**Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации**

**Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский технический университет связи и информатики»**

**Разрешаю**

**допустить к защите**

**Зав. Кафедрой ИСУиА**

**Воронова Л. И.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_\_\_г.**

**БАКАЛАВРСКАЯ РАБОТА**

**НА ТЕМУ «Разработка клиент-серверного приложения для системы электронной очереди в транспортной компании»**

|  |  |  |
| --- | --- | --- |
| **Студент:** |  |  |
| **Научный руководитель:** |  |  |
|  |  |  |

**Москва 2021 г.**

**Федеральное агентство связи**

**Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский технический университет связи и информатики»**

Кафедра \_\_\_\_\_\_«Интеллектуальные системы в управлении и автоматизации»

«Утверждаю»

Зав. кафедрой ИСУиА

\_\_\_\_\_\_\_\_\_Воронова Л.И.

«\_\_\_\_»\_\_\_\_\_\_\_\_2020 г.

**З А Д А Н И Е**

**на выпускную квалификационную работу**

Студенту Ларионову Борису Анатольевичу гр. БАП1651

Направление (специальность) 15.03.04 «Автоматизация технологических процессов и производств»

Форма выполнения выпускной квалификационной работы бакалаврская работа

Тема выпускной квалификационной работы: «Разработка клиент-серверного приложения для системы электронной очереди в транспортной компании»

Утверждена приказом ректора № 1405-С от 26.11.2020 г.

|  |  |
| --- | --- |
| 1. Исходные данные   1. ГОСТ 19.701 ЕСПД. Схемы алгоритмов, программ, данных и систем 2. ГОСТ 34.321-96. Информационные технологии. Система стандартов по базам данных. Эталонная модель управления данными   2. Содержание расчетно-пояснительной записки  (перечень подлежащих разработке вопросов)  Введение  Глава 1. Анализ предметной области для разработки системы  1.1 Общие сведения об организации  1.2 Организационная структура управления транспортной компании  1.3 Анализ методов обработки информации на предприятии  1.4 Функциональные требования к разрабатываемой системе  1.5 Анализ средств и инструментов для разработки ИС  Выводы  Глава 2. Проектирование информационной системы для учета посетителей транспортной компании  2.1 Анализ принципов работы с СУБД  2.2 Структурный анализ входной информации системы  2.3 Структурный анализ выходной информации системы  2.4 Выбор аппаратных и программных средств для разработки системы  2.5. Даталогическое и инфологическое проектирование ИС  Выводы  Глава 3. Разработка информационной системы для учета посетителей транспортной компании  3.1 Разработка СУБД для информационной системы  3.2 3.2. Разработка программного обеспечения информационной системы учета посетителей  3.3. Разработка интерфейса программного продукта  3.4. Отладка и внедрение программного продукта  Выводы  Заключение  Список использованной литературы  3. Вопросы конструктивных разработок  4. Разработка вопросов по экологии и безопасности жизнедеятельности  5. Технико-экономическое обоснование (подлежащее расчету)  6. Перечень графического материала (с точным указанием обязательных чертежей)   1. Организационная структура ТК 2. Требования к разрабатываемой ИС 3. Анализ входной и выходной информации 4. Разработка СУБД для ИС, интерфейса пользователя 5. Отладка системы, результаты внедрения | Объем работы в % и сроки выполнения по разделам  5% 09.12.2020  25% 15.12.2020  30% 22.12.2020  35% 09.01.2021  5% 22.01.2021 |

7. Консультанты по ВКР (с указанием относящихся к ним разделов проекта):

\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (ФИО)

\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (ФИО)

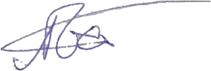
8. Срок сдачи студентом законченной ВКР: 08.02.2021

Дата выдачи задания: 27.11.2020

Руководитель Усачев В. А., ассистент (подпись) (ФИО)

нагрузка

(штатная или почасовая)

Задание принял к исполнению 

(подпись студента)

Отзыв научного руководителя

**Оглавление**

[Отзыв научного руководителя 4](#_Toc66390541)

[Введение 7](#_Toc66390542)

[Глава 1. Анализ предметной области для разработки системы. 10](#_Toc66390543)

[1.1. Общие сведения об организации. 10](#_Toc66390544)

[1.2. Организационная структура управления транспортной компанией. 12](#_Toc66390545)

[1.3. Анализ методов обработки информации на предприятии. 21](#_Toc66390546)

[1.4. Функциональные требования к разрабатываемой системе. 22](#_Toc66390547)

[1.5. Анализ средств и инструментов для разработки. 24](#_Toc66390548)

[Выводы 27](#_Toc66390549)

[Глава 2. Проектирование информационной системы для учета посетителей транспортной компании. 29](#_Toc66390550)

[2.1. Анализ принципов работы СУБД. 29](#_Toc66390551)

[2.2. Структурный анализ входной информации системы. 31](#_Toc66390552)

[2.3. Структурный анализ выходной информации системы. 33](#_Toc66390553)

[2.4. Выбор аппаратных и программных средств для разработки системы. 35](#_Toc66390554)

[2.5. Даталогическое и инфологическое проектирование ИС. 41](#_Toc66390555)

[Выводы 53](#_Toc66390556)

[Глава 3. Разработка информационной системы для учета посетителей транспортной компании. 54](#_Toc66390557)

[3.1. Разработка СУБД для информационной системы. 54](#_Toc66390558)

[3.2. Разработка программного обеспечения информационной системы учета посетителей. 59](#_Toc66390559)

[3.3. Разработка интерфейса программного продукта. 63](#_Toc66390560)

[3.4. Отладка и внедрение программного продукта. 73](#_Toc66390561)

[Выводы 74](#_Toc66390562)

[Заключение 76](#_Toc66390563)

[Список использованной литературы 77](#_Toc66390564)

[Приложение A. Код создания класса Manager 79](#_Toc66390565)

[Приложение B. Код класса ManagesrViewStatuses 82](#_Toc66390566)

[Приложение C. Алгоритм работы метода распределения талонов 84](#_Toc66390567)

[Приложение D. Алгоритм работы метода авторизации 85](#_Toc66390568)

[Приложение E. Описание программного интерфейса API для рабочего места оператора 86](#_Toc66390569)

Введение

Программные продукты электронной очереди распространены в крупных организациях, со спецификой работы, предусматривающей общение с большим количеством людей. Электронная очередь позволяет оптимизировать время ожидание посетителей: проинформировав посетителя количеством взятых талонов перед ним, уведомляя о состоянии потоков клиентов на специализированном табло и распределяя очередь по компетенции сотрудников, уменьшает психологическую нагрузку на сотрудников и клиентов, позволяет производит анализ посещаемости и нагрузки офисов в разрезах разных промежутков времени, ведет учет времени обслуживания по конкретным услугам и конкретных операторов.

Целью выпускной квалификационной работы является оптимизация процесса учета посетителей с помощью системы управления электронными очередями в транспортной компании.

Задачами выпускной квалификационной работы являются:

* проведения анализа исследуемой предметной области;
* выполнение концептуального проектирования программного продукта и представление в виде информационно-логической модели предметной области;
* обоснование необходимости оптимизации текущей реализации системы учета посетителей в компании;
* обоснование выбора программныхсредств для разработки программного продукта;
* определение требований к функциональным характеристикам разрабатываемой информационной системы;
* разработка пользовательского и программного интерфейсов программного продукта;

Данная выпускная квалификационная работа состоит из трёх частей.

* В аналитической части предметной области разработки системы был произведен комплекс работ, который направлен на обоснование надобности автоматизации компании:

1. установлена сущность задачи;

2. изображены основные свойства информационной системы;

3. предоставлено описание основных бизнес-процессов рассматриваемой компании;

4. проанализированы вопросы, которые связанны с анализом имеющихся разработок в данной области.

Также в первой главе были проведены обоснования проектных решений, которые должны быть приняты по информационному, программному, техническому и технологическому обеспечению компании.

* Во второй главе проектирования информационной системы рассматривается выбор программных комплексов и технологий для создания приложения учета посетителей транспортной компании, где затрагиваются вопросы основ архитектуры информационных систем и технологий «клиент-сервер», возможности и особенности различных систем баз данных, а также перспективы развития систем управления базами данных.
* В третьей главе выпускной квалификационной работы был произведен обзор методик разработки информационной системы для учета посетителей транспортной компании, где рассмотрен следующий фундамент информационной системы: СУБД, анализ технологического процесса, разработка интерфейса, отладка и внедрение.

При разработке выпускной квалификационной работы использовались следующие методы исследования:

* Анализа – исследование в разрабатываемом проекте составных частей и их последующее изучение.
* Классификации – это метод, основанный на распределении области разработки по присущим им признакам.
* Эксперимента – это практический метод исследования информационной системы с вытекающими из него заключениями.

# Анализ предметной области для разработки системы.

В данной главе рассматривается устройство транспортной компании, используемые технические средства и целесообразность разработки рассматриваемой информационной системы.

## Общие сведения об организации.

В данной выпускной квалификационной работе рассматривается транспортная компания, занимающаяся перевозкой сборных грузов. Рассматриваемая компания является крупным игроком на рынке услуг по перевозкам грузов, оказывает услуги как юридическим лицам, так и физическим лицам. В связи с чем активно конкурирует с другими участниками рынка. У данной компании имеется множество офисов по стране для обслуживания клиентов. В текущей работе рассматривается устройство компании внутри одного города с головным офисом и шестью точками приёмки-выдачи грузов.

Компания позиционируется как IT-компания, за счет внедрения автоматизации для предоставления более качественных услуг.

Автоматизация затрагивает следующие информационные системы:

### Документооборот

В документообороте выполняется составление договоров, упрощенная передача внутри компании, а также при взаимодействии с клиентами компании.

### ERP

Система «ERP» – это комплексная автоматизация бизнес-процессов компании и учета товаров на базе «1С».

### **Мобильное** **приложение**

Мобильное приложение выполняет несколько функций:

* взаимодействие с самим перевозчиком – отслеживание груза, оперативная связь и своевременное реагирование на форс-мажорные обстоятельства;
* взаимодействие с клиентом – передача документов, предоставление информации о местоположении груза и прочей информации о нем, а так же оформление документов для предстоящей отправки посылки с оформлением записи на прием в отделение для отправки или получения консультаций.

### Сайт

Представляющий функционал сайта аналогичный представленному в информационной системе «Мобильное приложение» описанному выше, а также обновляемую информацию согласно указанной выше информационной системе «Документооборот» для предоставления самой актуальной информации: документов, договоров и тарифов по перевозке.

В головном офисе имеется свой отдел разработки программного обеспечения, системные администраторы и инженеры, в котором работают специалисты разных направлений и технологий. Благодаря этому компания сама обеспечивает свои потребности в сфере информационных технологий, обслуживает сайт компании, систему «ERP» построенную на базе «1С», сетевую связь устройств всей компании. Таким образом рассматриваемой компании нет необходимости лишний раз обращаться за услугами по разработке к сторонним лицам, что подтверждает успешность внедрения и рационального использования ресурсов компании и автоматизации процессов производств.

## Организационная структура управления транспортной компанией.

Структура компании классифицируется как линейная (административное подчинение). В рассматриваемой компании насчитывается большое количество подразделений различной специализации. Схема организационной структуры представлена на рисунке 1.



Рисунок 1. Схема организационной структуры.

### Генеральный директор

Главой компании является генеральный директор, это главное ответственное лицо, имеющее право подписи документов. Он является последней инстанцией при обработке договоров и финансовых служебных записок, отвечают за утверждение этих документов в системе «Документооборот».

### Администрация

Для помощи директору в различных направлениях деятельности, сформирован отдел «Администрация». В составе администрации находятся заместители генерального директора по основным видам деятельности: логистическое управление, стратегическое развитие и цифровая интеграция бизнеса.

### Юридический департамент

Юридический отдел обеспечивает соблюдение законности в деятельности компании и защите его правовых интересов, а также контролирует ведение корпоративной работы. Участвует в разработке и осуществлении мероприятий по укреплению договорной, финансовой и трудовой дисциплины, обеспечивает сохранность имущества компании, участвует в урегулировании спорных вопросов, конфликтных ситуаций по вопросам налогообложения и регистрирует договора в органах государственной регистрации.

В штат юридического отдела входят следующие работники: начальник отдела, юрисконсульт, помощник юрисконсульта.

### Департамент управления персоналом

Департамент управления персоналом отвечает за рабочую дисциплину, повышение квалификации и комфортное материальное обеспечение рабочего места сотрудника. оформление и учет кадров, контроль трудовой дисциплины, организацию и проведения обучающих мероприятий, оценку компетентности сотрудников, контроль за пожарной безопасностью.

В подчинении у директора по персоналу состоят следующие отделы: отдел кадров, отдел по обучению и развитию персонала, административно-хозяйственный отдел.

* *Отдел кадров*

В задачи отдела кадров входит:

* Оформление и учет кадров
* Расстановка и воспитание кадров
* Контролирование состояния трудовой дисциплины
* Обеспечение прав, льгот и гарантий работников компании

В штат отдела кадров входят следующие работники: начальник отдела и менеджер по персоналу.

* *Отдел по обучению и развитию персонала*

В задачи данного отдела входит:

* Организация, проведение и контролирование мероприятий по обучению и развитию персонала компании
* Организация и проведение оценки персонала компании

В штат отдела по обучению и развития персонала входят следующие работники: начальник отдела и менеджер по обучению.

* *Административно-хозяйственный отдел*

В задачи данного отдела входит:

* создание комфортных условий /функционирование офиса/склада компании
* Материально-хозяйственное обеспечение офиса/склада компании
* Обеспечение противопожарной безопасности в офисе/на складе компании

В штат административно-хозяйственного отдела входят следующие работники: начальник отдела, разнорабочий и уборщица

### Департамент экономики и финансов

Департамент экономики и финансов отвечает за финансовый контроль компании (включая планирование, реализацию и контроль финансово-хозяйственной деятельности), бюджетирование и формирование финансовой отчетности. Руководителями данного департамента являются финансовый директор и заместитель финансового директора, в подчинении экономисты и следующие отделы: централизованная бухгалтерия, служба оформления.

* *Централизованная бухгалтерия*

В задачи централизованной бухгалтерии входит:

* Ведение бухгалтерского учета хозяйственно-финансовой деятельности компании;
* Контролирование экономного использования материальных, трудовых и финансовых ресурсов, сохранностью собственности компании;
* Формирование полной и достоверной информации о результатах финансовой и хозяйственной деятельности компании. Составление баланса и оперативных сводных отчетов о доходах и расходах денежных средств, об использовании бюджета, другой бухгалтерской, статистической и управленческой отчетности, предоставление их по необходимости в соответствующие органы.

В штат бухгалтерии входят следующие работники: главный бухгалтер, бухгалтер по учету расчета заработной платы и бухгалтер.

* *Служба оформления*

Кассир-операционист отвечает за правильное оформление первичных бухгалтерских документов (при осуществлении приема/выдачи или хранения груза) для клиента.

В обязанности работы входит работа с наличными денежными средствами и платежными картами (выставление счетов клиентам, учет наличных денежных средств, кассовая отчетность)

### Департамент логистики

Департамент логистики занимается обеспечением своевременной доставки грузов клиентам компании. В задачи отделов департамента логистики входит:

* *Департамент ответ. хранения*

В задачи департамента ответственного хранения входит: хранение, прием и выдача грузов, загрузка/выгрузка большегрузного транспорта, а также оформление складской документации на основном складе (Москва и МО).

В штат департамента ответственного хранения входят следующие работники: начальник склада ответственного хранения, приемосдатчик, упаковщик грузов, кладовщик-приемосдатчик, упаковщик грузов и кладовщик.

* *Склад*

В задачи данного отдела входит: хранение, прием и выдача грузов, загрузка/выгрузка большегрузного транспорта, а также оформление складской документации на складе временного хранения (Регионы).

В штат склада входят следующие работники: начальник склада, приемосдатчик, упаковщик грузов, кладовщик, и грузчик.

* *Отдел логистики*

В задачи отдела логистики входит: организация и обеспечение функционирования логистических процессов компании, а именно: анализ, планирование, организация, управление, контролирование за исполнением движения товаров до грузополучателя (включая погрузо-разгрузочных работ), а также оценка сметы затрат и статей бюджета.

В штат отдела логистики входят следующие работники: начальник отдела, специалист отдела логистики, менеджер по логистике и менеджер по оформлению сопроводительной документации.

* *Отдел транспортировки*

В задачи отдела транспортировки входит: обеспечение транспортно-экспедиционной перевозки, доставки, сохранности и иного обслуживания грузов до грузополучателя, согласно координации маршрута сотрудником отдела логистики.

В штат отдела транспортировки входят следующие работники: начальник отдела, водитель-экспедитор и водитель погрузчика и экспедитор.

### Департамент продаж

Департамент продаж взаимодействует с клиентами компании.

В задачи отделов департамента продаж входит:

* *Отдел продаж*

Отдел продаж делится на несколько отделов: «Отдел продаж (Москва и МО)» и «Отдел продаж (Регионы)».

Отдел продаж (Москва и МО) отвечает за Москву и Московскую область и делиться на округа (ЦАО, САО, ВАО, Подольск и т.д.). За каждый округ отвечает начальник отдела продаж, у которого в подчинении директора по продажам.

Отдел продаж (Регионы) отвечает за регионы России (Красноярск, Омск, Волгоград и т.д.). За каждый регион отвечает региональный управляющий, у которого в подчинении директора по продажам.

В задачи отдела продаж входит: выполнение плана продаж логистических услуг, развитие и поддержание партнерских отношений с ключевыми клиентами, презентации, возвращение ушедших клиентов, заключение договоров, прием и обработка заказов клиентов, оперативное решение всех возникающих вопросов с клиентами, ведение отчетной документации и контроль дебиторской задолженности.

В штат департамента продаж входят следующие работники: региональный управляющий, начальник отдела продаж, директор по продажам, старший менеджер по продажам, менеджер по продажам, менеджер по работе с ключевыми клиентами, менеджер по продаже логистических услуг и ассистент отдела продаж.

* *Отдел по работе с претензиями*

Отдел по работе с претензиями (Далее - ОРП) отвечает за обработку входящих претензий от клиентов с целью урегулирования спорных ситуаций (потеря, повреждение, задержка груза и т.д.), с помощью юридического отдела или предоставления скидок в ограниченные сроки.

В штат отдела «ОРП» входят следующие работники: начальник отдела и специалист по работе с претензиями.

* *Отдел контроля качества*

Отдел контроля качества (Далее - ОКК) контролирует обработку входящих заявок от клиентов, отвечая на благодарственные письма или отправляя претензионные заявки в отдел «ОРП» в ограниченные сроки.

В штат отдела «ОКК» входят следующие работники: начальник отдела и специалист отдела контроля качества.

### Департамент информационных технологий

Департамент информационных технологий отвечает за техническую часть процессов компании.

В задачи отделов департамента информационных технологий входит:

* *Отдел разработки ПО*

В задачи данного отдела входит:

* Автоматизация процессов компании в соответствии с поступающими заявками от сотрудников компании.
* Поддержание информационных систем компании в работоспособном и актуальном состоянии.

В штат отдела разработки программного обеспечения входят следующие работники: начальник отдела и программисты.

* *Отдел технической поддержки*

В задачи данного отдела входит: поддержание в работоспособном состоянии ИС компании

В штат отдела технической поддержки входят следующие работники: начальник отдела и специалист технической поддержки.

* *Отдел системного администрирования*

В задачи данного отдела входит:

* Поддержание в работоспособном состоянии серверов и рабочих станций
* Обеспечение сохранности и безопасности обработки данных, содержащихся в информационных системах компании.
* Обеспечение работы сетей передачи данных, связности сетей передачи данных с соблюдением сетевой безопасности и безопасности межсетевого взаимодействия
* Проведение инвентаризации оргтехники компании (ПО, ПК, Сервера и т.д.)

В штат отдела системного администрирования входят следующие работники: начальник отдела и системный администратор.

### Департамент защиты ресурсов

Департамент защиты ресурсов отвечает за экономическую, информационную и пропускную безопасность компании.

В задачи отделов департамента защиты ресурсов входит:

* *Отдел экономической безопасности*

В задачи данного отдела входит:

* Выполнение задач по выявлению и пересечению рисков и угроз экономической и информационной безопасности компании
* Обеспечение комплексной защиты информации, соблюдения государственной тайны.
* Обеспечение режима сохранения коммерческой тайны, принятие мер по предупреждению нарушения ее конфиденциальности.

В штат отдела экономической безопасности входят следующие работники: начальник отдела и специалист по информационной безопасности.

* *Отдел охраны*

В задачи данного отдела входит контролирование системы охраны объектов, пропускного и внутриобъектового режима компании, несанкционированного выноса ТМЦ компании, а также контролирование выполнения требований Законов РФ (ограничительные меры индивидуальной защиты).

### ЭСК

Отдел ЭСК (Экспедиционно-складской комплекс) отвечает за хранение, прием и выдачу грузов, оформление складской документации на складе временного хранения и транспортировку груза.

В штат сотрудников отдела ЭСК входят отделы из департамента логистики и департамента экономики и финансов (полный функционал отделов перечислен на странице выше).

В задачи департамента логистики входит: обеспечение сохранности груза сотрудником склада, планирование транспортировки груза сотрудником отдела логистики, транспортно-экспедиционные услуги сотрудником отдела транспортировки.

В задачи департамента экономики и финансов входит: оформление бухгалтерской документации (при приеме/выдачи или хранении груза) кассиром-операционистом службы оформления.

## Анализ методов обработки информации на предприятии.

В рассматриваемой транспортной компании обработка информации автоматизирована, у каждого сотрудника имеется рабочее место, оборудованное компьютером. Сотрудники работают в ERP системе, отражая через объекты оной информацию о движении бизнес-процессов предприятия. Взаимоотношения рассматриваемой компании и контрагентов отражается в виде расчетных документов в ERP, а заключение договоров и разрешение разногласий происходит через программу 1С Документооборот, в которой также происходит обработка внутренних документов компании.

Основным процессом работы является приёмка и выдача грузов. Клиенты, приходя в компанию, попадают на склад, где их встречает приемосдатчик. Приёмосдатчик оформляет груз, используя мобильное приложение на базе 1С, взвешивает, замеряет габариты груза и производит фотофиксацию груза. Данные отражаются в ERP программе, с которой приложения приёмосдатчика обменивается информацией по сети. По итогу оформления груза, клиент получает документы, подтверждающие прием груза транспортной компанией. С этими документами он проходит в зал оформления. В котором встает в очередь, получая талон с индивидуальным номером, для продолжения оформления перевозки груза. Система электронной очереди распределит выданный талон на ближайшее свободное окно, в котором оператор сможет продолжить оформление перевозки, используя штрихкод на документах о приёмке груза. В рамках дальнейшего оформления, клиент регистрируется в системе по паспортным данным, если он ранее не пользовался услугами. В ERP программе создается квитанция, главный документ в котором отражаются все услуги, связанные с перевозкой грузов клиента. Оформленная квитанция выдается клиенту, после чего оформление отправки полностью завершено.

Передвижение груза внутри компании отражается по документу «квитанция» и доступно для просмотра клиентом на сайте компании в личном кабинете или мобильном приложении. При выдаче доставленного груза достаточно предъявить номер квитанции и документы, соответствующие указанным ранее при оформлении.

Данный процесс основной со стороны клиента, так как именно этот бизнес-процесс позволяет клиенту получить услугу, за которой он пришел. Приложение, разрабатываемое в данной бакалаврской работе, участвует в описанном бизнес-процессе, заменяя существующее решение.

## Функциональные требования к разрабатываемой системе.

В рассматриваемой компании внедрена система электронных очередей собственной разработки в составе системы «ERP». Вся работа системы осуществляется в среде 1С: Предприятие. Серверная обработка служб электронной очереди (распределение талонов, хранение информации о посетителях, составление отчетов для отображения на табло, обработка создания талонов) работает в корпоративной информационной системе «ERP», что влечет за собой использование дополнительных лицензий при подключении по Web-интерфейсу, запуску фоновых заданий. Табло для отображения состояния электронной очереди работает на ОС «Windows» и запускает клиентский сеанс «1С», в котором отображается форма с распределенными талонами. К минусам такого устройства табло относится наличие дополнительного компьютера, к которому подключается экран. Киоски выдачи талонов работают аналогично табло и требуют дополнительного лицензирования операционной системы и клиентских лицензий «1С» для своей работы.

Такая схема работы электронной очереди является не эффективной в экономическом плане. Еще одной проблемой функционирования информационной системы электронной очереди на базе корпоративной информационной системы «ERP» является низкая стабильность работы в целом всей системы «ERP». В связи с периодическими сбоями в работе программ возникают большие задержки очередей, что понижает лояльность клиентов к компании.

Рассматривая все вышеперечисленное, совместно с руководством компании мы пришли к решению о замене существующей системы электронных очередей на более оптимизированную и решающую выявленные проблемы в существующей системе. Требования по оптимизации процессов компании сформулированы и перечислены ниже.

### Требования с экономической стороны, следующие:

* Возможность использовать свободно распространяемые дистрибутивы операционных систем.
* Информационная система работает без дополнительного лицензирования среды исполнения.
* Низкие требования к мощности компьютеров пользователей системы.

### Архитектурные требования сформулированы следующим образом:

* Информационная система учета посетителей транспортной компании должна работать как независимый сервис, то есть при сбоях работы в основной системе «ERP», сервис продолжает работу самостоятельно.
* Сервер электронной очереди должен иметь программный интерфейс «API» для интеграции с другими системами на предприятии, в том числе с основной системе «ERP».

### Функциональные требования к базе данных, следующие:

* База данных должна отражать всю информацию о выданных талонах, сотрудниках и установленных в офисах дополнительных устройствах;
* В базе данных должна быть информация о существующих услугах; информация о привязке этих услуг к кассирам.
* Должна быть возможность вносить изменения в данные и пополнения новыми данными.
* Хранение информации должно осуществляться на серверах внутри компании, для недопущения доступа к статистике посещений третьих лиц.
* В программе должна присутствовать функции поиска, выполнения определенных запросов, формирования отчетов.

### Помимо этого, необходимо создать программный продукт, отвечающий следующим требованиям:

* разделенный доступ по ролям;
* возможность добавлять, удалять, редактировать данные о сотрудниках;
* возможность добавлять, удалять, редактировать данные о услугах кассиров;
* возможность изменять состояния обслуживания талона;
* возможность распечатать талон.

## Анализ средств и инструментов для разработки.

Первое на что необходимо обратить внимание при выборе способа реализации информационной системы на готовые решения сторонних разработчиков. На рынке имеются готовые модульные продукты и облачные сервисы. Рассмотрим по одному решению из этих способов внедрения.

Примером модульного внедрения рассматривается решение от «BM-GROUP Фабрика решений» [1]. Данная компания предлагает внедрение следующими модулями, продавая как программные решения, так и устройства. К предлагаемым модулям относятся:

* Базовый комплект ПО «Сервер очереди» (Стоимость 53000 рублей). Включает по одной лицензии «Регистратор», «Администратор сервера», «Зал оповещения», «Администратор зала», «Электронное табло». Следовательно, в дальнейшем на каждое подключаемое устройство необходимо приобретать лицензию.
* АРМ Оператора (Стоимость 4900 рублей). Требует лицензии для работы операторов.

В качестве устройств для обеспечения работы системы электронной очереди предлагается к приобретению:

* Сенсорный терминал Line Mini 21,5" (Стоимость: 161000 рублей).
* Информационное табло 50 дюймов (Стоимость: 26000 рублей).

В целом решение данной компании отвечает основным сформулированным требованиям к программному продукту электронной очереди, но данной решение требует приобретения дополнительных лицензий по мере увеличения подключенных офисов к системе, что не удовлетворят экономическим требованиям заказчика. Дополнительно «BM-GROUP Фабрика решений» не афиширует способов интеграции с другими программными продуктами и изменения дизайна.

В качестве облачного решения рассмотрим молодую компанию «BR Systems» [2]. Решение этой компании находится в статусе «Бетта», но довольно интересно. Внедрение предлагается абсолютно бесплатно. Система устроена через чтение QR-кодов встроенными возможностями мобильных устройств посетителей, при этом не требуется печати талонов, что в современных реалиях очень экологично. После сканирования QR-кода открывается веб-страница на которой отображен номер в очереди или предлагается записаться на будущее время [3].

Данное решение позволяет быстро внедряться в ново подключённых офисах к системе, не требует приобретения дорогостоящих устройств для первичной регистрации посетителей, но не позволяет хранить данные о посещениях на серверах компании заказчика. Самое главное интеграцию с сторонними программными продуктами и персональный дизайн компания «BR Systems» обсуждает индивидуально за дополнительные средства.

Таким образом самостоятельная разработка выглядит привлекательно. Для разработки необходимо определить СУБД для управления базой данных сервиса, язык программирования для разработки серверных функций системы и устройства для размещения в офисах обслуживания.

В качестве СУБД можно рассмотреть наиболее популярные свободно распространяемые решения PostgreSQL и MySQL. Распространяются бесплатно на условиях лицензий распространения программного обеспечения с открытом кодом GNU GPL [4] (MySQL) и PostgreSQL License [5] (PostgreSQL). СУБД PostgeSQL, считается более стабильной при построении баз данных с большим количеством хранимых данных, какой может стать разрабатываемая информационная система в данной выпускной квалификационной работе. Также PostgeSQL имеет оптимизированную поддержку вложенных запросов, полное индексирование, обеспечивающее работу полнотекстового поиска, типа данных uuid позволяющий создавать таблица в которых будет использоваться в качестве первичного ключа, что позволит хранить очень большое количество данных в сравнении с другими доступными типами для первичных ключей [6], в отличии от решения MySQL.

MySQL считается одной из самых быстрых СУБД, так же очень привлекательна со стороны администрирования, в связи с тем, что разработчики этой СУБД заложили мало настраиваемых параметров. Такой подход позволяет разворачивать базы данных без сложных настроек работы СУБД, что отличено подходит для небольших приложений. Один из больших минусов — это отсутствие транзакций в системе, что может нарушить целостность данных в случае ошибки, при последовательном выполнении связанных операций.

Для разработки программного продукта рассматривается язык программирования Java 11. Это очень популярная технология разработки серверных решений, занимает третье место в мире по популярности по результатам ежегодного отчета State of the Octoverse 2020 [7]. Является лидером в мире при создании приложений связанными с интернетом, хоть и создавался изначально для кроссплатформенной разработки для бытовой техники [12]. Основным отличием о многих других языков программирования является кроссплатформенность Java. Можно вести как разработку, так и запуск на разных платформах (Windows, Linux, MacOS). Само приложение компилируется в исполняемы файл, для работы которого необходимо иметь только установленную JVM (виртуальная машина Java), входящую в состав Java JRE (Среда исполнения Java) [8]. Наличие большого количества фреймворков для решения разных рутинных задач, таких как построение ORM моделей и публикация на веб-сервере позволяют более подробно заняться проектирование качественной архитектуры приложения, не тратя много времени на создание взаимодействия с базовыми операциями в информационной системе.

Выводы

В первой главе рассмотрены организация-заказчик, в лице крупной транспортной компании. Описана структура этой компании и иерархия управления, выявлены области ответственности подразделений.

Изучена существующая система автоматизации управления электронной очередью. Выявлены проблемы в архитектуре и стабильности существующей системы. Обозначены требования к реализации продукта решающего выявленные проблемы.

Произведен анализ готовых продуктов на рынке и произведено сравнение с сформулированными требованиями. Составлен краткий обзор технологий с помощью, которые возможно приметь при собственной разработке информационной системы учета посетителей.

# Проектирование информационной системы для учета посетителей транспортной компании.

Во второй главе выпускной квалификационной работы рассматривается проектирование информационной системы клиент-серверного приложения для учета посетителей транспортной компании. Будут составлены EER модель и физическая модель базы данных, а также спроектирован интерфейс пользователей.

## Анализ принципов работы СУБД.

Вместе с развитием компьютеров росло количество информации, обрабатываемой ими. Этот рост привел к развитию новых способов хранения информации и появлению понятия системы управления базами данных – СУБД в 1960х. годах. База данных — это поименованный комплекс структурированных данных входящих в одну предметную область. СУБД – это комплекс программных и языковых методов, обеспечивающих создание баз данных, сохранность их в актуальном состоянии и осуществления поиска в них требуемой информации [9].

По мере развития систем управления базами данных можно выделить три основных этапа.

Первый этап можно назвать *иерархические и сетевые* системы управления базами данных. На этом этапе СУБД опирались на иерархическую и сетевую модель данных. В это время доминировали большие вычислительные машины, которые в совокупности с СУБД первого поколения составили аппаратнопрограммную платформу больших информационных систем. Связь данных из разных файлов происходила через физические указатели. Это означало что в том случае, когда в разных файлах хранятся логически связанные данные, а физическая связь отсутствует между файлами, то для получения выборки необходимой информации приходилось использовать низкоуровневые методы работы с файлами.

Началом второго этапа эволюции баз данных можно считать начало 70х годов. В это время выдвигались идеи подключить к обработке данных теорию множеств, реляционную алгебру и реляционное исчисление. Идея заключалась в том, что связь между данными должна устанавливаться в соответствии с их логическими взаимоотношениями. Сегодня реляционные базы данных очень широко распространены, их можно назвать стандартом СУБД.

Развитие реляционных систем привело к необходимости решать более сложные задачи, что вызвало появление объектно-ориентированных СУБД. Для таких систем управления базами данными характерно использование идей объектно-ориентированного подхода, управления распределенными данными, активного сервера данных и параллельной обработки данных.

Основными функциями СУБД являются:

* Физическое размещение в памяти данных и из описаний.
* Осуществление механизмов поиска данных.
* Обеспечение одновременных запросов разными пользователями к одним и тем же данным.
* Защита от некорректных обновлений или несанкционированного доступа.
* Обеспечение актуального состояния данных.
* Поддержание целостности и непротиворечивости данных.

В совокупности перечисленные функции баз данных объединены в парадигме ACID, сформированной в 70х годах и описывающей требования к транзакционным СУБД, которыми являются все основные на сегодня СУБД в мире.

* Atomicity (атомарность) – никакая транзакция не может быть выполнена частично.
* Consistency (согласованность) – транзакции переводят базу данных из одного корректного состояния в другое.
* Isolation (изоляция) – одновременно выполняющиеся транзакции не влияют дран на друга, а результат параллельного выполнения транзакций равен результату – последовательного.
* Durability (долговечность) – успешное завершение транзакции гарантирует, что никакое другое событие не изменит этого факта.

Операции в базах данных можно разделить на 4 типа:

* Select (выбор) – получение выборки данных из таблиц.
* Insert (запись) – добавление новых данных в таблицы.
* Remove (удаление) – удаление данных из таблиц.
* Update (обновление) – замена существующей информации в записи на новую актуальную. Работает как последовательная совокупность операций Remove и Insert.

Таким образом, базы данных созданы для надежного хранения информации и получения этой информации с учетом описанных связей. Современные решения в области СУБД обеспечивают эти возможности на высоком уровне.

## Структурный анализ входной информации системы.

Разрабатываемое приложение должно оперировать с различной информациях о сотрудниках и услугах, настройках и результатами. Всю информацию, обрабатываемую системой учета посетителей в транспортной компании, можно разделить на входную и выходную.



Рисунок 2. 1. Схема входной информации.

На рисунке 2.1 отображена схема входной информации системы. Информация о сотрудниках содержит ФИО, данные входа такие, как логин и пароль, роли доступа, доступные к работе услуги. Информация о сотрудниках может вноситься ответственными лицами такая, как информация об услугах и доступные роли, или автоматически, такая как ФИО и логин, так как при авторизации пользователей система проверяет учетные данные в корпоративной базе Active directory и в случае успеха сохраняет данные о пользователе в системе.

Офисы в приложении учета посетителей представлены как способ разделения на разные электронные очереди, и являются обязательными к заполнению у пользователей и дополнительных устройств для успешного функционирования. При настройке информации об офисе заполняется общий перечень оказываемых услуг на конкретном месте, который будет доступен для выбора на киосках в зале обслуживания посетителей.

Устройства информационного табло, это устройства с большими экранами для отображения приглашенных посетителей в окна. Наделены вычислительными устройствами, благодаря чему могут взаимодействовать с веб-интерфейсом сервера информационной системы учета. Для успешного функционирования данные устройства автоматически регистрируются по IP адресам в системе, после чего администратор заполнят настройки такие, как количество отображаемых строк, ориентация отрисовки страницы табло, офис в котором установлено тало и другие.

Устройства киосков выдачи талонов «встречают» посетителей. Посетители, подходя к киоску, выбирают необходимую услугу и получают распечатанный талон со своим номером в очереди. Работа киосков устроена аналогично информационному табло. Данные устройства регистрируются по IP адресам. Для настройки требуется указание офиса, в котором установлено описываемое устройство, по информации указанной в настройках офиса будет получен перечень услуг и выведен на экране.

Услуги, хранящиеся в базе данных, описывают характер действий операторов информационной системы, такие как выдача грузов или получение грузов. Указание услуг позволяет разделять клиентов по уровням компетенций, например не все операторы могут быть допущены к приему оплат.

## Структурный анализ выходной информации системы.

В качестве выходной информации системы, требуется рассмотреть сущности получаемые в ходе работы, а также возможное изменение состояний объектов входной информации.

Главной сущностью разрабатываемой информационной системы является талон, создаваемый при выборе клиентов желаемой услуги. Талон отражает в своих полях время обслуживания и ожидания, позволяет анализировать занятость сотрудников и нагрузку офисов.



Рисунок 2. 2. Схема выходных данных.

Главной сущностью разрабатываемой информационной системы является талон, создаваемый при выборе клиентов желаемой услуги. Талон отражает в своих полях время обслуживания и ожидания, позволяет анализировать занятость сотрудников и нагрузку офисов.

Для понимания состояния очереди собирается информация о текущих талонах и занятости сотрудников. По таким данным можно понять есть ли перегрузка клиентами в данный момент или нет. Если в очереди слишком много людей, то старший кассир может принять решение о переключении деятельности свои подчиненных на обслуживание клиентов, чтобы больше рабочих окон принимало посетителей.

У сотрудников в течении работы меняются статусы. Такие статусы как «Рабочее время» или «Личное время» инициализируются пользователями вручную. Находясь в статусах «Нерабочее время» или «Личное время» сотрудникам не назначаются талоны, при этом позволяют понимать на месте сотрудник или нет. Статус «Рабочее время» определяет состояние сотрудника, в котором на него можно назначать талоны посетителей из очереди. Остальные статусы сотрудников назначаются в ходе взаимодействия с интерфейсом при обслуживании клиента.

## Выбор аппаратных и программных средств для разработки системы.

Разрабатываемая информационная система в данной бакалаврской работе подразумевает использование маломощных клиентских устройств, для этого основной пользовательский интерфейс разрабатывается с использование «HTML» и «JavaScript», таким образом интерфейс будет «легким» для устройств и кроссплатформенным. На момент запуска разрабатываемой системы предполагается низкая нагрузка на сервер, так как планируется пилотная работа в малом количестве офисов.

Для хранения данных информационной системы выбрана СУБД PostgreSQL. Данная СУБД имеет множество настроек, с помощью которых в дальнейшем можно будет оптимизировать, при необходимости, работу с данными. Данная СУБД полностью соответствует парадигме ACID (Atomicity – атомарность, Consistency – согласованность, Isolation – изоляция, Durability – долговечность), что повышает защиту целостности данных в базе [11]. PostgreSQL поддерживает хранимые процедуры на большом количестве используемых сегодня языков программирования, таких как Java, Perl, Python, Ruby, Tcl, C/C++, и собственном PL/pgSQL, аналогичном Oracle's PL/SQL. Для обеспечения целостности данных в системе используются такие средства, как составные первичные ключи, внешние ключи с поддержкой запрета и каскадирования изменений/удалений, проверку ограничений (constraints), ограничения уникальности и ограничения на непустые значения. Таким образом, PostgreSQL это не только мощная система управления базами данных, позволяющая обеспечить деятельность организации, но и платформа разработки приложений, требующих использования реляционной СУБД.

В качестве языка программирования серверной части информационной системы учета посетителей, принято решение использовать Java 11. При использовании сборки «Open JDK» [18] в коммерческих решениях, не требуется дополнительного лицензирования. В экосистему Java входят различные фреймворки, повышающие качество программного кода и уменьшающие время, требуемое для разработки. В приложении, разрабатываемом в данной выпускной квалификационной работе, используется следующие фреймворки:

* **Spring Boot** [19]. Данный фреймворк является управляющим для всего набора инструментов Spring. С помощью данного фреймворка можно сконфигурировать основные используемые инструменты Spring (Data, Security, Web MVC), также он управляет работой приложения и созданием объектов в памяти.
* **Hibernate** [20]. Данный фреймворк является реализацией ORM модели, позволяет связать объекты, создаваемые при разработке, со всеми полями и ссылками на другие объекты с базой данных. Упрощает создание базовых запросов входящих в CRUD.
* **Thymeleaf** [21]. Используется при динамической загрузке веб страниц. Используя шаблоны HTML-страниц, позволяет заполнять их данными из памяти приложения.

Для публикации веб-сервера используется встроенный в «Spring Web MVC» сервер «Tomcat». Страницы HTML, созданные с помощью Thymeleaf, заполняются начальными параметрами для отрисовки интерфейса, но остаются без возможности динамического обновления данных с сервера. Для решения этой проблемы и используется язык программирования «JavaScript» с фреймворком «Ajax» для асинхронного обращения к серверу и отрисовки полученных данных. Таким образом страницы будут обновлять только необходимые в тот или иной момент времени данные. «JavaScript» позволяет обращаться к свойствам окна браузера таким, как разрешение экрана, размеры окна в текущий момент в пикселях. Эти данные позволяют менять размеры элементов интерфейса в зависимости от разрешения экрана клиентского компьютера и от того какой размер рабочей области браузера в данный момент используется.

Серверная часть информационной системы учета посетителей будет запущена на виртуальной машине, далее Сервер. Характеристиками представлены в таблице 2.1. На сервере будет установлена «JVM» для работы программной части системы учета посетителей, а также «PostgreSQL 12», на котором будет создана и настроена база данный разрабатываемой системы.

Талица 2. 1. Характеристики виртуальной машины сервера.

|  |  |
| --- | --- |
| Процессор | 2 Ядра |
| Оперативная память | 2 ГБ |
| Жесткий диск | 20 ГБ |
| Операционная система | CentOS 8 |

Виртуальная машина будет запущена на сервере виртуализации, установленном в собственном центре обработки данных компании. Характеристики сервера представлены в таблице 2.2.

Талица 2. 2. Характеристики сервера виртуализации.

|  |  |
| --- | --- |
| Форм-фактор корпуса | 2 Units |
| Процессор | 2 шт. Intel® Xeon Gold 5220  (18 ядер / 36 потоков, базовая частота 2.20GHz, макс. 3.90 GHz, 24.75MB SmartCache) |
| Оперативная память | 8 шт. 32GB DDR4 2666MHz ECC REG |
| Система хранения | Контроллер LSI 9361-8i, 1Gb, CacheVault c поддержкой RAID 1/5/6/10 |
| Жесткий диск | 8 шт.× 1200GB SAS 10000rpm Hot Swap |
| Блок питания | 2 шт. 740 ватт |

В транспортной компании все рабочие места оснащены компьютерами для обеспечения потребностей сотрудников в выполнении своих профессиональных обязанностей. Типичные характеристики компьютера сотрудника, работающего вне отдела разработки представлены в таблице 2.3. Такой компьютер позволяет взаимодействовать с корпоративной «ERP» системой, работающей по принципу терминального сервера, запуск программы происходит через протокол «RDP» для приложений, а также позволяет работать в «Microsoft Office» и пользоваться корпоративными сервисами в веб-браузере «Google Chrome».

Талица 2. 3. Характеристики офисного компьютера.

|  |  |
| --- | --- |
| Процессор | Intel Celeron G1610 Ivy Bridge (2600MHz, LGA1155, L3 2048Kb) |
| Оперативная память | 2 шт. 2 ГБ DDR3 1300Mhz |
| Видеокарта | Интегрированная |
| Жесткий диск | WD 500 ГБ |
| Блок питания | 350 ватт |
| Монитор | Aser 19” разрешение 1366х768 |

Для разработки программного обеспечения требуется больше мощности от компьютера чем при обычной офисной работе, так как необходимо поддерживать работу окружение разработки: базы данных разработчика, дополнительное программное обеспечения для анализа и проектирования, программная среда разработки (IDE). Для комфортной работы разработчиков в компании используются компьютеры с оборудованием, представленным в таблице 2.4.

Талица 2. 4. Характеристики компьютера разработчика.

|  |  |
| --- | --- |
| Процессор | Intel Core i7-7700 4х3.6 GHz |
| Оперативная память | 4 шт. 4 ГБ DDR3 1600Mhz |
| Видеокарта | Интегрированная |
| Жесткий диск | WD 500 ГБ |
| Блок питания | 500 ватт |
| Монитор | 2 шт. AOC 24”  разрешение 1920х1080 |

Разработка программной части ведется в среде программирования или по-другому IDE (Integrated Development Environment) «Intelij IDEA» [22] от компании «JetBrains». Данное программное обеспечение завоевало популярность своим удобным интерфейсом и поддерживаемыми технологиями. Эта среда поддерживает синтаксис многих популярных языков программирования и описания данных, включая Java, JavaScript, CSS, HTNL, XML, JSON, SQL. Присутствует встроенная поддержка администрирования популярных баз данных таких, как PostgreSQL, MySQL, MsSQL и других, это позволит просматривать состояние и структуры базы данных, с которой ведем разработку, без необходимости переключаться на другие программы. В этой IDE имеется поддержка фреймворков для Java, позволяющая контролировать правильность настройки и вызова методов в этих фреймворках.

Для работы информационного табло используется телевизор Samsung UE43J5202 43 дюйма с функцией перехода по URL при запуске. Используя такой экран можно отказаться от дополнительных компьютеров или микрокомпьютеров для отображения интерфейса в веб страницах.

Посетители непосредственно взаимодействую с киосками выдачи талонов. У таких устройств должен быть качественный сенсорный экран, для комфортного взаимодействия с интерфейсом и принтер. В компании закупаются сенсорные терминалы типа К-28. В таких терминалах установлен миникомпьютер «Intel Nettop», характеристики представлены в таблице 2.5, сенсорный экран 21.5 дюйма, термальный чековый принтер фирмы «HSPOS» HS-K33 с шириной ленты 80 миллиметров. Подключается принтер по интерфейсу USB с эмуляцией COM-порта, необходимо для более гибкой отправки команд. У стандартных драйверов принтера есть ограничения по чтению статусов печати. В качестве операционной системы установлена «Ubuntu Desktop 16.04 lts», выбор столь старой операционной системы обусловлен работой с сенсорными экранами. В операционных системах Ubuntu более новых версий интерфейс сильно интегрирован с жестами которые можно делать с помощью сенсорных дисплеев, что в свою очередь вносить потенциальные опасности в работу приложения. С помощью таких жестов можно свернуть открытое приложение и получить доступ к операционной системе.

Талица 2. 5. Характеристики Intel Nettop.

|  |  |
| --- | --- |
| Процессор | Celeron® J4005 |
| Оперативная память | DDR4 4 Гб |
| Жесткий диск | 60 Гб |

## Даталогическое и инфологическое проектирование ИС.

При проектировании информационной системы необходимо начинать с описания концептуальной модели модулей разрабатываемого приложения. Концептуальная модель является абстрактной моделью, позволяющей определить структуру моделируемой системы, ее элементы, свойства и причинно-следственные связи [13]. При разработке концептуальной модели необходимо описать сущности предметной области, в которой функционирует, разрабатываемая информационная система. Затем атрибуты этих сущностей. Атрибутами является набор свойств, описывающий конкретные объекты реального объединяемых в одну сущность. Далее в таблице 2.6 описаны представления сущностей в разрабатываемой базе данных, соответствующих предметной области информационной системы.

|  |  |
| --- | --- |
| **Сущность** | **Описание** |
| Пользователь | Информация о пользователе, он же сотрудник. Его принадлежности к офису и данные входа. |
| Талон | Является основной сущностью системы. Описывает время прихода и обслуживания клиента, имя клиента, если указано, документы, оформленные во время обслуживания клиента по данному талону. |
| Киоски выдачи талонов | Сущность хранит описание и настройки дополнительных устройств – киосков выдачи талонов, разрабатываемой системы. |
| Информационное табло | Описание и настройки дополнительных устройств – информационных табло. |
| Статусы пользователей | Информация о статусе пользователя привязанная ко времени. |
| Талоны пользователей | Сущность отражает связь талонов к пользователям работающим с ними с привязкой ко времени. |
| Офис | Сущность позволяющая разграничить работу системы на отдельные очереди, логически связана с офисом, в котором происходит обслуживание клиентов. |
| Услуги | Сущность разграничивает операторов по компетенциям. |
| Меню киосков | В данной сущности хранится структура меню, отображаемых на киосках выдачи талонов. |
| Контрагенты | Сущность является описанием клиента. |

Талица 2. 7. Список сущностей информационной системы

Теперь выделены основные абстракции, описывающие предметную область в которой будет функционировать разрабатываемая информационная система учета посетителей. Далее необходимо выделить требуемые свойства сущностей, для описания их состояния или особенностей, определить тип данных для значений этих свойств, которые они могут принимать. Такие свойства называются атрибутами. В таблице 2.7 представлено описание атрибутов, описывающих в необходимом объеме сущности предметной области, используемые в разрабатываемой информационной системе.

|  |  |
| --- | --- |
| Поле | Тип |
| Таблица "Контрагенты" | |
| id\_контрагента | Уникальный идентификатор |
| наименование | Текстовое |
| Таблица "Киоски" | |
| id\_киоска | числовое |
| наименование | Текстовое |
| id\_офиса | числовое |
| IP адрес | Текстовое |
| Комментарий | Текстовое |
| Устройство активно | Булево |
| Тестовое устройство | Булево |
| Сообщение об ошибке | Текстовое |
| Статус устройства | Текстовое |
| Таблица "Меню киосков" | |
| id\_элемента | числовое |
| имя | Текстовое |
| Тип элемента | Текстовое |
| id\_дочернего элемента 1го уровня | числовое |
| id\_дочернего элемента 2го уровня | числовое |
| id\_дочернего элемента 3го уровня | числовое |
| Таблица "Сотрудники" | |
| id\_сотрудника | числовое |
| ФИО | Текстовое |
| логин | Текстовое |
| пароль | Текстовое |
| id\_офиса | числовое |
| активный | Булево |
| Таблица "Талоны сотрудников" | |
| id\_записи | Уникальный идентификатор |
| дата | Дата |
| Тип распределения | Текстовое |
| id\_сотрудника | числовое |
| id\_талона | Уникальный идентификатор |
| Таблица "Статусы сотрудников" | |
| id\_записи | числовое |
| Рабочее окно | числовое |
| дата | Дата |
| статус | Текстовое |
| id\_сотрудника | числовое |
| Таблица "Офисы" | |
| id\_офиса | Числовое |
| наименование | Текстовое |
| id\_меню киоска | Числовое |
| активный | Булево |
| Таблица "Табло" | |
| id\_табло | числовое |
| наименование | Текстовое |
| id\_офиса | числовое |
| IP адрес | Текстовое |
| Комментарий | Текстовое |
| Устройство активно | Булево |
| Тестовое устройство | Булево |
| Сообщение об ошибке | Текстовое |
| Статус устройства | Текстовое |
| Количество строк на экране | числовое |
| Ориентация | числовое |
| Флаг для обновления страницы | Булево |
| Версия интерфейса | числовое |
| Таблица "Талоны" | |
| id\_талона | Уникальный идентификатор |
| Комментарий | Текстовое |
| дата создания | Дата |
| дата печати | Дата |
| дата распределения | Дата |
| дата начала обслуживания | Дата |
| дата окончания обслуживания | Дата |
| номер талона | Текстовое |
| статус | Текстовое |
| id\_офиса | числовое |
| id\_услуги | числовое |
| id\_контрагента | числовое |
| Таблица "Услуги" | |
| id\_услуги | числовое |
| наименование | Текстовое |
| приоритет | числовое |
| Доступный статус | числовое |
| Доступна пауза | Булево |
| Показывать на табло | Булево |
| Префикс | Текстовое |

Талица 2. 8. Модель сущностей с атрибутами, описывающих их.

При проектировании информационных систем используют подход основанный на построении «Диаграмм прецедентов» [15], такие диаграммы отражают действия выполняемые в программе, но не отражают внутреннего устройства классов и методов. Благодаря этому такие диаграммы позволяют определить перечень функций и действий, которые должна выполнять получившаяся в ходе разработки информационная система. При построении таких диаграмм формируют действия, как задачи, называемые «прецедентами», и устанавливают связи между такими прецедентами и актёрами взаимодействующими с получаемой информационной системой. «Диаграммы прецедентов» или «диаграмма вариантов использования» позволяет точно определить понимания требований к действиям в системе и разграничению уровней доступа к действиям, за счет упрощенного графического представления на языке UML [15]. Такие схемы понятны как разработчикам так пользователям для которых создается информационная система, что в свою очередь позволяет избежать противоречий при понимании требований к разработке системы. На рисунке 2.3 изображена диаграмма прецедентов, разрабатываемой информационной системы, с использование графического языка UML.



Рисунок 2. 3. Диаграмма прецедентов системы электронной очереди.

Пользователей разрабатываемой информационной системы в данной выпускной работе можно разделить на 4 типа:

* *Администраторы.* Имеют доступ ко всему функционалу приложения, могут настраивать пользователей, вспомогательные устройства системы и саму работу системы учета посетителей.
* *Старшие операторы.* Имеют доступ к просмотру состояния очереди в офисе, могут настраивать перечень оказываемых услуг, а подчиненных операторов, изменять их статус работы в случае необходимости, формировать отчетность о исторических данных. Наделены всеми возможностями операторов, стоящих младше по подчиненности.
* *Операторы.* Ведут непосредственное оформление клиентов, при этом изменяют статусы обслуживания по талону в рабочем месте.
* *Клиенты.* Клиентам доступно взаимодействие только с киоском выдачи талонов и наблюдения своего положения в очереди по информационному табло.

Перечисленные типы пользователей на языке UML называются актерами, они отражены в диаграмме на 2.3. Сформировав основные задачи рассматриваемой информационной системы учета посетителей, можно поделить системы на функциональные модули, обеспечивающие работу системы путем взаимного обращения к методам друг друга. Такая концептуальная модель функциональных элементов разрабатываемой, в данной выпускной квалификационной работе, системы учета посетителей в транспортной компании изображена на рисунке 2.4.

Разрабатываемая система является многоточечным клиент-серверным приложением, так как в системе имеются сервер базы данных, сервер приложений и клиентские устройства различных типов применения, при это клиентские устройства на прямую не взаимодействуют с сервером баз данных. Далее перечисляются функциональные модули системы.

### База данных.

База данных используется для хранения информации системы. На поля в таблицах наложены ограничения такие, проверяющие данные при исполнении транзакций на заполненность, тип данных, длину информации, это обеспечивает правильность внесения информации – целостность базы данных.



Рисунок 2. 4. Концептуальная модель разрабатываемой информационной модели.

### Серверное приложение.

С базой данных обменивается информацией только серверное приложение, которое производит обработку входной и выходной информации, контролирует раздельный доступ к интерфейсу и данным. Приложение имеет программный интерфейс «API» для интеграции с другими информационными системами такими, как «ERP» на базе 1С, также программный интерфейс позволит скриптам на «JavaScript» асинхронно получать данные и динамически обновлять информацию на странице. Серверное приложение публикует HTML-страницы, составляющие интерфейс пользователя в браузере, формирует задание на печать для сформированных талонов и распределяет их на операторов. Отражая изменения объектов в базе данных для сохранения их и хранения истории изменений об объектах.

### Компьютер сотрудника.

Работу сотрудников с разрабатываемым приложений можно разделить на два типа.

Первые работают в корпоративной системе «ERP», обслуживая клиентов, отражают информацию о клиенте и его грузах. Для удобства пользователь, чтобы избежать действий в окнах разных программ, разработано «рабочее место оператора электронной очереди». Это рабочее место отображает информацию о текущем состоянии оператора: статус оператора, назначенный талон, талоны на паузе. Для этого взаимодействует с интерфейсом API сервера.

Вторые взаимодействуют с разрабатываемой системой по средством браузера. В браузере на страницах пользователи формируют отчеты о состоянии очереди в том или ином офисе, формируют отчеты о исторических данных работы очередей. Также через браузер происходит администрирование информационной системы учета посетителей. На случай сбоя работы корпоративной системы «ERP» веб-интерфейсе существует рабочее место оператора, выполняющее весь необходимый функционал. Таким образом посетители всегда смогут обслуживаться по электронной очереди и не будут ощущать повышенного дискомфорта.

### Киоск выдачи талонов.

Данные устройства также обмениваются информацией с сервером по программному интерфейсу API через веб соединение. Для устройств этого типа, разработана HTML страница со скриптами на «JavaScript», загружаемая с сервера. Скрипты на странице периодически получают информацию об оказываемых услугах в текущем офисе. В случае отсутствия связи с сервером разрабатываемой информационной системы, переводят страницу в режим ожидания соединения, показывая сообщение посетителям о временных неполадках в работе системы. При выборе клиентом услуги, скрипт отправляет эту информацию на сервер системы учета посетителей. Сервер в свою очередь отправляет команду печати обратно на кисок, но с некоторым отличием. В связи с тем, что принтер, установленный в киоске, не дает достоверной информации о состоянии печати, разработан дополнительный необходимо разработать дополнительный драйвер, принимающий команды на печать по сети и с помощью «ESC/POS» протокола выполнял и контролировал печать. Драйвер должен быть максимально упрощен по функционалу, должен принимать на вход массив байт, соответствующий протоколу «ESC/POS» обмена с принтером, и проверяющий по мере выполнения печати состояние частей принтера на ошибку, таким образом возможно сделать выводы об успехе печати, возвращать на сервер необходимо информацию о статусах принтера. Таким образом, сервер будет делать вывод об успехе печати талона и добавлять в очередь только те талоны, у которых во время печати не возникло ошибок.

### Информационное табло.

Еще одно клиентское устройство, взаимодействующее с сервером по программному интерфейсу «API». Функционирование построено аналогично киоску выдачи талонов. HTML-страница с встроенными скриптами контролирует связь сервером и получает информацию о талонах помещенными в очередь или назначенными в окно. При назначении нового талона в окно, анимировано выделяет его в списке помещая на самую верхнюю строку и издает звуковое уведомление.

После описания функциональных элементов становится понятно, как будет устроена работа с информационной системой, разрабатываемой в данной выпускной квалификационной работе, из каких модулей она состоит и к каким видам пользователей должны быть доступны эти модули.

Последним этапом проектирования базы данных является построения модели связей сущностей, или по-другому ER-модели.

При проектировании модели базы данных необходимо помнить нормализации данных. Процесс проектирования базы данных с использование метода нормальных форм является итерационным и заключается в последовательном переводе отношения из первой нормальной формы в нормальную форму более высокого порядка по определенным правилам. Каждая следующая нормальная форма ограничивается определенным типом функциональных зависимостей и устранением соответствующих аномалий при выполнении операций над отношениями базы данных, а также сохранении свойств предшествующих нормальных форм [14]. Обычно достаточным является приведение к первым трем нормальным формам.

### Первая нормальная форма.

Для соответствия первой нормальной форме, все атрибуты сущности должны быть простыми. Допустимые значения атрибутов должны быть скалярными. Не должно быть повторения строк в таблице. Таким образом значения атрибутов должность атомарными, тесть хранить одно значение и не являться ни списком, ни множеством значений. Таблица должна иметь первичный ключ, из минимально необходимого количества полей.

### Вторая нормальная форма.

Отношение находится во второй начальной форме, если оно находится в первой начальной форме и каждый не ключевой атрибут неприводимо зависит от Первичного Ключа(ПК). Неприводимость означает, что в составе потенциального ключа отсутствует меньшее подмножество атрибутов, от которого можно также вывести данную функциональную зависимость.

### Третья нормальная форма.

Отношение находится в 3НФ, когда находится во 2НФ и каждый не ключевой атрибут нетранзитивно зависит от первичного ключа. Проще говоря, второе правило требует выносить все не ключевые поля, содержимое которых может относиться к нескольким записям таблицы в отдельные таблицы.

Для завершения даталогического проектирования базы данных необходимо построить модель «сущность-связь» или ER-модель.

Концептуальная ER-модель – это семантическая модель и она выступает стандартным способом, позволяющим выделить сущности и отношения между ними. С ее помощью выполняется детализация хранилищ данных.

В состав ER-диаграммы входит информация о сущностях системы, способах их взаимодействия, а также информация по идентификации объектов, которые важны для предметной области, их свойствах, атрибутах и установленных отношениях между объектами. Во многих случаях информационная модель является сложной и включает в свой состав несколько объектов. На рисунке 2.5 представлена ER-модель базы данных, разрабатываемой в данной выпускной квалификационной работе. Для понимания условных обозначений необходимо знать: каждый прямоугольник является представление сущности – таблицей базы данных, список внутри прямоугольников это атрибуты, описывающие сущность, у атрибутов указан их тип, а также слева от названия атрибутов указана роль этих атрибутов, так желтый ром обозначает *первичный ключ* (PK) и обеспечивает уникальность данных в таблице, ромбы красного цвета обозначают *вторичные ключи* (FK), такие ключи осуществляют связь между таблицами разных сущностей. Белыми ромбами помечены простые атрибуты для полноты описания сущности.

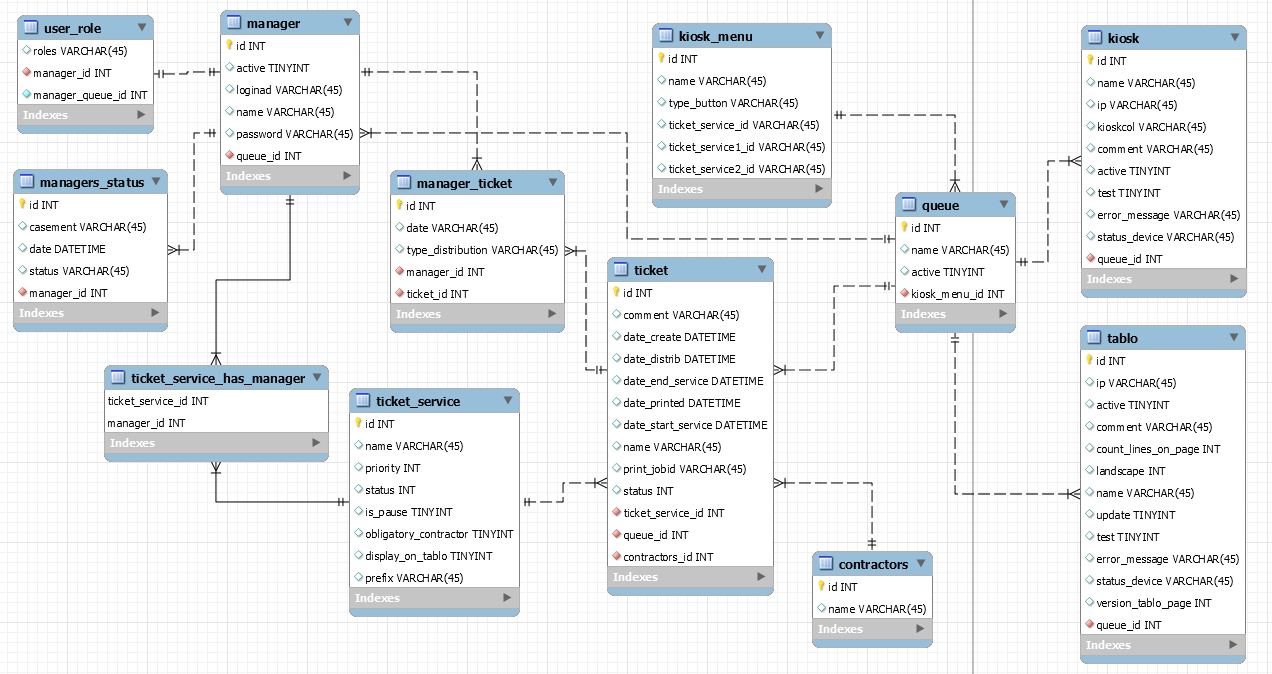


Рисунок 2. 5. ER-модель базы данных.

На этом проектирование основных элементов разрабатываемой системы можно завершить, так как теперь понятно какие сущности будут обрабатываться в информационной системе и как они связаны между собой, какие программные элементы необходимо разработать для обеспечения требований к разрабатываемой системе учета посетителей.

Выводы

По итогам второй главы получено понимание основных принципов работы с системами управления базами данных. Понимание этих принципов влияет на модель данных разрабатываемых систем. Также определен состав входных и выходных данных информационной системы, выбраны программные и аппаратные средства для разработки и внедрения продукта, разрабатываемого в данной бакалаврской работе. Проведен анализ и сформирована модель данных, обрабатываемых в данной информационной системе, определен состав отдельных элементов системы для серверной обработки и взаимодействия пользователями.

# Разработка информационной системы для учета посетителей транспортной компании.

В третьей главе данной выпускной квалификационной работы рассматриваются методы, используемые при разработке информационной системы учета посетителей, особенности реализации архитектуры информационной системы, разработка внешнего вида графического интерфейса пользователей и внедрение программного продукта.

## Разработка СУБД для информационной системы.

При разработке СУБД обычно по построенной ER-модели создается скрипт SQL, создающий структура базы данных для разрабатываемой информационной системы, но в данной бакалаврской работе можно поступить по-другому. Применение в проекте фреймворка Hibernate позволяет описывать классы объектов сущностей, используемых в программе, с использованием специальных системных аннотаций в коде, на основе которых будет сформирован скрипт, создающий структуру базы данных. Такой подход оптимально связывает описание объектов в объектно-ориентрованном представлении языка программирования с реляционным представлением в базе данных. Процесс такого связывания называют ORM (Object-relational mapping - рус. Объектно-реляционное отображение). Обычно разработка модели ORM требуют значительных затрат времени разработчика, но использование фреймворка Hibernate позволяет сократить эти затраты, что упрощает разработку новых приложений, использующих для долгосрочного хранения информации базы данных.

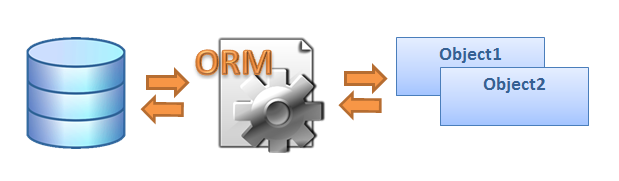


Рисунок 3. 1. Схема применения ORM. [16]

Таким образом разработка базы данных для приложения, разрабатываемого в данный выпускной квалификационной работе, сводится к описанию классов объектов сущностей, с использованием системных аннотаций фреймворка Hibernate. Для рассмотрения процесса создания такого класса в *Приложении А* представлен код класса «Manager», этот класс является описанием сущности «Пользователи». Далее описано значение аннотаций, используемых в коде класса.

### @Entity

Сам класс помечен аннотацией «@Entity», она сообщает обработчику фреймворка, что описанный далее класс являются сущностью, которую необходимо сохранять в базе данных. Использование этой аннотации накладывает некоторые ограничения на класс, создаваемый далее, одно из таких ограничений – это наличие атрибута «Id», обеспечивающего уникальность записей в получаемой таблице.

При работе с базами данных иногда необходимо использовать функционал временных таблиц. Такой функционал реализован в СУБД объектом **«Views»**, который реализован через помещение в него в него запроса к таблицам используемой базы данных. Результатом работы такого объекта является таблица с полями, описанными в запросе, как конечные данные, а обновление этой таблице обычно происходит при изменении данных, на которых строится запрос SQL. Для решения этой задачи в фреймворке Hibernate предусмотрено описание запроса, на основании которого будет формироваться таблица в базе данных и связываться с объектом, созданным в программе, работающей с этой базой данных. Для этого класс этого объекта помечается помимо аннотации «@Entity» еще двумя «@Immutable» и «@Subselect()». Первая аннотация помечает созданные в памяти объекты, родителями которых является класс, к которому приписана эта аннотация, неизменяемыми. Таким образом запрошенное состояния по таблице view-объекта базы данных, не может быть изменено, а только прочитано. Вторая аннотация передает в теле скрипт запроса, который будет выполняться при создании объекта view. Пример реализации view в разрабатываемом приложении представлен в *Приложении В*. В нем описан класс «ManagersViewStatuses», позволяющий получать текущие статусы пользователей системы учета посетителей. Запрос этого объекта view связывает данные о пользователе и текущем его статусе. Так реализовано создание таблиц в базе данных при использовании фреймворка Hibernate.

### @Id

У каждой таблицы в базе данных должен быть атрибут или атрибуты, являющиеся первичным ключом. По таким полям проверяется уникальность записей в таблицах базы данных. Свойства объекта входящие в состав первичного ключа, при разработке с использование фреймворка Hibernate, помечаются аннотацией «@Id». Тип значений в свойствах, помеченных этой аннотацией, должен поддерживаться СУБД, на которой работает база данных. Для генерации автоматических значений полей, которая обычно используется при создании полей первичных ключей в база данных, используется аннотация «@GeneratedValue()». В теле этой аннотации передаётся ссылка на стратегию формирования значений. Фреймворк Hibernate поддерживает генерацию числовых значений, а также значений уникального идентификатора – «UUID». Пример использования перечисленных аннотаций представлен в *Приложении А*.

### @NotNull

В *Приложении А* можно заметить аннотацию «@NotNull». Эта аннотация накладывает ограничение на поля класса и таблицы, получаемой на основании этого класса, а именно не позволяет сохранять незаполненные значения.

### @ElementCollection

В классе, описанном в *Приложении А*, есть поле «roles». Это поле является «списочном» и у созданных объектов на основании этого класса в данном поле можно записывать массив объектов «Role», что в свою очередь можно назвать коллекцией значений. Логически это список доступных пользователю ролей. Для хранения коллекций, записываемых в полях классов, в фреймворке Hibernate используется аннотация «@ElementCollection», сообщающая обработчику этого фреймворка о том, что необходимо преобразовывать данное поле при сохранении набор значений другой таблицы. Для описания таблицы в которой будут храниться данные этого поля используется аннотация «@CollectionTable(name = "user\_role", joinColumns = @JoinColumn(name = "user\_id"))», в теле которой передается название будущей таблицы и поле вторичного ключа для связи с изначальным объектом, использующим эти данные. Это яркий пример несоответствия представлений объектного и реляционного подхода к модели данных. У поля «roles» есть еще аннотация требующая описания – «@Enumerated(EnumType.STRING)». Эта аннотация обозначает, что поле является перечислением. Перечисления — это объекты с предопределенным списком значений, который нельзя изменить путем взаимодействия с программой. Преимуществом таких перечислений является возможность обращаться и сравнивать такие значения прямо в коде, напрямую, без получения дополнительной ссылки на объект. Применятся обычно к объектам, которые проецируя на реальный мир, не могут изменить свой набор, к таким объектам можно отнести месяцы в году, например. В теле аннотации «@Enumerated()» передается способ хранения в базе данных значений. Так как перечисления не меняют своего порядка значений в течении работы программы, то, для экономии места, можно сохранять их в виде порядкового номера, или использовать представление в виде строки, как это сделано в примере *Приложения А*.

### @ManyToOne

При описании связей сущностей, для применения их к реляционной модели данных, используются аннотации такие, как «@OneToOne», «@OneToMany», «@ManyToOne» и «@ManyToMany». Названия аннотаций соответствуют существующим типам связей в реляционных базах данных. В *Приложении А* имеется поле «queue», логически обозначающее офис, в котором работает данный сотрудник, оно помечено аннотацией «@ManyToOne», реализующим связь «многие – к одному». Смысл этой связи обозначает что у одного сотрудника может быть только одна очередь, значение которой можно выбрать из множества очередей. Для того чтобы данная связь работала, необходимо дополнительно в классе, в котором будет сохранятся значение данного поля, описать поле, в котором хранится значение первичного ключа записи, связанной с этим полем. Реализовано такое описание аннотацией «@JoinColumn()», в теле которой передается имя колонки таблицы в которой будут хранится значения ссылающиеся на другую сущность. Для понимая необходимо заметить, что функционал, реализованный в предыдущем абзаце с использованием аннотаций, описывающих коллекции, можно было реализовать используя аннотации создания связей, но в этом случае придется хранить значения перечислений в отдельной таблице, чтобы после через третью таблицу по связи «многие – ко многим» связывать данные. В связи с тем, что перечисления уже хранятся в коде приложения, использовать такой подход, с использованием связей, не оптимально.

Описание методов создание базы данных для приложения, разрабатываемого в данной выпускной квалификационной работе, можно считать достаточным. По итогу, при первом запуске приложения, Фреймворк Hibernate проверит существующую структуру базы данных на соответствие с описанной в классах и при необходимости обновит ее.

## Разработка программного обеспечения информационной системы учета посетителей.

При разработке информационной системы, необходимо понимать из каких элементов на будет состоять. Для понимания состава элементов, разрабатываемого приложения, в главе 2.5 разработана концептуальная модель функциональных элементов, изучив которую становится понятно, что реализация подразумевает наличие трех независимых программных комплексов: СУБД, сервер приложений, драйвер принтера. Создание базы данных для разрабатываемого программного продукта, рассмотрено в подглаве 3.1. В данной подглаве рассматривается разработка сервера и программы для работы с принтером.

Разработка серверного приложения происходит с использование фреймворка «Spring Boot», особенностью которого является работа с «бинами». Бин (Bean) – это объект, основанный на каком-либо классе участвующем в программе, построенной с использованием Spring Boot, жизненным циклом которого управляет описываемый фреймворк. Созданные объекта помещаются в специальный контейнер, и доступны к подключению в другие объекты, живущие внутри этого контейнера. Такой подход позволяет упростить использование функционала, существующего в разных сервисах приложения. Архитектура приложения строить на разделении классов на функциональные роли: сервисы, сущности и репозитории.

Сущности описывают объекты данных, с которыми оперирует разрабатываемое приложение.

Сервисами являются объекты в памяти, которые осуществляют операции над данными, иными словами, реализуют бизнес логику приложения.

Репозитории являются частной реализаций сервиса, отвечающей за CRUD операции над данными, находящимися в долгосрочном хранилище (Базы данных, файлы и т. д.).

В разрабатываемом приложении сервисы можно разделить на группы:

* **Репозитории.** Выполняют функции загрузки экземпляров сущностей из базы данных и записи их обратно.
* **Веб-контроллеры.** Веб-контроллеры делятся в свою очередь еще на две группы. Первые, при обращении к их методам, возвращают HTML-страницы и обеспечивают заполнение начальных данных в них. Реализованы с использованием фреймворка Thymeleaf. Вторые реализуют программный интерфейс API, для обеспечения работы динамически обновляемых HTML-страниц приложения и интеграции с другими информационными системами.
* **Безопасность.** Классы, входящие в эту группу, обеспечивают проверку авторизации пользователей, регистрацию пользователей в системе, проверку доступности функционала по ролям.
* **Бизнес логика.** Контроллеры этой группы выполняют серверную обработку основных функций, заложенных в программу. В неё входят: сервис распределения талонов на операторов, сервис формирования задания на печать и отправки его на принтеры по сети, сервис создания талонов, сервис обработки пользовательских действий в интерфейсе.

Рассмотрим некоторые основные сервисы подробнее. Основным сервисом, обеспечивающим функционирование разрабатываемого приложения, является сервис распределения талонов. Алгоритм работы этого сервиса представлен в *Приложении С*. Реализован данный сервис в классе «TicketDistribution». В классе созданы ссылки, заполняемые фреймворком Spring во время работы приложения, на репозитории необходимых сущностей: пользователи, талоны, статусы пользователей. У класса описываемого сервиса имеется публичный метод с аннотацией «@Scheduled(fixedDelay = 5 \* 1000)», эта аннотация позволяет фреймворку Spring выполнять метод по расписанию, в конкретном случае, метод будет вызываться раз в 5 секунд. Обработка происходит порциями по 100 талонов, чтобы не создавать единовременно слишком большую нагрузку на приложение.

Для авторизации пользователей применена интеграция с корпоративной системой «Active directory». Данная система является централизованным хранилищем информации о сотрудниках компании и внедрена во многих системах. В основном позволяет использовать единые учетные данные пользователей для авторизации в разных информационных системах, а также разделять доступ пользователей по правам. Интеграция с описанной системой хранения информации о пользователях, в разрабатываемом в данной выпускной квалификационной работе приложении учета посетителей, реализована в классе «AuthProvider». Алгоритм работы метода авторизации представлен в *Приложении D*.

Обязанностью любого разработчика, является составление документации к созданному продукту. Документация может полезна другим разработчикам, которые будут дорабатывать функционал созданного продукта, такой документацией могут служить комментарии в коде, описывающее назначения методов и переменных, описывающие структура возвращаемых значений методов и типы атрибутов этих методов, или документация может быть представлена проектной документацией к продукту, которая описывает процессы, выполняемые программным продуктом. Также документация может описывать внешние программные интерфейсы, для интеграции программного продукта в другие информационные системы. Пример описания программного интерфейса API для интеграции с другими системами рабочего места оператора, разрабатываемой в данной бакалаврской работе системы учета посетителей, представлена в *Приложении E*.

Поддержание документации в актуальном состоянии требует дополнительных затрат внимания и времени разработчика. Для минимизации таких затрат существуют фреймворки для автоматической генерации документации такие, как «Swagger UI» [17]. Этот фреймворк создан для автоматического поиска «REST» контроллеров в классах приложения, чтения аннотаций, комментирующих методы в этих классах и формирования динамической страницы с документацией к методам программного интерфейса API разработанного приложения. Элемент страницы сформированной этим фреймворком представлен на рисунке 3.2.

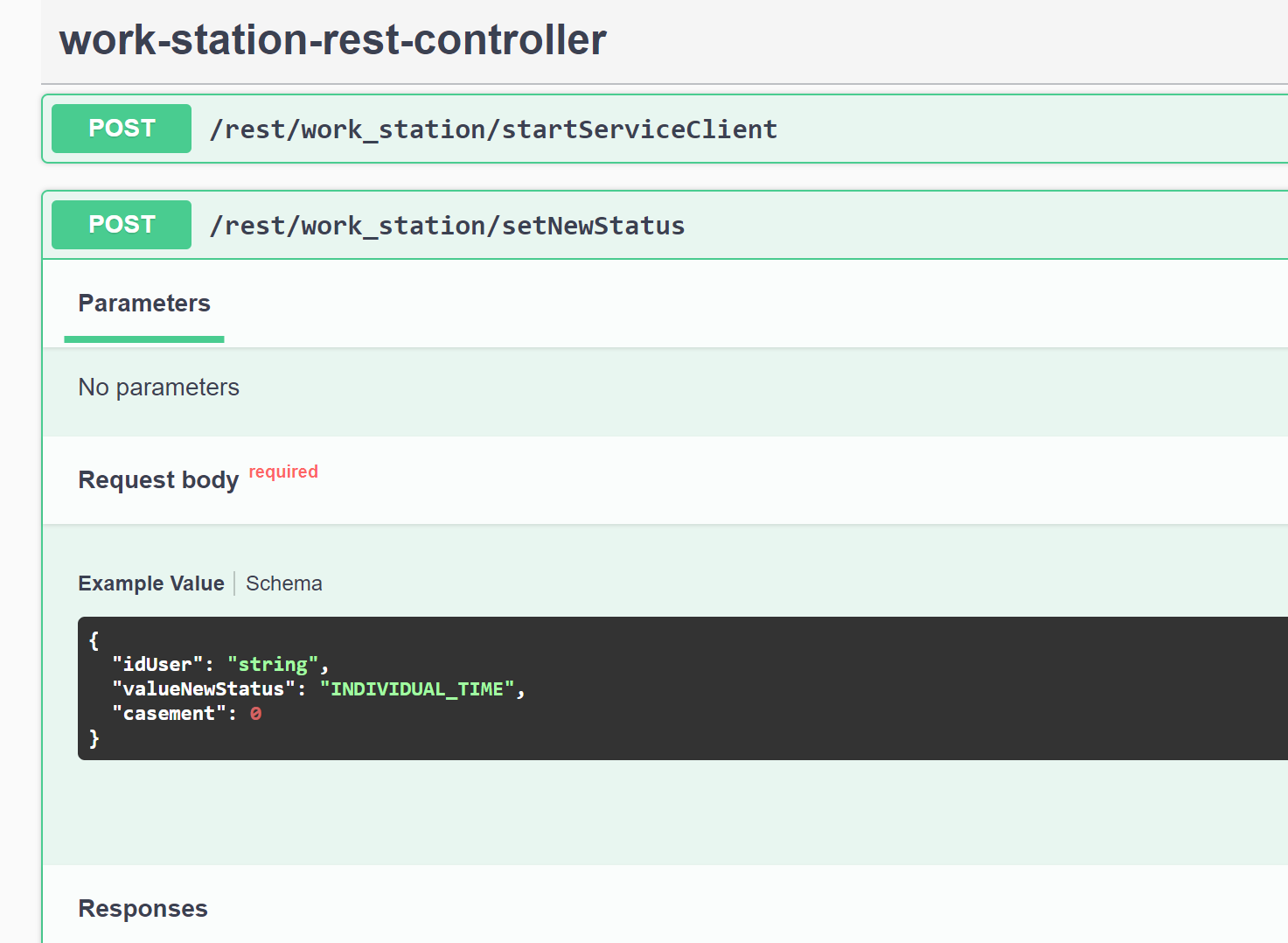


Рисунок 3. 2. Документация Swagger

## Разработка интерфейса программного продукта.

Интерфейс программы поделен по уровням доступа пользователей. Следовательно, пользователям необходимо проходить авторизацию в программе. Без авторизации пользователям в функциональный интерфейс нет доступа. На рисунке 3.3 представлена страница авторизации пользователей, доступная в веб-версии интерфейса пользователя. При подключениях к системе учета посетителей, авторизация проверяется только для сервисной

учетной записи, под которой выполняется авторизация интегрируемой информационной системы. Программный интерфейс рассчитан на передачи в параметрах идентификатора пользователя, с которым происходит в работа в текущем обращении к программному интерфейсу API системы учета

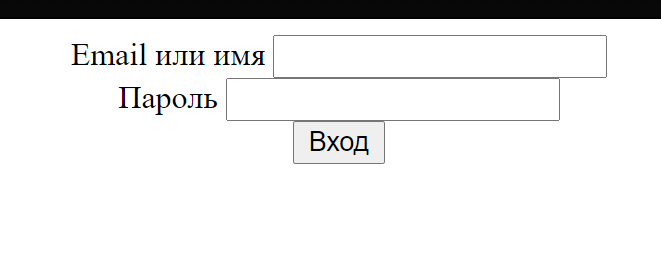


Рисунок 3. 3. Страница авторизации пользователя.

посетителей, и подразумевает проверку авторизации пользователей на стороне интегрируемой системы.

После авторизации пользователь попадает на главную страницу. Главная страница представляет собой отчет, в котором можно получить список талонов за требуемый период, с отбором по услугам, номеру талона или офиса обслуживания. Внешний вид главной страницы представлен на рисунке 3.4. При загрузке страницы период заполняется текущим днем. Отчет формируемый на данный странице доступен всем пользователям системы. Ограничения накладываются на поле «Очередь», ограничивающее отбор по офисам. Только «Администраторы» могут просматривать талоны по всем очередям, для пользователей с более низким уровнем доступа, заполняется их текущий офис, без права на изменение. На рисунке 3.5 продемонстрирован внешний вид интерфейса главной страницы для пользователя с уровнем доступа «Старший оператор».

Для всех страниц в веб-версии интерфейса разрабатываемой информационной системы свойственна отрисовка «шапки». «Шапка» представляет собой кнопки меню, открывающие функциональные разделы интерфейса, текущий рабочий статус пользователя и его имя. Для избавления от дублирования кода, шапка реализована вставкой отдельного «JavaScript» скрипта. Таким образом вставкой одной строчки на всех страницах обрисовывается единая шапка.

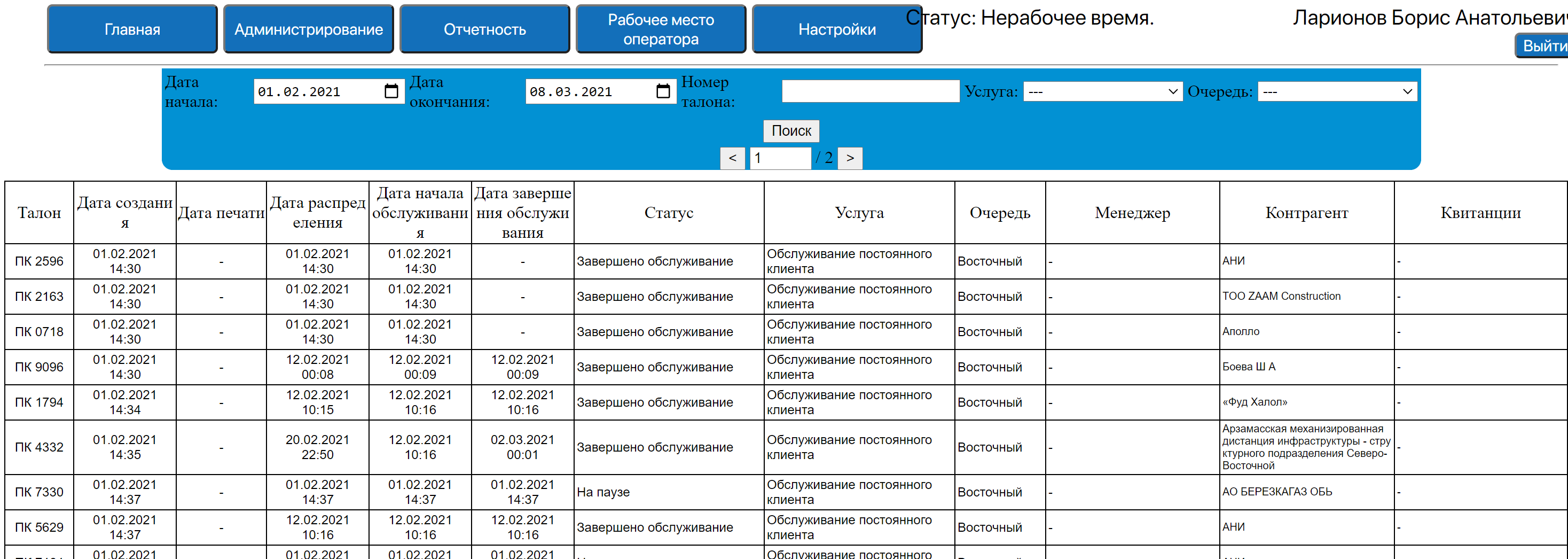


Рисунок 3. 4. Интерфейс главной страницы.

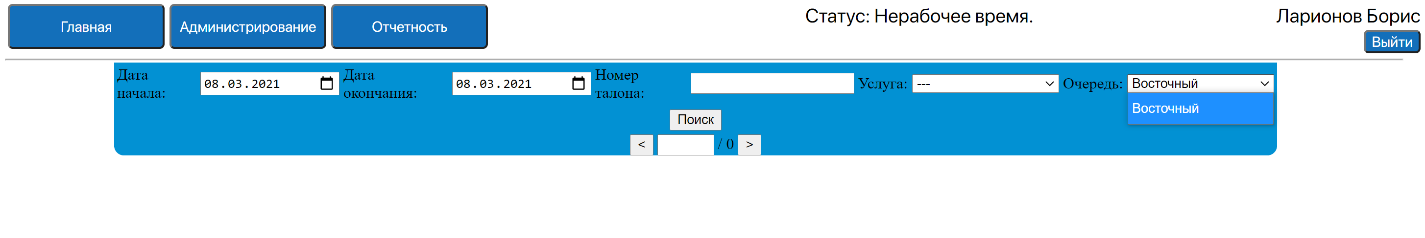


Рисунок 3. 5. Интерфейс главной страницы для "Старшего администратора".

Самыми многочисленными пользователями, разрабатываемой в данной бакалаврской работе информационной системы, являются операторы-кассиры. Они обслуживают посетителей, для этого разработано «рабочее место оператора электронной очереди», представленное на рисунке 3.6. В форме рабочего места оператора созданы кнопки для выполнения возможных действий с талоном. По мере обработки талона, в нем отображается информация о времени обслуживания и ожидания. По итогам внедрения, вариант используемый в веб-версии интерфейса, является запасным, на случай сбоев в работе корпоративной системы «ERP», а основной интерфейс разработан в этой системе «ERP».

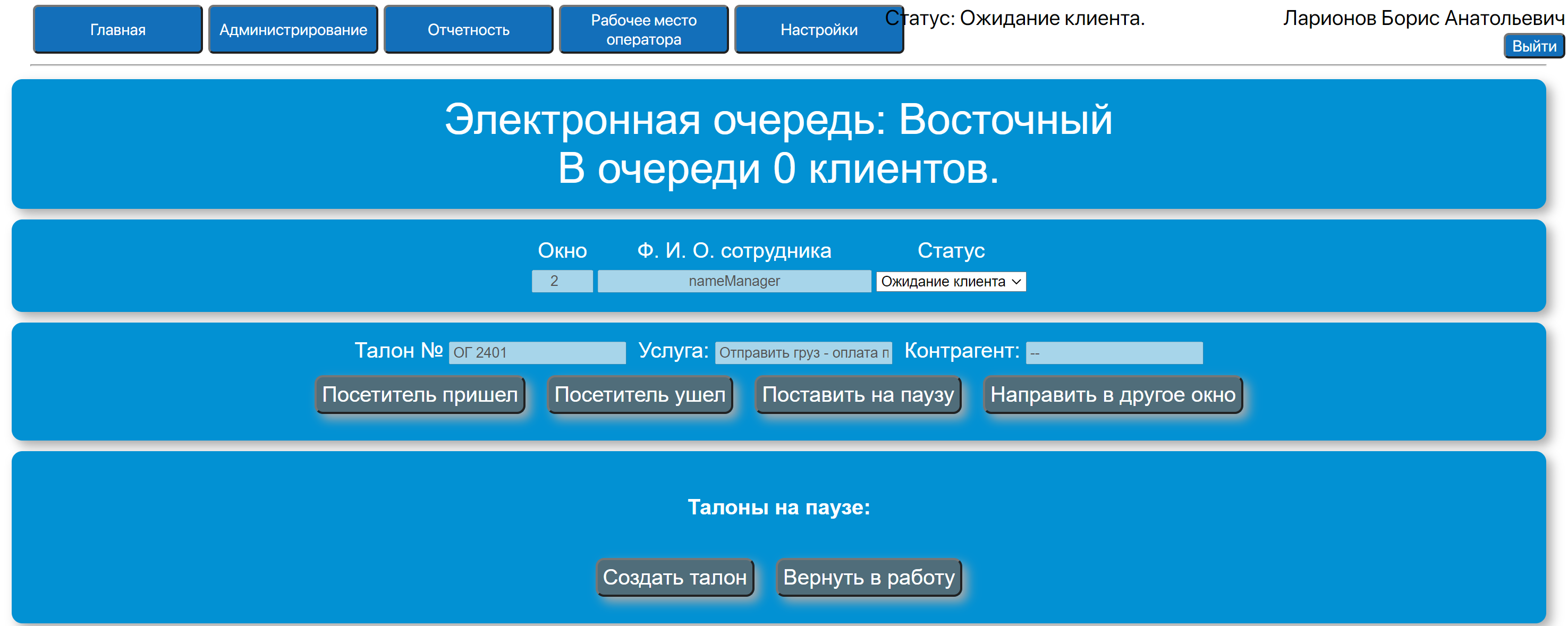


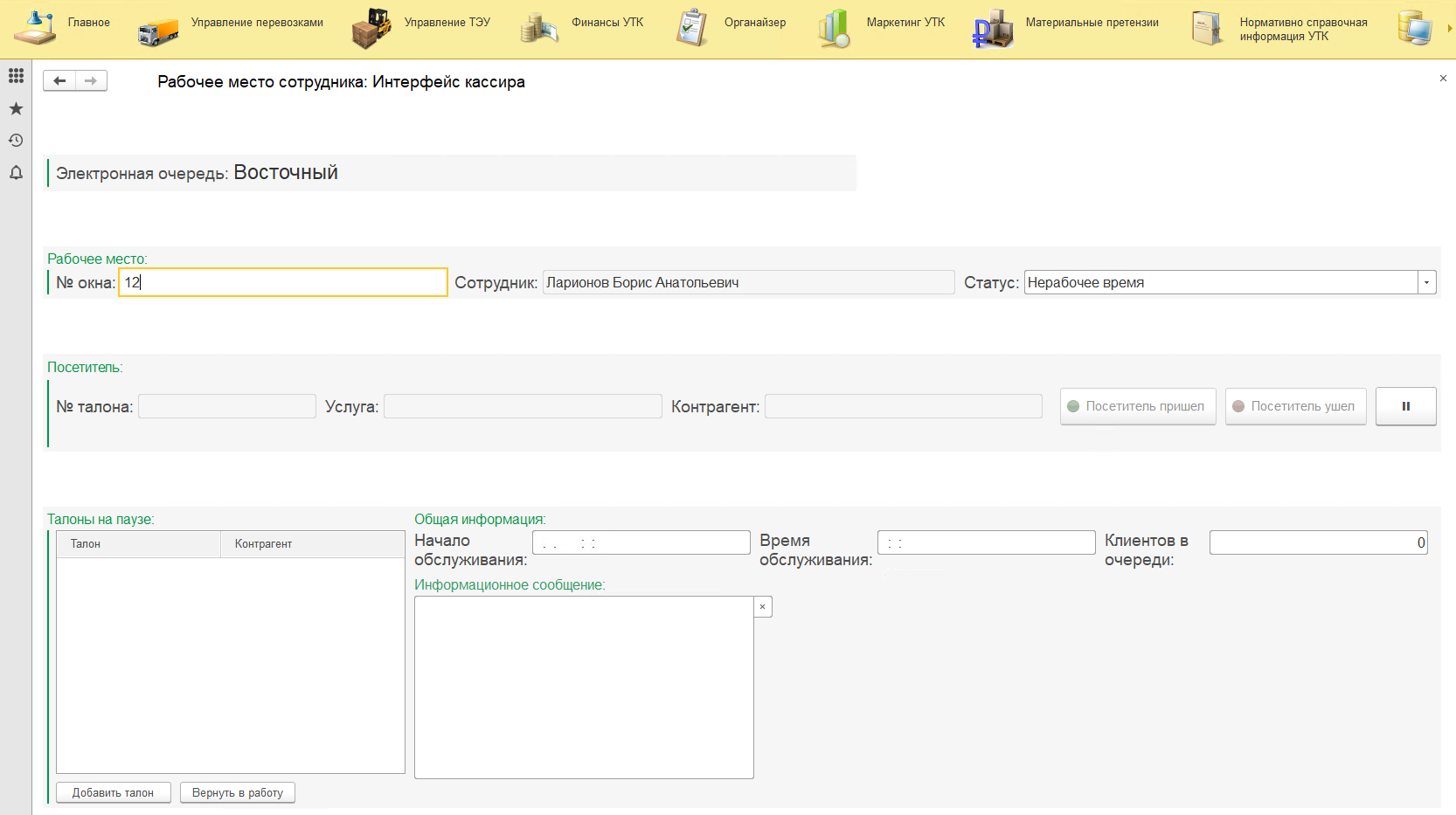
Рисунок 3. 6. Интерфейс рабочего места оператора.

Рисунок 3. 7. Интерфейс рабочего места оператора в "ERP".

На рисунке 3.7 представлен интерфейс рабочего места, реализованный в корпоративной системе «ERP». Для минимизации обращений к серверу, кнопки блокируется, когда статус пользователя не подразумевает взаимодействия с ними. Данный интерфейс имеет явное поле «Информационные сообщения», в котором выводится информация о возникающих ошибках при работе с системой учета посетителей. Эти сообщения отправляются с сервера, разрабатываемой системы, в случае возникновения ошибок. Благодаря интеграции с системой «ERP» пользователям, занимающимся обслуживанием клиентов, нет необходимости работать в нескольких окнах. Такой подход экономит время операторов и увеличивает скорость обслуживания.

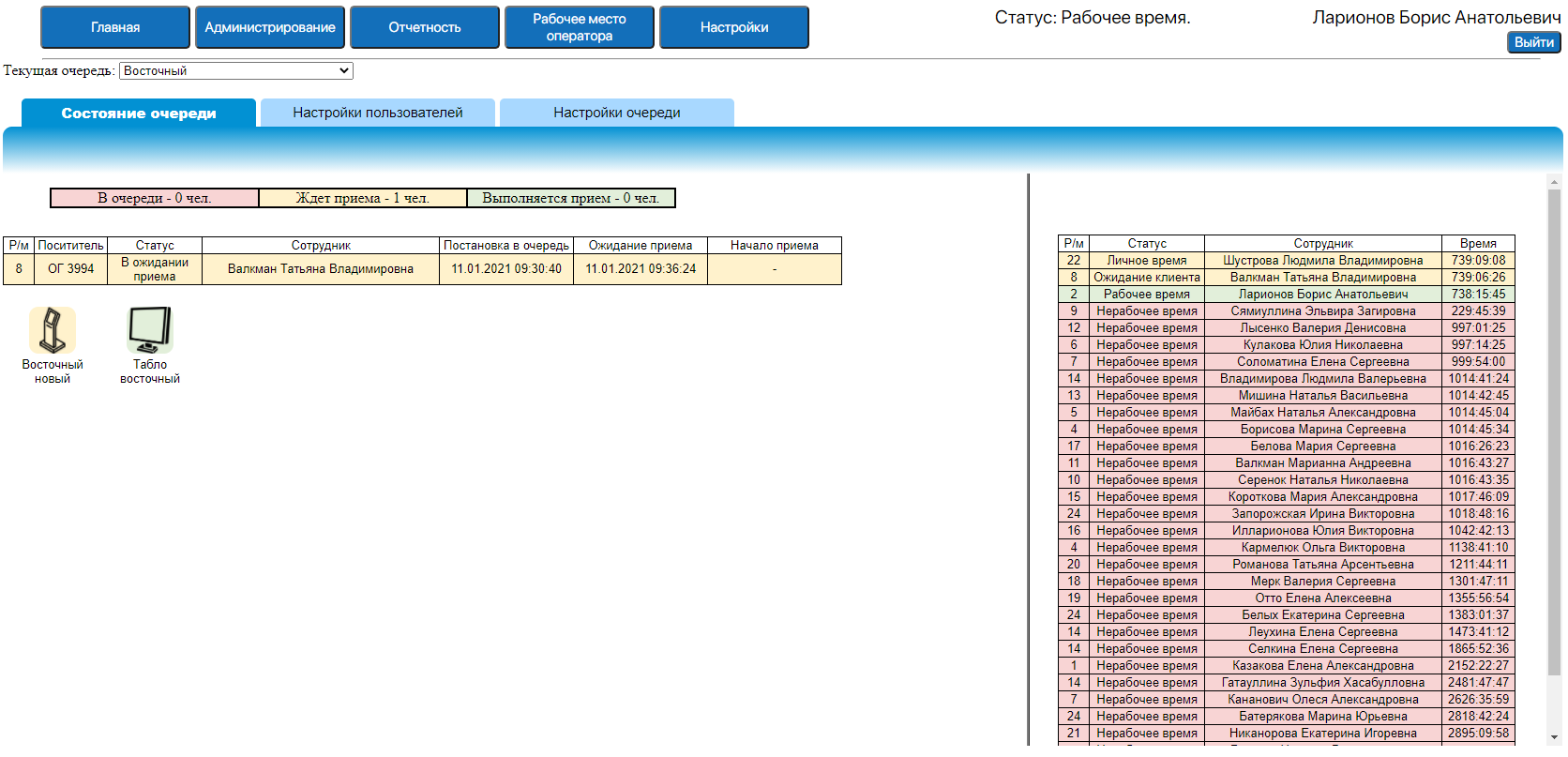


Рисунок 3. 8. Администрирование электронной очереди. "Состояние очереди"

Основной задачей старших операторов является контроль исполнения обязанностей подчиненными операторами, обеспечение максимально эффективного обслуживания клиентов. В случае необходимости они подключаются к обслуживанию клиентов. Для этого разработана форма администрирования электронной очереди. Внешний вид этой формы представлен на рисунках 3.8 – 3.10. Форма поделена на 3 вкладки.

Первая вкладка показывает оперативную информацию о состоянии очереди: статусы сотрудников и время нахождения в этих статусах, список талонов, находящихся в очереди или в обслуживании с основной информацией об этих талонах, состояния дополнительных устройств электронной очереди. Для отображения состояния устройств применен подход сокрытия информации и выделения цветом. Таким образом информацию о состоянии можно получить при необходимости наведя курсором на устройство. Работа этого механизма представлена на рисунке 3.11. Устройства могут быть помечены тремя цветами: красным, желтым и зеленым. Красный цвет обозначает критическое состояние, например пропала связь или закончилась бумага. Желтый обозначает нормальное функционирование, но скоро статус может перейти в состояние критического. Таким является сообщение о том, что скоро закончится бумага. Зеленые цвет позволяет понять что с устройством все в порядке. Такой подход позволяет не засорять видимую область страницы информацией, но держать в курсе о состоянии. Операторы всегда смогут обратить внимание на возникающие проблемы.

Вторая вкладка «Настройки пользователей» позволяет редактировать текущий рабочий статус сотрудника и перечень доступных к обслуживанию услуг. Также позволяет изменить текущую очередь сотрудника. Такая функция полезна тем, что старший оператор может сам включить в свою очередь новых сотрудников, не обращаюсь в техническую поддержку. Сотрудники, создаваемые автоматический регистрацией, получают права оператора и не имеют указанного офиса обслуживания. Интерфейс поделен на две половины. Слева выводятся атрибуты пользователя для настройки, а справа выведен список сотрудников, работающих в том же офисе, что и старший оператор. Для списка сотрудников доступна функция поиска.

На третьей вкладке «Настройки очереди» старший оператор может включать и выключать дополнительные устройства такие, как киоски выдачи талонов и информационные табло. Устройства при отключении таким способ выведут сообщение: «Электронная очередь временно не работает». Пользователи при этом будут понимать, что в системе наблюдается проблема и сразу перейдут на живую очередь, без попыток воспользоваться отключенным устройством.

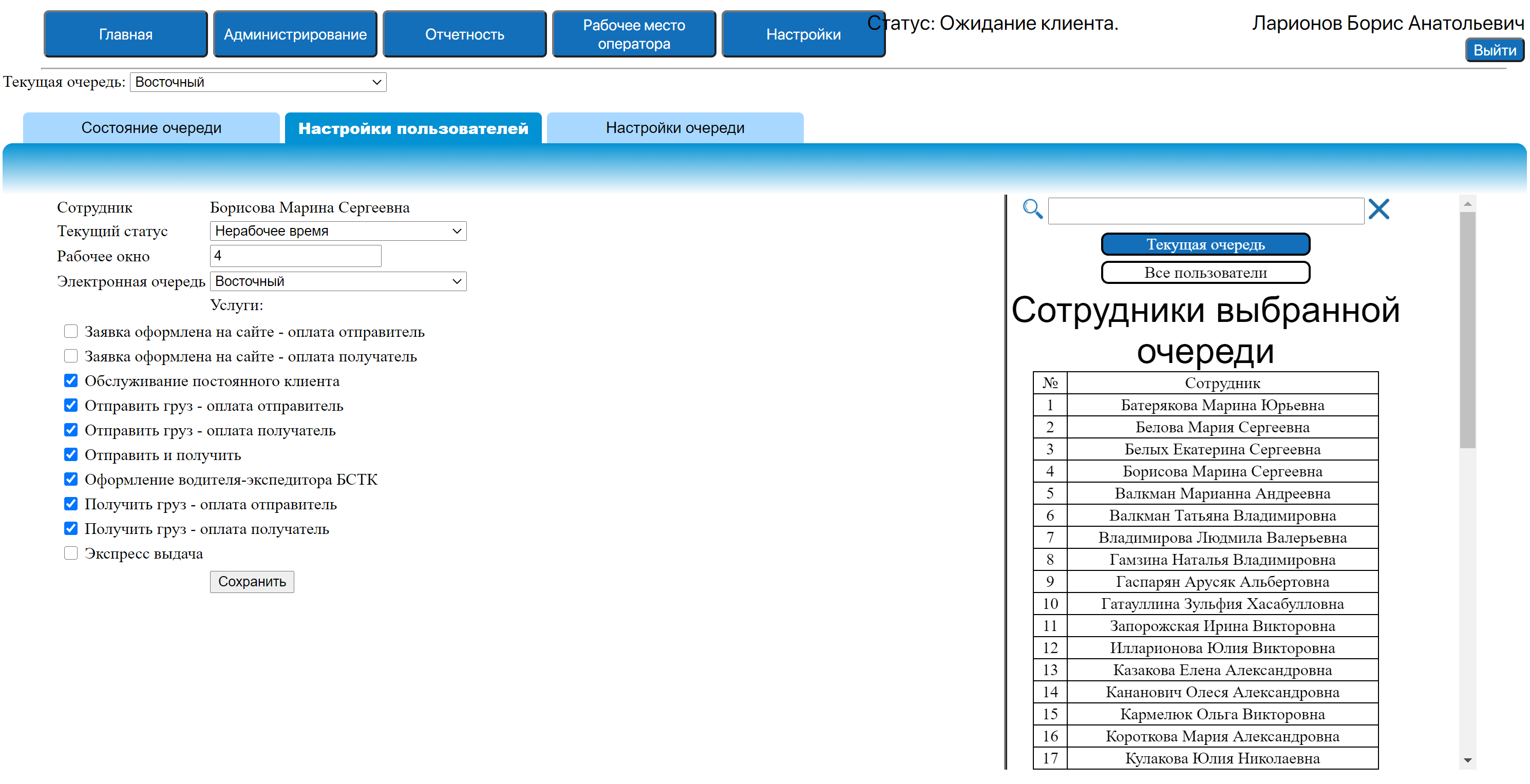


Рисунок 3. 9. Администрирование электронной очереди. "Настройки пользователей".

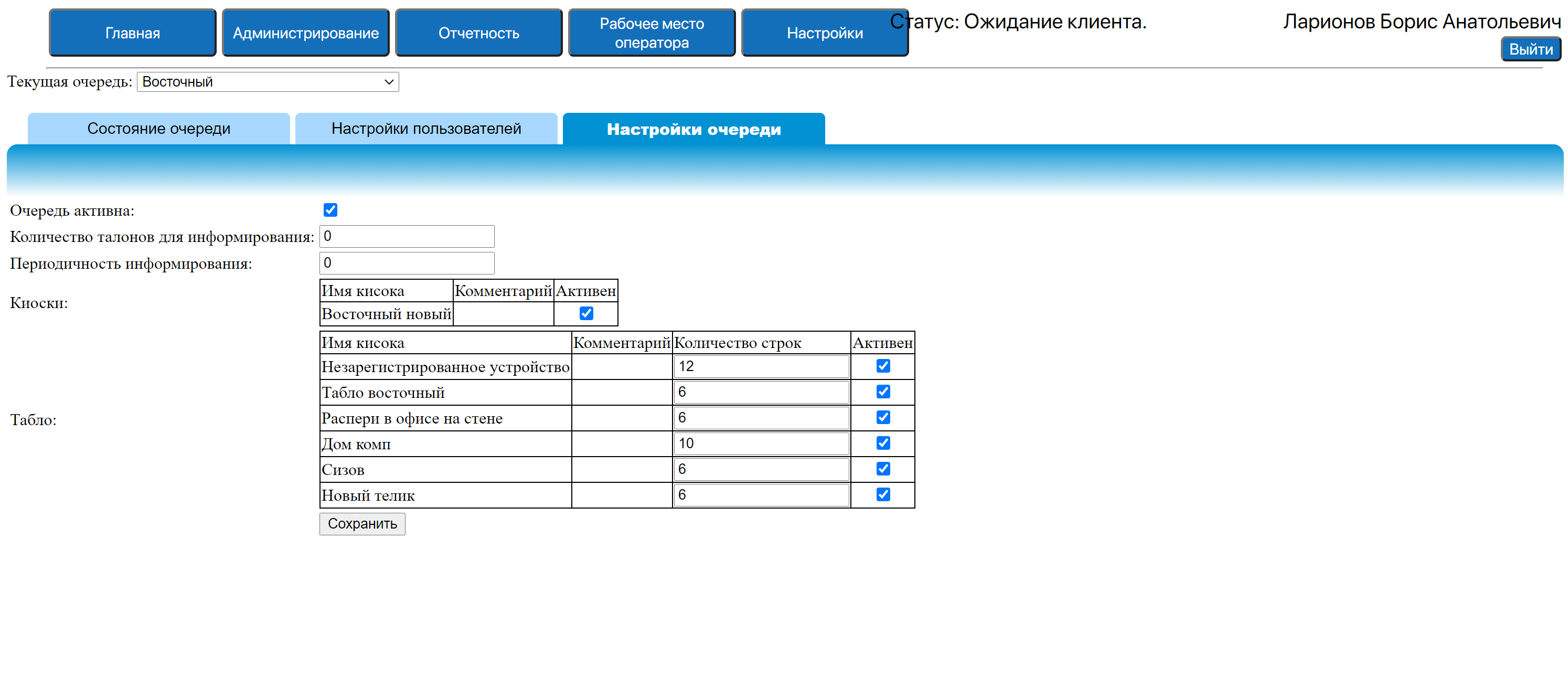


Рисунок 3. 10. Администрирование электронной очереди. "Настройки очереди".

Форма администрирования электронной очереди доступна только старшим операторам и администраторам системы. При этом старшие администраторы могут просматривать и редактировать информацию только в офисе, который указан у них.

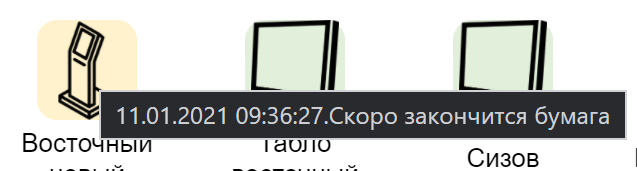


Рисунок 3. 11. Сообщение о состоянии дополнительного устройства.

Для формирования отчетов разработана форма вывода этих отчетов, представленная на рисунке 3.12. Эта форма разработана как универсальное средство вывода результатов отчетов. Большую часть страницы занимает область вывода результатов. В случае необходимости, у этой области появляются свои полосы прокрутки. Для ввода параметров отчета создано скрывающееся меню, обозначенное шестеренкой. Это удобно, так как параметров может быть довольно много и они будут съедать полезною область просмотра, а так после формирования меню ввода параметров сворачивается в маленькую иконку. В данный момент реализован только один отчет, показывающий сводную информацию по офисам о времени обслуживания клиентов с дроблением на показатели нормы.

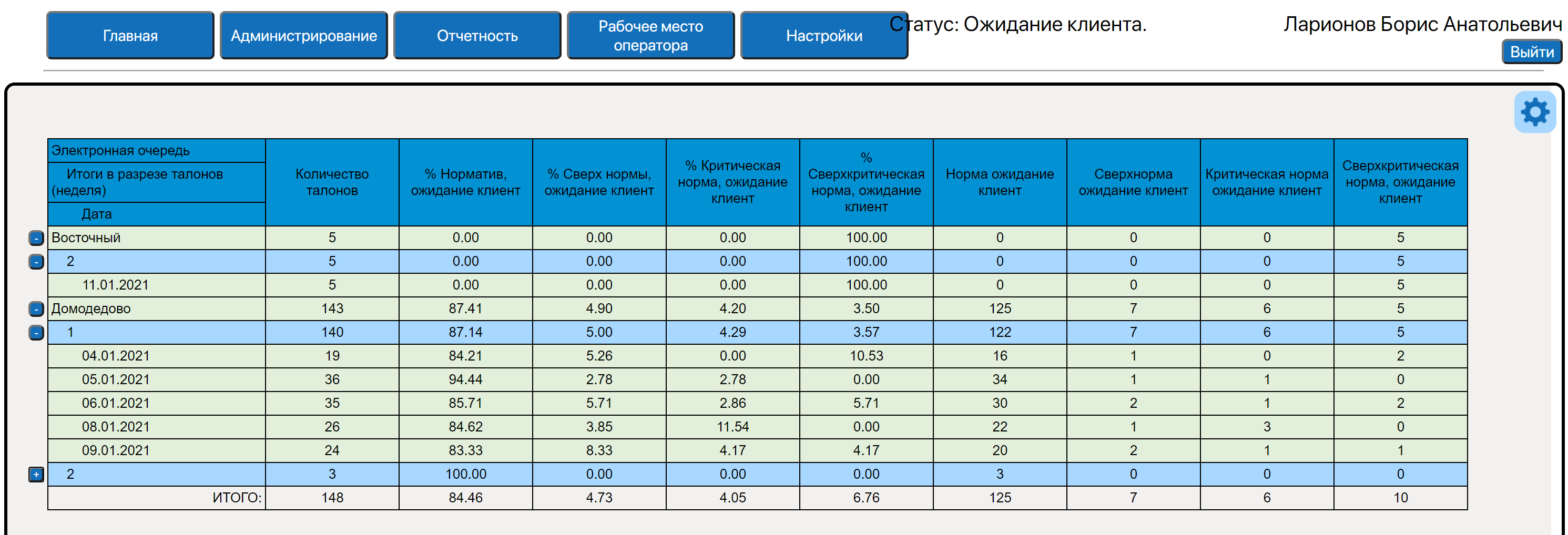


Рисунок 3. 12. Форма формирования отчетов.

Информационное табло является также HTML-страницей со скриптами написанными на «JavaScript». Принцип работы страницы информационного табло и киоска выдачи талонов схож, поэтому дальнейшая информация актуальна для обоих пользовательских интерфейсов. Разница заключается только в том, что информационное табло не подразумевает ввода данных пользователем. Внешний вид интерфейсов дополнительных устройств представлен на рисунках 3.13 – 3.14. При работе скриптов на данных страницах периодически выполняются обращения к серверу. В случае отсутствия связи данные интерфейсы переходят в режим ожидания, в котором проверка связи осуществляется реже, а на экран выводится сообщение «Электронная очередь временно не работает».

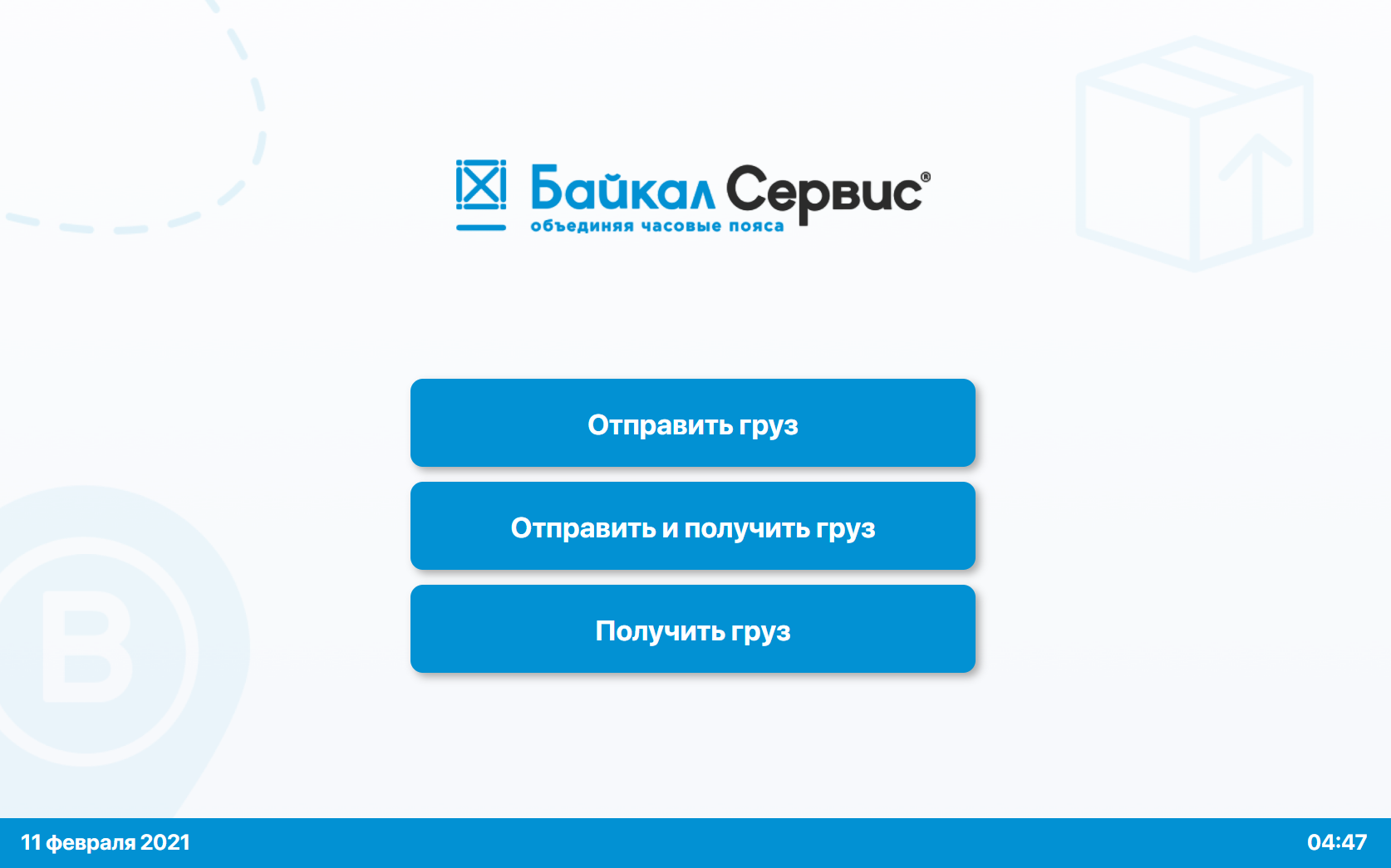


Рисунок 3. 13. Интерфейс киоска выдачи талонов.

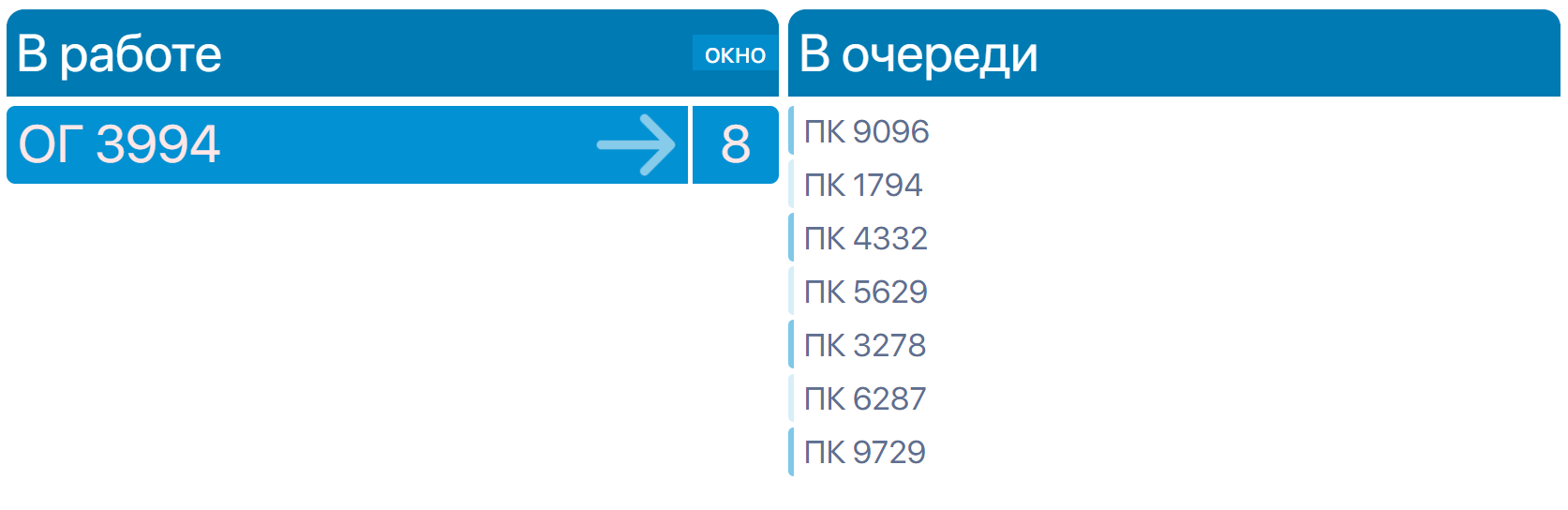


Рисунок 3. 14. Интерфейс информационного табло.

Информационное табло выводит список талонов, находящихся в очереди, не распределенных на операторов, справа. Слева выводятся талоны, назначенные в окно, при этом каждый новый назначенные выводит вверх списка, смещая предыдущие вниз. При распределении талона, издается звуковое оповещение, а представление талона на экране некоторое время переливается цветом для привлечения внимания.

## Отладка и внедрение программного продукта.

Завершение написания программного кода продукта не является завершением разработки. Последним этапом разработки программного обеспечения является отладка. Отладка бывает локальной, на оборудовании разработчика, и выполняется обычно по мере написания кода, и бывает предрелизной. Обычно предрелизная отладка происходит в условиях близких к условиям рабочего запуска.

Для отладки приложения, разработанного в данной бакалаврской работе, необходимо произвести настройку рабочего окружения, а именно, настроить систему управления базами данных, поместить исполняемый файл приложения, на сервер которые будет его выполнять, подготовить клиентские устройства.

При настройке работы серверного приложения необходимо определить способ его запуска. Одним из самых надежных подходов является регистрация исполняемого файла как службы в операционной системе. В операционных системах Linux это осуществляется с помощью утилиты «Systemctl». Работа приложения как службы имеет ряд плюсов:

* Работа приложения контролируется операционной системой, по средствам получения кода завершения программы. В случае сбоя ОС перезапустит приложение самостоятельно.
* Запуск службы можно привязать к загрузке операционной системы. Таким образом при перезагрузке сервера, приложение-служба будет запускаться автоматически вместе со стартом ОС.
* Обычно вывод в консоль от служб записывается в системные журналы. В ОС CentOS запись происходит в журнал, читать записи которого можно с помощью утилиты «journal», используя множество параметров отбора. Таким образом решается проблема логирования и хранения логов.

При настройке информационного табло, необходимо прописать URL-адрес страницы сервера системы учета посетителей. Так устройство (телевизор), реализующее информационное табло, при запуске перейдет по этой странице и отрисует необходимый интерфейс.

Настройка киоска выдачи талонов делится на два этапа. Первым этапом необходимо настроить драйвер, разработанный в рамках проекта данный выпускной квалификационной работы. Запускать драйвер необходимо как службу, основываясь на плюсах, описанных выше. Вторым этапом является настройка запуска браузера с параметрами, обеспечивающими безопасность работы такого устройства. Основным параметров является URL-адрес для получения интерфейса. Дополнительные параметры переводят браузер в режим киоска, в котором браузер запускается во весь экран, и проверяют состояния браузера, в случае закрытия перезапускают его.

Со стороны системы «ERP», в которую интегрировано рабочее место оператора, требуется указание рабочего URL-адреса сервера, на котором опубликован программный интерфейс API, и данных авторизации в этом интерфейсе: логин и пароль

На этом процесс внедрения можно считать оконченным. После необходимо произвести отладку, в ходе которой отработать все сценарии взаимодействия пользователей с системой.

Выводы

В третьей главе рассмотрены методы создания базы данных, для хранения данных, разработанной в данной бакалаврской работе, информационной системы. Описана технология строения программного продукта сервера учета посетителей, представлены алгоритмы работы некоторых основных процессов. Разработан графический интерфейс для взаимодействия с пользователями.

Заключение

В ходе работы над данной выпускной квалификационной работой, произведен анализ предприятия в лице крупной транспортной компании. Изучены проблемы существующей информационной системы для учета посетителей. Выявлены требования для внедрения новой оптимизированной системы учета.

Итогом работы стал готовый программный комплекс, реализующий функции учета посетителей. Данные комплекс является самостоятельным, узконаправленным в деятельности, сервисом. Благодаря вынесению функционала изучаемой системы из крупной корпоративной системы «ERP», постоянно совершенствующейся и обновляемой, в отдельную систему учета, минимизирован человеческий фактор, из-за которого возможны частые сбои в работе программы. Разработанная система не требует постоянных доработок.

Разработка происходила с использованием программных продуктов, распространяемых на основе свободных лицензий, что уменьшило стоимость содержания программного продукта, реализующего функции системы учета посетителей.

Список использованной литературы

1. «BM-GROUP Фабрика решений». [Электронный ресурс]. URL:<https://bm-technology.ru/solutions/elektronnaya-ochered1/ >.
2. «BR Systems». [Электронный ресурс]. URL:< <https://app.ocheredi.net/>>.
3. Статья представителя «BR Systems» от 4.06.2020 года. [Электронный ресурс]. URL:< <https://vc.ru/tribuna/131672-oblachnyy-servis-dlya-organizacii-elektronnoy-ocheredi>>.
4. Лицензионное соглашение MySQL Server v. 5.6. [Электронный ресурс]. URL:< <https://dev.mysql.com/doc/refman/5.6/ja/license-gnu-gpl-v3-runtime-exception-v31.html>>.
5. Лицензионное соглашение PostgreSQL. [Электронный ресурс]. URL:< https://www.postgresql.org/about/licence/ >.
6. MySQL и PostgreSQL. Часть 1. Сравнительный анализ. Сергей Яковлев от 27.07.2010. [Электронный ресурс]. URL:< <https://www.ibm.com/developerworks/ru/library/os-mysql-postgresql/01/index.html>>.
7. Отчет State of the Octoverse 2020. [Электронный ресурс]. URL:< https://octoverse.github.com/ >.
8. Статья «Язык программирования Java: особенности, популярность, ситуация на рынке труда» Дмитрий Дементий. [Электронный ресурс]. URL:<https://yandex.ru/turbo/ru.hexlet.io/s/blog/posts/yazyk-programmirovaniya-java-osobennosti-populyarnost-situatsiya-na-rynke-truda >.
9. Создание баз данных. Учебник [Текст]. Автор: Кудрявцев К. Я. НИЯУ МИФИ 2010. 155 с.
10. MySQL 5. Книга [Текст]. Автор: Максим Кузнецов, Игорь Симдянов. БХВ-Петербург 2010, 1007 с.
11. PostgreSQL 11 Мастерство разработки. Книга [Текст]. Автор: Ганс-Юрген Шёниг. ДМК Пресс, 2019. – 352 с.
12. Java. Полное руководство. 10-е издание. Книга [Текст]. Автор: Герберт Шилдт. Диалектика 2019, 1488 с.
13. Проектирование информационных систем. Учебник [Текст]. Автор: Шамсутдинов Т.Ф. КГАСУ 2018, 110 с.
14. Базы данных : учебник и практикум для академического бакалавриата. Учебник [Текст]. Австор: Нестеров С. А. Юрайт 2017, 230 с.
15. Применение объектного моделирования с использованием UML и анализ прецедентов. Учебник [Текст]. Автор: Розенберг Д., Скотт К. ДМК Пресс, 2002, 160 с.
16. Введение в ORM (Object Relational Mapping). Статья [Электронный ресурс]. 2012. URL: < <http://internetka.in.ua/orm-intro/>>.
17. Документация Swagger UI. Документация [Электронный ресурс]. URL: <https://swagger.io/solutions/api-documentation/>>.
18. Официальный сайт поддержки «Open JDK». [Электронный ресурс]. URL: <https://openjdk.java.net/>.
19. Официальный сайт поддержки фреймворка «Spring». [Электронный ресурс]. URL: <https://spring.io/projects/spring-boot>.
20. Официальный сайт поддержки фреймворка «Hibernate». [Электронный ресурс]. URL: <https://hibernate.org/tools/>.
21. Официальный сайт поддержки фреймворка «Thymeleaf». [Электронный ресурс]. URL: <https://www.thymeleaf.org/documentation.html>.
22. Официальный сайт «Jetbrains» IDE «InteliJ IDEA». [Электронный ресурс]. URL: < https://www.jetbrains.com/idea/ >.
23. Код создания класса Manager

@Entity  
public class Manager implements UserDetails {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.*AUTO*)  
 private Long id;  
  
 @NotNull  
 private String loginAD;  
  
 @NotNull  
 private String password;  
  
 @NotNull  
 private String name;  
  
 private boolean active;  
  
 @NotNull  
 private String restToken;  
  
 @ElementCollection(targetClass = Role.class, fetch = FetchType.*EAGER*)  
 @CollectionTable(name = "user\_role", joinColumns = @JoinColumn(name = "user\_id"))  
 @Enumerated(EnumType.*STRING*)  
 private Set<Role> roles;  
  
 @ManyToOne  
 @JoinColumn(name = "queue\_id")  
 private Queue queue;  
  
 @ManyToMany(mappedBy = "managers", fetch = FetchType.*LAZY*)  
 private List<TicketService> ticketServices;  
  
 public Manager() {  
 }  
  
 public Manager(UserEdit userEdit) {  
 this.id = userEdit.getId();  
 this.loginAD = userEdit.getLoginAD();  
 this.name = userEdit.getName();  
 this.active = userEdit.isActive();  
 this.roles = userEdit.getRoles();  
 this.queue = userEdit.getQueue();  
 this.ticketServices = userEdit.getTicketServices();  
 if (!(userEdit.getNewSecurityPass()==null))  
 this.password = userEdit.getNewSecurityPass();  
 else  
 this.password = userEdit.getCurrentPassword();  
 }  
  
 @Override  
 public boolean isAccountNonExpired()  
 {  
 return true;  
 }  
  
 @Override  
 public boolean isAccountNonLocked()  
 {  
 return true;  
 }  
  
 @Override  
 public boolean isCredentialsNonExpired()  
 {  
 return true;  
 }  
  
 @Override  
 public boolean isEnabled()  
 {  
 return isActive();  
 }  
  
 @Override  
 public Collection<? extends GrantedAuthority> getAuthorities()  
 {  
 return getRoles();  
 }  
  
 @Override  
 public boolean equals(Object o)  
 {  
 try {  
 return this.id == ((Manager) o).getId();  
 } catch (Exception e) {  
 return false;  
 }  
 }  
  
 @Override  
 public int hashCode()  
 {  
 return Objects.*hash*(getId(), getUsername(), getPassword(), getName(), isActive(), getRoles());  
 }  
  
 public void setId(Long id) {  
 this.id = id;  
 }  
  
 public void setLoginAD(String loginAD) {  
 this.loginAD = loginAD;  
 }  
  
 public void setPassword(String password) {  
 this.password = password;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public void setActive(boolean active) {  
 this.active = active;  
 }  
  
 public void setRoles(Set<Role> roles) {  
 this.roles = roles;  
 }  
  
 public Long getId() {  
 return id;  
 }  
  
 public String getLoginAD() {  
 return loginAD;  
 }  
  
 @Override  
 public String getPassword() {  
 return password;  
 }  
  
 @Override  
 public String getUsername() {  
 return getLoginAD();  
 }  
  
 @Override  
 public String toString() {  
 return name;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public boolean isActive() {  
 return active;  
 }  
  
 public Set<Role> getRoles() {  
 return roles;  
 }  
  
 public Queue getQueue() {  
 return queue;  
 }  
  
 public void setQueue(Queue queue) {  
 this.queue = queue;  
 }  
  
 public List<TicketService> getTicketServices() {  
 return ticketServices;  
 }  
  
 public void setTicketServices(List<TicketService> ticketServices) {  
 this.ticketServices = ticketServices;  
 }  
}

1. Код класса ManagesrViewStatuses

@Entity  
@Immutable  
@Subselect("WITH managers AS(\n" +  
 "\tSELECT \*\n" +  
 "\tFROM manager\n" +  
 "\tWHERE active),\n" +  
 "\tlast\_statuses AS(\n" +  
 "\tSELECT\n" +  
 "\t\tMAX(date) AS date,\n" +  
 "\t\tmanager\_id\n" +  
 "\tFROM managers\_status\n" +  
 "\tGROUP BY manager\_id)\n" +  
 "\t\n" +  
 "SELECT \n" +  
 "\tm.id,\n" +  
 "\tm.loginad,\n" +  
 "\tm.name,\n" +  
 "\tm.queue\_id,\n" +  
 "\tms.date,\n" +  
 "\tms.casement,\n" +  
 "\tms.status\n" +  
 "FROM managers m LEFT OUTER JOIN last\_statuses ls\n" +  
 "ON (m.id = ls.manager\_id)\n" +  
 "LEFT OUTER JOIN managers\_status ms\n" +  
 "ON (m.id = ms.manager\_id\n" +  
 " AND ls.date = ms.date);")  
@Table(name = "managers\_statuses")  
public class ManagersViewStatuses {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.*AUTO*)  
 @Column(name = "id", updatable = false, nullable = false)  
 private Long id;  
  
 @Column(name = "loginad")  
 private String loginAd;  
  
 private String name;  
  
 @ManyToOne  
 @JoinColumn(name = "queue\_id")  
 private Queue queue;  
  
 private Date date;  
  
 private Integer casement;  
  
 @Enumerated(EnumType.*STRING*)  
 private Status status;  
  
 public Long getId() {  
 return id;  
 }  
  
 public void setId(Long id) {  
 this.id = id;  
 }  
  
 public String getLoginAd() {  
 return loginAd;  
 }  
  
 public void setLoginAd(String loginAd) {  
 this.loginAd = loginAd;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public Queue getQueue() {  
 return queue;  
 }  
  
 public void setQueue(Queue queue) {  
 this.queue = queue;  
 }  
  
 public Date getDate() {  
 return date;  
 }  
  
 public void setDate(Date date) {  
 this.date = date;  
 }  
  
 public Integer getCasement() {  
 return casement;  
 }  
  
 public void setCasement(Integer casement) {  
 this.casement = casement;  
 }  
  
 public Status getStatus() {  
 return status;  
 }  
  
 public void setStatus(Status status) {  
 this.status = status;  
 }  
}

1. Алгоритм работы метода распределения талонов



1. Алгоритм работы метода авторизации



1. Описание программного интерфейса API для рабочего места оператора

При подключении к интерфейсу использовать базовую авторизацию по логину и паролю сервисного пользователя.

Методы GET:

/rest/work\_station/getElQueue (Возвращает текущую очередь пользователя)

params

- idUser: String

Response

- {'name': 'String',

'clientsInQueue':'int'}

//////////////////////////////////////////////////////////

/rest/work\_station/getStatuses (Возвращает текущий статус, доступные статусы для пользователя, и его окно)

params

- idUser: String

Response

- {'currentStatus':{'name':'String', value: String},

'statuses':['name': 'String','value': 'String'],

'casement':'int'}

//////////////////////////////////////////////////////////

/rest/work\_station/getStatus (Возвращает текущий статус пользователя)

params

- idUser: String

Response

- {'name':'String',

'value': 'String'}

//////////////////////////////////////////////////////////

/rest/work\_station/isTicket (Возвращает идентификатор назначенного талона)

params

- idUser: String

Response

- {'isTicket':'boolean',

'id':'uuid'}

//////////////////////////////////////////////////////////

/rest/work\_station/getTicket (Возвращает данные о талоне)

params

- idTicket: uuid

Response

- {'id':'uuid',

'name':'String',

'status':'String',

'dateStartService':'05.12.2020 09:22:32',

'timeOnPause':'long', //Время пребывания в паузе в милисекундах

'service':'String',

'contractor':{'id':'String',

'name':'String'},

'sellings':['number':'String']}

//////////////////////////////////////////////////////////

/rest/work\_station/getTicketsPaused (Возвращает список талонов на паузе у указанного оператора)

params

- idUser: String

Response

- {['id':'uuid',

'name':'String',

'dateStartService':'05.12.2020 09:22:32',

'service':'String',

'contractor':{'id':'String',

'name':'String'},

'sellings':['number':'String']]}

//////////////////////////////////////////////////////////

/rest/work\_station/getAvailableOperators (Возвращает список операторов доступных для перенаправдления талона)

params

- idUser: String

Response

- {'idUser':'String',

'fullName':'String',

'casement':'int'}

//////////////////////////////////////////////////////////

POST:

/rest/work\_station/setNewStatus (Устанавливает новый статус у оператора)

Request:

{'idUser':'String',

'valueNewStatus':'String',

'casement':'int'}

Response

- {'status':'ok'}

//////////////////////////////////////////////////////////

/rest/work\_station/startServiceClient (Устанавливает статусы обслуживания клиента у оператора и его текущего талона)

Request:

{'idUser':'String'}

Response

- {'status':'ok'}

//////////////////////////////////////////////////////////

/rest/work\_station/endServiceClient (Устанавливает статусы звершения обслуживания клиента у оператора и его текущего талона)

Request:

{'idUser':'String'}

Response

- {'status':'ok'}

//////////////////////////////////////////////////////////

/rest/work\_station/pauseServiceTicket (Ставит талон на паузу. У оператора статус сохраняется, но можно сменить статус на новые доступные)

Request:

{'idUser':'String'}

Response

- {'status':'ok'}

//////////////////////////////////////////////////////////

/rest/work\_station/resumeServiceTicket (Возвращает талон в работу. У оператора устанавливает стаус обслуживания клиента)

Request:

{'idUser':'String',

'idTicket':'uuid'}

Response

- {'status':'ok'}

//////////////////////////////////////////////////////////

/rest/work\_station/redirectTicket (Направляет талон к назначеному оператору)

Request:

{'idUser':'String',

'idTicket':'uuid',

'idNewOperator':'String'}

Response

- {'status':'ok'}

//////////////////////////////////////////////////////////

/rest/work\_station/createTicketPartner (Создает талон для постоянного клинета. Присваивает в работу к указанному оператору)

Request:

{'idUser':'String'}

Response

- {'status':'ok',

'idTicket':'uuid'}

//////////////////////////////////////////////////////////

/rest/work\_station/setContractor (Указывает контрагента для талона)

Request:

{'idTicket':'uuid',

'contractor':{'id':'String',

'name':'String'}}

Response

- {'status':'ok'}

//////////////////////////////////////////////////////////

/rest/work\_station/addSelling (Добавляет квитанцию в указанный талон)

Request:

{'idTicket':'uuid',

'numberSelling':'String'}

Response

- {'status':'ok'}

//////////////////////////////////////////////////////////