ОЛИМПИАДА ШКОЛЬНИКОВ «ШАГ В БУДУЩЕЕ»

НАУЧНО-ОБРАЗОВАТЕЛЬНОЕ СОРЕВНОВАНИЕ
«ШАГ В БУДУЩЕЕ, МОСКВА»

регистрационный номер

Информатика и системы управления

Системы обработки информации и управления

Автоматизация процессов работы предприятия путем разработки программного модуля «ТЕХНОЛОГ» и интеграции его в облачную платформу Битрикс24

Автор:

Ткаченко Владислав Львович,
МБОУ СОШ №13 города Белгорода

Научный руководитель:

Новогрудская Ольга Павловна,
МБОУ СОШ №13 города Белгорода,
учитель информатики

_____

подпись научного руководителя

Консультант:

Астахов Данил Владимирович,
Positive Technologies,
разработчик

Москва - 2020

Автоматизация процессов работы предприятия путем разработки программного модуля «ТЕХНОЛОГ» и интеграции его в облачную платформу Битрикс24

Аннотация

Данный проект – значимая часть запланированной и частично реализованной системы действий для автоматизации производства, проводимой по заказу реального предприятия по производству металлоконструкций. Специфика производства на предприятии не позволяла использовать непосредственные возможности платформы Битрикс24 для удобного обмена данными и планирования производственных задач. Что, в свою очередь, затрудняло и замедляло процесс планирования и выполнения экономических расчетов получения прибыли.

Было предложено принять участие в работе в качестве разработчика одного из программных модулей.

Требовалось выполнить программный модуль как приложение на языке системы Битрикс24, интегрируемое в общую платформу автоматизации и включающее в себя две рабочих программы: библиотеку изделий с необходимыми параметрами и алгоритм преобразования параметров изделий в параметры планирования производства.

Цель:

Разработать программный модуль «ТЕХНОЛОГ» и интегрировать его с облачную платформой Битрикс24 для автоматизации процессов работы предприятия по производству металлоконструкций.

Задачи:

1) Ознакомиться с требованиями, предъявляемыми к модулю «Технолог»;
2) Определить способ интеграции программного модуля с Битрикс24;
3) Спроектировать архитектуру приложения;
4) Выбрать стек технологий и языков программирования;
5) Разработать приложение – модуль «Технолог»;
6) Интегрировать программу с Битрикс24;
7) Добиться стабильной работы у конечного пользователя.

Содержание

Введение

Данный проект – значимая часть системы действий для автоматизации производства, проводимой по заказу реального предприятия металлоконструкций. Предприятие производит более тысячи сборных и односоставных металлических деталей, имеющих разные характеристики. Сами характеристики деталей описывались в виде базы данных в формате Excel, где таблица «Отгрузочная ведомость» (Приложение А) включала более 20 полей разных типов. Руководители предприятия и экономический отдел тратили много времени на сбор, классификацию, обработку и преобразование этих данных. Это затрудняло и замедляло процесс планирования и выполнения экономических расчетов получения прибыли.

Обдумывая возможности автоматизации информационных процессов, заказчик столкнулся с ограничениями функциональности платформы Битрикс24, которая не позволяла реализовать функционал загрузки данных проекта в формате Excel, и последующую графическую визуализацию и проверку этих данных. Без дополнительного специфического модуля, платформа Битрикс24 была не способна анализировать параметры выполненных проектов, а, следовательно, не позволяла просматривать статистику, и вести какие-либо экономические расчеты.

Исходя из Технического задания, разработанного руководителем отдела проектирования и инженером-проектировщиком (Приложение Б), Модуль «Технолог» должен был автоматизировать процесс сбора, классификации и обработки данных о спроектированных деталях и изделиях и преобразования их в данные для планирования рабочих задач на производстве. Требовалось выполнить программный модуль как приложение на языке системы Битрикс24, интегрируемое в общую платформу автоматизации и включающее в себя две рабочих программы: библиотеку изделий с необходимыми параметрами и алгоритм преобразования параметров изделий в параметры планирования производства.

Таким образом, актуальность данного проекта достаточно очевидна и

своевременна для работающего производства.

Было предложено принять участие в работе в качестве веб-разработчика одного из программных модулей, а точнее, модуля «Технолог». Такое предложение заинтересовало возможностью применения знаний в области веб-программирования, получения опыта взаимодействия между участниками проекта: заказчиком (пользователями предприятия), менеджером проекта, инженером-проектировщиком, работающим с данными на платформе Битрикс24. Кроме этого, по устной договоренности с менеджером проекта, при успешной реализации модуля «Технолог», предполагалась денежная премия.

Цель:

Разработать программный модуль «ТЕХНОЛОГ» и интегрировать его с облачную платформой Битрикс24 для автоматизации процессов работы предприятия по производству металлоконструкций.

Задачи:

1) Ознакомиться с требованиями, предъявляемыми к модулю «Технолог»;
2) Определить способ интеграции программного модуля с Битрикс24;
3) Спроектировать архитектуру приложения;
4) Выбрать стек технологий и языков программирования;
5) Разработать приложение – модуль «Технолог»;
6) Интегрировать программу с Битрикс24;
7) Добиться стабильной работы у конечного пользователя.

Основная часть

1 Аналитический раздел

В данном разделе будет произведен анализ предметной области, технического задания и платформы Битрикс24, а также будут выработаны функциональные требования к проекту.

Первый шаг данной работы - изучение возможностей и спецификации платформы Битрикс24, что позволит представить основные способы интеграции и выбрать один из них.

1.1 Функциональные требования к проекту

На основе технического задания, предоставленного заказчиком, были выработаны следующие функциональные требования:

1) Загрузка в приложение архива проекта, который включает «Отгрузочную ведомость» (приложение А) и чертежи деталей (Приложение В);
2) Автоматическое создание библиотеки деталей на основе загруженного архива;
3) Представление библиотеки деталей в пользовательском интерфейсе в виде карточек;
4) Поддержка разных типов карточек деталей;
5) Карточка деталей должна включать поля детали, которые автоматически заполняются, и чертеж детали;
6) Возможность изменения полей, и последующее сохранение;
7) Валидация полей ввода;
8) Режим просмотра чертежа;
9) Выделение карточек, которые содержат ошибки;
10) Поддержка неограниченной вложенности деталей в сами детали;
11) Поиск детали по названию;
12) Фильтрации деталей по категориям;

13) Удаление библиотеки деталей;
14) Вкладка с экономическими расчетами;
15) Загрузка Excel таблицы “Нормы экономиста”;
16) Расчет трудозатрат;
17) Расчет массы материалов с отходами;
18) Печать материалов и трудозатрат.

1.2 Битрикс24

Битрикс24 - российский кроссплатформенный сервис для управления бизнесом. Сервис запущен 12 апреля 2012 года. Прародителем «Битрикс24» стал корпоративный портал, который существовал на базе CMS «1С-Битрикс: Управление сайтом» с 2008 года. Идея создать отдельный SAAS-сервис появилась в 2010 году на одной из традиционных ежегодных стратегических сессий. Сервис позволяет управлять бизнесом в режиме «одного окна». Существует облачная версия и версия для установки на собственный сервер с возможностью доработки [1].

1.3 Возможности интеграции программного модуля с Битрикс24

В Битрикс24 существуют инструменты, упрощающие вопросы интеграции. Однако возникает необходимость изучения вопросов безопасности таких интеграций. Их безопасность обеспечивается ограниченным контекстом использования. Такими инструментами являются локальные приложения и webhooks, поэтому далее будем рассматривать именно их [2].
В рамках конкретного проекта есть два варианта расширения функциональных возможностей Битрикс24 на основе REST API:

- локальные приложения;
- пользовательские webhooks.

1.4 Локальные приложения

Локальные приложения подходят для тех задач, которые требуют создания пользовательского интерфейса: дополнительные автообработчики в рамках специфической бизнес-логики, различные отчеты, дополнительные операции для автоматических бизнес-процессов Битрикс24. Локальные приложения имеют следующие виды:

- Статичные приложения.
- Серверные приложения.

Статичные приложение – такие, которые состоят только из клиентской части в виде пользовательского интерфейса. Реализуются при помощи технологий: HTML, CSS, JavaScript. Этот вариант подходит для приложений, которым не требуется хранить данные пользователя дольше одной сессии.

Серверные приложения – такие, которые состоят из клиентской и серверной частей. Пользовательский интерфейс реализуется аналогично интерфейсу статичных приложений. Backend может быть разработан на разных языках программирования: PHP, Python, Ruby, Java, C#. Наличие дополнительного программного компонента в виде сервера приложений подразумевает разделение логики приложения между сервером и клиентом, хранение данных на сервере и обмен информацией по сети [3].

1.5 Пользовательские webhooks

Альтернативной возможностью интеграции являются пользовательские webhooks используются для организации низкопроизводительного импорта-экспорта данных, использования в автоматизации обработки транзакций и сделок (триггеры и действия в CRM-роботах), малофункциональных интеграций с внешними и внутренними системами компании (ERP, KPI, авто-мониторинг программных и аппаратных средств и т.д.) [4].

1.6 Выбор способа интеграции

Так как программный модуль предполагает наличие пользовательского интерфейса, то был выбран вариант интеграции на основе локального

серверного приложения.

По результатам сравнения двух видов локальных приложений и с учетом требований заказчика был сделан вывод о необходимости использования централизованного структурированного хранилища данных. Поскольку Битрикс24 не имеет внешней реализации такого хранилища, было принято решение об использовании БД, расположенной на удаленном сервере. Что повлекло за собой необходимость разработки веб-приложения для последующей интеграции с Битрикс24.

1.7 Вывод

Были выработаны функциональные требования к проекту, изучены и предложены способы интеграции программного модуля с Битрикс24 и был выбран наиболее подходящий из них.

2 Технологический раздел

Исходя из функциональных и нефункциональных требований, способа интеграции и использования облачной версии Битрикс24, разрабатываемой системой будет веб-приложение. Оно будет разбито на четыре независимые подсистемы: клиент, веб-сервер, сервер приложений, СУБД. Веб-сервер, сервер приложений и СУБД будут находится на одном сервере, который предоставляет заказчик. Клиент будет взаимодействовать с веб-сервером при помощи браузера по протоколу HTTP.

В данном разделе будет выбран тип разрабатываемого веб-приложения, представлена декомпозиция подсистем клиент-серверной архитектуры и описаны важные свойства системы, на которые рекомендует обращать внимание С. Макконнелл [5].

2.1 Архитектура приложения

Важным вопросом на момент разработки архитектуры стал выбор типа разрабатываемого веб-приложения. Существует два основных: одностраничные приложения (SPA) и многостраничные приложения (MPA).

Одностраничное приложение (single page application, SPA) — это веб-приложение или веб-сайт, использующий единственный HTML-документ как оболочку для всех веб-страниц и организующий взаимодействие с пользователем через динамически подгружаемые HTML, CSS, JavaScript, обычно посредством AJAX. При этом логика пользовательского интерфейса приложения выполняется преимущественно в веб-браузере, а взаимодействие с веб-сервером осуществляется главным образом через веб-API.

Многостраничное приложение (Multi page application, MPA) — это веб-приложение или веб-сайт, состоящее из нескольких и более веб-страниц со статической информацией и ссылок на другие страницы с одинаковым

содержанием. Это традиционные веб-приложения, большая часть логики которых выполняется на сервере [6].

Проанализировав и сравнив преимущества и недостатки двух вариантов из таблицы Приложения Г, было принято решение об использовании SPA-приложения.

2.2 Клиент-серверная архитектура

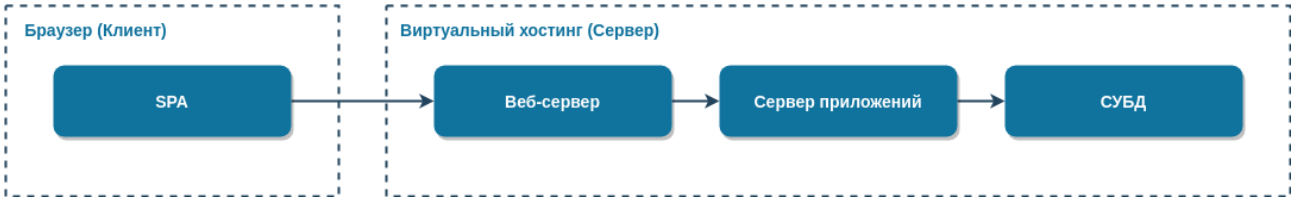


Рисунок 1 — Подсистемы веб-приложения

Основные функции подсистем:

- SPA является клиентом с широкой функциональностью, высоким быстродействием и интерактивностью, представленном в виде одной страницы.
- Веб-сервер принимает HTTP-запросы от клиентов, передает далее управление серверу приложений и в конце возвращает ответ обратно клиенту.
- Сервер приложений обеспечивает обработку данных
- СУБД занимается хранением данных

2.2.1 Архитектура клиента

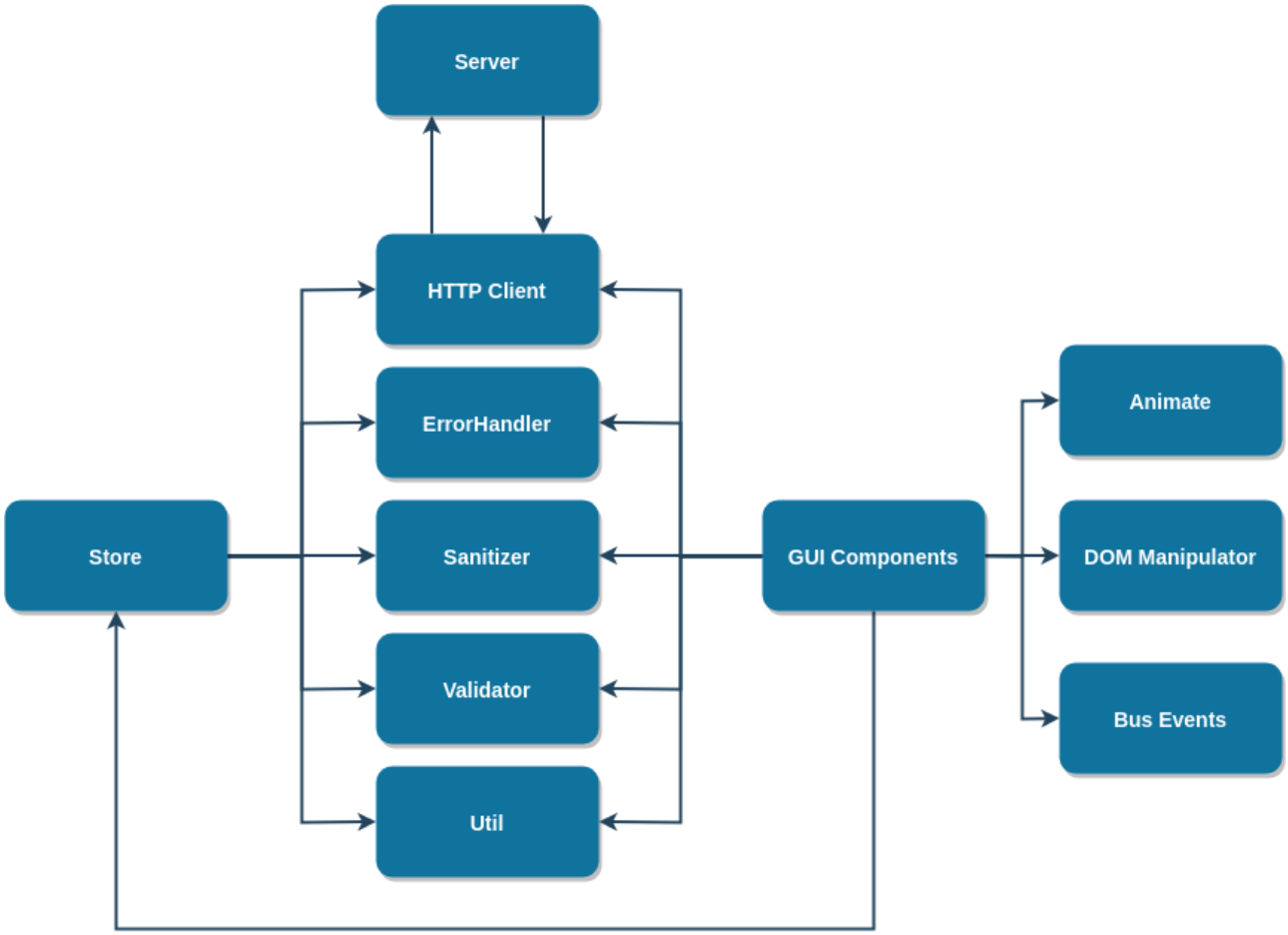


Рисунок 2 — Основные компоненты клиента

Store – источник истинных данных. HTTP Client, ErrorHandler, Sanitizer, Validator, Util – утилитарные компоненты. GUI Components – дерево компонентов пользовательского интерфейса. DOM Manipulator – Объект, производящий операции с DOM деревом.

2.2.2 Архитектура сервера

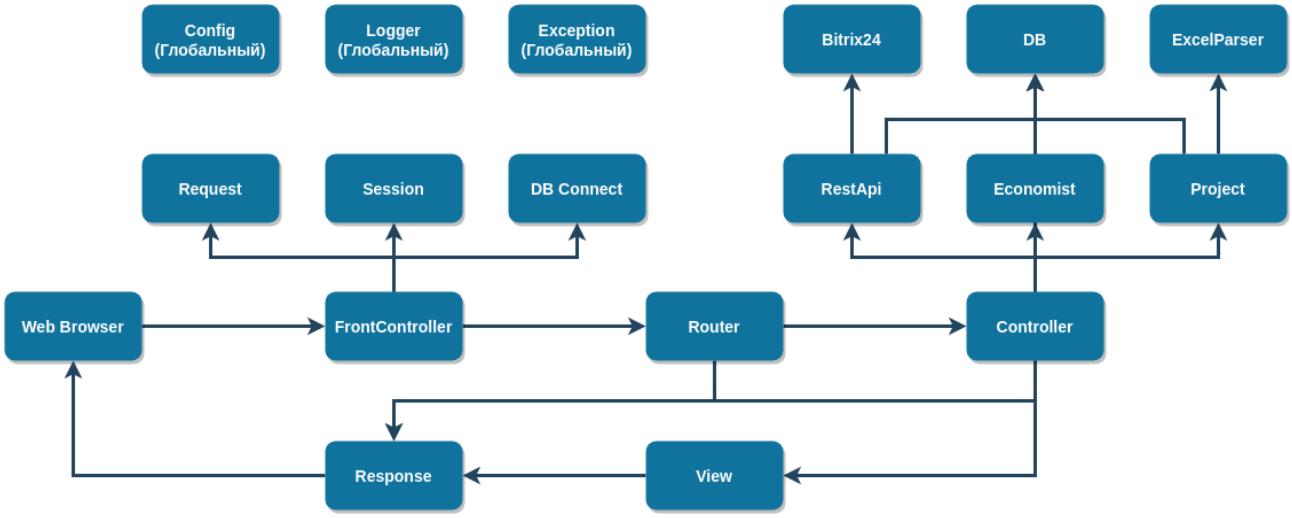


Рисунок 3 — Основные компоненты сервера

Logger, Config, Exception, Session – глобальные утилитарные объекты. FrontController, Request, Router, Controller, View, Response, DB – компоненты, реализующие архитектурный шаблон MVC. Economist, Project, ExcelParser, RestApi, Bitrix24 – классы, отвечающие за бизнес-логику.

2.3 Организация данных

Согласно требованиям заказчика входные данные представлены в виде таблицы Excel формате .xls «Отгрузочная ведомость» (Приложение А). База данных будет использована для хранения записей из «Библиотеки изделий», которые загружаются из таблицы «Отгрузочная ведомость».

2.4 Бизнес-правила

Каждое изделие должно отображаться в виде «карточки изделия» в пользовательском интерфейсе. Поля карточки заполняются автоматически. Запасной способ заполнения – возможность заполнить поля «вручную». По «Библиотеке изделий» должен быть интерактивный поиск. Необходимо наличие кнопки «Загрузить ведомость», как компонента пользовательского интерфейса и возможность создавать, удалять и редактировать «Библиотеку изделий». В «Карточке детали» должен находиться чертеж изделия.

2.5 Управление ресурсами

Так как клиентов у веб-приложения не более 50 человек, то нагрузка на сервер приложения будет минимальной и проблем с ограниченности ресурсов не предполагается.

2.6 Безопасность

Необходимо обеспечить безопасное хранение логина и паролей от БД. Сервер приложения должен обеспечивать тщательную валидацию принимаемых данных и проверку клиентов, которые запрашивают данные.

2.7 Производительность

Ввиду небольшого количества пользователей приложения, проблем с производительностью на стороне сервера не предполагается.
Так как клиент имеет расширенную функциональность и вся работа по обработке и представлению данных переносится на него, то возможна потеря быстродействия на его стороне, поэтому при выборе frontend-технологий следует обратить внимание на их производительность.

2.8 Масштабируемость

Проблема масштабируемости для данного приложения не актуальна, потому что не планируется увеличение числа пользователей серверов и сетевых узлов. Записей в БД и транзакций становится больше, но их количество остается в пределах нормы.

2.9 Взаимодействие с другими системами

Предполагается взаимодействие веб-приложения с Битрикс24 при помощи REST API.

2.10 Обработка ошибок

Стратегия обработки ошибок планируется как пассивно-корректирующая. Программа будет пытаться восстановиться от последствий ошибки и уведомлять пользователя, если избежать их не удалось.

2.11 Отказоустойчивость

При возникновении критической ошибки и невозможности восстановления от ее последствий на стороне сервера, необходимо обеспечить уведомление пользователя об этом в пользовательском интерфейсе.

Одним из способов восстановления системы при возникновении ошибки должна являться замена ошибочного значения поддельным значением, которое положительно скажется на работе системы.

2.12 Возможность реализации архитектуры

Реализация каждого компонента веб-приложения возможна, поэтому система технически осуществима.

2.13 Купить или создавать самим

Проанализировав marketplace Битрикс24, было установлено, что подобных приложений с нужным функционалом еще не разработано. Поэтому остановились на создании собственного специфического программного модуля.

2.14 Стратегия изменений

Все вероятные изменения форматов ввода или вывода данных, стиля взаимодействия с пользователями или требований к обработке, были предусмотрены и каждое было ограничено.

2.15 Вывод

Итоги технологический раздела: выбран тип разрабатываемого веб-приложения, декомпозированы подсистемы клиент-серверной архитектуры, описаны важные свойства системы.

3 Конструкторский раздел

В рамках данного раздела будет произведен выбор технологий, а также описаны основные использованные технические решения.

3.1 Клиентская часть

Раздел посвящен разработке клиента (SPA), выбору технологий для его реализации

3.1.1 Выбор языка программирования

Для реализации клиента выбран язык программирования JavaScript, потому что Браузеры предоставляют только этот язык для реализации клиентских сценариев веб-приложения.

3.1.2 Выбор используемых фреймворков

SPA предполагает разработку клиента с широкой функциональностью, высоким быстродействием и интерактивностью. Большую часть обработки и представления данных обеспечивает клиент, а сервер выступает в роли хранилища данных. В итоге клиент декомпозирован на большое количество компонентов на этапе разработки архитектуры. Ввиду этого решено было использовать большую часть компонентов в виде готовых библиотек, чтобы не разрабатывать все с нуля и не затягивать сроки выполнения проекта.

3.1.3 Выбор утилитарных библиотек

Выбор JavaScript библиотек производился с учетом следующих критериев: максимальная совместимость с Vue.js, популярность, простота, опыт использования. На их основе были выбраны следующие библиотеки:

- Vuex - паттерн управления состоянием и библиотека для приложений на Vue.js. Реализует Store;
- Axios - широко известная JavaScript-библиотека, используется в примерах документации Vue.js. Реализует HTTP-Client;
- jQuery - набор функций JavaScript, фокусирующийся на взаимодействии JavaScript и HTML. Реализует DOM Manipulator;
- Lodash - библиотека JavaScript, которая предоставляет вспомогательные функции для общих задач программирования с использованием парадигмы функционального программирования. Реализует Util;
- Velocity - кроссплатформенная JavaScript-библиотека, предназначенная для упрощения создания сценариев анимации на стороне клиента. Реализует Animate.

Остальные программные части клиента: ErrorHandler, Validator, Sanitizer, GUI Components - были реализованы самостоятельно

3.1.4 Реализация клиента

Основная часть работы велась над деревом пользовательских компонентов. Верхний слой дерева представлен на Рис. 4.

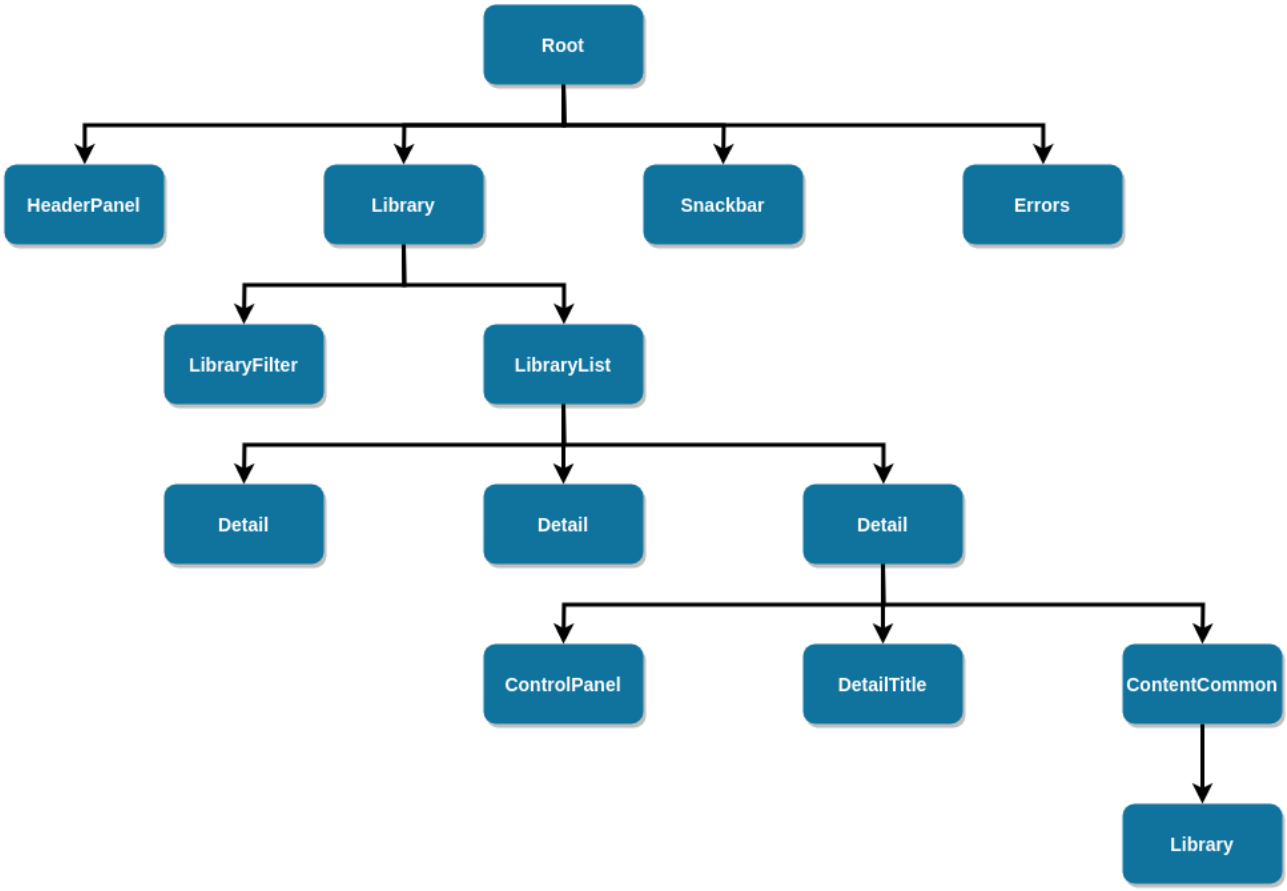


Рисунок 4 — Верхний слой дерева компонентов

Наиболее интересной в реализации стала функция рекурсивной вложенности деталей. Компонент ContntCommon получает объект detail от своего родительского компонента - Detail. Далее просматривается свойство children полученного объекта detail, и, если оно является не пустым массивом, тогда ContentCommon рендерит дополнительную кнопку в карточке деталей - “Просмотреть детали”. По нажатии данной кнопки пользователем происходит рекурсивных рендеринг компонента Library прямо в ContentCommon. Данная реализация поддерживает неограниченную вложенность деталей.

Рассмотрим подробнее компонент ErrorHandler, который представлен в виде класса BaseError и подклассов: JsonError, ServerError, LogicError, для представления типов ошибок. Основная обработка ведется в классе BaseError (Приложение Е.1). Конструктор класса принимает объект хранилища (Store) и сообщение об ошибке. Далее происходит добавление ошибки в очередь, затем обновляется хранилище, что влечет за собой повторный рендеринг компонента Snackbar, который показывает пользователю всплывающую карточку с ошибкой. После чего Snackbar посылает запрос в

Store на удаление ошибки из очереди.

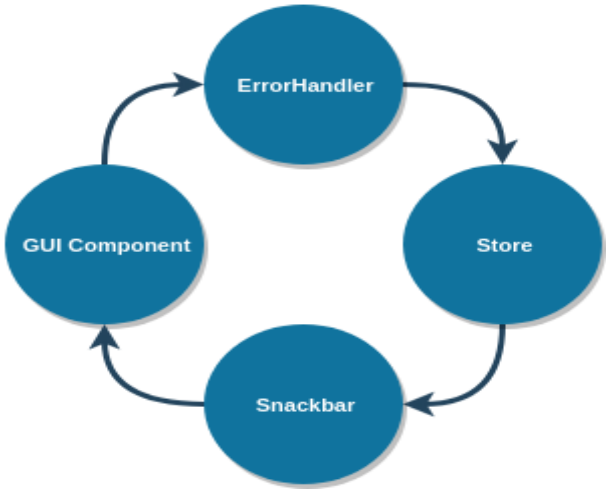


Рисунок 5. Процесс обработки ошибок

3.2 Серверная часть

Раздел посвящен разработке сервера приложений, выбору технологий для его реализации

3.2.1 Выбор языка программирования

Для реализации сервера выбран язык программирования PHP, потому что он поддерживается подавляющим большинством хостинг-провайдеров, в частности, предоставляемый заказчиком хостинг поддерживает только PHP. Дополнительным аргументом стало наличие опыта разработки программ на этом языке у автора проекта.

3.2.2 Выбор используемых фреймворков

В данном проекте сервер приложений не является сложной подсистемой с широкой функциональностью, поэтому решено было выбрать в виде фреймворка самописное, ранее разработанное решение, которое зарекомендовало себя в предыдущих проектах. Фреймворк называется PHP Lite, его исходный код представлен в GitHub репозитории на следующей странице - https://github.com/webkadiz/php-lite. Инструмент состоит из

компонентов, реализующих архитектурный шаблон MVC, и дополнительных утилитарных классов, которые используются для конфигурирования системы, контроля сессии, работы с миграциями, логирования событий. Этих возможностей было достаточно для реализации проекта.

Также фреймворк имеет свой шаблон структуры папок и файлов проекта, PHP Lite Template, находящийся в следующем GitHub репозитории - https://github.com/webkadiz/php-lite-template.

3.2.3 Выбор утилитарных библиотек

Единственной сторонней библиотекой является PHPExcel, которая будет осуществлять парсинг Excel файлов. Весь остальной функционал сервера приложений реализован самостоятельно

3.2.4 Реализация сервера

Основные этапы реализации сервера приложений

Во-первых, была разработана структура запросов и механизм их обработки, так как сервер приложений в первую очередь принимает запрос от клиента, а потом в зависимости от запроса выполняет действие.

В фреймворке PHP Lite уже был реализован подобный механизм, поэтому оставалось предложить запросы в соответствии правилам фреймворка.

3.2.5 Маршрутизация

В Приложении Е.2 представлен файл конфигурации, в котором определены все запросы в поле routes. В качестве ключей к маршрутам выступают регулярные выражения, а в качестве маршрутов строка в формате контроллер/действие. Особенность проверки: URL-путь запроса сопоставляется с каждым регулярным выражением, если проверка проходит, то цикл проверки прекращается и запускается соответствующий контроллер и его действие. Если же ни одно регулярное выражение не подошло, то возвращается так называемая

страница 404 - Not Found. Рассмотрим первый маршрут, technologist/project/create. Форма записи URL: <схема>:[//[<логин>[:<пароль>]@]<хост>[:<порт>]][/<URL-путь>][?<параметры>][#<якорь>] [7]. Если компонент URL, URL-путь, будет эквивалентен technologist/project/create, тогда вызывается обработчик, соответствующий формату записи маршрута, technologist/createProject, для поиска класса TechnologistController и метода в нем, createProjectAction. Если класс и метод найдены, то управление передается им, иначе возбуждается исключение, и класс Response возвращает страницу 404. Наглядно этот процесс представлен ниже.

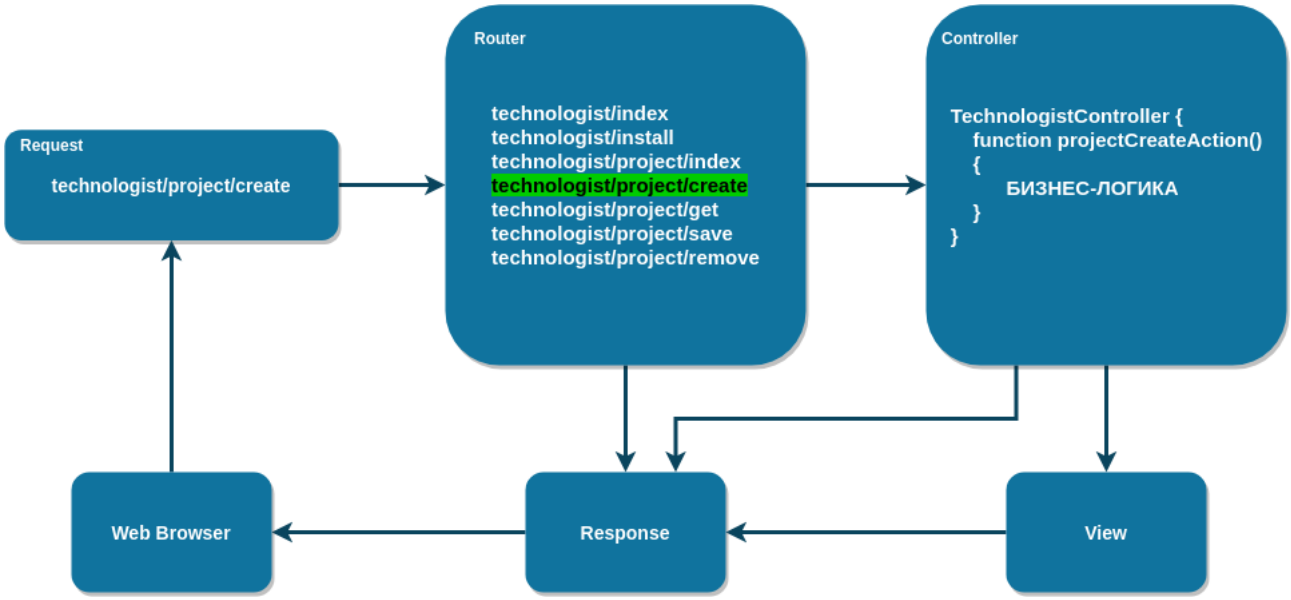


Рисунок 6 — Процесс работы маршрутизации

Фактически представленные маршруты в Приложении Д отражают все функции сервера приложения:

- technologist/index – выполнить рендеринг страницы приложения;
- technologist/install - установить приложение в Битрикс24;
- technologist/indexProject - выполнить рендеринг страницы проекта;
- technologist/createProject - создать проект;
- technologist/getProject - получить данные проекта из БД;
- technologist/saveProject - обновить данные проекта в БД;
- technologist/removeProject - удалить проект;
- technologist/loadEconomist - загрузить таблицу «Нормы экономиста»;

- technologist/computeMaterials - рассчитать материалы;
- technologist/computeLaborCosts - рассчитать трудозатраты;
- technologist/getFields - получить поля для карточки деталей.

Все эти функции сосредоточены в одном контроллере, поэтому следующим шагом стала разработка класса TechnologistController. Рассмотрим несколько функций более подробно.

Функция technologist/indexAction (Приложение Е.3) инициируется самим Битрикс24, когда пользователь переходит на вкладку приложения. При этом Битрикс передает GET и POST параметры авторизации и аутентификации. Поэтому основная задача функции сохранить эти параметры и выполнить рендеринг страницы приложения. Для парсинга, обработки и хранения http параметров и параметров веб-сервера в программе используется класс Request. Он предоставляет возможность получать значения параметров по ключу.

3.2.6 Механизм сессий

HTTP не сохраняет своего состояния, поэтому для сохранения промежуточного состояния между запросами используется механизм сессий, который реализует класс Session. Он представляет собой абстракцию над низкоуровневыми функциями по работе с сессией, которые предоставляет PHP, оставляя всего лишь три метода: set - установить и обновить значение, get - получить значение, unset - удалить значение. Метод renderPartial инкапсулирует обращение к компонентам View и Response. Внутри метода View находит автоматически шаблон, для которого нужно выполнить рендеринг, используя маршрут, найденный и предоставленный Router. В данном случае функции indexAction соответствует маршрут technologist/index, поэтому View будет искать шаблон по пути views/technologist/index.php относительно корня проекта. Далее шаблон будет обработан, буферизован и передан для отправки компоненту Response, который установит заголовки и отправит данные клиенту в правильном формате.

Так, в Приложении Е.4 представлена функция technologist/installAction, основной задачей которой является регистрация приложения в интерфейсе Битрикс24 путем отправки http-запроса. Функция использует 6 компонентов: Request, Session, Config, Logger, RestApi, Response. Метод makeRequest класса RestApi предоставляет интерфейс для создания http-запросов в соответствии REST API Битрикс24. Он принимает первым аргументом метод, а вторым - параметры, и возвращает ответ в формате JSON. Далее компонент Logger логирует ответ, который пришел от Битрик24, удаляется поле auth_id из сессии и в контроллере вызывает метод отправки данных в формате JSON, который использует компонент Response (Приложение Е.5), при этом этап View пропускается, так как ответ не подразумевает рендеринга шаблона.

В функции installAction используется класс конфигурации Config, сохраняющий все поля из файлов конфигурации. С точки зрения гибкости и сопровождения ПО этот компонент очень удобен и важен. В дальнейшем значение поля TITLE (название приложения в пункте меню) можно будет легко изменить, используя лишь файл конфигурации.

3.3 Развертывание приложения на рабочий сервер

Для развертывания приложения на рабочем сервере были предоставлены заказчиком авторизационные данные для доступа к виртуальному хостингу по ssh, логин и пароль от пустой базы данных, которая была создана на сервере предварительно.

Далее процесс перевода приложения на рабочий сервер осуществлялся следующими этапами:

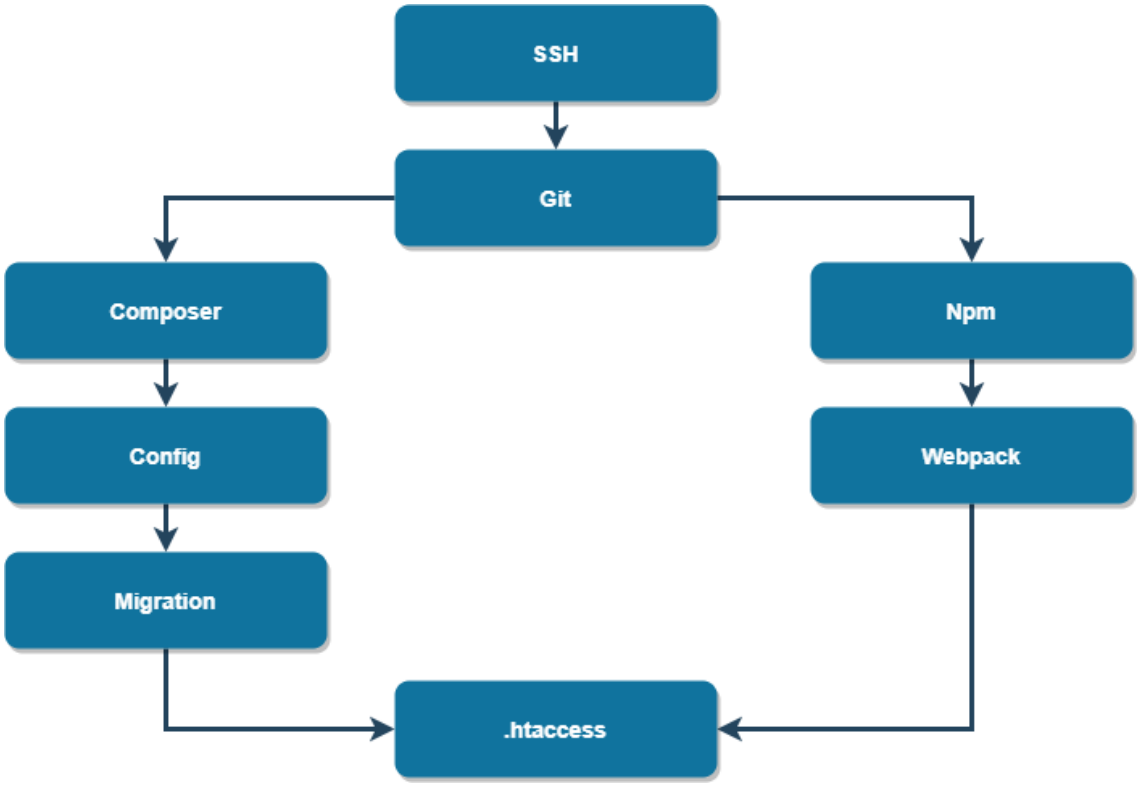


Рисунок 7 — Этапы развертывания приложения на рабочий сервер

3.4 Выводы

Итоги данного раздела: выбран стек технологий для клиентской части и произведена ее разработка, выбран стек технологий для серверной части и произведена ее разработка, выполнено развертывание приложения на рабочем сервере.

4 Схема взаимодействия с ПО

В итоге приложение выполняет свою главную функцию - создание библиотеки деталей - максимально автоматизированно для пользователя. Проектировщик металлоконструкций после разработки чертежей в ПО SCAD, экспортирует из него архив zip, содержащий набор excel таблиц и файлов чертежей в формате pdf. Далее загружает полученный архив (Приложение Ж.1). И после нескольких секунд получает огромную библиотеку деталей прямо в пользовательском интерфейсе (Приложение Ж.2).

Карточки с ошибками помечаются красной рамкой. Далее пользователь может открыть эту карточку и исправить в ней поля с ошибками. Нижнюю часть карточки, поля которой являются ошибочными, можно увидеть в Приложении Ж.3, верхнюю - в Приложении Ж.4. Функций просмотра чертежа представлена в Приложении Ж.5

Для удобного нахождения нужной детали в библиотеки предусмотрена функция фильтрации деталей по категориям (Приложение Ж.6) и функция поиска детали по названию (Приложение Ж.7).

Функция рекурсивного рендеринга библиотеки деталей уже в самих деталях представлена в Приложении Ж.8.

В приложении присутствует вкладка с экономическими расчетами: материалов (Приложение Ж.9) и трудозатрат текущего проекта (Приложение Ж.10). Для расчета используется загруженная экономистом в приложение excel таблица “Нормы экономиста”. Оттуда берутся начально предложенные “Нормы выработки в час” и “Нормы отходов”. Экономист всегда может их поменять, загрузив новую измененную таблицу. Потом эти значения используются в формулах для расчета массы материала с отходами и трудозатрат. Есть возможность распечатать трудозатраты и материалы (Приложение Ж.11).

Заключение

В рамках данной работы была достигнута цель и решены все поставленные задачи:

- Выработаны функциональные требования;
- Изучен Битрикс24 и выбран способ интеграции программного модуля;
- Спроектирована архитектура приложения;
- Выбран стек технологий и языков программирования;
- Разработано веб-приложение – модуль «Технолог»;
- Приложение интегрировано с Битрикс24;
- Работа приложения стала стабильной у конечного пользователя.

Модуль «Технолог» является одной из трех частей проекта. Дальнейшим развитием заключается в разработка оставшихся двух модулей: «Планировщик» и «Экономист».

Список используемых источников

1. Битрикс 24 [Электронный ресурс] // Википедия [сайт]. [2019]. URL: https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D1%82%D1%80%D0%B8%D0%BA%D1%8124(https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D1%82%D1%80%D0%B8%D0%BA%D1%8124) (дата обращения 10. 09.2019).
2. Локальные приложения и вебхуки [Электронный ресурс] // Приложения Битрикс 24 [сайт]. [2019]. URL: https:// https://dev.1c-bitrix.ru/learning/course/index.php?COURSE_ID=99&CHAPTER_ID=08583 (https://dev.1c-bitrix.ru/learning/course/index.php?COURSE_ID=99&CHAPTER_ID=08583) (дата обращения 15. 09.2019).
3. Локальные приложения. Быстрый старт [Электронный ресурс] // Приложения Битрикс 24 [сайт]. [2019]. URL: https://dev.1c-bitrix.ru/learning/course/index.php?COURSE_ID=99&CHAPTER_ID=08593&LESSON_PATH=8771.8583.8593 (https://dev.1c-bitrix.ru/learning/course/index.php?COURSE_ID=99&CHAPTER_ID=08593&LESSON_PATH=8771.8583.8593) (дата обращения 17. 09.2019).
4. Веб-хуки. Быстрый старт [Электронный ресурс] // Приложения Битрикс 24 [сайт]. [2019]. URL: https://dev.1c-bitrix.ru/learning/course/index.php?COURSE_ID=99&LESSON_ID=8581&LESSON_PATH=8771.8583.8581 (https://dev.1c-bitrix.ru/learning/course/index.php?COURSE_ID=99&LESSON_ID=8581&LESSON_PATH=8771.8583.8581) (дата обращения 17. 09.2019).
5. Макконнелл С. Совершенный код. Мастер-класс / Пер. с англ. — М.: Издательство «Русская редакция», 2010. — 896 стр.: ил..
6. Одностраничные и многостраничные веб-приложения [Электронный ресурс] // OZI. Web-Experts [сайт]. [2019]. URL: https://ozitag.com/ru/blog/spa-advantages/ (https://ozitag.com/ru/blog/spa-advantages/) (дата обращения 15. 09.2019).
7. Единый указатель ресурса [Электронный ресурс] // Википедия [сайт]. [2019]. URL:https://ru.wikipedia.org/wiki/URL (https://ru.wikipedia.org/wiki/URL) (дата обращения 21. 09.2019).

Приложение А

Таблица «Отгрузочная ведомость»

Ведомость деталей	
Название	Пристройка 5,8х13,9 м
Дата	04.04.2019 14:19:18

									Периметр листа, м		Масса, кг		Площадь покрытия, м2		Отверстия1						
Лист №	Позиция детали	Тип профиля	Профиль	Количество	Длина, мм	Ширина, мм	Высота, мм	Марка стали	1 дет.	Всех	1 дет.	Всех	1 дет.	Всех	Диаметр, мм	Овальное отв. X, мм	Овальное отв. Y, мм	Кол-во в 1 дет.	Кол-во всего, шт	Длина реза в 1 дет., м	Длина реза во всех дет
010	Дет.Пр-1	Лист	-6 х 60	5	156	60	6	С245	0,43	2,16	0,44	2,2	0,02	0,11							
020	Дет.П-5	Профиль ЛСТК	СЖ 200х75х25-ж10-	6	3844	200	75	350	0	0	23,81	142,86	0	0	19	0	0	6	36	0,12	2,15
030	Дет.П-6	Профиль ЛСТК	СЖ 200х75х25-ж10-	2	3844	200	75	350	0	0	23,81	47,62	0	0	15	0	0	2	4	0,05	0,19
040	Дет.П-7	Профиль ЛСТК	СЖ 200х75х25-ж10-	2	3844	200	75	350	0	0	23,81	47,62	0	0	15	0	0	2	4	0,09	0,19
050	Дет.П-8	Профиль ЛСТК	СЖ 200х75х25-ж10-	2	3844	200	75	350	0	0	23,81	47,62	0	0	15	0	0	2	4	0,09	0,19
060	Дет.Р-1	Профиль ЛСТК	СЖ 200х75х25-ж10-	1	3844	200	75	350	0	0	23,81	23,81	0	0	19	0	0	4	4	0,12	0,24
070	Дет.Р-2	Профиль ЛСТК	СЖ 200х75х25-ж10-	2	3844	200	75	350	0	0	23,81	47,62	0	0	19	0	0	6	12	0,12	0,72
080	Дет.Р-3	Профиль ЛСТК	СЖ 200х75х25-ж10-	1	3482	200	75	350	0	0	21,57	21,57	0	0	19	0	0	6	6	0,24	0,36
090	Дет.Р-4	Профиль ЛСТК	СЖ 200х75х25-ж10-	3	3844	200	75	350	0	0	23,81	71,43	0	0	19	0	0	4	12	0,12	0,72
100	Дет.Р-5	Профиль ЛСТК	СЖ 200х75х25-ж10-	2	5629	200	75	350	0	0	34,86	69,72	0	0	19	0	0	6	12	0,12	0,72
110	Дет.Р-6	Профиль ЛСТК	СЖ 200х75х25-ж10-	1	3482	200	75	350	0	0	21,57	21,57	0	0	19	0	0	4	4	0,12	0,24
120	Дет.Р-7	Профиль ЛСТК	СЖ 200х75х25-ж10-	1	5629	200	75	350	0	0	34,86	34,86	0	0	19	0	0	4	4	0,12	0,24
130	Дет.Ст-1	Профиль ЛСТК	СЖ 200х75х25-ж10-	2	968	200	75	350	0	0	5,99	11,98	0	0	19	0	0	4	8	0,12	0,48
140	Дет.1Б-1	Профиль ЛСТК	СЖ 200х75х25-ж10-	2	5810	200	75	350	0	0	35,98	71,96	0	0	19	0	0	34	68	0,12	4,06
150	Дет.1Б-2	Профиль ЛСТК	СЖ 200х75х25-ж10-	6	5810	200	75	350	0	0	35,98	215,88	0	0	19	0	0	32	192	0,12	11,46
160	Дет.1К-1	Профиль ЛСТК	СЖ 200х75х25-ж10-	4	3600	200	75	350	0	0	22,29	89,16	0	0	19	0	0	22	88	0,48	5,25
170	Дет.1К-2	Профиль ЛСТК	СЖ 200х75х25-ж10-	4	4379	200	75	350	0	0	27,12	108,48	0	0	19	0	0	24	96	0,48	5,73
180	Дет.1К-3	Профиль ЛСТК	СЖ 200х75х25-ж10-	2	4379	200	75	350	0	0	27,12	54,24	0	0	19	0	0	24	48	0,48	2,87
190	Дет.1К-4	Профиль ЛСТК	СЖ 200х75х25-ж10-	4	3600	200	75	350	0	0	22,29	89,16	0	0	19	0	0	22	88	0,48	5,25
200	Дет.1К-5	Профиль ЛСТК	СЖ 200х75х25-ж10-	1	4379	200	75	350	0	0	27,12	27,12	0	0	19	0	0	27	27	0,48	1,61
210	Дет.1Ог-1	Лист	-6 х 302	1	381	302	6	С245	1,31	1,31	70	60	0,22	0,22	19	0	0	6	6	0,12	0,36
220	Дет.1Пд-1	Уголок	Уголок100Х8	6	640	100	100	350	0	0	70	6000	0,26	1,55	19	0	0	4	24	0,24	1,43
230	Дет.1П-1	Профиль ЛСТК	СЖ 200х75х25-ж10-	6	5629	200	75	350	0	0	34,86	209,16	0	0	19	0	0	18	108	0,12	6,45
240	Дет.1П-2	Профиль ЛСТК	СЖ 200х75х25-ж10-	2	5629	200	75	350	0	0	34,86	69,72	0	0	15	0	0	2	4	0,05	0,19
250	Дет.1П-3	Профиль ЛСТК	СЖ 200х75х25-ж10-	2	5629	200	75	350	0	0	34,86	69,72	0	0	15	0	0	2	4	0,09	0,19
260	Дет.1П-4	Профиль ЛСТК	СЖ 200х75х25-ж10-	2	5629	200	75	350	0	0	34,86	69,72	0	0	15	0	0	2	4	0,09	0,19
270	Дет.1СтВ-1	Профиль ЛСТК	СЖ 200х75х25-ж10-	2	3805	200	75	350	0	0	23,56	47,12	0	0	19	0	0	16	32	0,12	1,91
280	Дет.2К-5	Профиль ЛСТК	СЖ 200х75х25-ж10-	1	4379	200	75	350	0	0	27,12	27,12	0	0	19	0	0	26	26	0,48	1,55
290	Дет.2Ог-1	Лист	-6 х 226	1	500	226	6	С245	1,45	1,45	5,32	5,32	0,23	0,23	19	0	0	4	4	0,24	0,24
300	Дет.2Пд-1	Уголок	Уголок100Х8	2	640	100	75	350	0	0	7,84	15,68	0,23	0,45	19	0	0	4	8	0,24	0,48
310	Дет.3Ог-1	Лист	-6 х 264	1	500	263	6	С245	1,26	1,26	3,63	3,63	0,16	0,16							
320	Дет.4Ог-1	Лист	-6 х 210	1	263	210	6	С245	0,94	0,94	2,61	2,61	0,12	0,12	19	0	0	4	4	0,24	0,24
0	Косынка-1	Лист	-6 х 200	8	420	200	6	С245	1,1	8,82	3,19	25,52	0,14	1,14							
0	Пр200х60-1	Лист	-6 х 60	111	200	60	6	С245	0,52	57,72	0,57	63,27	0,03	3,01	19	0	0	4	222	0,12	13,25
0	Пр200х60-2	Лист	-6 х 60	3	200	60	6	С245	0,52	1,56	0,57	1,71	0,03	0,08	15	0	0	4	6	0,09	0,28
0	Пятка базы-1	Лист	-20 х 300	8	420	300	20	С245	1,44	11,52	19,78	158,24	0,28	2,25	19	0	0	1	8	0,06	0,48
0	Св-1	Круг	Круг12	2	5440	12	12	Ст3пс	0	0	4,83	9,66	0,2	0,4							
0	Св-2	Круг	Круг12	2	5988	12	12	Ст3пс	0	0	5,32	10,64	0,22	0,44							
0	Св-3	Круг	Круг12	2	6377	12	12	Ст3пс	0	0	5,66	11,32	0,23	0,47							
0	Стенка-1	Лист	-6 х 260	4	420	260	6	С245	1,44	5,74	4,79	19,16	0,21	0,85	19	0	0	8	32	0,48	1,91
0	Стенка-2	Лист	-6 х 260	4	420	260	6	С245	1,3	5,2	4,83	19,32	0,21	0,85	19	0	0	8	32	0,48	1,91
0	УгСв50х5-1	Уголок	Уголок50Х5	4	60	43	50	Ст3пс	0	0	0,23	0,92	0,01	0,05	13	0	0	1	4	0,04	0,16
0	УгСв50х5-2	Уголок	Уголок50Х5	8	59	50	38	Ст3пс	0	0	0,25	10	0,01	0,09	13	0	0	1	8	0,04	0,33

Приложение Б

Техническое задание

Участники рабочего процесса:

Исполнитель задачи - инженер-проектировщик

Постановщик задачи - руководитель Отдела проектирования Наблюдатель -

директор

Описание рабочего процесса:

Программный модуль ТЕХНОЛОГ предназначен для сбора, классификации и обработки данных о спроектированных деталях и изделиях и преобразования их в данные для планирования рабочих задач на производстве.

Программный модуль ТЕХНОЛОГ выполнен как приложение на языке системы Битрикс 24, интегрируемое в общую платформу автоматизации и включает в себя две рабочих программы:

- Библиотеку изделий с необходимыми параметрами,
- Алгоритм преобразования параметров изделий в параметры планирования производства

Для каждого изделия (детали) в ТЕХНОЛОГЕ создается отдельная Карточка. Библиотека изделий в

ТЕХНОЛОГЕ пополняется двумя способами:

- загрузка в приложение таблицы «Отгрузочная ведомость» (основной способ),
- выбор Карточки изделия и внесение данных в поля приложения вручную (дополнительный способ).

Основной способ пополнения Библиотеки - через «Отгрузочную ведомость» осуществляется в цепочке автоматизированных задач инженера- проектировщика как обязательное действие, выполнение которого контролирует Руководитель проектного отдела.

Инженеры-проектировщики в ходе работы над проектом в ПО SCAD вносят в чертежи полный перечень информации, которые необходимы для создания Отгрузочной ведомости полного набора данных для работы ТЕХНОЛОГА.

Кроме уже имеющейся информации, проектировщикам потребуется вносить только информацию о типе сварки, все остальное уже имеется в ПО SCAD.

В результате внесения информации в «Отгрузочной ведомости» появляется полный набор данных о каждом изделии:

1. Наименование с унифицированным кодом
2. Размер
3. Вес
4. Диаметр и число отверстий
5. Площадь зачистки и окраски
6. 6. сортамент и количество метизов

7.Операции, необходимые для изготовления изделия

8.Длина сварного шва

9.Вид сварки

После разработки пакета чертежей «Отгрузочная ведомость» в виде таблицы формата Excel загружается в модуль ТЕХНОЛОГ.

Для гарантированной надежности полноты и качества заполнения Библиотеки в случае пустых полей приложение будет автоматически уведомлять пользователя об их наличии. То есть, если какое-то из полей данных после экспорта Отгрузочной ведомости оказывается незаполненным, то система оповещает об этом и руководитель Отдела проектирования получает уведомление и ставит задачу инженеру внести недостающие данные в модуль ТЕХНОЛОГ напрямую.

По ходу внесения в модуль ТЕХНОЛОГ каждой новой Отгрузочной ведомости, библиотека изделий пополняется. Фактически Библиотека будет содержать параметры всех изделий, когда-либо спроектированных инженерами.

Также в модуль ТЕХНОЛОГ напрямую или с помощью импорта таблицы Excel вносится следующие данные:

1. Плановая трудоемкость изделия
2. Норма расхода материала (указать допуск на отходы)
3. Норма расходных материалов на изготовление
4. Диаметр и число отверстий
5. Площадь зачистки и окраски
6. Сортамент и количество метизов
7. Длина сварного шва
8. Вид сварки

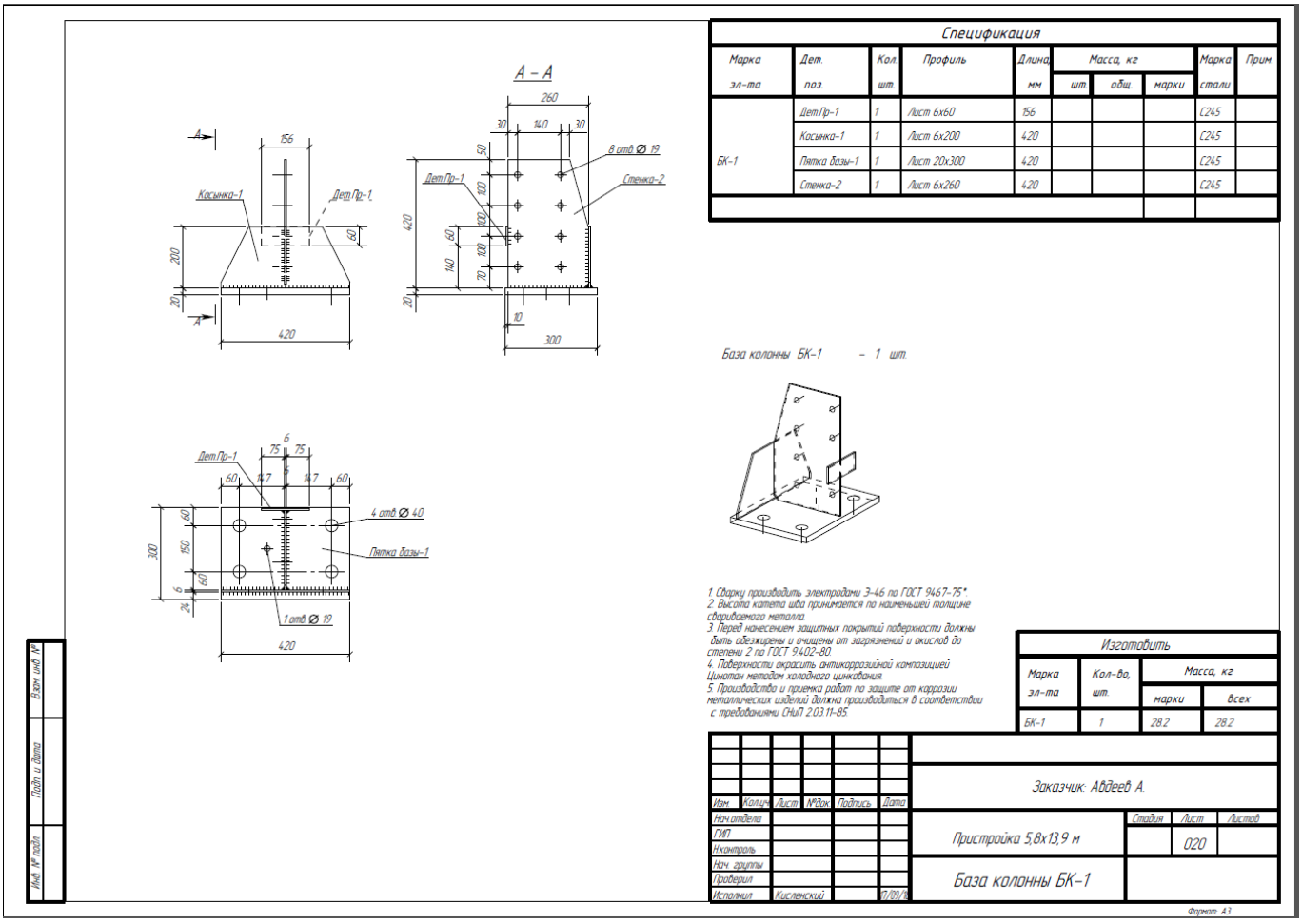
Для дальнейшей работы системы из приложения ТЕХНОЛОГ экспортируется файл *«Рабочие заказы/задания по клиенту (объекту) Х»* в приложение ПЛАНИРОВЩИК (см. модуль 3-3а).

Указанный файл экспортируется напрямую, но имеется и запасной вариант выгрузки через формат таблицы Excel.

Контроль Директора:

Отчет по выполнению задач инженерами по экспорту данных в ТЕХНОЛОГ.

Приложение В



Чертеж детали

Приложение Г

Таблица 1 — Преимущества и недостатки одностраничных и многостраничных веб-приложений

	Преимущества	**Недостатки**
SPA	Универсальность функционирования как на ПК, так и на мобильных устройствах.	Наличие JavaScript в активном режиме в браузерах пользователей.
	Быстродействие.	Падение производительности из-за утечки памяти в JS.
	Быстрота и эффективность разработки веб-приложений: наличие готовых библиотек и фреймворков.	Сложная настройка процессов автоматизированного построения и развертывания.
	Полнофункциональный пользовательский интерфейс.	Сложность SEO-оптимизации.
	Упрощение процедуры загрузки контента.	Необходимость обеспечить высокую защиту данных.
MPA	Понятный интерфейс и привычная навигация.	Универсальность функционирования отсутствует или требует адаптации к мобильным устройствам.
	Значительное упрощение SEO.	Малофункциональный пользовательский интерфейс.

	Неограниченная масштабируемость страниц.	Увеличение сроков и издержки разработки.
	Больший объем и разнообразие контента.	Низкое быстродействие и скорость загрузки.

Приложение Д

Таблица 2 — Сравнительная характеристика распространенных фреймворков

Фреймворки/Требования	Vue.js	React	Angular
Высокая скорость разработки	+	+-	-
Низкий уровень сложности освоения	+	+-	-
Быстрая скорость рендеринга и обработки DOM	+	+	+-
Архитектурный шаблон MVVM	+	-	+
Небольшой Размер библиотеки	+	+-	-
Понятная документация и наличие актуальных примеров использования	+-	+-	+
Поддержка сообщества	+-	+	+
Наличие богатой и развитой экосистемы	+	-	+
Имеющийся опыт использования	+	+-	-

Приложение Е

Листинги

```
class BaseError extends Error {
  constructor(store, message) {
    super(message)
    this.store = store
    this.name = this.constructor.name
    this.errors = this.store ? this.store.state.errors : []

    let id = _.last(this.errors) && _.last(this.errors).id + 1 || 1

    this.errors.push({
      id,
      message,
      type: this.name,
      isNonServed: true
    })
  }
}
```

Листинг 1 — Класс BaseError

```
<?php

return [
    'routes' => [
        'technologist/index' => 'technologist/index',
        'technologist/install' => 'technologist/install',
        'technologist/project/index' => 'technologist/indexProject',
        'technologist/project/create' => 'technologist/createProject',
        'technologist/project/get' => 'technologist/getProject',
        'technologist/project/save' => 'technologist/saveProject',
        'technologist/project/remove' => 'technologist/removeProject',
        'technologist/economist/load-economist' => 'technologist/loadEconomist',
        'technologist/economist/compute-materials' => 'technologist/computeMaterials',
        'technologist/economist/compute-labor-costs' => 'technologist/computeLaborCosts',
        'technologist/fields' => 'technologist/getFields',
    ],
    'project_handle_url' => '/technologist/project/index',
```

```
    'bitrix_base_url' => 'https://b24-52fjkz.bitrix24.ru',
    'app_title' => 'ТЕХНОЛОГ'
];
```

Листинг 2 — Файл конфигурации

```
function indexAction()
{
     $auth_id = Request::get('AUTH_ID');
     Session::set('auth_id', $auth_id);

     $this->renderPartial();
}
```

Листинг 3 — Функция indexAction

```
function installAction()
{
     $res = RestApi::makeRequest('placement.bind', [
     'auth' => Session::get('auth_id'),
     'PLACEMENT' => 'SONET_GROUP_DETAIL_TAB',
     'HANDLER'=>Request::getServer('request_scheme').'://'.
     Request::getServer('server_name') .
     Config::get('project_handle_url'),
     'TITLE' => Config::get('app_title')
     ]);

     Logger::logVarDump($res);
     Session::unset('auth_id');
     $this->sendJSON($res);
}
```

Листинг 4 — Функция installAction

```
function sendJSON($json)
{
    $this->response->sendJSON($json);
}
```

Листинг 5 — Функция sendJSON

Приложение Ж

Взаимодействия пользователя с приложением

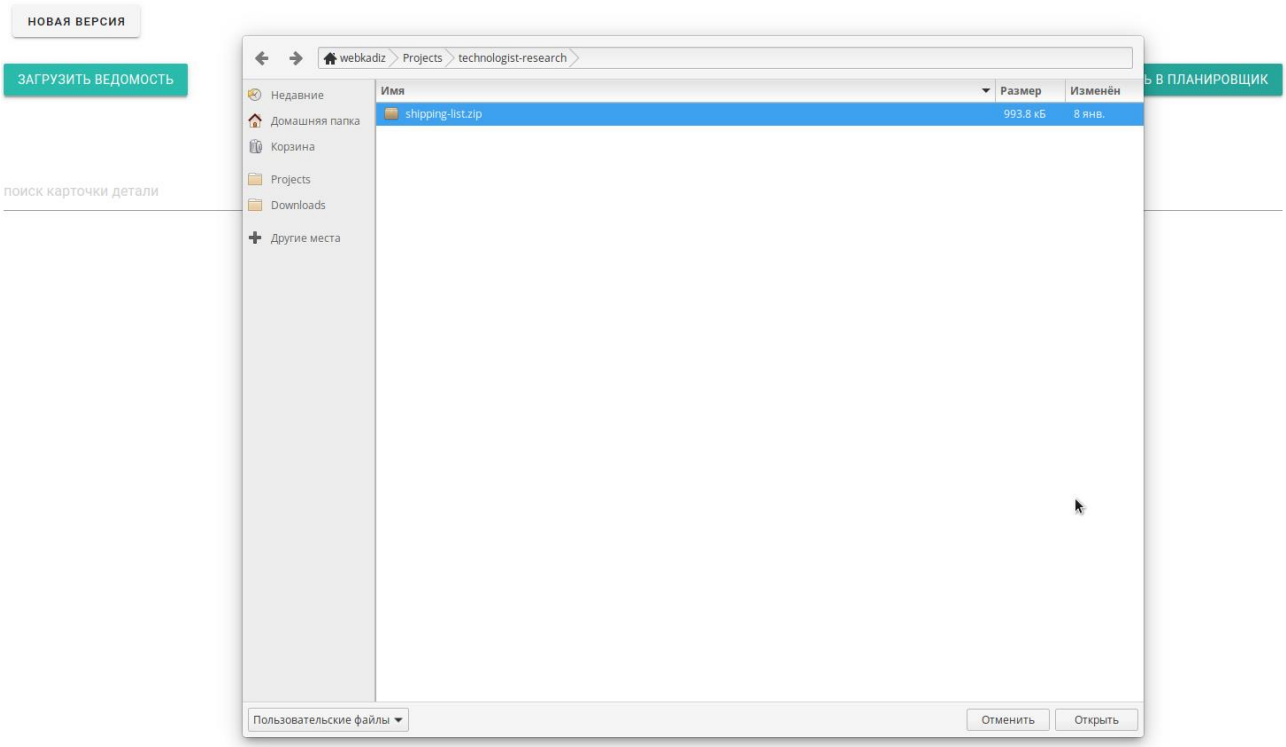


Рисунок 1 — Загрузка архива в приложение

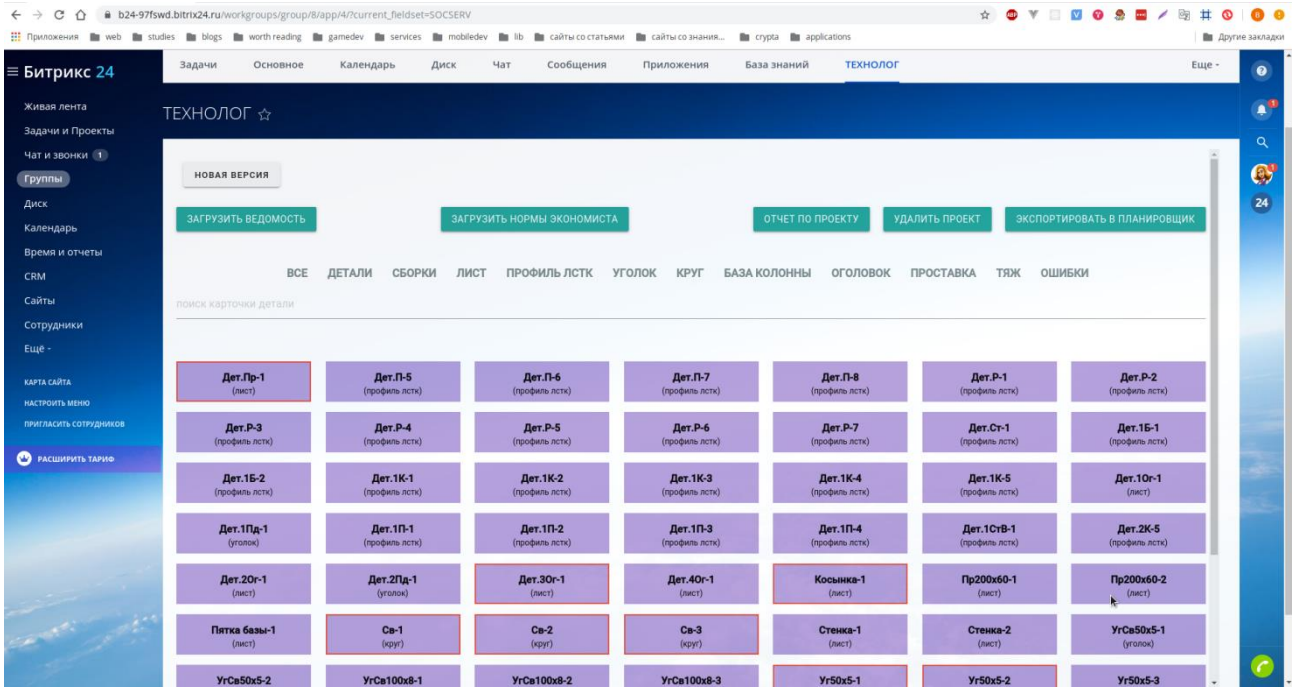


Рисунок 2 — Библиотека деталей

Рисунок 3 — Карточка детали. Нижняя часть

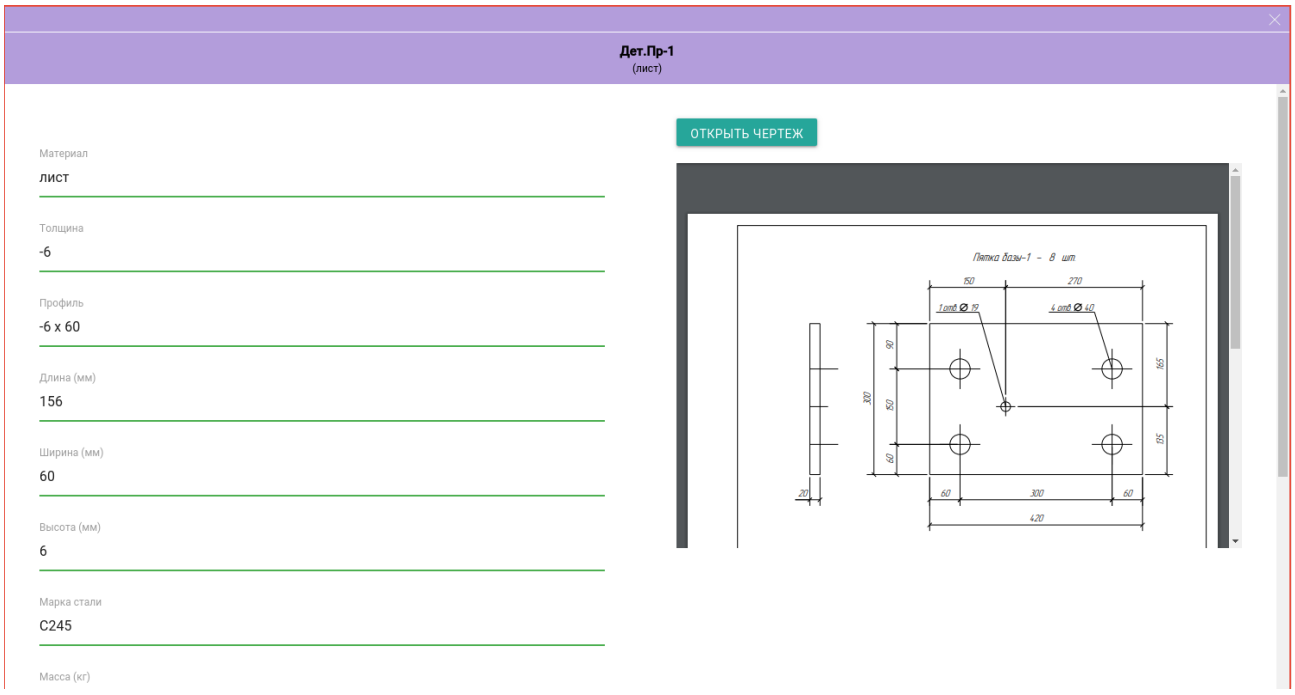


Рисунок 4 — Карточка детали. Верхняя часть

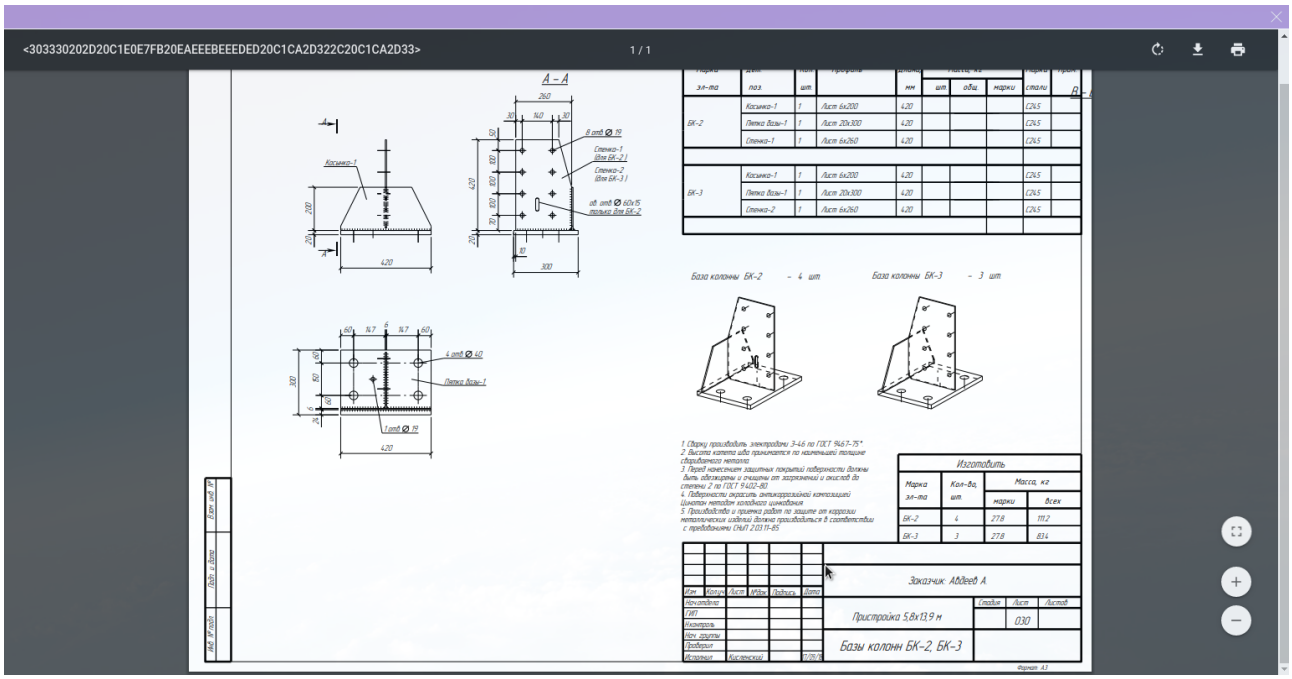


Рисунок 5 — Просмотр чертежа

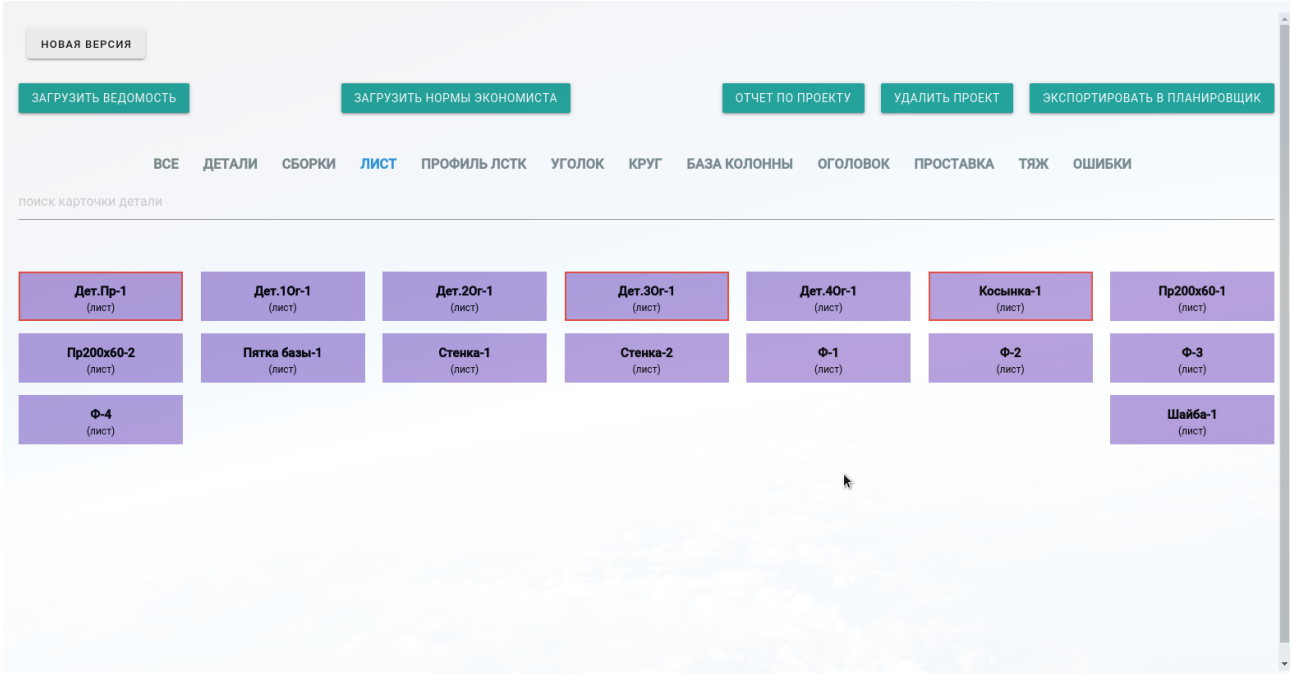


Рисунок 6 — Поиск детали по категории

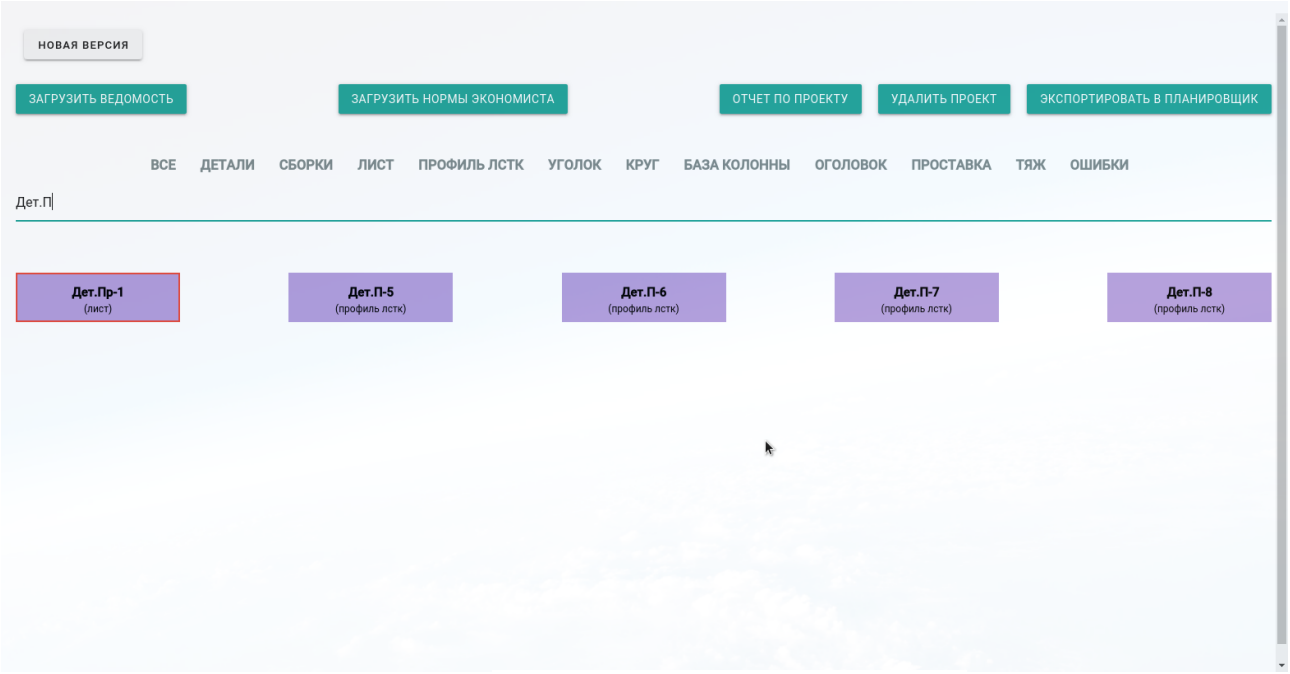


Рисунок 7 — Поиск летали по названию

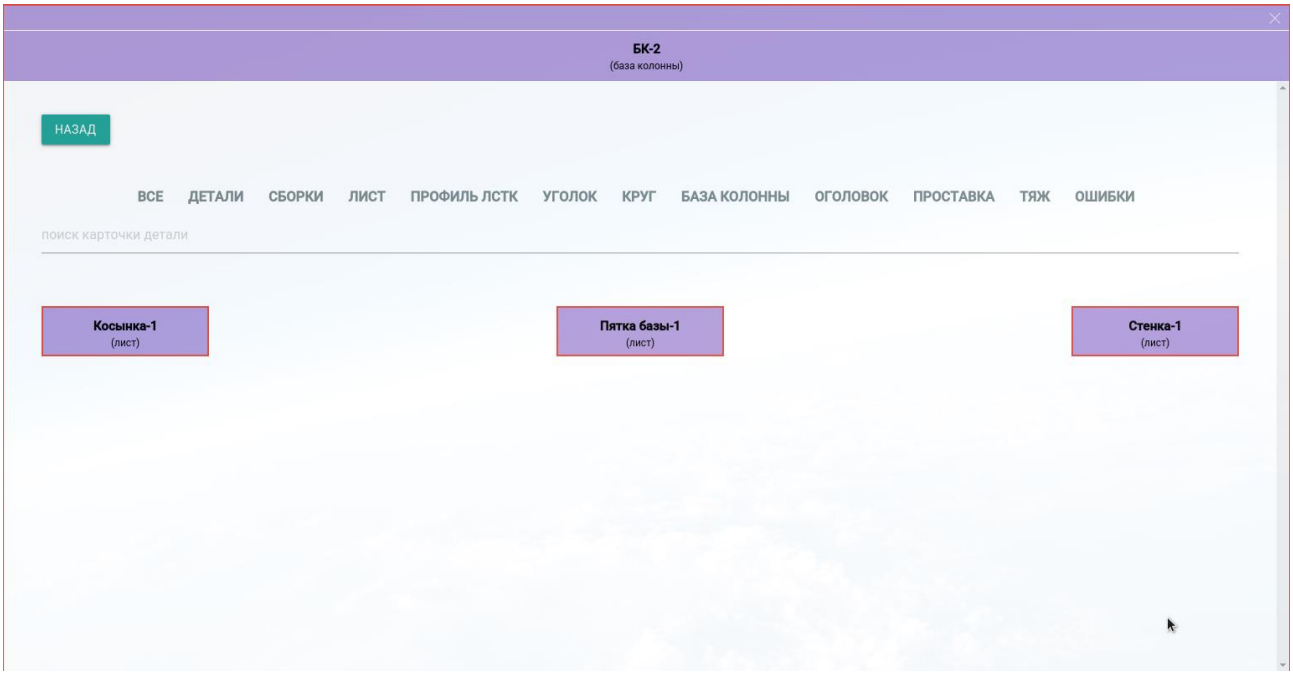
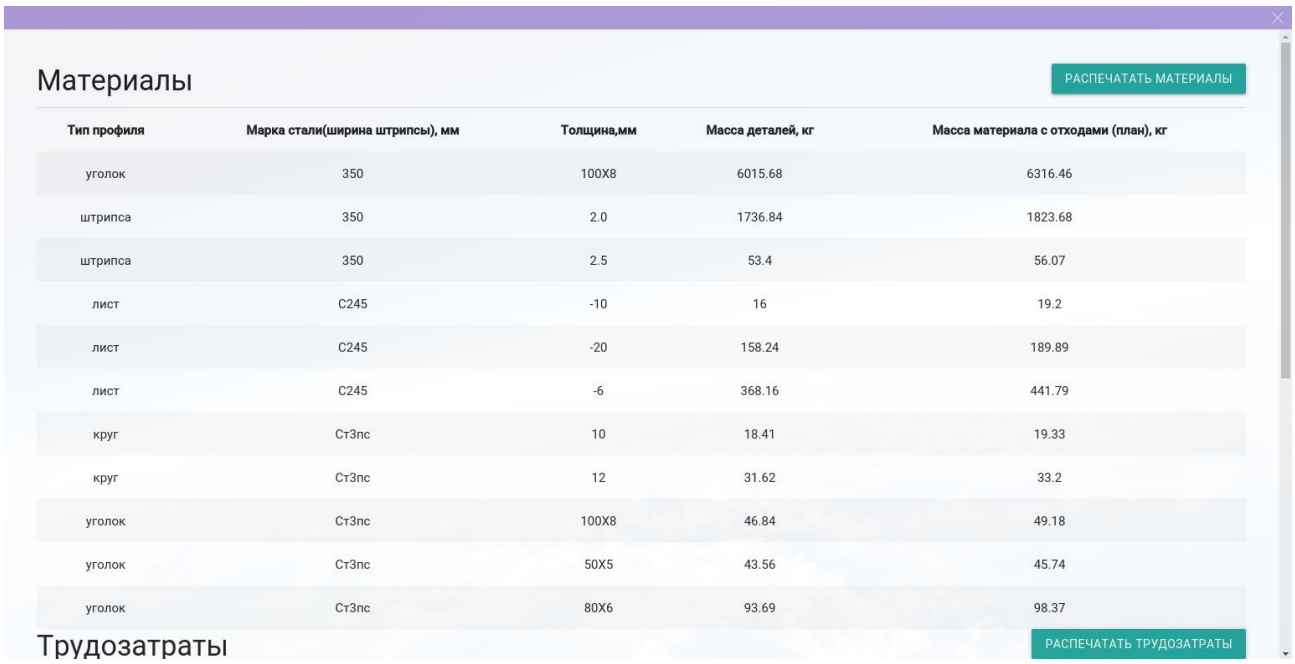


Рисунок 8 — Рекурсивный рендеринг



Материалы

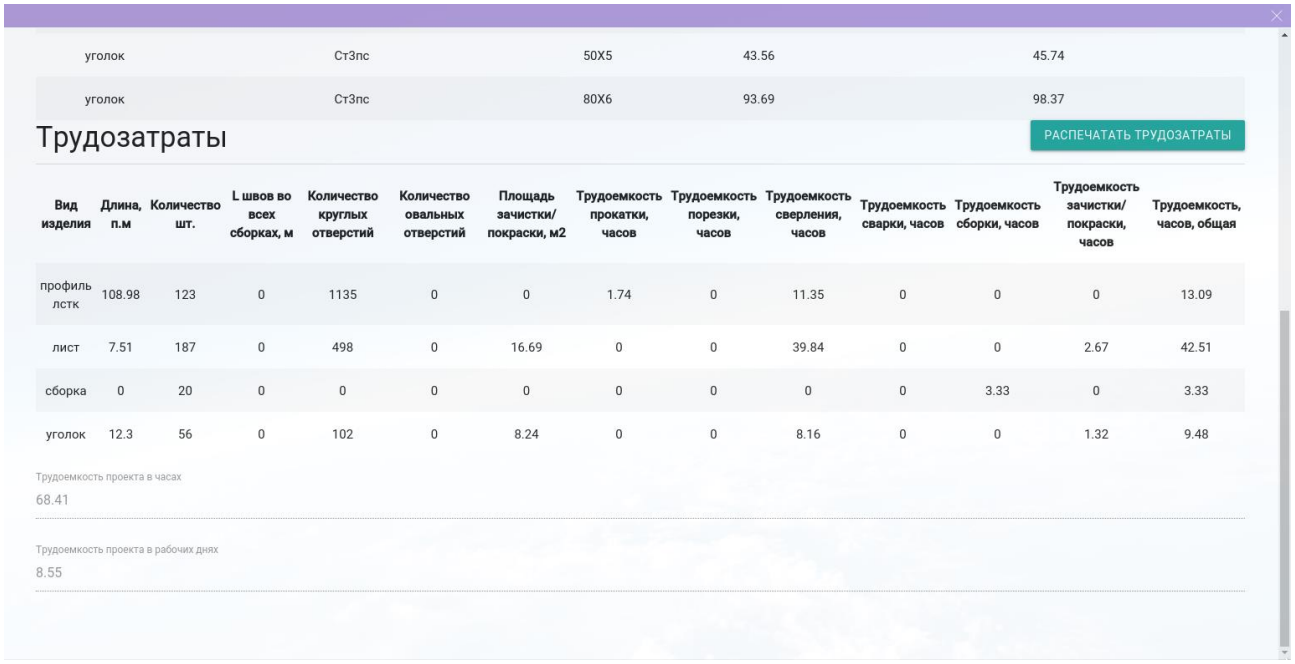
РАСПЕЧАТАТЬ МАТЕРИАЛЫ

Тип профиля	Марка стали(ширина штрипсы), мм	Толщина,мм	Масса деталей, кг	Масса материала с отходами (план), кг
уголок	350	100X8	6015.68	6316.46
штрипса	350	2.0	1736.84	1823.68
штрипса	350	2.5	53.4	56.07
лист	С245	-10	16	19.2
лист	С245	-20	158.24	189.89
лист	С245	-6	368.16	441.79
круг	Ст3пс	10	18.41	19.33
круг	Ст3пс	12	31.62	33.2
уголок	Ст3пс	100X8	46.84	49.18
уголок	Ст3пс	50X5	43.56	45.74
уголок	Ст3пс	80X6	93.69	98.37

Трудозатраты

РАСПЕЧАТАТЬ ТРУДОЗАТРАТЫ

Рисунок 9 — Расчет материалов



уголок	Ст3пс	50X5	43.56	45.74
уголок	Ст3пс	80X6	93.69	98.37

Трудозатраты

РАСПЕЧАТАТЬ ТРУДОЗАТРАТЫ

Вид изделия	Длина, п.м	Количество шт.	L швов во всех сборках, м	Количество круглых отверстий	Количество овальных отверстий	Площадь зачистки/ покраски, м2	Трудоемкость прокатки, часов	Трудоемкость порезки, часов	Трудоемкость сверления, часов	Трудоемкость сварки, часов	Трудоемкость сборки, часов	Трудоемкость зачистки/ покраски, часов	Трудоемкость, часов, общая
профиль лстк	108.98	123	0	1135	0	0	1.74	0	11.35	0	0	0	13.09
лист	7.51	187	0	498	0	16.69	0	0	39.84	0	0	2.67	42.51
сборка	0	20	0	0	0	0	0	0	0	0	3.33	0	3.33
уголок	12.3	56	0	102	0	8.24	0	0	8.16	0	0	1.32	9.48

Трудоемкость проекта в часах

68.41

Трудоемкость проекта в рабочих днях

8.55

Рисунок 10 — Расчет трудозатрат

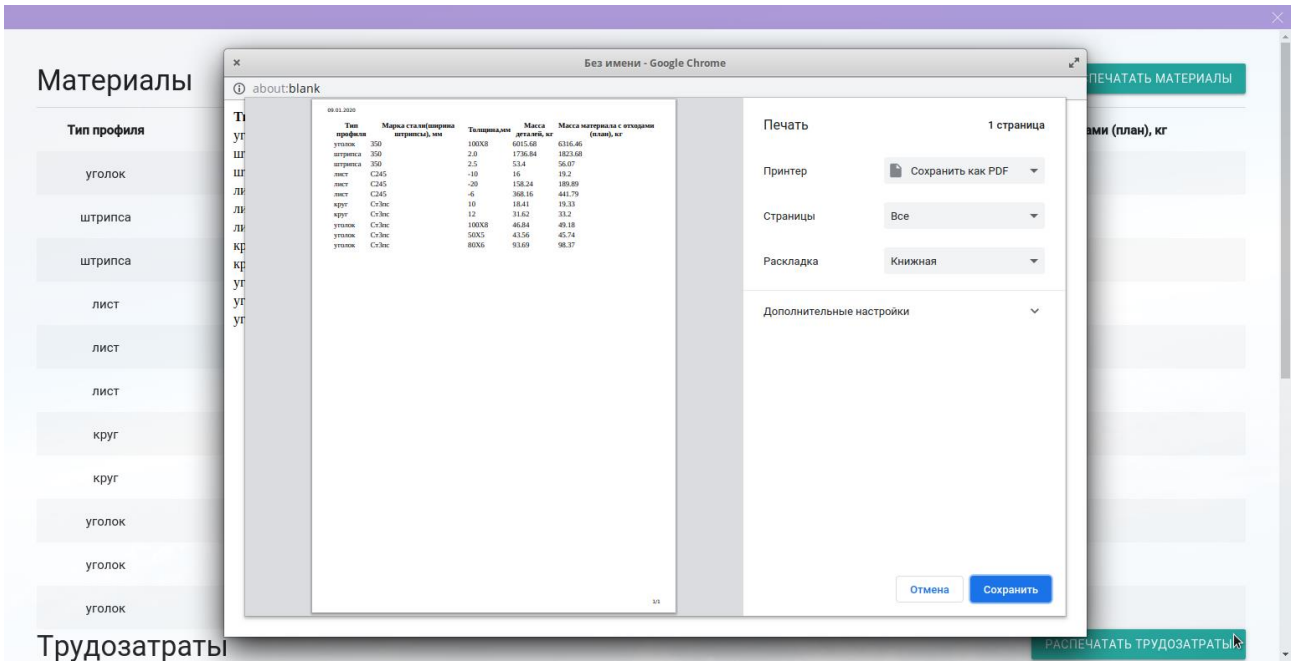


Рисунок 11 — Печать материалов