Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Хакасский технический институт – филиал ФГАОУ ВО
«Сибирский федеральный университет»

Кафедра прикладной информатики, математики и естественно-научных дисциплин

УТВЕРЖДАЮ
Заведующий кафедрой
_____ Е. Н. Скуратенко
подпись
« _____» _____ 2020 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.03 Прикладная информатика

Тема: Разработка информационной системы «Реестр работников системы здравоохранения Республики Хакасия »

Руководитель _____ зав. кафедрой, к.т.н Е.Н. Скуратенко
подпись, дата

Выпускник _____ А.С. Утьев
подпись, дата

Консультанты
по разделам:

Экономический _____ Е. Н. Скуратенко
подпись, дата

Нормоконтролер _____ В. И. Кокова
подпись, дата

Абакан 2020

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Хакасский технический институт – филиал ФГАОУ ВО
«Сибирский федеральный университет»

Кафедра прикладной информатики, математики и естественно-научных дисциплин

УТВЕРЖДАЮ
Заведующий кафедрой
_____Е. Н. Скуратенко
подпись

« _____» _____ 2020 г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
в форме бакалаврской работы

Студенту Утьеву Алексею Сергеевичу

Группа ХБ 16-03

Направление 09.03.03 Прикладная информатика

Тема выпускной квалификационной работы: Разработка информационной системы «Реестр работников системы здравоохранения Республики Хакасия».

Утверждена приказом по институту № 216 от 06.04.2020 г.

Руководитель ВКР: Е.Н. Скуратенко, зав. кафедрой, к.т.н, ХТИ – филиал СФУ

Исходные данные для ВКР: информация о деятельности системы Здравоохранения Республики Хакасия, информация о локальном сервере.

Перечень разделов ВКР:

1. Анализ предметной области.
2. Разработка информационной системы «Реестр работников системы здравоохранения Республики Хакасия».
3. Оценка экономической эффективности информационной системы.

Перечень графического материала: нет

Руководитель ВКР _____ Е. Н. Скуратенко
подпись

Задание принял к исполнению_____ А. С. Утьев
подпись

«___» _____ 2020 г.

РЕФЕРАТ

Выпускная квалификационная работа по теме «Разработка информационной системы «Реестр работников системы здравоохранения Республики Хакасия» содержит 61 страницу текстового документа, 30 рисунков, 8 таблиц, 18 формул, 1 приложение, 20 использованных источников.

ИНФОРМАЦИОННАЯ СИСТЕМА, ВЕБ-ПРИЛОЖЕНИЕ,NODE JS, MONGO DB, РЕЕСТР, ЭКОНОМИЧЕСКАЯ ЭФФЕКТИВНОСТЬ, КАПИТАЛЬНЫЕ ЗАТРАТЫ, ЭКСПЛУАТАЦИОННЫЕ ЗАТРАТЫ.

Цель работы: разработать реестр сотрудников и отчетов для системы здравоохранения Республики Хакасия.

Задачи:

– изучить предметную область;

– провести анализ деятельности системы здравоохранения Республики Хакасия;

– определить цель и задачи проектирования ИС;

– определить информационное обеспечение для разработки ИС;

– спроектировать информационную систему;

– разработать информационную систему;

– произвести расчеты экономической эффективности проекта.

В результате было разработано веб-приложение для регионального медицинского информационно-аналитического центра

SUMMARY

The theme of the graduation thesis is «Development of Staff and Reports Registry for the Public Healthcare System of the Republic of Khakassia». It comprises 61 pages, 30 figures, 8 charts, 18 formulae, 1 appendices, 20 reference items.

IT SYSTEM, WEB APPLICATION, NODE JS, MONGO DB, REGISTRY, ECONOMIC EFFICIENCY, CAPITAL COSTS, OPERATING COSTS.

The purpose of the thesis is to develop staff and reports registry for the Public Healthcare System of the Republic of Khakassia.

Objectives:

– to study the subject matter;

– to analyze the corporate activity of Public Healthcare System of the Republic of Khakassia;

– to determine the purpose and objectives of IT system design;

– to identify infoware for IT system design;

– to design an IT system;

– to develop an IT system;

– to calculate the project’s cost-effectiveness.ë

A web application has been developed for Regional Medical Information Analytical Center.

English language supervisor _____ Chezybaeva N.V.

signature, date — full name

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Актуальность выбранной темы исследования обусловлена следующими обстоятельствами.

Роль и важность системы хранения определяются постоянно возрастающей ценностью информации в современном обществе, возможность доступа к данным и управления ими является необходимым условием для выполнения каких-либо процессов.

От того насколько удобен будет процесс поиска и доступа к необходимой информации зависит часть возможностей руководителя.

Цель проекта: разработать реестр сотрудников и отчетов для системы здравоохранения Республики Хакасия.

Задачи проекта:

- Ознакомиться с основной деятельностью предприятия.
- Разработать план работ поэтапной разработки проекта.
- Выбрать средства проектного решения.
- Разработать прототип проекта.
- Разработать ИС реестр сотрудников и отчетов для РМИАЦ
- Разработать план тестирования ИС.
- Оценить экономические характеристики ИС.
- Оценить риски ИС.

1 Анализ предметной области

1.1 Организационная характеристика ГКУЗ РХ "РМИАЦ"

Заказчиком разработки является государственное казённое учреждение здравоохранения Республики Хакасия Республиканский медицинский информационно-аналитический центр, краткое наименование – ГКУЗ РХ "РМИАЦ".

Организационная характеристика ГКУЗ РХ "РМИАЦ"

Юридический адрес учреждения: ул. Крылова,72, Абакан, Республика Хакасия, 655003. Контактный телефон: 8(390)229-50-13.

Организации необходима автоматизированная система быстрого и удобного поиска информации о сотрудниках системы здравоохранения Республики Хакасия.

ГКУЗ РХ "РМИАЦ" состоит из отделов: медицинской статистики и мониторинга, автоматизированных систем управления, технического и адресно-справочного (Рисунок 1).

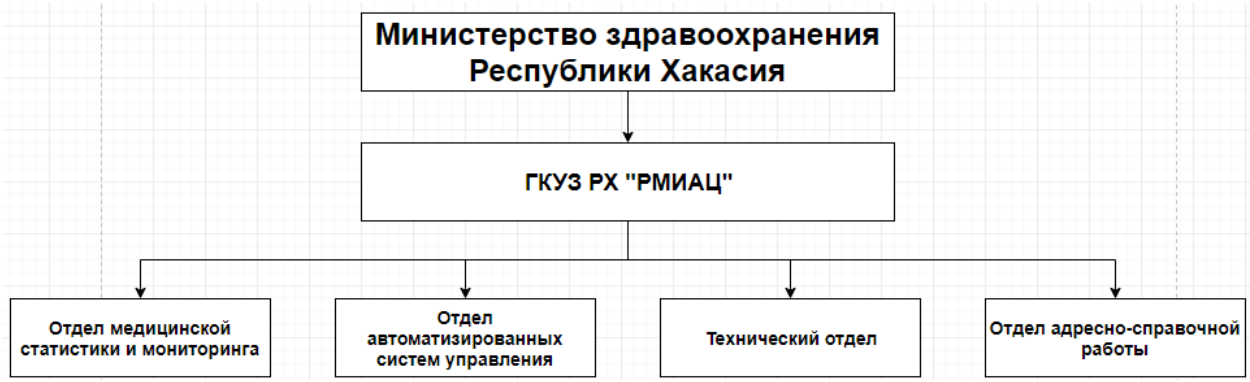

Рисунок 1 –Структурная схема ГКУЗ РХ "РМИАЦ"

Учреждение "РМИАЦ" выполняет следующие основные виды деятельности:

– Разработка концепций и программ информационной системы здравоохранения Республики Хакасия.

– Координация работ по созданию единой информационной системы здравоохранения Республики Хакасия.

– Формирование единой системы учета и отчетности медико-статистической информации с применением новых технологий её обработки.

– Прием и обработка статистических отчетов от учреждений здравоохранения.

– Разработка показателей, характеризующих деятельность учреждений здравоохранения Республики Хакасия, состояние здравоохранения в рамках утвержденной статистической отчетности.

– Разработка, внедрение и сопровождение автоматизированных систем сбора, хранения и передачи информации.

– Анализ полученной информации с привлечением главных штатных и внештатных специалистов Министерства здравоохранения Республики, организационно-методических отделов Республики, больниц и диспансеров.

– Осуществление взаимодействия с территориальным фондом обязательного медицинского страхования, территориальным органами государственной статистики, образовательными и научными учреждениями и другими сторонними организациями.

– Осуществление контроля над использованием в работе учреждений здравоохранения международных классификаций при ведении медицинской документации.

– Выработка управленческих решений по повышению эффективности деятельности учреждений здравоохранения.

– Анализ медико-статистической информации о состоянии здоровья населения и состояния здравоохранения Республики Хакасия.

Анализируя представленный перечень видов деятельности ГКУЗ РХ «РМИАЦ», очевидным, является необходимость сбора, хранения и

систематизации большого перечня разного рода информации, в том числе о сотрудниках системы здравоохранения Республики Хакасия.

1.2 Анализ потребностей заказчика и обоснование проектного решения

При интервьюировании заказчика были определены следующие требования к информационной системе:

1. База данных должна содержать данные о сотрудниках:

– Фамилия.

– Имя.

– Отчество.

– Номер телефона.

– Адрес электронной почты.

– Место работы.

– Должность сотрудника.

2. В базе данных должна содержаться информация о наименованиях отчетов, формируемых сотрудниками.

3. Внесение данных в базу должны осуществлять администраторы медицинских учреждений через пользовательский интерфейс.

4. Для удаленного доступа к базе с различных типов устройств должно быть сделано веб-приложение.

5. Информационная система должна формировать отчеты на выборку, имея параметры места работы сотрудника, а также должности сотрудника.

На основе представленного обоснования к функциям ИС проведен анализ сред разработки системы, наиболее отвечающих требованиям.

1.3 Выбор программных средств разработки ИС

Не смотря на то, что PHP и Node.js могут справляться с приложениями любой сложности, они созданы на основе разных концепций и архитектур. Node.js и PHP одни из самых часто используемых средств для разработки веб-сайтов. PHP — скриптовый язык созданный RasmusLerdorf в 1994, являлся языком номер один Web 1.0. Показательное проявление успеха PHP являются системы управления контентом, такие как WordPress, Joomla и Drupal, с их помощью работают миллионы блогов и веб порталов. Node.js это представитель более новых технологий веб-разработки. В отличии от PHP, Node.js не является языком программирования, это среда выполнения, которая использует JavaScript для написания приложений на стороне сервера. Представлен в 2009, Node.js продемонстрировал силу JavaScript при разработке событийно-ориентированных приложений.

Плюсы PHP:

1. Большая кодовая база: PHP огромною кодовою базу для всевозможных решений от систем управления контентом до мощных фреймворков таких как Laravel, Symfony. Например, с помощью WordPress можно запустить свой блог за считанные минуты.

2. Переносимое решение: PHP достаточно независим от платформы, он может быть запущен почти на любом сервере и на любой платформе.

3. Спроектирован для WEB разработки: В отличие от Java или Python и других языков программирования общего назначения, PHP был разработан специально для Web. Именно поэтому он содержит всю необходимую функциональность для работы с HTML, серверами, базами данных.

Минусы PHP:

1. Устаревшая клиент-серверная модель: PHP следует классической клиент-серверной модели, где запрос страницы инициирует приложение, соединение с базой данных, их обработкой и рендерингом HTML. Это делает PHP несколько медленнее в сравнении с Node.js приложениями, которые

инициируются при запуске, и поэтому Node.js больше подходит для написания приложений реального времени.

Плюсы Node.js:

1. Быстрое серверное решение: Node.js позволяет, используя очередь событий JavaScript, создавать приложения с неблокирующим вводом/выводом, которые способны обрабатывать несколько запросов одновременно. Используя встроенною в JavaScriptасинронность, можно создавать высокомасштабируемые серверные приложения, которые максимизируют использование одного CPU и памяти компьютера при одновременной обработке большего количества запросов, чем обычные многопоточные серверы. Такая функциональность делает Node.js прекрасным выбором для приложений реального времени и тех приложений, которые требуют большого количества операций ввода/вывода.

2. Гибкость: Node.js не имеет строгих правил или жестких зависимостей, что оставляет простор для созидания и креативности при разработке приложений. Разработчики сами выбирают архитектуру, зависимости.

3. Один язык на Frontendи backend: много популярных JS фреймворков таких как React или Vue написаны на JavaScript, который является основным языком всех современных браузеров. Используя Node.js на сервере можно получить все преимущества скриптового языка на обеих платформах.

4. Встроенный менеджер пакетов npm: Встроенный менеджер пакетов позволяет подключать пакеты, которые упростят и ускорят разработку проекта.

Минусы Node.js

1. Малая эффективность в операциях, интенсивно использующих CPU: Событийно-ориентированная архитектура Node.js имеет некоторые ограничения, а именно низкую эффективность при большой нагрузке на CPU. Хотя и Node.js хорошо справляется конкурентной обработкой множества запросов, он все же плохо справляется с такими операциями, как

генерирование графики, обработка изображений. К счастью, существует обходной путь, в котором можно выполнять такие операции по очереди или в отдельном процессе.

Таблица 1 показывает сравнение Node JS и PHP в цифрах.

Таблица 1 – Сравнение Node JS и PHP

	Node JS	PHP
Интеграция пакетов(дополнений)	5	4
Количество обучающей литературы	4	5
Клиент-сервеная модель(быстродействие)	5	3
Гибкость	5	4
Зрелость платформы	4	5
Итого	23	21

Для разработки ИС был выбран Node.js с фреймворком Express, так как у Node.js плюсов больше, чем у PHP.

1.4 Выбор системы управления базами данных для разработки

СУБД – это общий термин, относящийся ко всем видам абсолютно разных инструментов, от компьютерных программ до встроенных библиотек. Эти приложения управляют или помогают управлять наборами данных. Так как эти данные могут быть разного формата и размера, были созданы разные виды СУБД.

СУБД основаны на моделях баз данных – определённых структурах для обработки данных. Каждая СУБД создана для работы с одной из них с учётом особенностей операций над информацией.

Реляционные системы управления базами данных берут своё название от реализуемой модели — реляционной. Сейчас они остаются, да и ещё

какое-то время будут, самым популярным выбором для надёжного, безопасного и производительного хранения данных. Схемы очень похожи на таблицы, столбцы которых отражают порядковый номер и тип информации в каждой записи, а строки — содержимое этих записей.

NoSQL-СУБД не используют реляционную модель структуризации данных. Существует много реализаций, решающих этот вопрос по-своему, зачастую весьма специфично. Эти бессхемные решения допускают неограниченное формирование записей и хранение данных в виде ключ-значение.

В отличие от традиционных РСУБД, некоторые базы данных NoSQL, например, MongoDB, позволяют группировать коллекции данных с другими базами данных. Такие СУБД хранят данные как одно целое. Эти данные могут представлять собой одиночный объект наподобие JSON и вместе с тем корректно отвечать на запросы к полям.

NoSQL базы данных и СУБД не подразумевают внутренних связей. Они не основываются на одной модели, каждая база данных в зависимости от целей использует различные модели.

Существует несколько различных моделей и функциональных систем для NoSQL баз данных:

- Хранилище ключ-значение.
- Распределенное хранилище.
- Документо-ориентированные.
- БД на основе графов.

MongoDB — документно-ориентированная система управления базами данных (СУБД) с открытым исходным кодом, не требующая описания схемы таблиц. Классифицирована как NoSQL, использует JSON-подобные документы и схему базы данных. Написана на языке C++.

Для разработки была выбрана СУБД MongoDB, так как на сервере заказчика уже есть реализованный локальный сервер MongoDB. Также данная СУБД подходит по требованиям хранения информации.

1.5 Выводы по разделу «Анализ предметной области»

В ходе выполнения первого раздела была проанализирована предметная область проекта, было выполнено интервьюирование заказчика проекта, был проведен выбор средств реализации проекта. Также после сравнения двух языков был выбран язык для написания информационной системы(Node.js). После выбора языка была выбрана СУБД для хранения в ней данных о сотрудниках (MongoDB).

2 Разработка информационной системы «Реестр работников системы здравоохранения Республики Хакасия»

2.1 Проектирование информационной системы

В процессе разработки диаграммы UML и IDEF3 помогут учесть функции и события, необходимые для реализации функционала системы. Также диаграммы помогают увидеть события, происходящие в системе при разном поведении пользователя, и функции, которые должны быть прикреплены к событиям, вызванным пользователем. При реализации большой системы разработчику нужно всегда знать, какое событие вызывается каким компонентом, помнить эту схему не всегда бывает возможно, и чтобы не запутаться в большом количестве элементов разработчикам требуются диаграммы. Чем детальнее проработана диаграмма, тем потом проще будет разработчику каждый раз к ней обращаться и сразу видеть места инициализации функций или событий.

Для реализации проекта первоначальное планирование структуры и работы информационной системы осуществлялось с помощью UML диаграммы и с применением нотации IDEF3. Диаграмма UML была спроектирована, чтобы показать различные состояния системы при работе пользователя с системой (Рисунок 2). В самом начале пользователь проходит

авторизацию, далее у него отображается форма с выбором действий. Пользователь выбирает подходящее для него действие и переносится на ветку с выбранным действием. В данной диаграмме показана работа с данными сотрудников (создание, редактирование и поиск). При выборе одной из веток открываются формы для данной ветки. При открытии формы пользователь должен проделать необходимые действия с формой для того, чтобы получить необходимый результат. К примеру, для создания сотрудника пользователь должен быть авторизован как администратор и внести данные в поля формы, после чего нажать на кнопку «Добавить пользователя». После этого будет выполняться запрос в базу данных на добавление записи в документ «Пользователи». Такая же логика процесса будет и с алгоритмом добавления отчетов.

Схема IDEF3 показывает процессы внутри системы с момента входа в систему до выхода из нее. После запуска приложения пользователю необходимо внести свой логин и пароль, на основании базы данных пользователей происходит определение прав данного пользователя. Существует три уровня ранжирования пользователей: авторизованный пользователь, администратор. Авторизированный пользователь будет иметь доступ к веткам поиска данных в базе данных, ему будут доступны только формы поиска сотрудника и отчетов. Администратору будут доступны все ветки, имеющиеся в системе, такие как: создание сотрудника, редактирование сотрудника, создание отчета, редактирование отчета.

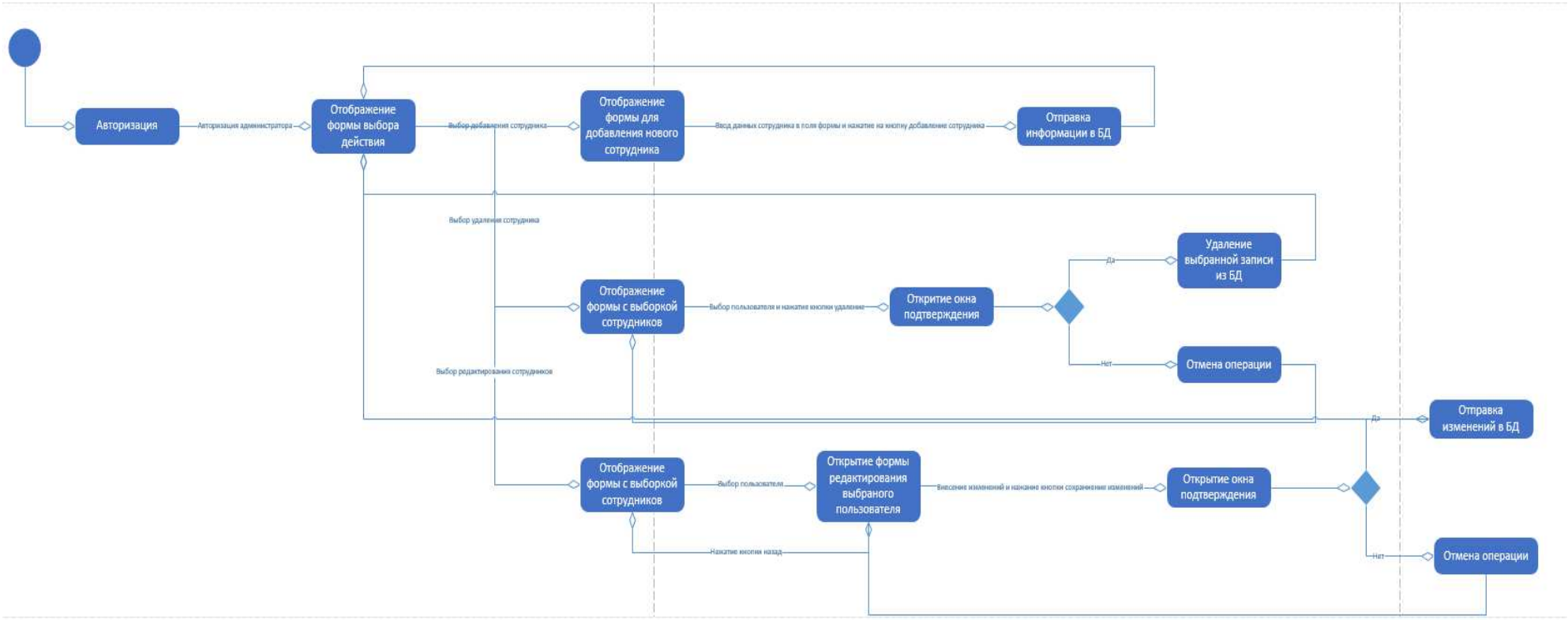

Рисунок 2 – Диаграмма состояний UML

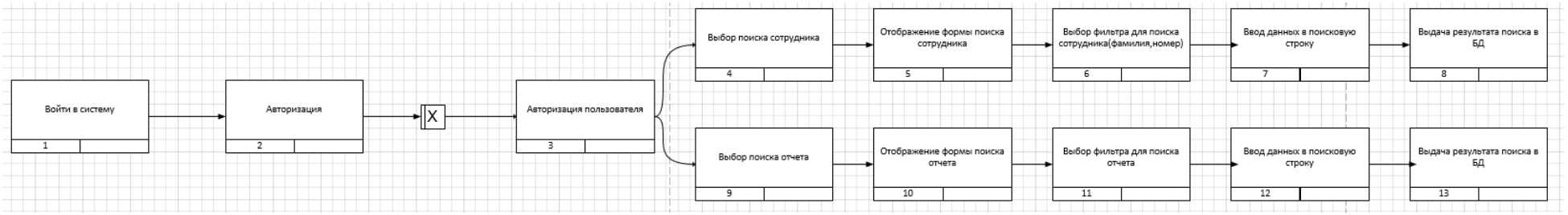

Рисунок 3 – IDEF 3для авторизованного пользователя

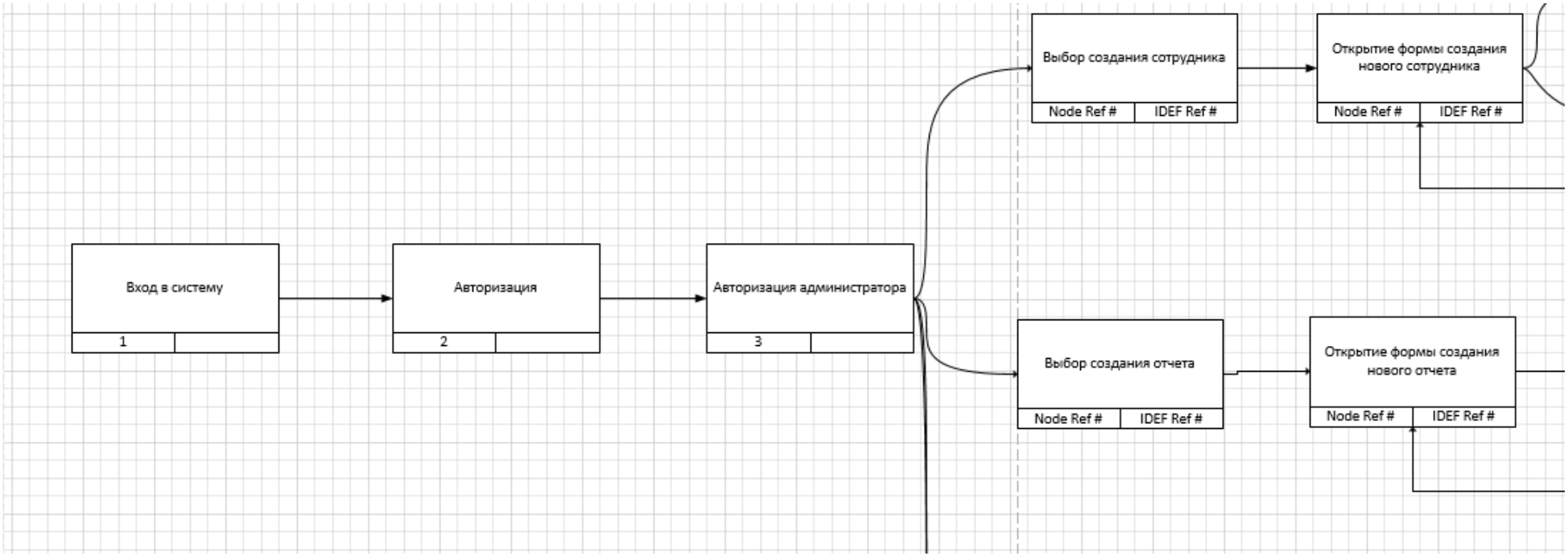

Рисунок 4 – IDEF 3для администратора, лист 1

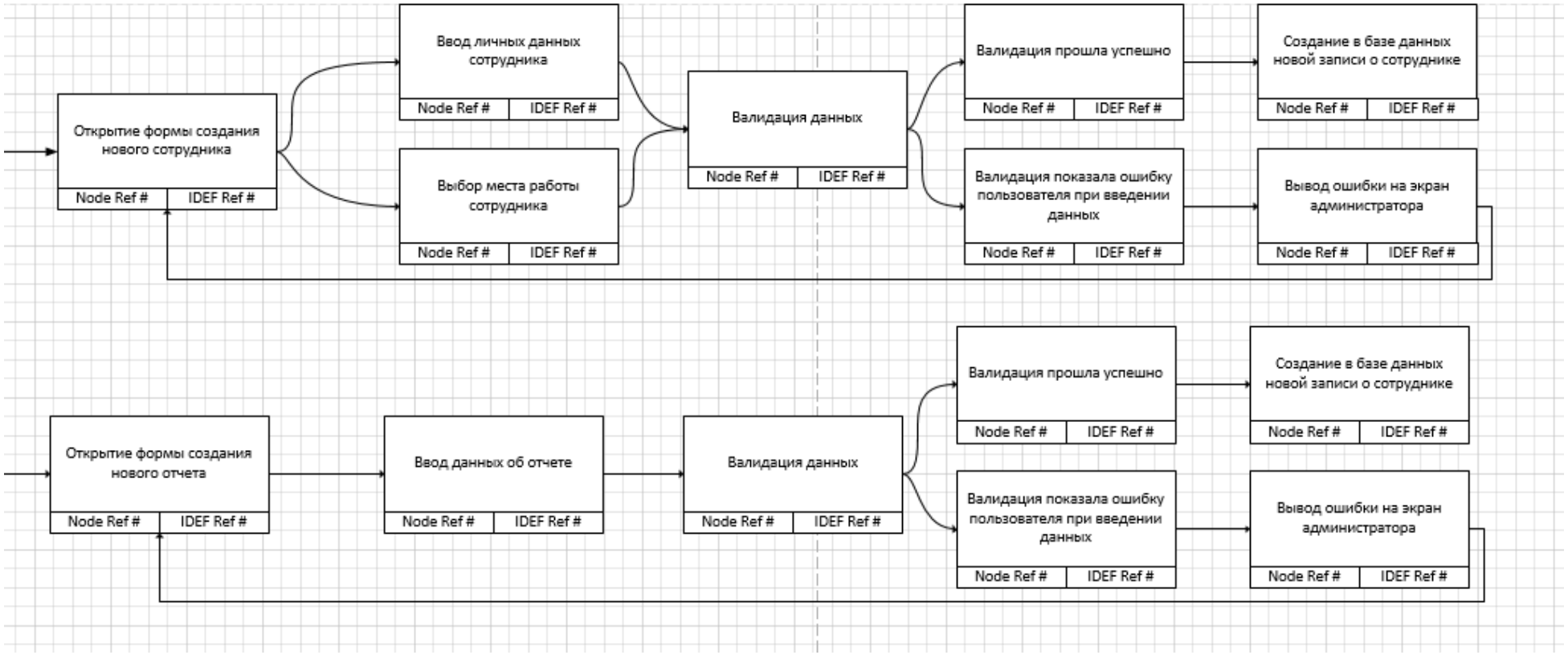

Рисунок 4, лист 2

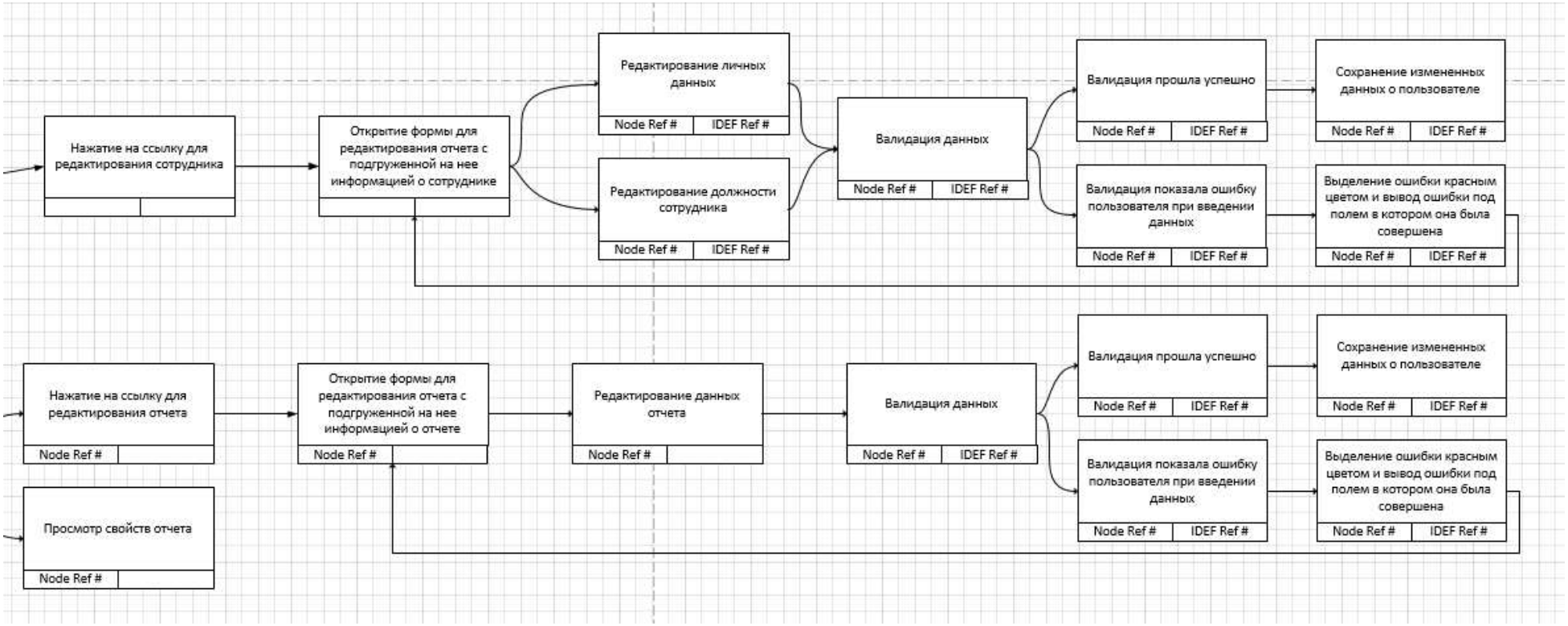

Рисунок 4, лист 3

Спроектирована база данных для хранения информации. В базе данных должна храниться информация о сотрудниках, отчетах, местах работы сотрудников, пользователей, прав доступа пользователей и дополнительный набор справочников для уменьшения повторяемости однотипной информации, и уменьшение вероятности ошибки внесения некорректных данных.

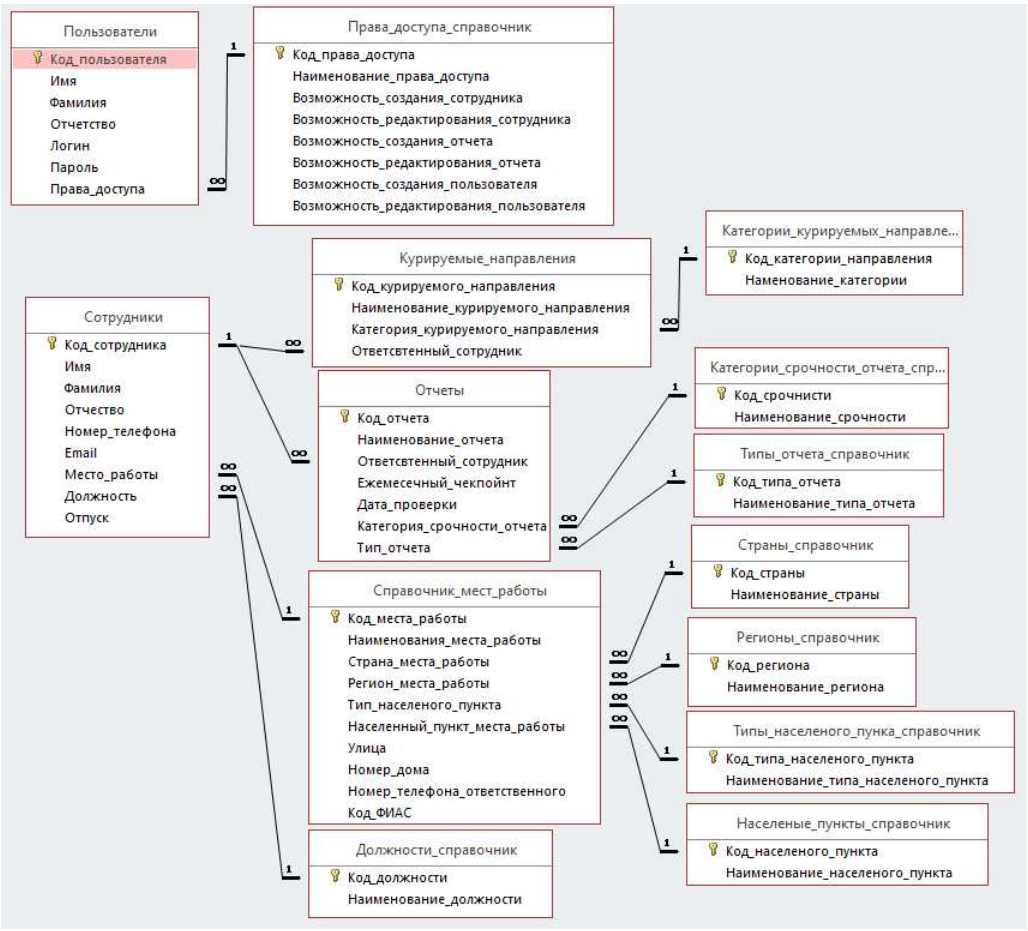

Рисунок 5 – Схема данных базы данных

2.2 Подготовка рабочей среды

Для создания информационной системы использовался редактор исходного кода VisualStudioCode, распространяющийся по бесплатной лицензии. Текстовый редактор показан на рисунке 6.

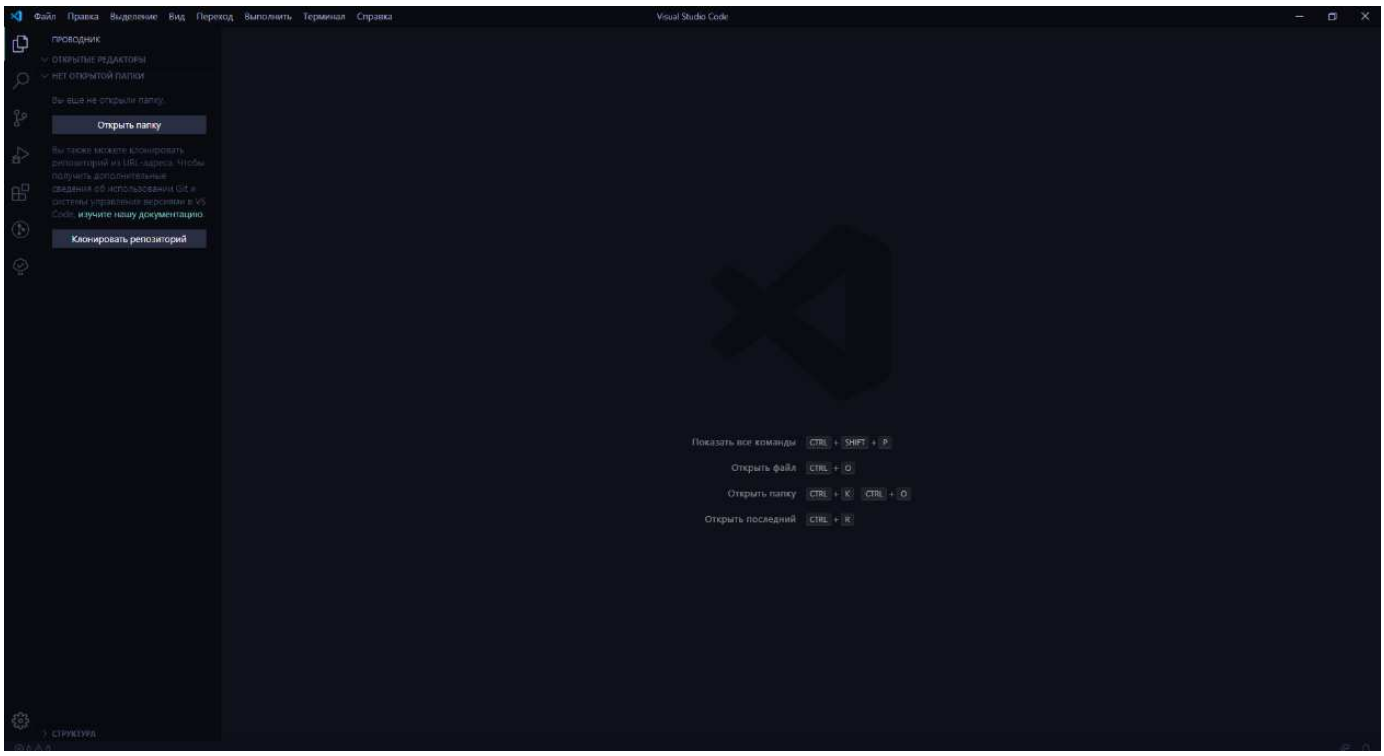

Рисунок 6 – Главный экран Visualstudiocode

В данный редактор исходного кода были установлены расширения:

– BracketPairColorizer: Выделяет парные скобки различными цветами для удобного нахождения пар скобок.

– Easy LESS: Расширение, которое автоматически компилирует файлы, написанные на препроцессоре lessв обычные файлы css.

– ESLint: Один из самых распространённых валидаторовjsкода, помогает избежать большинства банальных ошибок и привести код к одному стилю.

– MaterialTheme: Для красивой темной цветовой схемы редактора исходного кода.

– openinbrowser: Расширение, позволяющее открывать файлы в браузере при помощи горячих клавиш.

– PathIntellisense: Расширение помогает написать правильный путь до файлов.

– Prettier - Codeformatter: Автоматически форматирует код под наиболее читаемый стиль кода.

Рисунок 7 – Список используемых пакетов

Для создания информационной системы будут использоваться пакеты, показанные на рисунке 7.

– Bcryptjs: используется для шифрования пароля при сохранении пользователя и для проверки совпадения пароля при авторизации пользователя.

– Bootstrap3: используется для построения таблиц для вывода информации.

– Config: используется для создания конфигурационного файла к параметрам которого будет доступ со всех частей информационной системы.

– Express: использовался для создания серверной части информационной системы.

– Express-validator: использовался для валидации данных при отправки в роуты.

– Jsonwebtoken: используется для выдачи токена авторизованым пользователям системы на 1 час.

– Mongoose: используется для подключения и выполнения запросов к базе данных.

Далее пакеты, которые находятся в категории devDependencies:

– Concurrently: используется для запуска одним скриптом сразу и серверной части и клиентской.
– Nodemon: используется для автоматического перезапуска сервера при внесении изменений в код и структуру севера.

Пакеты, находящиеся в категории devDependencies, использовались только на стадии разработки информационной системы.

2.3 Разработка проекта

Для начала работы был создан облачный кластер для хранения первоначальных данных для хранения и систематизации данных о пользователях, это показано на рисунках 8, 9.

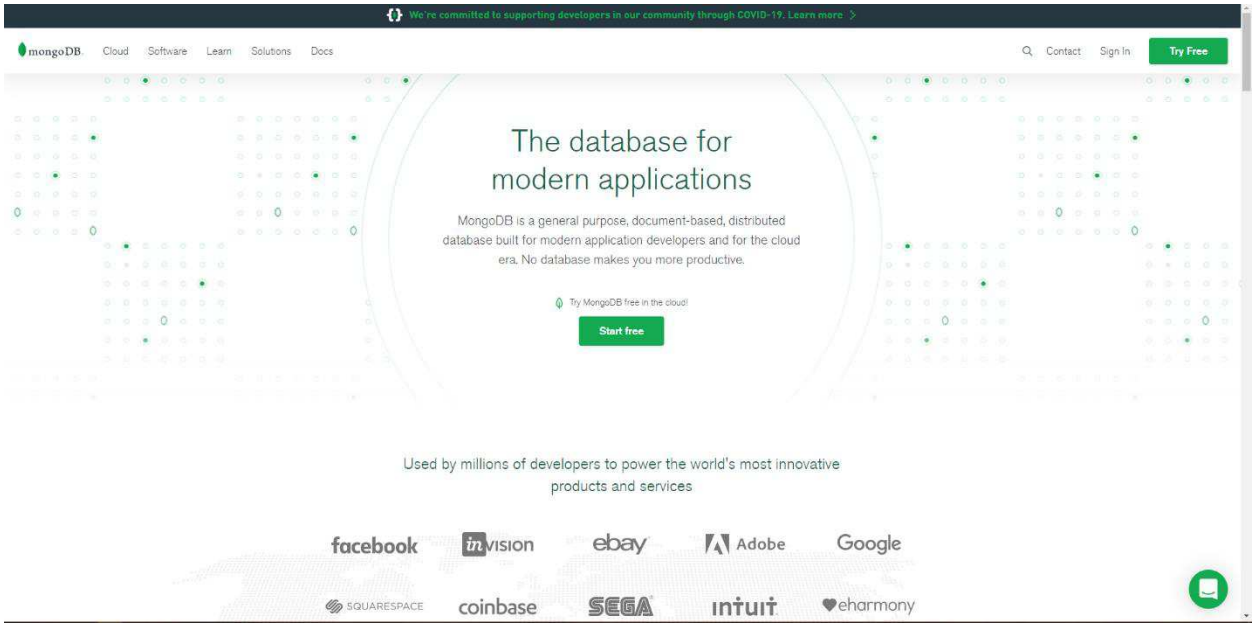

Рисунок 8 – Главная страница сайта Mongo DB

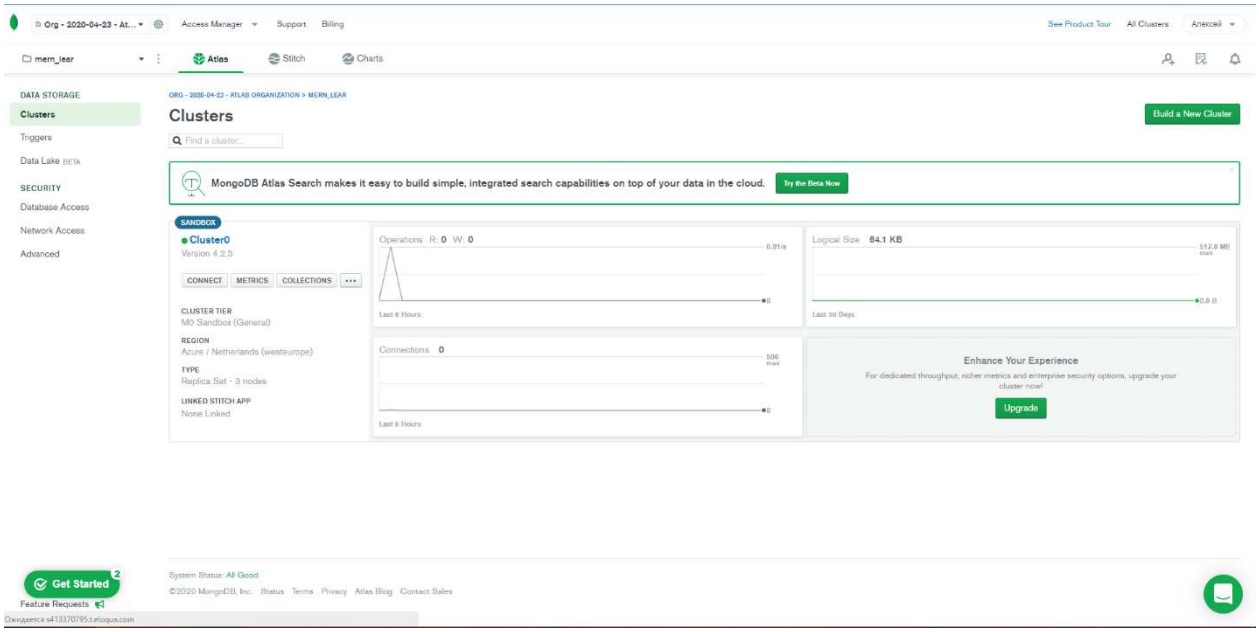

Рисунок 9 – Облачный кластер на сайте MongoDB

После создания кластера была сгенерирована ссылка для возможности подключения кластера к приложению. Далее была разработана основа серверной части приложения, где осуществляется подключение к базе данных при помощи пакета mongoose. Основная серверная часть приложения содержится на рисунке 10.

В основной серверной части подключаются основные пакеты: express и mongoose. Так же прописаны различные роуты. Далее задается порт для сервера, если порта нет в файле настроек сервера, то сервер встанет на порте 5000. Далее объявляется асинхронная функция старт, в которой происходит подключение к базе данных, если подключение не происходит, то сервер не стартует, а выдает ошибку сервера и завершает процесс. Далее была создана логика регистрации и входа. За авторизацию отвечает роут «Auth», он показан на рисунке 11.

Рисунок 10 – Основная серверная часть приложения

Роут «Auth» имеет роутер «Login». Данные, попадая в данный роутер, разбиваются на две переменные: login и password. Далее в базе данных при помощи запроса производится поиск пользователя с логином, записанным в переменную login. Если пользователь не найден, то в систему отправляется статус 400 и сообщение о том, что пользователь не найден. Если пользователь найден, то запрос выдает его параметры, и далее идет проверка его пароля. Пароль, взятый из формы, проходит шифрование и сравнивается с зашифрованным паролем, который хранился в базе данных, и если они совпадают, то пользователю выдается токен на 1 час. и он попадает в систему. Если же пароли не совпадают, то в систему возвращается ответ 400, и сообщение: «Неверный пароль, попробуйте снова».

```
router.post(
    '/login',
    async (req, res) => {

      const {login, password} = req.body

      const user = await User.findOne({ login })

      if (!user) {
        return res.status(400).json({ message: 'Пользователь не найден' })
      }

      const isMatch = await bcrypt.compare(password, user.password)

      if (!isMatch) {
        return res.status(400).json({ message: 'Неверный пароль, попробуйте снова' })
      }

      const token = jwt.sign(
        { userId: user.id },
        config.get('jwtSecret'),
        { expiresIn: '1h' }
      )

      res.json({ token, userId: user.id })

  })

module.exports = router
```

Рисунок 11 – Роут «Auth»

Форма для входа имеет два поля, в одно вводится логин, а во второе пароль. Далее данные с формы отправляются на обработку в роут «Auth». Форма для авторизации показана на рисунке 12.

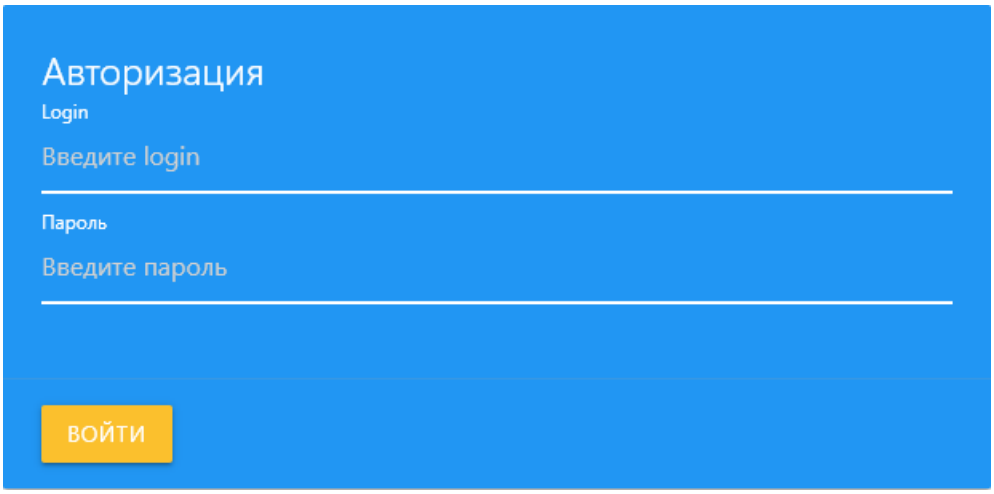

Рисунок 12 – Форма авторизации

После авторизации пользователь попадает в систему. Пользователю выдается сообщение о том, что он успешно авторизовался. Далее пользователю становятся доступны все функции системы. Параллельно с разработки авторизации велись и разработка моделей для хранения информации. Список моделей показан на рисунке 13. А пример модели показан на рисунке 14.

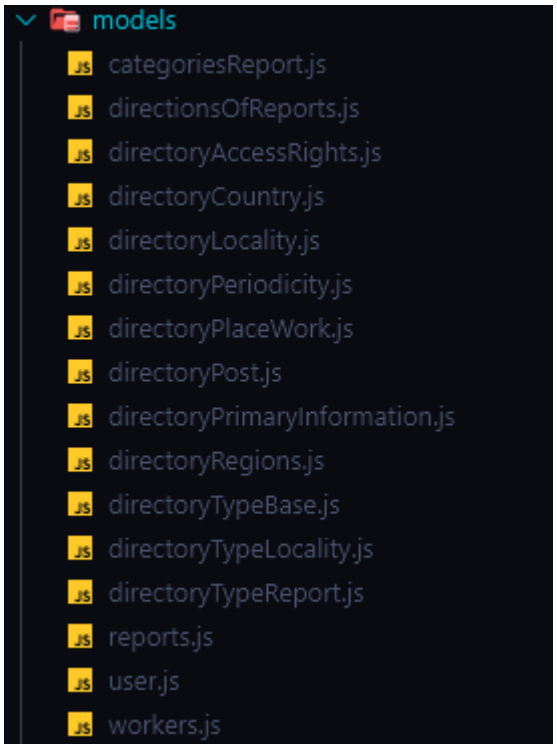

Рисунок 13 – Список моделей

Mongo DB – документно-ориентированная СУБД. Mongo DB имеет огромную маштабируемость. Для данного проекта такой уровень маштабируемости не нужен, поэтому создавались модели, по которым система обращалась к базе данных. Данные модели помогают указать строгую структуру для базы данных системы. В моделях указываются поля, которые находятся в базе данных и типы значения полей. Более строгая модель используется так как в базе данных буду хранится персональные данные, и строгая модель позволяет избежать ошибок в выполнении различных запросов с неправильными наименованиями полей.

```
const schema = new Schema({
    name: {type: String, required: true},
    surname: {type: String, required: true},
    patronymic: {type: String, required: true},
    phoneNumber: {type: String},
    phoneNumberWork: {type: String},
    email: {type: String},
    telegram: {type: String},
    placeWork: {type: Types.ObjectId, ref: 'directoryPlaceWork' },
    post: {type: Types.ObjectId, ref: 'directoryPost' },
    dataCreate: {type: Date},
    dataClose: {type: Date}

})
```

Рисунок 14 – Пример модели сотрудника

Далее была создана форма создания сотрудника. С формой также взаимодействует роут. Роут получает данные с формы и начинает внесение данных, полученных с формы в базу данных путем запроса. Форма создания сотрудника показана на рисунке 15.

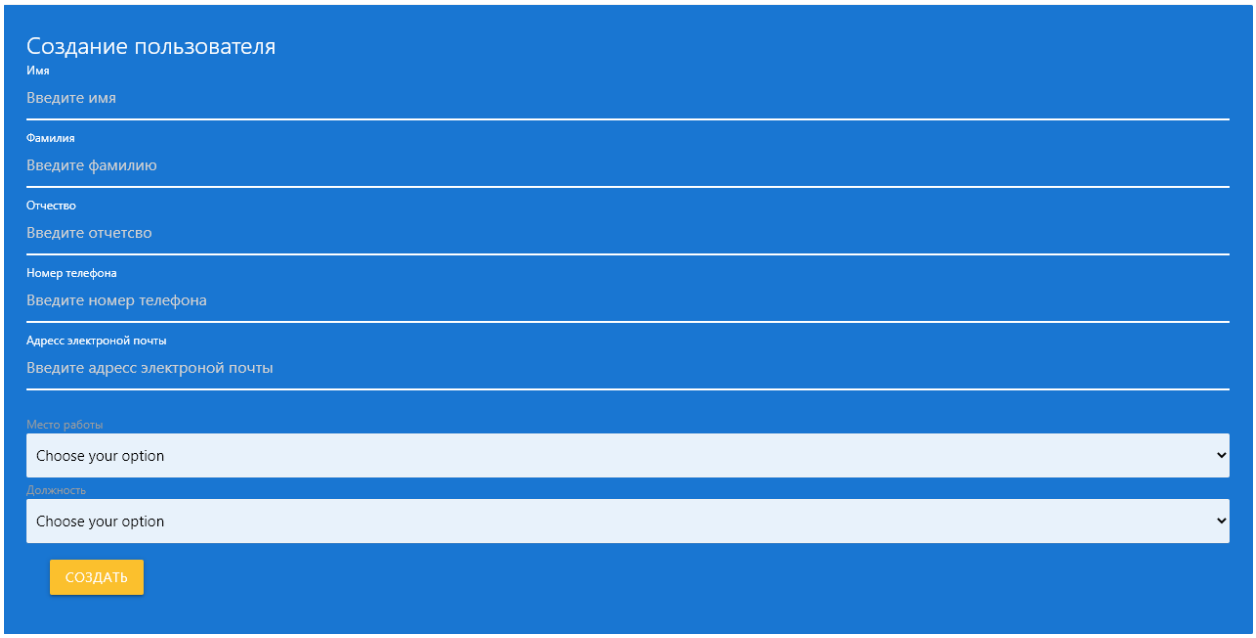

Рисунок 15 – Форма создания сотрудника

Форма создания отчета. Была создана аналогично форме создания сотрудника. Форма создания отчета показана на рисунке 16.

Рисунок 16 – Форма создания отчета

Страница для поиска отчета по выбранному сотруднику. Используется useState для внесения данных в массив form. Далее при помощи функции, прикрепленной к событию нажатие на кнопку, срабатывает функция RenderTable. В функции RenderTable происходит запрос для нахождения всех id сотрудников, которые отвечают за отчет, далее выполняется второй запрос, который по id сотрудников выдает остальную информацию о сотрудниках. Полученной после второго запроса информацией заполняется таблица вывода ответственных сотрудников. Функция «changeHandler» при каждом изменении полей в форме вносит их изменения в массив данных form. Также на данной странице присутствует функция useEffect, которая при загрузке страницы выполняет функцию searchReports. Функция searchReports выполняет запрос к роуту search и далее к роутеру searchReports, который находится в самом роуте. Отправка запроса происходит методом POST. Так как запрос должен выдать все названия отчетов, то для запроса не нужны входные данные, поэтому при поступлении данных в роут ,данные с запроса не проходят обработку и не используются. Выполняется запрос на выборку всех отчетов для вывода наименования отчетов в выпадающий список. Затем результат запроса отправляется обратно в систему и там при помощи цикла заполняются значения выпадающего списка. Данная страница показана на рисунке 17.

```
export const SearchReports = () => {
  const { request } = useHttp()
  const [reports, setReports] = useState([])
  const [form, setForm] = useState({
    id: "",
    workerid: "",
    workerids: "",
    workeridunset: "",
  })
  const [workers, setWorkers] = useState([])
  const [workersList, setWorkersList] = useState([])

  const searchReports = useCallback(async () => {
    try {
      const data = await request("/api/search/searchReports", "POST", {
        ...form,
      })
      setReports(data)
    } catch (error) {
      console.log("Chto-to poshlo ne tak")
    }
  }, [request])

  const RenderTable = async () => {
    try {
      const ids = await request("/api/search/workersReportsById", "POST", {
        ...form,
      })
      const data = await request("/api/search/workersById", "POST", ids)
      setWorkers(data)
    } catch (error) {}
  }

  const searchWorkers = useCallback(async () => {
    try {
      const data = await request("/api/search", "POST", null)
      setWorkersList(data)
    } catch (error) {
      console.log("Chto-to poshlo ne tak")
    }
  }, [request])

  const changeHandler = (event) => {
    setForm({ ...form, [event.target.name]: event.target.value })
  }
```

Рисунок 17 – Страница для поиска отчета по выбранному сотруднику

Страница для поиска сотрудников, ответственных за отчет, выполнена аналогичным образом. Для заполнения выпадающего списка с наименованиями отчетов при загрузке страницы выполняется запрос, на выборку который получает все наименования отчетов.

Страница со списком сотрудников выдает результат запроса на выборку по всем сотрудникам. Результат запроса сформирован в виде таблицы. В данной странице использовался компонент. Компонент — это заранее подготовленный элемент, который выгружается на страницу при помощи вызова компонента. На рисунке 18 показана страница со списком сотрудников. На строке 25 виден вызов компонента, в который передается массив данных. Далее на рисунке 19 показан сам компонент, который формируется из полученного массива и выводится на страницу.

```
export const ListWorkerPage = () => {
  const [workers, setWorkers] = useState([])
  const { request, loading } = useHttp()

  const renderHandler = useCallback(async () => {
    try {
      const data = await request("/api/search", "POST", null)
      setWorkers(data)
    } catch (e) {}
  }, [request])

  useEffect(() => {
    renderHandler()
  }, [renderHandler])

  if (loading) {
    return <Loader />
  }

  return <div>{!loading && <LinksList workers={workers} />}</div>
}
```

Рисунок 18 – Страница для вывода списка сотрудников

```
export const LinksList = ({ workers }) => {
  // console.log(workers)
  if (!workers.length) {
    return <p className="center">Сотрудников пока нет</p>
  }

  return (
    <table>
      <thead>
        <tr>
          <th>№</th>
          <th>Имя</th>
          <th>Фамилия</th>
          <th>Отчество</th>
          <th>Номер телефона</th>
          <th>Емайл</th>
          <th>Редактировать</th>
        </tr>
      </thead>

      <tbody>
        {workers.map((worker, index) => {
          return (
            <tr key={worker._id}>
              <td>{index + 1}</td>
              <td>{worker.name}</td>
              <td>{worker.surname}</td>
              <td>{worker.patronymic}</td>
              <td>{worker.phoneNumber}</td>
              <td>{worker.email}</td>
              <td>
                <Link to={`/detail/${worker._id}`}>Редактировать</Link>
              </td>
            </tr>
          )
        })}
      </tbody>
    </table>
  )
```

Рисунок 19 – Компонент вывода списка сотрудников

Страница редактирования отчета. На данной странице есть два выпадающих списка, первый выпадающий список нужен для выбора отчета, а второй выпадающий список нужен для выбора сотрудника, которого необходимо добавить, как ответственного за выбранный отчет. Также при

выборе отчета выводится список ответственных сотрудников, в котором можно удалить выбранного сотрудника. Данные записываются и удаляются с помощью отправки данных, в роуты в которых выполняются запросы с данными, поступающими со страницы.

Роуты нужны для функционирования системы, и они играют огромную роль. Роут создания сотрудника показан на рисунке 20. Данный роут получает данные с формы создания сотрудника и проводит валидацию данных и выполняет запрос в базу данных на сохранение сотрудника.

Рисунок 20 – Роут создания сотрудника

Далее на рисунке 21 показаны два роута для создания записи в базу данных в документе мест работы и должностей. После получения запроса к роутам они берут тело запроса и формируют запрос к базе данных, совмещая его с телом запроса. Далее идет процедура сохранения.

Рисунок 21 – Роуты для создания мест работы и должностей

Роут для удаления ответственного сотрудника из документа отчета показан на рисунке 22. Получает запрос, в теле которого находится id отчета и id сотрудника, которого требуется удалить. Далее путем запроса удаляют нужного сотрудника из документа отчета.

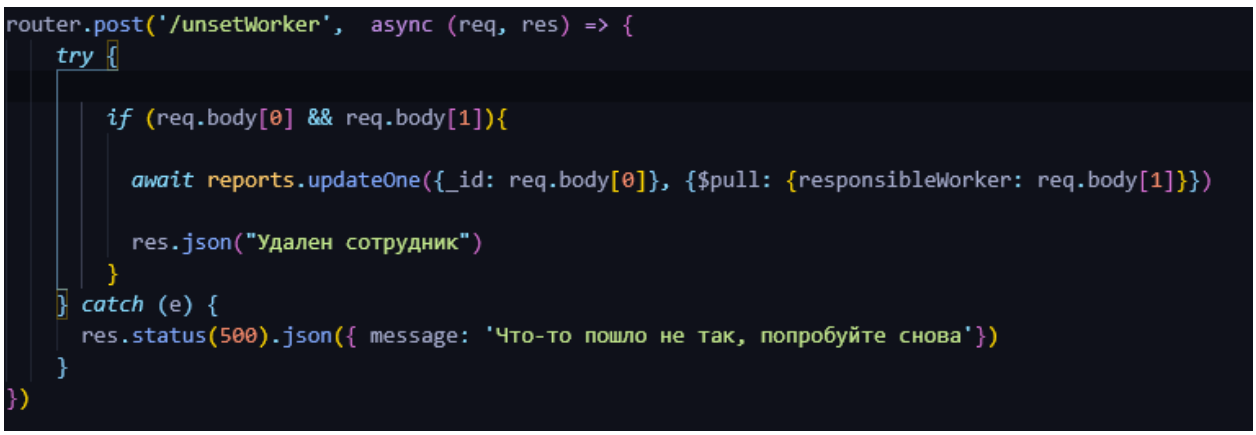

Рисунок 22 – Роут для удаления сотрудника из документа отчета

2.4 Описание информационной системы

После авторизации пользователь попадает на страницу для дальнейших переходов. Страница показана на рисунке 23.

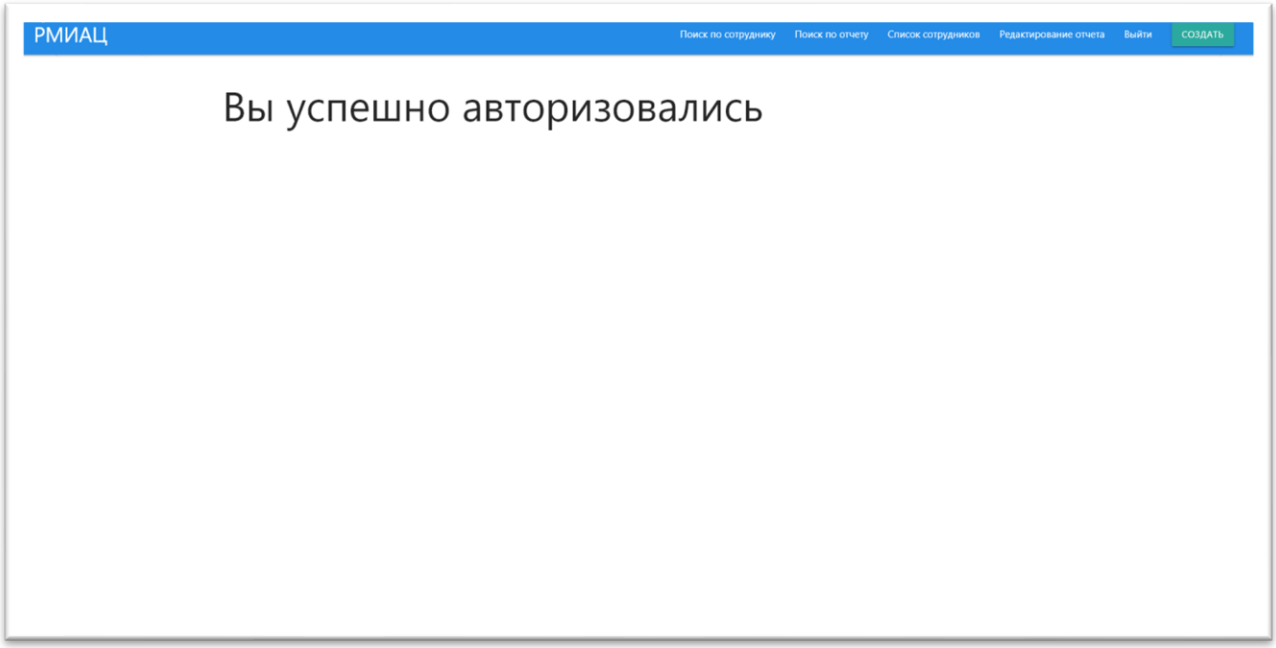

Рисунок 23 – Страница для дальнейших переходов

На данной странице пользователю открывается компонент NavBar. В компоненте NavBar присутствуют кнопки для навигации по системе. Нажимая на соответствующие кнопку можно попадать на разные страницы системы. Первая кнопка «Поиск по сотруднику», при нажатии на данную кнопку пользователь попадает на страницу поиска отчетов, за которые ответственен сотрудник. Страница поиска показана на рисунке 24.

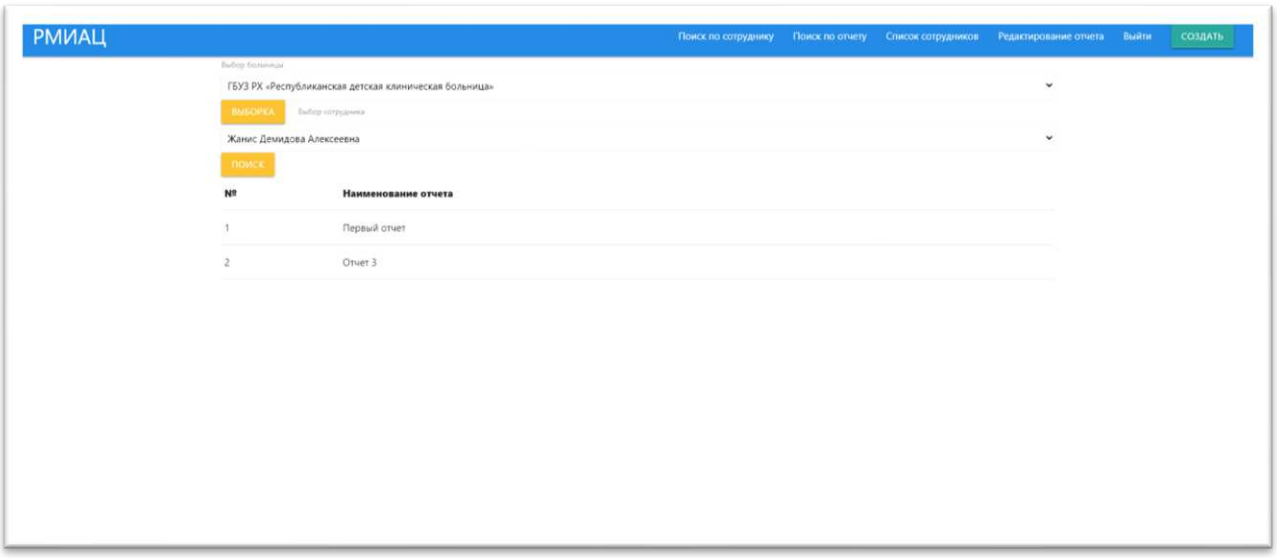

Рисунок 24 – Страница «Поиск по сотруднику»

Далее идет кнопка «Поиск по отчету», при нажатии на эту кнопку открывается страница для поиска ответственных сотрудников за отчет. На данной странице есть выпадающий список с наименованиями отчетов. После выбора отчета и нажатии на кнопку «Поиск» таблица заполняется сотрудниками, ответственными за данный отчет. Страница показана на рисунке 25.

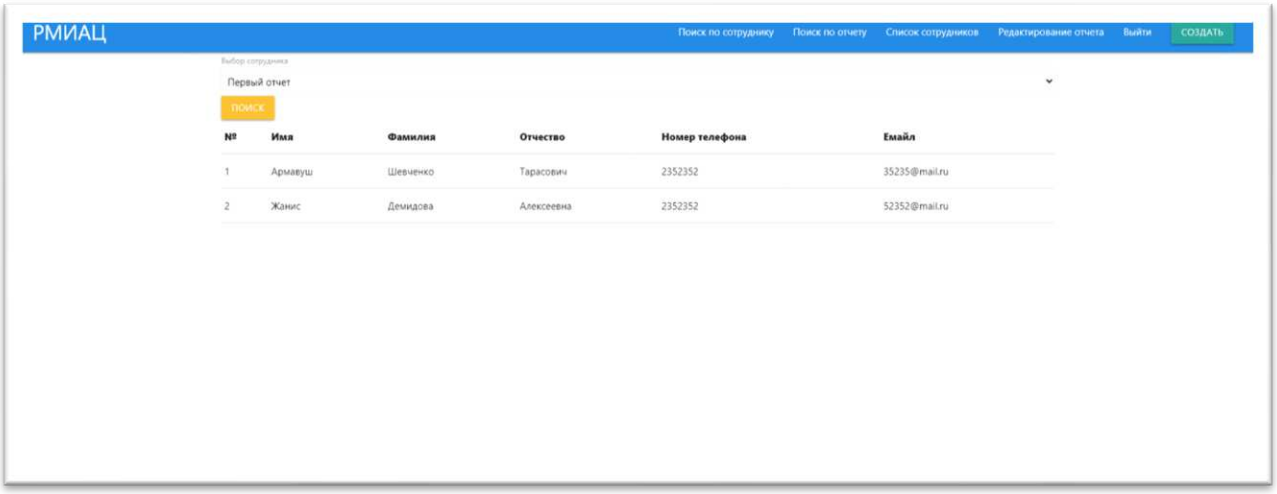

Рисунок 25 – Страница «Поиск по отчету»

Далее идет кнопка «Список сотрудников», при нажатии на эту кнопку открывается страница, на которой выводится список всех сотрудников. В данном списке у каждого сотрудника есть ссылка на форму для редактирования сотрудника. Переходя по ссылке, id сотрудника передаётся методом GET, то есть указывается в ссылке на страницу. Далее страница использует этот id для запроса в базу данных и получения информации о сотруднике для первоначального заполнения полей формы. Также массив данных form заполняется после выполнения первого запроса. При редактировании полей в форме срабатывает функция change Handler, которая меняет данные в массиве данных form. После изменений массив данных form отправляется при запросе на редактирование, где и заменяет первоначальные значения в базе данных. Страница показана на рисунке 26.

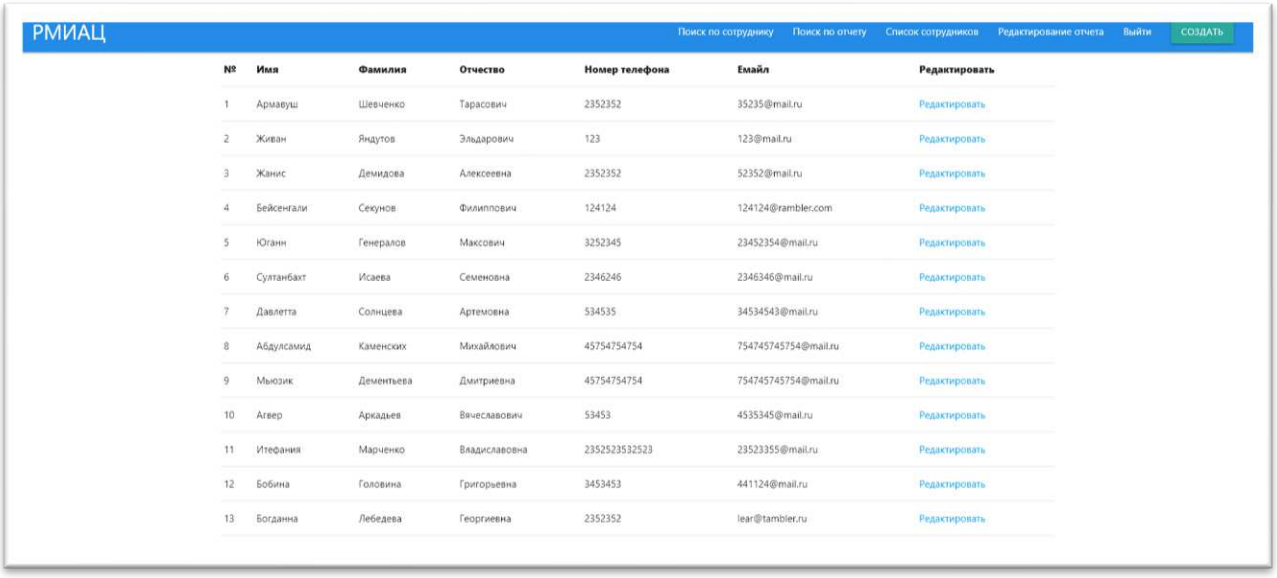

Рисунок 26 – Страница «Список сотрудников»

При нажатии на ссылку «Редактировать» пользователя перебрасывает на форму редактирования сотрудника с уже заполненными полями, которые можно редактировать. При нажатии кнопки «СОХРАНИТЬ ИЗМЕНЕНИЯ» изменения, внесённые в форму, сохраняются. Форма показана на рисунке 27.

Рисунок 27 – Страница редактирования сотрудников

Далее идет кнопка «Редактирование отчета», при нажатии на эту кнопку открывается страница, на которой можно добавлять и удалять ответственных сотрудников из отчетов. На данной странице есть два выпадающих списка и таблица. Первый выпадающий список отвечает за

выбранный отчет, а второй выпадающий список нужен при добавлении сотрудника в отчет. Если функция добавления сотрудника не нужна, то список можно не выбирать. Для удаления сотрудника нужно выбрать отчет и нажать на кнопку «Таблица», тогда выведется таблица с отечественными сотрудниками за данный отчет, и у каждого сотрудника будет кнопка «Удалить», при нажатии на кнопку сотрудник будет удален из данного отчета. Страница показана на рисунке 28.

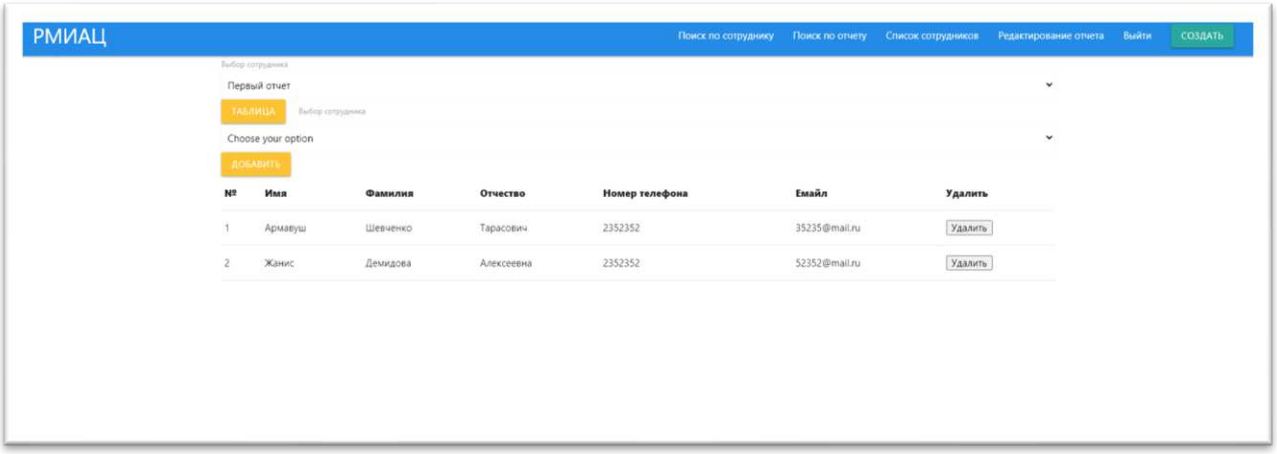

Рисунок 28 – Страница редактирования отчетов

Следующая кнопка «Выйти», при нажатии на эту кнопку пользователь выходит из системы. Вызывается функция logout, которая удаляет из локального хранилища id пользователя и токен пользователя. Если система при проверке локального хранилища не может найти токен пользователя, то система считает, что пользователь не авторизован и переадресовывает его на форму авторизации.

При нажатии на кнопку «Создать» появляется выпадающее меню, в котором есть пункты: создание сотрудника и создание отчета. Выпадающее меню показано на рисунке 29.

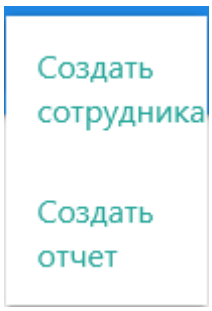

Рисунок 29 – Выпадающий список

При переходе по первой ссылке открывается форма для создания сотрудника. Форма сделана аналогичной форме редактирования сотрудника, а форма создания отчета сделана аналогично форме создания сотрудника.

2.5 Выводы по разделу «Разработка информационной системы «Реестр работников системы здравоохранения Республики Хакасия»

Созданная система имеет доступ к данным в базе данных. Может выполнять запросы на создание данных и редактирование данных. В данной системе можно отслеживать, какие сотрудники, за какие отчеты отвечают. Что и требовалось для заказчика. В системе есть функции:

- Создание сотрудника.
- Создание отчета.
- Вывод списка сотрудников.
- Редактирование сотрудников.
- Редактирование отчетов.
- Добавление сотрудников к отчетам.
- Удаление сотрудников из отчетов.

В ходе создания информационной системы были использованы Node JS, Mongo DB, React, Express. Mongo DB использовалась как база данных, в ходе разработки использовалось облачное хранилище с бесплатным тарифом, но при вводе в эксплуатацию сервер будет локальным на серверах заказчика, что позволит хранить в базе конфиденциальную информацию. В показанных

примерах была внесена информация, случайно сгенерированная генератором личностей. Поэтому в тестировании системы конфиденциальная информация не участвовала. Node JS был использован для установки и правильной связи пакетов. Express использовался для серверной части. Она запускал сервер и подключался к базе данных. Также с помощью Express создавались роуты в системе, которые позволили обрабатывать информацию и выполнять запросы к базе данных. React использовался для создания интерфейса сайта. При помощи React были заполнены таблицы при помощи циклов и перебора массива.

Итогом разработки получена система, которая удовлетворяет задачи заказчика.

3 Оценка экономической эффективности информационной системы

На сегодняшний день, универсальным методом интегральной оценки затрат по приобретению и владению объектами является TCO. Это метод основан на использовании критерия под названием «TotalCostofOwnership» (TCO) – «совокупная стоимость владения». Расчет затраты реализации проекта согласно методике TCO производится по формуле:

$$TCO = DE + IC_1 + IC_2, \quad (1)$$

где DE – прямые затраты;

IC – косвенные затраты первой и второй группы.

Так как в данном проекте косвенные затраты IC_1, IC_2 значительно малы в сравнении с прямыми, они не учитываются, следовательно, TCO = DE. Далее проводится расчет составляющих прямых затрат.

3.1 Капитальные затраты

Расчет капитальных затрат выполняется с помощью следующей формулы:

$$К = К_{пр} + К_{тс} + К_{лс} + К_{по} + К_{ио} + К_{об} + К_{оэ}, \quad (2)$$

где $К_{пр}$ – затраты на разработку информационной системы;

$К_{тс}$ – затраты на технические средства управления;

$К_{лс}$ – затраты на создание линий связи, а также интернет соединения;

$К_{по}$ – затраты на программные средства для использования готового программного продукта;

$К_{ио}$ – затраты на формирование информационной базы;

$К_{об}$ – затраты на обучение персонала;

$К_{оэ}$ – затраты на опытную эксплуатацию.

Расчет затрат на разработку информационной системы производится по следующей формуле:

$$К_{пр} = К_{зп} + К_{ипс} + К_{свт} + К_{проч}, \quad (3)$$

где $К_{зп}$ – затраты на заработную плату разработчиков;

$К_{ипс}$ – затраты на инструментальные программные средства для проектирования;

$К_{свт}$ – затраты на средства вычислительной техники для проектирования;

$К_{проч}$ – прочие затраты на разработку.

Затраты на заработную плату разработчиков $К_{зп}$ включают базовый оклад в размере 13 618 рублей и повышающие коэффициенты. Расчет приведен в таблице 2.

Так же необходимо включить в расчет отчисления во внебюджетные фонды, которые составляют 30,2% от размера заработной платы разработчика. Таким образом затраты на заработную плату разработчиков составляют:

$К_{зп}$ = 13618*1,3*1,3 *1,302 = 28369 руб.

Таблица 2 – Расчет заработной платы программиста

Состав заработной платы	Сумма, руб
Оклад	13 618
Районный коэффициент	4085
Северный коэффициент	4085
Итого	21788
НДФЛ (13%)	2704
Итого (на руки)	18 956

Сроки реализации проекта составляют 30 календарных дней.

1. Консультация с заказчиком, анализ предметной области – 7 дней.
2. Разработка прототипа – 10 дней.
3. Тестирование – 3 дня.
4. Доработка – 6 дней.
5. Введение в эксплуатацию – 4 дня.

Оценим затраты на программное обеспечение для реализации проекта.

Затраты на инструментальные ПО для проектировщика в данном проекте рассчитываются из суммы затрат на программное обеспечение, их стоимость указана в таблице 3.

Таблица 3 – Затраты на программное обеспечение

Наименование	Назначение	Стоимость, руб.	Срок использования
Операционная система MicrosoftWindows 10 Домашняя	Интерфейс для взаимодействия человека с машиной	8499	30дней
KasperskyAnti-Virus 2016	Для защиты данных от утечки	1299/год	30дней
MicrosoftOffice 2019	Для создания документации по проекту	3450	5дней
VisualStudio Code	Программирование функционала приложения	бесплатно	30дней

ОС Windows и MicrosoftOffice будут актуальны ещё два года, за это время создается примерно 2 проекта в год, следовательно, общая стоимость делится 4, а стоимость KasperskyAnti-Virus делится на 2.

$К_{ипс}$ = (8499+3450)/4 + 1299/2=3636 руб.

Затраты на средства вычислительной техники рассчитываются из суммы стоимости комплектующих для ПК, стоимость которых указана в таблице 4.

Так как длительность проекта составляет 1 месяца, то стоимость вычислительной техники полностью записать в проект нельзя, нужно узнать срок службы комплектующих и рассчитать амортизацию оборудования на срок эксплуатации в проекте.

По статистике, средний срок службы компьютерных комплектующих составляет:

Процессор – до 5 лет.

Материнская плата –от 2 – до 5 лет.

Жесткий диск (ЖД) – от 5 – до 7 лет (600 000 циклов start/stop).

Модуль оперативной памяти (ОЗУ) – около 5 лет.

Блокпитания –от 3 – до 5 лет.

Видеокарта – от 3 – до 5 лет.

ЖК-монитор – от 4 – до 5 лет.

Мышь и клавиатура – около 3-5 лет.

Исходя из этого можно, предположить, что при нормальной работе срок службы компьютера составит 5 лет. Следовательно, норма амортизации $Н_{ам}$ равна 0,2.Кроме того, срок эксплуатации ряда элементов может быть меньше 5 лет нужно предусмотреть возможность их замены. Судя по информации, следует предусмотреть возможность замены процессора и материнской платы. Для них норма амортизации $Н_{ам}$ равна 0,3.

Таблица 4 – Затраты на технические средства

Наименование оборудования	Наименование элементов	Количество	Стоимость элемента, руб.	Стоимость всего, руб.
Компьютер для программиста и дизайнера	Процессор Intel Core i3-9100F BOX	1	7 299	7 299
	Видеокарта ASUS GeForce GT 730	1	6199	6199
	Оперативная память Kingston RAM 4 ГБ	1	3099	3099
	Материнская плата GIGABYTE GA-H110N	1	4499	4499
	Корпус AeroCool Qs-182 черный	1	1899	1899
	500 ГБ Жесткий диск WD Black [WD5003AZEX]	1	4950	4950
	БлокпитанияInWinPowerman 600W	1	1950	1950

Продолжение таблицы 4

	3.8" Монитор Acer KA242Ybi [90LMF1001T02201C]	1	8 499	8 499
	Клавиатура DEXP K-5003BU	1	699	699
	Мышь проводная Jet.A OM-U54 серый	1	399	399
			Итого	39592

$$A_{год} = C_б * Н_{ам}, \quad (4)$$

где $А_{год}$–амортизация за год использования;

$С_б$ – балансовая стоимость;

$Н_{ам}$– норма амортизации.

$$А_{пр} = \frac{А_{год}}{К_{рдг}} * К_{дэ}, \quad (5)$$

где $А_{пр}$– проектная амортизация;

$К_{рдг}$ - количество рабочих дней в 2020 году;

$К_{дэ}$ - количество дней эксплуатации.

$А_{год}$ =27794 *0.2 + (7299+4499) *0.30=9097 руб.

$$А_{пр} = \frac{9097}{247} * 30 = 1141 \text{ руб.}$$

Таким образом, затраты на средства вычислительной техники для проектирования равны:

$К_{свт}$ =$А_{пр}$=1141 руб.

Так как возможна большая нагрузка на средства вычислительной техники для проектирования проекта вследствие чего есть риск поломки данных комплектующих в ходе работы необходимо заложить 1000 рублей на обслуживание.

Прочие затраты на разработку это затраты на непредвиденные расходы. Принято, что на прочие затраты необходимо оставлять не меньше 3% от общих расходов на ПО и затраты на вычислительную технику.

$К_{проч}$= 710руб.

После определения составляющих затрат на разработку информационной системы проведем их расчет:

Кпр = 28369 + 3636 + 1141 + 710= 33856 руб.

Состав проектных затрат в процентном соотношении по компонентам представлен на рисунке 30.

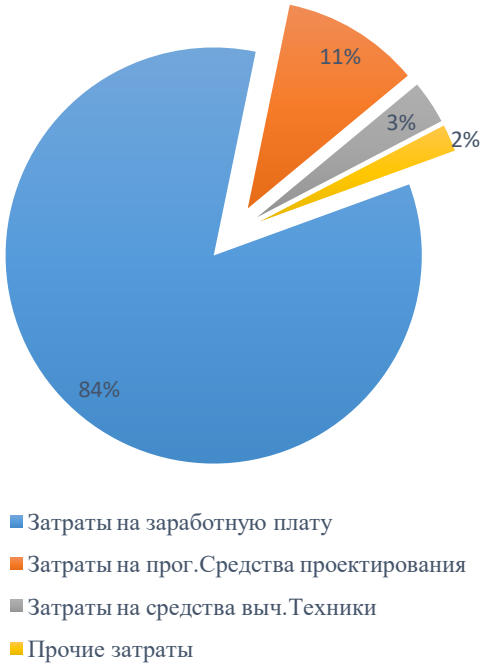

Рисунок 30– Структура затрат на разработку

Исходя из этой диаграммы можно сказать, что самым затратным пунктом затрат на разработку является зарплата разработчика на весь срок проекта, данный пункт равняется 83-ем процентам от всех затрат на разработку.

Далее проводим расчет части капитальных затрат на период внедрения системы это: *Ктс*, *К*лс, *К*по, *К*ио, *К*об, *К*оэ. Оборудование уже есть на предприятии и человек, который выполнял эту работу, также есть на предприятии. Следовательно, нет расходов на новое оборудование и на новый персонал:

*К*тс – затраты на технические средства управления, равны 0, так как используем ПК, который имеется на предприятии;

*К*по – затраты на программные средства, для работы в системы нет необходимости в программных средствах, поэтому затраты 0;

*К*лс – затраты на создание линий связи локальных сетей не требуются поэтому затраты 0;

*К*ио – затраты на формирование информационной базы, данная работа будет проводиться по ходу использования системы, поэтому затраты 0;

*К*об – затраты на обучение персонала и *Коэ* – затраты на опытную эксплуатацию, незначительны, система проста и интуитивно понятна в эксплуатации, принимаем их равными 0.

Таким образом часть капитальных затрат идущих на внедрение системы в работу принимаем равными 0 и тогда капитальные затраты равны:

К = 34453+0 = 34453 руб.

3.2 Эксплуатационные затраты

Для определения части затрат на владение информационной системой необходимо определить эксплуатационные затраты за 1 год. Формула расчета данных затрат:

$$С = С_{зп} + С_{ао} + С_{то} + С_{лс} + С_{ни} + С_{проч}, \quad (6)$$

где $С_{зп}$ – зарплата персонала, работающего с информационной системой;

$С_{ао}$ – амортизационные отчисления;

$С_{то}$ – затраты на техническое обслуживание;

$С_{лс}$ – затраты на использование глобальных сетей;

$С_{ни}$ – затраты на носители информации;

$С_{проч}$ – прочие затраты.

Согласно данным, затраты на оплату труда специалисту, работающему с информационной системой, с учетом всех повышающих коэффициентов и внебюджетных отчислений составляют 28 000 рублей. Если учесть, что сотруднику потребуется 3 часа работы с системой в месяц, то данные затраты за 1 год составят:

$С_{зп}$= (((28000 / 21)/8) * 3)*12 = 6000 рублей

Затраты на амортизационные отчисления, на техническое обслуживание, на использование глобальных сетейи на носители информации отсутствуют из-за учета их в деятельности всего предприятия,а также невозможности и нецелесообразности их выделения в отдельную статью затрат по предприятию в виду их незначительности по использованию именно данной информационной системы.

Таким образом эксплуатационные затраты — это затраты на зарплату персонала, работающего с информационной системой.

$С = С_{зп}$

С=6000 рублей

3.3 Расчет ТСО

Прямые затраты рассчитываются по формуле:

$$DE=DE_1+DE_2+ DE_3 +DE_4+ DE_5+ DE_6+ DE_7+ DE_{8,} \quad (7)$$

где DE_1– капитальные затраты. По результатам расчетов равняются 34453руб;

DE_2 – расходы на управление ИТ По результатам расчетов равняются 6000р;

DE_3 – расходы на техническую поддержку. Техническая поддержка не требуется для данного проекта следовательно данные расходы равняются;

DE_4 – расходы на разработку прикладного внутреннего ПО внутренними силами. Внутреннее ПО не требуется, следовательно, данный пункт расходов равен 0.

DE_5 – расходы на аутсорсинг. Так же не требуются, следовательно, равны 0;

DE_6 – командировочные расходы. В командировках нет необходимости, следовательно, командировочные расходы равны 0;

DE_7 – расходы равны 0 по причине использования интернета уже используемого для поддержания других систем;

DE_8 – прочие расходы. По результатам расчетов равняются 1710 руб.

DE=33856+6000+ 0 + 0 + 0 + 0+ 0+ 1710 = 39856руб.

ТСО = 42163+ 0 + 0 = 42163 руб.

Все категории затрат на разработку и на эксплуатацию описанны в таблице 5.

Таблица 5 – Составляющие стоимости владения системой

Затраты	Состав затрат	Планируемая сумма
Затраты на разработку	затраты на заработную плату разработчиков	28369руб

Продолжение таблицы 5

	затраты на инструментальные программные средства для проектирования	3636 руб
	затраты на средства вычислительной техники для проектирования	1141руб
	прочие затраты на разработку	707,7 руб
Эксплуатационные затраты	Заработная плата персонала, работающего с информационной системой	6000руб.
	прочие затраты	1710 руб

3.4 Расчет экономической эффективности ИС

Разработка систем учета уровня автоматизации и оснащённости информационно-техническим оборудованием медицинских учреждений Республики Хакасия создается с целью сокращения времени доступа к этой информации и упрощении создания отчетности, связанной с автоматизацией процессов в медицинских учреждениях.

Для оценки эффективности разрабатываемого продукта было проведено сравнение основных характеристик с текущим вариантом выполнения данной работы по пятибалльной шкале (таблица 6).

Таблица 6 – Расчет показателя качества

Показатель качества	Весовой коэффициент, b_i	Оценка X_i	
		Разраб. проект	Базовый проект
Удобство работы	0,1	4	1
Надежность	0,2	2	1
Функциональные возможности	0,2	3	1
Временная эффективность	0,4	5	1
Время обучения персонала	0,1	1	1
Комплексный показатель качества $I_{эту}$		3,5	1

Коэффициент технического уровня:

$$k_т = I_{этупр} / I_{этубаз}, \quad (8)$$

где $I_{этубаз}$ и $I_{этупр}$ – комплексные показатели качества, разрабатываемого и базового проектов.

$$k_т = 3,5 / 1 = 3,5 \quad (9)$$

Для расчета экономического эффекта рассчитаем приведенные затраты $З_i$ на единицу работ, выполняемых по базовом и разрабатываемому вариантам по формуле:

$$З_i = C_i + E_н * З_{ппі}, \quad (10)$$

где C_i – текущие эксплуатационные затраты единицы іго вида работ, руб.

$З_{ппі}$ – суммарные затраты, связанны с внедрением проекта.

$E_н = 0,33$ – нормативный коэффициент экономической эффективности.

$$С_{баз} = 28000 / 21 * 3 * 4= 16000руб/месяц, \quad (11)$$

где 28000 – оклад сотрудника, который занимается формированием отчетов об автоматизации и информатизации медицинских учреждений.

21 – количество рабочих дней в месяце у сотрудников.

3 – количество дней, когда сотрудник формирует отчеты.

4 – отчеты необходимо собирать 4 раза в год.

Для базового проекта:

$$З_{баз}= 16000 + 0,33 * 0 = 16000 \text{ руб.} \quad (12)$$

Для проекта:

$$З_{пр} = 6000 + 0,33 * 34453 = 17369 \text{ руб.} \quad (24)$$

Экономический эффект от использования разрабатываемо системы определяется по формуле:

$$Э = (З_{баз} * k_т - З_{пр}) * V, \quad (13)$$

где $З_{баз}$, $З_{пр}$ – приведенные затраты на единицу работ, выполняемых с помощью базового и проектируемого вариантов процессов обработки информации.

$k_т$ – коэффициент эксплуатационно-технической эквивалентности.

V – объем работ, выполняемых с помощью разрабатываемого проекта, натуральные единицы.

Экономический эффект от использования разрабатываемой системы:

$$Э = (16000* 3,5 – 17369) * 1 =38631 \quad (14)$$

Также необходимо рассчитать срок окупаемости затрат на разработку проекта о формуле:

$$Т_{ок} = З_{пп} / Э, \quad (15)$$

где $З_{пп}$ – единовременные затраты на разработку проекта;

Э – готовая эффективность.

Рассчитываемый срок окупаемости затрат на разработку продукта:

$$Т_{ок} = 34453 / 38631 = 0,89 \quad (16)$$

Таким образом, срок окупаемости составляет примерно два года.

Фактический коэффициент экономической эффективности разработки:

$$Е_{ф} = 1/ Т_{ок} \quad (17)$$

Нормативное значение коэффициента эффективности капитальных вложений $Е_{н} = 0,33$, если $Е_{ф} > Е_{н}$, то делается вывод об эффективности капитальных вложений.

Рассчитаем фактический коэффициент эконмической эффективности:

$$Е_{ф} = 1 / 0,85 = 1,12 \quad (18)$$

Так как $Е_{ф} = 0,68 > Е_{н}$, то разработка и внедрение разрабатываемого продукт являются эффективными, т.е. эффект от использования данной системы окупает все затраты, связанные с проектированием и эксплуатацией.

В таблице 7 приведены сводные данные экономического обоснования.

Таблица 7 – Сводные данные экономического обоснования

Показатель	Величина
Затраты на разработку системы	34453
Общие эксплуатационные затраты	60000
Экономический эффект	38631
Коэффициент экономической эффективности	1,12
Срок окупаемости	11 месяцев

Разработанная система повышает скорость получения актуальной информации о сотрудниках и формируемыми ими отчетах. Что в свою очередь экономит время и тем самым увеличивает эффективность сотрудника работающего с системой.

3.5 Оценка рисков при реализации ИС

Риски бывают двух типов:

- Внешние факторы.
- Внутренние факторы.

Данный проект подвержен небольшому количеству рисков, но некоторые из них могут оказать большое влияние на выполнение проекта. Риски проекта приведены в таблице 8.

Таблица 8 – Оценка рисков проекта

№	Описание риска	Вероятность возникновения риска			Степень воздействия риска на проект		
		Маловероятно	Вероятно	Весьма вероятно	Неопасный	Допустимый	Опасный
Внешние факторы риска							
1	Реализационный риск			+		+	
2	Риск денежных потоков		+			+	
3	Человеческий фактор			+		+	
Внутренние факторы риска							
1	Снижение качества продукции		+			+	
2	Стабильность групп разработчиков			+			+

Разработка данного проекта приходится на весьма нестабильный период, в связи с чем возникают довольно серьёзные риски.

Внешние факторы:

1. Реализационный риск – возникновение данного риска весьма вероятно, но он не является очень опасным для проекта, на момент разработки проекта экономика России нестабильна. Соответственно возможно увеличение стоимости проекта из-за форс-мажорных ситуаций, таких как подорожание программных продуктов для разработки или же комплектующих. Для устранения данного риска нет оптимального решения или же решения для его минимизации.

2. Риск денежных потоков – данный риск не сильно влияет на проект, но исключать его нельзя. Так как данный проект разрабатывается с целью получения косвенной выгоды, то напрямую его выгоду определить нельзя, при этом так же нельзя посчитать на сколько успешным будет проект и как он повлияет на организацию заказчика. Данный риск нельзя исключить или минимизировать.

3. Человеческий фактор – данный риск довольно опасен для проекта, так как велика вероятность заболевания персонала из-за мировой пандемии, вследствие чего сроки выполнения проекта могут очень сильно увеличится. Минимизировать данный риск можно по средствам изоляции персонала от внешнего мира.

Внутренние факторы:

1. Снижение качества продукции – данный риск возможен по причине демотивации персонала, но его вероятность и воздействие не велики, но могут привести к недоработкам в проекте. Снижение вероятности и опасности данного риска можно добиться повышением частоты внесения корректировок заказчиком.

2. Стабильность групп разработчиков – Возникновение данного риска весьма вероятно, и он очень опасен для данного проекта. Данный риск возможен по причине демотивации персонала.

3.6 Выводы по разделу «Оценка экономической эффективности информационной системы»

В третьем разделе были посчитаны экономические характеристики проекта. По расчетам срок окупаемости проекта составил 1 год и 4 месяца. Затраты на разработку проекта составляют 34453 руб. Совокупная стоимость владения системой, включающая затраты на создание системы, внедрение и эксплуатацию составляют 60000 руб. Коэффициент экономической эффективности проекта равен 0.71.

ЗАКЛЮЧЕНИЕ

Цель выпускной квалификационной работы – разработать информационную систему для хранения и обработки информации о сотрудниках здравоохранения Республики Хакасия, достигнута. Поставленные задачи выполнены в полном объёме.

Изучив сферу разработки веб-приложений и изучив место внедрения программного продукта, были выбраны следующие программные средства:

- Visual Studio Code.
- Node JS.
- Express.
- Mongo DB.
- React.

Выбранные программные средства являются бесплатными. Также их внедрение будет значительно комфортнее, так как у заказчика уже есть сервер с Mongo DB и Node JS.

Результатом проделанной работы является информационная система, которая включает базовый функционал.

Кроме создания информационной системы была выполнена оценка экономической эффективности, оценены риски. Затраты на разработку проекта составляют 34453 руб. Совокупная стоимость владения системой, включающая затраты на создание системы, внедрение и эксплуатацию составляют 42163 руб.

Информационная система будет использоваться руководителем ГКУЗ РХ «РМИАЦ» для своевременного поиска сотрудников, ответственных за разные отчеты.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Выполнение и защита выпускной квалификационной работы по направлению 09.03.03 «Прикладная информатика» [Электронный ресурс] : метод.указания / сост. Е. Н. Скуратенко, В. И. Кокова, И. В. Янченко ; Сиб. федер. ун-т, ХТИ – филиал СФУ. – Электрон.текстовые, граф. дан. (0,71 МБ). – Абакан : ХТИ – филиал СФУ, 2017. – 1 файл. – Режим доступа: https://e.sfu-kras.ru/pluginfile.php/1368122/mod_reso.

2. Изучение React. Полное руководство по React [Электронный ресурс]. – Режим доступа: https://learn-reactjs.ru/home

3. Особенности работы и внутреннего устройства express.js [Электронный ресурс]. – Режим доступа: https://habr.com/ru/company/ruvds/blog/414079/

4. Полный курс по JavaScript + React - с нуля до результата [Электронный ресурс]. – Режим доступа: https://www.udemy.com/course/javascript_full/

5. СТО 4.2–07–2014 Система менеджмента качества. Общие требования к построению, изложению и оформлению документов учебной деятельности – Введ. 02.07.2014. – Красноярск: ИПК СФУ, 2014. – 60 с.

6. Управление IT-проектом. Курсовая работа [Электронный ресурс] : методические указания / Е. Н. Скуратенко, И. В. Янченко, В. И. Кокова ; Сиб. федер. ун-т; ХТИ - филиал СФУ. - Электрон. текстовые дан. Электрон. граф. дан. (файла : 0,04 Мбайтов). - Абакан : ХТИ - филиал СФУ, 2018. – Режим доступа:http://89.249.130.59/docs/Met_1082.pdf.

7. Уроки React JS для начинающих [Электронный ресурс]. – Режим доступа: https://www.udemy.com/course/react-2020-complete-guide/

8. Учебный курс по React, часть 1: обзор курса, причины популярности React, ReactDOM и JSX [Электронный ресурс]. – Режим доступа: https://habr.com/ru/company/ruvds/blog/432636/

9. Что такое стек MERN, и как с ним работать? [Электронный ресурс]. – Режим доступа: https://habr.com/ru/company/piter/blog/458096

10. Documentation–Materialize[Электронный ресурс]. – Режим доступа: https://materializecss.com

11. Express – фреймворк веб-приложений Node.js [Электронный ресурс]. – Режим доступа: https://expressjs.com/ru/

12. Learn the MERN Stack - Full Tutorial (MongoDB, Express, React, Node.js) [Электронный ресурс]. – Режим доступа: https://www.youtube.com/watch?v=7CqJlxBYj-M&t

13. MERN - Сокращение Ссылок с Нуля до Деплоя (Mongo, Express, React, Node) [Электронный ресурс]. – Режим доступа: https://www.youtube.com/watch?v=ivDjWYcKDZI&t

14. Mongo DB | Выборка из БД [Электронный ресурс]. – Режим доступа: https://metanit.com/nosql/mongodb/2.4.php

15. React – A JavaScript library for building user interfaces [Электронный ресурс]. – Режим доступа: https://reactjs.org/

16. React JS. Полный Курс 2020 [Электронный ресурс]. – Режим доступа: https://www.youtube.com/watch?v=xJZa2_aldDs

17. React JS. Практический Курс 2020 (вкл. Хуки, Классы, Redux) [Электронный ресурс]. – Режим доступа: https://www.youtube.com/watch?v=ftrn50AJa2w&list=PL0lO_mIqDDFWhkCEMnLsBP51K7o78dbAJ

18. React JS на практике / Создание веб приложения [Электронный ресурс]. – Режим доступа: https://www.youtube.com/watch?v=0grybmrgc2Q&list=PL0lO_mIqDDFWjZpUTRJ8cBAsTJ5WFk4Cs

19. The most popular database for modern apps | MongoDB [Электронный ресурс]. – Режим доступа: https://www.mongodb.com/

20. Visual Studio Code - Code Editing. Redefined [Электронный ресурс]. – Режим доступа: https://code.visualstudio.com.

ПРИЛОЖЕНИЕ А

Компонент LinkList.js

```
import React from "react"
import { Link } from "react-router-dom"

export const LinksList = ({ workers }) => {
  // console.log(workers)
  if (!workers.length) {
    return <p className="center">Сотрудников пока нет</p>
  }

  return (
    <table>
      <thead>
        <tr>
          <th>№</th>
          <th>Имя</th>
          <th>Фамилия</th>
          <th>Отчество</th>
          <th>Номер телефона</th>
          <th>Емайл</th>
          <th>Редактировать</th>
        </tr>
      </thead>
      <tbody>
        {workers.map((worker, index) => {
          return (
            <tr key={worker._id}>
              <td>{index + 1}</td>
              <td>{worker.name}</td>
              <td>{worker.surname}</td>
              <td>{worker.patronymic}</td>
              <td>{worker.phoneNumber}</td>
              <td>{worker.email}</td>
              <td>
                <Link to={`/detail/${worker._id}`}>Редактировать</Link>
              </td>
            </tr>
          )
        })}
      </tbody>
    </table>
  )
}
```

Компонент ReportCard.js

```
import React, { useState, useEffect, useCallback } from "react"
import { useMessage } from "../hooks/message.hook"
//import {useHistory} from 'react-router-dom'
import { useHttp } from "../hooks/http.hook"


export const ReportCard = ({ report }) => {
  const [workers, setWorkers] = useState([])
  const [typesReports, setTypesReports] = useState([])
  const [reports, setReports] = useState([])
  const [baseTypes, setBaseTypes] = useState([])
  const [directions, setDirections] = useState([])
  const [caterogies, setCaterogies] = useState([])
  const [pereodicitys, setPereodicitys] = useState([])
  const [primaryInformations, setPrimaryInformation] = useState([])
  const { request } = useHttp()
  const message = useMessage()
  //const history = useHistory()
  const [form, setForm] = useState({
    _id: report._id,
    nameReport: report.nameReport,
    typeReport: report.typeReport,
    codePaternts: report.codePaternts,
    baseType: report.base.type,
    baseName: report.base.name,
    baseDate: report.base.date,
    baseNumber: report.base.number,
    baseOrganization: report.base.organization,
    baseAuthor: report.base.author,
    direction: report.direction,
    caterogies: report.caterogies,
    pereodicity: report.pereodicity,
    formResult: report.formResult,
    deadline: report.deadline,
    responsibleWorker: report.responsibleWorker,
    primaryInformation: report.primaryInformation,
    dataCreate: report.dataCreate,
    dataClose: report.dataClose
  })

  const updateHandler = async () => {
    console.log(form)
    const data = await request("/api/update/report", "POST", { ...form })
```

```
    message(data.message)
    //history.push('/listworker')
  }

  const searchWorkers = useCallback(async () => {
    try {
      const data = await request("/api/search", "POST", null)
      setWorkers(data)
    } catch (error) {
      console.log("Chto-to poshlo ne tak")
    }
  }, [request])
  const searchTypeReport = useCallback(async () => {
    try {
      const data = await request("/api/search/typeReport", "POST", null)
      setTypesReports(data)
    } catch (error) {
      console.log("Chto-to poshlo ne tak")
    }
  }, [request])
  const searchReport = useCallback(async () => {
    try {
      const data = await request("/api/search/reports", "POST", null)
      setReports(data)
    } catch (error) {
      console.log("Chto-to poshlo ne tak")
    }
  }, [request])
  const searchBaseType = useCallback(async () => {
    try {
      const data = await request("/api/search/baseType", "POST", null)
      setBaseTypes(data)
    } catch (error) {
      console.log("Chto-to poshlo ne tak")
    }
  }, [request])
  const searchDirection = useCallback(async () => {
    try {
      const data = await request("/api/search/direction", "POST", null)
      setDirections(data)
    } catch (error) {
      console.log("Chto-to poshlo ne tak")
```

```
    }
  }, [request])
  const searchCategories = useCallback(async () => {
    try {
      const data = await request("/api/search/categories", "POST", null)
      setCaterogies(data)
    } catch (error) {
      console.log("Chto-to poshlo ne tak")
    }
  }, [request])
  const searchPereodicitys = useCallback(async () => {
    try {
      const data = await request("/api/search/pereodicitys", "POST", null)
      setPereodicitys(data)
    } catch (error) {
      console.log("Chto-to poshlo ne tak")
    }
  }, [request])
  const searchPrimaryInformations = useCallback(async () => {
    try {
      const data = await request("/api/search/primaryInformations", "POST", null)
      setPrimaryInformation(data)
    } catch (error) {
      console.log("Chto-to poshlo ne tak")
    }
  }, [request])

  const changeHandler = (event) => {
    setForm({ ...form, [event.target.name]: event.target.value })
  }

  useEffect(() => {
    window.M.updateTextFields()
    searchWorkers()
    searchTypeReport()
    searchReport()
    searchBaseType()
    searchDirection()
    searchCategories()
    searchPereodicitys()
    searchPrimaryInformations()
  }, [searchWorkers,
      searchTypeReport,
      searchReport,
      searchBaseType,
      searchDirection,
```

```
    searchCategories,
    searchPereodicitys,
    searchPrimaryInformations])

  var elems = document.querySelectorAll('.datepicker')
  var instances = window.M.Datepicker.init(elems, {
    format: 'dd/mm/yyyy',
    showClearBtn: true,
    firstDay: 1,
    isRTL: true
  })

 return (
  <div className="card blue darken-2">
  <div className="card-content white-text">
    <span className="card-title">Создание отчета</span>
    <div>
     <div className="input-field">
      <input
        placeholder="Введите наименование отчета"
        id="nameReport"
        type="text"
        name="nameReport"
        className="yellow-input"
        value={form.nameReport}
        onChange={changeHandler}
      />
      <label htmlFor="nameReport">Наименование отчета</label>
    </div>
    <label htmlFor="typeReport">Тип отчета</label>
    <select
      class="browser-default"
      id="typeReport"
      name="typeReport"
      onChange={changeHandler}
    >
      <option value="" disabled selected>
        Choose your option
      </option>
      {typesReports.map((type) => {
       return (
        <option value={type._id}>
          {type.nameTypeReport}
        </option>
       )
      })}
    </select>
    <label htmlFor="codePaternts">Код родительской записи</label>
```

```
<select
 class="browser-default"
 id="codePaternts"
 name="codePaternts"
 onChange={changeHandler}
>
 <option value="" disabled selected>
  Choose your option
 </option>
 {reports.map((report) => {
  return (
   <option value={report._id}>
    {report.nameReport}
   </option>
  )
 })}
</select>
 <div class="blue darken-4">
 <label htmlFor="baseType">Тип основания</label>
<select
 class="browser-default"
 id="baseType"
 name="baseType"
 onChange={changeHandler}
>
 <option value="" disabled selected>
  Choose your option
 </option>
 {baseTypes.map((baseType) => {
  return (
   <option value={baseType._id}>
    {baseType.nameTypeBase}
   </option>
  )
 })}
</select>
<div className="input-field">
 <input
  placeholder="Введите наименование основания"
  id="baseName"
  type="text"
  name="baseName"
  className="yellow-input"
  value={form.baseName}
  onChange={changeHandler}
 />
 <label htmlFor="nameReport">Наименование основания</label>
</div>
<label htmlFor="responsibleWorker">Дата основания</label>
```

```
<input type="text" class="datepicker" name="baseDate" id="baseDate"></input>
<div className="input-field">
<input
placeholder="Введите номер основания"
id="baseNumber"
type="text"
name="baseNumber"
className="yellow-input"
value={form.baseNumber}
onChange={changeHandler}
/>
<label htmlFor="nameReport">Номер основания</label>
</div>
<div className="input-field">
<input
placeholder="Введите организацию основания"
id="baseOrganization"
type="text"
name="baseOrganization"
className="yellow-input"
value={form.baseOrganization}
onChange={changeHandler}
/>
<label htmlFor="nameReport">Организация основание</label>
</div>
<div className="input-field">
<input
placeholder="Введите автора основания"
id="baseAuthor"
type="text"
name="baseAuthor"
className="yellow-input"
value={form.baseAuthor}
onChange={changeHandler}
/>
<label htmlFor="nameReport">Автор основания</label>
</div>
</div>
<label htmlFor="direction">Направление</label>
<select
class="browser-default"
id="direction"
name="direction"
onChange={changeHandler}
>
<option value="" disabled selected>
```

```
          Choose your option
        </option>
        {directions.map((direction)  =>
{
          return (
            <option
value={direction._id}>
              {direction.nameDirections}
            </option>
          )
        })}
      </select>
      <label
htmlFor="caterogies">Категория</la
bel>
      <select
        class="browser-default"
        id="caterogies"
        name="caterogies"
        onChange={changeHandler}
      >
        <option value="" disabled
selected>
          Choose your option
        </option>
        {caterogies.map((categori) => {
          return (
            <option
value={categori._id}>
              {categori.nameCategories}
            </option>
          )
        })}
      </select>
      <label
htmlFor="pereodicity">Переодичнос
ть</label>
      <select
        class="browser-default"
        id="pereodicity"
        name="pereodicity"
        onChange={changeHandler}
      >
        <option value="" disabled
selected>
          Choose your option
        </option>
        {pereodicitys.map((pereodicity)
=> {
          return (
            <option
value={pereodicity._id}>

{pereodicity.namePeriodicity}
            </option>
          )
        })}
      </select>
      <div className="input-field">
        <input
          placeholder="Введите  форму
сдачи"
```

```
          id="formResult"
          type="text"
          name="formResult"
          className="yellow-input"
          value={form.formResult}
          onChange={changeHandler}
        />
        <label
htmlFor="nameReport">Форма
сдачи</label>
      </div>
      <div className="input-field">
        <input
          placeholder="Введите срок
исполнения"
          id="deadline"
          type="text"
          name="deadline"
          className="yellow-input"
          value={form.deadline}
          onChange={changeHandler}
        />
        <label
htmlFor="nameReport">Срок
исполнения</label>
      </div>
      <label
htmlFor="responsibleWorker">Ответ
свтвенный сотрудник</label>
      <select
        class="browser-default"
          id="responsibleWorker"
          name="responsibleWorker"
          onChange={changeHandler}
        >
          <option value="" disabled
selected>
            Choose your option
          </option>
          {workers.map((worker) => {
            return (
              <option
value={worker._id}>
                {worker.name}
{worker.surname}
{worker.patronymic}
              </option>
            )
          })}
        </select>
        <label
htmlFor="primaryInformation">Уров
ень сбора первичной
информации</label>
        <select
          class="browser-default"
          id="primaryInformation"
          name="primaryInformation"
          onChange={changeHandler}
        >
          <option value="" disabled
selected>
```

```
          Choose your option
        </option>

{primaryInformations.map((primaryInformation) => {
          return (
            <option value={primaryInformation._id}>

{primaryInformation.namePrimaryInformation}
            </option>
          )
        })}
      </select>
      <label htmlFor="responsibleWorker">Дата создания</label>
      <input type="text" class="datepicker" name="dateCreate"></input>
      <label htmlFor="responsibleWorker">Дата закрытия</label>
      <input type="text" class="datepicker" name="dateClose"></input>
      <div className="card-action">
        <button
          className="btn waves-effect waves-ligh yellow darken-2 marginRight10"
          onClick={updateHandler}
          // disabled={loading}
        >
          Сохранить изменения
        </button>
      </div>
    </div>
  </div>
 </div>
 )
}
```

Компонент WorkerCard.js

```
import React, { useState, useEffect, useCallback } from "react"
import { useMessage } from "../hooks/message.hook"
//import {useHistory} from 'react-router-dom'
import { useHttp } from "../hooks/http.hook"

export const WorkerCard = ({ worker }) => {
 const [placeWorks, setPlaceWorks] = useState([])
 const [posts, setPosts] = useState([])
 const { request } = useHttp()
```

```
  const message = useMessage()
  //const history = useHistory()
  const [form, setForm] = useState({
    _id: worker._id,
    name: worker.name,
    surname: worker.surname,
    patronymic: worker.patronymic,
    phoneNumber:
worker.phoneNumber,
    email: worker.email,
    placeWork: worker.placeWork,
    namePost: worker.post,
  })

  const searchPlaceWorksHandler =
useCallback(async () => {
    try {
      const data = await
request("/api/search/med", "POST",
null)
      setPlaceWorks(data)
    } catch (error) {
      console.log("Chto-to poshlo ne
tak")
    }
  }, [request])

  const searchPostsHandler =
useCallback(async () => {
    try {
      const post = await
request("/api/search/posts", "POST",
null)
      setPosts(post)
    } catch (error) {
      console.log("Chto-to poshlo ne
tak")
    }
  }, [request])

  const updateHandler = async () => {
    const data = await
request("/api/update/worker",
"POST", { ...form })
    message(data.message)
    //history.push('/listworker')
  }

  useEffect(() => {
    window.M.updateTextFields()
    searchPlaceWorksHandler()
    searchPostsHandler()
  }, [searchPlaceWorksHandler,
searchPostsHandler, setForm])

  const changeHandler = (event) => {
    setForm({ ...form,
[event.target.name]: event.target.value
})
  }
```

```
 return (
   <div className="card blue darken-2">
     <div className="card-content white-text">
       <span className="card-title">Создание сотрудника</span>
       <div>
         <div className="input-field">
           <input
             placeholder="Введите имя"
             id="name"
             type="text"
             name="name"
             className="yellow-input"
             value={form.name}
             onChange={changeHandler}
           />
           <label htmlFor="name">Имя</label>
         </div>
         <div className="input-field">
           <input
             placeholder="Введите фамилию"
             id="surname"
             type="text"
             name="surname"
             className="yellow-input"
             value={form.surname}
             onChange={changeHandler}
           />
           <label htmlFor="surname">Фамилия</label>
         </div>
         <div className="input-field">
           <input
             placeholder="Введите отчетсво"
             id="patronymic"
             type="text"
             name="patronymic"
             className="yellow-input"
             value={form.patronymic}
             onChange={changeHandler}
           />
           <label htmlFor="patronymic">Отчество</label>
         </div>
         <div className="input-field">
           <input
             placeholder="Введите номер телефона"
             id="phoneNumber"
             type="text"
             name="phoneNumber"
             className="yellow-input"
             value={form.phoneNumber}
             onChange={changeHandler}
           />
```

```
          <label
htmlFor="phoneNumber">Номер
телефона</label>
        </div>
        <div className="input-field">
          <input
            placeholder="Введите
адресс электроной почты"
            id="email"
            type="text"
            name="email"
            className="yellow-input"
            value={form.email}
            onChange={changeHandler}
          />
          <label
htmlFor="email">Адресс
электроной почты</label>
        </div>
        <label
htmlFor="placeWork">Место
работы</label>
        <select
          class="browser-default"
          id="placeWork"
          name="placeWork"
          onChange={changeHandler}
        >
          <option
value={worker.placeWork}    disabled
selected>
            Текущее место работы
          </option>

{placeWorks.map((placeWork) => {
            return (
              <option
value={placeWork._id}>{placeWork.
namePlaceWork}</option>
            )
          })}
        </select>
        <label
htmlFor="namePost">Должность</la
bel>
        <select
          class="browser-default"
          id="namePost"
          name="namePost"
          onChange={changeHandler}
        >
          <option    value={worker.post}
disabled selected>
            Текущая должность
          </option>
          {posts.map((post) => {
            return                    <option
value={post._id}>{post.namePost}</
option>
          })}
        </select>
        <div className="card-action">
```

```
          <button
            className="btn waves-effect waves-ligh yellow darken-2 marginRight10"
            onClick={updateHandler}
            // disabled={loading}
          >
            Сохранить изменения
          </button>
        </div>
      </div>
    </div>
  </div>
 )
}
```

Хук http

```
import { useState, useCallback } from "react"

export const useHttp = () => {
 const [loading, setLoading] = useState(false)
 const [error, setError] = useState(null)
 const request = useCallback(
  async (url, method = "GET", body = null, headers = {}) => {
    setLoading(true)
    try {
      if (body) {
        body = JSON.stringify(body)
        headers["Content-Type"] = "application/json"
      }
      const response = await fetch(url, { method, body, headers })
      const data = await response.json()
      if (!response.ok) {
        throw new Error(data.message || "Что-то пошло не так")
      }
      setLoading(false)
      return data
    } catch (e) {
      setLoading(false)
      setError(e.message)
      throw e
    }
  },
  []
 )

 const clearError = useCallback(() => setError(null), [])
 return { loading, request, error, clearError }
}
```

Модель worker.js

```
const {Schema, model, Types} = require('mongoose')

const schema = new Schema({
   name: {type: String, required: true},
   surname: {type: String, required: true},
   patronymic: {type: String, required: true},
   phoneNumber: {type: String, required: true},
   phoneNumberWork: {type: String},
   email: {type: String},
   telegram: {type: String},
   placeWork: {type: Types.ObjectId, ref: 'directoryPlaceWork', required: true},
   post: {type: Types.ObjectId, ref: 'directoryPost', required: true},
   dataCreate: {type: Date},
   dataClose: {type: Date}

})

module.exports = model('workers', schema)
```

Модель report.js

```
const {Schema, model, Types} = require('mongoose')

const schema = new Schema({
   nameReport: {type: String, required: true},
   typeReport: { type: Types.ObjectId, ref: 'directoryTypeReport' },
   codePaternts: {type: Types.ObjectId, ref: 'reports' },
   base: {
      type: {type: Types.ObjectId, ref: 'directoryTypeBase'},
      name: {type: String},
      date: {type: Date},
      number: {type: String},
      organization: {type: String},
      author: {type: String}
   },
   direction: {type: Types.ObjectId, ref: 'directionsOfReport'},
   caterogies: {type: Types.ObjectId, ref: 'catogoriesReport'},
   pereodicity: {type: Types.ObjectId, ref: 'directoryPereodicity'},
   formResult: {type: String},
   deadline: {type: String},
   responsibleWorker: [{ type: Types.ObjectId, ref: 'workers' }],
```

```
  primaryInformation: {type: Types.ObjectId, ref: 'directoryPrimaryInformation'},
  dateCreate: {type: Date},
  dateClose: {type: Date}
})

module.exports = model('reports', schema)
```

Модель placeWork.js

```
const {Schema, model, Types} = require('mongoose')

const schema = new Schema({
  namePlaceWork: {type: String, required: true},
  countryPlaceWork: [{ type: Types.ObjectId, ref: 'directoryCountry' }],
  regionPlaceWork: [{ type: Types.ObjectId, ref: 'directoryRegions' }],
  typeLocalityPlaceWork: [{ type: Types.ObjectId, ref: 'directoryTypeLocality' }],
  localityPlaceWork: [{ type: Types.ObjectId, ref: 'directoryLocality' }],
  streetPlaceWork: {type: String},
  houseNumberPlaceWork: {type: String},
  email: {type: String},
  phoneNumberResponsibleWorker: {type: String},
  codePaternts: {type: Types.ObjectId, ref: 'directoryPlaceWork' },
  OID: {type: String},
  dataCreate: {type: Date},
  dataClose: {type: Date}
})

module.exports = model('directoryPlaceWork', schema)
```

Основаная серверная часть app.js

```
const express = require('express')
const config = require('config')
const mongoose = require('mongoose')

const app = express()

app.use(express.json({ extended: true}))

app.use('/api/auth', require('./routes/auth.routes'))
app.use('/api/create', require('./routes/create.routers'))
```

```
app.use('/api/search',
require('./routes/search.routers'))
app.use('/api/update',
require('./routes/update.routers'))
app.use('/api/delete',
require('./routes/delete.routers'))

const PORT = config.get('port') ||
5000

async function start() {
  try {
    await
mongoose.connect(config.get('Mongo
Url'), {
      useNewUrlParser: true,
      useUnifiedTopology: true,
      useCreateIndex: true,
      useFindAndModify: false
    })
    app.listen(PORT, () =>
console.log(`App has been started on
port ${PORT}`))
  } catch (e) {
    console.log('Server error',
e.message)
    process.exit(1)
  }
}

start()
```

Основная часть приложения React

```
import React from "react"
import { BrowserRouter as Router }
from "react-router-dom"
import "materialize-css"
import { useRoutes } from "./routes"
import { useAuth } from
"./hooks/auth.hook"
import { AuthContext } from
"./context/AuthContext"
import { Navbar } from
"./components/Navbar"

function App() {
  const { token, userId, login, logout }
= useAuth()
  const isAuthenticated = !!token
  const routes =
useRoutes(isAuthenticated)
  return (
    <AuthContext.Provider
      value={{
        token,
        login,
        logout,
        userId,
        isAuthenticated,
      }}
    >
      <Router>
```

```
      {isAuthenticated && <Navbar />}
      <div
className="container">{routes}</di
v>
    </Router>
  </AuthContext.Provider>
 )
}

export default App
```

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

Отпечатано в одном экземпляре.

Библиография 20 наименований.

Один экземпляр сдан на кафедру.

«____» _____ 2020 г.

_____ Утьев Алексей Сергеевич

(подпись)

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Хакасский технический институт – филиал ФГАОУ ВО
«Сибирский федеральный университет»

Кафедра прикладной информатики, математики и естественно-научных дисциплин

УТВЕРЖДАЮ
Заведующий кафедрой
_____ Е. Н. Скуратенко
подпись
« 26 » 06 2020 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.03 Прикладная информатика

Тема: Разработка информационной системы «Реестр работников системы здравоохранения Республики Хакасия »

Руководитель 26.06.20г. зав. кафедрой, к.т.н Е.Н. Скуратенко
подпись, дата

Выпускник Утв 26.06.20 А.С. Утьев
подпись, дата

Консультанты
по разделам:

Экономический 26.06.20г. Е. Н. Скуратенко
подпись, дата

Нормоконтролер 26.06.20г. В. И. Кокова
подпись, дата

Абакан 2020

Студенту Утьеву Алексею Сергеевичу

Группа ХБ 16-03

Направление 09.03.03 Прикладная информатика

Тема выпускной квалификационной работы: Разработка информационной системы «Реестр работников системы здравоохранения Республики Хакасия ».

Утверждена приказом по институту № 216 от 06.04.2020 г.

Руководитель ВКР: Е.Н. Скуратенко, зав. кафедрой, к.т.н, ХТИ – филиал СФУ

Исходные данные для ВКР: информация о деятельности системы Здравоохранения Республики Хакасия, информация о локальном сервере.

Перечень разделов ВКР:

1. Анализ предметной области.
2. Разработка информационной системы «Реестр работников системы здравоохранения Республики Хакасия».
3. Оценка экономической эффективности информационной системы.

Перечень графического материала: нет

Руководитель ВКР _____ Е. Н. Скуратенко
подпись

Задание принял к исполнению _____ А. С. Утьев
подпись

«06» 04 2020 г.