МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование)

09.03.03 Прикладная информатика
(код и наименование направления подготовки, специальности)

Бизнес-информатика
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему Автоматизация логистических процессов предприятия пищевой промышленности

Обучающийся А.А. Клышко
(Инициалы Фамилия) (личная подпись)

Руководитель Н.Н. Рогова
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2023

Аннотация

Бакалаврская работа выполнена на тему «Автоматизация логистических процессов предприятия пищевой промышленности».

Цель работы заключается в автоматизации логистических процессов предприятия пищевой промышленности.

Во введении описывается: актуальность темы, объект и предмет, а также цель работы и требуемые задачи.

В первой главе рассмотрена характеристика предприятия пищевой промышленности АО "Дружба народов Нова", проведено моделирование бизнес-процесса, который отражает логистические процессы на территории предприятия, определены требования к разрабатываемому программному продукту, оценены существующие программы, для решения рассматриваемых задач.

Во второй главе проведено проектирование программного продукта, а именно созданы диаграммы UML, а также, построены концептуальная и логическая модели базы данных.

В третьей главе показана реализация нескольких сценариев работы системы, а также оценена экономическая эффективность внедрения информационной системы.

Бакалаврская работа состоит из 68 страниц и включает 30 рисунков, 4 таблицы, 20 источников, 5 приложений.

Оглавление

Введение

Тема выпускной квалификационной работы «Автоматизация логистических процессов предприятия пищевой промышленности». В работе рассматривается автоматизации внутренних логистических процессов в работе предприятия пищевой промышленности, которые потребовали изменения и автоматизации одной из функции отдела логистики после внедрение новых технических средств контроля доступа автомобилей, ужесточение правил перевозки грузов по территории Крыма, запрет на парковку транспорта на территории предприятия, сезонное увеличение отгрузки продукции.

Данная тема является актуальной, поскольку в АО "Дружба народов Нова" постоянно идет автоматизация процессов, которые повышают эффективность работы предприятия и его прибыль, а также помогают выстраивать бизнес-процессы, согласно требованиям и рекомендациям органов местного самоуправления. Поэтому актуально разработать автоматизированную систему логистических процессов предприятия пищевой промышленности, для повышения эффективности работы склада компании и загрузки территории предприятия.

В работе объект исследования – логистические процессы.

Предмет исследования – автоматизация логистических процессов предприятия.

Цель работы – автоматизация логистических процессов предприятия пищевой промышленности.

Для достижения поставленной цели необходимо решить ряд задач:

- исследовать проблему, затрагиваемую в теме, обосновать ее актуальность;
- провести сравнительный анализ аналогов;
- описать идею предлагаемого решения;

- провести проектирование базы данных для информационной системы;
- разработать прототип информационного обеспечения и программного модуля;
- провести тестирование различных сценариев работы программы;
- рассчитать экономическую эффективность проекта.

Структура бакалаврской работы состоит из введения, трех глав, в которых рассматривается вопросы постановки задачи, проектирования и реализации системы для автоматизации логистических процессов предприятия пищевой промышленности, а также заключения, списка литературы и используемых источников и приложения с кодом программы.

Глава 1 Анализ предметной области АО «Дружба народов Нова»

1.1 Технико-экономические характеристики предметной области

«Дружба народов» основано в 1957 году. Предприятия агрохолдинга - четыре флагмана АПК Республики Крым. В состав вертикально интегрированного агропромышленного холдинга входят несколько ключевых направлений: растениеводство, садоводство, производство кормов, свиноводство, птицеводство, мясопереработка. В агрохолдинге работает более 3 тысяч человек.

«Дружба народов» реализует колбасные изделия, мясо птицы, субпродукты, мясные полуфабрикаты, зерновые культуры, фрукты через торговые сети и дистрибьюторов России, а также в Китае, Вьетнаме, Казахстане, Таджикистане, Армении, Киргизии, Абхазии, Республике Беларусь.

Структура управления АО «Дружба народов Нова» представлена на рисунке 1.

Главой организации является генеральный директор - отвечает за управление компанией, и которому в свою очередь подчиняются все остальные отделы, а именно:

- финансовый отдел;
- производственный отдел;
- отдел растениеводства;
- отдел продаж,
- отдел кадров,
- юридический отдел,
- склад.

Рисунок 1 – Структура управления компанией

Данная схема является упрощенной схемой представления структуры компании, основными процессами компании является растениеводство и производство мясной продукции. Но в ходе выпускной квалификационной работы большее внимание уделено не производственным процессам компании, а поддерживающим, в работе будут рассмотрены отделы, влияющие больше на логистические процессы компании, так как отгрузка всех товаров осуществляется централизовано со склада компании.

Финансовый отдел включает в себя финансового директора и бухгалтерию. Бухгалтерия отвечает за ведение бухгалтерского, финансового, налогового и управленческого учета и прочие функции финансового отдела:

- формирование финансовых планов и контроль исполнения;
- организация и ведение налогового и бухгалтерского учета;
- выставление счетов, актов;
- составление и подписание договоров;
- управление денежными средствами предприятия.

Отдел продаж состоит из коммерческого директора и менеджеров по продажам. Отдел продаж занимается поиском клиентов, обзвоном клиентов, ведет с ними переписку и т.д.

Функции отдела продаж:

- исследование потребителей;
- обработка заявок клиентов;
- анализ и выбор поставщиков товара;
- поиск клиентов;
- составление и контроль воронки продаж;
- анализ рынка;
- ведение отчетности;
- обработка входящих запросов.

Отдел складирования – в этот отдел входит заведующим складом и

работники склада, они отвечают за следующие функции:

- разгрузка и приемка грузов;
- комплектация;
- внутрискладская транспортировка;
- операции, связанные с отпуском материалов и изделий;
- учет движения запасов материалов на складе.

Обязанности логиста:

- обработка поступающих заявок на выделение необходимого для выполнения перевозки грузов транспортных средств;
- подготовка и оформление сопроводительных и путевых документов, контроль документооборота;
- контроль выполнения водителем регламента доставки грузов;
- работа с информационными базами данных.

1.2 Функциональная модель деятельности предприятия пищевой промышленности

Сегодня существует две популярные методологии описания функциональной модели организации» [5]:

- структурный подход;
- объектно-ориентированный подход.

Структурный подход «основан на принципе алгоритмической декомпозиции. Объектно-ориентированный подход основан на декомпозиции объектов» [5].

Модель IDEF0 принято начинать «с представления всей системы в целом - интерфейс функционального блока с дугами, выходящими за пределы обрабатываемой области» [19, с.19]. Такая диаграмма с одной функциональной единицей называется контекстной и показана на рисунке 2.

Входной информацией служат:

- информация об автомобильном номере;
- договор на отгрузку товара;
- товары на склад;
- информация по времени загрузки и отгрузки продукции.

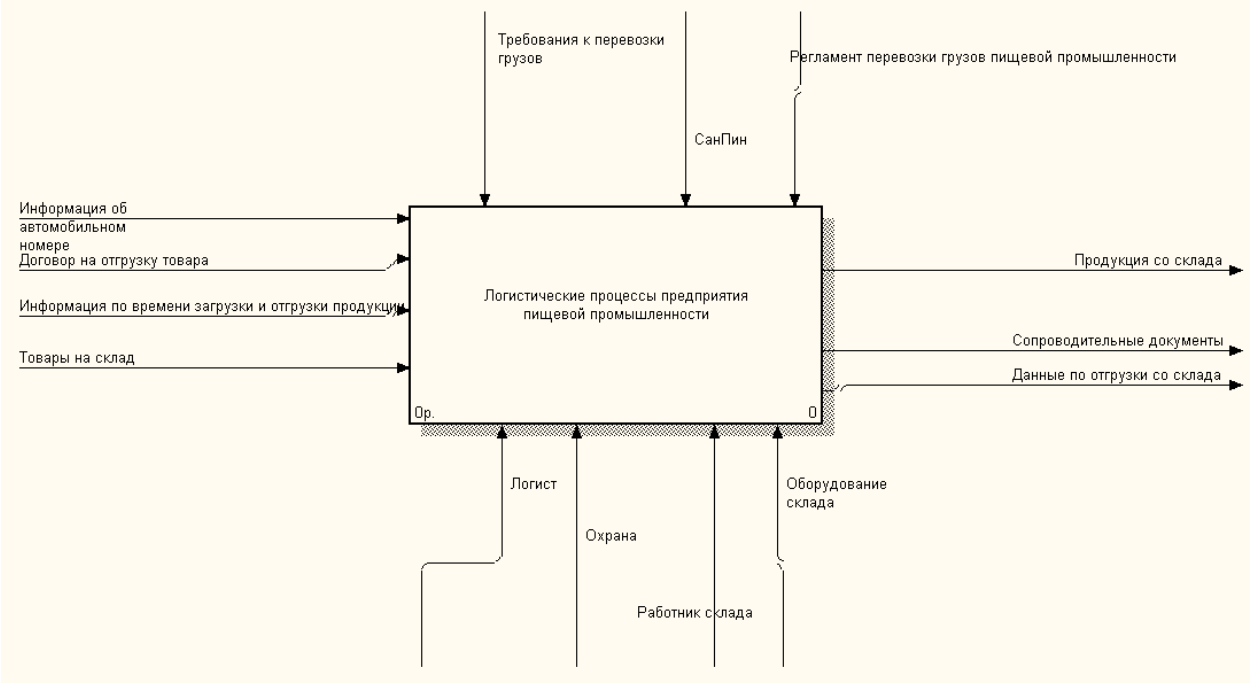


Рисунок 2 – Контекстная диаграмма «Логистические процессы предприятия пищевой промышленности»

Выходной информацией является:

- продукция со склада;
- сопроводительные документы
- данные по отгрузки со склада.

Управляющей информацией являются:

- требования к перевозке грузов,
- регламент перевозки грузов пищевой промышленности
- СанПин.

Механизмами являются люди и оборудование, которые задействованы в логистических процессах предприятия, а именно:

- логист;
- охрана,
- работник склада,
- оборудование склада.

Рассмотрим основные операции более подробно на рисунке 3

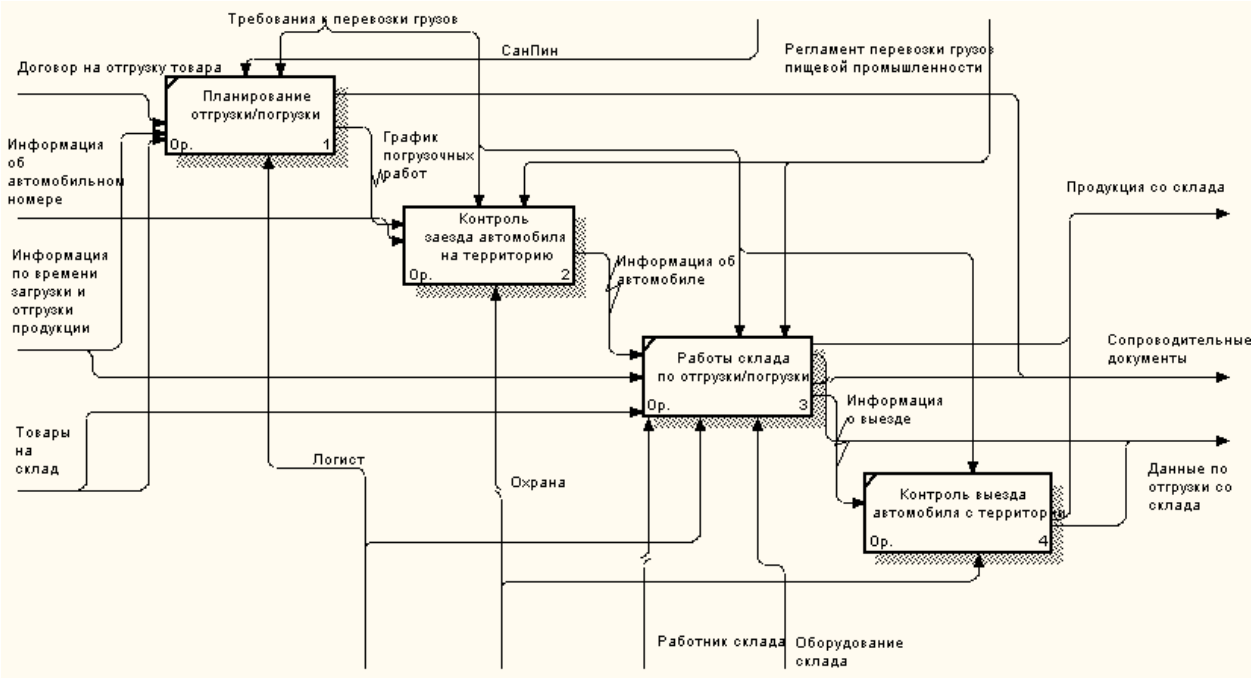


Рисунок 3 –Логистические процессы на предприятии пищевой промышленности. Модель «Как есть»

Основными операциями верхнего уровня являются:

- планирование отгрузки/погрузки,
- контроль заезда автомобиля на территорию,
- работы склада по отгрузки/погрузки,
- контроль выезда автомобиля с территории.

Рассмотрены подробно операции бизнес-процесса логистические процессы на предприятии пищевой промышленности более подробно в таблице 1.

Таблица 1 - Бизнес-процесс «Логистические процессы на предприятии пищевой промышленности». Модель «Как есть»

Название	Вход	Выход	Управление	Механизм
Планирование отгрузки/ погрузки	Договор на отгрузку товара. Информация по времени загрузки и отгрузки продукции. Товары на склад.	Сопроводительные документы. График погрузочных работ	Требования к перевозке грузов СанПин	Логист
Контроль заезда автомобиля на территорию	График погрузочных работ информация об автомобильном номере	Информация об автомобиле	Требования к перевозке грузов Регламент перевозки грузов пищевой промышленности	Охрана
Работы склада по отгрузки/ погрузки	Информация об автомобиле Информация по времени загрузки и отгрузки продукции Товары на склад	Информация о выезде. Данные по отгрузки со склада. Продукция со склада	Требования к перевозке грузов Регламент перевозки грузов пищевой промышленности	Работник склада Оборудование склада Логист
Контроль выезда автомобиля с территории	Информация о выезде	Данные по отгрузки со склада Продукция со склада	Требования к перевозке грузов	Охрана

Далее проведем декомпозицию процесса контроль заезда автомобиля на территорию. Рассмотрение и автоматизация данного процесса более подробно возникло по следующим причинам:

- ужесточения требований к контролю грузового транспорта на территории предприятия;

– внедрение системы видеонаблюдения на контрольно-пропускных пунктах предприятия;
– более жестким требованиям по простою и парковке транспорта на территории предприятия, с учетом текущего уменьшения территории;
– планированием посменной работы склада в три смены с учетом сезонности работы.

Процесс заезда автомобиля на территорию состоит из следующих блоков:

– контроль доступа автомобиля,
– фиксация заезда автомобиля,
– контрольное взвешивание.

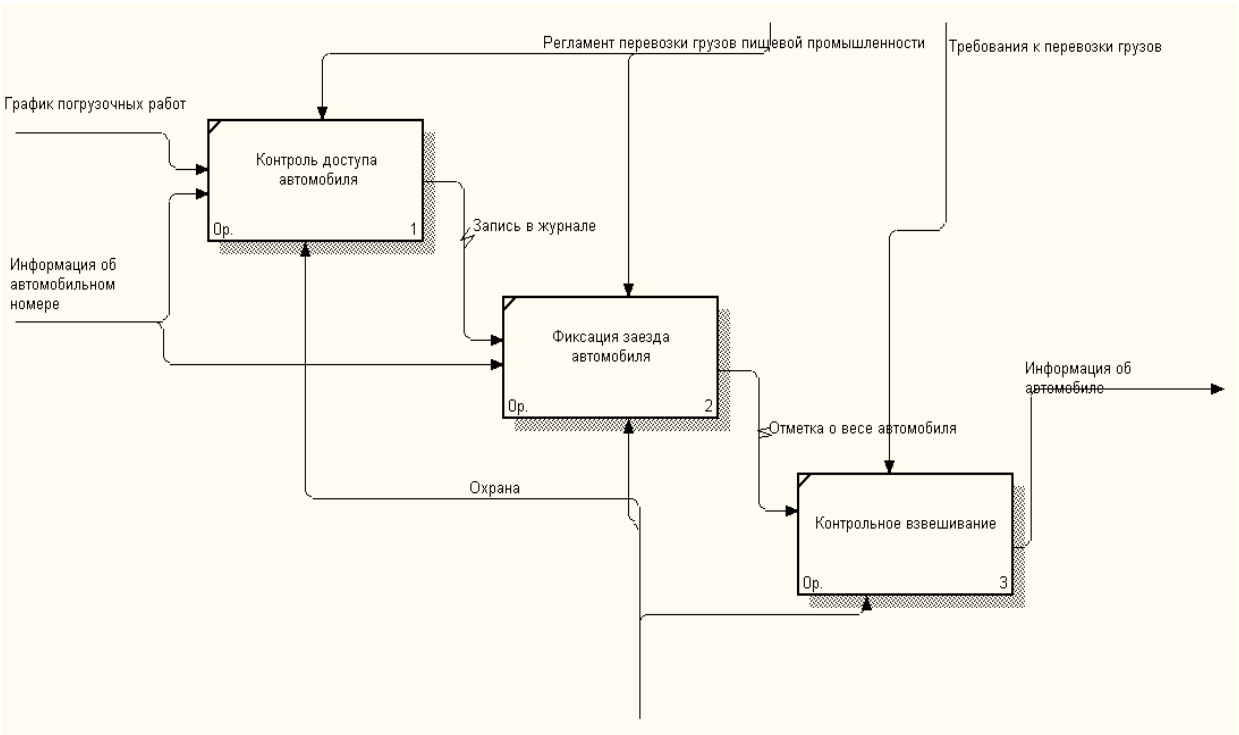


Рисунок 4 – Контроль заезда автомобиля на территорию. Модель «Как есть»

Рассмотрим операции более подробно (таблица 2).

Таблица 2 - Бизнес-процесс «Контроль заезда автомобиля на территорию». «Как есть»

Название	Вход	Выход	Управление	Механизм
Контроль доступа автомобиля	График погрузочных работ Автомобиль	Запись в журнале	Регламент перевозки грузов пищевой промышленности	Охрана
Фиксация заезда автомобиля	Автомобиль Запись в журнале	Отметка о весе автомобиля	Регламент перевозки грузов пищевой промышленности	Охрана
Контрольное взвешивание	Отметка о весе автомобиля	Информация об автомобиле	Требования к перевозке грузов	Охрана

Далее представлена диаграмма дерева узлов, которая описывает логистические процессы предприятия пищевой промышленности рисунок 5

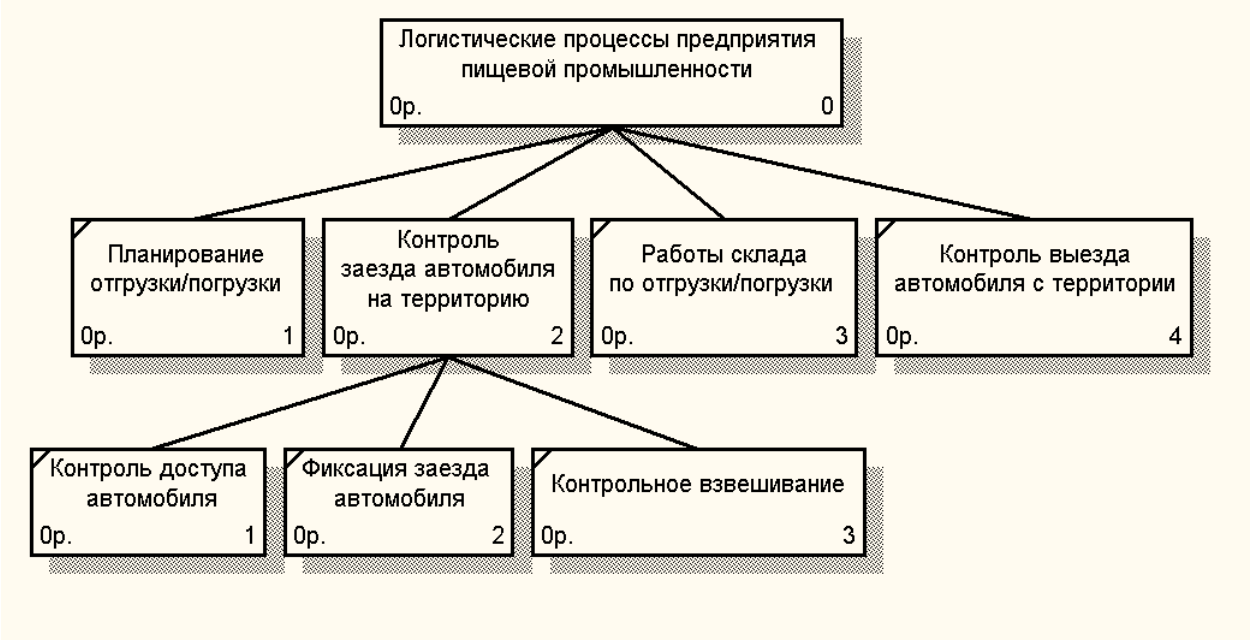


Рисунок 5 – Диаграмма дерева узлов

В нынешнем состоянии деятельность внутренних логистических процессов внутри предприятия не была автоматизирована. Внедрение новых технических средств контроля доступа автомобилей, ужесточение правил перевозки грузов по территории Крыма, запрет на парковку транспорта на

территории предприятия, сезонное увеличение отгрузки продукции предприятия потребовало изменение и автоматизации одной из функции отдела логистики.

1.3 Анализ существующих разработок для решения задачи по автоматизации логистических процессов предприятия

В настоящее время существует большое количество автоматизированных систем для внутренних логистических задач предприятия, поэтому далее, рассмотрим некоторые аналоги.

Компания High Information Technologies SIA (HIT, HITGROUP) предлагает автоматическую систему контроля пропусков автотранспорта для систем контроля доступа и видеонаблюдения. HIT ANPR предоставляет автоматический пропускной режим при минимальном задействовании сотрудников.

Алгоритм работы системы HIT ANPR:

- оператор выписывает пропуск;
- автомобиль подъезжает к шлагбауму. Видеокамера считывает номер автомобиля и сравнивает с базой данных. Шлагбаум автоматически поднимается, если данные о транспорте найдены;
- автомобиль подъезжает к шлагбауму, который автоматически поднимается, если у автомобиля есть пропуск на выезд;
- автомобиль подъезжает к подъемному барьеру, который автоматически открывается, если у автомобиля есть пропуск на выезд.

В данной системе реализованы следующие возможности:

- создание пропуска;
- принудительное открытие шлагбаума;
- ведение базы данных (поиск в базе данных);

– просмотр активных клиентов;
– контроль времени нахождения транспорта на территории;
– добавление пользователя с ограничением деятельности в системе.

База данных позволяет хранить следующие данные:

– ФИО;
– номер машины;
– персональный код;
– информация о въездах и выездах [3].

Система автоматизации проезда транспорта с распознаванием номера VIDEONET-AUTO PSIM делает контроль проезда транспорта на территорию управляемым, автоматизирует процесс въезда и выезда и исключает человеческий фактор.

Алгоритм работы системы VIDEONET-AUTO PSIM:

– автомобиль подъезжает к КПП, которое оснащено воротами или шлагбаумом;
– камера считывает номер автомобиля и ищет его в базе данных;
– на основе заданного сценария и результата поиска по базе данных, система принимает решение об открытии проезда;

Данная система решает следующие задачи:

– запрет доступа транспорту;
– контроль въезда и выезда автомобилей на территорию;
– сбор статистики и формирование отчетов;
– реализация сложных сценариев въезда;
– многофакторный режим доступа: распознанный номер, карта доступа, тип транспорта.

В базе данных описываемой системы хранятся следующие данные:

– номер транспортного средства;
– дата и время перемещений через КПП;

– направление движения [4].

Автоматизированная система КОДОС-Транспорт предназначена для организации контроля доступа и проезда автомобилей и железнодорожного транспорта через КПП с организацией заданного алгоритма проезда и сбора данных.

При применении комплекса как части общей системы «Кодос», он может взаимодействовать с другими компонентами и использовать алгоритмы организации бюро пропусков автомобилей.

Самый простой вариант применения системы предполагает следующий алгоритм работы:

– автомобиль подъезжает к КПП;
– камера считывает номер автомобиля и ищет его в базе данных если номер не найден, то он вносится в базу данных;
– если у данного транспортного средства есть разрешение на проезд, автомобиль пропускается;
– в базу данных вносятся данные о проезде.

В число данных, собираемых системой, входят:

– номерной знак;
– направление движения;
– полномочия транспортного средства;
– вес автомобиля;
– дата и время въезда или выезда.

Так же данная система предоставляет следующие возможности:

– управление правами доступа автомобилей;
– регистрация номеров автомобилей при въезде на КПП;
– занесение дополнительных данных в базу данных [5].

Сравнительная характеристика аналогов приведена в таблице 3.

Таблица 3 – Обзор аналогов

Характеристики	HIT ANPR	VIDEONET-AUTO PSIM	КОДОС-Транспорт
Аппаратная зависимость	Нет	Требуется дополнительный котроллер	Требуются определенные виды камер
Интеграция с другими системами	Только с определенным набором систем	Только с системами VideoNet	Только с системами комплекса КОДОС
Простота интерфейса	Устаревший, англоязычный	Избыточный интерфейс	Устаревший, избыточный
Используемое СУБД	Нет информации	Нет информации	FireBird 2.5
Данные, которые могут храниться в БД	ФИО, персональный код, номер машины, компания, допустимое время пребывания, информация о въездах/выездах	Номер машины, данные о въездах /выездах, тип транспортного средства	ФИО, группа доступа, номер и марка машины, данные о въездах/выездах
Составление отчетов	Нет	Да	Да
Экспорт отчетов	Нет	csv, bmp, html	html, MS Excel

На основе данных приведенной таблице можно сделать вывод о том, что никакая из описанных систем не может обеспечить выполнение всех задач, которые поставлены перед информационной системой для логистических процессов на предприятии пищевой промышленности.

Так же, немаловажен факт того, что ни один из аналогов нельзя интегрировать в стороннюю, несвязанную с ним систему, что может значительно затруднить процесс внедрения такого комплекса.

Из выше сказанного следует, что ни один из аналогов не может стать решением рассматриваемой нами проблемы и требуется разработать собственную систему.

1.4 Функциональные требования к разрабатываемой системе

Назначением системы является автоматизация логистических процессов предприятия пищевой промышленности. Планируется разработка интерфейса для работы с системой и создание БД для сбора основной информации.

Целью создания системы является повышение эффективности работы склада, соблюдение требований и регламентов перевозки, минимизация человеческого фактора и ускорении процесса работы контрольно-пропускных пунктов промышленного предприятия.

База данных создается с целью:

- обеспечения сбора и обработки информации, необходимой для формирования статистики;
- создание системы отчетности по показателям деятельности системы;
- повышения качества хранения и структуризации информации.

В результате разработки информационного обеспечения будут улучшены следующие показатели:

- время сбора и первичной обработки данных;
- время подготовки аналитической отчетности;
- пропускная способность склада и контрольно-пропускных пунктов.

В интерфейсе системы необходимо реализовать авторизацию пользователей. С целью разграничение прав доступа, пользователю должна быть присвоена одна из ролей:

- администратор;
- сотрудник охраны;
- логист.

Администратор имеет право создания новых пользователей в систему и добавления информации в базу данных.

Сотрудник охраны непосредственно следит за допуском машин в системе и при не распознании номера оперативно может ввести номер машины вручную.

Логисту доступен просмотр данных о работе КПП и формирование необходимых отчетов.

В базе данных должны храниться следующие данные:

- ФИО;
- дата выдачи прав доступа на территорию и дата их истечения – план погрузочно-разгрузочных работ склада;
- номер машины;
- марка машины;
- информация о всех въездах и выездах автомобиля.

Задачи, которые должны решаться системой:

- контроль въезда и выезда, посредством реализации поиска записей о автомобилях у которых присутствуют или отсутствуют права доступа;
- добавление записей об автомобилях в базу данных;
- добавление записей о владельцах автомобилей в базу данных;
- добавление новых пользователей в систему;
- удаление пользователей;
- прекращение доступа автомобилей на территорию;
- сбор данных о въездах и выездах с учетом данных контрольного взвешивания;
- формирование отчетов по загрузке-погрузке товаров и продукции предприятия;
- экспорт отчетов;
- изменение данных об автомобиле;
- обновление прав доступа.

Необходимо спроектировать реляционную базу данных. В качестве СУБД - MySQL. Формируемые отчеты должны экспортироваться в файлы с расширением .pdf и .xlsx.

Сама система должна быть реализована в виде веб-ресурса с использованием JavaScript-библиотеки React.js. Серверная часть должна быть написана посредством использования программной платформы Node.js.

Интерфейс системы должен быть реализован с использованием языка разметки HTML и препроцессора Sass формального языка стилей CSS. Интерфейс должен быть простым, в целях минимизации времени обучения сотрудников.

1.5 Разработка модели бизнес-процесса «как должно быть»

Для автоматизации логистического процесса предприятия пищевой промышленности предлагается внедрить информационную систему (рисунок 6).

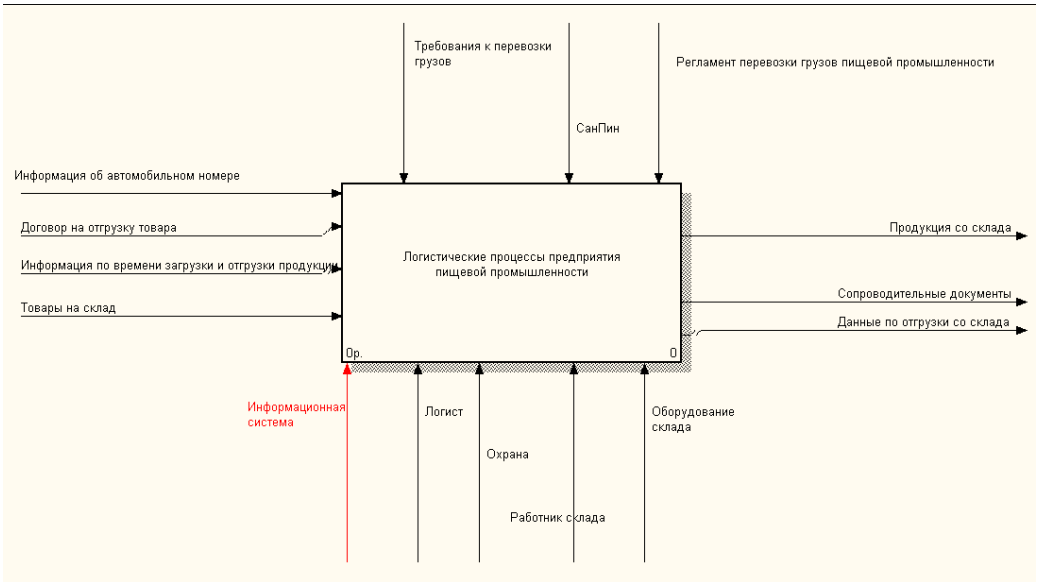


Рисунок 6 - Логистические процессы предприятия пищевой промышленности. Модель «Как должно быть» после автоматизации

После автоматизации бизнес-процесса «Логистические процессы предприятия пищевой промышленности» выглядят следующим образом (рисунок 7). Добавляется новый механизм – «информационная система», которая включает систему видеонаблюдения, датчики.

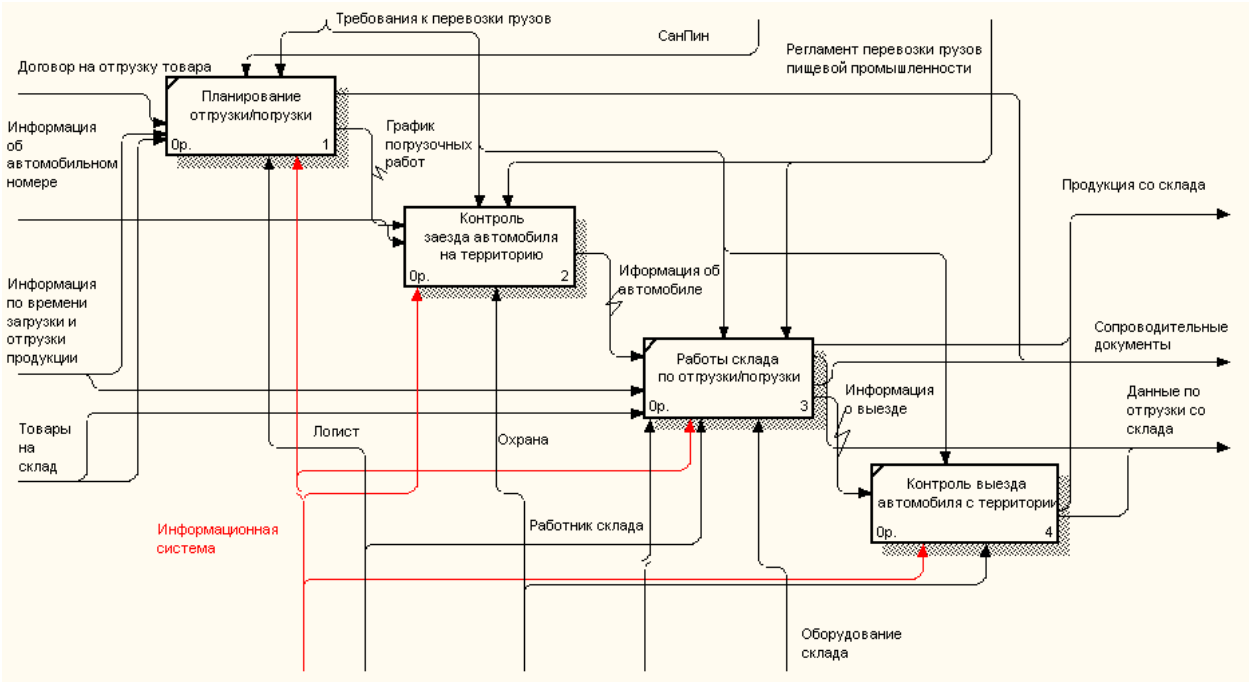


Рисунок 7 - Логистические процессы предприятия пищевой промышленности, модель «Как должно быть»

После автоматизации бизнес-процесса «Контроль заезда автомобиля на территорию» выглядит, как показано на рисунке 8. Добавляется новые блоки и действия некоторых блоков изменяются.

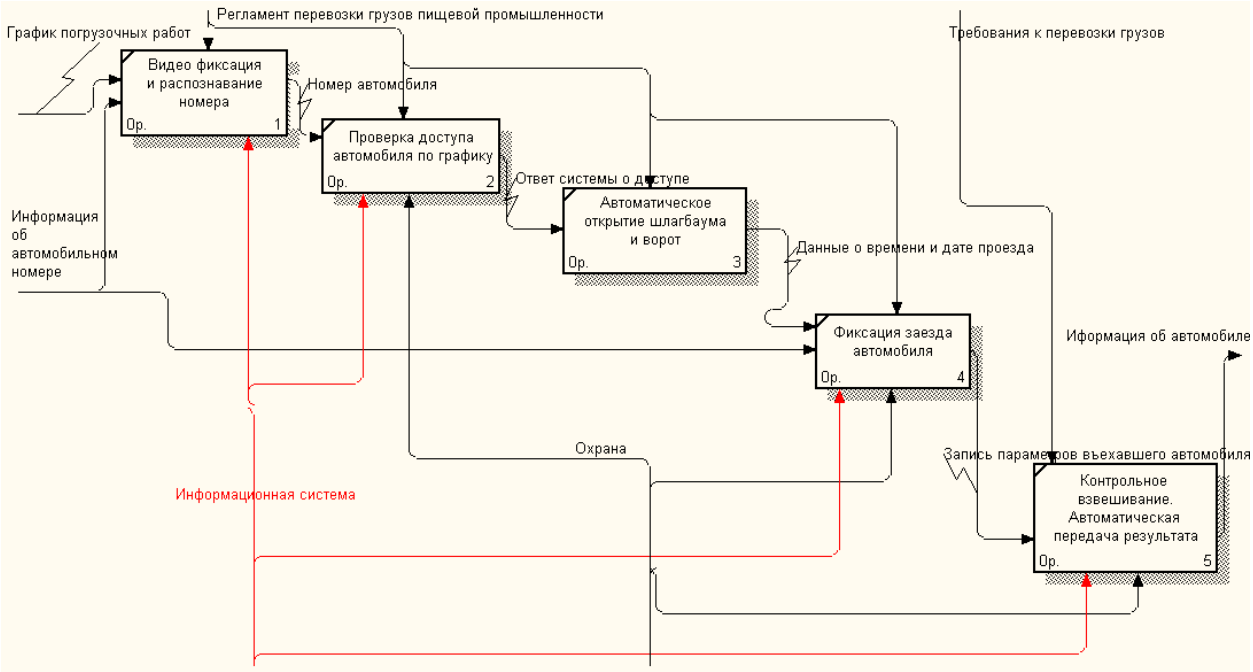


Рисунок 8 - Контроль заезда автомобиля на территорию. Модель «Как должно быть» после автоматизации

Введение данного механизма позволит на следующем уровне декомпозиции избегать ошибок и соблюсти все регламенты и требования к логистике на предприятии [13].

Выводы по главе 1

В первой главе выпускной квалификационной работы были рассмотрены общие вопросы бизнес-процессов отдела логистики компании.

Определена потребность в разработки информационной системы по контролю логистических задач на территории предприятия пищевой промышленности.

Глава 2 Логическое проектирование информационной системы

2.1 Проектирование информационной системы

Важная часть любого проектирования – этап визуального моделирования т.е. создание графической нотации для описания каких-либо аспектов разрабатываемой системы. Описанное моделирование должно быть выполнено с применением стандартной нотации.

Стандартом объектно-ориентированной нотации является унифицированный язык моделирования (англ. Unified Modeling Language). UML – язык графического описания объектного моделирования системного проектирования, функциональных процессов и представления организационных структур [6].

Унифицированный язык является открытым стандартом, который использует графические обозначения для создания абстрактных UML-диаграмм. Такие модели позволяют визуализировать представления проектируемой системы с различных точек зрения [6]. Большинство компаний и производителей программного обеспечения поддерживают данный язык. В 1997 году рабочая группа OMG объявила UML промышленным стандартом [7].

В первую очередь создается модель вариантов использования для описания функционального назначения проектируемой системы.

На основе поставленных перед системой задач, построена диаграмма вариантов использования (рисунок 9).

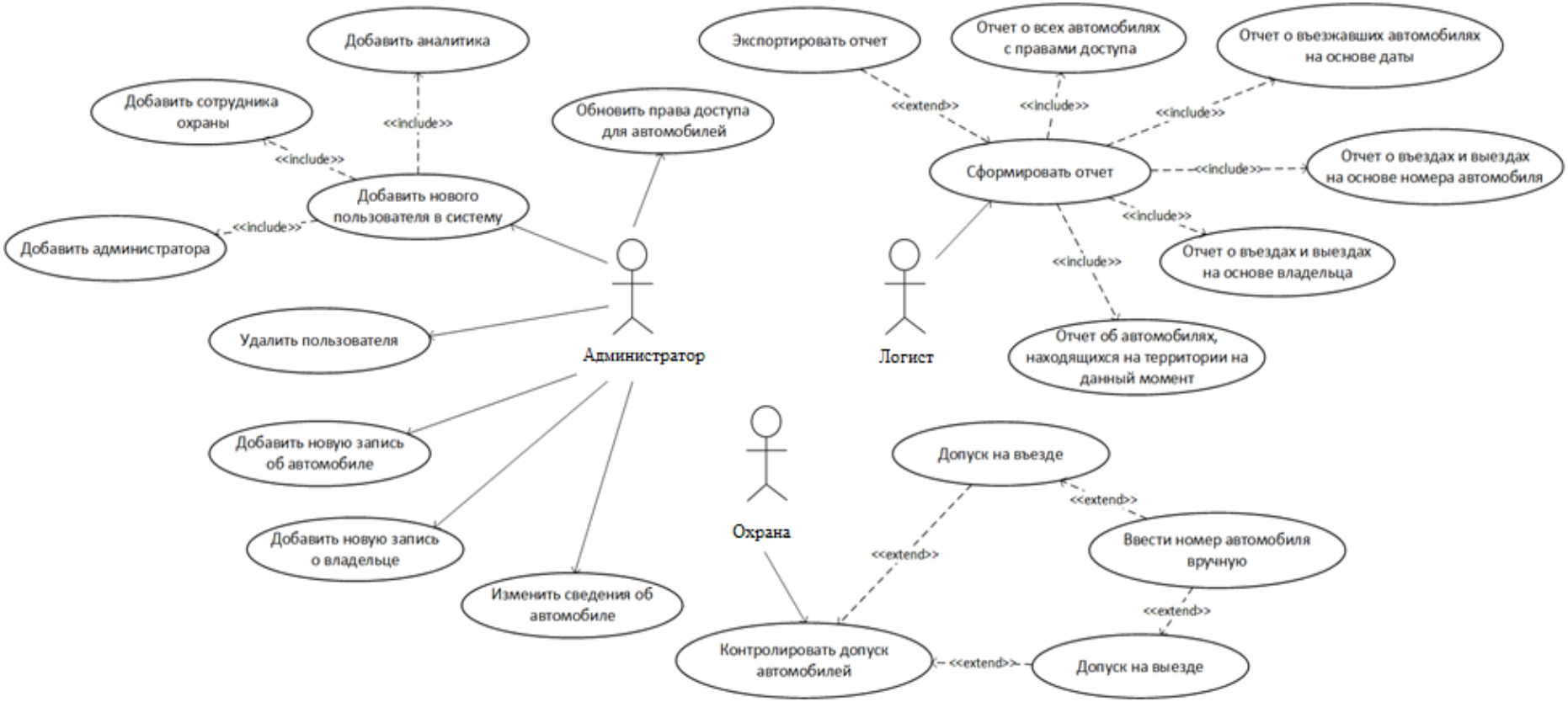


Рисунок 9 – Диаграмма вариантов использования

Целью построения данной диграммы является обеспечение донесения информации о работе системы до заказчика.

Задачи построения модели вариантов использования приведены далее:

- определение основных функций системы;
- выявление внешнего окружения, которое должно взаимодействовать с системой;
- определение нефункциональных требований.

Графические элемент «Актер» представляет внешние классы, которые предполагают взаимодействие с системой. В качестве актеров, в данном случае, выступают пользователи системы – администратор, логист, сотрудник охраны. Символ «Вариант использования» описывает какое-либо действие, которое должно быть выполнено при взаимодействии с актером.

Элементу «Администратор» должны быть доступны такие функции как, добавление новой записи в базе данных, добавление нового пользователя в систему, которое расширяется посредством добавления элементов «Добавить администратора», «Добавить сотрудника охраны». Так же администратор имеет право на удаление пользователя из системы, добавление новой записи об автомобиле, о владельце в базу данных и изменение данных об автомобиле.

«Логист» должен формировать отчеты о въездах и выездах на основе даты, номера автомобиля или данных владельца, отчет со всеми автомобилями, которые имеют право проезда на территорию и отчет об автомобилях, которые находятся на территории на данный момент. Так же должен быть возможен экспорт отчета.

Сотрудник охраны должен выполнять контроль доступа автомобилей на въезде и выезде. Если на въезде или выезде номер не распознан, то сотрудник может ввести номер вручную.

Посредством создания модели вариантов использования, было отображены основные функции разрабатываемой системы и ее взаимодействие с пользователями. Реализация каждого варианта

использования подразумевает взаимодействие некоторых объектов актеров и классов. Для его описания используются диаграммы последовательности. Они представляют собой хронологически упорядоченную серию взаимодействий объектов в процессе выполнения варианта использования. Диаграммы реализуют возможность представления последовательности в виде приема и передачи сообщений между объектами.

Для того, чтобы продемонстрировать алгоритм действий, осуществляемых администратором при выполнении основных функции, была создана диаграмма последовательности, изображенная на рисунке 10. Данная диаграмма описывает добавление новой записи в базу данных.

Элемент «Администратор» представляет собой пользователя, который взаимодействует с другими классами, реализующими вариант использования.

Для начала работы в системе администратор посылает сообщение о входе объекту «Страница входа в систему». Тот, в свою очередь создает рефлексивное сообщение о запросе ввода логина и пароля. После успешного ввода данных, осуществляется переход на главную страницу, и администратор продолжает работу.

Для добавления записи об автомобиле в базу данных, администратор переходит на соответствующую страницу, вводит всю необходимую информацию в формах и нажимает кнопку подтверждения. После осуществления проверки данных на непротиворечивость, введенные данные отправляются на сервер, где формируется запрос. Далее, осуществляется запрос к СУБД. Запись добавляется или обновляется согласно запросу. Осуществляется отправка возвращаемого значения запроса к базе данных и результат выполнения передается обратно на сервер. На клиентскую часть передается сообщение о выполненном действии, которая, в свою очередь выводит результат выполненного пользователем действия.

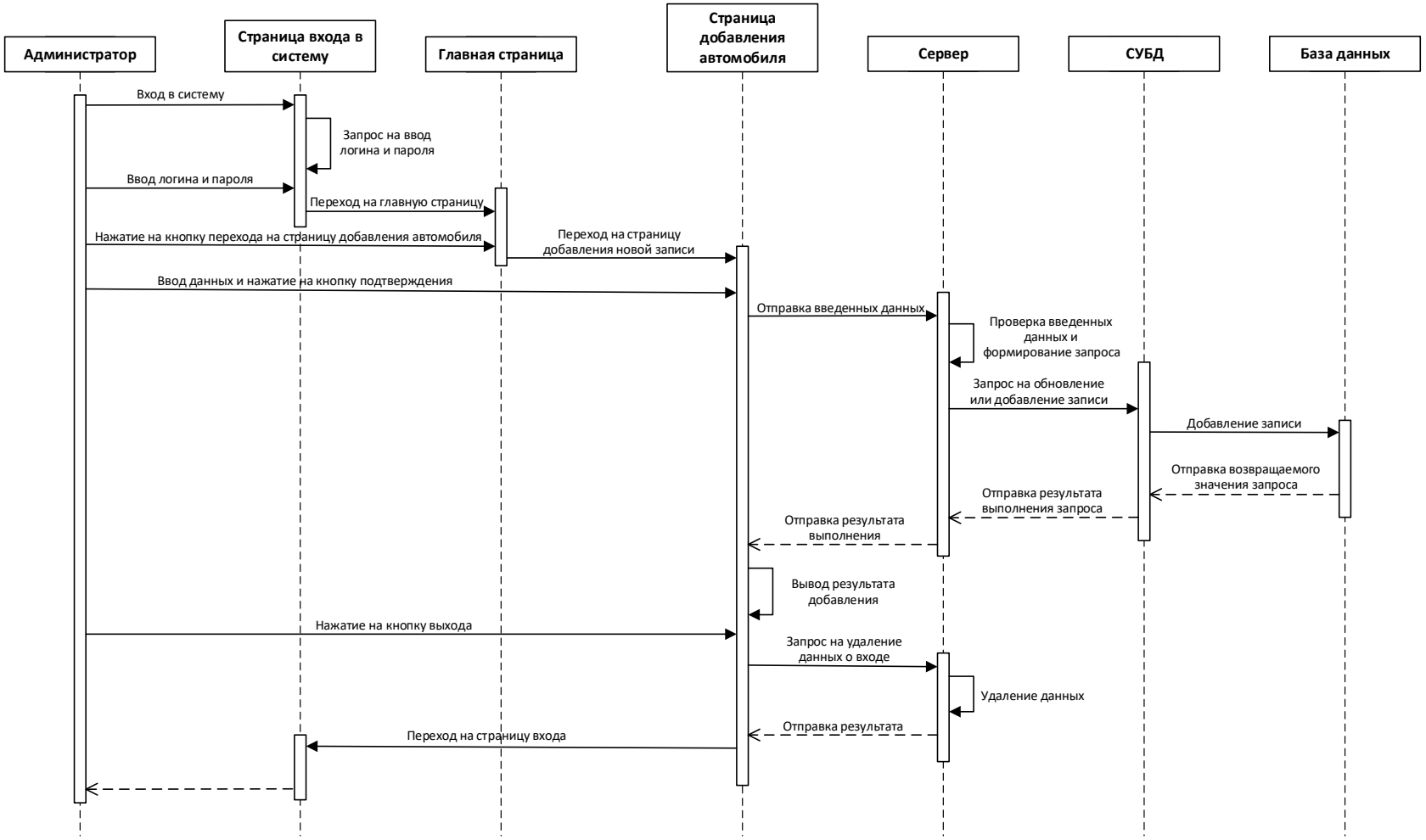


Рисунок 10 – Диаграмма последовательности. Пользователь – «Администратор»

По завершении работы, администратор может выйти из системы посредством нажатия на кнопку выхода странице. Запрос на удаление данных о входе посылается на сервер, где выполняется их удаление.

Последовательности действий, предпринимаемых при выполнении остальных функции аналогичны алгоритму, отображенному на диаграмме.

Так же была построена диаграмма, описывающая последовательность действий логиста для формирования и экспорта одного из отчетов (рисунок 11). Вход в систему и выход из нее в диаграмме последовательности логиста реализован так же, как и в диаграмме администратора.

После того как логист ввел необходимые параметры и нажал кнопку формирования отчета на одной из соответствующих страниц, данные посылаются на сервер. Сформированный на сервере запрос на выборку посылается на СУБД, которое, в свою очередь, реализует его выполнение. После отправки возвращаемого значения в СУБД, сервер получает результат выполнения запроса. После получения сообщения о выполнении действия, на странице выводится требуемый отчет.

Если логисту необходимо экспортировать отчет, то сформированная таблица записывается в файл с необходимым расширением и начинается загрузка.

В целях демонстрации процесса контроля въезда и выезда автомобилей со стороны сотрудника охраны, была создана соответствующая диаграмма последовательности, продемонстрированная на рисунке 12.

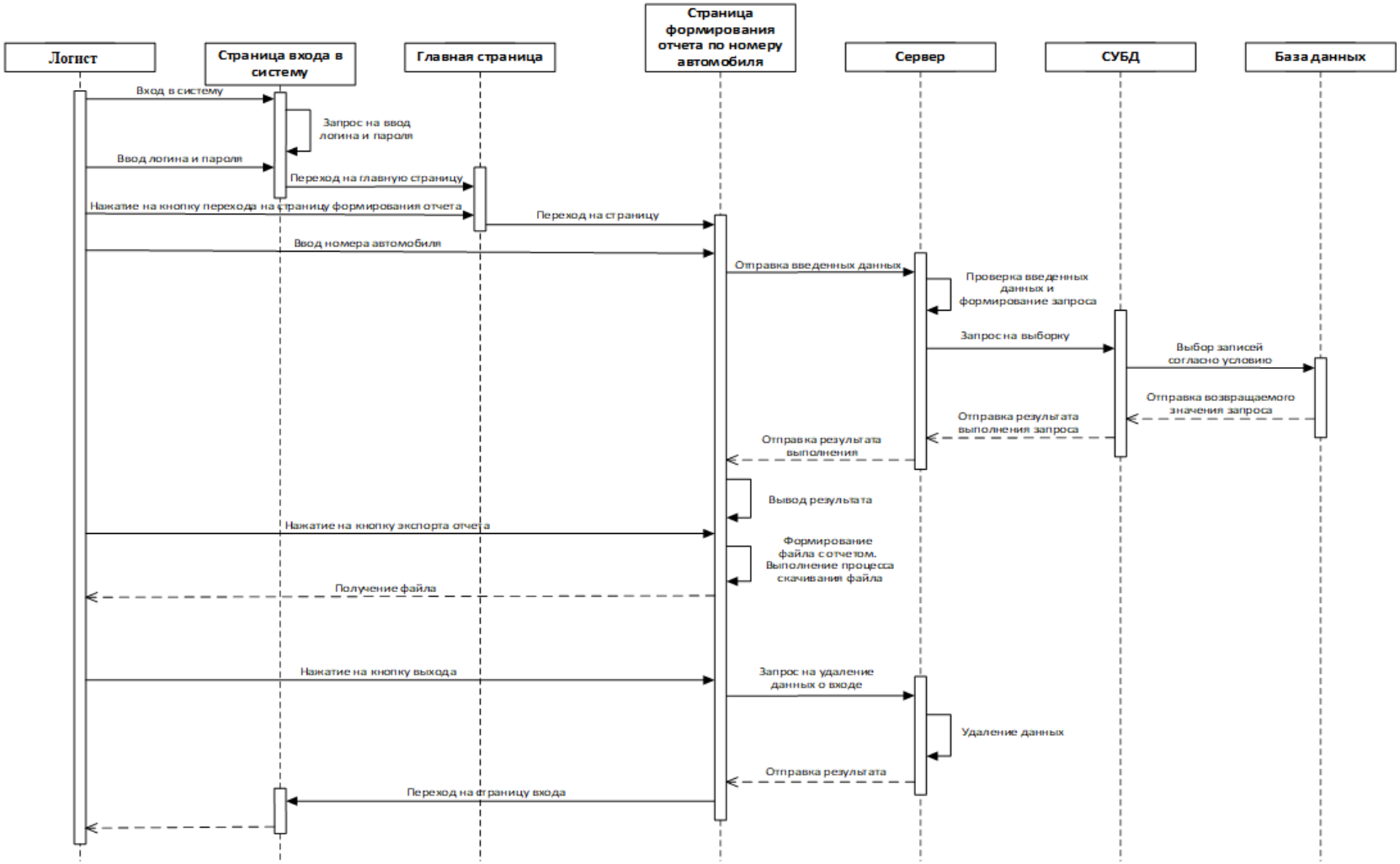


Рисунок 11 - Диаграмма последовательности. Пользователь – «Логист»

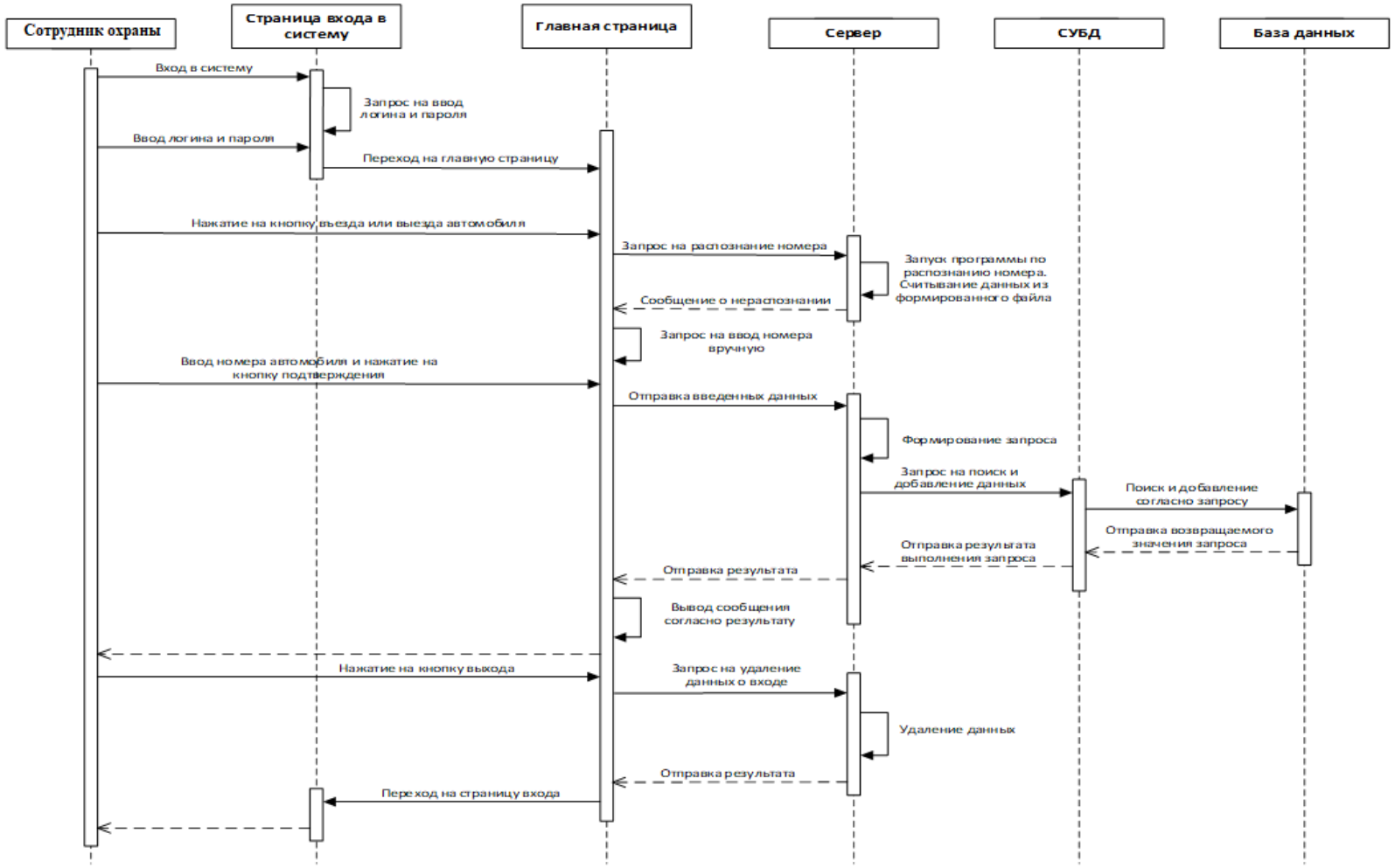


Рисунок 12 – Диаграмма последовательности. Пользователь – «Сотрудник охраны»

Авторизация сотрудника охраны и выход из системы осуществляется аналогично учетной записи администратора и логиста.

Так как процессы пропуска автомобиля в оба направления осуществляются аналогично, они объединены в одну диаграмму.

По нажатии на кнопку въезда (выезда) автомобиля, на сервер посылается запрос на распознание номера. Сервер вызывает выполнение программы по распознанию номера. Считываются данные из формированного файла. Так как в данном случае изображен процесс не распознания номера, сервер получает код об ошибке. Результат возвращается клиентской части, которая в свою очередь запрашивает ручной ввод номера. Сотрудник вводит номер и нажимает кнопку подтверждения.

На сервере формируется запрос на поиск нужной записи и дальнее добавление записи о въезде (выезде). Далее, выполняется сам запрос и отправляется возвращаемое значение. На основе результата запроса, на экране выводится соответствующее сообщение.

Благодаря построению диаграмм последовательности действий, хронология выполнения задач системы и взаимодействие пользователей системы определены более явно.

С помощью диаграммы взаимодействия, графически отображено разграничение возможностей и прав пользователей системы по ролям, а также требования, предъявляемые к системе.

2.2 Проектирование базы данных

Основными задачами проектируемой системы являются сбор информации о въездах и выездах, оперативное обращение к данным, добавление новых записей и их непротиворечивая модификация. Следовательно, главной составляющей разрабатываемого информационного обеспечения является база данных.

Внедряемая база данных позволит хранить всю необходимую информацию и взаимодействовать с ней. Находящиеся в ней данные должны характеризовать состав объектов предметной области, а также их свойства и взаимодействия. Таким образом, база данных является отражением предметной области в виде структурированного набора данных.

База данных содержит сведения о следующих объектах:

- информация об автомобильных номерах и автомобили,
- пропускные пункты,
- разрешенные к въезду пропускные пункты предприятия для каждого автомобиля,
- дни,
- дни въезда и выезда автомобилей для разгрузочных и погрузочных работ.

Для графического представления таблиц, сущностей и связей используется методология IDEF1. IDEF1X является новой, усовершенствованной методологией. Она отличается реализацией возможности автоматизации и простотой изучения [9].

На основе анализа предметной области и возможных запросов пользователей была разработана инфологическая модель базы данных в виде логического уровня диаграммы IDEF1X (рисунок 13).

Идентифицирующие отношение изображено на рисунке сплошной линией между прямоугольниками связанных сущностей, неидентифицирующие отношение показано на рисунке 13 пунктирной линией.

Неидентифицирующие отношение между сущностями машина и водитель, т.к. управлять автомобилем может и не закрепленной за ней водитель.

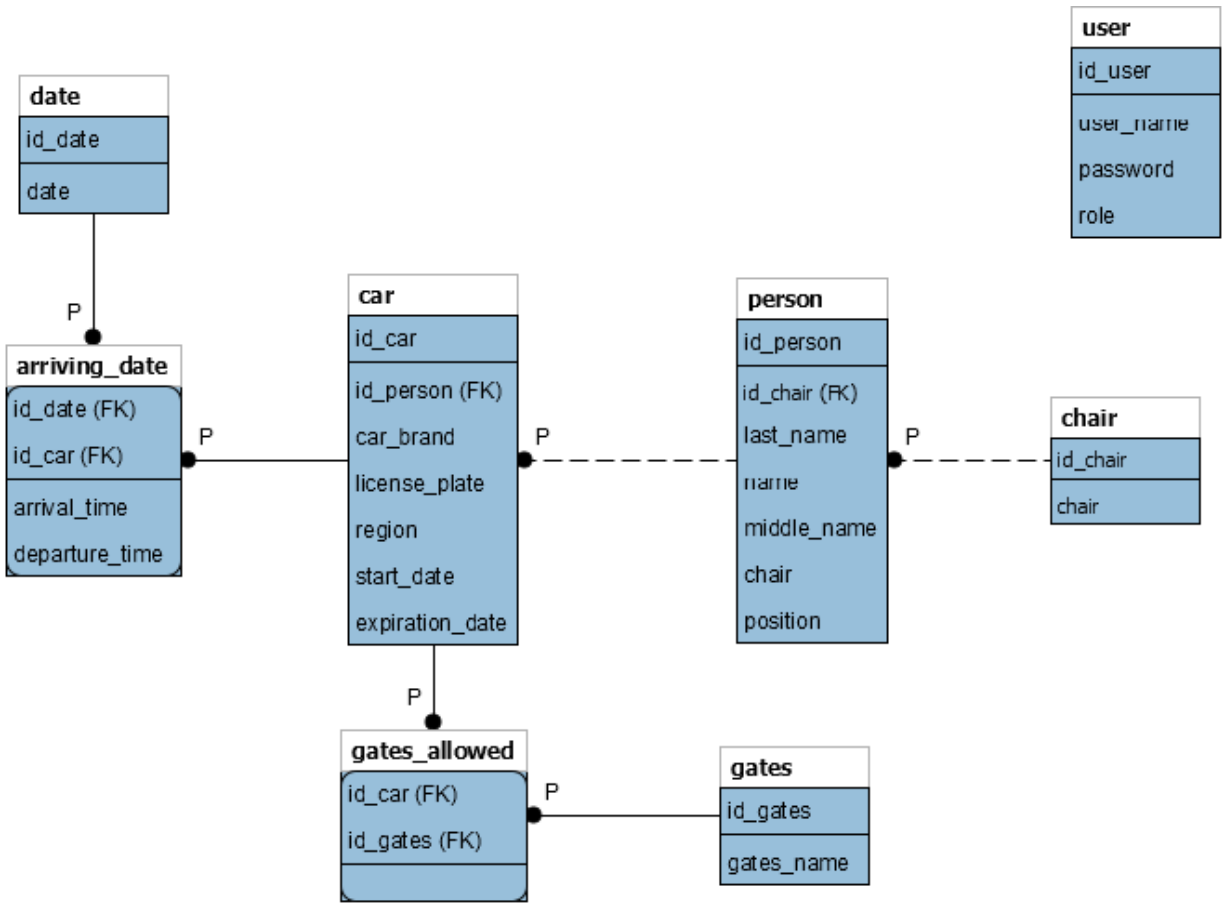


Рисунок 13 – Инфологическая модель данных

Название таблиц и соответствующие ей атрибуты представлены в таблице 4.

Таблица 4– Сущности и их атрибуты

Название таблицы	Описание	Атрибуты	Первичный ключ
person	id водителя фамилия имя отчество id подразделения должность	id_person last_name name middle_name id_chair position	id_person
car	id автомобиля id водителя марка автомобиля номерной знак регион дата предоставления доступа дата приостановки доступа	id_car id_person car_brand license_plate region start_date expiration_date	id_car

Продолжение таблицы 4

Название таблицы	Описание	Атрибуты	Первичный ключ
date	id даты дата	id_date date	id_date
arriving_date	id даты id автомобиля время въезда время выезда	id_date id_car arrival_time departure_time	id_date id_car
gates	id пропускного пункта название пропускного пункта	id_gates gates_name	id_gates
gates_allowed	id автомобиля id разрешенного пропускного пункта	id_car id_gates	id_car id_gates
chair	id подразделения подразделения	id_chair chair	id_chair
user	id пользователя логин пароль роль пользователя	id_user user_name password role	id_user

Связи между таблицами, которые присутствуют в базе данных:

person - car – неидентифицирующая связь 1 : n;

date - arriving_date – идентифицирующая связь 1: n;

car - arriving_date – идентифицирующая связь 1 : n;

gates - gates_allowed – идентифицирующая связь 1 : n;

chair - person – неидентифицирующая связь 1 : n;

car - gates_allowed – идентифицирующая связь 1 : n.

Соответствие нормальным формам:

Таблица car

- id_person (таблица person) → id_person (внешний ключ);
- id_car → car_brand;
- id_car → license_plate;
- id_car → region;
- id_car → start_date;
- id_car → expiration_date.

Таблица person

– id_chair (таблица chair) → id_person (внешний ключ);
– id_person → last_name;
– id_person → name;
– id_person → middle_name;
– id_person → chair;
– id_person → position.

Таблица date

– id_date → date.

Таблица arriving_date

– id_date (таблица date) → id_date (составной первичный ключ);
– id_car (таблица car) → id_car (составной первичный ключ).

Таблица gates

– id_gates → gates_name.

Таблица gates_allowed

– id_car (таблица car) → id_car (составной первичный ключ);
– id_gates (таблица gates) → id_gates (составной первичный ключ).

Таблица chair

– id_chair → chair;

Таблица user

– id_user → user_name;
– id_user → password;
– id_user → role.

Все приведенные отношения соответствуют третьей нормальной форме т.к. они соответствуют второй нормальной форме и каждый неключевой атрибут зависит только от первичного ключа.

Построение инфологической модели помогло детально определить структуру проектируемой базы, описать сущности и отношения между ними.

Для более полного понимания способа хранения информации необходимо создать физическую модель базу данных.

В качестве системы управления базой данных использовано MySQL. С целью администрирования СУБД было выбрано веб-приложение phpMyAdmin.

С помощью встроенных инструментов phpMyAdmin, создана физическая модель базы данных, представленная на рисунке 14.

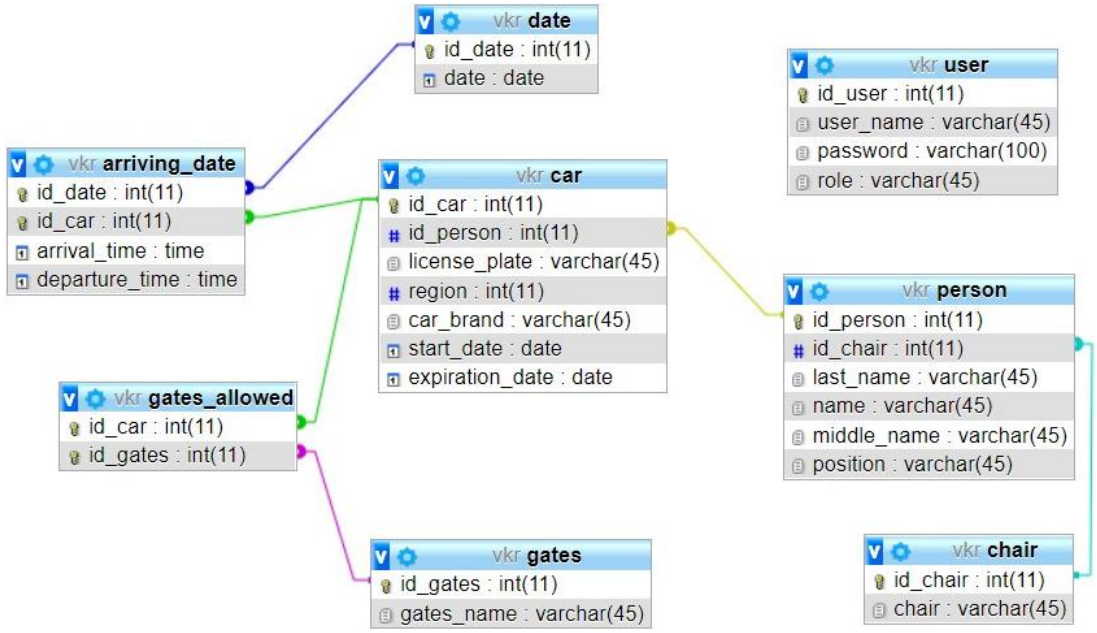


Рисунок 14 – Физическая модель данных

После окончания проектирования базы данных разрабатывается пользовательское приложение для взаимодействия со всеми таблицами.

Выводы по главе 2

Во второй главе рассмотрены вопросы логического проектирования системы для автоматизации логистических процессов предприятия пищевой промышленности.

Приведена диаграмма вариантов использования и диаграммы последовательностей для каждой роли пользователя (администратор, логист и сотрудник охраны). Определена информационная модель, где отражены входные и выходные документы. На физической схеме показаны сущности базы данных и атрибуты.

Глава 3 Реализация проекта разработки информационной системы и оценка его эффективности

3.1 Описание используемых технологий

Информационное обеспечение реализовано с сопутствующим программным обеспечение в виде веб-ресурса. Это обусловлено необходимостью в беспрепятственном взаимодействии с базой данных.

Разработана структурная схема (рисунок 15) разрабатываемого программного обеспечения, которая позволяет графически изобразить связь между подсистемами.

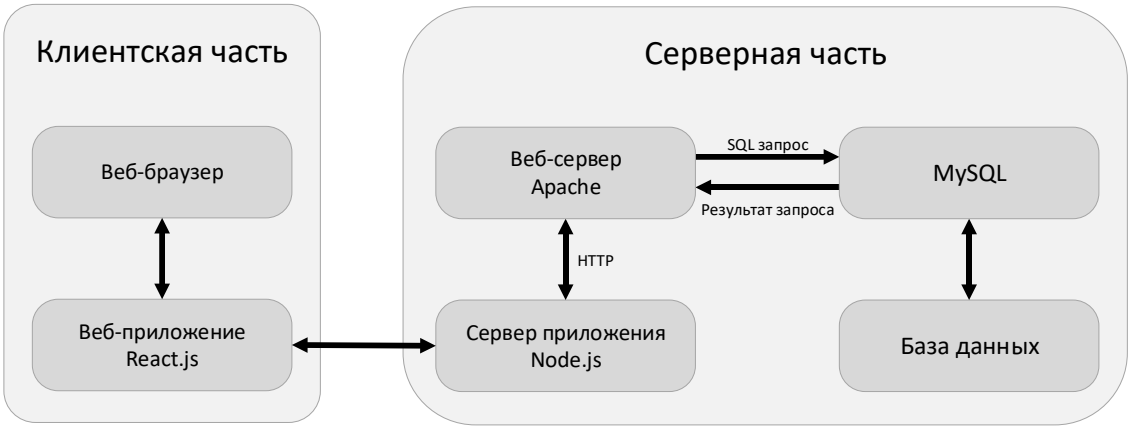


Рисунок 15 – Структурная схема

Одной из главных составляющих веб-системы является клиентская часть, которая реализована с помощью фреймворка React.js. Фреймворк позволяет разрабатывать гибкие, многомодульные и быстродействующие системы.

Одним из главных преимуществ данного фреймворка является возможность реализации одностраничных сайтов (Single-page application). Идея SPA технологии заключается в том, что все необходимые компоненты сайта загружаются в самом начале, в одну сессию. При переходе на другую страницу, сайт подгружает запрашиваемый компонент, без необходимости

загрузки другой HTML страницы. Это обеспечивает высокую скорость переходов между страницами без перезагрузки самого сайта.

Такие компоненты в React.js реализуются путем использования JSX – расширения языка JavaScript. Он позволяет более гибко управлять HTML элементами, используя JavaScript. JSX компонент представляет из себя объект, который формируется посредством вызова React.createElement() [10].

В приложении приведен пример JSX компонента, который возвращает модальное окно.

У данного компонента, как и у JavaScript функций есть параметры, в зависимости от значений которых, возвращается соответствующий компонент. Стоит отметить, что JSX компоненты, вызываются в том же виде, что размечаются HTML тэги.

Еще одно из немаловажных преимуществ React.js это использование виртуальной объектной модели документа (Virtual DOM). Классическая модель определяет древовидную структуру HTML страницу. Браузер непрерывно следит за любыми изменениями в, каждый раз ее обновляя. Это сильно сказывается на производительности системы, так как обновление может занимать много времени.

React решает данную проблему, посредством обновления копии объектной модели документа – Virtual DOM. После сравнения копий, React обновляет только изменившуюся часть реального DOM.

Простота и удобство интерфейса являются важными аспектами любого приложения. От этого зависит скорость обучения пользователя и эффективность дальнейшей работы [11]. Интерфейс должен быть понятен и не избыточен.

Интерфейс системы разработан с использованием языка стилей CSS. В разрабатываемом приложении используется препроцессор Sass. Его преимуществом является упрощение написание стилей, что повышает скорость их написания и чистоту кода. Так же немаловажным преимуществом является удобство использования медиа запросов. Результат работы

программы показан на диаграмме компонентов рисунок 16. Пример медиа запроса приведен в приложении Б.

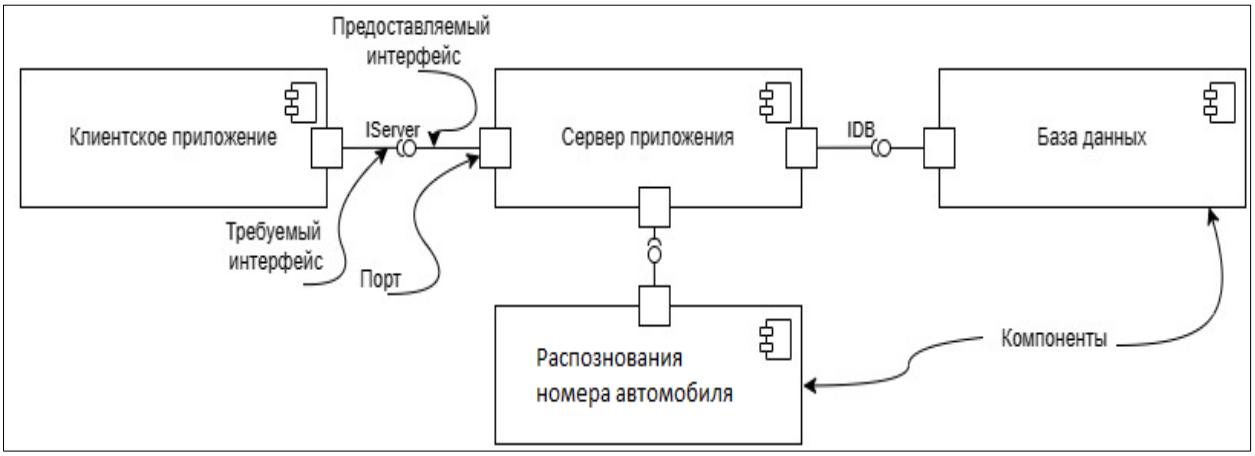


Рисунок 16 – Диаграмма компонентов

Пример реализации адаптивной верстки приведен на рисунке 17, где изображена главная страница администратора на экране с соотношением сторон 16:9 и разрешением 1920x1080.

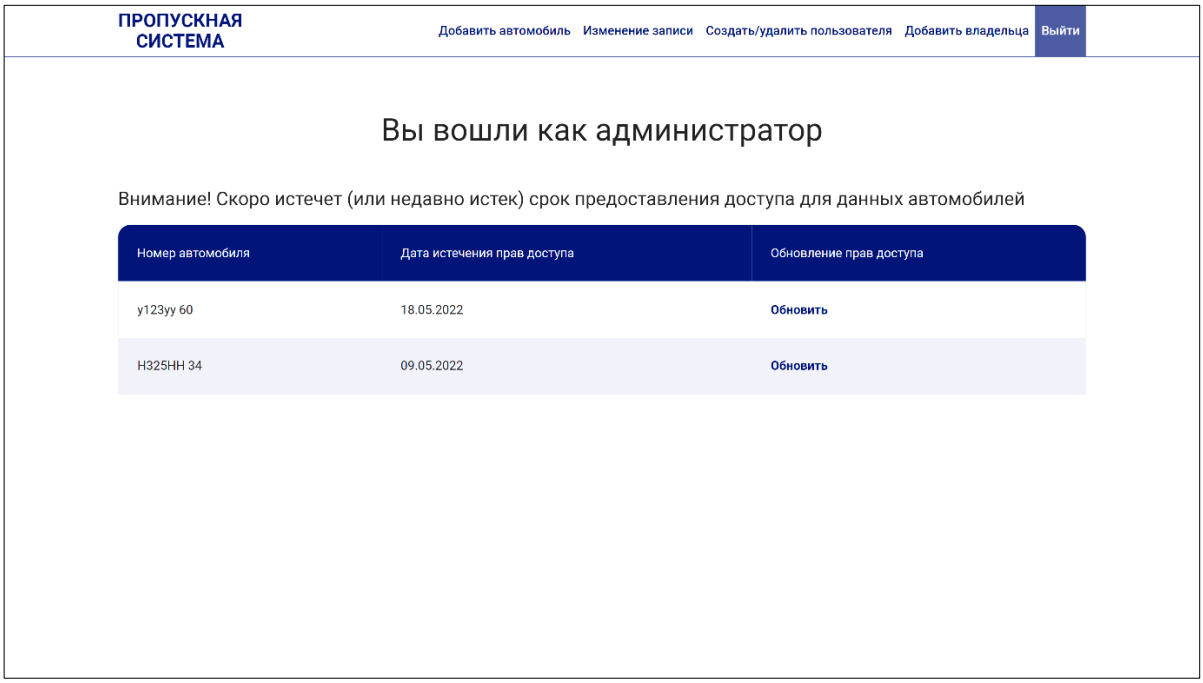


Рисунок 17 – Интерфейс страницы на десктопе

В скобках медиа запроса указывается минимальное, максимальное или промежуточное разрешение, при котором указанное свойство стиля используется. В данном случае, меняется размер шрифта текста. Использование медиа запросов необходимо для написания адаптивной верстки, идея которой заключается в подстраивании интерфейса веб-страницы под разрешение экрана пользователя.

Далее на рисунке 18 приведен пример интерфейса той же самой страницы, но для экрана мобильного устройства с разрешением 375x667.

Рисунок 18 – Страница входа на экране мобильного устройства

Меню на экране мобильного устройства реализовано в виде «бургер-меню», которое открывается по нажатии на соответствующую иконку (рисунок 19).

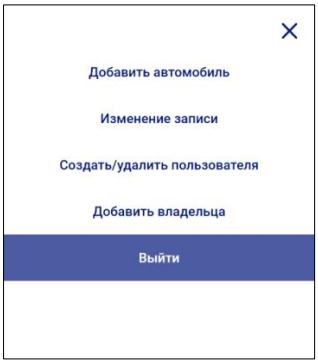


Рисунок 19 – Меню на экране мобильного устройства

Адаптивная верстка необходима для того, чтобы реализовать возможность использования системы без написания отдельного программного модуля для мобильных устройств.

Интерпретатором веб-системы выступает один из современных браузеров. Так как в разрабатываемой системе используется современный стандарт JavaScript, некоторые браузеры могут его не понимать. Для решения данной проблемы, задействован транскомпилятор Babel, преобразующий современный JavaScript в его упрощенную версию, которую устаревший браузер сможет интерпретировать.

Серверная часть написана на JavaScript и реализована с использованием программной платформы Node.js. С помощью нее JavaScript может использоваться, не только как сценарный язык веб-браузера, но и как инструмент полноценной разработки сервера приложений.

Зачастую, данную среду выполнения используют для разработки систем, обрабатывающих большие объемы данных в реальном времени. Node.js использует механизм событий, поэтому обладает высоким уровнем масштабируемости и позволяет реализовывать асинхронные запросы [12].

Для реализации HTTP запросов, таких как GET и POST, используется веб-сервер Apache. Он выступает в виде системы виртуальных хостов и отвечает за обслуживание размещенных на нем сайтов. В данном случае, Apache необходим для реализации взаимодействия с СУБД.

3.2 Описание работы информационной системы по автоматизации логистических процессов предприятия

Работа системы во время выполнения процесса пропуска автомобилей представлена на рисунке 20.

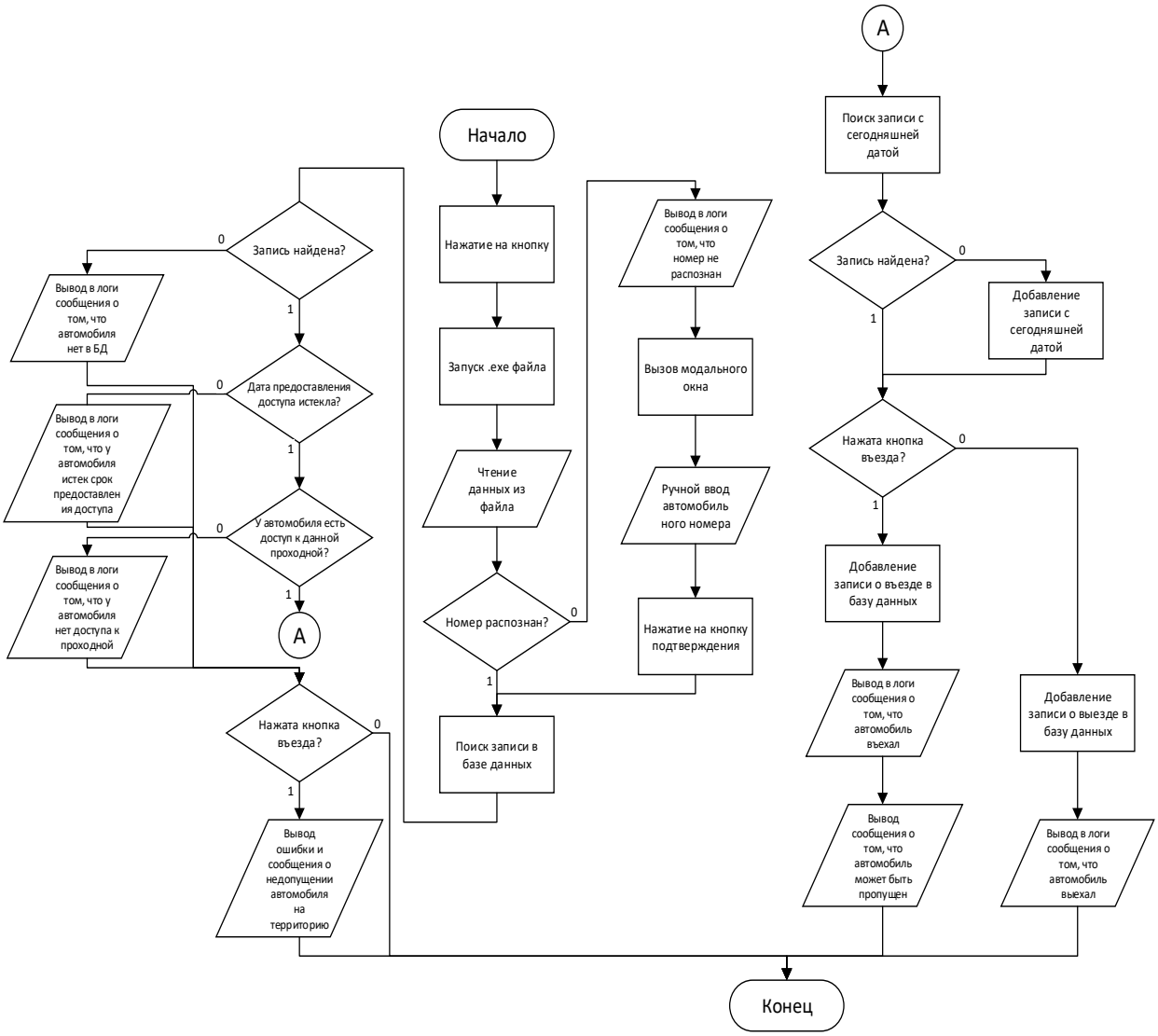


Рисунок 20 – Блок-схема алгоритма

После того как машина подъехала к КПП, сотрудник охраны нажимает на кнопку «Автомобиль въезжает» или «Автомобиль выезжает» (рисунок 21) вызывается выполнение исполняемого .exe файла.

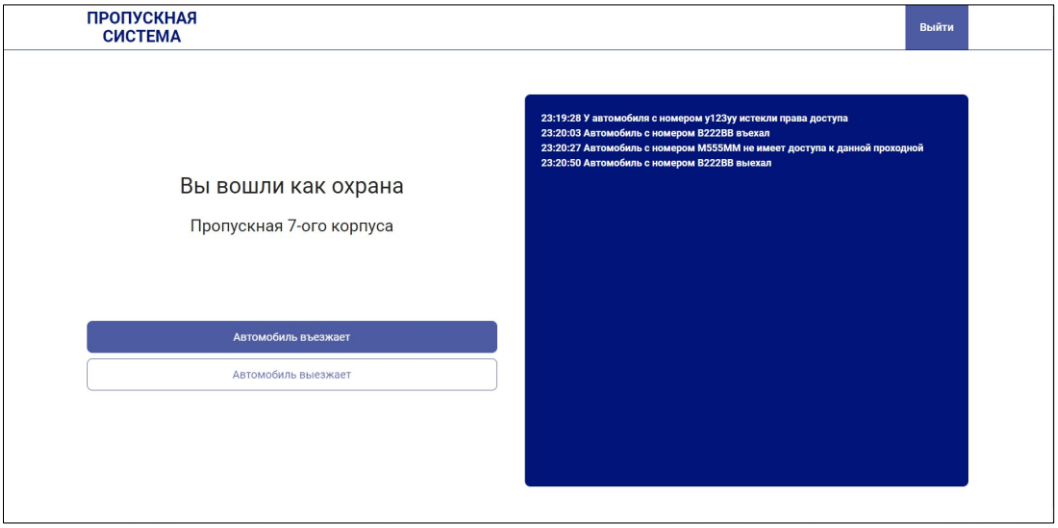


Рисунок 21 – Интерфейс сотрудника охраны

На вход подается сформированный .txt файл, из которого считываются данные в виде номера автомобиля и региона (рисунок 22).

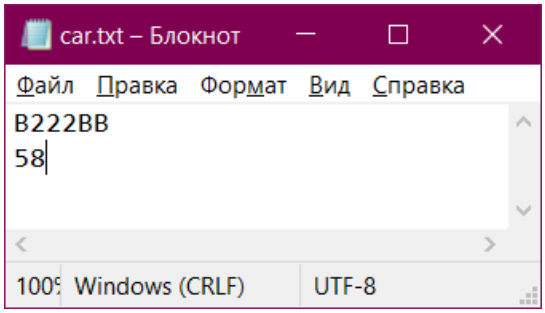


Рисунок 22 – Распознанный номер во входном файле

Эти данные могут быть представлены в виде кода ошибки (рисунок 23), если номер не распознан.

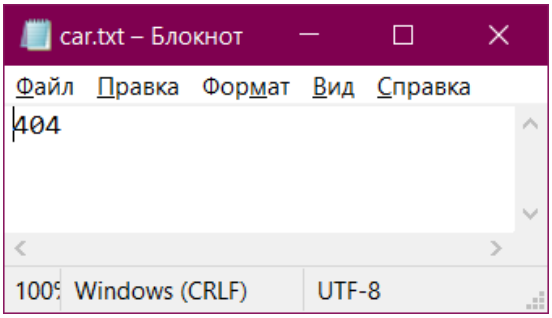


Рисунок 23 – Код ошибки во входном файле

Если считанные данные содержат код ошибки, то в логи (журнал записей) выводится сообщение о не распознании и в интерфейсе сотрудника охраны открывается модальное окно для ручного ввода номера (рисунок 24).

Хоть и предполагается, что если автомобиль выезжает, то у него есть права доступа, номер все равно необходимо ввести для ведения более точной статистики. Элемент ввода имеет систему подсказок в виде списка имеющихся номеров, который сужается во время ввода сотрудником необходимого номера.

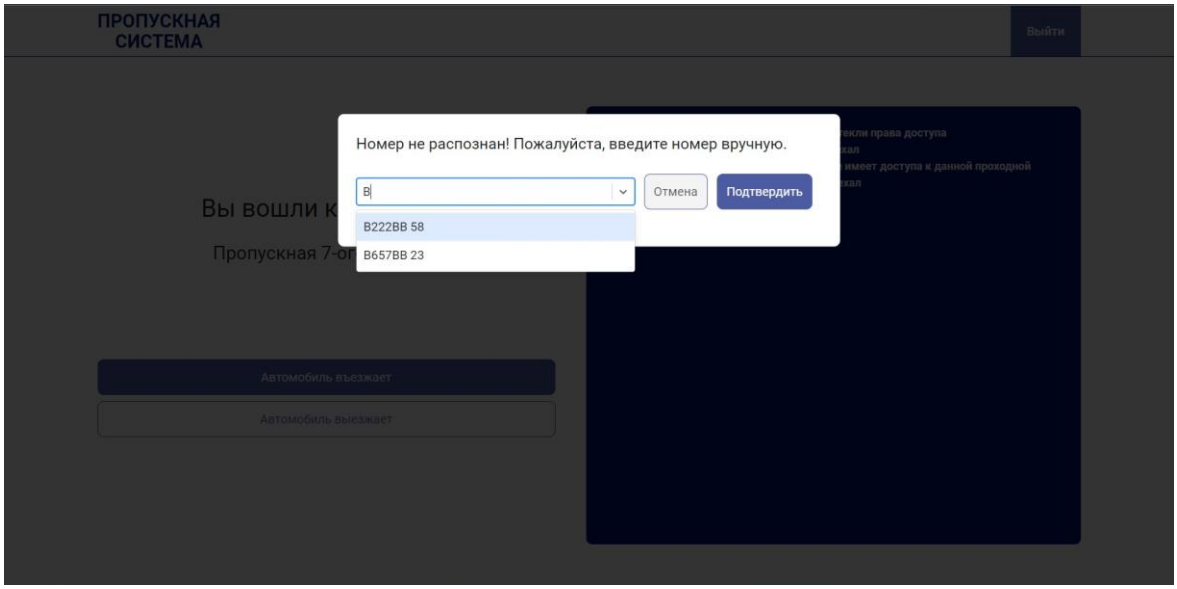


Рисунок 24 – Модальное окно ввода номера автомобиля

После ввода данных и нажатия на кнопку подтверждения возможны следующие варианты работы программы, первое - номер не был распознан, второе - номер успешно распознан и тогда осуществляется поиск номера в базе данных с помощью оператора SELECT. Если запись с таким номером в базе данных не найдена, выводится соответствующее сообщение в лог-файл программы (рисунок 25). Если запись в базе данных с таким номером найдена, то далее проверяется - не истекли ли у автомобиля права доступа на территорию предприятия. Если обнаружено нарушение в доступе автомобиля, то выводится сообщение в лог-файл и пишется предупреждение.

Рисунок 25 – Сообщения в лог-файл

Далее, если у автомобиля есть доступ к данной проходной, выводится сообщение о том, что въезжающий автомобиль может быть пропущен (рисунок 26). Данные в въезде или выезде автомобиля с правами доступа вносятся в базу данных. В данном случае, как пример, взято одно из КПП поэтому вся работа, ведущаяся в данной системе, выполняется для одного из них.

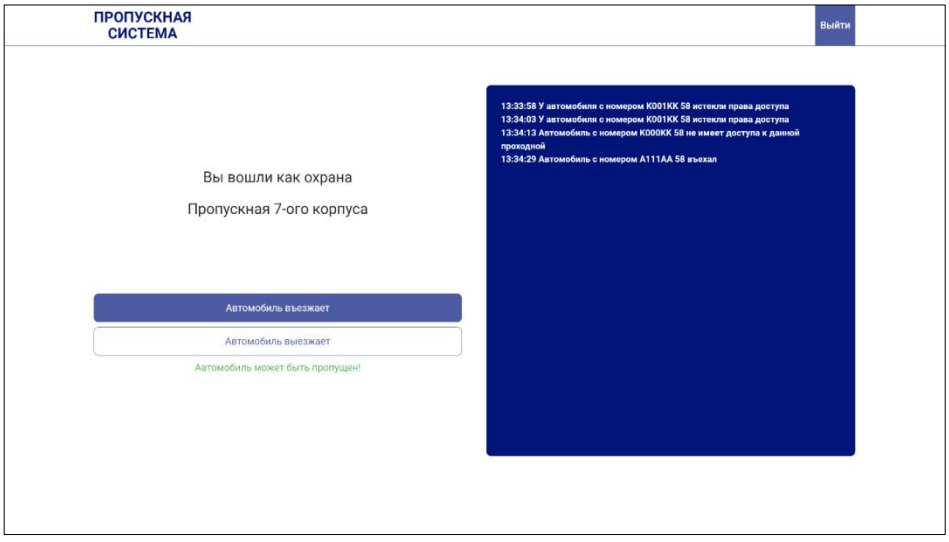


Рисунок 26 – Сообщение о пропуске автомобиля

Все сообщения, выводимые в логи, так же хранятся и в cookies, которые создают временные текстовые файлы о работе пользователя в браузере. Они сохраняют какие-либо данные на стороне клиента в течение какого-то времени (в данном случае cookies хранятся в течение суток) и позволяют восстановить состояние предыдущей сессии. Поэтому сообщения из логи будут храниться даже в случае закрытия веб-страницы или выхода из учетной записи.

В данной работе cookies реализованы посредством использования Node.js библиотек: express-session, cookie-parser, body-parser. Express-session необходим для непосредственного создания cookies, cookie-parser разбирает получаемые файлы cookie. Body-parser извлекает и анализирует тело HTTP запроса, он необходим для корректной обработки передаваемых данных [12].

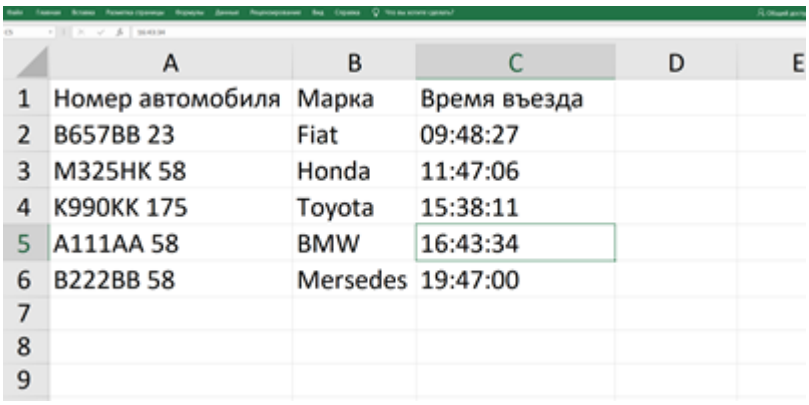
Одним из главных компонентов разрабатываемого программного модуля является система авторизации, которая позволит разграничить обязанности сотрудников и минимизировать вероятность несанкционированного доступа в систему.

Полученный результат работы системы – отчеты в виде файлов, со сформированной, отсортированной таблицей (рисунки 27 и 28).

Автомобили на территории на 10.10.2023 22:59
Количество автомобилей на территории- 5

Номер автомобиля	Марка	Время въезда
B657BB 23	Fiat	09:48:27
M325HK 58	Honda	11:47:06
K990KK 175	Toyota	15:38:11
A111AA 58	BMW	16:43:34
B222BB 58	Mersedes	19:47:00

Рисунок 27 – PDF-файл с отчетом с автомобилями на территории



	A	B	C	D	E
1	Номер автомобиля	Марка	Время въезда		
2	B657BB 23	Fiat	09:48:27		
3	M325HK 58	Honda	11:47:06		
4	K990KK 175	Toyota	15:38:11		
5	A111AA 58	BMW	16:43:34		
6	B222BB 58	Mersedes	19:47:00		
7					
8					
9					

Рисунок 28 – Excel-файл с отчетом с автомобилями на территории

Возможность формирования отчета въездах и выездах, но основе владельца, сортировка в таблице и экспорт отчета.

3.3 Оценка и обоснование экономической эффективности внедрения информационной системы

При появлении на предприятии необходимости автоматизации каких–либо бизнес–процессов, проектирования и внедрения какой–либо системы, возникает потребность в расчете экономической эффективности планируемого проекта. Внедрение нового программного продукта требует немалых финансовых вложений. По этой причине перед началом реализации проекта необходимо рассчитать затраты на его проектирование, программирование отладку и внедрение. А также проанализировать экономическую эффективность проекта.

Экономическая эффективность – это сравнение существующего технологического процесса и внедряемого. Другими словами, базового и проектируемого варианта решения задачи.

Обоснование экономической эффективности позволяет увидеть, насколько вложения в разработку новых решений или доработку уже имеющихся являются полезными.

Экономическая эффективность строится из двух основных составляющих, косвенного и прямого эффекта. Прямой экономический эффект, еще называемый сравнительным анализом, оценивается денежным выражением и, как правило, характеризуется понижением затрат на обработку информации, уменьшением трудовых затрат и увеличением производительности [11].

Далее будут рассмотрены показатели, относящиеся к трудовым:

«Абсолютное снижение трудовых затрат (ΔТ) в часах за год:

$$\Delta T = T_0 - T_1 \quad (1)$$

где:

T_0 – трудовые затраты на обработку информации по базовому (до автоматизации) варианту в часах за год;

T_1 – трудовые затраты на обработку информации по разрабатываемому варианту в часах за год» [3].

«Коэффициент относительного снижения трудовых затрат, иными словами, повышение производительности труда ($К_т$) в %:

$$К_т = \left(\frac{\Delta T}{T_0}\right) \times 100\% \quad (2)$$

Индекс снижения трудовых затрат ($Y_т$):

$$Y_т = \frac{T_0}{T_1} \quad (3)$$

Коэффициенты $К_т$ и $Y_т$ характеризуют повышение производительности труда за счет внедрения более экономичного варианта проектных решений» [4].

К стоимостным показателям относят следующие:

«Абсолютное снижение стоимостных затрат (ΔС) в рублях за год:

$$\Delta C = C_0 - C_1 \quad (4)$$

где:

C_0 – стоимостные затраты на обработку информации по базовому (до автоматизации) варианту в рублях за год,

C_1 – стоимостные затраты на обработку информации по проектируемому варианту в рублях за год» [5].

«Коэффициент относительного снижения стоимостных затрат ($К_С$) в %:

$$К_С = \left(\frac{\Delta C}{C_0}\right) \times 100\% \quad (5)$$

Индекс снижения стоимостных затрат (Y_c):

$$Y_c = \frac{C_0}{C_1} \quad (6)$$

Для проведения анализа эффективности проекта используются обобщающие и частные показатели.

К основным обобщающим показателям экономической эффективности относятся:

– годовой экономический эффект в рублях;

– расчетный коэффициент эффективности капитальных вложений;

– срок окупаемости системы» [6].

«Годовой экономический эффект от внедрения проекта (Э) (в рублях в год) определяется, как разность между годовой экономией и нормативной прибылью и рассчитывается по представленной формуле:

$$Э = \Delta C - К_п \times Е_н \quad (7)$$

где:

$К_п$ – единовременные затраты или капитальные вложения, тыс. руб.,

$Е_н$нормативный коэффициент эффективности капитальных вложений.

К капитальным затратам ($К_п$) относятся затраты на проектирование, программирование комплекса задач, а также затраты на отладку и внедрение программ» [7].

Осуществим расчет показателей экономической эффективности проекта автоматизации логистических процессов предприятия пищевой промышленности. Характеристика затрат по базовому варианту по автоматизации логистических процессов предприятия пищевой промышленности на обработку по базовому плану составили 3141000 рублей.

После внедрения информационной системы, автоматизирующей логистические процессы предприятия, в процесс допуска автомобилей на территорию предприятия и на составления плана погрузочно-разгрузочных работ будут внесены изменения. Например, появится возможность формирования отчетности в различных форматах.

Характеристика затрат по проектному варианту по автоматизации логистических процессов предприятия пищевой промышленности составили 2379000 рублей.

Рассчитаем прямой эффект от разработки системы для автоматизации логистических процессов предприятия пищевой промышленности, который включает в себя расчет трудовых и стоимостных показателей.

«Абсолютное снижение трудовых затрат в год в часах составило:

$$_{\Delta}T = 10300 – 7800 = 2500 \text{ часов.}$$

Рассчитаем коэффициент относительного снижения трудовых затрат:

$$К_Т = 2500 / 10300 * 100\% = 24\%.$$

Рассчитаем индекс снижения трудовых затрат или повышение производительности труда:

$$Y_Т = 10300 / 7800 = 1,32» [7]$$

«Осуществим расчет стоимостных показателей. Абсолютное снижение стоимостных затрат в год составит:

$$_{\Delta}C = 4141000 – 2379000 = 862000 \text{ рублей»[14].}$$

Динамика трудовых затрат представлена на рисунке 36. Данные до автоматизации посчитаны на основе приведенных формул, а трудовые затраты до автоматизации и после определены экспериментальным путем, замером времени обработки проезда одного автомобиля

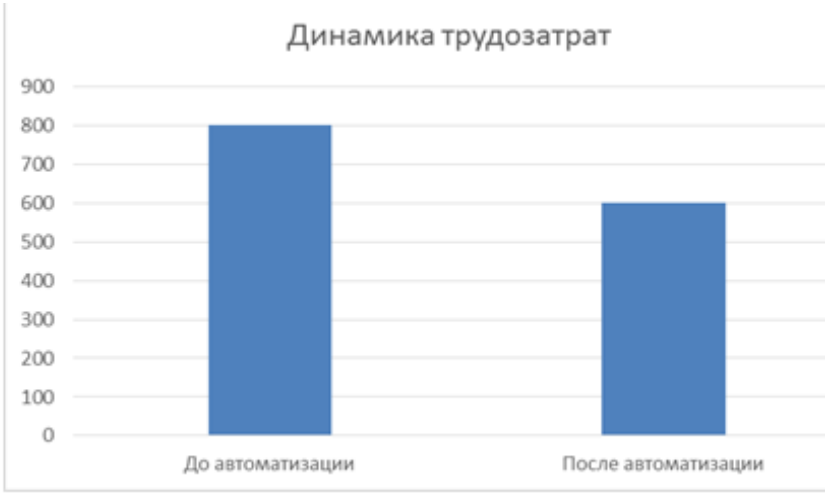


Рисунок 29- График трудовых затрат

«Коэффициент относительного снижения стоимостных затрат составит:

$$К_С = 762000 / 3141000 * 100\% = 25\%»[15].$$

Индекс снижения стоимостных затрат составит:

$$Y_С = 3141000 / 2379000 = 1,33» [8].$$

Динамика стоимостных затрат представлена на рисунке 37

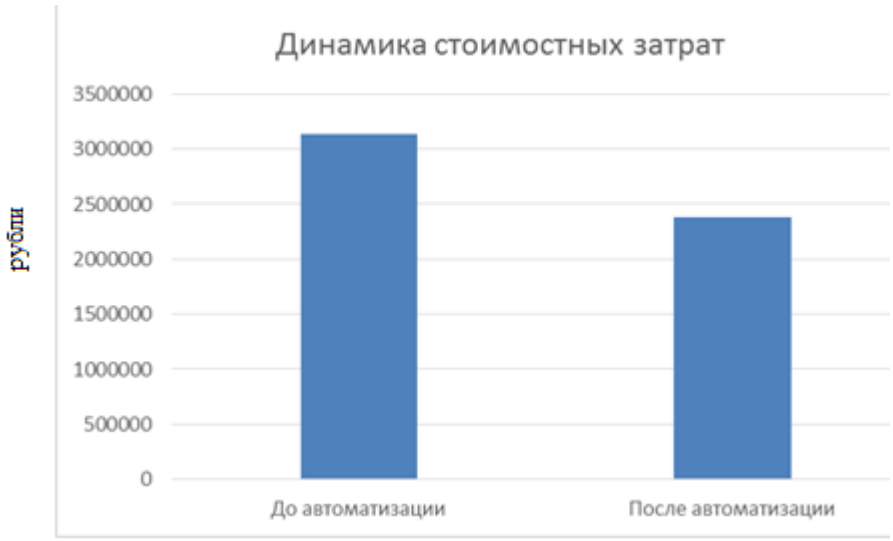


Рисунок 30 - График стоимостных затрат

«Произведем расчет срока окупаемости проекта. Для этого необходимо оценить затраты на разработку и внедрение информационной системы.

Далее рассчитаем единовременные капитальные затраты на разработку и внедрение системы ($К_П$). Затраты на разработку и внедрение системы рассчитываются по формуле:

$$K_{П} = C_{проект} + C_{прогр} + C_{внедр+отладка} + C_{доп}$$

$С_{проект}$ = 150 070 рублей.

$С_{прогр}$ = 341 360 рублей.

$С_{внедр}$ = 119 840 рублей.

$С_{доп}$ = 88 530 рублей.

$К_п$= 799 800 рублей» [16].

«Рассчитаем срок окупаемости проекта:

$$Т_{ок} = 799\,800 / 862\,000 = 7 \text{ месяцев}» [18]$$

В результате получилось, что трудовые затраты снизятся в год на 2500 часов, что составит 24 %. Стоимостные затраты составят 862 000 рублей или 25 %. Стоимостные затраты по проектному варианту ниже, чем по базовому в 1,32 раз, а трудовые затраты по проектному варианту ниже базовых в 1,33 раз. Срок окупаемости проекта составил 7 месяцев.

Выводы по главе 3

В третьей главе описан алгоритм работы информационной системы, а также произведена оценка экономическая эффективность проекта. Срок окупаемости проекта составит 7 месяцев. Представлен контрольный пример прототипа системы для автоматизации логистических процессов предприятия пищевой промышленности

Заключение

В ходе выполнения работы определена актуальность проблемы, рассмотрены аналоги выявлены их достоинства и недостатки. Предложена идея решения поставленной проблемы. Построены основные UML-диаграммы.

В результате было разработано информационное обеспечение для автоматизации логистических процессов предприятия пищевой промышленности. Разработан прототип программного модуля.

Информационная система для автоматизации логистических процессов предприятия имеет систему авторизации с разграничением ролей. Позволяет добавлять записи об автомобилях в целях предоставления доступа к проезду на территорию. Изменять сведения об автомобилях. Добавлять записи о владельцах автомобилей. Добавлять и удалять пользователей в системе. Обновлять данные о сроках истечения права доступа.

Система позволяет контролировать процесс пропуска автомобилей. Информационное обеспечение собирает и хранит данные о въездах и выездах, на основе которых формируются отчеты в виде таблицы и текста. Условиями, по которым формируется отчет являются – въезды и выезды в определенный день, въезды и выезды, но номеру автомобиля или по владельцу. Так же можно получить данные о том, какие автомобили имеют доступ к проезду на территорию и о том, какие автомобили находятся на территории на данный момент. Реализована возможность сортировки таблиц по столбцам и возможность экспорта отчетов.

Информационная система для автоматизации логистических процессов предприятия упрощает работу с данными и как следствие формирование отчетов. Хранение сведений об автомобилях в базе данных позволяет ускорить процесс пропуска автомобилей и оптимизировать работу сотрудников.

Рассчитана экономическая часть проекта. На основании проведенной работы были получены следующие практические результаты: разработан

прототип информационной системы по обработки заявок, внедрение которого позволяет снизить трудовые затраты сотрудников на 2500 часов в год.

Таким образом, выпускная квалификационная работа выполнена в полном соответствии с требованиями технического задания и с соблюдением сроков, установленных календарным планом.

Теоретическая значимость выпускной квалификационной работы состоит в систематизации полученных теоретических знаний

Результаты работы могут быть успешно внедрены в любую компанию, где протекают похожие бизнес-процессы.

Список используемой литературы и используемых источников

1. Бедердинова, О. И. Моделирование информационных систем на платформе SOFTWARE IDEAS MODELER: учеб. пособие / О.И. Бедердинова, Л.В. Кремлева, С.В. Протасова. — Москва: ИНФРА-М, 2019. — 166 с. - ISBN 978-5-16-107692-7

2. Дадян, Э. Г. Проектирование современных баз данных: учебное пособие / Дадян Э.Г. - Москва: НИЦ ИНФРА-М, 2017. - 120 с. ISBN 978-5-16-106529-7.

3. Кантелон М.. Node.js в действии / М. Кантелон, М. Хартер, Т. Головайчук, Н. Райлих - СПб.: Питер, 2018. – 548 с. ISBN 978-5-496-01079-5

4. Мартишин, С. А. Проектирование и реализация баз данных в СУБД MySQL с использованием MySQL Workbench. Методы и средства проектирования информационных систем и технологий. Инструментальные средства информационных систем: учебное пособие / С.А. Мартишин, В.Л. Симонов, М.В. Храпченко. — Москва: ФОРУМ: ИНФРА-М, 2022. — 160 с. — (Среднее профессиональное образование). - ISBN 978-5-8199-0811-2.Никитаева А. Ю. Корпоративные информационные системы: учебное пособие / Никитаева А.Ю. - Таганрог: Южный федеральный университет, 2019. - 149 с.

5. Немцова, Т. И. Компьютерная графика и web-дизайн: учебное пособие / Т.И. Немцова, Т.В. Казанкова, А.В. Шнякин; под ред. Л.Г. Гагариной. — Москва: ФОРУМ: ИНФРА-М, 2022. -400 с. - ISBN 978-5-8199-0790-0.

6. Озёрский С. В.. Информационная безопасность: практикум / С. В. Озёрский, И. В. Попов, М. Е. Рычаго, Н. И. Улендеева. - Самара: Самарский юридический институт ФСИН России, 2019. - 84 с. - ISBN 978-5-91612-276-3.

7. Полищук, Ю. В. Базы данных и их безопасность : учебное пособие / Ю.В. Полищук, А.С. Боровский. — Москва: ИНФРА-М, 2022. — 210 с. — (Высшее образование: Специалитет). — DOI 10.12737/1011088. - ISBN 978-5-

16-014924-0.

8. Похилько, А.Ф. CASE-технология моделирования процессов с использованием средств BPWin и ERWin: учебное пособие / А.Ф. Похилько, И.В. Горбачев. - Ульяновск: УлГТУ, 2019. - 120 с.

9. Проектирование современных баз данных: учебно-методическое пособие / Дадян Э.Г. - М.:НИЦ ИНФРА-М, 2019. - 120 с.

10. Рычаго, М. Е. Основы защиты информации: учебное пособие / М. Е. Рычаго, И. В. Ершова, Р. Н. Тихомиров. - Владимир: ВЮИ ФСИН России, 2017. - 68 с. - ISBN 978-5-93035-622-9.

11. Свод знаний по управлению бизнес-процессами. BPM СВОК 3.0: Учебное пособие / Под ред. Белайчук А.А. - М.:Альпина Пабл., 2018. - 480 с

12. Симдянов И. В., Программирование. Ступени успешной карьеры. / Симдянов И. В., Кузнецов М. В. // - БХВ-Петербург, 2018. – 320 с.

13. Учитесь видеть бизнес-процессы: Практика построения карт потоков создания ценности учебное пособие / Ротер М., Шук Д., Муравьева Г., - 4-е изд. - М.:Альп. Бизнес Букс, 2019.

14. About the Unified Modeling Language Specification Version 2.5.1 // OMG | Object Management Group. - [Электронный ресурс] - URL: https://www.omg.org/spec/UML (дата обращения: 27.09.2023).

15. BpWin [Электронный ресурс] URL: http://habrahabr.ru/.(дата заявки: 27.09.2023)

16. React – JavaScript-библиотека для создания пользовательских интерфейсов // Знакомство с JSX – React. - [Электронный ресурс] - URL: https://ru.reactjs.org/docs/introducing-jsx.html

17. VideoNet PSIM - система автоматизации проезда транспорта с распознаванием номеров // Интеллектуальная система видеонаблюдения и безопасности VideoNet. - [Электронный ресурс] - URL: http://www.videonet.ru/reshenie-avto-kpp.html (дата обращения: 27.09.2023).

18. Visio 2010: руководство для начинающих [Электронный ресурс]. URL: support.office.com (дата заявки: 27.09.2023)

19. КОДОС-Транспорт. ПО Интеграционной системы на сайте КОДОС // ИСБ КОДОС - официальный сайт производителя. - [Электронный ресурс] - URL: https://kodos.ru/product/kodos-transport (дата обращения: 27.09.2023).

20. Система HIT ANPR: // HIGH INFORMATION TECHNOLOGIES SIA. - [Электронный ресурс] - URL: https://www.hi-technologies.lv/index.php/ru/produktsiya-uslugi1/sistemy-kontrolya-propusknykh-punktov/hit-anpr-opredelenie-nomerov-avtomobilej (дата обращения: 27.09.2023).

Приложение А

Пример JSX компонента – модальное окно

```
export const Modal = ({ isModalOpen, clickCloseModal, modalData }) => {
    return (
      <>
            <div  className={`modal__backdrop  ${isModalOpen  ?  `modal__show`  :  `modal__hide`}`}
tabIndex="-1">
          </div>
          <div className={`modal__container ${isModalOpen ? `modal__show` : `modal__hide`}`}>
            {modalData.modalType === 'plate' &&
             <PlateModal closeModal={clickCloseModal} accept={modalData.accept} />}
            {modalData.modalType === 'change' &&
             <ChangeModal closeModal={clickCloseModal} plate={modalData.plate}/>}
          </div>
      </>
    );
};
```

Приложение Б

Пример использования медиа запросов

```
.signin__logo {
    display: flex;
    flex-direction: column;
    text-align: center;
    color: $color-main;
    font-weight: bold;
    line-height: 1.1;
    text-transform: uppercase;
    white-space: nowrap;
    margin-bottom: 3rem;

    @media (min-width: $screen-lg) {
        font-size: 6rem;
    }

    @media (min-width: $screen-md) and (max-width: ($screen-xl - 1)) {
        font-size: 5rem;
    }

    @media (max-width: ($screen-md - 1)) {
        font-size: 4.5rem;
    }
}
```

Приложение В

POST и GET запросы входа в учетную запись

```
app.get('/login', (req, res) => {
    if (req.session.user) {
        res.send({ loggedIn: true, user: req.session.user });
    } else
        res.send({ loggedIn: false });
});

app.post('/login', (req, res) => {
    const username = req.body.username;
    const password = req.body.password;

    db.query(
        "SELECT * FROM user WHERE user_name = ?",
        [username],
        async (err, result) => {
            if (err)
                res.send({ err: 'Произошла ошибка. Пожалуйста, попробуйте снова позже!' });

            if (result.length > 0 && username === result[0].user_name) {
                const comparison = await bcrypt.compare(password, result[0].password);
                if (comparison) {
                    req.session.user = {userName: result[0].user_name, role: result[0].role};
                    req.session.loggedIn = true;
                    res.send(result);
                } else res.send({ err: 'Неверный логин или пароль!' });
            } else res.send({ err: 'Неверный логин или пароль!' });
        }
    );
});
```

Приложение Г

POST запрос добавления автомобиля в БД

```
app.post("/addCar", (req, res) => {
    let plate = req.body.plate;
    const region = req.body.region;
    const brand = req.body.brand;
    const idGates = '1';

    let idCar;
    let isCarAlreadyExist = false;
    let isCarExpired = false;
    let hasGates = false;

    const fixPlate = () => {
      return new Promise((resolve, reject) => {
        plate = translit(plate);
        if (!plate) {

          return reject('Неверный формат номера!');
        }

        return resolve(plate);
      });
    }

    const checkCarExist = () => {
      return new Promise((resolve, reject) => {
        db.query(
          "SELECT id_car FROM car WHERE license_plate=? AND region=?",
          [plate, region], (err, result) => {
            if (err) {
              console.log(err);
              return reject('Произошла ошибка. Пожалуйста, попробуйте снова позже!');
            }

            if (result.length !== 0) {
              isCarAlreadyExist = true;
              idCar = result[0].id_car;
            }

            return resolve(result);
          }
        );
      });
    }

    const checkDateExpired = () => {
      return new Promise((resolve, reject) => {
        db.query(
          "SELECT id_car FROM car WHERE id_car = ? AND expiration_date < CURDATE()",
          [idCar], (err, result) => {
            if (err) {
              console.log(err);
              return reject('Произошла ошибка. Пожалуйста, попробуйте снова позже!');
            }

            if (result.length !== 0)
              isCarExpired = true;

            return resolve(result);
```

```
                });
            });
        }

        const addCar = () => {
            return new Promise((resolve, reject) => {
                db.query(
                    "INSERT INTO car (car_brand, license_plate, region, start_date, expiration_date) VALUES (?,
?, ?, CURDATE(), DATE_ADD(CURDATE(), INTERVAL 1 YEAR))",
                    [brand, plate, region], (err, result) => {
                        if (err) {
                            console.log(err);
                            return reject('Произошла ошибка. Пожалуйста, попробуйте снова позже!');
                        }

                        idCar = result.insertId;
                        return resolve(result);
                    });
            });
        }

        const checkGates = () => {
            return new Promise((resolve, reject) => {
                db.query(
                    "SELECT * FROM gates_allowed WHERE id_car = ? AND id_gates = ? ",
                    [idCar, idGates], (err, result) => {
                        if (err) {
                            console.log(err);
                            return reject('Произошла ошибка. Пожалуйста, попробуйте снова позже!');
                        }

                        if (result.length !== 0) {
                            hasGates = true;
                            if (!isCarExpired)
                                res.send({ err: 'У автомобиля уже есть доступ к данной проходной!' });
                        }
                        return resolve(result);
                    });
            });
        }

        const addGates = () => {
            return new Promise((resolve, reject) => {
                db.query(
                    "INSERT INTO gates_allowed (id_car, id_gates) VALUES (?, ?)",
                    [idCar, idGates], (err, result) => {
                        if (err) {
                            console.log(err);
                            return reject('Произошла ошибка. Пожалуйста, попробуйте снова позже!');
                        }

                        if (isCarAlreadyExist) {
                            if (!isCarExpired)
                                res.send({ message: 'Автомобилю предоставлен доступ к данной проходной!' });
                        } else res.send({ message: 'Запись успешно добавлена!' });
                        return resolve(result);
                    });
            });
        }
```

```
            const updateDate = () => {
                return new Promise((resolve, reject) => {
                    db.query(
                        "UPDATE car SET expiration_date = DATE_ADD(CURDATE(), INTERVAL 1 YEAR) WHERE id_car=?",
                        [idCar], (err, result) => {
                            if (err) {
                                console.log(err);
                                return reject('Произошла ошибка. Пожалуйста, попробуйте снова позже!');
                            }

                            if (!hasGates)
                             res.send({ message: 'Дата истечения прав доступа обновлена, автомобилю предоставлен доступ к данной проходной!' });
                            else res.send({ message: 'Дата истечения прав доступа успешно обновлена!' });
                            return resolve(result);
                        });
                });
            }

            async function CarAdding() {

                try {
                    await fixPlate();
                    await checkCarExist();
                    if (isCarAlreadyExist) {
                        await checkDateExpired();
                        await checkGates();
                    }
                    if (!isCarAlreadyExist)
                        await addCar();
                    if (!hasGates)
                        await addGates();
                    if (isCarExpired)
                        await updateDate();

                } catch (error) {
                    res.send({ err: error });
                }
            }
            CarAdding();
        });
```

Приложение Д

Запрос проверки прав доступа автомобиля

```
app.post('/inOutCar', (req, res) => {
    const direction = req.body.direction;

    let plate = req.query.plate;
    if (!plate) {
      execFileSync('tempExe.exe', []);
      // console.log('fine');
      plate = fs.readFileSync('car.txt', 'utf8').toString().split("\n");
    }
    else {
      plate = plate.split(',');
    }
    console.log(direction);
    let log = [];
    if (req.session.log)
      log = req.session.log;

    console.log(plate);

    if (plate[0].trim() === '404') {
      log.push(new Date().toLocaleTimeString([], { hour: '2-digit', minute: '2-digit', second: '2-digit' }) + '
Номер автомобиля не распознан');
      req.session.log = log;
      res.send({ log: log, carPlateErr: 'Номер автомобиля не распознан!' });
    }
    else {
      let warning = false;
      let idCar, idDate;
      let idGates = '1';

      plate[0] = translit(plate[0].trim());

      let CheckCarInOut = () => {
        return new Promise((resolve, reject) => {
          db.query(
            "SELECT id_car, expiration_date FROM car WHERE license_plate = ? AND region = ?",
            [plate[0], plate[1]], (err, result) => {
              if (err) {
                console.log(err);
                return reject(err);
              }
              else {
                if (result.length === 0) {
                  log.push(new Date().toLocaleTimeString([], { hour: '2-digit', minute: '2-digit', second:
'2-digit' }) + ' Автомобиль с номером ' + plate[0] + ' ' + plate[1] + ' отсутствует в базе данных');
                  req.session.log = log;
                  warning = true;
                  if (direction === 'in') {
                    res.send({ log: log, err: 'Автомобиля с данным номером нет в базе данных!' });
                  } else res.send({ log: log });
                }
                else {
                  let expDate = new Date(result[0].expiration_date);
                  expDate.setDate(expDate.getDate() + 1);
                  if (new Date() < expDate) {
                    idCar = result[0].id_car;
                  }
                  else {
```

```
                                warning = true;
                                    log.push(new Date().toLocaleTimeString([], { hour: '2-digit', minute: '2-digit',
second: '2-digit' }) + ' У автомобиля с номером ' + plate[0] + ' ' + plate[1] + ' истекли права доступа');
                                req.session.log = log;
                                if (direction === 'in') {
                                   res.send({ log: log, err: 'У данного автомобиля истекли права доступа!' });
                                } else res.send({ log: log });
                             }
                          }
                          return resolve(result);
                       }
                    }
                 );
              });
           }

           let CheckRightGates = () => {
              return new Promise((resolve, reject) => {
                 db.query(
                    "SELECT * FROM gates_allowed WHERE id_car = ? AND id_gates = ?",
                    [idCar, idGates], (err, result) => {
                       if (err) {
                          console.log(err);
                          return reject(err);
                       } else {
                          if (result.length === 0) {
                             warning = true;
                              log.push(new Date().toLocaleTimeString([], { hour: '2-digit', minute: '2-digit', second:
'2-digit' }) + ' Автомобиль с номером ' + plate[0] + ' ' + plate[1] + ' не имеет доступа к данной проходной');
                             req.session.log = log;
                             if (direction === 'in') {
                                   res.send({ log: log, err: 'У автомобиля с данным номером нет доступа к этой
проходной!' });
                             } else res.send({ log: log });
                          }
                          return resolve(result);
                       }
                    }
                 );
              });
           }

           let AddDate = () => {
              return new Promise((resolve, reject) => {
                 db.query(
                    "SELECT id_date FROM `date` WHERE `date` = CURDATE()",
                    (err, result) => {
                       if (err) {
                          console.log(err);
                          return reject(err);
                       } else {
                          if (result.length === 0) {
                             db.query(
                                "INSERT INTO `date` (`date`) VALUES (CURDATE())",
                                (err, result) => {
                                   if (err) {
                                      console.log(err)
                                   } else idDate = result.insertId;
                                });
```

```
                        }
                        else {
                            idDate = result[0].id_date;
                        }
                        return resolve(result);
                    }
                }
            );
        });
    }

    let CheckAddArrDate = () => {
        return new Promise((resolve, reject) => {
            db.query(
                "SELECT * FROM arriving_date WHERE id_date = ? AND id_car = ?",
                [idDate, idCar], (err, result) => {
                    if (err) {
                        console.log(err);
                        return reject(err);
                    }
                    else {
                        if (result.length === 0) {
                            db.query(
                                "INSERT INTO arriving_date (id_date, id_car) VALUES (?, ?)",
                                [idDate, idCar]
                            );
                        }
                        return resolve(result);
                    }
                }
            );
        });
    }

    let AddCarInOut = () => {
        return new Promise((resolve, reject) => {

            if (direction === 'in') {
                db.query(
                    "UPDATE arriving_date SET arrival_time=CURTIME(), departure_time=NULL WHERE
id_date=? AND id_car=?",
                    [idDate, idCar], (err, result) => {

                        if (err) {
                            console.log(err);
                            return reject(err);
                        } else {
                            console.log(direction);
                            log.push(new Date().toLocaleTimeString([], { hour: '2-digit', minute: '2-digit', second:
'2-digit' }) + ' Автомобиль с номером ' + plate[0] + ' ' + plate[1] + ' въехал');
                            req.session.log = log;
                            res.send({ log: log, message: 'Автомобиль может быть пропущен!' });
                            return resolve(result);
                        }
                    }
                );
            }
            else {
                db.query(
```

```
                    "UPDATE arriving_date SET departure_time=CURTIME() WHERE id_date = ? AND
id_car = ?",
                [idDate, idCar], (err, result) => {

                    if (err) {
                        console.log(err);
                        return reject(err);
                    }
                    else {
                        log.push(new Date().toLocaleTimeString([], { hour: '2-digit', minute: '2-digit', second:
'2-digit' }) + ' Автомобиль с номером ' + plate[0] + ' ' + plate[1] + ' выехал');
                        req.session.log = log;
                        res.send({ log: log });
                        return resolve(result);
                    }
                }
            )
        }
    });
}

async function CarInOut() {

    try {
        await CheckCarInOut();
        if (!warning)
            await CheckRightGates();
        if (!warning)
            await AddDate();
        if (!warning)
            await CheckAddArrDate();
        if (!warning)
            await AddCarInOut();
    } catch (error) {
        console.log(error)
    }
}

CarInOut();
}
```