Министерство науки и высшего образования РФ
Федеральное государственное автономное
образовательное учреждение высшего образования
«**СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ**»

Институт космических и информационных технологий

Кафедра «Системы автоматики,
автоматизированное управление и проектирование»

УТВЕРЖДАЮ
Заведующий кафедрой
_____ А.С. Климов

«____» июня 2023 г.

БАКАЛАВРСКАЯ РАБОТА

15.03.04 – Автоматизация технологических процессов и производств

АВТОМАТИЗИРОВАННАЯ СИСТЕМА ОПЕРАТИВНОГО КОНТРОЛЯ ПОЗАКАЗНОГО ПРОИЗВОДСТВА РАДИОЭЛЕКТРОННОЙ АППАРАТУРЫ

			доцент, канд. техн. наук
Руководитель	_____	___.06. 2023 г.	О.В. Дрозд
Выпускник	_____	___.06. 2023 г.	И.А. Любухина
Нормоконтролер	_____	___.06. 2023 г.	доцент, канд. техн. наук
			О.В. Дрозд

Красноярск 2023

РЕФЕРАТ

Бакалаврская работа на тему «Автоматизированная система оперативного контроля позаказного производства радиоэлектронной аппаратуры» содержит 128 страниц текстового документа, 62 иллюстраций, 15 таблицы, 2 приложения, 29 использованных источников.

АВТОМАТИЗИРОВАННАЯ СИСТЕМА, ОПЕРАТИВНЫЙ КОНТРОЛЬ, ПОЗАКАЗНОЕ ПРОИЗВОДСТВО РАДИОЭЛЕКТРОННОЙ АППАРАТУРЫ

Цель данной работы заключается в обеспечении позаказного производства РЭА посредством оперативного контроля отдельных рабочих мест и производственной цепочки в целом.

Задачи, которые решались в ходе выполнения данной работы:

– разработка архитектуры автоматизированной системы оперативного контроля позаказного производства радиоэлектронной аппаратуры;

– разработка клиент-серверного программного обеспечения системы оперативного контроля позаказного производства радиоэлектронной аппаратуры;

– рассмотрение реализации автоматизированного рабочего места проведения испытаний навигационной аппаратуры потребителя с использованием предлагаемой системы оперативного контроля.

При выполнении были задействованы следующие программные продукты: *Python*, *PostgreSQL* 10, *pgAdmin* 3, *EAExample*.

В данной работе предложены различные решения и методы, направленные на разработку автоматизированной системы оперативного контроля позаказного производства радиоэлектронной аппаратуры.

Работа основана на тщательном исследовании и анализе существующих проблем и вызовов, связанных с контролем и управлением процессами производства радиоэлектронной аппаратуры в заказной среде.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Автоматизация технологического процесса – совокупность методов и средств, предназначенная для реализации системы или систем, позволяющих осуществлять управление самим технологическим процессом без непосредственного участия человека, либо оставления за человеком права принятия наиболее ответственных решений [1].

Введение автоматизации на производстве позволяет значительно повысить производительность труда, обеспечить стабильное качество выпускаемой продукции. Актуальность внедрения автоматизации производства обусловлена такими факторами как:

- максимизация экономии времени и точность выполнения всех процессов в срок;
- высокий контроль над правильностью выполнения работ;
- необходимость рационального распределения организационных функций между административным и производственным управлением [2].

Цифровизация производственных процессов в той или иной степени охватывает все сферы промышленности. Практически все отрасли – агропромышленный комплекс, транспорт, металлургия, добыча природных ресурсов, машиностроение, химическая промышленность, приборостроение и другие – движутся в сторону Индустрии 4.0. Область производства радиоэлектронной аппаратуры является тем сектором, где общемировые тренды развития и внедрения новых технологий реализуются наиболее активно [3].

1 Аналитический обзор предметной области

1.1 Проектно-производственные цепочки

Проектно-производственные цепочки, ориентированные на удовлетворение потребностей клиента, являются ключевым элементом передовой производственной концепции «Индустрия 4.0», их организация обусловлена объективными тенденциями как технологического, так и общественного развития. Предельным случаем адаптации проектно-производственной цепочки под требования заказчика является организационная структура «Проектирование на заказ», которая предполагает конфигурацию жизненного цикла изделия под уникальное сложное техническое изделие (СТИ), производимое либо в единственном экземпляре, либо малой серией. Проектирование на заказ характеризуется отсутствием закрепления операций за рабочими местами, применением уникального оборудования, частой переналадкой оборудования, общей высокой трудоемкостью изделий и длительным циклом их изготовления, высокой себестоимостью выпускаемой продукции [4]. Специфика сложного технического изделия заключается в том, что его проектирование и производство требует использования разнотипных моделей представления изделия и его компонентов, а процесс проектирования изделия имеет междисциплинарный характер. В целом создание СТИ представляет собой многоэтапный процесс, характеризующийся значительными материальными затратами, длительными сроками реализации и внедрения, существенной неопределенностью, связанной с возможными изменениями целей проектирования и применения изделия, а также воздействием различного рода возмущений внешней среды на изделие в процессе реализации его жизненного цикла. Комплексная автоматизация высокотехнологичного производства неизбежно связана с применением методов и средств автоматизации проектно-конструкторской деятельности. Управление процессами проектно-конструкторской подготовки производства

изделий, их последовательная интеграция с процессами изготовления и эксплуатации с целью организации единой системы сбора и обработки данных, повышает качество и эффективность работы всех звеньев производства. Необходимость такой интеграции особенно остро возникает на предприятиях, реализующих организационную структуру «Проектирование на заказ»: на этапах взаимодействия с заказчиком, формирования и актуализации технического задания, собственно проектирования и изготовления изделий, а также при последующем авторском надзоре.

Позаказный метод производства – это производство изделий под заказ; подразумевается, что каждое изделие уникально (имеет свою спецификацию) и производится при наличии непосредственного заказа покупателя с предоплатой или постоплатой заказа [5].

Позаказное производство является более гибким и адаптивным, чем серийное производство, поскольку каждый заказ может иметь свои особые требования и характеристики, что требует соответствующей настройки оборудования и технологических процессов.

Позаказное производство может быть выгодным для производителей, которые работают на специализированном рынке или выпускают товары с высокой добавленной стоимостью [6].

При проектировании и производстве СТИ следует учитывать необходимость обработки и отслеживания изменений проектно-конструкторских данных об изделии структурной неопределенности процесса проектирования, типичной для организационной структуры «Проектирование на заказ». Такие условия продиктованы сложностью конечной продукции и требуют применения широкой номенклатуры территориально-распределенных программных и аппаратных средств обеспечения проектирования. Это, в свою очередь, значительно затрудняет создание единой системы сбора и обработки проектно-конструкторских данных из-за отсутствия функциональной совместимости между отдельными системами и средствами проектирования [7].

1.2 Производство радиоэлектронной аппаратуры

Радиоэлектронная аппаратура (РЭА) представляет собой совокупность элементов, объединённых в сборочные единицы и устройства, предназначенные для преобразования и обработки электромагнитных сигналов в диапазоне от инфранизких до сверхвысоких (СВЧ) частот [8].

Производство радиоэлектронной аппаратуры по заказу основано на применении технологических принципов, которые включают использование универсального технологического оборудования и специальной оснастки. Оснастка позволяет быстро перенастраивать оборудование для перехода от одной операции обработки детали к другой, обеспечивая гибкость и эффективность производственного процесса.

В современных условиях, где сокращается разнообразие деталей и сборочных единиц, производимых на различных предприятиях, все чаще применяют автоматизированные системы управления производством (АСУТП). Они способствуют более эффективной организации работы и повышению качества производства радиоэлектронной аппаратуры.

Однако проблема повышения качества и эффективности производства РЭА является сложной и многогранным заданием, требующим комплексного подхода и системного решения. Для решения этой проблемы необходимо применять целый ряд мероприятий, включая:

– внедрение современных методов управления и контроля качества производства, которые позволят своевременно выявлять и устранять дефекты и недостатки;

– оптимизация производственных процессов и использование передовых технологий, которые позволят сократить время и затраты на производство РЭА;

– обучение и повышение квалификации персонала, чтобы они были владельцами необходимых знаний и навыков для успешной реализации производственных процессов;

– внедрение автоматизированных систем управления производством, которые обеспечат точность, надежность и оперативность контроля процессов и ресурсов.

Стремительное развитие радиоэлектронной аппаратуры характеризуется неизбежным увеличением сложности ее конструкций. Это обусловлено расширением спектра задач, решаемых данным оборудованием, и повышением требований к его эффективности. Схемотехнические и конструкторские решения становятся все более сложными, а функциональные связи становятся более интегрированными, что сопровождается значительным увеличением числа компонентов в РЭА. В результате производство такой аппаратуры становится сложным и вызывает значительные трудности, особенно при сборке, монтаже, наладке и регулировке.

Эти трудности являются объективным вызовом для производства РЭА, поскольку увеличение сложности и числа компонентов создает новые проблемы и требует разработки эффективных подходов к производственным процессам. Решение данных проблем требует внедрения передовых технологий, разработки новых методов сборки и монтажа, а также использования специализированного оборудования и инструментов для наладки и регулировки.

Относительная трудоёмкость производства сборочных единиц РЭА может быть представлена в таком соотношении: механическая обработка – 8...15%, сборка – 15...20%, электрический монтаж – 40...60%, наладка – 20...25% . Следовательно, основными технологическими задачами производства РЭА являются: механизация и автоматизация сборки и электрического монтажа модулей второго, третьего и четвёртого уровней; развитие автоматизированных и автоматических методов, а также средств наладки и регулировки аппаратуры сложных изделий; автоматизация операций контроля функциональных параметров; создание гибких комплексно-автоматизированных производств, функционирующих совместно с системами автоматизированного проектирования [8].

Процесс производства радиоэлектронной аппаратуры включает в себя большой комплекс взаимосвязанных работ, состав и последовательность выполнения которых зависят от специфики конструкций изделий и типа производства.

Для того чтобы определиться с порядком сборки радиоэлектронной аппаратуры и приборов, в первую очередь необходимо изучить их конструкцию.

Монтаж – это технический процесс (ТП) электрического соединения радиоэлектронной аппаратуры в соответствии с электрической или электромонтажной схемой. Для осуществления монтажа потребуются платы (печатные, проводные), одиночные проводники, жгуты, кабели [9].

Выполнение технологических операций при сборке радиоэлектронной аппаратуры (РЭА) всегда требует определенного порядка. Сам процесс сборки включает монтаж сборочных элементов и самого прибора. В зависимости от конкретных условий сборка РЭА может осуществляться стационарно или подвижно, с использованием концентрации или дифференциации манипуляций.

Сборка РЭА является технологическим процессом, который включает в себя соединение, координирование, фиксацию и монтаж всех сборочных единиц и деталей в пределах требуемой точности. Этот процесс предполагает выполнение ряда манипуляций, чтобы достичь готового и работоспособного РЭА.

При стационарной сборке РЭА важно обеспечить неподвижность самого процесса. Все необходимые материалы, инструменты и оборудование доставляются к месту сборки для проведения соответствующих работ.

Подвижная сборка РЭА осуществляется путем установки изделия на конвейер, по которому оно перемещается от одного рабочего места к другому. На каждом этапе сборки осуществляются необходимые манипуляции. Изделие может перемещаться принудительно вместе с конвейером или свободно продолжать движение после выполнения каждой манипуляции.

При разработке технологических процессов сборки и монтажа радиоэлектронной аппаратуры необходимо начать с изучения соответствующих материалов, включающих основные функциональные характеристики изделия, технические условия и требования, комплект конструкторской документации, программу производства и плановые сроки. Также важно ознакомиться с руководящими техническими, нормативными и справочными документами.

Основными документами, необходимыми для проведения сборки радиоэлектронной аппаратуры, являются технологические карты. В них указываются необходимые приспособления, инструменты и последовательность операций. Ручная сборка деталей или сборочных единиц выполняется опытными рабочими на специально оборудованных монтажных столах или конвейерах. После сборки деталей, сборочных единиц и блоков аппаратуры на основе платы (шасси) осуществляется электрический монтаж.

1.3 Автоматизация производства радиоэлектронной аппаратуры

В автоматизации производства РЭА есть ряд особенностей по сравнению с машино- и приборостроением, обусловленных сложностью РЭА и серийным характером производства. Из всей совокупности технологических процессов, применяемых в производстве РЭА, самыми сложными и наиболее трудоемкими являются процессы сборки, регулировки, комплексных испытаний. Автоматизация сборки в настоящее время охватывает часть технологических операций: подготовка электрорадиоэлементов (ЭРЭ) и микросхем к монтажу, установка их на платы, контроль и др.

Применение АСУТП помогает:

- повысить качество и надежность выпускаемых изделий;
- сократить потери от брака и увеличить процент выхода годных изделий;
- обеспечить управление производством с учетом заданного плана выпуска изделий;

- сократить трудоемкость изготовления;
- обеспечить контроль качества изделия на всех этапах технологического процесса и связь между различными уровнями производства.

Компания «*Eltex Alatau*» запустила линию поверхностного монтажа *SMD*. Сборочное производство полностью автоматизировано и представлено на рисунке 1 [10].

Монтажные линии *SMD* включают в себя:

- принтер трафаретной печати *MPM Momentum* для нанесения паяльной пасты на печатные платы в автоматическом режиме;
- автоматический установщик *SMD* компонентов на печатные платы *MG-1R*;
- система для конвекционного оплавления паяльной пасты фирмы *Speedline*;
- конвейерные системы *NUTEK*.

Рисунок 1 – Линия поверхностного монтажа *SMD* «*Eltex Alatau*»

Тульский производитель «*ExLab*» располагает 3 линиями автоматического монтажа, мощности которых при необходимости могут быть объединены для выполнения крупной серии одинаковых изделий. На рисунке 2 представлена «Линия 1» [11].

Основные характеристики линии автоматического монтажа:

– назначение: крупносерийное производство, монтаж печатных плат большого размера, монтаж светодиодов;

– производительность: до 42000 компонентов в час;

– установщик компонентов: *Yamaha YG100R + Assembleon Topaz XII*;

– нанесение паяльной пасты: автоматическое, точность 25 микрон;

– печь оплавления: Конвекционная 12-зонная *ERSA HotFlow*.

Рисунок 2 – Линия автоматического монтажа «*ExLab*»

1.4 Автоматизированное рабочее место оператора

Несмотря на значительный прогресс в автоматизации сборочных процессов в электронной индустрии, даже на самых передовых предприятиях пока еще невозможно полностью избежать ручного труда. Организации сталкиваются с проблемой, что развитие автоматизированных производственных участков приводит к неравенству в уровне производительности по сравнению с теми, где преобладает ручной труд.

Ручной труд сопряжен с присутствием человеческого фактора, который в радиоэлектронике, а также в других областях, стремятся минимизировать. Оперативный контроль является неотъемлемой частью эффективного производства в данном случае. Он необходим для обеспечения надлежащего качества и точности в процессе сборки радиоэлектронной аппаратуры.

Автоматизированное рабочее место (АРМ) позволяет снизить риск человеческой ошибки, обеспечить контроль качества изделия.

Под автоматизированным рабочим местом понимается комплекс программных и аппаратных средств, обеспечивающих нижеперечисленные задачи:

- учет рабочего времени;
- авторизация сотрудника;
- фотовидеофиксация операций;
- идентификация изделий по штрихкоду;
- учет операций и выработки;
- фиксация фактической трудоемкости.

Комплекс состоит из двух основных частей: рабочие места, система оперативного планирования и управления комплексом рабочих мест.

Рабочие места оснащены оборудованием для осуществления производственных операций, средствами автоматизации и т. д. Рабочие места формируют производственную цепочку, связанную с системой управления.

Комплекс позволяет распределить задачи между сотрудниками и контролировать выполнение производственной программы в режиме реального времени.

Таким образом, с использованием автоматизированного рабочего места (АРМ) все производственные операции, включая автоматизированные и ручные, могут быть отслеживаемыми в режиме реального времени, а также анализируются их эффективность. Это позволяет контролировать фактическое время выполнения отдельных операций и получать реалистичное представление о рабочем дне сотрудников. Полученные данные используются для более эффективного управления ресурсами предприятия и выявления моментов, требующих корректировки.

Автоматизированное рабочее место также решает задачу актуализации рабочей документации в режиме реального времени на всех рабочих местах, что позволяет избежать использования устаревшей документации и возможные ошибки при сборке изделий, связанные с этим. Бумажный документооборот устраняется, а информация обновляется автоматически, минимизируя вероятность использования устаревшей информации.

Применение программно-аппаратного комплекса освобождает большой временной ресурс у руководителя подразделения, который ранее был занят сбором и мониторингом информации. Теперь все данные поступают непосредственно с рабочих мест, где фиксация и отметка о выполнении операций происходят с минимальными затратами труда. Такая фиксация позволяет в дальнейшем создать цифровой паспорт для каждого изделия с полной информацией обо всех этапах производственного процесса, исполнителях, параметрах оборудования, внесенных изменениях и дате и времени выполнения работ. АРМ должно обеспечивать следующие функциональные требования:

1) идентификация и аутентификация;
2) учет использования оборудования;
3) видеофиксация этапов проведения работы;

4) перемещение образцов во входной/выходной накопители в рабочей зоне УРМ;

5) регистрация объектов;

6) отслеживание изделий по производственной цепочке;

7) отслеживание временных меток в базе данных (БД);

8) доступ к БД;

9) анализ результатов работы, фиксация результатов в БД СУД;

10) формирование протокола в БД СУД и на бумажном носителе;

11) получение в режиме реального времени полной информации о состоянии выполнения заказов и выполнения плановых показателей.

1.5 Анализ существующих аналогов систем оперативного мониторинга

В рамках дипломной работы был проведен анализ существующих комплексов, осуществляющих подобную деятельность.

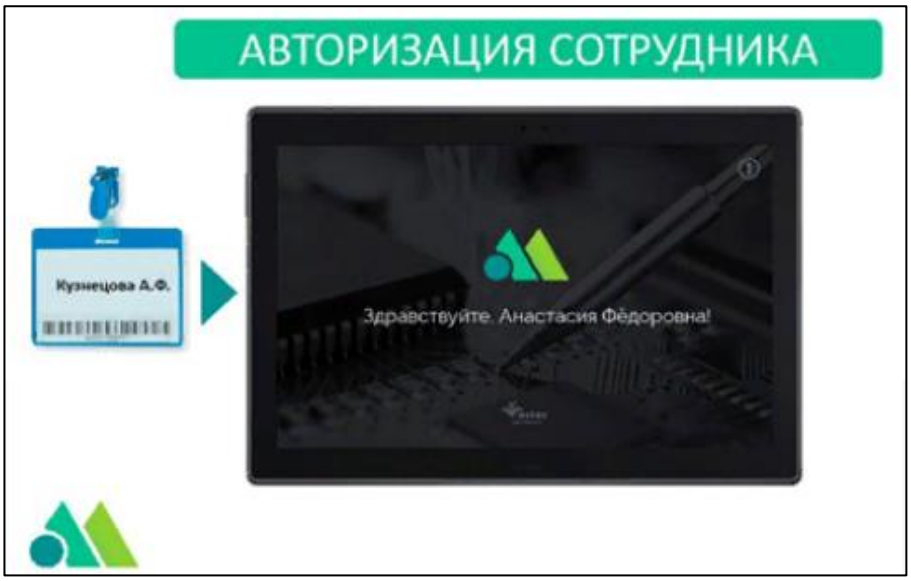
Для качественного изменения организации рабочих процессов на участках ручного труда в апреле 2018 года компания «Остек» предложила отрасли новое решение – программно-аппаратный комплекс «Умное рабочее место» (УРМ) (рисунок 3). УРМ охватывает все ручные операции: монтаж, сборку, настройку, контроль, ремонт и ведет пооперационный учет, контролируя последовательность выполнения этих операций. Это позволяет исключить брак, связанный с некорректной документацией на рабочих местах, а также сократить задержки на сбор старой, копирование и раздачу новой документации [12].

Рисунок 3 – «Умное» рабочее место

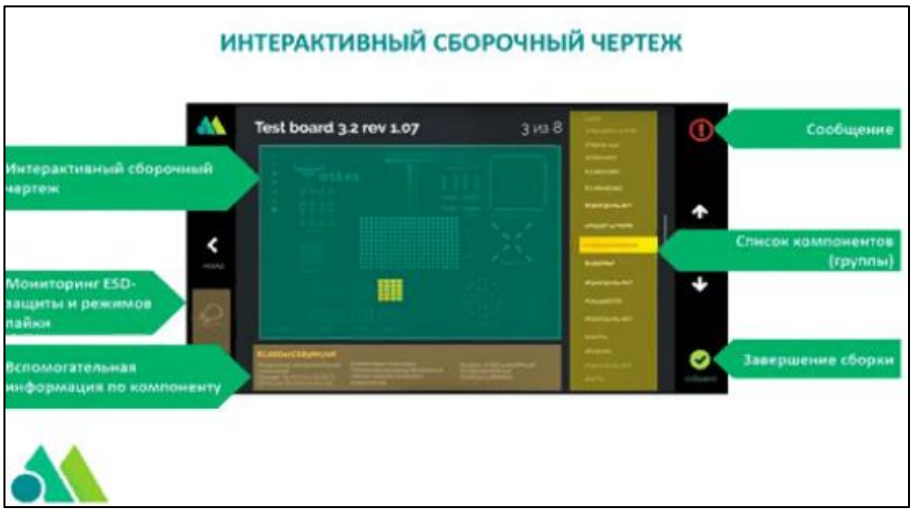
Каждый сотрудник, по завершении операции, вносит отметку в систему. Немедленно после нажатия кнопки, информация передается на сервер «Умного рабочего места» и поступает в учетную систему предприятия в режиме реального времени. Актуальная информация о выполненных операциях и собранных изделиях отображается наглядно на экранах, что позволяет отслеживать динамику производства и оперативно реагировать на отклонения от плановых показателей.

«Умное рабочее место» позволяет создать цифровой паспорт, который отражает информацию о том, какие сотрудники выполняли операции с конкретным изделием и когда они это делали. Этот комплекс успешно внедрен на нескольких российских производств, а также интегрирован с *MES*- и *ERP*-системами этих предприятий.

Интерфейс программного обеспечения «Умного рабочего места» представлен на рисунке 4.



(а)



(б)

Рисунок 4 – Элементы интерфейса программного обеспечения « Умного рабочего места»: окно «Авторизация сотрудника» *(а)*, окно «Интерактивный сборочный чертеж» *(б)*

Холдинг «Росэлектроника» госкорпорации «Ростех» разработал и ввел в опытную эксплуатацию умные рабочие места на базе собственной платформы промышленного интернета вещей *IIoT.Istok*. Решение позволяет отслеживать процесс сборки изделий в режиме онлайн, рассчитывать выработку, выполнять мониторинг текущей ситуации на производстве. На базе собранной информации система формирует цифровой образ каждого работника и

«цифровую тень» каждого изделия – набор данных о процессе его производства. На базе собранной информации система формирует цифровой образ каждого работника и «цифровую тень» каждого изделия – набор данных о процессе его производства [13].

Smart Workplace IIoT.Istok, разработанное НПП «Исток» им. Шокина холдинга «Росэлектроника», включает в себя программно-аналитический комплекс, планшетный компьютер и сканер штрих-кодов. После выполнения идентификации изделия по штрих-коду на умное рабочее место загружается конструкторская и технологическая документация. На экране планшета отображается 3*D*-модель изделия и перечень операций, которые должны быть последовательно выполнены работником.

Умные рабочие места *Smart Workplace IIoT.Istok* интегрированы в систему промышленного интернета вещей *IIoT.Istok*, которая обеспечивает контроль за технологическим и инженерным оборудованием, а также устанавливает зависимости в их работе. Платформа была внедрена в цехах НПП «Исток» им. Шокина в 2020 году. *IIoT.Istok* подключает инженерное оборудование, оборудование с числовым программным управлением (ЧПУ) и новую производственную линию для выпуска интеллектуальных приборов учета электроэнергии и устройств защиты при дуговом пробое. *IIoT.ISTOK* собирает информацию с встроенных в оборудование датчиков предприятия, что позволяет контролировать работу и состояние производственных линий в режиме реального времени, а также осуществлять мониторинг загрузки оборудования на разных уровнях – от конкретного рабочего места до завода в целом. Датчики, установленные на оборудовании, объединены в единую виртуальную сеть для создания централизованной системы мониторинга и управления технологическими процессами. Информация с датчиков выводится в центр управления производством. Оборудование самостоятельно обменивается информацией для оптимизации работы системы и обеспечения автономности, включая самодиагностику и самообслуживание. Кроме того,

система выявляет отклонения от выполнения управляющей программы в случае ручного воздействия на оборудование.

Система позволила на 10% снизить затраты на техническое облуживание, сократить эксплуатационные расходы, уменьшить количество простоев.

Государственная корпорация «Цифра» выпустила российскую систему мониторинга промышленного производства «Диспетчер». В данном комплексе сформирована продуктовая линейка из трех самостоятельных классов продуктов: *MDC, MES, EAM* [14].

Диспетчер *MDC* (*Manufacturing Data Collection*) представляет собой базовое *IIoT*-решение для контроля и аналитики производства. Это система мониторинга промышленного оборудования и администрирования программ на станках с ЧПУ, осуществляющая непрерывный сбор данных.

Система предоставляет возможность осуществлять удаленный мониторинг работы оборудования по локальной сети предприятия, посредством сети интернет. Возможен мониторинг, как станков с ЧПУ, так и универсального оборудования.

Станки с подключаются к существующей локальной сети предприятия, посредством специальных устройств – терминалов. Терминалы получают информацию о состояниях станков (работа по программе, простой, нет заготовок и пр.) и по сети передают ее на сервер для дальнейшего анализа и обработки. Оператор осуществляет мониторинг оборудования посредством специального ПО или через интернет-сервис.

Возможности системы:

– учет использования оборудования – автоматический сбор информации (дискретных и аналоговых сигналов) о состояниях работы оборудования с возможностью указания причин простоя;

– учет энергозатрат оборудования – регистрация во времени потребляемой станком полной и активной мощности, а также расчет затрат электроэнергии, как на каждую единицу оборудования, так и на выпускаемую продукцию;

– диспетчеризация служб – система позволяет автоматизировать оповещение и вызов служб (ремонтников, технологов и др.) для устранения простоев в работе оборудования и осуществить контроль за их работой;

– подсчет количества произведенной продукции – система позволяет вести автоматический учет количества произведенных деталей;

– передача программ на станки с ЧПУ по сети – система позволяет осуществлять централизованное хранение и передачу управляющих программ на станки с ЧПУ, а также управлять правами пользователей для доступа к УП;

– отчетность и аналитика – система проводит аналитику и формирует необходимые отчеты. На формирование необходимого отчета уходит не более 5 минут.

1.6 Актуальность, цель, задачи

Актуальность: обеспечение позаказного производства радиоэлектронной аппаратуры предполагает оперативный контроль производственных процессов. На рынке отсутствует техническое решение, которое способно обеспечить необходимые функциональные требования в условиях производства РЭА. В связи с этим, создание автоматизированной системы оперативного контроля позаказного производства радиоэлектронной аппаратуры является актуальной задачей, направленной на повышение качества и эффективности производства.

Цель: обеспечение позаказного производства РЭА посредством оперативного контроля отдельных рабочих мест и производственной цепочки в целом.

Задачи:

– разработка архитектуры автоматизированной системы оперативного контроля позаказного производства радиоэлектронной аппаратуры;

– разработка клиент-серверного программного обеспечения системы оперативного контроля позаказного производства радиоэлектронной аппаратуры;

– рассмотрение реализации автоматизированного рабочего места проведения испытаний навигационной аппаратуры потребителя с использованием предлагаемой системы оперативного контроля.

1.7 Выводы по главе 1

В ходе работы была рассмотрена проблематика радиоэлектронного производства и необходимость в его автоматизации. Основной целью работы было изучение возможностей и преимуществ автоматизации радиоэлектронного производства и определение наиболее эффективных решений в данной области.

Исследование различных методов и технологий автоматизации позволило определить наиболее подходящие инструменты, которые могут существенно повысить эффективность радиоэлектронного производства. Это включает в себя использование программных систем управления производством, автоматизированных систем сборки, роботизированных линий и применение принципов Индустрии 4.0.

Полученные результаты исследования показывают, что автоматизация радиоэлектронного производства имеет множество преимуществ, таких как повышение производительности, снижение затрат, улучшение качества продукции и сокращение времени производственного цикла. Она также позволяет улучшить контроль и мониторинг процессов производства, а также обеспечить высокую гибкость и адаптивность к изменяющимся требованиям рынка.

2 Система оперативного контроля позаказного производства радиоэлектронной аппаратуры

Автоматизированная система контроля позаказного производства радиоэлектронной аппаратуры (далее – Система оперативного мониторинга) предполагает программно-аппаратный комплекс, обеспечивающий контроль производства посредством реализации производственной цепочки.

2.1 Периферийные устройства Системы оперативного мониторинга

Периферийными устройствами Системы оперативного мониторинга являются:

– *RFID*-ридер;

– камера машинного зрения;

– камера для стереоскопического микроскопа.

При выборе средств, для регистрации изделия необходимо учитывать характеристики технологии (табл.1), которые подходят для конкретного типа применения.

Таблица 1 – Характеристики технологий регистраций изделий

Характеристики технологии	***RFID***	**Штрих-код**	***QR*-код**
Необходимость в прямой видимости метки	Чтение даже скрытых меток	Чтение без прямой видимости невозможно	Чтение без прямой видимости невозможно
Объём памяти	от 10 до 512000 байт	до 100 байт	до 3072 байт
Возможность перезаписи данных	Есть	Нет	Нет
Дальность регистрации	до 100м	до 4 м	до 1 м

Окончание таблицы 1

Характеристики технологии	***RFID***	**Штрих-код**	***QR*-код**
Одновременная идентификация нескольких объектов	До 200 меток в секунду	Невозможна	Зависит от считывателя
Устойчивость к воздействиям окружающей среды.	Повышенная прочность и сопротивляемость	Зависит от материала, на который наносится	Зависит от материала, на который наносится
Срок жизни метки	Более 10 лет	Зависит от способа печати и материала, из которого состоит отмечаемый объект	Зависит от способа печати и материала, из которого состоит отмечаемый объект
Безопасность и защита от подделки	Подделать невозможно	Подделать легко	Подделать легко
Работа при повреждении метки	Невозможна	Затруднена	Затруднена
Идентификация движущихся объектов	Да	Затруднена	Затруднена
Подверженность помехам в виде электромагнитных полей	есть	нет	нет
Идентификация металлических объектов	Возможна	Возможна	Возможна
Использование как стационарных, так и ручных терминалов для идентификации	Да	Да	Да
Возможность введения в тело человека или животного	Возможна	Затруднена	Затруднена
Габаритные характеристики	Средние и малые	Малые	Малые

RFID (*Radio Frequency IDentification*, радиочастотная идентификация) – это метод автоматического обнаружения объектов путем взаимодействия с *RFID*-метками посредством радиочастотных сигналов. Радиочастотный чип метки состоит из двух основных компонентов: антенны и самого чипа. Чип служит для хранения информации. Для запуска процесса необходимо использовать считыватель *RFID*-меток. Считыватель начинает передавать радиосигнал через антенну к пассивной метке. Под воздействием радиосигнала метка активизируется и получает электропитание, затем передает информацию, хранящуюся в чипе.

Сканер *RFID* меток – оборудование для считывания и перезаписи информации в *RFID* карту. Сканеры этого назначения постоянно подключены к учетной системе или работают в автономном режиме [15].

На предприятии целесообразно применять *RFID* для наибольшего сокращения времени сотрудника за рабочим местом.

Каждая радиочастотная метка включает в себя несколько компонентов: чип, антенну, приемник, передатчик и область памяти для хранения данных. В случае пассивного чипа требуется внешний радиосигнал от антенны считывателя или сигнал от встроенного источника питания для его активации. Антенна служит для приема электромагнитных волн от считывателя. После получения внешнего сигнала радиочип отвечает обратным импульсом, передавая свой идентификатор (*ID*).

После получения идентификатора, присвоенного метке, система определяет соответствующую информацию, которая будет загружена и отображена в интерфейсе программного обеспечения.

RFID считыватель настольный *RRU9816-USB/232* (рисунок 5) – это высокопроизводительный настольный *UHF* ридер, который, поддерживает быстрое чтение/запись метки с высокой скоростью идентификации. Может применяться во многих проектах, таких как логистика, контроль доступа, системы защиты от подделки, а также в системах управления производством [16].

Рисунок 5 – *RFID* считыватель настольный *RRU9816-USB/232*

Технические характеристики данного считывателя представлены в таблице 2.

Таблица 2 – Технические характеристики *RFID* считывателя *RRU9816*

Параметр	**Значение**	**Единица**
Питание	6	*V*
Рабочая температура	от -10 до +70	°*C*
Температура хранения	от -20 до +85	°*C*
Частота	865-868	МГц
Мощность	26	*dBm*

Royal Ray Настольный *UHF* ридер *RRU*1861 *USB* (рисунок 6) – настольное устройство чтения/записи представляет собой современное, компактное и удобное устройство, предназначенное для работы с *RFID*-метками. Благодаря небольшому размеру считыватель идеально подходит для отслеживания документов на предприятиях, в архивах, библиотеках, магазинах розничной торговли и не только [17].

Рисунок 6 – *UHF* ридер *RRU*1861 *USB*

Технические характеристики данного ридера представлены в таблице 3.

Таблица 3 – Технические характеристики *UHF* ридера *RRU*1861 *USB*

Параметр	**Значение**	**Единица**
Питание	6	*V*
Рабочая температура	от -10°*C* до +60°*C*	°*C*
Температура хранения	от -25°C до +80°C	°*C*
Частота	866-868	МГц
Мощность	30	*dBm*

Считыватель *CL*7206*A*2 (рисунок 7) предназначен для оборудование мест выдачи пропусков или складов. Позволяет считывать любую информацию с *RFID* меток или карт и производить запись во все сектора памяти меток, доступных для записи [18].

Рисунок 7 – Считыватель *CL*7206*A*2

Технические характеристики данного считывателя представлены в таблице 4.

Таблица 4 – Технические характеристики считывателя *CL*7206*A*2

Параметр	**Значение**	**Единица**
Питание	6	*V*
Рабочая температура	от -20°*C* до +70°*C*	°*C*
Температура хранения	от-40°C ~ +85°C	°*C*
Частота	865-868	МГц
Мощность	20-25	*dBm*

В сравнительной таблице 5 представлены возможные считыватели *RFID* меток, которые ставятся непосредственно на рабочее место.

Таблица 5 – Сравнительная таблица *RFID*-считывателей

	RRU9816	***RRU1861DK3324 USB***	***CLOU CL7206A2***
Диапазон частот	865–868 МГц	866–868 МГц	865–868 МГц
Протокол	*ISO18000-6C (EPC C1G2)*	*EPC Class 1 Gen 2, ISO 18000-6C, IEC18000-6B*	
Дальность чтения	до 50см (в зависимости от используемых меток)	до 30 см	10 см
Питание	по *USB* 6 В, *VCC* (внешний) 9В	*USB* 6В, *VCC* (внешний) 9В, 300мА	*USB Type-C*
Рабочая температура	от -10 до 70	от -10°*C* до +60°*C*	от -20°*C* до +70°*C*
Температура хранения	от -20 до 85	от -25°C до +80°C	от-40°C ~ +85°C

Для автоматизированного рабочего места наиболее подходящий вариант – настольный *UHF RFID*-считыватель *Royal Ray RRU1861DK3324 USB.*

Универсальный драйвер ЭРФИД – Программное обеспечение для управления *RFID UHF* считывателями. Полностью интегрирован с программными продуктами компании 1С. Работает с любым *RFID*-оборудованием и с любой системой учета через простой открытый *API* [19]. Выбранный считыватель входит в перечень поддерживаемых моделей.

RFID карта состоит из секторов и блоков. Необходимо поднести *RFID* карту к считывателю. В интерфейсе ПО отображаются сектора и блоки. В нулевом блоке хранится служебная информация. Эта информация не перезаписываемая и прошита производителем. В первых 4 байтах находится серийный номер карты, а в оставшихся байтах информация о производителе (рисунок 8).

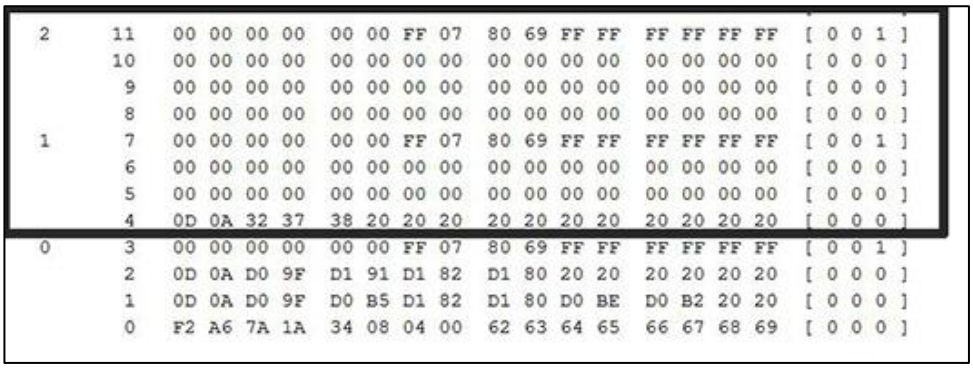


Рисунок 8 – Реализация работы *RFID* считывателя

В первом блоке находятся нули, это говорит о том, что там нет информации. Её также нет и во втором и четвёртом блоке. Эти блоки используются для записи и хранения информации [20].

Камеры для систем обработки изображений подразделяются на камеры машинного зрения, сетевые (*IP*) – и *WEB*-камеры. Основные отличия отображены в таблице 6 [21]:

Таблица 6 – Отличия средств визуального контроля

	***WEB*-камеры**	***IP*-камеры**	**Камеры машинного зрения**
Сжатие изображения с потерей данных	Да	Да	Нет
Задача	Первичное тестирование программ компьютерного зрения	Чаще всего видеонаблюдение	Решение задач оптического контроля
Обработка изображения	Путем визуального контроля		С помощью программного обеспечения
Возможность подбора оптимальных компонентов	Нет	Чаще всего нет	Да
Встраиваемость системы	Нет	Нет	Да

Если говорить о решении задач оптического контроля, об обработке изображений с помощью специализированного ПО и в целом о получении достоверных данных с изображения, то необходимо использовать камеры машинного зрения. Именно отсутствие сжатия гарантируют получение полной информации об объекте съёмки и выгодно отличает камеры машинного зрения от *IP* и *WEB* камер [22]. Лидеры мирового рынка поставщиков решений для машинного зрения представлены на рисунке 9.

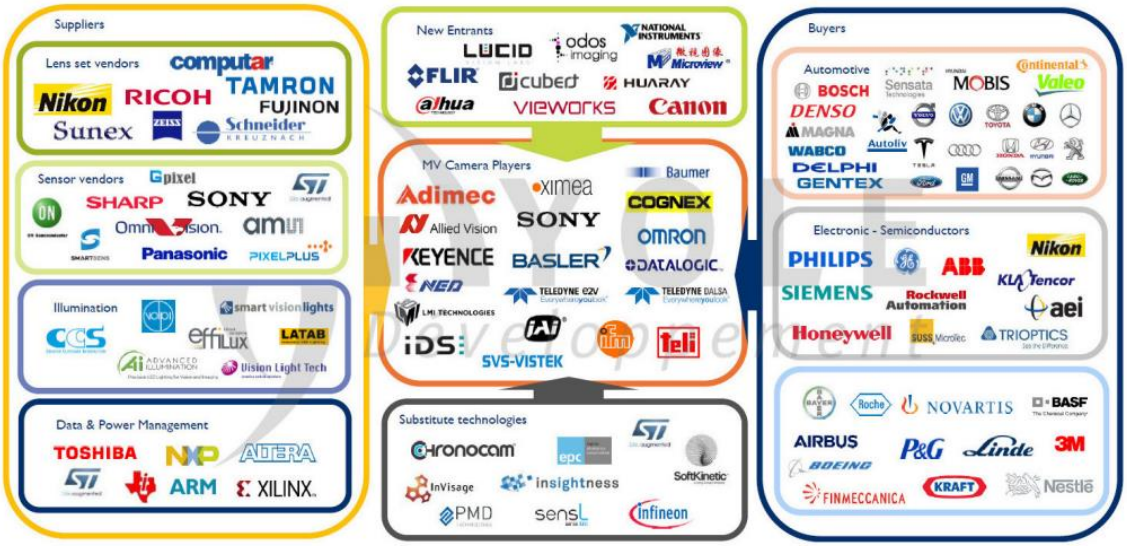


Рисунок 9 – Лидеры мирового рынка поставщиков решений для машинного зрения

Российские производители решений для машинного зрения [23]:

– *VisionLabs* – компании занимаются распознаванием лиц и компьютерным зрением. Разрабатывают продукты для финансовой сферы, ритейла, видеонаблюдения и безопасности;

– *Yandex Data Factory* – подразделение компании «Яндекс», которое специализируется на анализе больших данных и применении технологий машинного обучения для решения задач промышленности;

– *Cognitive Technologies* – российская компания в области разработки и внедрения программного обеспечения, которая разрабатывает системы машинного зрения и обработки изображений;

– *Synesis* – Разработчик систем интеллектуального видеонаблюдения и бизнес-аналитики на основе компьютерного зрения;

– АО «Элвис-НеоТек» – Ведущий разработчик и производитель высокотехнологичных систем безопасности с применением технологий распознавания образов, компьютерного зрения, радиолокационного, видео, тепловизионного наблюдения.

Довольно широко распространено ошибочное мнение, что использование простых и дешёвых *IP* или *WEB* камер позволит решить промышленные задачи машинного зрения.

В действительности же существует ряд требований, выдвигаемых к камерам машинного зрения на любом производстве [22]:

– надёжность, качество сборки, высокая отказоустойчивость;

– качество изображений*;*

– высокая чувствительность сенсора*;*

– наличие цифровых линий для подключения внешних устройств;

– промышленные интерфейсы*;*

– наличие средств разработки для различных языков программирования;

– промышленное исполнение, защита от воды и пыли, виброустойчивость, высокий температурный диапазон.

При выборе камеры необходимо учитывать требования поставленной задачи и тип изображения, которое требуется получить. Цветное изображение подходит для оценки определенных характеристик, спектральное изображение помогает выявить сложные параметры, а монохромное изображение может быть достаточным.

Если цвет не является обязательным условием, рекомендуется выбрать монохромную камеру, так как они обладают более высокой чувствительностью и пространственным разрешением.

При выборе затвора необходимо определить его тип: глобальный или скользящий. Глобальный затвор полностью открывается, позволяя свету достигнуть всей поверхности сенсора. Это оптимальный выбор для съемки быстро движущихся объектов.

Скользящий затвор экспонирует матрицу построчно. В случае перемещения объекта во время экспозиции могут возникать искажения на изображении, известные как «эффект плавающего затвора». Однако камеры со скользящим затвором обычно имеют более доступные цены.

Подведя итог, для автоматизированного рабочего места нужна монохромная камера с глобальным затвором и интерфейсом *USB*.

Для выбранных характеристик у российской организации «*Camera IQ*» модели представлены в таблице 7.

Таблица 7 – Модели камер машинного зрения

Модель	**Модель сенсора**	**Разрешение**	**Размер пикселя**	**Скорость съемки**	**Интерфейс**
MV-CS004-10UM	*Sony IMX287*	720 × 540	6,9 мкм	526,5 к/с	*USB* 3.0
MV-CS016-10UM	*Sony IMX273*	1440 × 1080	3,45 мкм	249,1 к/с	*USB* 3.0
MV-CS028-10UM	*Sony IMX421*	1936 × 1464	4,5 мкм	121,1 к/с	*USB* 3.0
MV-CS050-10UM	*Sony IMX264*	2448 × 2048	3,45 мкм	60 к/с	*USB* 3.0

Для автоматизированного рабочего места подходящая модель – *MV-CS*050-10*UM* (рисунок 10).

Рисунок 10 – Камера машинного зрения *MV-CS*050-10*UM*

Для интеграции используется программное обеспечение *MVTec* – универсальное программное обеспечение для машинного зрения [24].

При выборе камеры следует обратить внимание на ее технологию, поскольку на рынке преобладают два типа сенсоров: *CMOS* (комплементарный металлокислородный полупроводник) и *CCD* (зарядовая связь). Эти сенсоры обеспечивают существенно отличающиеся изображения. В *CMOS*-матрицах процесс цифрового преобразования аналогового изображения происходит в каждом чувствительном элементе, что позволяет сделать это быстрее. Однако, в результате получается более «шумное» изображение, возможны искажения цветов. В *CCD*-матрицах аналоговый сигнал преобразуется в цифровой на выходе каждого пикселя. Плотность фотодиодов в *CCD* значительно выше, и больше света участвует в формировании изображения. Поэтому динамический диапазон у камер с *CCD*-матрицами значительно выше, чем у их конкурентов.

Однако *CMOS*-матрицы обладают более высокой скоростью съемки, поэтому их применяют в высокоскоростных камерах. Какой бы чувствительной камера ни была, но она не сможет дать достойную детализацию, если необходимо отобразить множество мелких деталей. Чем меньше увеличение, тем больше нужно пикселей [25]. В сравнительной таблице 8 представлены возможные камеры на микроскоп.

Таблица 8 – Сравнительная таблица камер на микроскоп

	NIKON DS-FI3	***NIKON DS-RI2***	***NIKON DS-QI2***
Сенсор	Цветной КМОП сенсор Количество пикселей: 5,9 Мегапикселей (2880 × 2048 пикселей)	Формат *Nikon FX*, цветная КМОП (*CMOS*) матрица/ размер 36,0 × 23,9 мм Количество эффективных мегапикселей – 16,25	Формат *Nikon FX*, монохромная КМОП (*CMOS*) матрица/ размер 36,0 × 23,9 мм Количество эффективных мегапикселей – 16,25
Эффективное количество пикселей	Максимальное разрешение: 2880 × 2048 пикселей Усредненный режим «2 x 2 пикселя»: 1440 × 1024 пикселя	4908 × 3264, 1636 × 1088 (3 × 3 пикселя)	4908 × 3264, 1636 × 1088 (3 × 3 пикселя)
Размер и скорость получения изображения	Максимальное разрешение (2880 × 2048 пикселей): 15 кадров/сек Усредненный режим «2 × 2 пикселя»: (1440 × 1024 пикселя): 30 кадров/сек	4908 × 3264 (макс. 6 кадров/сек), 1636 × 1088 (макс. 45 кадров/сек) 3 × 3 пикселя	4908 × 3264 (макс. 6 кадров/сек), 1636 × 1088 (макс. 45 кадров/сек) 3 × 3 пикселя
Время экспозиции	От 100 мкс до 30 сек	100 мксек – 60 сек	100 мксек – 60 сек
Посадочный размер	*C-mount*	*F-mount*	*F-mount*
Интерфейс	*USB*3.0 (соединение с ПК) × 1, внешний триггер × 1	*USB*3.0	*USB*3.0
Энергопотребление	Максимальное потребление 4,8 Вт	13 Вт	24 Вт

Для стереоскопических микроскопов с тринакулярной головкой для автоматизированного рабочего места подходит камера *NIKON DS-FI3* (рисунок 11).

Рисунок 11 – Камера *NIKON DS-FI3*

Просмотр и обработка изображений доступны в программном обеспечении *NIS-Elements*.

NIS-Elements – это интегрированная программная платформа, разработанная компанией *Nikon*, которая позволяет полностью управлять микроскопом, регистрацией изображений, документированием, обработкой и анализом данных.

NIS-Elements безупречно справляется с комплексными задачами документирования, такими как регистрация, визуализация изображений, контроль периферийных устройств, управление данными и анализ многомерных изображений (в том числе шестимерных). Эта система также повышает эффективность исследований, поскольку имеет функцию построения базы данных, созданную для архивирования, поиска и анализа большого количества файлов, содержащих многомерные изображения [26].

2.3 Архитектура системы

Изъято содержимое раздела 2.3 (12 страниц).

2.4 База данных

Изъято содержимое раздела 2.4 (5 страниц).

2.5 Выводы по главе 2

В данной главе был осуществлен комплексный анализ и выбор необходимого оборудования для реализации автоматизированной системы оперативного контроля позаказного производства радиоэлектронной аппаратуры. При выборе оборудования учитывались его технические характеристики и соответствие требованиям производства.

Также был проведен анализ различных языков программирования и выбран подходящий для разработки программы, входящей в состав программно-аппаратного комплекса. Основные критерии выбора включали возможности языка, его распространенность, поддержку сообществом разработчиков и интеграцию с другими компонентами системы.

Для улучшения понимания архитектуры программы и взаимодействия ее компонентов были построены *UML*-диаграммы. Они позволяют наглядно представить структуру программы, взаимосвязи между классами и последовательность их взаимодействия.

Таким образом, выполненный анализ и выбор оборудования, а также определение языка программирования и построение *UML*-диаграммы сыграли важную роль в проектировании и разработке системы оперативного контроля позаказного производства радиоэлектронной аппаратуры. Эти шаги обеспечили основу для дальнейшей реализации и интеграции системы, что позволит достичь требуемого уровня автоматизации и эффективности производства.

3 Реализация Системы оперативного мониторинга

На различных этапах производства контроль качества радиоэлектронной аппаратуры играет важную роль благодаря программным и аппаратным средствам. Они выполняют измерения, проводят тестирование и анализируют характеристики продукции, что способствует выявлению и предотвращению возможных дефектов и отклонений от стандартов качества. Такие программные и аппаратные средства могут быть интегрированы с другими компонентами системы, такими как роботизированные системы и автоматизированные линии сборки и тестирования. Это позволяет достичь высокой производительности и повысить эффективность процесса позаказного производства.

Проведение экспериментальных исследований опытных образцов цифровой угломерной навигационной аппаратуры потребителя реализуется в соответствии с таблицей 12.

Таблица 12 – Программа экспериментальных исследований

Вид исследований (проверок)	**Ед. изм.**	**Номинальное значение**	**Предельные отклонения**
Проверка ПД на соответствие установленной комплектности и оценка её качества	–	Соотв.	–
Проверка соответствия комплекса его ПД	–	Соотв.	–
Проверка работоспособности формирователя радионавигационного сигнала в диапазоне $L1$ от 2 НКА	–	Соотв.	–
Проверка работоспособности формирователя радионавигационного сигнала в диапазоне $L1$ от 4 НКА	–	Соотв.	–
Проверка работоспособности формирователя радионавигационного сигнала в диапазоне $L1$ от 6 НКА	–	Соотв.	–

Окончание таблицы 12

Вид исследований (проверок)	**Ед. изм.**	**Номинальное значение**	**Предельные отклонения**
Оценка уровня формируемых радио-навигационных сигналов	дБВт	Соотв.	–145; –165
Проверка измерения трех углов пространственной ориентации	–	Соотв.	–
Проверка диапазона измеряемых углов	Угл. град.	Соотв.	0°; 360°
Проверка метода измерения фазовых сдвигов радионавигационных сигналов	Угл. град.	Соотв.	5°
Проверка метода определения углов пространственной ориентации на основе измеренных фазовых сдвигов	Угл. мин.	Соотв.	10'
Проверка метода разрешения фазовых неоднозначностей в принимаемых радионавигационных сигналах	–	Соотв.	–
Выбор метода расчета эфемеридно-временной информации по критерию максимума количества одновременно наблюдаемых НКА из точек стояния КА на ГСО и КА на НО	–	Соотв.	–
Определение точности позиционирования и измерения ориентации КА выбранных методов расчета эфемеридно-временной информации на временном интервале автономного функционирования не менее 10 суток.	–	Соотв.	–

Рассмотрим реализацию предлагаемой Системы оперативного мониторинга на примере задачи автоматизации процесса проведения испытаний цифровой угломерной навигационной аппаратуры потребителя.

3.1 Объект испытаний

Объектом испытаний является перспективная навигационная аппаратура потребителя (НАП), в состав которой входят: антенная система из трех микрополосковых антенн, приемный модуль МРК-101, программное обеспечение приемного модуля, соединительные кабели, эксплуатационная документация (рисунок 25).

Рисунок 25 – Перспективная НАП МРК-101 с антенной системой из трех микрополосковых антенн

Конструкция НАП МРК-101 включает следующие компоненты: фильтр питания, модуль источника питания, платы радиочастотного модуля, аналого-цифрового преобразователя (АЦП) и цифрового вычислителя, плата синтезатора частот и средства индикации. Все компоненты опытного образца ГНСС-приемника размещены в герметичном корпусе из алюминиевого сплава размером 194×175×60 мм. Вес составляет 1,5 кг. Потребляемая мощность НАП составляет 25 Вт.

НАП МРК-101 предназначена для использования в транспортных средствах (автомобилях, судах, самолетах), системах безопасности, системах управления подвижными объектами, бортовых системах автоматического управления беспилотными летательными аппаратами и других областях. Кроме того, НАП может использоваться как самостоятельное устройство или в составе различных навигационных комплексов и систем управления.

Общее количество каналов приема радиосигналов равно 96. Распределение каналов приема радиосигналов осуществляется в автоматическом режиме.

Время первого определения навигационных параметров НАП составляет:

– в режиме «горячего» старта – не более 30 с;

– в режиме «холодного» старта – не более 90 с.

Частота определения навигационных параметров – не менее 1 Гц.

Навигационная аппаратура потребителя МРК-101 обеспечивает следующие точностные характеристики определения значений навигационных параметров и углов пространственной ориентации:

– определение плановых координат с СКО 5 м, не более;

– определение составляющих вектора скорости с СКО 0,1 м/с, не более;

– определение углов пространственной ориентации с СКО 0,1°, не более – по углу курса, 0,2°, не более – по углам крена и дифферента.

Предельные погрешности определения НАП МРК-101 пространственной ориентации антенной системы при работе по реальному навигационному полю, не должны превышать:

а) для базы между составляющими антенной системы, равной 2 м:

– угла курса – 10 угловых минут;

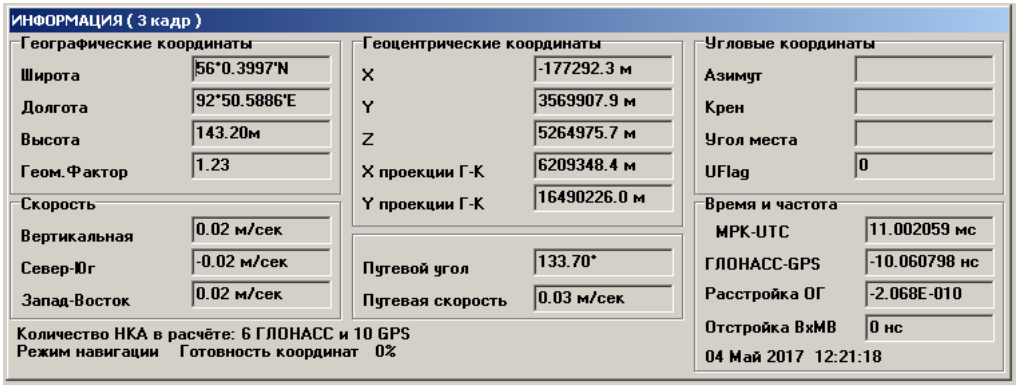
– углов крена и тангажа – 16 угловых минут;

б) – для базы между составляющими антенной системы, равной 1 м:

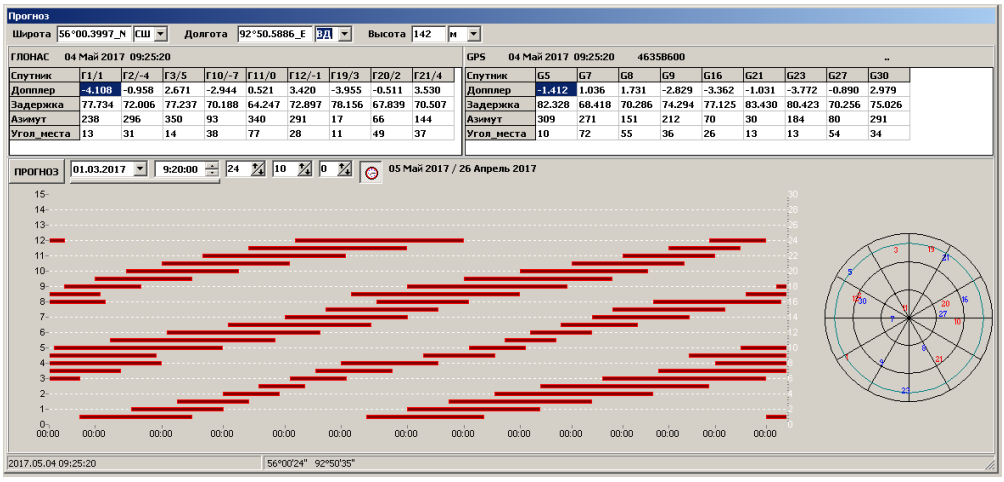
– угла курса – 15 угловых минут;

– углов крена и тангажа – 25 угловых минут.

К НАП МРК-101 разработчиком прилагается сервисное программное обеспечение, элементы интерфейса которого представлены на рисунке 26, предназначенное для управления НАП, ввода первоначальных настроек, ввода и вывода навигационных параметров и т. д.



(*а*)



(*б*)

Рисунок 26– Элементы интерфейса сервисного программного обеспечения НАП МРК-101: окно «Информация» (*а*), окно «Прогноз радиовидимости» (*б*)

Встроенное программное обеспечение СБИС СнК НАП предназначено для вычисления эфемерид навигационных космических аппаратов, расчета вектора состояния объекта и углов пространственной ориентации в составе аппаратуры МРК-101. Программное обеспечение обеспечивает управление НАП по бинарному протоколу *VIN* УЭ0.506.010-02Д26 и выдачу навигационных параметров потребителю по протоколам *VIN* УЭ0.506.010Д26 или *NMEA*-0183.

3.2 Средства автоматизации проведения испытаний навигационной аппаратуры потребителя

Программно-аппаратный комплекс *PXI-GNSS* обеспечивает возможность эмуляции сигналов от спутниковых систем ГЛОНАСС и *GPS* с настраиваемыми параметрами орбит и положения приемника. Благодаря гибкому программному обеспечению этот комплекс позволяет легко тестировать несколько стандартов спутниковой навигации, включая *GPS, GLONASS* и другие, на базе одной ВЧ-платформы *PXI*.

Графический интерфейс программного обеспечения системы прост в использовании и предоставляет возможности для тестирования и проверки *GPS*/ГЛОНАСС-приемников. С помощью этого инструмента можно формировать коды грубого позиционирования (*C*/*A*) с использованием более 12 спутников в полосе *L1*.

Программный интерфейс также обладает функциями формирования сигналов для определения местоположения и скорости приемника. Он также позволяет непрерывно генерировать поток неповторяющихся навигационных сигналов продолжительностью до 24 часов, что позволяет проводить более глубокие испытания для проверки надежности тестируемых устройств. Кроме того, предусмотрены возможности внесения помех в сигнал или ухудшения его качества.

Используя программируемое измерительное оборудование, можно создавать собственные периодические тесты с формированием сигналов, зависящих от траектории навигационного приемника, без необходимости проведения дорогостоящих испытаний с контролем движения. Кроме того, с помощью комплекса *PXI-GNSS* можно регулировать мощность генерируемых сигналов для проверки динамического диапазона входного приемного устройства или для проведения тестирования с заданными сценариями изменения параметров навигационного сигнала.

Комплекс *PXI-GNSS* позволяет регистрировать такие параметры приемников как чувствительность, время выхода в рабочий режим *TTFF* (в режиме «холодного», «теплого» и «горячего старта»), точность определения местоположения (абсолютную и относительную) в статическом (приемник неподвижен) и динамическом режимах (с заданием траекторий движения для приемника) [29].

1. Измерение чувствительности:

– минимальная выходная мощность имитатора *PXI-GNSS* -165 дБм;

– погрешность выходной мощности ±1.0 дБ вплоть до -127 дБм;

– измерение коэффициента шума в режимах моделирования одного и нескольких спутников.

2. Измерение времени выхода в рабочий режим (*TTFF*):

– одновременное моделирование 12 спутников для определения приемником своей позиции;

– моделирование сигналов общей длительностью до 72 часов, необходимое для длительной проверки работоспособности приемника и регистрации битовых ошибок *BER*;

– управление выходной мощностью сигналов со спутника, формирование сдвига частоты по Доплеру и информации о дальности.

3. Измерение погрешности определения местоположения в статическом и динамическом режимах:

– одновременное моделирование нескольких спутников для определения приемником своих координат;

– задание координат спутника в системах *ECEF* и *LLA* для определения точности позиционирования;

– режим задания траекторий движения приемника с постоянной скоростью или с ускорением.

Комплекс *PXI-GNSS* предоставляет возможность записи и воспроизведения реальных навигационных сигналов, передаваемых по эфиру. Этот метод лабораторного тестирования навигационных приемников является простым и экономически выгодным решением, исключающим необходимость затратного процесса формирования спутниковых навигационных сигналов.

Использование функции записи/воспроизведения позволяет анализировать воздействие внешних излучений на навигационный сигнал, а также на функциональные характеристики приемника. Записав однократно продолжительный поток навигационных сигналов в различных реальных условиях приема (в движении, в хорошей или плохой погоде, на открытой или частично закрытой местности), вы сможете провести лабораторное тестирование *GNSS*-приемника в условиях, максимально приближенных к реальным рабочим условиям.

При наличии нескольких векторных генераторов и анализаторов сигналов в одной системе, может также осуществляться многодиапазонная запись и воспроизведение сигналов (Например *L1* и *L2* для *GPS*, ГЛОНАСС и т. д.)

Характеристики:

– диапазон частот записываемых/воспроизводимых сигналов от 9 кГц до 6.6 ГГц;

– до 72 часов записи/воспроизведения навигационных сигналов с использованием внешнего накопителя объемом 3 ТБ *NI* 8264;

– запись навигационных сигналов в мгновенной полосе 80 МГц;

– динамический диапазон свободный от гармоник более 80 дБ: позволяет анализировать интерференционную картину в спектре, а также определять уровни слабых сигналов в присутствии мощных спектральных составляющих;

– собственный уровень шума порядка -165 дБм;

– скорость записи данных на внешний накопитель 600 МБ/с.

Система имитации, записи и воспроизведения *GPS*-сигналов представлена на рисунке 27.

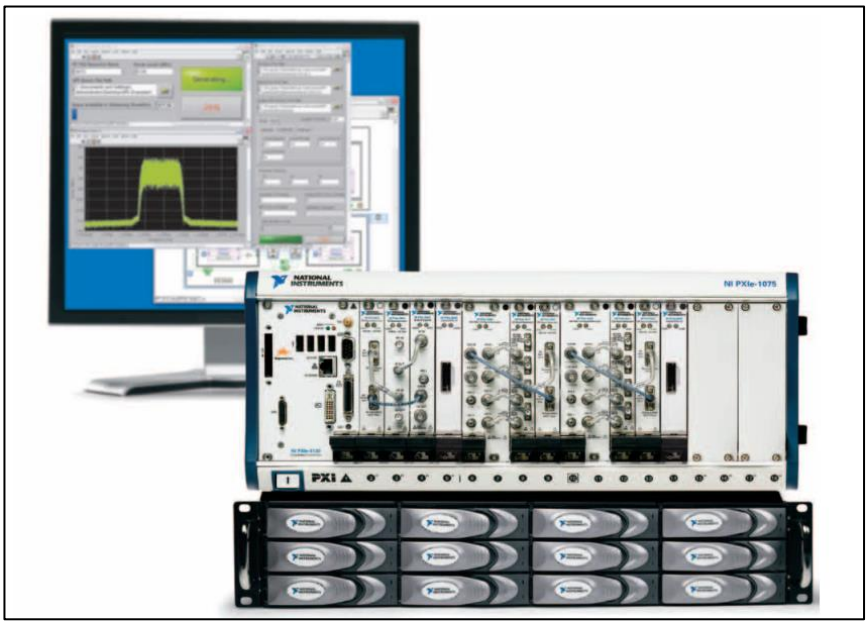


Рисунок 27 – Система имитации, записи и воспроизведения *GPS*-сигналов

Данная конфигурация комплекса *PXI-GNSS* позволяет тестировать приемное навигационное оборудование сразу в двух режимах: режим записи и воспроизведения сигналов в реальных условиях, режим имитации.
Состав системы представлен в таблице 13.

Таблица 13 – Состав системы имитации, записи и воспроизведения *GPS*-сигналов

Оборудование	**Характеристики**	**Количество**
NI PXIe-1075	Шасси *PXI Express* на 18 слотов	1
NI PXIe-8108	Двухъядерный контроллер с ОЗУ 4 ГБ	1
NI PXIe-5673 или *PXIe-5672*	Векторный генератор сигналов с ОЗУ 512 МБ	от 1 до 2
NI PXIe-5661 или *NI PXIe-5663*	Векторный анализатор сигналов с ОЗУ 512 МБ	от 1 до 2

Окончание таблицы 13

Оборудование	**Характеристики**	**Количество**
NI PXI-5695	Двухканальный программируемый ВЧ–аттенюатор с рабочим диапазоном частот от 50	1
NI PXI-5691	Двухканальный программируемый усилитель ВЧ-сигналов с диапазоном частот от 50 МГц до 8 ГГц	1
NI HDD-8264	Внешний *RAID*-массив данных объёмом 3 ТБ для чтения и записи до 72 часов моделированных или реальных *GPS*/ГЛОНАСС сигналов	1
Программное обеспечение		
NI LabVIEW	Профессиональная среда разработки	1
NI Modulation Toolkit	Библиотека функций для модуляции и формирования сигналов	1
NI GPS Toolkit	Библиотека функций для имитации спутниковых *GPS*-сигналов	1
NI GLONASS Toolkit	Библиотека функций для имитации спутниковых *GLONASS*-сигналов	1

На рисунке 28 представлена расширенная структурная схема Системы оперативного контроля позаказного производства радиоэлектронной аппаратуры. Расширенная схема иллюстрирует компоненты и взаимосвязи внутри системы контроля. Она включает в себя различные модули и компоненты, отвечающие за сбор и анализ данных, контроль качества, отслеживание стадий производства и взаимодействие с другими системами и подразделениями предприятия.

Расширенная структурная схема Системы оперативного контроля позаказного производства радиоэлектронной аппаратуры помогает визуализировать организацию и взаимодействие компонентов этой системы, а также понять ее функциональные возможности. Она является важным инструментом для понимания работы системы контроля и управления в рамках данного производственного процесса.

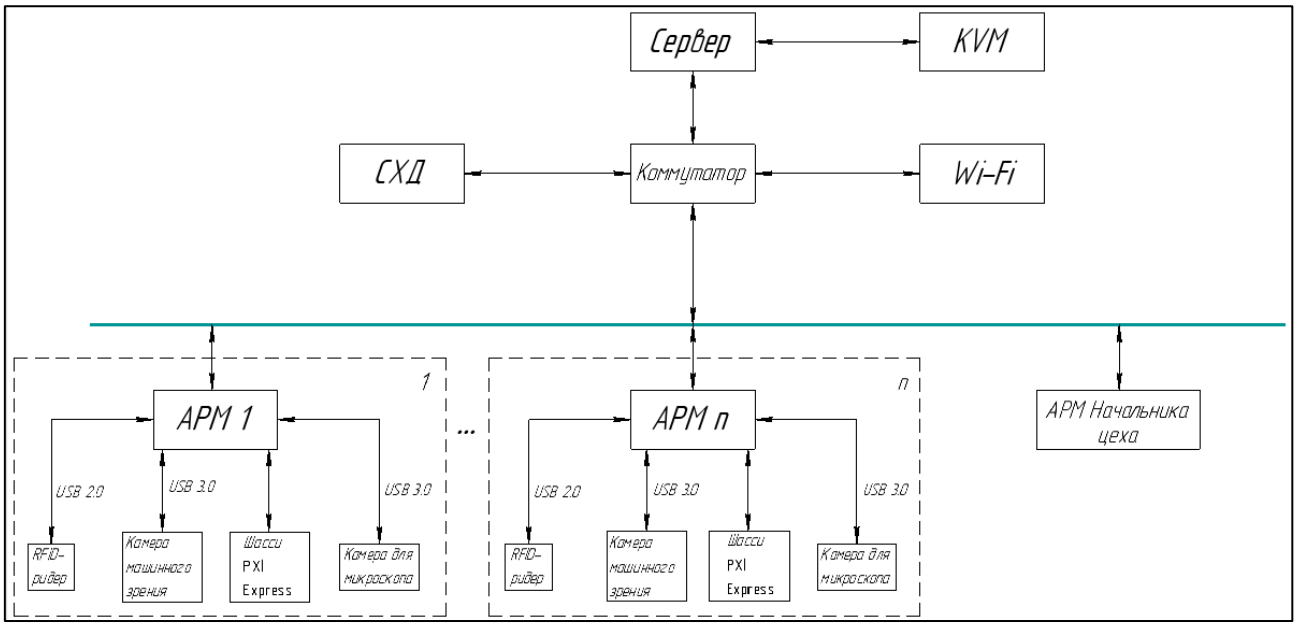


Рисунок 28 – Расширенная структурная схема Системы оперативного мониторинга

В отличии от структурной схемы Системы оперативного мониторинга (рис. 24), расширенная структурная схема включает в себя шасси *PXI Express*, в состав которого входят:

– двухъядерный контроллер *NI PXIe-8108*;

– векторный генератор сигналов *NI PXIe-5673*;

– векторный анализатор сигналов *NI PXIe-5661*;

– двухканальный программируемый ВЧ–аттенюатор *NI PXI-5695*;

– двухканальный программируемый усилитель ВЧ-сигналов *NI PXI-5691*;

– внешний *RAID*-массив *NI HDD-8264*.

Генерация *GPS*-сигналов осуществляется посредством использования программного обеспечения, общие характеристики которого представлены в таблице 14.

Таблица 14 – Общие характеристики программного обеспечения

Параметры	**Значения**
Диапазоны для формирования кода позиционирования	*L1*, *L2* и *L5*
Моделирование спутников	Автоматическое, на основе файлов расположения ИСЗ
Число одновременно моделируемых спутников	до 12
Контроль частоты смещения Доплера на отстройке	±600 кГц
Контроль уровня мощности во время генерации сигнала в пределах	-159 дБм до + 25 дБм, с разрешением 0,02 дБ
Формирование пользовательских траекторий движения для моделирования работы мобильных устройств	
Готовые маршруты движения	загрузка карт *google maps*
Собственный маршрут движения	загрузка скриптов
Задание вектора направления движения	по начальному месту положения (широта, долгота, высота), скорости и ускорению
Максимальная скорость движения	свыше 514 м/с
Длительность непрерывного воспроизведения сигналов	до 24 часов
Регистрация времени выхода в рабочий режим *TTFF*	

Возможности имитации навигационных сигналов в системе представлены в таблице 15.

Таблица 15 – Основные возможности имитации навигационных сигналов

Основные возможности	
Поддерживаемые *GNSS*	*GPS, GLONASS*
Тип кода	*C/A*
Диапазоны формирования кода	*L1, L2* (генерация несущей + возможность конфигурирования битовой последовательности под задачи пользователя)
Встроенные сценарии генерации сигналов со спутника	Имеется
Загрузка пользовательских сценариев генерации сигналов со спутника	Имеется
Определение чувствительности приемника в режимах	Имеется
холодного старта	Имеется
слежения	Имеется
при наличии интерференции или действия помеховых сигналов	Имеется
Измерение *BER*-ошибок	Имеется
Определение *TTFF* в режимах холодного, теплого и горячего запуска	Имеется
Определение ошибки местоположения (широта, долгота, высота)	Имеется
абсолютной	Имеется
относительной (в режимах холодного, теплого и горячего запуска)	Имеется
абсолютной, относительной, при движении *GPS*-приемника	Имеется
Формирователь траекторий движения приемника с контролем скорости, ускорения и установкой начальных координат	Имеется
Формирование траекторий движения на основе скриптов	Имеется

Окончание таблицы 15

Моделирование интерференции и ввод помеховых сигналов	имеется (при использовании двух генераторов в системе)
Моделирование реальных условий распространения сигналов в тропосфере/ионосфере	Имеется
Измерение параметров приемника в режиме *multipath* (переотражений сигнала со спутника от земных объектов)	в процессе внедрения (при использовании двух генераторов в системе)
Возможность работы в *Real-Time*	появится в следующей версии ПО

Описание возможностей системы имитации и воспроизведения *GPS*-сигналов позволяет сделать вывод о её важной роли в тестировании и анализе *GPS*-приемников. Благодаря данной системе можно проводить тестирование различных стандартов спутниковой навигации и формировать разнообразные навигационные сигналы с заданными параметрами. Это обеспечивает возможность более глубокого и надежного тестирования приемников, а также повышает эффективность процесса разработки и контроля качества *GPS*-устройств.

3.3 Программа-сервер Системы оперативного мониторинга

Серверное программное обеспечение для обмена и хранения данных комплекса (далее – Программа) входит в состав системы оперативного контроля позаказного производства радиоэлектронной аппаратуры (далее – Система), код представлен в Приложении А. Программа предназначена для управления процессом сбора и анализа производственных данных в процессе позаказного производства радиоэлектронной аппаратуры. Данная программа для ЭВМ обеспечивает: регистрацию сотрудников и распределение рабочих мест в соответствии с заданными технологическими картами, визуализацию производственной цепочки, регистрацию деталей, сборочных единиц и изделий

в целом, сбор, хранение и отображение производственных данных, обмен данными с программой-клиентом Системы.

Функциональные возможности Программы доступны пользователю посредством графического интерфейса пользователя *SWS*-Управление. В соответствии с назначением и функциональными возможностями, Программа имеет главное интерфейсное информационное окно (рисунок 29).

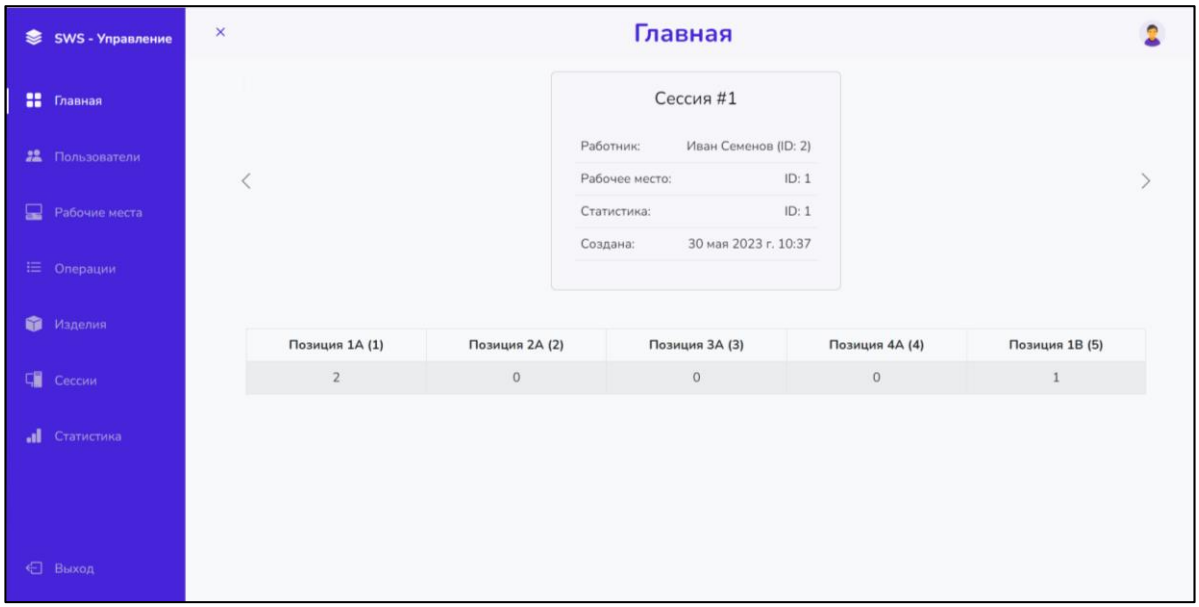


Рисунок 29 – Главное интерфейсное информационное окно

На главном интерфейсном информационном окне программы выводится следующая информация:

– меню вспомогательных окон;

– активные сессии: данные работника, соответствующее ему рабочее место, *ID* статистических данных по данной сессии, дата и время создания сессии;

– количество изделий на физических позициях (рабочих местах) в цехе.

При запуске Программы осуществляется подключение к базе данных с последующим сбором, сохранением в таблицы базы данных и анализом данных с управляющих рабочих станций Системы.

Минимальный состав используемых технических (аппаратных) средств серверной рабочей станции Системы:

– процессор не хуже *Intel Core* 2 (или эквивалент), 64-разряда;
– количество вычислительных ядер процессора, не менее: 2;
– частота процессора базовая, не менее: 2400 МГц;
– количество разъемов *Ethernet*, не менее: 1×1000 Мбит/с;
– количество интерфейсов *USB*, не менее: 5;
– объем оперативной памяти, не менее: 8 Гб;
– объем свободного дискового пространства, не менее: 1 Тб.

На северной рабочей станции Системы устанавливаются следующие системные программные средства:

– операционная система *Microsoft Windows* версии, не ниже 7/8/10 (64 разряда);
– система управления базами данных *PostgreSQL* 10;
– графический клиент *pgAdmin* 3 для управления базами данных в *PostgreSQL;*

Перед запуском программы необходимо убедиться в надежности подключения и надлежащем функционировании сетевого коммутатора Системы, после чего следует проверить возможность подключения серверной рабочей станции автоматизированным рабочим станциям стендов Системы.

Загрузка и запуск клиентского программного обеспечения для обмена и хранения данных комплекса осуществляется двойным нажатием левой кнопкой мыши на исполняемый файл Программы *SWSControl.exe*.

После загрузки Программы открывается окно авторизации (рисунок 30).

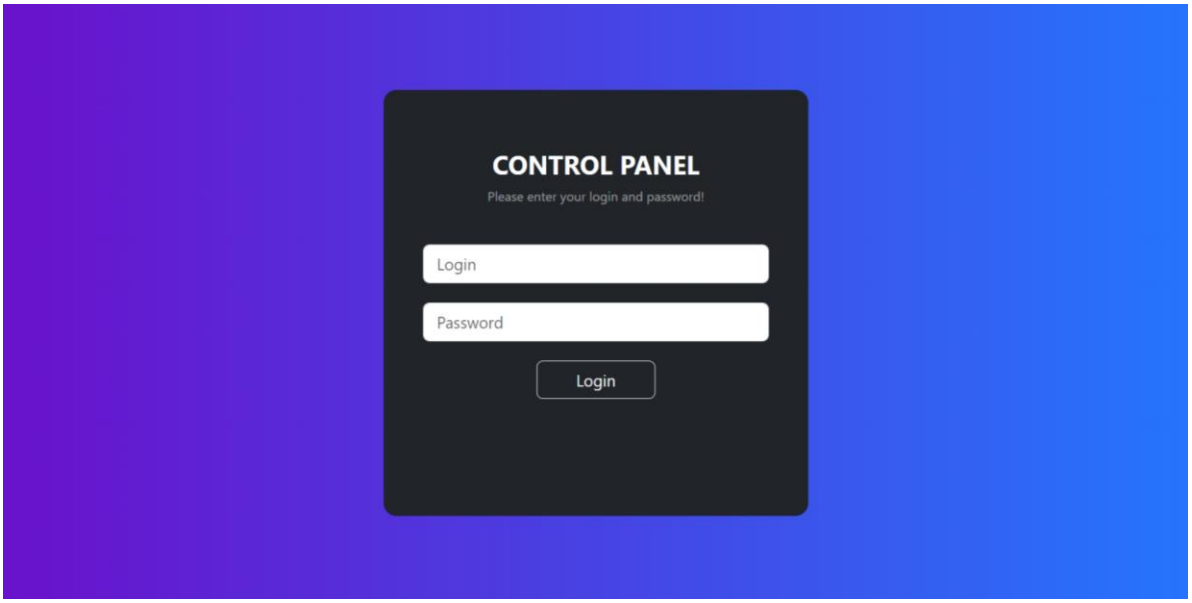


Рисунок 30 – Окно авторизации

Графический интерфейс Программы, приведенный на рисунке 31, обеспечивает отображение текущих параметров сессий. Для управления данными необходимо в меню выбрать необходимую вкладку из списка:

– пользователи;
– рабочие места;
– операции;
– изделия;
– сессии;
– статистика.

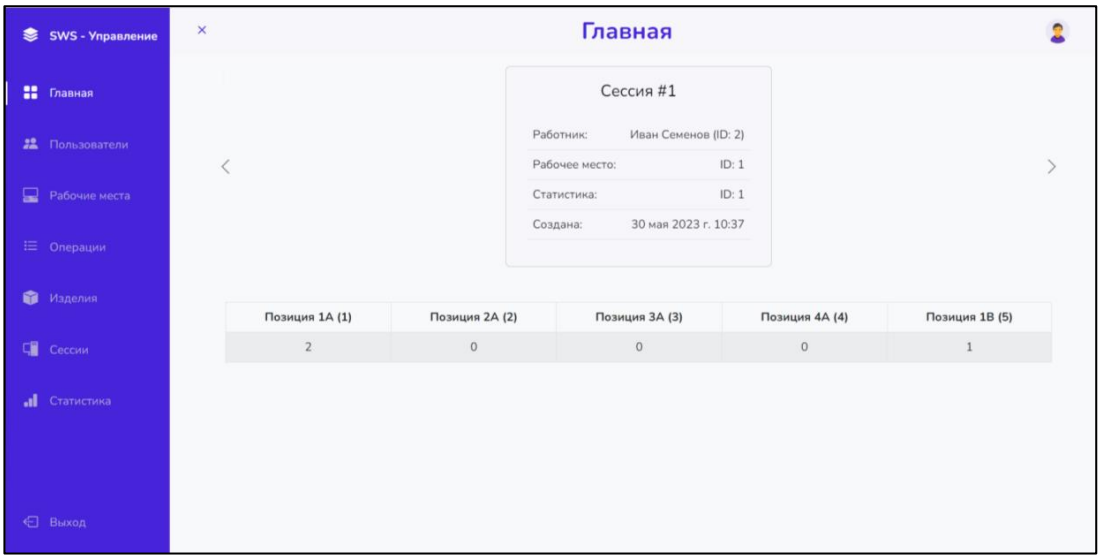


Рисунок 31 – Графический интерфейс главного окна

Модуль «Пользователи» предназначен для просмотра зарегистрированных пользователей (рисунок 32), регистрации пользователей (рисунок 33), управление информацией о пользователях (рисунок 34), назначения операций пользователям (рисунок 35).

На рисунке 32 представлен графический интерфейс вкладки «Пользователи», где можно наблюдать поисковое поле, перечень пользователей с расширенной информацией: *ID* пользователя, пользовательское имя, *email*, имя и фамилия, должность, создание пользователя, дата обновления данных пользователя.

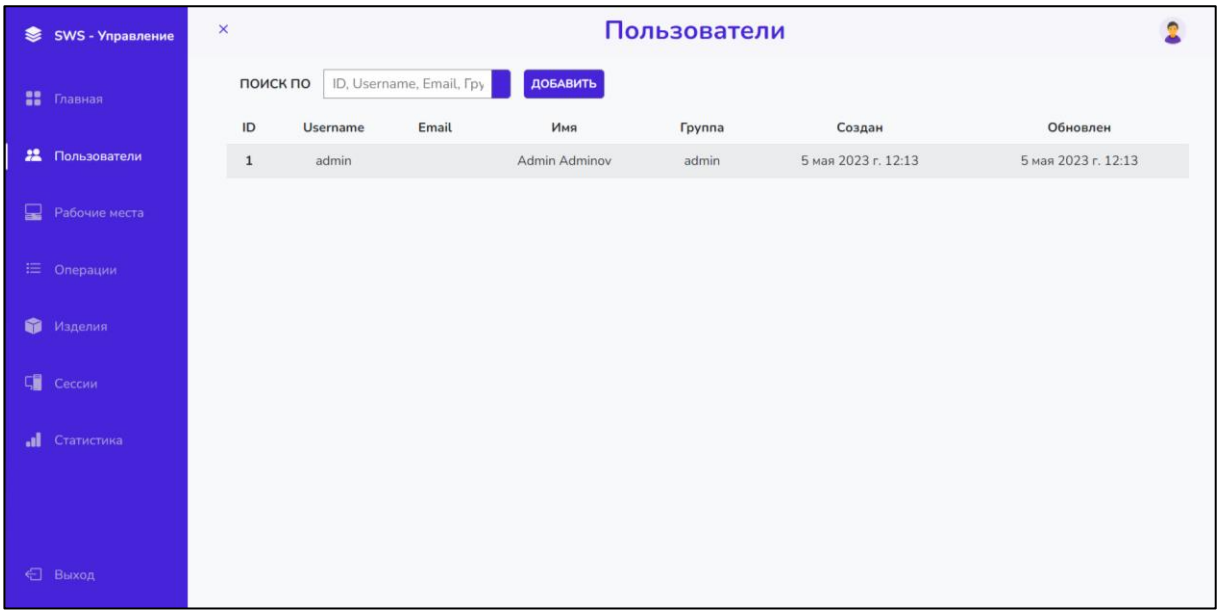


Рисунок 32 – Просмотр зарегистрированных пользователей

На рисунке 33 представлено окно «Управление пользователями», которое открывается при нажатии на кнопку «Добавить» во вкладке «Пользователи» графического интерфейса программы-сервер (рисунок 32). В окне выводится следующая информация: пользовательское имя, пароль, повтор пароля, *email*, имя и фамилия. При нажатии на кнопку «Сделать работником» пользователю назначается должность. При нажатии на кнопку «Закрыть» закрывается окно «Управление пользователями» без сохранения введенных данных. При нажатии на кнопку «Сохранить» вручную введенная информация сохраняется в базу данных и при дальнейшем открытии вкладки «Пользователи» сохраненные данные будут отображаться в графическом интерфейсе программы-сервер.

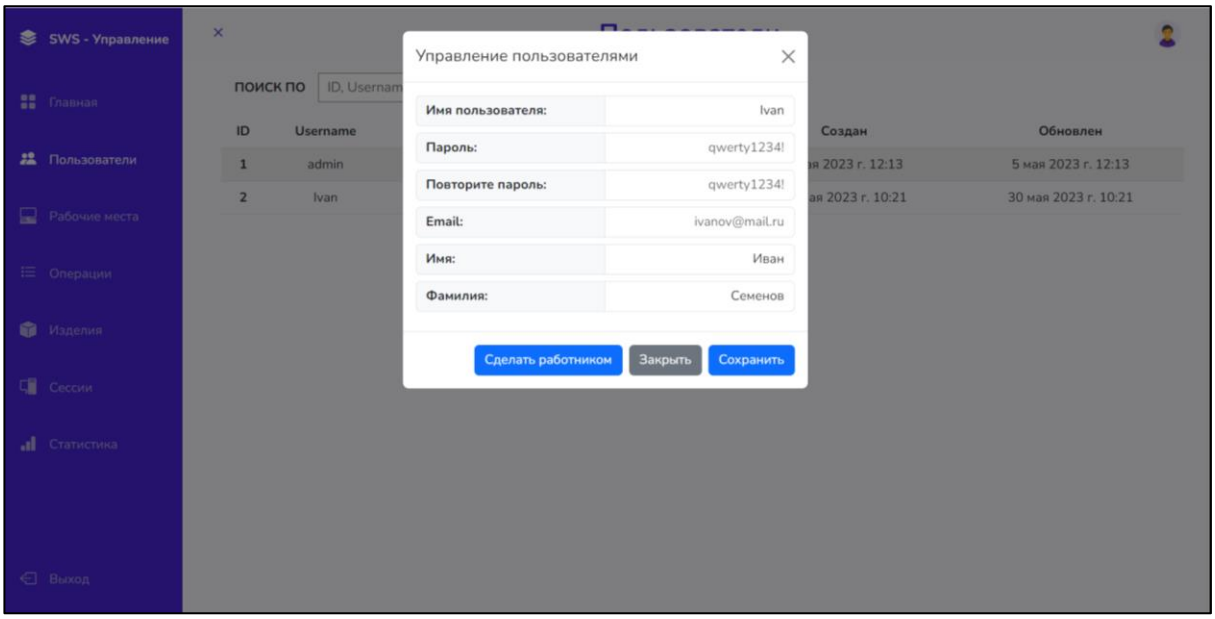


Рисунок 33 – Регистрация пользователей

На рисунке 34 представлено окно «Управление пользователями» при нажатии кнопки «Сделать работником» (рисунок 33). При нажатии на кнопку «Удалить работника» все данные работника удаляются с базы данных и, соответственно, не отображаются в графическом интерфейсе программы-сервер. При нажатии на кнопку «Управление операциями» открывается соответствующее окно (рисунок 35). При нажатии на кнопку «Закрыть» закрывается окно «Управление пользователями» без сохранения введенных данных. При нажатии на кнопку «Сохранить» вручную введенная информация сохраняется в базу данных и при дальнейшем открытии вкладки «Пользователи» сохраненные данные будут отображаться в графическом интерфейсе программы-сервер.

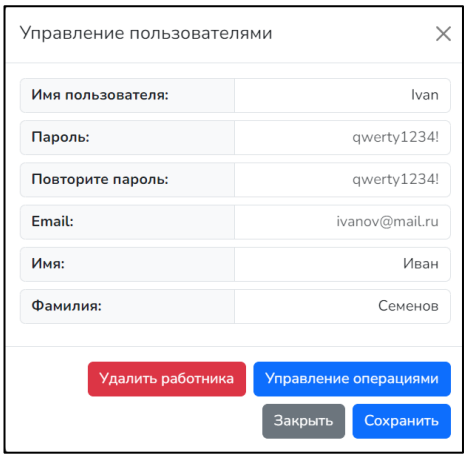


Рисунок 34 – Управление информацией о пользователях

На рисунке 35 представлено окно «Управление работником» при нажатии на кнопку «Управление операциями». В данном окне отображается список доступных и текущих операций пользователя с расширенной информацией: *ID* операции, название операции, описание операции. При нажатии на одну из доступных операций появляется информационное сообщение пользователю «Данные изменены!» и операция заносится в список текущих операций и удаляется из списка доступных.

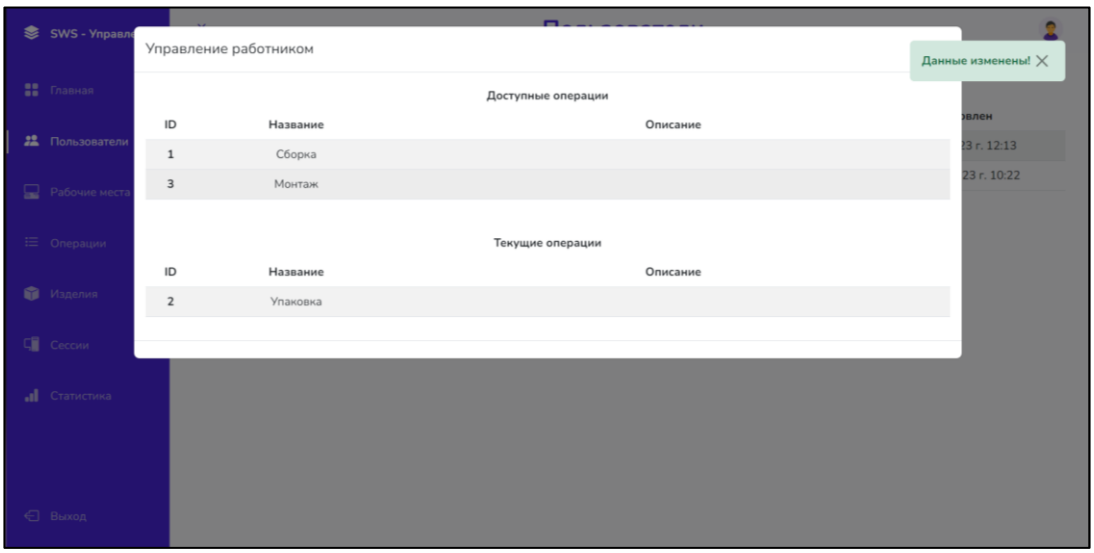


Рисунок 35 – Назначение операций пользователям

Модуль «Рабочие места» предназначен для просмотра рабочих мест и позиций (рисунок 36), управление позициями (рисунок 37) и рабочими местами (рисунок 38), поиска информации.

На рисунке 36 представлен графический интерфейс вкладки «Рабочие места и позиции», где можно наблюдать:

– отдельное поисковое поле для позиций;

– отельное поисковое поле для рабочих мест;

– перечень позиций с расширенной информацией: *ID* позиции, обозначение позиции, отметка начальной/конечной позиции, дата создания и обновления позиции;

– перечень рабочих мест с расширенной информацией: *ID* рабочего места, обозначение и *ID* позиции, отметка занятости рабочего места, дата создания и обновления рабочего места.

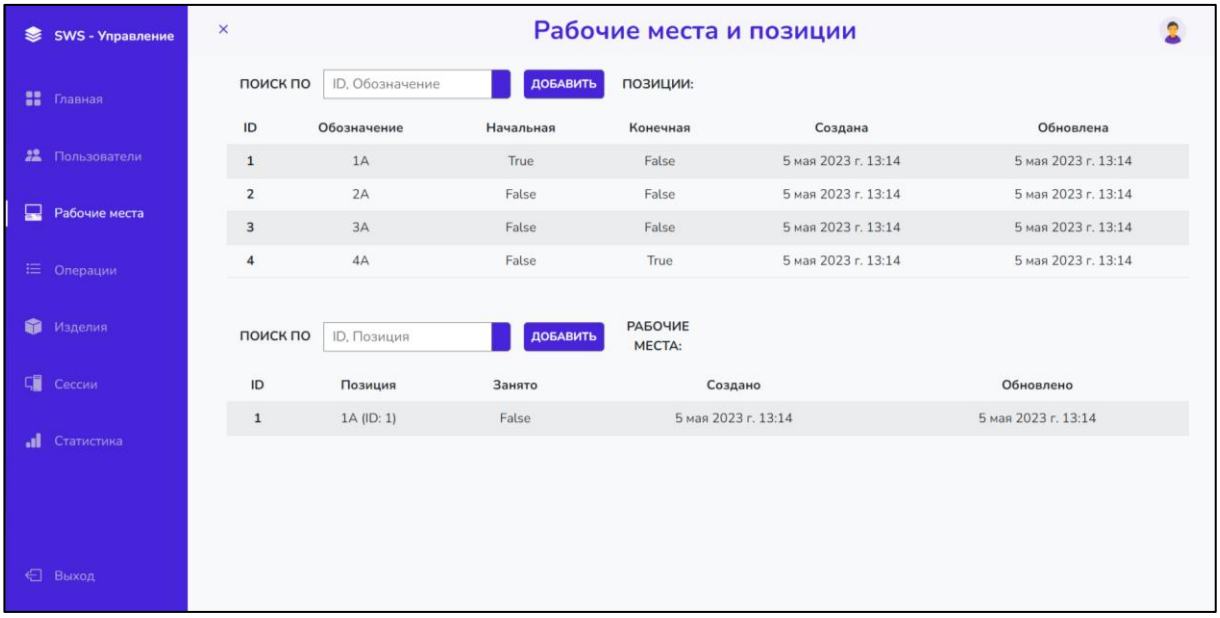


Рисунок 36 – Просмотр рабочих мест и позиций

На рисунке 37 представлено окно «Управление рабочими местами», которое открывается при нажатии на кнопку «Добавить» во вкладке «Рабочие места» графического интерфейса программы-сервер (рисунок 36). В окне представлена следующая информация: название позиции, отметка начальной позиции, отметка конечной позиции. При нажатии на кнопку «Закрыть» окно «Управление рабочими местами» закрывается без сохранения введенной информации. При нажатии на кнопку «Сохранить» вручную введенная информация сохраняется в базу данных и при дальнейшем открытии вкладки «Управление рабочими местами» сохраненные данные будут отображаться в графическом интерфейсе программы-сервер. При сохранении данных выводится советующее сообщение оператору «Позиция успешно создана!».

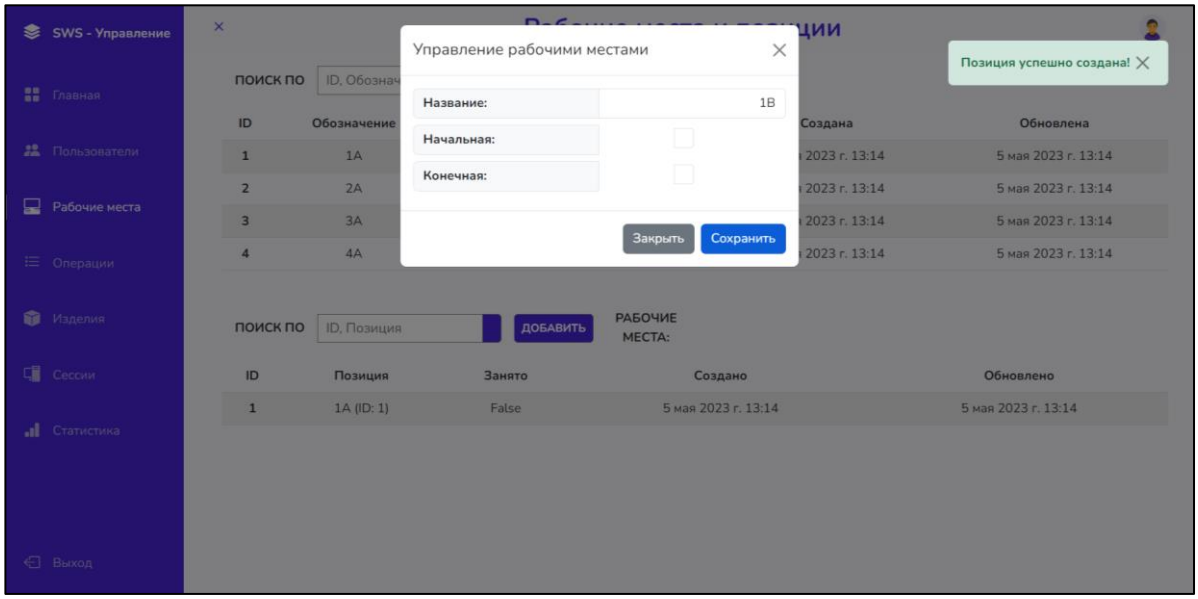


Рисунок 37 – Управление позициями

На рисунке 38 представлено окно «Управление рабочими местами», которое открывается при нажатии на кнопку «Добавить» во вкладке «Рабочие места» графического интерфейса программы-сервер (рисунок 36). В окне представлена следующая информация: обозначение позиции. При нажатии на кнопку «Закрыть» окно «Управление рабочими местами» закрывается без сохранения введенной информации. При нажатии на кнопку «Сохранить» вручную введенная информация сохраняется в базу данных и при дальнейшем открытии вкладки «Управление рабочими местами» сохраненные данные будут отображаться в графическом интерфейсе программы-сервер. При вводе несуществующего обозначения позиции оператору выводится сообщение об ошибке «Введите правильное число».

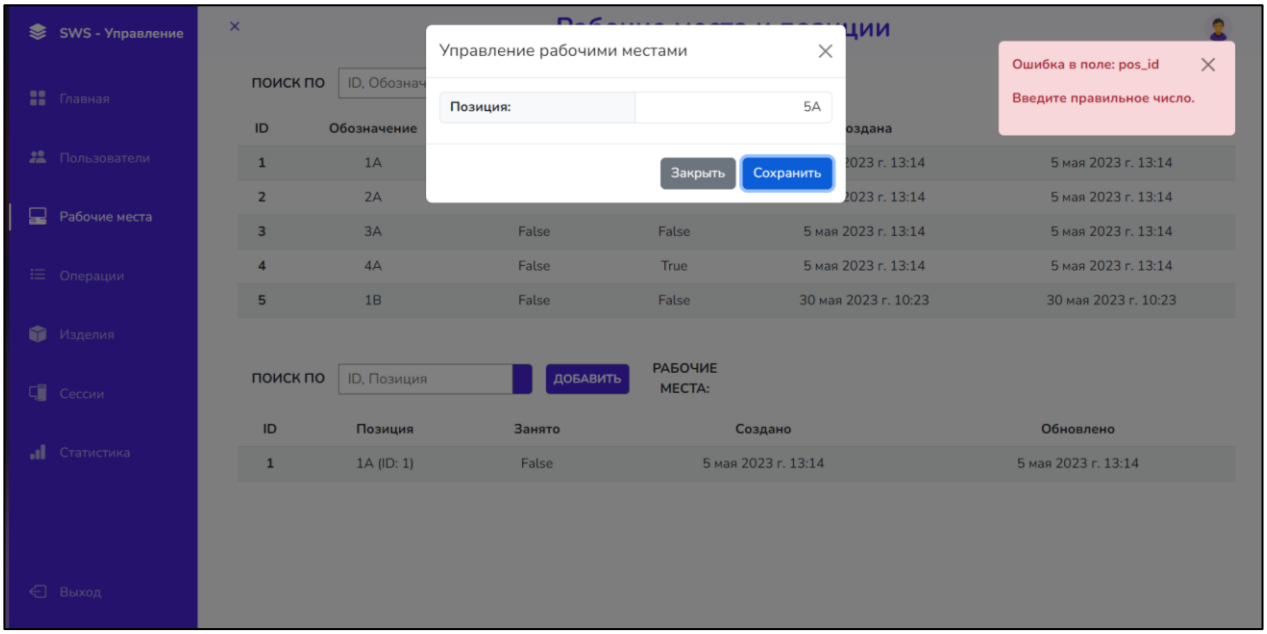


Рисунок 38 – Управление рабочими местами

Модуль «Операции» предназначен для просмотра зарегистрированных операций (рисунок 39), управление информацией об операциях (рисунок 40), поиска операций.

На рисунке 39 представлен графический интерфейс вкладки «Операции», где можно наблюдать поисковую строку, перечень операций с расширенной

информацией: *ID* операции, название операции, описание операции, дата создания и обновления операции. При нажатии на кнопку «Добавить» открывается соответствующее окно (рисунок 40).

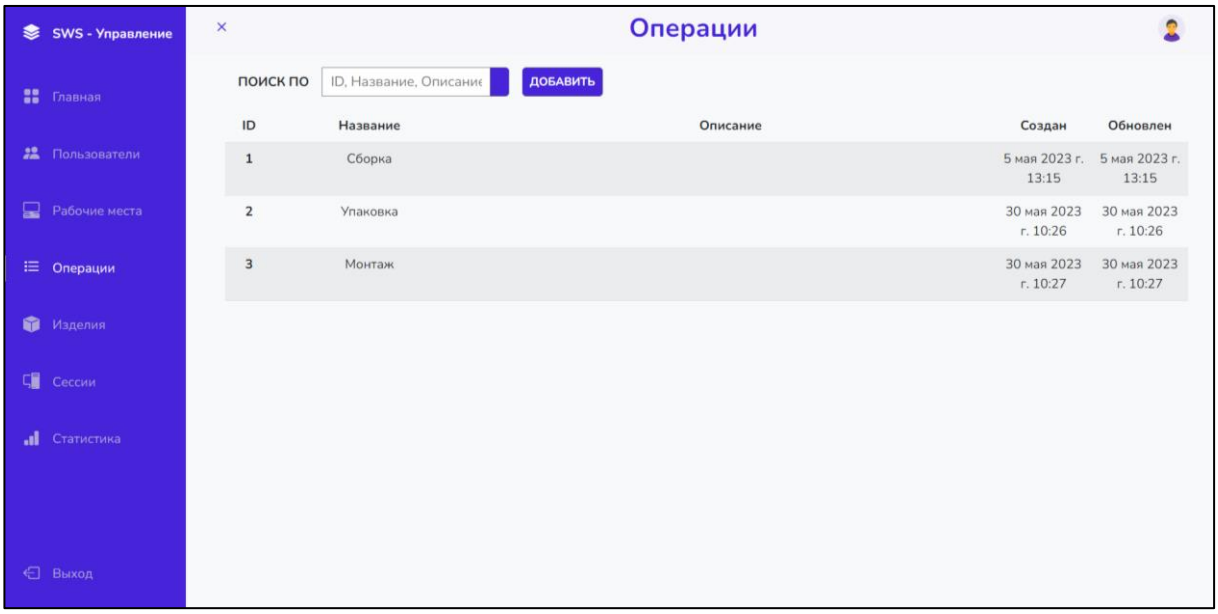


Рисунок 39 – Просмотр операций

На рисунке 40 представлено окно «Управление операциями», которое открывается при нажатии на кнопку «Добавить» во вкладке «Операции» графического интерфейса программы-сервер (рисунок 39). В окне представлена следующая информация: название операции, описание операции. При нажатии на кнопку «Закрыть» окно «Управление операциями» закрывается без сохранения введенной информации. При нажатии на кнопку «Сохранить» вручную введенная информация сохраняется в базу данных и при дальнейшем открытии вкладки «Управление операциями» сохраненные данные будут отображаться в графическом интерфейсе программы-сервер. При сохранении информации выводится соответствующее сообщение оператору «Запись успешно создана!».

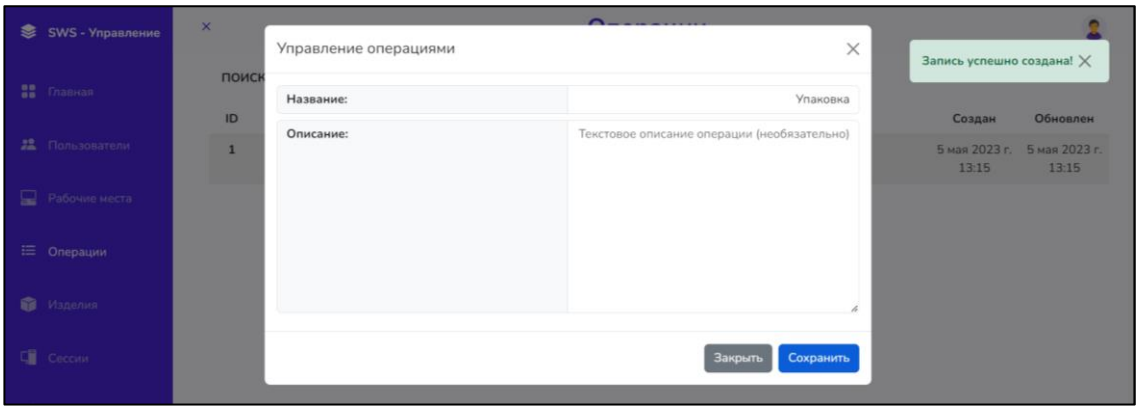


Рисунок 40 – Управление информацией об операциях

Модуль «Изделия» предназначен для просмотра зарегистрированных изделий (рисунок 41), регистрации изделий (рисунок 42), управление информацией об изделиях (рисунок 43), поиска операций.

На рисунке 41 представлен графический интерфейс вкладки «Изделия», где можно наблюдать поисковую строку, перечень изделий с расширенной информацией: *ID* изделия, обозначение изделия, серийный номер изделия, соответствующая позиция, описание изделия, дата начала и конца работы с изделием, дата создания и обновления изделия. При нажатии на кнопку «Добавить» открывается соответствующее окно (рисунок 42).

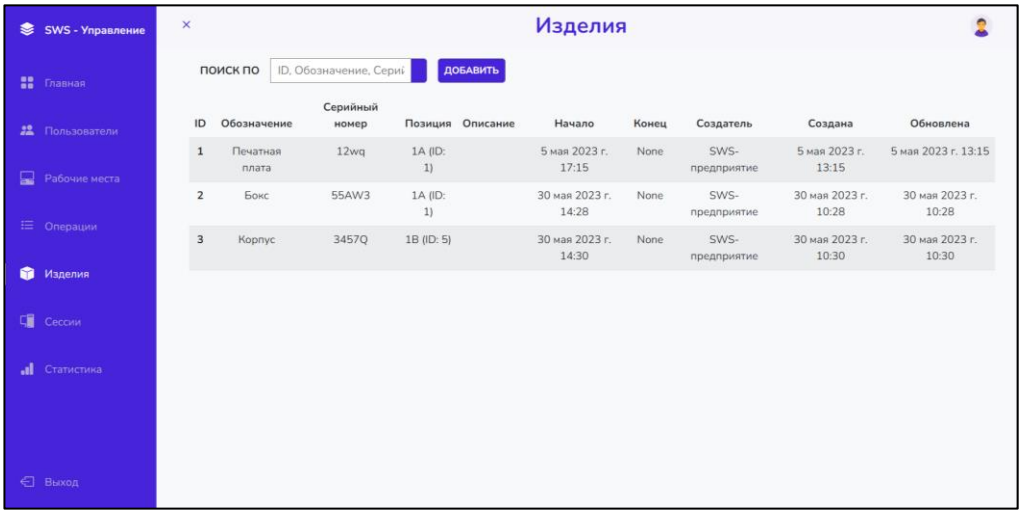


Рисунок 41 – Просмотр зарегистрированных изделий

На рисунке 42 представлено окно «Управление изделиями», которое открывается при нажатии на кнопку «Добавить» во вкладке «Изделия» графического интерфейса программы-сервер (рисунок 41). В окне представлена следующая информация: название изделия, серийный номер изделия, *ID* позиции, на которой изделие находится изначально, описание изделия. При нажатии на кнопку «Закрыть» окно «Управление изделиями» закрывается без сохранения введенной информации. При нажатии на кнопку «Сохранить» вручную введенная информация сохраняется в базу данных и при дальнейшем открытии вкладки «Управление изделиями» сохраненные данные будут отображаться в графическом интерфейсе программы-сервер. При сохранении информации выводится соответствующее сообщение оператору «Изделие успешно добавлено!».

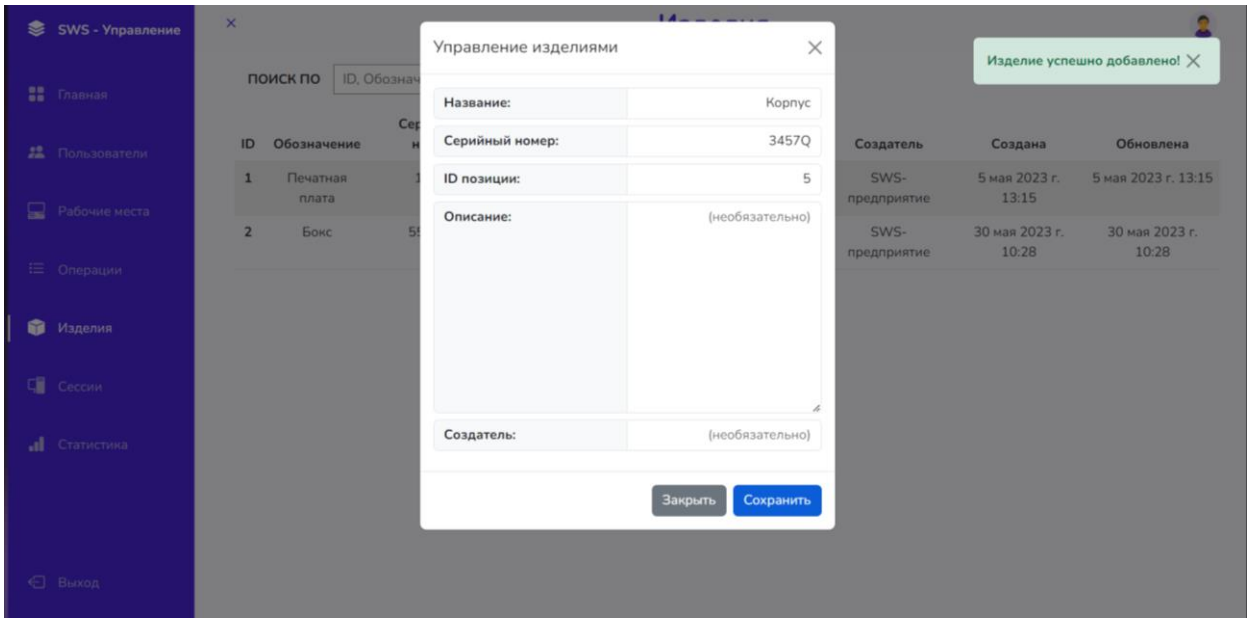


Рисунок 42 – Регистрация изделий

На рисунке 43 представлено окно «Управление изделиями» при нажатии на любое изделия из перечня, изображенном во вкладке «Изделия» (рисунок 41). При нажатии на кнопку «Удалить изделие» все данные изделия удаляются с базы данных и, соответственно, не отображаются в графическом интерфейсе программы-сервер. При нажатии на кнопку «Закрыть» закрывается окно «Управление пользователями» без сохранения введенных данных. При нажатии на кнопку «Сохранить» вручную введенная информация сохраняется в базу данных и при дальнейшем открытии вкладки «Изделие» сохраненные данные будут отображаться в графическом интерфейсе программы-сервер.

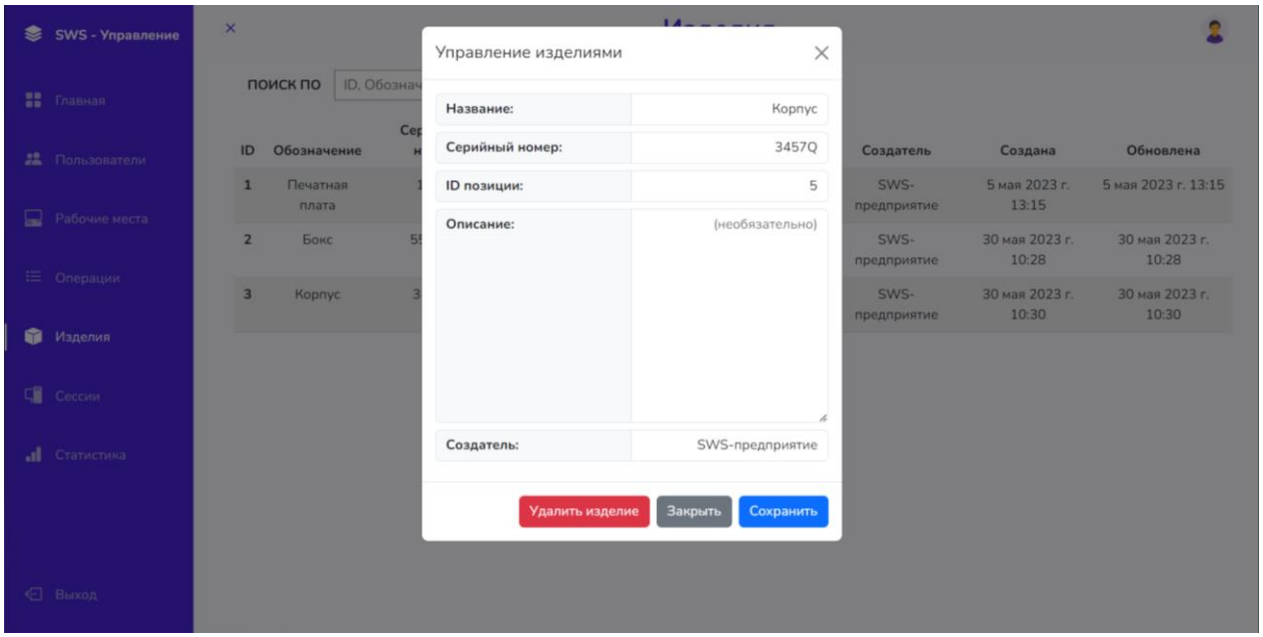


Рисунок 43 – Управление информацией об изделиях

Модуль «Сессии» предназначен для просмотра зарегистрированных сессий (рисунок 44), поиска сессий.

На рисунке 44 представлен графический интерфейс вкладки «Сессии», где можно наблюдать поисковую строку, перечень рабочих сессий с расширенной информацией: *ID* сессии, фамилия и имя работника, *ID* рабочего места, *ID* статистики, метка активности сессии, дата создания и обновления сессии.

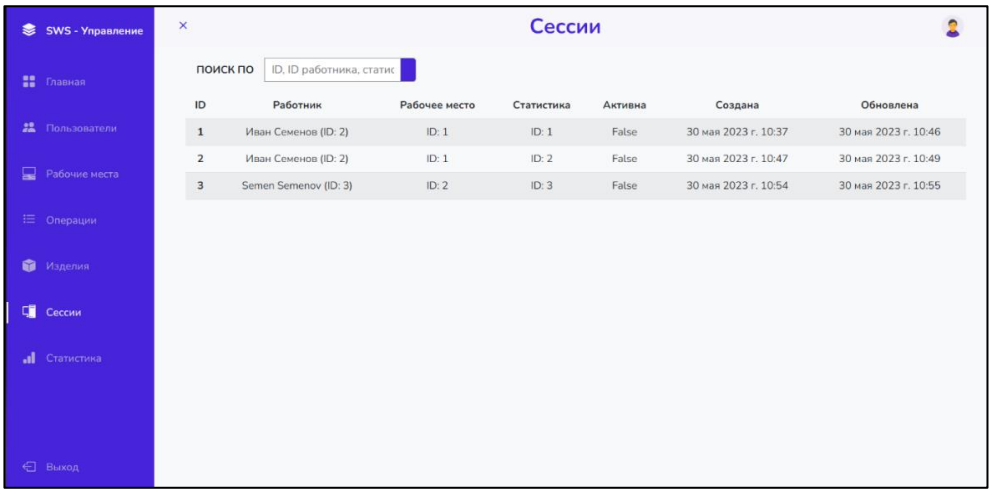


Рисунок 44 – Просмотр зарегистрированных сессий

Модуль «Статистика» предназначен для статистики по сессиям (рисунок 45), поиска статистических данных.

На рисунке 45 представлен графический интерфейс вкладки «Статистика», где можно наблюдать поисковую строку, перечень статистических данных по результатам работы операторов с расширенной информацией: *ID* статистики, количество отработанных изделий, *ID* рабочего места, *ID* статистики, среднее время работы оператора, общее время работы, метка прерывности сессии, дата создания и обновления сессии.

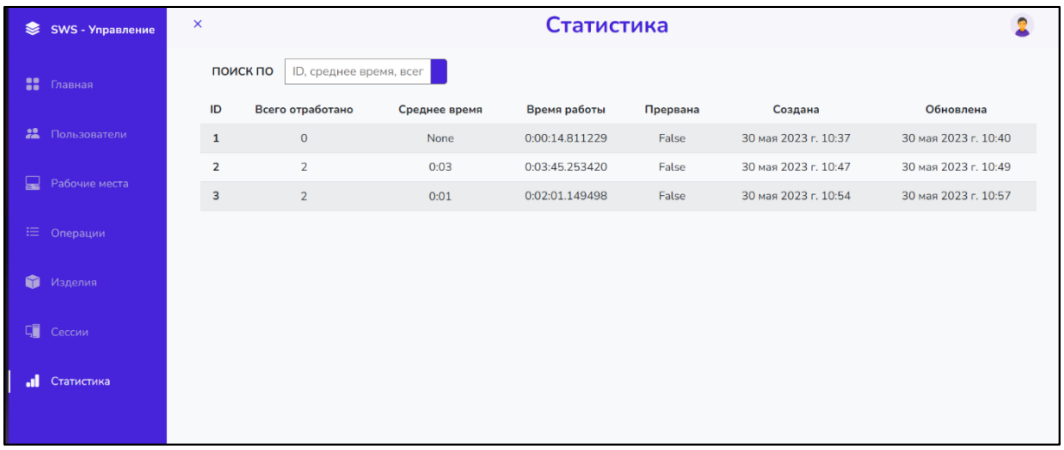


Рисунок 45 – Просмотр статистики по сессиям

Программа имеет графический интерфейс, и оператор самостоятельно выбирает в какой момент ему необходимо завершить работу с Программой. Для завершения работы с Программой следует активировать манипулятором «мышь» кнопку выхода из интерфейсного окна в нижнем левом углу «Выход».

Информационное сообщение о слишком коротком пароле, представленное на рисунке 46, возникает, если при регистрации пользователя ввести короткий пароль (до 8 символов).

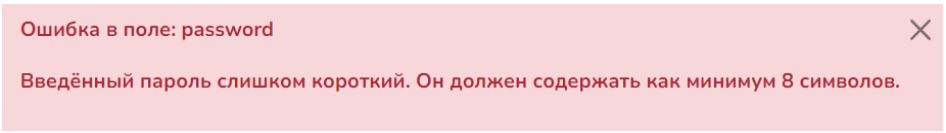


Рисунок 46 – Информационное сообщение о слишком коротком пароле

Пользователь может только закрыть данное информационное сообщение, что приведет к закрытию окна сообщения.

Информационное сообщение об успешном создании записи, представленное на рисунке 47, возникает, если зарегистрировать пользователя, рабочее место, позицию, операцию, изделие.

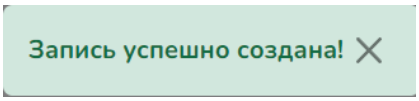


Рисунок 47 – Информационное сообщение об успешном создании записи

Пользователь может только закрыть данное информационное сообщение, что приведет к закрытию окна сообщения.

Информационное сообщение о назначении статуса «работник» пользователю, представленное на рисунке 48, возникает, если зарегистрированного пользователя назначить работником.

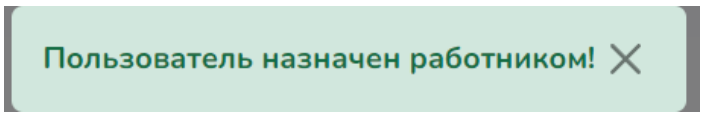


Рисунок 48 – Информационное сообщение о назначении статуса «работник» пользователю

Пользователь может только закрыть данное информационное сообщение, что приведет к закрытию окна сообщения.

Информационное сообщение об ошибке при неправильном вводе *ID*

Информационное сообщение об ошибке при неправильном вводе *ID*, представленное на рисунке 49, возникает, если неправильно внести *ID* позиции, *ID* рабочего места.

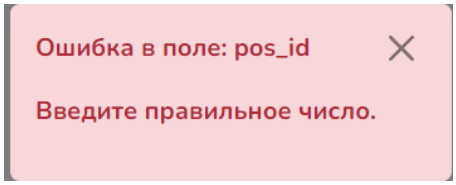


Рисунок 49 – Информационное сообщение об ошибке при неправильном вводе *ID*

Пользователь может только закрыть данное информационное сообщение, что приведет к закрытию окна сообщения.

Информационное сообщение об ошибке при неправильном вводе *ID*, представленное на рисунке 50, возникает при изменении данных пользователей, рабочих мест, позиций, операций, изделий.

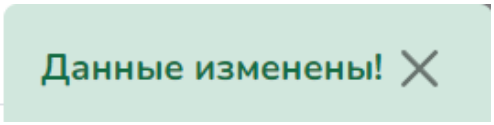


Рисунок 50 – Информационное сообщение об изменении данных

Пользователь может только закрыть данное информационное сообщение, что приведет к закрытию окна сообщения.

3.4 Программа-клиент Системы оперативного мониторинга

Клиентское программное обеспечение (далее – Программа) входит в состав автоматизированной системы оперативного контроля позаказного производства радиоэлектронной аппаратуры (далее – Система), код представлен в Приложении Б. Программа предназначена для оперативной регистрации объектов обработки, поступающих на отдельные рабочие места и фиксации технологических операций и процедур, осуществляемых на данных рабочих местах в процессе позаказного производства радиоэлектронной аппаратуры. Данная программа для ЭВМ обеспечивает: регистрацию объектов обработки по индивидуальным штрих-кодам, временную фиксацию проведения технологических операций и процедур, видеофиксацию технологических операций и процедур как непосредственно в рабочей зоне, так и в пределах зоны видимости монтажного микроскопа или иных оптических увеличительных приборов, обмен данными с программой-сервером системы оперативного контроля позаказного производства радиоэлектронной аппаратуры.

Функциональные возможности Программы доступны пользователю посредством графического интерфейса пользователя *SWS v*1.0. В соответствии с назначением и функциональными возможностями, Программа имеет основное интерфейсное окно (рисунок 51).

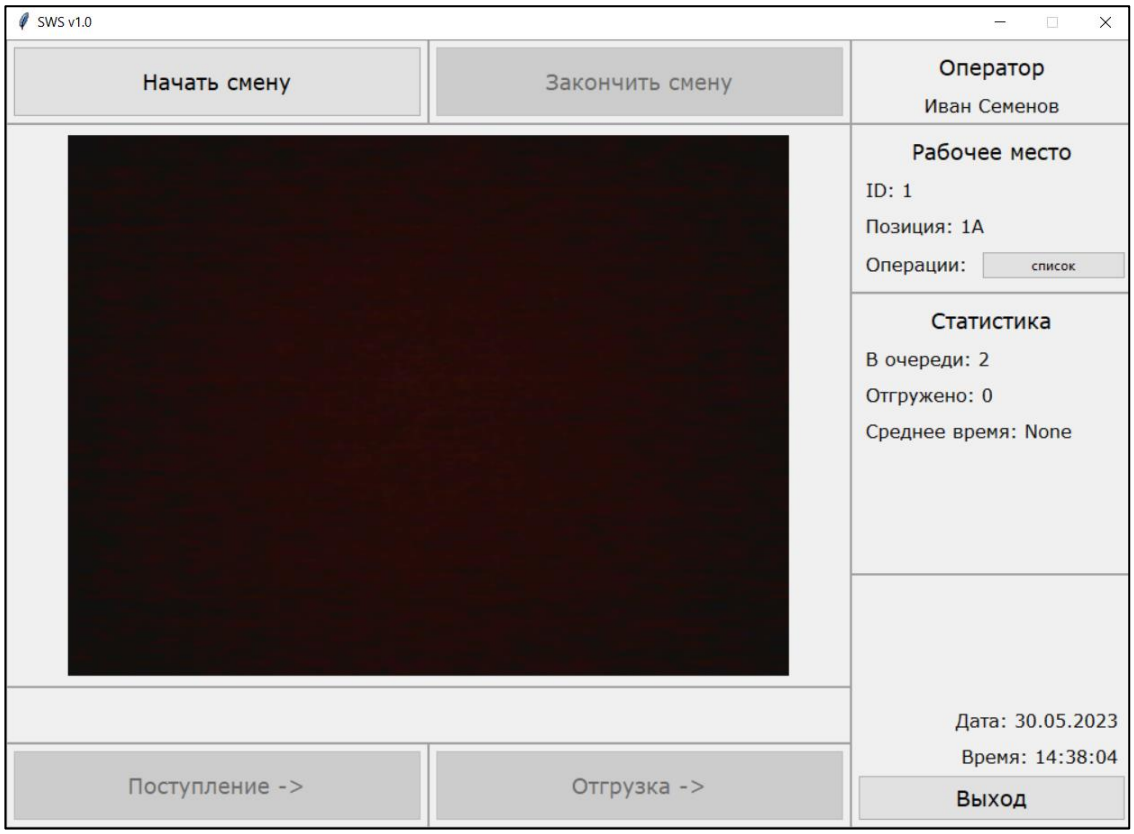


Рисунок 51 – Основное интерфейсное окно Программы

На основном интерфейсном окне программы выводится следующая информация:

– изображение с камеры;
– данные оператора (имя, фамилия);
– *ID* рабочего места;
– *ID* позиции;
– количество изделий в очереди;
– количество отгруженных изделий;
– среднее время работы оператора за текущую сессию;
– текущая дата;
– текущее время.

Общий алгоритм функционирования программы следующий.

При запуске Программы осуществляется подключение к базе данных с последующим сбором, сохранением в таблицы базы данных и анализом данных.

Изъят фрагмент текста

На северной рабочей станции Системы устанавливаются (разворачиваются) следующие системные программные средства:

– операционная система *Microsoft Windows* версии, не ниже 7/8/10 (64 разряда);

– система управления базами данных *PostgreSQL* 10;

– графический клиент *pgAdmin* 3 для управления базами данных в *PostgreSQL*;

Перед запуском программы необходимо убедиться в надежности подключения и надлежащем функционировании сетевого коммутатора Системы, после чего следует проверить возможность подключения клиентской рабочей станции автоматизированным рабочим станциям стендов Системы.

Загрузка и запуск клиентского программного обеспечения для обмена и хранения данных комплекса осуществляется двойным нажатием левой кнопкой мыши на исполняемый файл Программы *SWSv*1.0.*exe*.

После загрузки Программы открывается окно авторизации (рисунок 52).

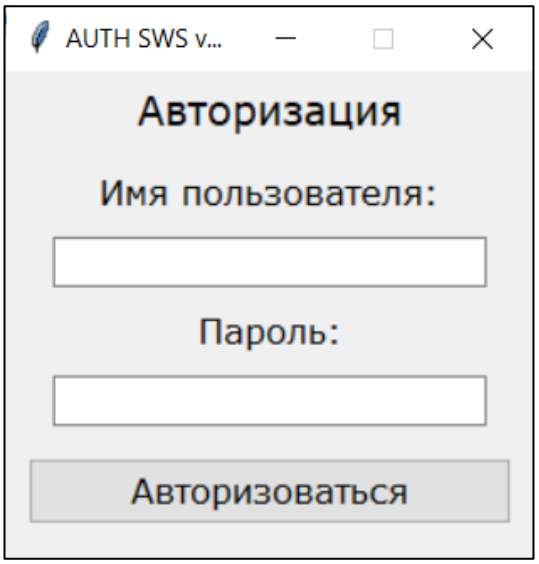


Рисунок 52 – Графический интерфейс Программы при авторизации

Графический интерфейс Программы, приведенный на рисунке 53, обеспечивает отображение текущих параметров сессии оператора.

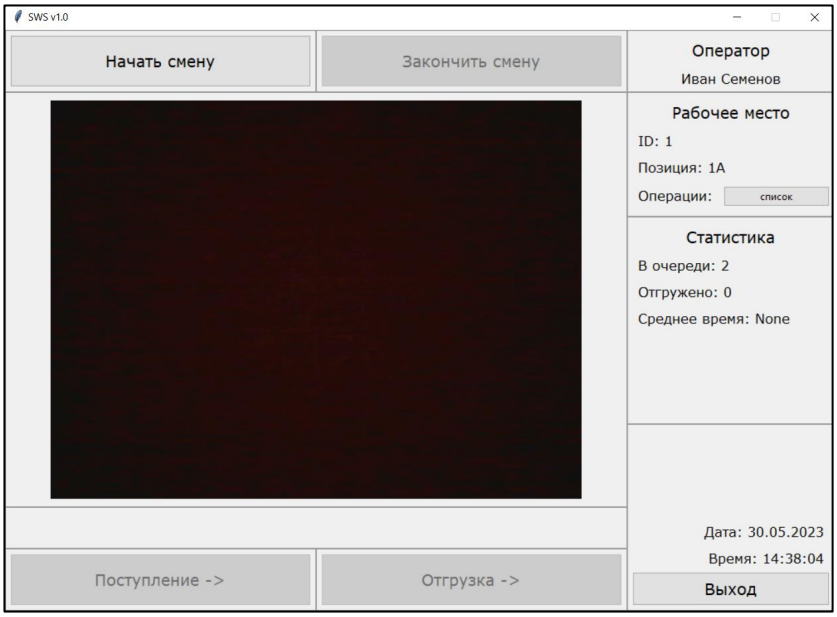


Рисунок 53 – Графический интерфейс пользователя Программы

На основном интерфейсном окне программы выводится следующая информация:

– изображение с камеры;

– данные оператора (имя, фамилия);
– *ID* рабочего места*;*
– *ID* позиции*;*
– количество изделий в очереди*;*
– количество отгруженных изделий*;*
– среднее время работы оператора за текущую сессию;
– текущая дата*;*
– текущее время.

С использованием графического интерфейса пользователь может отметить начало смены, отметить поступление изделия. После отметки поступления изделия нажатием кнопки «Поступление», на главном экране отображается время с момента поступления. Соответствующий скриншот представлен на рисунке 54.

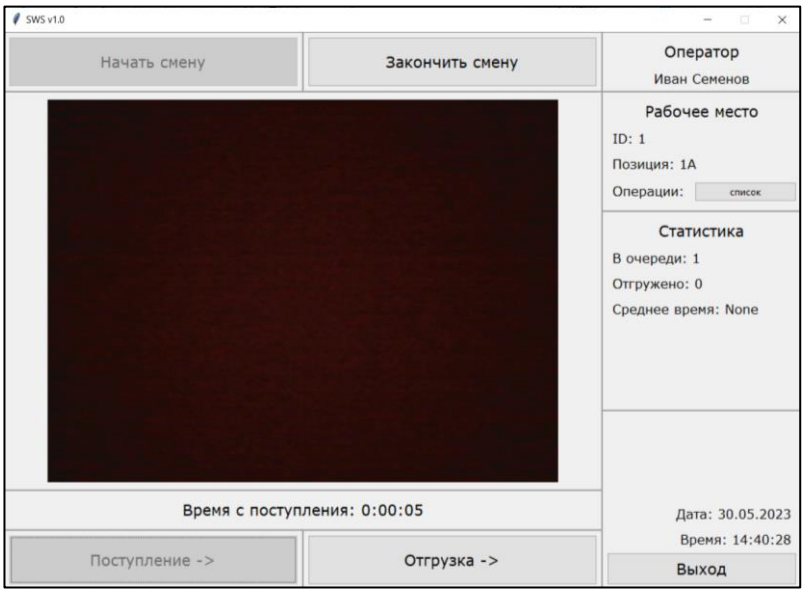


Рисунок 54 – Графический интерфейс пользователя Программы после отметки поступления изделия

Программный модуль обеспечивает фиксирование времени начала работы с изделием и выводит значения секундомера на графический интерфейс пользователя.

С помощью графического интерфейса Программы пользователь может выбрать операцию в списке доступных операций при нажатии на кнопку «список». Соответствующий скриншот представлен на рисунке 55.

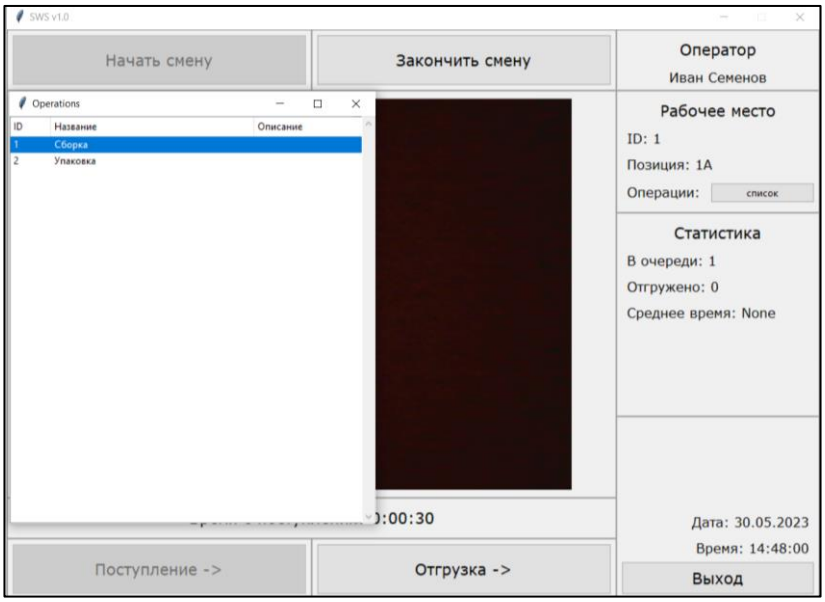


Рисунок 55 – Графический интерфейс пользователя Программы после нажатия на кнопку «список»

После проведенных операций с изделиями в графическом интерфейсе Программы изменяются статистические данные. Соответствующий скриншот представлен на рисунке 56.

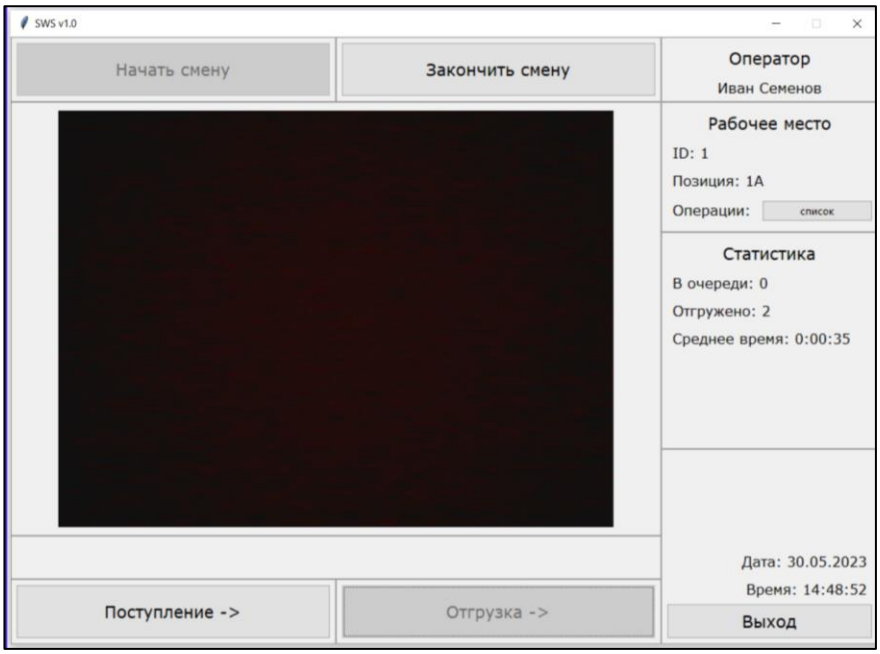


Рисунок 56 – Графический интерфейс пользователя Программы после проведенных операций с изделиями

Программа имеет графический интерфейс, и оператор самостоятельно выбирает в какой момент ему необходимо завершить работу с Программой. Для завершения работы с Программой следует активировать манипулятором «мышь» кнопку закрытия интерфейсного окна в нижнем верхнем углу окна «Выход».

Сообщение о подтверждении действия «Начать смену», представленное на рисунке 57 возникает, если нажать кнопку «Начать смену».

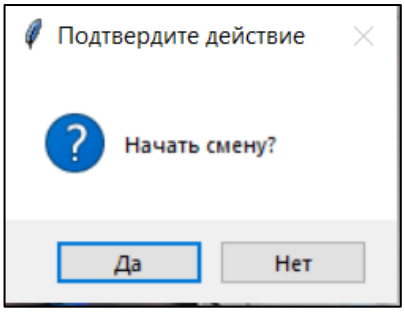


Рисунок 57 – Окно подтверждения действия «Начать смену»

В случае, если оператор нажмет кнопку «Да», начнется смена и зафиксируется время начала работы оператора. В ином случае, если пользователь нажмет «Нет», окно не закроется.

Сообщение об отсутствии изделий в очереди, представленное на рисунке 58, возникает в том случае, если оператор нажал кнопку «поступление», когда значение количества изделий в очереди равно нулю.

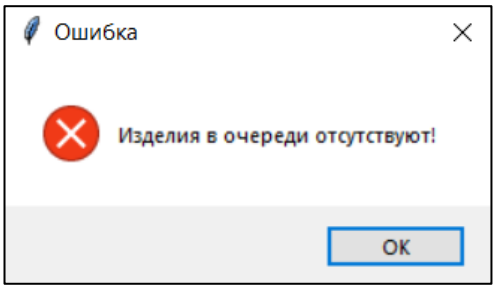


Рисунок 58 – Окно ошибки об отсутствии изделий в очереди

Пользователь может только закрыть данное информационное сообщение или нажать кнопку «ОК», что также приведет к закрытию окна сообщения.

Сообщение о подтверждении действия «Закончить смену», представленное на рисунке 59, возникает, если нажать кнопку «Закончить смену».

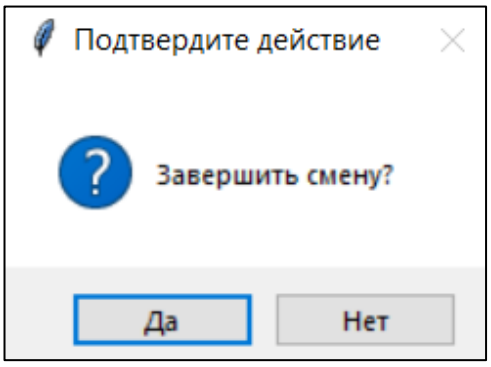


Рисунок 59 – Окно подтверждения действия «Закончить смену»

В случае, если оператор нажмет кнопку «Да», закончится смена и зафиксируется время конца работы оператора. В ином случае, если пользователь нажмет «Нет», смена не закончится и время не зафиксируется.

Информационное сообщение о конце смены, представленное на рисунке 60, возникает при завершении смены сразу же, после нажатия кнопки «да» в окне подтверждения действия «Закончить смену».

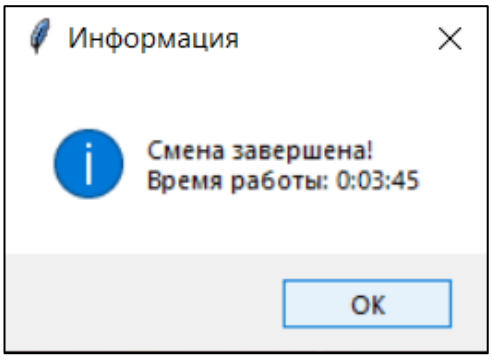


Рисунок 60 – Информационное сообщение о конце смены

Пользователь может только закрыть данное информационное сообщение или нажать кнопку «ОК», что также приведет к закрытию окна сообщения.

Сообщение о подтверждении действия «Выход», представленное на рисунке 61, возникает, если нажать кнопку «Выход».

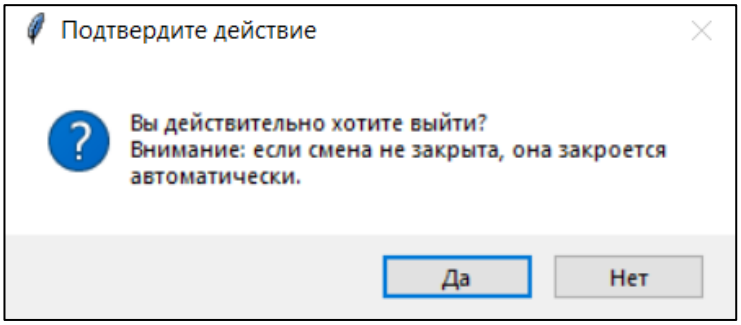


Рисунок 61 – Окно подтверждения действия «Выход»

В случае, если оператор нажмет кнопку «Да», смена закончится автоматически и зафиксируется время конца работы оператора. В ином случае, если пользователь нажмет «Нет», смена не закончится и выход из программы не выполнится.

Информационное сообщение слишком долгом бездействии оператора, представленное на рисунке 62, возникает в случае, когда оператор прекращает работу с изделием, не нажав кнопку «Отгрузка».

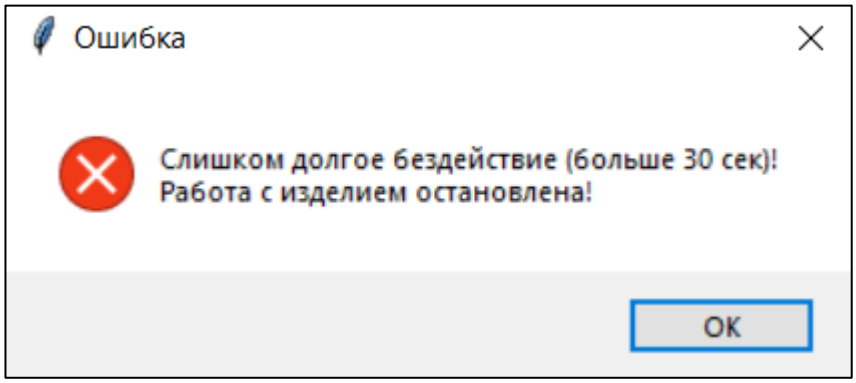


Рисунок 62 – Окно сообщения о долгом бездействии

При нажатии на кнопку «ОК» или закрытии данного окна, автоматически завершается смена и работа с изделием и на сервер приходит сообщение о прерывании сессии.

3.5 Выводы по главе 3

В данной главе была представлена реализация автоматизированной системы оперативного контроля позаказного производства радиоэлектронной аппаратуры. В ходе работы были рассмотрены основные аспекты системы, включая объект испытаний и возможную систему для внедрения.

Одним из ключевых моментов реализации было определение объекта испытаний, который включает в себя радиоэлектронную аппаратуру, проходящую производственный процесс. Были учтены особенности данного объекта, его технические требования и стандарты испытаний.

Важным шагом в реализации системы был выбор подходящей системы для внедрения. Учитывая специфику предметной области, были проанализированы различные варианты и выбрана наиболее подходящая система, удовлетворяющая требованиям контроля и управления производственными процессами.

В рамках работы были разработаны руководства пользователя для программы-клиент и программы-сервер автоматизированной системы оперативного контроля показазного производства радиоэлектронной аппаратуры, которые содержат подробную информацию об их функциональности, возможностях и способах использования.

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной работы была разработана и представлена автоматизированная система оперативного контроля позаказного производства радиоэлектронной аппаратуры.

В результате работы была достигнута поставленная цель – обеспечение позаказного производства РЭА посредством оперативного контроля отдельных рабочих мест и производственной цепочки в целом, которая обеспечивается созданием системы оперативного контроля позаказного производства радиоэлектронной аппаратуры.

Основными результатами работы являются:

– разработана архитектура автоматизированной системы оперативного контроля позаказного производства радиоэлектронной аппаратуры;

– разработано клиент-серверное программное обеспечение системы оперативного контроля позаказного производства радиоэлектронной аппаратуры;

– рассмотрена реализация автоматизированного рабочего места проведения испытаний навигационной аппаратуры потребителя с использованием предлагаемой системы оперативного контроля.

Система обладает значительным потенциалом для оптимизации производственных процессов, сокращения времени на контроль и испытания продукции, а также повышения качества и надежности выпускаемых изделий.

Однако следует отметить, что разработанная система является предметом дальнейших исследований и развития. Рассматривается расширение ее функциональных возможностей, улучшение алгоритмов и методов контроля, а также более широкая интеграция с другими системами производства и управления.

Ее результаты и рекомендации могут быть использованы для улучшения производственных процессов в данной области и способствовать повышению конкурентоспособности предприятий, занимающихся производством радиоэлектронной аппаратуры.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Автоматизация технологического процесса // Энергостроймонтаж : официальный сайт. – 2022. – URL: https://esm-t.ru/services/avtomatizatsiya-tekhnologicheskogo-protsessa/ (дата обращения: 11.04.2023).
2. Жандармов, Г. А. Актуальность внедрения организационного проектирования в рамках строительного предприятия / Г. А. Жандармов // Молодой ученый. – 2021. – № 24 (366). – С. 235–238.
3. Лыско, Р. Умное рабочее место. Как внедрить цифровое производство на участках ручного труда? / Р. Лыско // Вектор высоких технологий – 2018. – № 4. – С. 46–49.
4. Сафронов, Н. А. Типы промышленного производства. / Н. А. Сафронов // Экономика предприятия. – 1999. – № 2. – С. 20.
5. Позаказный метод производства в 1С / Обслуживание и внедрение 1С: Предприятие : официальный сайт. – 2019. – URL: https://nn.koderline.ru/expert/instruktsii/article-pozakaznyy-metod-proizvodstva-v-1s/ (дата обращения 19.11.2023)
6. Глухов, А. В., Замятин, Д. Н. Производственная стратегия компании и ее реализация на основе принципов «lean production». Менеджмент в России и за рубежом – 2018. – № 17(4) – С. 53–62.
7. Дрозд, О. В. Проблематика информационного сопровождения «проектирования на заказ» сложных технических изделий / О. В. Дрозд // Промышленные АСУ и контроллеры – 2022. – № 4. – С. 8–9.
8. Методологические основы проектирования сложных наукоемких изделий и принципы построения интегрированной информационной среды на базе CALS-технологий / А. А. Вичугова, В. Н. Вичугов, Е.А. Дмитриева, Г.П. Цапко, С.Г. Цапко – Томск : Томский политехнический институт, 2013. – 180 с.
9. Как осуществляется монтаж и сборка радиоэлектронной аппаратуры / Треком : официальный сайт. – Москва, 2020. – URL:http://korpusa-trekom.ru/sborka-radioehlektronnoj-apparatury.html (дата обращения: 10.12.2022)

10. Запуск линии поверхностного монтажа SMD // ТОО «ЭлтексАлатау» : официальный сайт. – 2020. – URL: https://eltexalatau.kz/pressroom/news/elteks_alatau_zapustil_liniyu_poverkhnostnogo_montazha_SMD.php (дата обращения: 10.12.2022)

11. Автоматический поверхностный монтаж // ExLab : официальный сайт. – 2019. – URL: http://www.exlab.ru/montazh-pechatnykh-plat/avtomaticheskij-poverkhnostnyj-montazh (дата обращения: 10.12.2022)

12. Мытников, А. Н. Умное рабочее место / А. Н. Мытников, И. А. Сушков // NovaUm.Ru. – 2017. – № 10. – С. 46–48.

13. Ростех начал внедрять умные рабочие места на базе промышленного интернета вещей // Ростех : официальный сайт. – 2020. – URL: https://rostec.ru/news/rostekh-nachal-vnedryat-umnye-rabochie-mesta-na-baze-promyshlennogo-interneta-veshchey/ (дата обращения: 22.12.2022)

14. Мы обновили систему мониторинга промышленного производства «Диспетчер» // Группа компаний Цифра : официальный сайт. – 2021. – URL: https://intechnology.ru/about/novosti-produkta/my-obnovili-sistemu-monitoringa-promyshlennogo-proizvodstva-dispetcher/ (дата обращения: 22.12.2022)

15. Переносные RFID-считыватели и сканеры // Scanberry : официальный сайт. – 2019. – URL: https://krasnoyarsk.scanberry.ru/news/perenosnye-RFID-schityvateli-i-skanery/ (дата обращения: 31.01.2023)

16. RFID считыватель настольный USB UHF ридер RRU9816-USB/232 // RFID-Scan Технологии сканирования : официальный сайт. – 2021. – URL: https://RFID-scan.ru/catalog/RFID_schityvateli_metok/RFID_schityvateli_USB/18249/ (дата обращения: 31.01.2023)

17. Настольный RFID UHF ридер RRU1861DK3324/USB // PRO RFID RFID-оборудование : официальный сайт. – 2019. – URL: https://pro-RFID.ru/nastolnyy-RFID-uhf-rIDer-rru1861dk3324USB/ (дата обращения: 31.01.2023)

18. RFID считыватель настольный UHF со встроенной антенной CLOU CL7206A2// RFID-Scan Технологии сканирования : официальный сайт. – 2019. – URL: https://RFID-scan.ru/catalog/RFID_schityvateli_metok/nastolnye_RFID_schityvateli/18606/ (дата обращения: 31.01.2023)

19. Универсальный драйвер ЭРФИД // ЭРФИД : официальный сайт. – 2019. – URL: https://eRFID.ru/RFID_software/RFID_driver/ (дата обращения: 15.02.2023)

20. RFID запись // NFC эксперт : официальный сайт. – 2020. – URL: https://nfcexpert.ru/RFID-zapis (дата обращения: 15.02.2023)

21. Что такое матричные камеры // СервисКлимат : официальный сайт. – 2021. – URL: https://servis-climat.ru/chto-takoye-matrichnyye-kamery/ (дата обращения: 23.02.2023)

22. Камеры для компьютерного зрения: характеристики // vc.ru : официальный сайт. – 2023 – URL: https://vc.ru/ml/87418-kamery-dlya-kompyuternogo-zreniya-harakteristiki (дата обращения: 23.02.2023)

23. Что такое машинное зрение? // RB.RU : официальный сайт. – 2021. – URL: https://rb.ru/story/chto-takoe-mashinnoe-zrenie/ (дата обращения: 23.02.2023)

24. Программное обеспечение MVTec // CameraIQ : официальный сайт. – 2023 – URL: https://www.cameraiq.ru/news/baumer-new-software-for-efficient-camera-integration/ (дата обращения: 23.02.2023)

25. Камера для микроскопа // Microsystemy : официальный сайт. – 2021 – URL: https://www.microsystemy.ru/info/articles/kamera-dlya-mikroskopa/ (дата обращения: 06.03.2023)

26. NIS Elements // Люкон : официальный сайт. – 2022 – URL: https://lucon.pro/oi/sai/nis-elements (дата обращения: 06.03.2023)

27. Клиент – серверная архитектура// QA_Bible : официальный сайт. – 2018 – URL: https://vladislaveremeev.gitbook.io/qa_bible/seti-i-okolo-nikh/klient-servernaya-arkhitektura-client-server-architecture (дата обращения: 30.03.2023)

28. Операционная система Android: учебное пособие для вузов / М.А. Дмитриев, А.В. Зуйков, А.А. Кузин, П.Е. Минин. – Москва : НИЯУ МИФИ, 2012. – 64 с.

29. Лабораторный комплекс «Спутниковые и навигационные системы» // Технолюкс – Техническое описание – 2022 – URL: http://tehnolux.org/doc/PXI-GNSS.pdf (дата обращения 30.05.2023)

ПРИЛОЖЕНИЕ А

api.py

0001_initial.py

```
# Generated by Django 4.1.7 on 2023-04-13 14:16

from django.conf import settings
from django.db import migrations, models
import django.db.models.deletion


class Migration(migrations.Migration):

    initial = True

    dependencies = [
        migrations.swappable_dependency(settings.AUTH_USER_MODEL),
    ]

    operations = [
        migrations.CreateModel(
            name='Profile',
            fields=[
                ('id', models.BigAutoField(auto_created=True,
primary_key=True, serialize=False, verbose_name='ID')),
                ('is_active', models.BooleanField(default=True)),
                ('user',
models.OneToOneField(on_delete=django.db.models.deletion.CASCADE,
to=settings.AUTH_USER_MODEL)),
            ],
        ),
    ]
```

admin.py

```
from django.contrib import admin

# Register your models here.
```

apps.py

```
from django.apps import AppConfig


class AdminfrontConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'adminfront'
```

authentication.py

```
from django.contrib.auth.backends import ModelBackend
from api.models import Account


SESSION_KEY = "_auth_user_id"


class UsernameAuthBackend(ModelBackend):
```

```
    def authenticate(self, request, **kwargs):
        username = kwargs['username']
        password = kwargs['password']
        try:
            account = Account.objects.get(username=username)
            if account.check_password(password):
                return account
            return None
        except Account.DoesNotExist:
            return None

    def get_user(self, user_id):
        try:
            return Account.objects.get(pk=user_id)
        except Account.DoesNotExist:
            return None
```

forms.py

```
from django import forms
from api.models import Operation

from api.models import Account


class LoginForm(forms.Form):
    username_attrs = {"class": "form-control form-control-lg",
                      "placeholder": "Login"}
    password_attrs = {"class": "form-control form-control-lg",
                      "placeholder": "Password"}

    username =
forms.CharField(widget=forms.TextInput(attrs=username_attrs),
required=True, min_length=3, max_length=40)
    password =
forms.CharField(widget=forms.PasswordInput(attrs=password_attrs),
required=True, min_length=3, max_length=40)


class UserForm(forms.Form):
    BASE_CLASSES = 'modal-field text-end'
    username_attrs = {"class": f"form-control {BASE_CLASSES}",
                      "placeholder": "ivanov123"}
    password_attrs = {"class": f"form-control {BASE_CLASSES}",
                      "placeholder": "qwerty1234!"}
    confirm_password_attrs = {"class": f"form-control
{BASE_CLASSES}",
                              "placeholder": "qwerty1234!"}
    email_attrs = {"class": f"form-control {BASE_CLASSES}",
                   "placeholder": "ivanov@mail.ru"}
    first_name_attrs = {"class": f"form-control {BASE_CLASSES}",
                        "placeholder": "Иван"}
    last_name_attrs = {"class": f"form-control {BASE_CLASSES}",
                       "placeholder": "Иванов"}
    group_attrs = {"class": f"form-select {BASE_CLASSES}"}
```

```
    username =
forms.CharField(widget=forms.TextInput(attrs=username_attrs),
required=False, label='Имя пользователя')
    password =
forms.CharField(widget=forms.TextInput(attrs=password_attrs),
required=False, label='Пароль')
    confirm_password =
forms.CharField(widget=forms.TextInput(attrs=confirm_password_attr
s), required=False, label='Повторите пароль')
    email =
forms.EmailField(widget=forms.EmailInput(attrs=email_attrs),
required=False, label='Email')
    first_name =
forms.CharField(widget=forms.TextInput(attrs=first_name_attrs),
required=False, label='Имя')
    last_name =
forms.CharField(widget=forms.TextInput(attrs=last_name_attrs),
required=False, label='Фамилия')
    #group = forms.ChoiceField(choices=(('user', 'User'),
('worker', 'Worker'), ('admin', 'Admin')), label='Группа',
widget=forms.Select(attrs=group_attrs))


class OperationForm(forms.Form):
    BASE_CLASSES = 'modal-field text-end'
    title_attrs = {"class": f"form-control {BASE_CLASSES}",
                   "placeholder": "Упаковка"}
    description_attrs = {"class": f"form-control {BASE_CLASSES}",
                         "placeholder": "Текстовое описание
операции (необязательно)"}

    title =
forms.CharField(widget=forms.TextInput(attrs=title_attrs),
label='Название', max_length=200)
    description =
forms.CharField(widget=forms.Textarea(attrs=description_attrs),
label='Описание', required=False)


class GoodsForm(forms.Form):
    BASE_CLASSES = 'modal-field text-end'
    title_attrs = {"class": f"form-control {BASE_CLASSES}",
                   "placeholder": "Бокс 2x4"}
    serial_number_attrs = {"class": f"form-control
{BASE_CLASSES}",
                           "placeholder": "2аБ"}
    pos_id_attrs = {"class": f"form-control {BASE_CLASSES}",
                           "placeholder": "1"}
    description_attrs = {"class": f"form-control {BASE_CLASSES}",
                         "placeholder": "(необязательно)"}
    creator_attrs = {"class": f"form-control {BASE_CLASSES}",
                     "placeholder": "(необязательно)"}
```

```
    title =
forms.CharField(widget=forms.TextInput(attrs=title_attrs),
label='Название', max_length=200)
    serial_number =
forms.CharField(widget=forms.TextInput(attrs=serial_number_attrs),
label='Серийный номер', max_length=200)
    pos_id =
forms.IntegerField(widget=forms.TextInput(attrs=pos_id_attrs),
label='ID позиции', min_value=1, step_size=1)
    description =
forms.CharField(widget=forms.Textarea(attrs=description_attrs),
label='Описание', required=False)
    creator =
forms.CharField(widget=forms.TextInput(attrs=creator_attrs),
label='Создатель', max_length=200, required=False)
```

models.py

```
from django.db import models
from django.conf import settings


class Profile(models.Model):
    user = models.OneToOneField(settings.AUTH_USER_MODEL,
on_delete=models.CASCADE)
    is_active = models.BooleanField(default=True)

    def __str__(self):
        return 'Profile for user'.format(self.user.username)
```

tests.py

```
from django.test import TestCase

# Create your tests here.
```

urls.py

```
from django.urls import path
from . import views


urlpatterns = [
     path('login', views.user_login, name='sws-login'),
     path('logout', views.user_logout, name='sws-logout'),
     path('', views.main, name='sws-admin'),
     path('users', views.users, name='sws-users'),
     path('operations', views.operations, name='sws-operations'),
     path('goods', views.goods, name='sws-goods'),
     path('workspaces', views.workspaces, name='sws-workspaces'),
     path('sessions', views.sessions, name='sws-sessions'),
     path('statistic', views.statistic, name='sws-statistic'),
]
```

views.py

```
from django.db.models import Q
from django.contrib.auth.decorators import login_required
from django.core.serializers import serialize
from django.shortcuts import render
```

```
from django.http import JsonResponse, HttpResponseRedirect
from django.contrib.auth import login, logout
from django.contrib import messages

from .forms import LoginForm, UserForm, OperationForm, GoodsForm
from api.models import Account, Worker, Operation, Pos, Workspace,
Session, Statistic, Goods


def make_message_data(request):
    return {'message': [f'{i}', i.level_tag] for i in
messages.get_messages(request)}


def user_login(request):
    loginform = LoginForm()
    if request.method == 'POST':

        loginform = LoginForm(request.POST)
        if not loginform.is_valid():
            messages.error(request,
'{0}'.format(loginform.errors))
            return JsonResponse(make_message_data(request),
status=404)

        data = loginform.cleaned_data
        user = Account.objects.get(username=data['username'])

        if not user.group == 'admin' or not user.is_active:
            messages.error(request, 'You dont have permissions')
            return JsonResponse(make_message_data(request),
status=403)

        login(request, user)
        return JsonResponse({'url': '/'})

    return render(request, "login.html", {"form": loginform})


@login_required(login_url='/login', redirect_field_name='')
def main(request):
    active_sessions = Session.objects.filter(is_active=True)
    all_pos = Pos.objects.all()
    goods_count = []
    for pos in all_pos:
        count = pos.goods.all().count()
        goods_count.append(count)
    return render(request, "index.html", {'sessions':
active_sessions, 'poses': all_pos, 'goods': goods_count})


@login_required(login_url='/login', redirect_field_name='')
def users(request):
```

```
    user_form = UserForm()

    if request.method == 'GET' and request.GET.get('search',
None):
        search_param = request.GET.get('search')
        accounts =
Account.objects.filter(Q(group__icontains=search_param) |
Q(pk__icontains=search_param)
                                            |
Q(username__icontains=search_param) |
Q(email__icontains=search_param)
                                            |
Q(first_name__icontains=search_param) |
Q(last_name__icontains=search_param))
        return render(request, "users.html", {'accounts':
accounts, 'form': user_form})

    if request.method == 'GET' and request.GET.get('worker_id',
None):
        w_id = request.GET.get('worker_id', None)
        if w_id:
            worker = Worker.objects.filter(pk=w_id)
            all_operations = Operation.objects.all()
            worker_json = serialize('json', worker)
            all_operations_json = serialize('json',
all_operations)
            return JsonResponse({'worker': worker_json,
'all_operations': all_operations_json})

    accounts = Account.objects.all()
    return render(request, "users.html", {'accounts': accounts,
'form': user_form})


@login_required(login_url='/login', redirect_field_name='')
def workspaces(request):
    search_pos = request.GET.get('search_pos', None)
    search_workspace = request.GET.get('search_workspace', None)

    all_poses = Pos.objects.all()
    all_workspaces = Workspace.objects.all()

    if request.method == 'GET' and (search_pos or
search_workspace):
        if search_pos:
            pos = Pos.objects.filter(Q(pk__icontains=search_pos) |
Q(title__icontains=search_pos)
                                     |
Q(is_start__icontains=search_pos) |
Q(is_end__icontains=search_pos))
            return render(request, "workspaces.html", {'poses':
pos, 'workspaces': all_workspaces})
```

```
        if search_workspace:
            workspace =
Workspace.objects.filter(Q(pk__icontains=search_workspace) |
Q(pos__id__icontains=search_workspace)
                                                    |
Q(is_busy__icontains=search_workspace))
            return render(request, "workspaces.html", {'poses':
all_poses, 'workspaces': workspace})

    return render(request, "workspaces.html", {'poses': all_poses,
'workspaces': all_workspaces})


@login_required(login_url='/login', redirect_field_name='')
def operations(request):
    operation_form = OperationForm()

    if request.method == 'GET' and request.GET.get('search',
None):
        search_param = request.GET.get('search')
        operations_filter =
Operation.objects.filter(Q(pk__icontains=search_param) |
Q(title__icontains=search_param)
                                                      |
Q(description__icontains=search_param))
        return render(request, "operations.html", {'operations':
operations_filter, 'form': operation_form})

    operations_all = Operation.objects.all()
    return render(request, "operations.html", {'operations':
operations_all, 'form': operation_form})


@login_required(login_url='/login', redirect_field_name='')
def goods(request):
    goods_form = GoodsForm()

    if request.method == 'GET' and request.GET.get('search',
None):
        search_param = request.GET.get('search')
        goods_filter =
Goods.objects.filter(Q(pk__icontains=search_param) |
Q(title__icontains=search_param)
                                              |
Q(serial_number__icontains=search_param) |
Q(pos__id__icontains=search_param)
                                              |
Q(description__icontains=search_param) |
Q(creator__icontains=search_param))
        return render(request, "goods.html", {'goods':
goods_filter, 'form': goods_form})

    goods_all = Goods.objects.all()
```

```
    return render(request, "goods.html", {'goods': goods_all,
'form': goods_form})


@login_required(login_url='/login', redirect_field_name='')
def sessions(request):
    if request.method == 'GET' and request.GET.get('search',
None):
        search_param = request.GET.get('search')
        sessions_filter =
Session.objects.filter(Q(pk__icontains=search_param) |
Q(worker__id__icontains=search_param)
                                                    |
Q(workspace__id__icontains=search_param) |
Q(statistic__id__icontains=search_param)
                                                    |
Q(is_active__icontains=search_param))
        return render(request, "sessions.html", {'sessions':
sessions_filter})

    sessions_all = Session.objects.all()
    return render(request, "sessions.html", {'sessions':
sessions_all})


@login_required(login_url='/login', redirect_field_name='')
def statistic(request):
    if request.method == 'GET' and request.GET.get('search',
None):
        search_param = request.GET.get('search')
        statistics_filter =
Statistic.objects.filter(Q(pk__icontains=search_param) |
Q(total_unload__icontains=search_param)
                                                      |
Q(aver_time__icontains=search_param) |
Q(work_duration__icontains=search_param)
                                                      |
Q(pause_stop__icontains=search_param))
        return render(request, "statistic.html", {'statistics':
statistics_filter})

    statistics_all = Statistic.objects.all()
    return render(request, "statistic.html", {'statistics' :
statistics_all})


@login_required(login_url='/login', redirect_field_name='')
def user_logout(request):
    logout(request)
    return HttpResponseRedirect('/login')
admin.py
from django.contrib import admin
```

```
# Register your models here.
apps.py
from django.apps import AppConfig


class ApiConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'api'
models.py
from django.db import models
from django.contrib.auth.models import AbstractBaseUser,
BaseUserManager

GROUPS = [
    ('admin', 'admin_group'),
    ('user', 'user_group'),
    ('worker', 'worker_group'),
]


class AccountManager(BaseUserManager):
    def create_user(self, password, **kwargs):
        if not kwargs.get('username'):
            raise ValueError('Users must have a valid username')

        if not kwargs.get('first_name'):
            raise ValueError('Users must have a first name')

        if not kwargs.get('last_name'):
            raise ValueError('Users must have a last name')

        account = self.model(
            username=kwargs.get('username'),
            first_name=kwargs.get('first_name'),
            last_name=kwargs.get('last_name')
        )
        account.set_password(password)
        account.save()
        return account

    def create_superuser(self, password, **kwargs):
        account = self.create_user(password, **kwargs)
        account.group = 'admin'
        account.save()
        return account


class Account(AbstractBaseUser):
    username = models.CharField(max_length=40, unique=True)
    email = models.EmailField(blank=True)
    first_name = models.CharField(max_length=40)
    last_name = models.CharField(max_length=40)
```

```
    group = models.CharField(choices=GROUPS, default='user',
max_length=20)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)

    objects = AccountManager()

    USERNAME_FIELD = 'username'
    REQUIRED_FIELDS = ['first_name', 'last_name']

    class Meta:
        ordering = ['id']

    def __str__(self):
        return self.username


class Operation(models.Model):
    title = models.CharField(max_length=200, unique=True,
blank=False, null=False, default='Operation')
    description = models.TextField(default='')
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)

    class Meta:
        db_table = 'SWS_Operation'
        ordering = ['id']

    def __str__(self):
        return self.title


class Worker(models.Model):
    user = models.OneToOneField(Account,
on_delete=models.DO_NOTHING, primary_key=True)
    status = models.CharField(max_length=100, default='')
    operations = models.ManyToManyField(Operation)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)

    class Meta:
        db_table = 'SWS_Worker'
        ordering = ['user']


class Pos(models.Model):
    title = models.CharField(max_length=20, unique=True,
blank=True, null=True, default='Undef_pos')
    is_start = models.BooleanField(default=False)
    is_end = models.BooleanField(default=False)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
```

```
    class Meta:
        db_table = 'SWS_Pos'
        ordering = ['id']


class Goods(models.Model):
    title = models.CharField(max_length=200, unique=True,
blank=False, null=False)
    serial_number = models.CharField(max_length=20, unique=True,
blank=False, null=False)
    pos = models.ForeignKey(Pos, on_delete=models.DO_NOTHING,
null=True, related_name='goods')
    description = models.TextField(default='')
    start_time = models.DateTimeField(blank=True, null=True)
    end_time = models.DateTimeField(blank=True, null=True)
    creator = models.CharField(max_length=200, default='SWS-
предприятие')
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)

    class Meta:
        db_table = 'SWS_Goods'
        ordering = ['id']

    def __str__(self):
        return self.title


class Workspace(models.Model):
    operations = models.ManyToManyField(Operation)
    pos = models.ForeignKey(Pos, on_delete=models.DO_NOTHING,
null=True)
    is_busy = models.BooleanField(default=False)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)

    class Meta:
        db_table = 'SWS_Workspace'
        ordering = ['id']


class Statistic(models.Model):
    total_unload = models.PositiveSmallIntegerField(default=0)
    aver_time = models.TimeField(null=True)
    work_duration = models.DurationField(null=True)
    pause_stop = models.BooleanField(default=False)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)

    class Meta:
        db_table = 'SWS_Statistic'
        ordering = ['id']
```

```
class Session(models.Model):
    worker = models.ForeignKey(Worker,
on_delete=models.DO_NOTHING)
    workspace = models.ForeignKey(Workspace,
on_delete=models.DO_NOTHING)
    statistic = models.ForeignKey(Statistic,
on_delete=models.DO_NOTHING)
    is_active = models.BooleanField(default=True)
    has_photo = models.BooleanField(default=False)
    photo_path = models.FilePathField(path='', recursive=True,
blank=True)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)

    class Meta:
        db_table = 'SWS_Session'
        ordering = ['id']
permissions.py
from rest_framework.permissions import BasePermission


class IsAdminGroup(BasePermission):
    def has_permission(self, request, view):
        return request.user.group == 'admin'


class AccountPermission(BasePermission):
    def has_permission(self, request, view):
        if view.action == 'list':
            if request.user and request.user.is_authenticated:
                return request.user.group == 'admin'
            return False
        return True

    def has_object_permission(self, request, view, account):
        if request.user and request.user.is_authenticated:
            if account == request.user:
                return True
            return request.user.group == 'admin'
        return False


class WorkerPermission(BasePermission):
    def has_permission(self, request, view):
        if request.user and request.user.is_authenticated:
            if view.action in ['list', 'create', 'update',
'partial_update', 'destroy']:
                return request.user.group == 'admin'
            return True
        return False

    def has_object_permission(self, request, view, worker):
```

```
        if worker.user == request.user:
            return True
        return request.user.group == 'admin'


class WorkspacePermission(BasePermission):
    def has_permission(self, request, view):
        if request.user.group == 'worker' or request.user.group ==
'admin':
            if view.action in ['create', 'destroy']:
                return request.user.group == 'admin'
            return True
        return False


class StatisticPermission(BasePermission):
    def has_permission(self, request, view):
        if request.user.group == 'worker' or request.user.group ==
'admin':
            if view.action in ['list', 'destroy']:
                return request.user.group == 'admin'
            return True
        return False


class SessionPermission(BasePermission):
    def has_permission(self, request, view):
        if request.user.group == 'worker' or request.user.group ==
'admin':
            if view.action in ['list', 'destroy']:
                return request.user.group == 'admin'
            return True
        return False

    def has_object_permission(self, request, view, session):
        if session.worker.user == request.user:
            return True
        return request.user.group == 'admin'
serializers.py
from datetime import datetime as dt
from rest_framework import serializers
from rest_framework.validators import UniqueValidator
from rest_framework_simplejwt.serializers import
TokenObtainPairSerializer
from django.conf import settings
from django.contrib.auth import update_session_auth_hash
from django.contrib.auth.password_validation import
validate_password
from api.models import Account, Operation, Worker, Pos, Goods,
Workspace, Statistic, Session
from api.validators import AccountIdValidator,
OperationIdValidator, PosIdValidator, WorkspaceIdValidator
```

```
class TokenSerializer(TokenObtainPairSerializer):
    @classmethod
    def get_token(cls, user):
        token = super(TokenSerializer, cls).get_token(user)
        token['username'] = user.username
        return token


class AccountSerializer(serializers.ModelSerializer):
    email = serializers.EmailField(required=False,
validators=[UniqueValidator(queryset=Account.objects.all())])
    password = serializers.CharField(write_only=True,
required=False, validators=[validate_password])
    confirm_password = serializers.CharField(write_only=True,
required=False)

    class Meta:
        model = Account
        fields = ['id', 'username', 'email', 'first_name',
'last_name', 'created_at',
                  'updated_at', 'password', 'confirm_password']
        read_only_fields = ['created_at', 'updated_at']

    def validate(self, attrs):
        if attrs.get('password') != attrs.get('confirm_password'):
            raise serializers.ValidationError({"confirm_password":
"Password fields didn't match."})
        return attrs

    def update(self, instance, validated_data):
        instance.email = validated_data.get('email',
instance.email)
        instance.first_name = validated_data.get('first_name',
instance.first_name)
        instance.last_name = validated_data.get('last_name',
instance.last_name)
        instance.save()

        password = validated_data.get('password', None)
        confirm_password = validated_data.get('confirm_password',
None)
        if password and confirm_password and password ==
confirm_password:
            instance.set_password(password)
            instance.save()

        update_session_auth_hash(self.context.get('request'),
instance)

        return instance
```

```
class OperationSerializer(serializers.ModelSerializer):
    class Meta:
        model = Operation
        fields = ['id', 'title', 'description', 'created_at',
'updated_at']
        read_only_fields = ['id', 'created_at', 'updated_at']
        extra_kwargs = {'title': {'required': True}}


def operation_action(operation_id, operation_mode, instance):
    operation = Operation.objects.get(pk=operation_id)
    if operation_mode == 'add':
        instance.operations.add(operation)
    elif operation_mode == 'del':
        instance.operations.remove(operation)
    return instance


class WorkerSerializer(serializers.ModelSerializer):
    user = AccountSerializer(read_only=True, required=False)
    operations = OperationSerializer(read_only=True,
required=False, many=True)
    user_id = serializers.IntegerField(write_only=True,
required=False, validators=[AccountIdValidator()])
    operation_id = serializers.IntegerField(write_only=True,
required=False, validators=[OperationIdValidator()])
    operation_mode = serializers.CharField(write_only=True,
required=False, max_length=10)

    class Meta:
        model = Worker
        fields = ['user', 'status', 'operations', 'created_at',
'updated_at',
                  'user_id', 'operation_id', 'operation_mode']
        read_only_fields = ['created_at', 'updated_at']

    def create(self, validated_data):
        user_id = validated_data.get('user_id')
        if user_id:
            validated_data.pop('user_id')
            user = Account.objects.get(pk=user_id)
            if Worker.objects.filter(user=user).exists():
                raise serializers.ValidationError({"user_id":
"Duplicated user id"})
            if user.group == 'user':
                user.group = 'worker'
                user.save()
            return Worker.objects.create(user=user,
**validated_data)
        else:
            raise serializers.ValidationError({"user_id": "User ID
is require"})
```

```
    def update(self, instance, validated_data):
        instance.status = validated_data.get('status',
instance.status)
        instance.save()

        operation_id = validated_data.get('operation_id', None)
        operation_mode = validated_data.get('operation_mode',
None)
        if operation_id:
            if operation_mode in ['add', 'del']:
                instance = operation_action(operation_id,
operation_mode, instance)
            else:
                raise
serializers.ValidationError({"operation_mode": "Unknown operation
mod ('add'/'del')"})
            instance.save()

        return instance


class PosSerializer(serializers.ModelSerializer):
    class Meta:
        model = Pos
        fields = ['id', 'title', 'is_start', 'is_end',
'created_at', 'updated_at']
        read_only_fields = ['id', 'created_at', 'updated_at']


class GoodsSerializer(serializers.ModelSerializer):
    pos = PosSerializer(read_only=True, required=False)
    pos_id = serializers.IntegerField(write_only=True,
required=False, validators=[PosIdValidator()])

    class Meta:
        model = Goods
        fields = ['id', 'title', 'serial_number', 'pos',
'description', 'start_time',
                  'end_time', 'creator', 'created_at',
'updated_at', 'pos_id']
        read_only_fields = ['id', 'created_at', 'updated_at']
        extra_kwargs = {'title': {'required': True}}

    def create(self, validated_data):
        pos_id = validated_data.get('pos_id', None)
        if pos_id:
            pos = Pos.objects.get(pk=pos_id)
            if pos.is_start:
                return Goods.objects.create(start_time=dt.now(),
**validated_data)
        else:
            return Goods.objects.create(**validated_data)
```

```
    def update(self, instance, validated_data):
        instance.title = validated_data.get('title',
instance.title)
        instance.serial_number =
validated_data.get('serial_number', instance.serial_number)
        instance.description = validated_data.get('description',
instance.description)
        instance.creator = validated_data.get('creator',
instance.creator)
        instance.save()

        pos_id = validated_data.get('pos_id', None)
        if pos_id:
            pos = Pos.objects.get(pk=pos_id)
            if pos.is_start:
                instance.start_time = dt.now()
            if pos.is_end:
                instance.end_time = dt.now()
            instance.pos = pos
            instance.save()

        return instance


class WorkspaceSerializer(serializers.ModelSerializer):
    operations = OperationSerializer(read_only=True,
required=False, many=True)
    pos = PosSerializer(read_only=True, required=False)
    operation_id = serializers.IntegerField(write_only=True,
required=False, validators=[OperationIdValidator()])
    operation_mode = serializers.CharField(write_only=True,
required=False, max_length=10)
    pos_id = serializers.IntegerField(write_only=True,
required=False, validators=[PosIdValidator()])

    class Meta:
        model = Workspace
        fields = ['id', 'operations', 'pos', 'is_busy',
'created_at', 'updated_at',
                  'operation_id', 'operation_mode', 'pos_id']
        read_only_fields = ['id', 'created_at', 'updated_at']

    def update(self, instance, validated_data):
        instance.pos = validated_data.get('pos', instance.pos)
        instance.is_busy = validated_data.get('is_busy',
instance.is_busy)
        instance.save()

        operation_id = validated_data.get('operation_id', None)
        operation_mode = validated_data.get('operation_mode',
None)
        if operation_id:
            request = self.context['request']
```

```
            if request.user.group != 'admin':
                raise serializers.ValidationError("You don't have
permissions")
            if operation_mode in ['add', 'del']:
                instance = operation_action(operation_id,
operation_mode, instance)
            else:
                raise
serializers.ValidationError({"operation_mode": "Unknown operation
mod ('add'/'del')"})
            instance.save()

        pos_id = validated_data.get('pos_id', None)
        if pos_id:
            pos = Pos.objects.get(pk=pos_id)
            instance.pos = pos
            instance.save()

        return instance


class StatisticSerializer(serializers.ModelSerializer):
    class Meta:
        model = Statistic
        fields = ['id', 'total_unload', 'aver_time',
'work_duration', 'pause_stop',
                  'created_at', 'updated_at']
        read_only_fields = ['id', 'created_at', 'updated_at']

    def update(self, instance, validated_data):
        instance.total_unload = validated_data.get('total_unload',
instance.total_unload)
        instance.aver_time = validated_data.get('aver_time',
instance.aver_time)
        instance.work_duration =
validated_data.get('work_duration', instance.work_duration)
        instance.pause_stop = validated_data.get('pause_stop',
instance.pause_stop)
        instance.save()
        return instance


class SessionSerializer(serializers.ModelSerializer):
    worker = WorkerSerializer(read_only=True, required=False)
    workspace = WorkspaceSerializer(read_only=True,
required=False)
    statistic = StatisticSerializer(read_only=True,
required=False)
    workspace_id = serializers.IntegerField(write_only=True,
required=False, validators=[WorkspaceIdValidator()])

    class Meta:
        model = Session
```

```
        fields = ['id', 'worker', 'workspace', 'statistic',
'is_active', 'has_photo', 'photo_path',
                  'created_at', 'updated_at', 'workspace_id']
        read_only_fields = ['id', 'photo_path', 'created_at',
'updated_at']

    def update(self, instance, validated_data):
        instance.is_active = validated_data.get('is_active',
instance.is_active)
        instance.has_photo = validated_data.get('has_photo',
instance.has_photo)
        instance.save()

        if instance.has_photo or (validated_data.get('photo_path')
and instance.photo_path != ''):
            instance.photo_path =
f"{settings.IMG_FILE_PATH}/{instance.id}/{dt.now().strftime('%d-
%m-%Y-%H_%M_%S')}/"
            instance.save()
        return instance
tests.py
from django.test import TestCase
# Create your tests here.
urls.py
from rest_framework_simplejwt.views import TokenRefreshView,
TokenVerifyView
from rest_framework.routers import SimpleRouter
from django.urls import path, include
from api import views
router = SimpleRouter()
router.register(r'account', views.AccountView, basename='account')
router.register(r'operation', views.OperationView,
basename='operation')
router.register(r'worker', views.WorkerView, basename='worker')
router.register(r'pos', views.PosView, basename='pos')
router.register(r'goods', views.GoodsView, basename='goods')
router.register(r'workspace', views.WorkspaceView,
basename='workspace')
router.register(r'statistic', views.StatisticView,
basename='statistic')
router.register(r'session', views.SessionView, basename='session')
urlpatterns = [
    # AUTH URLS:
    path('auth/login/', views.TokenView.as_view(),
name='token_obtain_pair'),
    path('auth/verify/', TokenVerifyView.as_view(),
name='token_verify'),
    path('auth/login/refresh/', TokenRefreshView.as_view(),
name='token_refresh'),
    path('auth/logout/', views.LogoutView.as_view(),
name='auth_logout'),
    path('', include(router.urls)),
]
```

ПРИЛОЖЕНИЕ Б

app.py

```
import tkinter as tk
from tkinter import messagebox

from pages.login import LoginPage
from pages.index import MainPage


class App(tk.Tk):
    def __init__(self):
        tk.Tk.__init__(self)
        self._frame = None
        self.change_frame("login")

    def change_frame(self, frame_class):
        if frame_class == "login":
            new_frame = LoginPage(self)
        elif frame_class == "main":
            new_frame = MainPage(self)
        if self._frame is not None:
            self._frame.destroy()
        self._frame = new_frame
        self._frame.pack()

    def set_window_prop(self, title, width, height):
        self.wm_title(title)

        sw = self.winfo_screenwidth()
        sh = self.winfo_screenheight()

        x = (sw - width) / 2
        y = (sh - height) / 2

        self.wm_geometry('%dx%d+%d+%d' % (width, height, x, y))
        # self.geometry('%dx%d+%d+%d' % (w, h, x, y))
        self.resizable(False, False)


if __name__ == '__main__':
    app = App()
    app.mainloop()
```

index.py

```
import tkinter as tk
from PIL import Image, ImageTk
from tkinter import messagebox
from time import time
from datetime import datetime as dt
from numpy import mean
```

```
from modules.video import VideoCapture
from src.styles import style
from src.content import msg, text, btn
from src.models import user


class MainPage(tk.Frame):
    def __init__(self, master):
        self.master = master
        tk.Frame.__init__(self, self.master)
        self.master.set_window_prop("SWS v1.0", 1000, 700)

        self.op_name = "Иванов Иван Иванович"
        self.w_id = 1
        self.pos = 1
        self.in_queue = 100
        self.total_unload = 0
        self.aver_time = None

        self.main_frame = tk.Frame(master=self)
        self.camera_cvs = None
        self.cur_time_lbl = None

        self.start_bt = None
        self.stop_bt = None
        self.time_lbl = None
        self.upload_bt = None
        self.unload_bt = None
        self.in_queue_lbl = None
        self.total_unload_lbl = None
        self.aver_time_lbl = None

        self.make_ui()
        self.master.update()

        self.video_src = 0
        self.cur_frame = None
        self.vid = VideoCapture(self.video_src)

        self.work = False
        self.pause_start_time = None
        self.start_time = None
        self.stop_time = None
        self.goods_time = None
        self.goods_times = []

        self.after_id = None
        self.delay = 15
        self.update()

    def update(self):
        self.camera_update()
        self.lbl_update()
```

```
        self.after_id = self.master.after(self.delay, self.update)

    def camera_update(self):
        flag, img = self.vid.get_frame()
        if flag and self.camera_cvs is not None:
            self.cur_frame                                          =
ImageTk.PhotoImage(image=Image.fromarray(img))

self.camera_cvs.create_image(self.camera_cvs.winfo_width()   /   2,
self.camera_cvs.winfo_height() / 2,
                                          image=self.cur_frame)
            if self.work:
                cur_hash = self.vid.get_hash(img)
                if self.vid.compare_hash(cur_hash) <= 3:
                    self.tracker()
                else:
                    self.pause_start_time = None

    def tracker(self):
        if self.pause_start_time is None:
            self.pause_start_time = time()
        cur_time = time()
        if cur_time - self.pause_start_time > 30:
            messagebox.showerror(title=msg.error_title,
message=msg.error_tracker)
            self.pause_start_time = None
            self.work = False

    def lbl_update(self):
        if self.goods_time is not None:

self.time_lbl.config(text=text.goods_time.format(time=str(dt.now()
- self.goods_time).split('.')[0]))
        else:
            self.time_lbl.config(text="")

self.cur_time_lbl.config(text=text.cur_time.format(time=dt.now().s
trftime('%H:%M:%S')))

self.in_queue_lbl.config(text=text.in_queue.format(in_queue=self.i
n_queue))

self.total_unload_lbl.config(text=text.total_unload.format(total_u
nload=self.total_unload))

self.aver_time_lbl.config(text=text.aver_time.format(aver_time=str
(self.aver_time).split('.')[0]))

    def  bt_switcher(self,  turn=False,  start=False,  stop=False,
upload=False, unload=False):
        if turn:
            if start:
                self.start_bt['state'] = 'normal'
```

```
            if stop:
                self.stop_bt['state'] = 'normal'
            if upload:
                self.upload_bt['state'] = 'normal'
            if unload:
                self.unload_bt['state'] = 'normal'
        else:
            if start:
                self.start_bt['state'] = 'disabled'
            if stop:
                self.stop_bt['state'] = 'disabled'
            if upload:
                self.upload_bt['state'] = 'disabled'
            if unload:
                self.unload_bt['state'] = 'disabled'

    def start_work(self):
        result    =    messagebox.askyesno(title=msg.confirm_title,
message=msg.ask_start)
        if result:
            self.work = True
            self.start_time = dt.now()
            self.bt_switcher(start=True)
            self.bt_switcher(turn=True, stop=True, upload=True)

    def stop_work(self):
        result    =    messagebox.askyesno(title=msg.confirm_title,
message=msg.ask_stop)
        if result:
            self.work = False
            self.goods_time = None
            self.stop_time = dt.now()
            self.bt_switcher(stop=True, upload=True, unload=True)
            self.bt_switcher(turn=True, start=True)
            messagebox.showinfo(title=msg.info_title,

message=msg.info_end_work.format(time=str(self.stop_time-
self.start_time).split('.')[0]))

    def upload(self):
        if self.in_queue > 0:
            self.bt_switcher(upload=True)
            self.bt_switcher(turn=True, unload=True)
            self.in_queue -= 1
            self.goods_time = dt.now()
        else:
            messagebox.showerror(title=msg.error_title,
message=msg.error_upload)

    def unload(self):
        self.bt_switcher(unload=True)
        self.bt_switcher(turn=True, upload=True)
        self.goods_times.append(dt.now() - self.goods_time)
```

```
        self.total_unload += 1
        self.goods_time = None
        self.aver_time = mean(self.goods_times)

    def get_operations(self):
        print("get_operations")

    def exit(self):
        result    =    messagebox.askyesno(title=msg.confirm_title,
message=msg.ask_exit)
        if result:
            if self.work:
                self.stop_work()
            self.master.after_cancel(self.after_id)
            self.main_frame.destroy()
            self.master.change_frame("login")

    def make_ui(self):
        frames = self.make_frames()

        left_frame, right_frame, start_frm, stop_frm, camera_frm,
time_frm, \
            upload_frm, unload_frm, operator_frm, workspace_frm,
statistic_frm, movements_frm = frames

################################################
# LEFT FRAME
################################################

        self.start_bt = tk.Button(start_frm, text=btn.start_work,
**style.button,
                                  command=lambda:
self.start_work())
        self.start_bt.pack(fill="both", expand=True)

        self.stop_bt   =   tk.Button(stop_frm,   text=btn.stop_work,
**style.button,
                                 command=lambda: self.stop_work())
        self.stop_bt.pack(fill="both", expand=True)

        self.camera_cvs = tk.Canvas(camera_frm)
        self.camera_cvs.pack(fill="both", expand=True)

        self.time_lbl = tk.Label(time_frm, **style.header)
        self.time_lbl.pack(expand=True)

        self.upload_bt   =   tk.Button(upload_frm,   text=btn.upload,
**style.button,
                                   command=lambda: self.upload())
        self.upload_bt.pack(fill="both", expand=True)

        self.unload_bt   =   tk.Button(unload_frm,   text=btn.unload,
**style.button,
```

```
                                    command=lambda: self.unload())
        self.unload_bt.pack(fill="both", expand=True)

################################################
# RIGHT FRAME
################################################

        operator_lbl = tk.Label(operator_frm, text=text.operator,
**style.header)
        operator_lbl.pack()
        op_name_lbl = tk.Label(operator_frm, text=self.op_name,
**style.base)
        op_name_lbl.pack()

        workspace_lbl          =          tk.Label(workspace_frm,
text=text.workspace, **style.header)
        workspace_lbl.pack()
        id_lbl              =              tk.Label(workspace_frm,
text=text.w_id.format(w_id=self.w_id), **style.base)
        id_lbl.pack(anchor="w")
        pos_lbl             =              tk.Label(workspace_frm,
text=text.pos.format(pos=self.pos), **style.base)
        pos_lbl.pack(anchor="w")
        operations_lbl          =          tk.Label(workspace_frm,
text=text.operations, **style.base)
        operations_lbl.pack(side="left")
        operations_btn          =         tk.Button(workspace_frm,
text=btn.operations_list, **style.button,
                                    command=lambda:
self.get_operations())
        operations_btn.pack(side="left", fill="x", expand=True)

        statistic_lbl          =          tk.Label(statistic_frm,
text=text.statistic, **style.header)
        statistic_lbl.pack()
        self.in_queue_lbl = tk.Label(statistic_frm, **style.base)
        self.in_queue_lbl.pack(anchor="w")
        self.total_unload_lbl       =      tk.Label(statistic_frm,
**style.base)
        self.total_unload_lbl.pack(anchor="w")
        self.aver_time_lbl = tk.Label(statistic_frm, **style.base)
        self.aver_time_lbl.pack(anchor="w")

        cur_date_lbl = tk.Label(movements_frm,

text=text.cur_date.format(date=dt.now().strftime('%d.%m.%Y')),
**style.base)
        cur_date_lbl.pack(anchor="se", expand=True)
        self.cur_time_lbl = tk.Label(movements_frm, **style.base)
        self.cur_time_lbl.pack(anchor="se")
        exit_bt    =    tk.Button(movements_frm,    text=btn.exit,
**style.button,
                           command=lambda: self.exit())
```

```
        exit_bt.pack(fill="x")

        self.main_frame.pack(fill="both", expand=True)

        self.bt_switcher(stop=True, upload=True, unload=True)

    def make_frames(self):
        left_frame = tk.Frame(master=self.main_frame)
        right_frame = tk.Frame(master=self.main_frame)

        frames = [left_frame, right_frame]

        for i in range(6):
            frames.append(tk.Frame(master=left_frame,
relief="solid", borderwidth=1, **style.padding_frame))

        for i in range(4):
            frames.append(tk.Frame(master=right_frame,
relief="solid", borderwidth=1, **style.padding_frame))

        frames = self.setup_frames_pos(frames)
        for frame in frames:
            frame.pack_propagate(False)

        return frames

    def setup_frames_pos(self, frames: list):
        left_frame, right_frame, start_frm, stop_frm, camera_frm,
time_frm, \
            upload_frm, unload_frm, operator_frm, workspace_frm,
statistic_frm, movements_frm = frames

        self.main_frame.columnconfigure(index=0, minsize=750)
        self.main_frame.columnconfigure(index=1, minsize=250)
        self.main_frame.rowconfigure(index=0, minsize=700)

        left_frame.grid(row=0, column=0, sticky="nsew")
        right_frame.grid(row=0, column=1, sticky="nsew")

        left_frame.columnconfigure(index=0, weight=1)
        left_frame.columnconfigure(index=1, weight=1)
        left_frame.rowconfigure(index=0, minsize=75)
        left_frame.rowconfigure(index=1, minsize=500)
        left_frame.rowconfigure(index=2, minsize=50)
        left_frame.rowconfigure(index=3, minsize=75)

        right_frame.columnconfigure(index=0, weight=1)
        right_frame.rowconfigure(index=0, minsize=75)
        right_frame.rowconfigure(index=1, minsize=150)
        right_frame.rowconfigure(index=2, minsize=250)
        right_frame.rowconfigure(index=3, minsize=225)

        start_frm.grid(row=0, column=0, sticky="nsew")
```

```
        stop_frm.grid(row=0, column=1, sticky="nsew")
        camera_frm.grid(row=1,        column=0,        columnspan=2,
sticky="nsew")
        time_frm.grid(row=2,         column=0,         columnspan=2,
sticky="nsew")
        upload_frm.grid(row=3, column=0, sticky="nsew")
        unload_frm.grid(row=3, column=1, sticky="nsew")

        operator_frm.grid(row=0, column=0, sticky="nsew")
        workspace_frm.grid(row=1, column=0, sticky="nsew")
        statistic_frm.grid(row=2, column=0, sticky="nsew")
        movements_frm.grid(row=3, column=0, sticky="nsew")

        return  [left_frame,  right_frame,  start_frm,  stop_frm,
camera_frm, time_frm,
                upload_frm,        unload_frm,        operator_frm,
workspace_frm, statistic_frm, movements_frm]
```

login.py

```
import jwt
import tkinter as tk
from tkinter import messagebox

from src.styles import style
from src.models import request, user, workspace, session
from src.content import msg
from config import SECRET_KEY


def check_user():
    decoded_data       =       jwt.decode(jwt=request.access_token,
key=SECRET_KEY, algorithms=["HS256"])
    user.id = decoded_data['user_id']
    user.username = decoded_data['username']
    r = request.get_worker(user.id)
    if r.status_code == 404:
        return False
    data = r.json()
    user.name             =             f"{data.get('first_name')}
{data.get('last_name')}"
    user.status = data.get('status')
    user.operations = data.get('operations')
    return True


def create_session():
    workspace.init_workspace()
    if workspace.compare_operations(user.operations):
        workspace.set_busy(True)
        session.init_session()
        session.set_active(True)
        return True
    return False
```

```
class LoginPage(tk.Frame):
    def __init__(self, master):
        self.master = master
        tk.Frame.__init__(self, self.master)
        self.master.set_window_prop("AUTH SWS v1.0", 250, 215)

        self.login_frame = tk.Frame(self)
        self.make_ui()

    def make_ui(self):
        tk.Label(self.login_frame,              text="Авторизация",
justify="center", **style.header).pack()
        tk.Label(self.login_frame,     text="Имя     пользователя",
**style.base).pack()
        login_entry          =          tk.Entry(self.login_frame,
font=style.font_base)
        login_entry.pack(padx=10)
        tk.Label(self.login_frame,                  text="Пароль",
**style.base).pack()
        password_entry         =         tk.Entry(self.login_frame,
font=style.font_base)
        password_entry.pack(padx=10)
        tk.Button(self.login_frame, text="Войти", **style.button,
                  command=lambda:   self.login_attempt(login_entry,
password_entry)).pack(padx=10, pady=10, fill="x")
        self.login_frame.pack()

    def login_attempt(self, username_ent, password_ent):
        username = username_ent.get()
        password = password_ent.get()
        if username and password:
            if username == '1' and password == '1':
                self.login_frame.destroy()
                self.master.change_frame("main")
```

video.py

```
import cv2


class VideoCapture:
    def __init__(self, video_src = 0):
        print("Loading camera source...")
        self.vid = cv2.VideoCapture(video_src)
        if not self.vid.isOpened():
            raise  ValueError("Источник  видеозахвата  не  найден!",
video_src)

        self.start_hash = self.get_hash(self.get_frame()[-1])
        print(f"LOAD                                          CAMERA:
{self.vid.get(cv2.CAP_PROP_FRAME_WIDTH)}x{self.vid.get(cv2.CAP_PRO
P_FRAME_HEIGHT)}")
```

```
    def __del__(self):
        if self.vid.isOpened():
            self.vid.release()

    def get_frame(self):
        if self.vid.isOpened():
            flag, img = self.vid.read()
            if flag:
                return           (flag,           cv2.cvtColor(img,
cv2.COLOR_BGR2RGB))
            else:
                return (flag, None)
        else:
            return (False, None)

    def get_hash(self, img):
        resized       =       cv2.resize(img,       (8,       8),
interpolation=cv2.INTER_AREA)
        gray_img = cv2.cvtColor(resized, cv2.COLOR_BGR2GRAY)
        avg = gray_img.mean()
        ret,  threshold_image  =  cv2.threshold(gray_img,  avg,  255,
0)

        _hash = ""
        for x in range(8):
            for y in range(8):
                val = threshold_image[x, y]
                if val == 255:
                    _hash = _hash + "1"
                else:
                    _hash = _hash + "0"

        return _hash

    def compare_hash(self, cur_hash):
        l = len(self.start_hash)
        i = 0
        count = 0
        while i < l:
            if self.start_hash[i] != cur_hash[i]:
                count += 1
            i += 1
        return count
```

Министерство науки и высшего образования РФ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

Кафедра «Системы автоматики,
автоматизированное управление и проектирование»

УТВЕРЖДАЮ
Заведующий кафедрой
_____ А.С. Климов

«_22_» июня 2023 г.

БАКАЛАВРСКАЯ РАБОТА

15.03.04 – Автоматизация технологических процессов и производств

АВТОМАТИЗИРОВАННАЯ СИСТЕМА ОПЕРАТИВНОГО КОНТРОЛЯ ПОЗАКАЗНОГО ПРОИЗВОДСТВА РАДИОЭЛЕКТРОННОЙ АППАРАТУРЫ

			доцент, канд. техн. наук
Руководитель	_____	21.06. 2023 г.	О.В. Дрозд
Выпускник	_____	21.06. 2023 г.	И.А. Любухина
Нормоконтролер	_____	21.06. 2023 г.	доцент, канд. техн. наук О.В. Дрозд

Красноярск 2023