 144 с.

11 Глушаков, С. В. Программирование Web-страниц. JavaScript. VBScript / С.В. Глушаков, И.А. Жакин, - М.: Фолио, 2015. - 390 с

12 URL: https://angular.io/

13 URL: https://materializecss.com/

14 URL:https://developer.mozilla.org/ru/docs/Learn/Server-side/Express_Nodejs/routes

15 Бэнкер, К. MongoDB в действии / К. Бэнкер. - Москва: Высшая школа, 2016. - 287 с.

16 Лазаро Исси Коэн, Джозеф Исси Коэн Полный справочник по HTML, CSS и JavaScript; ЭКОМ Паблишерз - М., 2016. - 311 с.

17 Роббинс Дженнифер HTML5, CSS3 и JavaScript. Исчерпывающее руководство; Эксмо - М., 2017. - 528 с.

Приложение А

Техническое задание

1 Основные требования:

- название разрабатываемой программы: разработка web-портала для ТОО «Первый БИТ»;

- цель разработки: автоматизировать процесс создания заявок на выполнение работ, обмен документами лист учета рабочего времени и акт выполненных работ;

Рекомендуемые платформы для разработки приложения (на выбор разработчика):

- Python;
- Node.js;
- Vue.js;
- React.js;
- Angular.js;
- Ruby on Rails;
- PostgreSQL;
- MongoDB.

2 Технические требования:

- для работы с сайтом возможно использование любого компьютера с доступом в интернет и установленным браузером Chrome 81.0 и выше, или Opera 22 и выше, также возможно использование браузеров UC, Mozilla, Яндекс и др. последних версий.

3 Экономические требования.

Расчет сметы разработки программного продукта

- стоимость готового продукта 892 323,6 тг;
- стоимость разработки 531 145 тг.

Приложение Б

Листинг программы

```
//настройка сервера
const app = require('./app')
const port = process.env.port || 5000

app.listen(port, () => console.log(`Server has been started on ${port}`))
const express = require('express')
const bodyParser = require('body-parser')
const mongoose = require('mongoose')
const passport = require('passport')
const authRoutes = require('./routes/auth')
const analyticsRoutes = require('./routes/analytics')
const categoryRoutes = require('./routes/category')
const orderRoutes = require('./routes/order')
const positionRoutes = require('./routes/position')
const keys = require('./config/keys')
const app = express()

mongoose.connect(keys.mongoURI, {
   useNewUrlParser: true,
   useUnifiedTopology: true,
   useCreateIndex: true
   })
   .then(() => console.log('MongoDB connected'))
   .catch(error => console.log(error))
app.use(passport.initialize())

require('./middleware/passport')(passport)
app.use(require('morgan')('dev'))
app.use('/uploads', express.static('uploads'))
app.use(require('cors')())
app.use(bodyParser.urlencoded({extended: true}))
app.use(bodyParser.json())

app.use('/api/auth', authRoutes)
app.use('/api/analytics', analyticsRoutes)
app.use('/api/category', categoryRoutes)
app.use('/api/order', orderRoutes)
app.use('/api/position', positionRoutes)

module.exports = app

//настройка роута

const express = require('express')
const passport = require('passport')
const upload = require('../middleware/upload')
const controller = require('../controllers/category')
```

```
const router = express.Router()

router.get('/', passport.authenticate('jwt', {session: false}), controller.getAll)
router.get('/:id', passport.authenticate('jwt', {session: false}), controller.getById)
router.delete('/:id', passport.authenticate('jwt', {session: false}), controller.remove)
router.post('/', passport.authenticate('jwt', {session: false}), upload.single('image'), controller.create)
router.patch('/:id', passport.authenticate('jwt', {session: false}), upload.single('image'), controller.update)

module.exports = router
//настройка контроллера
const mongoose = require('mongoose')
const Schema = mongoose.Schema

const categorySchema = new Schema({
  name: {
    type: String,
    required: true
  },
  imageSrc: {
    type: String,
    default: ''
  },
  user: {
    ref: 'users',
    type: Schema.Types.ObjectId
  }
})

//requiring passport settings
module.exports = mongoose.model('categories', categorySchema)

const JwtStrategy = require('passport-jwt').Strategy
const ExtractJwt = require('passport-jwt').ExtractJwt
const mongoose = require('mongoose')
const User = mongoose.model('users')
const keys = require('../config/keys')

const options = {
  jwtFromRequest: ExtractJwt.fromAuthHeaderAsBearerToken(),
  secretOrKey: keys.jwt
}
//export passport all
module.exports = passport => {
  passport.use(
    new JwtStrategy(options, async (payload, done) => {
      try {
        const user = await User.findById(payload.userId).select('email id')
```

```
        if (user) {
          done(null, user)
        } else {
          done(null, false)
        }
      } catch (e) {
        console.log(e)
      }

    })
  )
}

//creating model
const mongoose = require('mongoose')
const Schema = mongoose.Schema

const categorySchema = new Schema({
  name: {
    type: String,
    required: true
  },
  imageSrc: {
    type: String,
    default: ''
  },
  user: {
    ref: 'users',
    type: Schema.Types.ObjectId
  }
})
module.exports = mongoose.model('categories', categorySchema)

const mongoose = require('mongoose')
const Schema = mongoose.Schema

const orderSchema = new Schema({
    date: {
        type: Date,
        default: Date.now
    },
    order: {
        type: Number,
        required: true
    },
    list:[
        {
            name: {
                type: String
```

```
        },
        quantity: {
          type:Number
        },
        cost: {
          type: Number
        }
      }
  ],
  user:{
    ref: 'users',
    type: Schema.Types.ObjectId
  }
})

module.exports = mongoose.model('orders', orderSchema)
const Category = require('../models/Category')
const Position = require('../models/Position')
const errorHandler = require('../utils/errorHand')

module.exports.getAll = async function(req, res) {
 try {
  const categories = await Category.find({user: req.user.id})
  res.status(200).json(categories)
 } catch (e) {
  errorHandler(res, e)
 }
}

module.exports.getById = async function(req, res) {
 try {
  const category = await Category.findById(req.params.id)
  res.status(200).json(category)
 } catch (e) {
  errorHandler(res, e)
 }
}

module.exports.remove = async function(req, res) {
 try {
  await Category.remove({_id: req.params.id})
  await Position.remove({category: req.params.id})
  res.status(200).json({
   message: 'Категория удалена.'
  })
 } catch (e) {
  errorHandler(res, e)
 }
}
```

```
module.exports.create = async function(req, res) {
  const category = new Category({
    name: req.body.name,
    user: req.user.id,
    imageSrc: req.file ? req.file.path : ''
  })

  try {
    await category.save()
    res.status(201).json(category)
  } catch (e) {
    errorHandler(res, e)
  }
}

module.exports.update = async function(req, res) {
  const updated = {
    name: req.body.name
  }

  if (req.file) {
    updated.imageSrc = req.file.path
  }
  try {
    const category = await Category.findOneAndUpdate(
      {_id: req.params.id},
      {$set: updated},
      {new: true}
    )
    res.status(200).json(category)
  } catch (e) {
    errorHandler(res, e)
  }
}

module.exports = {
    mongoURI: 'mongodb+srv://eldar:**********@cluster0-xhvcw.mongodb.net/test?retryWrites=true&w=majority',
    jwt: 'dev-jwt'
}

import {NgModule} from '@angular/core'
import {RouterModule, Routes} from '@angular/router'
import {LoginPageComponent} from './login-page/login-page.component'
import {AuthLayoutComponent} from './shared/layouts/auth-layout/auth-layout.component'
import {SiteLayoutComponent} from './shared/layouts/site-layout/site-layout.component'
import {RegistrPageComponent} from './registr-page/registr-page.component'
import {AuthGuard} from './shared/classes/auth.guard'
```

```
import {OverviewPageComponent} from './overview-page/overview-page.component'
import {AnalyticsPageComponent} from './analytics-page/analytics-page.component'
import {HistoryPageComponent} from './history-page/history-page.component'
import {OrderPageComponent} from './order-page/order-page.component'
import {CategoriesPageComponent} from './categories-page/categories-page.component'
import {CategoriesFormComponent} from './categories-page/categories-form/categories-form.component'
import {OrderCategoriesComponent} from './order-page/order-categories/order-categories.component'
import {OrderPositionsComponent} from './order-page/order-positions/order-positions.component'
import {LurvPageComponent} from './lurv-page/lurv-page.component'
import {LurvFormComponent} from './lurv-page/lurv-form/lurv-form.component'
import {AvrFormComponent} from './avr-page/avr-form/avr-form.component'
import {AvrPageComponent} from './avr-page/avr-page.component'

const routes: Routes = [
  {
    path: '', component: AuthLayoutComponent, children: [
      {path: '', redirectTo: '/login', pathMatch: 'full'},
      {path: 'login', component: LoginPageComponent},
      {path: 'register', component: RegistrPageComponent}
    ]
  },
  {
    path: '', component: SiteLayoutComponent, canActivate: [AuthGuard], children: [
      {path: 'overview', component: OverviewPageComponent},
      {path: 'analytics', component: AnalyticsPageComponent},
      {path: 'history', component: HistoryPageComponent},
      {path: 'order', component: OrderPageComponent, children: [
          {path: '', component: OrderCategoriesComponent},
          {path: ':id', component: OrderPositionsComponent}
      ]},
      {path: 'categories', component: CategoriesPageComponent},
      {path: 'categories/new', component: CategoriesFormComponent},
      {path: 'categories/:id', component: CategoriesFormComponent},
      {path: 'lurv', component: LurvPageComponent},
      {path: 'lurv/form', component:LurvFormComponent},
      {path: 'avr/new', component:AvrFormComponent },
      {path: 'avr', component:AvrPageComponent }
    ]
  }
]
@NgModule({
  imports: [
    RouterModule.forRoot(routes)
  ],
  exports: [
    RouterModule
```

```
  ]
})
export class AppRoutingModule {
}
<ul class="sidenav sidenav-fixed a-sidenav">
  <h4>Первый БИТ
  </h4>

  <li
    class="bold"
    *ngFor="let link of links"
    routerLinkActive="active"
  >
    <a [routerLink]="[link.url]" class="waves-effect waves-orange">
      {{link.name}}
    </a>
  </li>

  <li class="bold last">
    <a href="#" class="waves-effect waves-orange" (click)="logout($event)">Выйти</a>
  </li>
</ul>

<main class="content">
  <router-outlet></router-outlet>
</main>

<div class="fixed-action-btn" #floating>
  <a class="btn-floating btn-large red">
    <i class="large material-icons">add</i>
  </a>
  <ul>
    <li>
      <a class="btn-floating green" routerLink="/order">
        <i class="material-icons">assignment</i>
      </a>
    </li>
    <li>
      <a class="btn-floating blue" routerLink="/categories/new">
        <i class="material-icons">list</i>
      </a>
    </li>

  </ul>
</div>


<a id="menu" class="waves-effect waves-light btn btn-floating"><i class="material-
icons">info</i></a>
```

```
import {AfterViewInit, Component, ElementRef, ViewChild} from '@angular/core'
import {AuthService} from '../../services/auth-service'
import {Router} from '@angular/router'
import {MaterialService} from '../../classes/material.service'

@Component({
  selector: 'app-site-layout',
  templateUrl: './site-layout.component.html',
  styleUrls: ['./site-layout.component.css']
})
export class SiteLayoutComponent implements AfterViewInit {

  @ViewChild('floating') floatingRef: ElementRef

  links = [
    {url: '/overview', name: 'Обзор'},
    {url: '/analytics', name: 'Аналитика'},
    {url: '/history', name: 'История'},
    {url: '/order', name: 'Создать наряд'},
    {url: '/categories', name: 'Номенклатура/услуги'},
    //{url: '/lurv', name: 'Выписать ЛУРВ'},
    {url: '/avr', name: 'Выписать Акт выполненных работ'}
  ]
  constructor(private auth: AuthService,
              private router: Router) {
  }

  ngAfterViewInit() {
    MaterialService.initializeFloatingButton(this.floatingRef)
  }

  logout(event: Event) {
    event.preventDefault()
    this.auth.logout()
    this.router.navigate(['/login'])
  }

}
//Use materialize CSS
import {ElementRef} from '@angular/core'

declare var M

export interface MaterialInstance {
  open?(): void
  close?(): void
  destroy?(): void
}
```

```
export interface MaterialDatepicker extends MaterialInstance {
  date?: Date
}

export class MaterialService {
  static toast(message: string) {
    M.toast({html: message})
  }

  static initializeFloatingButton(ref: ElementRef) {
    M.FloatingActionButton.init(ref.nativeElement)
  }

  static updateTextInputs() {
    M.updateTextFields()
  }

  static initModal(ref: ElementRef): MaterialInstance {
    return M.Modal.init(ref.nativeElement)
  }

  static initTooltip(ref: ElementRef): MaterialInstance {
    return M.Tooltip.init(ref.nativeElement)
  }

  static initDatepicker(ref: ElementRef, onClose: () => void): MaterialDatepicker {
    return M.Datepicker.init(ref.nativeElement, {
      format: 'dd.mm.yyyy',
      showClearBtn: true,
      onClose
    })
  }

  static initTapTarget(ref: ElementRef): MaterialInstance {
    return M.TapTarget.init(ref.nativeElement)
  }
}
```

Приложение В

Акт внедрения

«Первый БИТ» ЖШС
Алматы филиалы

Тұрған жерінің мекенжайы:
Қазақстан Республикасы, Алматы, БО «Sirius»
қ. Наурызбай батыр, 17 үй, 603 офистері
Тел/факс: (727) 344-29-99,
e-mail: info.almaty@1cbit.ru
www.1cbit.kz

первыйБит
международный интегратор
ИТ-решений

Алматинский филиал
ТОО «Первый БИТ»

Адрес фактического местонахождения:
Республика Казахстан, г. Алматы, БЦ «Sirius»,
ул. Наурызбай Батыра 17, оф. 603
Тел/факс: (727) 344-29-99,
e-mail: info.almaty@1cbit.ru
www.1cbit.kz

БИН: 170941012541, Банковские реквизиты: ИИК: KZ849470398990776310, АО ДБ «Альфа-Банк» г. Алматы, БИК: ALFAKZKA

№ 243/1 от 15.05.2020г.

Утверждаю _____
«20» мая 2020г.

Директор АФ ТОО «Первый БИТ»
Абилов Ж.С.

АКТ ВНЕДРЕНИЯ

Настоящий акт составлен о том, что результаты выпускной работы студента НАО АУЭС им. Гумарбека Даукеева гр. Инф-16-2 очной формы обучения Абдимижитова Эльдара на тему «*разработка web-портала Документооборот IT-организации*» внедрены в отдел внедрения ТОО «Первый БИТ» и используются в компании для взаимодействия сотрудников. Использование результата выпускной работы Абдимижитова Э.М. решило вопросы по организации обмена внутренними документами сотрудников предприятия, положительно повлияло на процесс деятельности организации.

Директор АФ ТОО «Первый БИТ» **Абилов Ж.С.**