МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Тверской государственный технический университет»**

(ТвГТУ)

Кафедра «Информационные системы»

*К защите допустить*

Заведующий кафедрой ИС

Б.В. Палюх

подпись И.О.Ф. «\_\_\_»\_\_\_\_\_\_\_ 20 \_ г.

**Д И П Л О М Н Ы Й П Р О Е К Т**

на тему «**Автоматизация деятельности по учету и согласованию выдачи оборудования сотрудникам для ООО "АКСЕНЧЕР", г. Тверь**»

Направление: 09.03.02 Информационные системы и технологии

Профиль: Информационные системы в административном управлении

Студент (ка): \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_

фамилия, имя, отчество подпись

Форма обучения: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Группа: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_

ученая степень, звание, фамилия, инициалы подпись

Консультанты по разделам:

Обоснование экономической эффективности

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

ученая степень, звание, фамилия, инициалы подпись

Нормоконтролер: Федотова Елена Петровна\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_

фамилия, имя, отчество подпись

Тверь 2020

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Тверской государственный технический университет»**

(ТвГТУ)

Кафедра «Информационные системы»

*Утверждаю*

Заведующий кафедрой ИС

Б.В. Палюх

подпись И.О.Ф.

«\_\_» \_\_\_\_\_\_\_ 20 \_ г.

**З А Д А Н И Е**

**на дипломный проект бакалавра**

Студент (ка): \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

фамилия, имя, отчество

Направление: 09.03.02 Информационные системы и технологии

Профиль: Информационные системы в административном управлении

Тема: «**Автоматизация деятельности по учету и согласованию выдачи оборудования сотрудникам для ООО "АКСЕНЧЕР", г. Тверь**»

(утверждена приказом ректора от «\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_г. № \_\_\_\_\_\_\_)

Срок сдачи студентом законченной работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Исходные данные к работе \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Перечень графического (иллюстративного) материала:

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

Консультанты по проекту (работе) с указанием относящихся к ним разделов проекта:

Обоснование экономической эффективности

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

ученая степень, звание, фамилия, инициалы подпись

Дата выдачи задания «\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_20\_г.

Руководитель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

.И.О., ученая степень, ученое звание, должность подпись

Задание получено\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ф.И.О. студента подпись

Тверь 2020

Изм.

Лист

№ докум.

Подпись

Дата

Лист

4

Содержание

Содержание

[Пояснительная записка 2](#_Toc37693934)

[Постановка задачи 3](#_Toc37693935)

[Введение 7](#_Toc37693936)

[1. Разработка модели предметной области автоматизации 9](#_Toc37693937)

[1.1. Анализ организационной структуры ООО «АКСЕНЧЕР» 9](#_Toc37693938)

[1.2. Сравнительный анализ программных средств уже существующих на рынке. 12](#_Toc37693939)

[Заключение 25](#_Toc37693940)

[Источники 27](#_Toc37693941)

Изм.

Лист

№ докум.

Подпись

Дата

Лист

6

Введение

# Введение

На данный момент, информационные технологии уже плотно закрепились во всех сферах деятельности человека. Нельзя представить ни одного бизнес-процесса, без использования информационной системы. Пару десятилетий назад, в большинстве случаев , речь шла о внедрении. Сейчас же, с развитием технологий и рынка, эта отрасль больше похожа на гонку. С каждым годом появляется все больше продуктов, которые могут удовлетворить любые потребности заказчика. Единственное что не изменилось – это цели, преследуемые в создании ИС:

* уменьшение затрат
* уменьшение времени на обработку данных
* увеличить производительность труда

Рассматриваемые дипломный проект написан на базе ООО «АКСЕНЧЕР» для собственного использования.

ООО «АКСЕНЧЕР» работает по различным направления, и делится на множество подразделений.

И все эти подразделения объединяет одно - в каждом насчитывается огромное количество сотрудников.

В свою очередь, для каждого сотрудника, оборудование играет важную роль в работе.

Данная ИС направлена на автоматизацию и максимальное ускорение процесса выдачи оборудования.

Изм.

Лист

№ докум.

Подпись

Дата

Лист

8

Пункт1.1

# 1. Разработка модели предметной области автоматизации

## 1.1. Анализ организационной структуры ООО «АКСЕНЧЕР»

Accenture — консалтинговая компания, оказывающая услуги организациям по консультированию в сферах [стратегического планирования](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D0%B0%D1%82%D0%B5%D0%B3%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5_%D0%BF%D0%BB%D0%B0%D0%BD%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), оптимизации и организации [аутсорсинга бизнес-процессов](https://ru.wikipedia.org/wiki/%D0%90%D1%83%D1%82%D1%81%D0%BE%D1%80%D1%81%D0%B8%D0%BD%D0%B3_%D0%B1%D0%B8%D0%B7%D0%BD%D0%B5%D1%81-%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D0%B2" \o "Аутсорсинг бизнес-процессов), [управления взаимоотношениями с клиентами](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B2%D0%B7%D0%B0%D0%B8%D0%BC%D0%BE%D0%BE%D1%82%D0%BD%D0%BE%D1%88%D0%B5%D0%BD%D0%B8%D1%8F%D0%BC%D0%B8_%D1%81_%D0%BA%D0%BB%D0%B8%D0%B5%D0%BD%D1%82%D0%B0%D0%BC%D0%B8), управления логистическими процессами, [управления персоналом](https://ru.wikipedia.org/wiki/%D0%A3%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D0%BF%D0%B5%D1%80%D1%81%D0%BE%D0%BD%D0%B0%D0%BB%D0%BE%D0%BC), внедрения [информационных технологий](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D1%8B%D0%B5_%D1%82%D0%B5%D1%85%D0%BD%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D0%B8). Компания, впервые зарегистрированная на [Бермудах](https://ru.wikipedia.org/wiki/%D0%91%D0%B5%D1%80%D0%BC%D1%83%D0%B4%D1%81%D0%BA%D0%B8%D0%B5_%D0%BE%D1%81%D1%82%D1%80%D0%BE%D0%B2%D0%B0), [1 сентября](https://ru.wikipedia.org/wiki/1_%D1%81%D0%B5%D0%BD%D1%82%D1%8F%D0%B1%D1%80%D1%8F) [2001 года](https://ru.wikipedia.org/wiki/2001_%D0%B3%D0%BE%D0%B4) получила статус акционерного общества в [Ирландии](https://ru.wikipedia.org/wiki/%D0%98%D1%80%D0%BB%D0%B0%D0%BD%D0%B4%D0%B8%D1%8F). Однако головной офис расположен в [Нью-Йорке](https://ru.wikipedia.org/wiki/%D0%9D%D1%8C%D1%8E-%D0%99%D0%BE%D1%80%D0%BA). Со штатом примерно около 450 тыс. сотрудников, компания обслуживает клиентов более чем в 120 странах мира.

Как и большинство консалтинговых компаний, Accenture имеет матричную структуру: структура управления организуется как по операционным группам (подразделениям по отраслям клиентов), так и по видам услуг, предоставляемых компанией.

*Операционные группы*

Для упрощения финансовой отчетности Accenture группирует свои услуги по 5 направлениям, в зависимости от класса отрасли заказчиков:

* коммуникации и высокие технологии: аэрокосмическая и [оборонная промышленность](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D0%BE%D1%80%D0%BE%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B8%D0%B7%D0%B2%D0%BE%D0%B4%D1%81%D1%82%D0%B2%D0%BE), [телекоммуникации](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BB%D0%B5%D0%BA%D0%BE%D0%BC%D0%BC%D1%83%D0%BD%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D0%B8), электроника и высокие технологии, [СМИ](https://ru.wikipedia.org/wiki/%D0%A1%D0%9C%D0%98) и индустрия развлечений;
* финансовые услуги: [банки](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%BD%D0%BA), финансовые услуги;
* продукты: [автомобильная промышленность](https://ru.wikipedia.org/wiki/%D0%90%D0%B2%D1%82%D0%BE%D0%BC%D0%BE%D0%B1%D0%B8%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D0%BC%D1%8B%D1%88%D0%BB%D0%B5%D0%BD%D0%BD%D0%BE%D1%81%D1%82%D1%8C), строительные материалы, потребительские товары и услуги, розничная торговля, [химическая промышленность](https://ru.wikipedia.org/wiki/%D0%A5%D0%B8%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D0%BC%D1%8B%D1%88%D0%BB%D0%B5%D0%BD%D0%BD%D0%BE%D1%81%D1%82%D1%8C), грузы и логистика, инфраструктура и транспорт, промышленное оборудование;
* ресурсы: ресурсы, лесные товары, полезные ископаемые, металлы, коммунальные услуги;
* работа с госсектором: таможенное дело, обороноспособность, здравоохранение и социальные услуги.

*Консалтинговые практики*

* Accenture Стратегия, основана в декабре 2013 года для оказания услуг по формулировке стратегии бизнеса, технологий и операций.
* Accenture Консалтинг, основан в июне 2015 года для предоставления технологического, бизнес и управленческого консалтинга.
* Accenture Digital, основан в декабре 2013 для предоставления сервисов в сфере диджитал-маркетинга, аналитики и мобильности.
* Accenture Technology сфокусирован на технологических решениях, их применении, исследованиях и разработке новых технологических трендов.
* Accenture Operations включает в себя аутсорсинг бизнес процессов, сервисы IT, облачные сервисы, безопасность и инфраструктуру.

Изм.

Лист

№ докум.

Подпись

Дата

Лист

11

Пункт1.2

## 1.2. Сравнительный анализ программных средств уже существующих на рынке.

Задача, решаемая в данном дипломном проекте, является узконаправленной. Готовые же решения являются более абстрактными и имеют избыточный функционал.

Разрабатываемую ИС можно разделить на 2 модуля:

1. ИС, имеющая удобный web-интерфейс для реализации хранения, и редактирования данных.
2. Система планирования задач и согласований, с возможность кастомизации бизнес-процесса согласования и жизненного цикла задачи.

Проанализировав задачи, которые должен решать данный дипломный проект, он попадает под определение «**Программное обеспечение для управления проектами**».

**Программное обеспечение для управления проектами** — комплексное программное обеспечение, включающее в себя приложения для планирования задач, составления [расписания](https://ru.wikipedia.org/wiki/%D0%A8%D1%82%D0%B0%D1%82%D0%BD%D0%BE%D0%B5_%D1%80%D0%B0%D1%81%D0%BF%D0%B8%D1%81%D0%B0%D0%BD%D0%B8%D0%B5), контроля цены и управления бюджетом, [распределения ресурсов](https://ru.wikipedia.org/w/index.php?title=%D0%A0%D0%B0%D1%81%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D1%80%D0%B5%D1%81%D1%83%D1%80%D1%81%D0%BE%D0%B2&action=edit&redlink=1), [совместной работы](https://ru.wikipedia.org/wiki/Groupware), [общения](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%89%D0%B5%D0%BD%D0%B8%D0%B5), быстрого управления, [документирования](https://ru.wikipedia.org/wiki/%D0%94%D0%BE%D0%BA%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%B0%D1%86%D0%B8%D1%8F) и администрирования системы, которая используется совместно для управления крупными [проектами](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B5%D0%BA%D1%82).

Задачи программного обеспечения для управления проектами:

*1.Планирование*

Одной из наиболее распространенных возможностей является возможность планирования событий и управления задачами. Требования могут различаться в зависимости от того, как используется инструмент. Наиболее распространенными являются:

* планирование различных событий, зависящих друг от друга;
* идентификация крупных составных частей проекта ([вехи проекта](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D1%85%D0%B0_(%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B0%D0%BC%D0%B8))) и их декомпозиция, посредством которой создается [структура декомпозиции работ](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%B4%D0%B5%D0%BA%D0%BE%D0%BC%D0%BF%D0%BE%D0%B7%D0%B8%D1%86%D0%B8%D0%B8_%D1%80%D0%B0%D0%B1%D0%BE%D1%82), также называемая иерархической структурой работ ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *work break-down structure — WBS*);
* планирование расписания работы сотрудников и назначение ресурсов на конкретные задачи;
* расчет времени, необходимого на решение каждой из [задач](https://ru.wikipedia.org/w/index.php?title=%D0%97%D0%B0%D0%B4%D0%B0%D1%87%D0%B0_(%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B0%D0%BC%D0%B8)&action=edit&redlink=1);
* сортировка задач в зависимости от сроков их завершения;
* презентация графика работ по проекту в виде [диаграммы Гантта](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0_%D0%93%D0%B0%D0%BD%D1%82%D1%82%D0%B0);
* управление несколькими проектами одновременно.

*2.Управление данными и предоставление информации*

Программное обеспечение для управления проектами предоставляет большое количество требуемой информации, такой как:

* список задач для сотрудников и информацию распределения ресурсов;
* обзор информации о сроках выполнения задач;
* ранние предупреждения о возможных [рисках](https://ru.wikipedia.org/wiki/%D0%A0%D0%B8%D1%81%D0%BA), связанных с проектом;
* информации о рабочей нагрузке;
* информация о ходе проекта, показатели и их прогнозирование.

*3.Управление коммуникациями команды проекта*

* Обсуждение и согласование рабочих вопросов проекта
* Фиксация проблем проекта и запросов на изменения, их обработка
* Ведение рисков проекта и проактивное управление ими
* Предоставление доступа к информации о ходе проекта в виде живой ленты событий

Типы программного обеспечения для управления проектами:

*1. Desktop (Настольные)*

Программное обеспечение находится на [десктопе](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%B0%D1%84%D0%BE%D1%80%D0%B0_%D1%80%D0%B0%D0%B1%D0%BE%D1%87%D0%B5%D0%B3%D0%BE_%D1%81%D1%82%D0%BE%D0%BB%D0%B0" \o "Метафора рабочего стола) каждого пользователя. Это предоставляет наиболее гибкий интерфейс. Такие приложения обычно позволяют сохранять информацию в файл, который в дальнейшем может быть выложен в общий доступ для других пользователей или же данные хранятся в центральной базе данных.

*2.Web-based (Веб-интерфейс)*

Программное обеспечение является [веб-приложением](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5" \o "Веб-приложение), доступ к которому осуществляется с помощью [браузера](https://ru.wikipedia.org/wiki/%D0%91%D1%80%D0%B0%D1%83%D0%B7%D0%B5%D1%80).

Плюсы и минусы:

* Доступ может быть осуществлен с любого компьютера, не требуется установка дополнительных приложений
* Простой контроль доступа
* Многопользовательский доступ
* Только одна программа, которая установлена на центральном сервере
* Скорость работы ниже, чем у обычных приложений
* Проблемы с доступом к серверу или его выход из строя ведут к полной недоступности информации

В случае реализации своего диплома, меня интересуют *Web-based.*

Рассмотрим основных представителей:

* Jira
* Easy Projects .NET
* PayDox

**Jira**

Jira Software@2x-blue.png

Рисунок 1. Логотип Jira

Jira — коммерческая [система отслеживания ошибок](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%BE%D1%82%D1%81%D0%BB%D0%B5%D0%B6%D0%B8%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F_%D0%BE%D1%88%D0%B8%D0%B1%D0%BE%D0%BA), предназначена для организации взаимодействия с пользователями, хотя в некоторых случаях используется и для [управления проектами](https://ru.wikipedia.org/wiki/%D0%A3%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B0%D0%BC%D0%B8). Разработана компанией [Atlassian](https://ru.wikipedia.org/wiki/Atlassian" \o "Atlassian), является одним из двух её основных продуктов (наряду с вики-системой [Confluence](https://ru.wikipedia.org/wiki/Confluence)). Имеет [веб-интерфейс](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81" \o "Веб-интерфейс).

Название системы получено путём усечения слова «Gojira» — японского имени монстра [Годзилла](https://ru.wikipedia.org/wiki/%D0%93%D0%BE%D0%B4%D0%B7%D0%B8%D0%BB%D0%BB%D0%B0), что, в свою очередь, является отсылкой к названию конкурирующего продукта — [Bugzilla](https://ru.wikipedia.org/wiki/Bugzilla" \o "Bugzilla); создавалась в качестве замены Bugzilla и во многом повторяет её архитектуру. Система позволяет работать с несколькими проектами. Для каждого из проектов создаёт и ведёт схемы безопасности и схемы оповещения.

До версии 3.13.5 (включительно) различались редакции Enterprise, Professional и Standard, после — осталась только редакция Enterprise (для крупных организаций).

Система основана на [Java EE](https://ru.wikipedia.org/wiki/Java_EE" \o "Java EE) и работает на нескольких популярных системах управления базами данных и операционных системах.

Основной элемент учёта в системе — задача ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *ticket* или *issue*). Задача содержит название проекта, тему, тип, приоритет, компоненты и содержание. Задача может быть расширена дополнительными полями (также и новые пользовательские поля могут быть определены), приложениями (например — фотографиями, скриншотами) или комментариями. Задача может редактироваться или просто изменять статус, например, из «открыт» в «закрыт». Какие переходы между состояниями возможны, определяется через настраиваемый [поток операций](https://ru.wikipedia.org/w/index.php?title=%D0%9F%D0%BE%D1%82%D0%BE%D0%BA_%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%B9&action=edit&redlink=1). Любые изменения в задаче протоколируются в журнал.

Jira имеет большое количество возможностей конфигурации: для каждого приложения может быть определён отдельный тип задачи с собственным workflow, набором статусов, одним или несколькими видами представления ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *screens*). Кроме того, с помощью так называемых «схем» можно определить для каждого индивидуального Jira-проекта собственные права доступа, поведение и видимость полей и многое другое.

Благодаря универсальному подходу можно приспособить Jira для многих непрофильных задач, например, [управления требованиями](https://ru.wikipedia.org/wiki/%D0%A3%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D1%82%D1%80%D0%B5%D0%B1%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F%D0%BC%D0%B8_%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%BC%D1%83_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8E), [управления рисками](https://ru.wikipedia.org/wiki/%D0%A3%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D1%80%D0%B8%D1%81%D0%BA%D0%B0%D0%BC%D0%B8), вплоть до реализации небольшой [системы бронирования](https://ru.wikipedia.org/w/index.php?title=%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%B1%D1%80%D0%BE%D0%BD%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F&action=edit&redlink=1), автоматизации процесса [рекрутинга](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BA%D1%80%D1%83%D1%82%D0%B8%D0%BD%D0%B3" \o "Рекрутинг).

Для интеграции с внешними системами поддерживает интерфейсы [SOAP](https://ru.wikipedia.org/wiki/SOAP), [XML-RPC](https://ru.wikipedia.org/wiki/XML-RPC) и [REST](https://ru.wikipedia.org/wiki/REST). Поставляется со средствами интеграции с такими [системами управления версиями](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%8F%D0%BC%D0%B8), как [Subversion](https://ru.wikipedia.org/wiki/Subversion" \o "Subversion), [CVS](https://ru.wikipedia.org/wiki/CVS), [Git](https://ru.wikipedia.org/wiki/Git" \o "Git), [Clearcase](https://ru.wikipedia.org/wiki/Clearcase" \o "Clearcase), [Team Foundation Server](https://ru.wikipedia.org/wiki/Team_Foundation_Server" \o "Team Foundation Server), [Mercurial](https://ru.wikipedia.org/wiki/Mercurial" \o "Mercurial) и [Perforce](https://ru.wikipedia.org/wiki/Perforce" \o "Perforce). Существуют дополнения, позволяющие встроить Jira в [интегрированные среды разработки](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D1%80%D0%B5%D0%B4%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8), в том числе [Eclipse](https://ru.wikipedia.org/wiki/Eclipse_(%D1%81%D1%80%D0%B5%D0%B4%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8)" \o "Eclipse (среда разработки)) и [IntelliJ IDEA](https://ru.wikipedia.org/wiki/IntelliJ_IDEA" \o "IntelliJ IDEA). Переведена на многие языки, включая русский, английский, японский, немецкий, французский, испанский.

Для сторонних разработчиков предоставляются средства разработки расширений системы — [плагинов](https://ru.wikipedia.org/wiki/%D0%9F%D0%BB%D0%B0%D0%B3%D0%B8%D0%BD" \o "Плагин). Разработчики расширений могут выкладывать плагины для продажи на специальный раздел сайта Atlassian.

Является [коммерческим](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BC%D0%B5%D1%80%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) продуктом, который может быть лицензирован для работы на локальном сервере или доступен в качестве удалённого приложения. Ценообразование зависит от максимального числа пользователей, при этом около $50 за пользователя для локального и $7 в месяц за пользователя для удаленного доступа являются типичными ценами.

Для академических и коммерческих клиентов доступен полный исходный код под лицензией разработчика.

Для проектов с открытым исходным кодом Atlassian предоставляет специальную бесплатную лицензию при соблюдении следующих правил:

* проект использует лицензии, одобренные [Open Source Initiative](https://ru.wikipedia.org/wiki/Open_Source_Initiative" \o "Open Source Initiative);
* исходный код проекта доступен для скачивания;
* у проекта есть публично доступный веб-сайт;
* программное обеспечение от Atlassian доступно на веб-сайте проекта.

Easy Projects .NET



Рисунок 2. Логотип EasyProject

Easy Projects — это [веб-приложение](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5" \o "Веб-приложение) для [управления проектами](https://ru.wikipedia.org/wiki/%D0%A3%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B0%D0%BC%D0%B8) по разработке [программного обеспечения](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5), написанное на [.NET](https://ru.wikipedia.org/wiki/.NET_Framework) компанией [Logic Software](https://ru.wikipedia.org/w/index.php?title=Logic_Software&action=edit&redlink=1" \o "Logic Software (страница отсутствует)).

Возможности:

*1.Управление проектами*

Easy Projects позволяет создавать неограниченное количество проектов, содержащих различные настраиваемые поля. [Пакетная обработка](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D0%BA%D0%B5%D1%82%D0%BD%D0%B0%D1%8F_%D0%BE%D0%B1%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0) данных позволяет выполнять типичные операции для нескольких проектов одновременно. Пользователям доступны интерактивная [диаграмма Ганта](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0_%D0%93%D0%B0%D0%BD%D1%82%D0%B0), графики и отчёты. Отдельные проекты могут группироваться в [портфели проектов](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%80%D1%82%D1%84%D0%B5%D0%BB%D1%8C_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%BE%D0%B2).

*2.Управление задачами и отслеживание ошибок*

Easy Projects поддерживает неограниченное количество задач и подзадач, а также настройку статусов, категорий и приоритетов задач. Возможно создание задач по [электронной почте](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%BF%D0%BE%D1%87%D1%82%D0%B0). Не только разработчики, но и клиенты имеют возможность добавлять запросы и требования.

*3.Управление временем и ресурсами*

Программа позволяет вносить и отслеживать оплачиваемое и неоплачиваемое время, потраченное на проект. Поддерживаются личные и корпоративные расписания, а также существует возможность просмотра графика загруженности ресурсов. Возможна гибкая настройка прав доступа пользователей.

*4. Управление бюджетом*

Easy Projects позволяет создавать [инвойсы](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D0%B2%D0%BE%D0%B9%D1%81), следить за [бюджетом](https://ru.wikipedia.org/wiki/%D0%91%D1%8E%D0%B4%D0%B6%D0%B5%D1%82) проекта и создавать необходимые отчёты.

*5.Интеграция*

Программа поддерживает [экспорт данных](https://ru.wikipedia.org/w/index.php?title=%D0%AD%D0%BA%D1%81%D0%BF%D0%BE%D1%80%D1%82_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85&action=edit&redlink=1) в [MS Project](https://ru.wikipedia.org/wiki/MS_Project), [Excel](https://ru.wikipedia.org/wiki/Excel" \o "Excel), [PDF](https://ru.wikipedia.org/wiki/PDF) и [iCal](https://ru.wikipedia.org/wiki/ICal" \o "ICal), интеграцию с [Vyew](https://ru.wikipedia.org/w/index.php?title=Vyew&action=edit&redlink=1" \o "Vyew (страница отсутствует)), [QuickBooks](https://ru.wikipedia.org/w/index.php?title=QuickBooks&action=edit&redlink=1" \o "QuickBooks (страница отсутствует)), [SmarterTrack](https://ru.wikipedia.org/w/index.php?title=SmarterTrack&action=edit&redlink=1" \o "SmarterTrack (страница отсутствует)), [Dbxtra](https://ru.wikipedia.org/w/index.php?title=Dbxtra&action=edit&redlink=1" \o "Dbxtra (страница отсутствует)), а также синхронизацию с [Easy Time Tracking](https://ru.wikipedia.org/w/index.php?title=Easy_Time_Tracking&action=edit&redlink=1" \o "Easy Time Tracking (страница отсутствует)). Разработчики имеют доступ к [API](https://ru.wikipedia.org/wiki/API) и [SDK](https://ru.wikipedia.org/wiki/SDK) программы.

*6.Интерфейс*

Внешний вид системы может быть настроен путём добавления или удаления [виджетов](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%B4%D0%B6%D0%B5%D1%82" \o "Виджет) с информацией о проектах. Пользователи могут использовать [веб-конференции](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%84%D0%BE%D1%80%D1%83%D0%BC" \o "Веб-форум) для совместной работы. Поддерживаются [английский](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA), [немецкий](https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D0%BC%D0%B5%D1%86%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA), [французский](https://ru.wikipedia.org/wiki/%D0%A4%D1%80%D0%B0%D0%BD%D1%86%D1%83%D0%B7%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) и [русский языки](https://ru.wikipedia.org/wiki/%D0%A0%D1%83%D1%81%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) интерфейса.

Лицензирование:

1. Бесплатный аккаунт создаётся и конфигурируется на [серверах](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%B5%D1%80_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5)) компании и поддерживает до 100 [МБ](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D0%B3%D0%B0%D0%B1%D0%B0%D0%B9%D1%82) дискового пространства и одного пользователя.
2. Хостинговый аккаунт также создаётся на серверах компании и поддерживает 1 [ГБ](https://ru.wikipedia.org/wiki/%D0%93%D0%B8%D0%B3%D0%B0%D0%B1%D0%B0%D0%B9%D1%82) дискового пространства и нескольких пользователей, от количества которых зависит цена услуги.
3. Внутренний аккаунт позволяет установить Easy Projects на внутренний сервер покупателя.

Бесплатный хостинговый аккаунт с неограниченным количеством пользователей может быть использован в течение испытательного срока в 15 дней. Существует также [условно бесплатная](https://ru.wikipedia.org/wiki/%D0%A3%D1%81%D0%BB%D0%BE%D0%B2%D0%BD%D0%BE_%D0%B1%D0%B5%D1%81%D0%BF%D0%BB%D0%B0%D1%82%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) версия программы, которую можно использовать в течение 30 дней после установки.

Несмотря на многочисленные достоинства, пользователи системы отмечают некоторые недостатки Easy Projects.

В программе отсутствует возможность задавать уровень подготовки членов команды, нет должных средств для отслеживания материальных ресурсов (не людей и не файлов), отсутствуют средства для анализа рисков. Компания-разработчик не предоставляет мобильной версии сервиса, однако современные мобильные устройства ([iPhone](https://ru.wikipedia.org/wiki/IPhone" \o "IPhone), [BlackBerry](https://ru.wikipedia.org/wiki/BlackBerry" \o "BlackBerry), [G1](https://ru.wikipedia.org/wiki/G1) и другие) позволяют отображать обычные сайты достаточно хорошо и дают возможность пользоваться данным сервисом без ограничений. Заявленная круглосуточная поддержка по [чату](https://ru.wikipedia.org/wiki/%D0%A7%D0%B0%D1%82_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0)) также не соответствует действительности.

Процесс регистрации на сайте недостаточно прост и требует от пользователя заполнения полей, которые могли бы быть опциональными. После подтверждения регистрации процесс создания аккаунта довольно длителен и может занимать до 5-7 минут.

После регистрации аккаунта каждое действие приводит к получению сообщения по электронной почте, что приводит к быстрому засорению [электронного ящика](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D1%8B%D0%B9_%D1%8F%D1%89%D0%B8%D0%BA) избыточными письмами.

На хостинговом аккаунте многие (но не все) ссылки открываются в новом окне, причём повторные нажатия на ссылку обновляют содержимое уже существующего окна без каких-либо предупреждений, что усложняет использование программы. Скорость работы хостингового аккаунта зависит от скорости интернет-соединения, поэтому для медленного соединения или большого количества людей, одновременно работающих с системой, внутренний аккаунт более предпочтителен. Ранние версии программы не поддерживали десятичные дроби в диаграмме Гантта.

**PayDox**

PayDox — [система электронного документооборота](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%8D%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%BE%D0%B3%D0%BE_%D0%B4%D0%BE%D0%BA%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%BE%D0%BE%D0%B1%D0%BE%D1%80%D0%BE%D1%82%D0%B0), управления [бизнес-процессами](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B7%D0%BD%D0%B5%D1%81-%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81) и совместной работой. Разработка компании ООО «Пэйбот». Компания занимается созданием программного обеспечения на web-технологиях для бизнеса.

PayDox — это реализованный на web-технологиях [корпоративный портал](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D1%80%D0%BF%D0%BE%D1%80%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D1%8B%D0%B9_%D0%BF%D0%BE%D1%80%D1%82%D0%B0%D0%BB), использующий [электронный документооборот](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D1%8B%D0%B9_%D0%B4%D0%BE%D0%BA%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%BE%D0%BE%D0%B1%D0%BE%D1%80%D0%BE%D1%82) как инструментальную базу с широким набором стандартных необходимых каждому предприятию функций и неограниченными возможностями расширения и интеграции.

Система PayDox предназначена для управления корпоративными документами, [бизнес-процессами](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B7%D0%BD%D0%B5%D1%81-%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81) предприятия, коллективной работой сотрудников, взаимоотношениями с клиентами и контрагентами.

Система позволяет:

* структурировать [бизнес-процессы](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B7%D0%BD%D0%B5%D1%81-%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81) и управлять [бизнес-процессами](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B7%D0%BD%D0%B5%D1%81-%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81) предприятия;
* организовать коллективную работу сотрудников: создавать корпоративные дискуссионные форумы, проводить групповое обсуждение проектов и документов, формировать отчеты о проделанной работе в среде форума;
* осуществлять контроль сроков исполнения документов, задач и поручений, контроль доступа пользователей к документам и сообщениям в корпоративных дискуссионных форумах;
* обеспечить безопасный web-доступ сотрудников к корпоративному файловому архиву — как извне (при необходимости), так и изнутри корпоративной локальной сети.

Функциональность «PayDox Электронный Документооборот» включает возможности для организации единого [электронного документооборота](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D1%8B%D0%B9_%D0%B4%D0%BE%D0%BA%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%BE%D0%BE%D0%B1%D0%BE%D1%80%D0%BE%D1%82) предприятия в соответствии с российскими нормами [делопроизводства](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%BB%D0%BE%D0%BF%D1%80%D0%BE%D0%B8%D0%B7%D0%B2%D0%BE%D0%B4%D1%81%D1%82%D0%B2%D0%BE):

* электронное согласование и утверждение документов,
* ведение версий документов,
* рассылку и ознакомление сотрудников с документами,
* работу с платежными документами: ведение и учет платежных документов, управление договорами, создание отчетов о платежных балансах по договорам, создание отчетов о платежных балансах по предприятию, контроль исполнения платежей и создание отчетов о задолженностях, контроль затрат и создание отчетов по статьям затрат, формирование и исполнение бюджетов.

Функциональные возможности:

* Case Management/Управление задачами и поручениями — управление совместной работой сотрудников ([Адаптивный кейс-менеджмент](https://ru.wikipedia.org/wiki/%D0%90%D0%B4%D0%B0%D0%BF%D1%82%D0%B8%D0%B2%D0%BD%D1%8B%D0%B9_%D0%BA%D0%B5%D0%B9%D1%81-%D0%BC%D0%B5%D0%BD%D0%B5%D0%B4%D0%B6%D0%BC%D0%B5%D0%BD%D1%82))
* Управление [бизнес-процессами](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B7%D0%BD%D0%B5%D1%81-%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81)
* Управление файловым каталогом, обеспечение безопасного web-доступа к архиву документов, фотографий и видео — как извне, так и изнутри корпоративной локальной сети
* Электронное согласование, утверждение и контроль исполнения документов
* Создание резолюций, ознакомление сотрудников с документами, автоматические e-mail-рассылки уведомлений и документов, формирование Дел
* Управление договорами, ОРД, платежными документами, документооборот бюджетирования
* Ведение всей истории работы с документами и проектами
* Календарное планирование
* Ведение версий документов
* Интеграция с [Microsoft Office](https://ru.wikipedia.org/wiki/Microsoft_Office" \o "Microsoft Office), почтовыми программами, бухгалтерскими системами и другими корпоративными приложениями
* Интеграция с [Microsoft SharePoint](https://ru.wikipedia.org/wiki/Microsoft_SharePoint" \o "Microsoft SharePoint)
* Создание произвольной отчетности, использование простых стандартных средств настройки на новые виды документов
* Безопасная работа с удаленными подразделениями предприятия через web-браузер, по электронной почте и SMS
* Поддержка территориально распределенного документооборота для крупных холдинговых и многофилиальных структур
* Простая установка [системы электронного документооборота](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B_%D1%8D%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%BE%D0%B3%D0%BE_%D0%B4%D0%BE%D0%BA%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%BE%D0%BE%D0%B1%D0%BE%D1%80%D0%BE%D1%82%D0%B0) для малых и средних предприятий
* Управление взаимоотношениями с клиентами и контрагентами ([CRM](https://ru.wikipedia.org/wiki/CRM))
* Поддержка мультиязычных интерфейсов. В настоящее время система позволяет в процессе работы переключаться между языками: русским, английским, украинским, чешским, венгерским. Система может быть быстро локализована для любого другого языка и национальных особенностей ведения корпоративных документов.
* Рабочие места для PayDox могут быть на планшетах [iPad](https://ru.wikipedia.org/wiki/IPad" \o "IPad) и [Android](https://ru.wikipedia.org/wiki/Android" \o "Android).

К недостаткам системы можно отнести отсутствие многоплатформенности для сервера системы — система базируется на использовании ПО [Microsoft](https://ru.wikipedia.org/wiki/Microsoft" \o "Microsoft): [Microsoft Internet Information Services](https://ru.wikipedia.org/wiki/Microsoft_Internet_Information_Services" \o "Microsoft Internet Information Services) и [СУБД](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B1%D0%B0%D0%B7%D0%B0%D0%BC%D0%B8_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) [Microsoft SQL Server](https://ru.wikipedia.org/wiki/Microsoft_SQL_Server" \o "Microsoft SQL Server) или [Microsoft Access](https://ru.wikipedia.org/wiki/Microsoft_Access" \o "Microsoft Access).

Следует отметить, что многоплатформенность для рабочих мест пользователей системы обеспечена — они могут быть как под операционной системой [Windows](https://ru.wikipedia.org/wiki/Windows" \o "Windows), так и под операционной системой [Linux](https://ru.wikipedia.org/wiki/Linux" \o "Linux) и использовать для работы браузеры: [Microsoft Internet Explorer](https://ru.wikipedia.org/wiki/Microsoft_Internet_Explorer" \o "Microsoft Internet Explorer), [Mozilla Firefox](https://ru.wikipedia.org/wiki/Mozilla_Firefox" \o "Mozilla Firefox), [Google Chrome](https://ru.wikipedia.org/wiki/Google_Chrome" \o "Google Chrome), [Apple Safari](https://ru.wikipedia.org/wiki/Apple_Safari" \o "Apple Safari).

# Проектирование информационной системы

## Разработка платформы проектирования и её обоснование

Прежде, чем приступить к разработке информационной системы, была изучена информация о существующих средствах для разработки ИС. После долгих поисков в интернете и сравнивания различных вариантов, для разработки данного дипломного проекта была выбрана клиент-серверная архитектура с сервером приложений.

В рамках многоуровневого представления вычислительных систем можно выделить три группы функций, ориентированных на решение различных подзадач:

функции ввода и отображения данных (обеспечивают взаимодействие с пользователем);

прикладные функции, характерные для данной предметной области;

функции управления ресурсами (файловой системой, базой даных и т.д.)

*[](http://www.4stud.info/networking/img/application-layers.png)*

Рисунок 3. Компоненты сетевого приложения

Выполнение этих функций в основном обеспечивается программными средствами, которые можно представить в виде взаимосвязанных компонентов (рис. 8), где:

*компонент представления* отвечает за пользовательский интерфейс;

*прикладной компонент* реализует алгоритм решения конкретной задачи;

*компонент управления* ресурсом обеспечивает доступ к необходимым ресурсам.

Автономная система (компьютер, не подключенный к сети) представляет все эти компоненты как на различных уровнях (ОС, служебное ПО и утилиты, прикладное ПО), так и на уровне приложений (не характерно для современных программ). Так же и сеть — она представляет все эти компоненты, но, в общем случае, распределенные между узлами. Задача сводится к обеспечению сетевого взаимодействия между этими компонентами.

**Архитектура «клиент-сервер»** определяет общие принципы организации взаимодействия в сети, где имеются *серверы*, узлы-поставщики некоторых специфичных функций (сервисов) и *клиенты*, потребители этих функций.

Практические реализации такой архитектуры называются **клиент-серверными технологиями**. Каждая технология определяет собственные или использует имеющиеся правила взаимодейстия между клиентом и сервером, которые называются *протоколом обмена (протоколом взаимодействия)*.

**Двухзвенная архитектура**

В любой сети (даже одноранговой), построенной на современных сетевых технологиях, присутствуют элементы клиент-серверного взаимодействия, чаще всего на основе **двухзвенной архитектуры**. Двухзвенной (two-tier, 2-tier) она называется из-за необходимости распределения *трех* базовых *компонентов* между *двумя узлами* (клиентом и сервером).

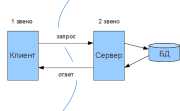
*[](http://www.4stud.info/networking/img/2-tier.png)*

Рисунок 4. Двухзвенная клиент-серверная архитектура

Двухзвенная архитектура используется в клиент-серверных системах, где сервер отвечает на клиентские запросы напрямую и в полном объеме, при этом используя только собственные ресурсы. Т.е. сервер не вызывает сторонние сетевые приложения и не обращается к сторонним ресурсам для выполнения какой-либо части запроса (рис. 9)

Расположение компонентов на стороне клиента или сервера определяет следующие основные модели их взаимодействия в рамках двухзвенной архитектуры:

**сервер терминалов** — распределенное представление данных;

**файл-сервер** — доступ к удаленной базе данных и файловым ресурсам;

**сервер БД** — удаленное представление данных;

**сервер приложений** — удаленное приложение.

Исторически первой появилась модель распределенного представления данных (модель сервер терминалов). Она реализовывалась на универсальной ЭВМ (мэйнфрейме), выступавшей в роли сервера, с подключенными к ней алфавитно-цифровыми терминалами. Пользователи выполняли ввод данных с клавиатуры терминала, которые затем передавались на мэйнфрейм и там выполнялась их обработка, включая формирование «картинки» с результатами. Эта «картинка» и возвращалась пользователю на экран терминала.

С появлением персональных компьютеров и локальных сетей, была реализована модель файлового сервера, представлявшего доступ файловым ресурсам, в т.ч и к удаленной базе данных. В этом случае выделенный узел сети является файловым сервером, на котором размещены файлы базы данных. На клиентах выполняются приложения, в которых совмещены компонент представления и прикладной компонент (СУБД и прикладная программма), использующие подключенную удаленную базу как локальный файл. Протоколы обмена при этом представляют набор низкоуровневых вызовов операций файловой системы.

Такая модель показала свою неэффективность ввиду того, что при активной работе с таблицами БД возникает большая нагрузка на сеть. Частичным решением является поддержка тиражирования (репликации) таблиц и запросов. В этом случае, например при изменении данных, обновляется не вся таблица, а только модифицированная ее часть.

С появлением специализированных СУБД появилась возможность реализации другой модели доступа к удаленной базе данных — модели сервера баз данных. В этом случае ядро СУБД функционирует на сервере, прикладная программа на клиенте, а протокол обмена обеспечивается с помощью языка SQL. Такой подход по сравнению с файловым сервером ведет к уменьшению загрузки сети и унификации интерфейса «клиент-сервер». Однако, сетевой трафик остается достаточно высоким, кроме того, по прежнему невозможно удовлетворительное администрирование приложений, поскольку в одной программе совмещаются различные функции.

С разработкой и внедрением на уровне серверов баз данных механизма хранимых процедур появилась концепция *активного сервера БД*. В этом случае часть функций прикладного компонента реализованы в виде хранимых процедур, выполняемых на стороне сервера. Остальная прикладная логика выполняется на клиентской стороне. Протокол взаимодействия — соответствующий диалект языка SQL.

Преимущества такого подхода очевидны:

возможно централизованное администрирование прикладных функций;

снижение стоимости владения системой (TOC, total cost of ownership) за счет [аренды сервера](http://www.di-net.ru/collocation/dedicated/), а не его покупки;

значительное снижение сетевого трафика (т.к. передаются не SQL-запросы, а вызовы хранимых процедур).

Основной недостаток — ограниченность средств разработки хранимых процедур по сравнению с языками высокого уровня.

Реализация прикладного компонента на стороне сервера представляет следующую модель — сервер приложений. Перенос функций прикладного компонента на сервер снижает требования к конфигурации клиентов и упрощает администрирование, но представляет повышенные требования к производительности, безопасности и надежности сервера.

В настоящее время намечается тенденция возврата к тому, с чего начиналась клиент-серверная архитектура — к централизации вычислений на основе модели терминал-сервера. В современной реинкарнации терминалы отличаются от своих алфавитно-цифровых предков тем, что имея минимум программных и аппаратных средств, представляют мультимедийные возможности (в т.ч. *графический пользовательский интерфейс*). Работу терминалов обеспечивает высокопроизводительный сервер, куда вынесено все, вплоть до виртуальных драйверов устройств, включая драйверы видеоподсистемы.

**Трехзвенная архитектура**

Еще одна тенденция в клиент-серверных технологиях связана со все большим использованием распределенных вычислений. Они реализуются на основе модели сервера приложений, где сетевое приложение разделено на две и более частей, каждая из которых может выполняться на отдельном компьютере. Выделенные части приложения взаимодействуют друг с другом, обмениваясь сообщениями в заранее согласованном формате. В этом случае двухзвенная клиент-серверная архитектура становится **трехзвенной (three-tier, 3-tier)**.

Как правило, третьим звеном в трехзвенной архитектуре становится сервер приложений, т.е. компоненты распределяются следующим образом (рис. 11):

Представление данных — на стороне клиента.

Прикладной компонент — на выделенном сервере приложений (как вариант, выполняющем функции [промежуточного ПО](http://www.4stud.info/networking/lecture6.html)).

Управление ресурсами — на сервере БД, который и представляет запрашиваемые данные.

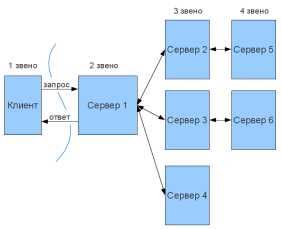
[](http://www.4stud.info/networking/img/N-tier.png)

Рисунок 5. Многозвенная (N-tier) клиент-серверная архитектура

Трехзвенная архитектура может быть расширена до **многозвенной (N-tier, Multi-tier)** путем выделения дополнительных серверов, каждый из которых будет представлять собственные сервисы и пользоваться услугами прочих серверов разного уровня. Абстрактный пример многозвенной модели приведен на рис. 12.

**Сравнение архитектур**

Двухзвенная архитектура проще, так как все запросы обслуживаются одним сервером, но именно из-за этого она менее надежна и предъявляет повышенные требования к производительности сервера.

Трехзвенная архитектура сложнее, но благодаря тому, что функции распределены между серверами второго и третьего уровня, эта архитектура представляет:

Высокую степень гибкости и масштабируемости.

Высокую безопасность (т.к. защиту можно определить для каждого сервиса или уровня).

Высокую производительность (т.к. задачи распределены между серверами).

**Клиент-серверные технологии**

Архитектура клиент-сервер применяется в большом числе сетевых технологий, используемых для доступа к различным сетевым сервисам. Кратко рассмотрим некоторые типы таких сервисов (и серверов).

**Web-серверы**

Изначально представляли доступ к гипертекстовым документам по протоколу HTTP (Huper Text Transfer Protocol). Сейчас поддерживают расширенные возможности, в частности работу с бинарными файлами (изображения, мультимедиа и т.п.).

**Серверы приложений**

Предназначены для централизованного решения прикладных задач в некоторой предметной области. Для этого пользователи имеют право *запускать серверные программы* на исполнение. Использование серверов приложений позволяет снизить требования к конфигурации клиентов и упрощает общее управление сетью.

**Серверы баз данных**

Серверы баз данных используются для обработки пользовательских запросов на языке SQL. При этом СУБД находится на сервере, к которому и подключаются клиентские приложения.

**Файл-серверы**

Файл-сервер *хранит* информацию в виде файлов и представляет пользователям доступ к ней. Как правило файл-сервер обеспечивает и определенный уровень защиты от несакционированного доступа.

**Прокси-сервер**

Во-первых, действует как посредник, помогая пользователям получить информацию из Интернета и при этом обеспечивая защиту сети.

Во-вторых, сохраняет часто запрашиваемую информацию в кэш-памяти на локальном диске, быстро доставляя ее пользователям без повторного обращения к Интернету.

**Файрволы** (брандмауэры)

Межсетевые экраны, анализирующие и фильтрующие проходящий сетевой трафик, с целью обеспечения безопасности сети.

**Почтовые серверы**

Представляют услуги по отправке и получению электронных почтовых сообщений.

Серверы удаленного доступа (RAS)

Эти системы обеспечивают связь с сетью по коммутируемым линиям. Удаленный сотрудник может использовать ресурсы корпоративной ЛВС, подключившись к ней с помощью обычного модема.

Это лишь несколько типов из всего многообразия клиент-серверных технологий, используемых как в локальных, так и в глобальных сетях.

Для доступа к тем или иным сетевам сервисам используются клиенты, возможности которых характеризуются понятием «толщины». Оно определяет конфигурацию оборудования и программное обеспечение, имеющиеся у клиента. Рассмотрим возможные граничные значения:

«Тонкий» клиент

Этот термин определяет клиента, вычислительных ресурсов которого достаточно лишь для запуска необходимого сетевого приложения через web-интерфейс. Пользовательский интерфейс такого приложения формируется средствами *статического* HTML (выполнение JavaScript не предусматривается), вся прикладная логика выполняется на сервере.  
Для работы тонкого клиента достаточно лишь обеспечить возможность запуска web-браузера, в окне которого и осуществляются все действия. По этой причине web-браузер часто называют "универсальным клиентом".

«Толстый» клиент

Таковым является рабочая станция или персональный компьютер, работающие под управлением собственной дисковой операционной системы и имеющие необходимый набор программного обеспечения. К сетевым серверам «толстые» клиенты обращаются в основном за дополнительными услугами (например, доступ к web-серверу или корпоративной базе данных).  
Так же под «толстым» клиентом подразумевается и клиентское сетевое приложение, запущенное под управлением локальной ОС. Такое приложение совмещает компонент представления данных (графический пользовательский интерфейс ОС) и прикладной компонент (вычислительные мощности клиентского компьютера).

В последнее время все чаще используется еще один термин: «rich»-client. «Rich«-клиент своего рода компромисс между «толстым» и «тонким» клиентом. Как и «тонкий» клиент, «rich»-клиент также представляет графический интерфейс, описываемый уже средствами XML и включающий некоторую функциональность толстых клиентов (например интерфейс drag-and-drop, вкладки, множественные окна, выпадающие меню и т.п.)

Прикладная логика «rich»-клиента также реализована на сервере. Данные отправляются в стандартном формате обмена, на основе того же XML (протоколы SOAP, XML-RPC) и интерпретируются клиентом.

Некоторые основные протоколы «rich»-клиентов на базе XML приведены ниже:

XAML (eXtensible Application Markup Language) — разработан Microsoft, используется в приложениях на платформе .NET;

XUL (XML User Interface Language) — стандарт, разработанный в рамках проекта Mozilla, используется, например, в почтовом клиенте Mozilla Thunderbird или браузере Mozilla Firefox;

Flex — мультимедийная технология на основе XML, разработанная Macromedia/Adobe.

Для решения данной задачи была выбрана трехзвенная архитектура на тонком клиенте. Сервер приложений так же будет совмещать в себе web-сервер, так как разделение этих модулей на 2 части не целесообразно.

**Использование BPMN-нотаци и движка бизнесспроцессов Camunda.**

BPMN (Business Process Management Notation) – это язык моделирования бизнес-процессов, который является промежуточным звеном между формализацией/визуализацией и воплощением бизнес-процесса.  
Говоря проще, такая нотация представляет собой описание графических элементов, используемых для построения схемы протекания бизнес-процесса.  
Как минимум, такая схема нужна, чтобы выстроить в соответствии с ней бизнес процесс и понятно регламентировать его для всех участников.  
Как максимум, [моделирование BPMN](https://www.comindware.com/ru/bpmn/) позволяет впоследствии провести автоматизацию бизнес-процессов в соответствии с имеющейся схемой.

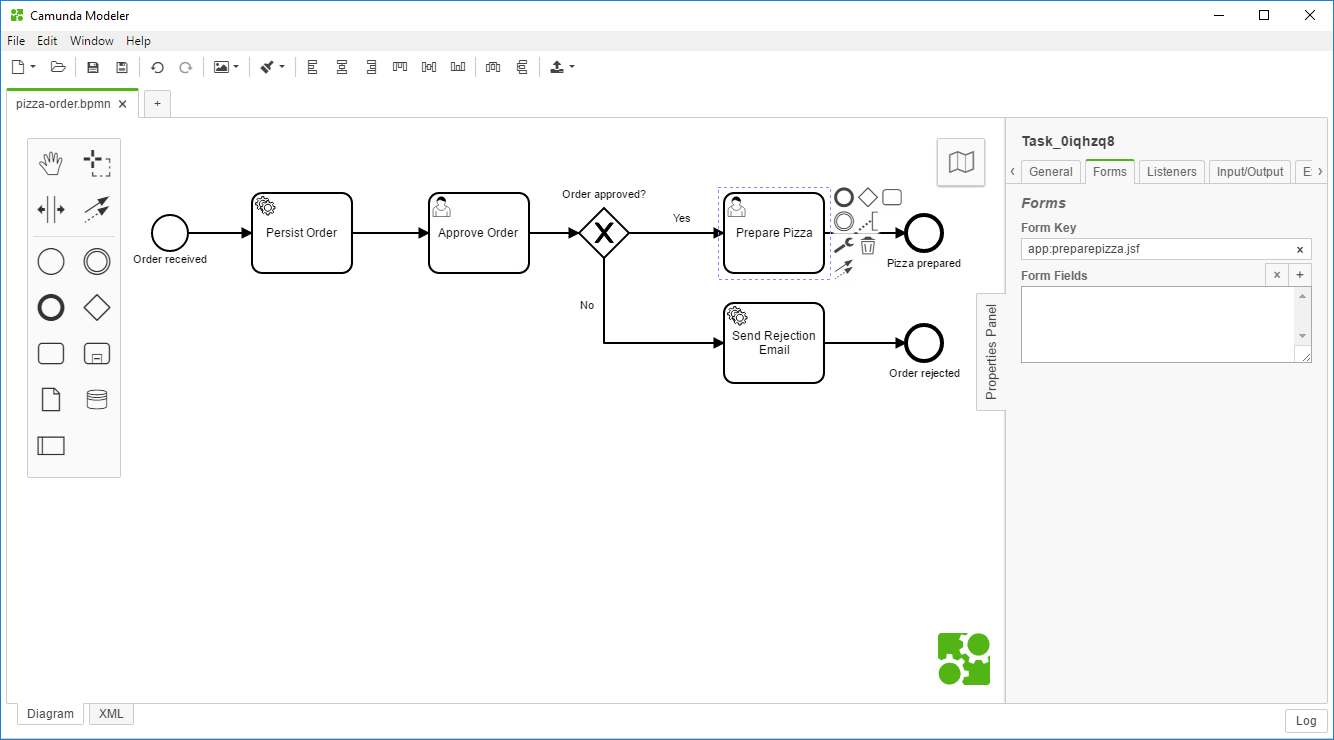


Рисунок 6. Интерфейс Camunda Modeler

Первая версия нотации BPMN вышла в мае 2004 года (BPMN 1.0). Следующая версия появилась в январе 2011 года (BPMN 2.0). Наконец, в январе 2013 года компания OMG выпустила ту версию, которая в основном используется и сегодня – BPMN 2.0.2.

BPMN-описание бизнес процесса имеет несколько преимуществ.  
Первое – простота трансляции диаграмм в исполняемые модели с помощью языка формального описания бизнес-процессов.  
Описание элементов BPMN является понятным для большинства участников бизнес-процессов и часто не требует никаких дополнительных разъяснений. С помощью простого графического выражения можно составить конкретные регламенты, которые будут исполняться сотрудниками.  
Наряду с тем, что описание нотации BPMN 2.0 позволяет добиться понимания сотрудниками того, как происходят бизнес-процессы, данную нотацию поддерживают большинство современных инструментов бизнес-моделирования, что позволяет импорт готовых схем бизнес-процессов в BPM-системы.

В случае же данного дипломного проекта, бизнесс-процессы будут использоваться в купе с мощным движком бизнесс процессов Camunda. Это требуется для возможности изменять очередность этапов процесса согласования, без изменения исходного кода.



Рисунок 7. Логотип Camunda

Camunda — это open-source-платформа для моделирования бизнес-процессов, которая написана на Java и в качестве языка разработки использует Java. Она представляет собой набор библиотек, которые и позволяют выполнять описанные процессы. Для интеграции Camunda в проект достаточно добавить несколько зависимостей. Для хранения процессов можно выбрать in-memory или персистентную СУБД — в зависимости от задач. Мы выбрали Postgre, так как нам важна история для «разбора полетов». По умолчанию платформа разворачивается на H2.  
  
Разработка состоит из двух частей: создание flow-процесса в специальной утилите Camunda Modeler и написание java-кода, который обрабатывает шаги процесса, описанные на диаграмме. Чтобы из процесса вызвать java-код, достаточно реализовать интерфейс JavaDelegate, поднять этот Bean (можно указать делагат по полному имени, но через Bean удобнее и гибче) в контексте и указать его id на нужном шаге процесса. На Kotlin делегат выглядит еще более лаконично. Логика работы делегатов довольно проста: что-то вычитали из контекста, выполнили какие-то действия и положили обратно в контекст.

## Выбор средства программирования

**Back-end часть**

**Среда разработки**

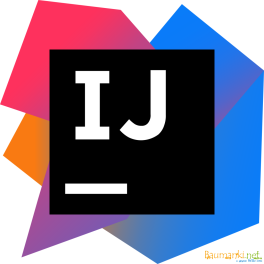


Рисунок 8. Логотип JetBrains IntelliJ IDEA

**JetBrains IntelliJ IDEA** – это ведущая среда быстрой разработки на языке Java. IntelliJ IDEA представляет собой высокотехнологичный комплекс тесно интегрированных инструментов программирования, включающий интеллектуальный редактор исходных текстов с развитыми средствами автоматизации, мощные инструменты рефакторинга кода, встроенную поддержку технологий J2EE, механизмы интеграции со средой тестирования Ant/JUnit и системами управления версиями, уникальный инструмент оптимизации и проверки кода Code Inspection, а также инновационный визуальный конструктор графических интерфейсов.

[**Cистема управления версиями**](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%8F%D0%BC%D0%B8)



Рисунок 9. Логотип Git

**Git** (произносится «гит») — распределённая [система управления версиями](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%8F%D0%BC%D0%B8). Проект был создан [Линусом Торвальдсом](https://ru.wikipedia.org/wiki/%D0%A2%D0%BE%D1%80%D0%B2%D0%B0%D0%BB%D1%8C%D0%B4%D1%81,_%D0%9B%D0%B8%D0%BD%D1%83%D1%81" \o "Торвальдс, Линус) для управления разработкой [ядра Linux](https://ru.wikipedia.org/wiki/%D0%AF%D0%B4%D1%80%D0%BE_Linux), первая версия выпущена [7 апреля](https://ru.wikipedia.org/wiki/7_%D0%B0%D0%BF%D1%80%D0%B5%D0%BB%D1%8F) [2005 года](https://ru.wikipedia.org/wiki/2005_%D0%B3%D0%BE%D0%B4). На сегодняшний день его поддерживает [Джунио Хамано](https://ru.wikipedia.org/wiki/%D0%A5%D0%B0%D0%BC%D0%B0%D0%BD%D0%BE,_%D0%94%D0%B6%D1%83%D0%BD%D0%B8%D0%BE" \o "Хамано, Джунио).

Среди проектов, использующих Git — [ядро Linux](https://ru.wikipedia.org/wiki/%D0%AF%D0%B4%D1%80%D0%BE_Linux), [Swift](https://ru.wikipedia.org/wiki/Swift_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)" \o "Swift (язык программирования)), [Android](https://ru.wikipedia.org/wiki/Android" \o "Android), [Drupal](https://ru.wikipedia.org/wiki/Drupal" \o "Drupal), [Cairo](https://ru.wikipedia.org/wiki/Cairo" \o "Cairo), [GNU Core Utilities](https://ru.wikipedia.org/wiki/GNU_Coreutils), [Mesa](https://ru.wikipedia.org/wiki/Mesa_3D" \o "Mesa 3D), [Wine](https://ru.wikipedia.org/wiki/Wine" \o "Wine), [Chromium](https://ru.wikipedia.org/wiki/Chromium" \o "Chromium), [Compiz Fusion](https://ru.wikipedia.org/wiki/Compiz_Fusion" \o "Compiz Fusion), [FlightGear](https://ru.wikipedia.org/wiki/FlightGear" \o "FlightGear), [jQuery](https://ru.wikipedia.org/wiki/JQuery" \o "JQuery), [PHP](https://ru.wikipedia.org/wiki/PHP), [NASM](https://ru.wikipedia.org/wiki/NASM), [MediaWiki](https://ru.wikipedia.org/wiki/MediaWiki" \o "MediaWiki), [DokuWiki](https://ru.wikipedia.org/wiki/DokuWiki" \o "DokuWiki), [Qt](https://ru.wikipedia.org/wiki/Qt" \o "Qt), ряд дистрибутивов [Linux](https://ru.wikipedia.org/wiki/Linux" \o "Linux).

Программа является свободной и выпущена под лицензией [GNU GPL](https://ru.wikipedia.org/wiki/GNU_General_Public_License) версии По умолчанию используется TCP порт 9418.

**Пакетный менеджер**

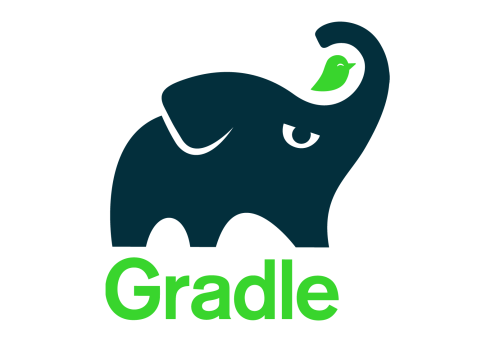


Рисунок 10. Логотип Gradle

**Gradle** — [система автоматической сборки](https://ru.wikipedia.org/wiki/%D0%90%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F_%D1%81%D0%B1%D0%BE%D1%80%D0%BA%D0%B8), построенная на принципах [Apache Ant](https://ru.wikipedia.org/wiki/Apache_Ant" \o "Apache Ant) и [Apache Maven](https://ru.wikipedia.org/wiki/Apache_Maven" \o "Apache Maven), но предоставляющая [DSL](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B5%D0%B4%D0%BC%D0%B5%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) на языках [Groovy](https://ru.wikipedia.org/wiki/Groovy" \o "Groovy) и [Kotlin](https://ru.wikipedia.org/wiki/Kotlin" \o "Kotlin) вместо традиционной [XML](https://ru.wikipedia.org/wiki/XML)-образной формы представления конфигурации проекта.

В отличие от [Apache Maven](https://ru.wikipedia.org/wiki/Apache_Maven" \o "Apache Maven), основанного на концепции жизненного цикла проекта, и [Apache Ant](https://ru.wikipedia.org/wiki/Apache_Ant" \o "Apache Ant), в котором порядок выполнения задач (targets) определяется отношениями зависимости (depends-on), Gradle использует [направленный ациклический граф](https://ru.wikipedia.org/wiki/%D0%9D%D0%B0%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%BD%D1%8B%D0%B9_%D0%B0%D1%86%D0%B8%D0%BA%D0%BB%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B3%D1%80%D0%B0%D1%84) для определения порядка выполнения задач.

Gradle был разработан для расширяемых многопроектных сборок, и поддерживает [инкрементальные сборки](https://ru.wikipedia.org/wiki/%D0%98%D1%82%D0%B5%D1%80%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%B0%D1%8F_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0), определяя, какие компоненты дерева сборки не изменились и какие задачи, зависимые от этих частей, не требуют перезапуска.

Основные [плагины](https://ru.wikipedia.org/wiki/%D0%9F%D0%BB%D0%B0%D0%B3%D0%B8%D0%BD" \o "Плагин) предназначены для разработки и развертывания [Java](https://ru.wikipedia.org/wiki/Java" \o "Java), [Groovy](https://ru.wikipedia.org/wiki/Groovy" \o "Groovy) и [Scala](https://ru.wikipedia.org/wiki/Scala_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)" \o "Scala (язык программирования)) приложений, но готовятся плагины и для других [языков программирования](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F).

**СУБД**

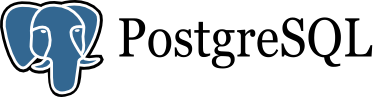


Рисунок 11. Логотип PostgreSQL

PostgreSQL - это свободно распространяемая объектно-реляционная система управления базами данных (ORDBMS), наиболее развитая из открытых СУБД в мире и являющаяся реальной альтернативой коммерческим базам данных.

**Система управления версии БД**

Файл:Liquibase Datical.png

Рисунок 12. Логотип Liquibase

Liquibase - независимая от базы данных библиотека для отслеживания, управления и применения изменений схемы базы данных. Библиотека была запущена в 2006 году, чтобы упростить отслеживание изменений в базе данных, особенно в гибкой среде разработки программного обеспечения.

**Язык программирования**



Рисунок 13. Логотип JDK 8

JDK 8(Java Development Kit 1.8) - объектно-ориентированный язык программирования, разрабатываемый компанией Sun Microsystems с 1991 года и официально выпущенный 23 мая 1995 года. Изначально новый язык программирования назывался Oak (James Gosling) и разрабатывался для бытовой электроники, но впоследствии был переименован в Java и стал использоваться для написания апплетов, приложений и серверного программного обеспечения.

Программы на Java могут быть транслированы в байт-код, выполняемый на виртуальной java-машине (JVM) — программе, обрабатывающей байт-код и передающей инструкции оборудованию, как интерпретатор, но с тем отличием, что байт-код, в отличие от текста, обрабатывается значительно быстрее.

Язык Java зародился как часть проекта создания передового программного обеспечения для различных бытовых приборов. Реализация проекта была начата на языке [C++](http://progopedia.ru/language/c-plus-plus/), но вскоре возник ряд проблем, наилучшим средством борьбы с которыми было изменение самого инструмента — языка программирования. Стало очевидным, что необходим платформо-независимый язык программирования, позволяющий создавать программы, которые не приходилось бы компилировать отдельно для каждой архитектуры и можно было бы использовать на различных процессорах под различными операционными системами.

Язык Java потребовался для создания интерактивных продуктов для сети Internet. Фактически, большинство архитектурных решений, принятых при создании Java, было продиктовано желанием предоставить синтаксис, сходный с [C](http://progopedia.ru/language/c/) и [C++](http://progopedia.ru/language/c-plus-plus/). В Java используются практически идентичные соглашения для объявления переменных, передачи параметров, операторов и для управления потоком выполнением кода. В Java добавлены все хорошие черты [C++](http://progopedia.ru/language/c-plus-plus/).

Три ключевых элемента объединились в технологии языка Java

Java предоставляет для широкого использования свои апплеты (applets) — небольшие, надежные, динамичные, не зависящие от платформы активные сетевые приложения, встраиваемые в страницы Web. Апплеты Java могут настраиваться и распространяться потребителям с такой же легкостью, как любые документы HTML

Java высвобождает мощь объектно-ориентированной разработки приложений, сочетая простой и знакомый синтаксис с надежной и удобной в работе средой разработки. Это позволяет широкому кругу программистов быстро создавать новые программы и новые апплеты

Java предоставляет программисту богатый набор классов объектов для ясного абстрагирования многих системных функций, используемых при работе с окнами, сетью и для ввода-вывода. Ключевая черта этих классов заключается в том, что они обеспечивают создание независимых от используемой платформы абстракций для широкого спектра системных интерфейсов

**Пакеты для разработки**

**Spring Boot**



Рисунок 14. Логотип Spring Boot

Авторы Spring решили предоставить разработчикам некоторые утилиты, которые автоматизируют процедуру настройки и ускоряют процесс создания и развертывания Spring-приложений, под общим названием **Spring Boot**

**Spring Boot** — это полезный проект, целью которого является упрощение создания приложений на основе Spring. Он позволяет наиболее простым способом создать web-приложение, требуя от разработчиков минимум усилий по его настройке и написанию кода

**Особенности Spring Boot**

**Spring Boot** обладает большим функционалом, но его наиболее значимыми особенностями являются: управление зависимостями, автоматическая конфигурация и встроенные контейнеры сервлетов

**Простота управления зависимостями**

Чтобы ускорить процесс управления зависимостями, Spring Boot неявно упаковывает необходимые сторонние зависимости для каждого типа приложения на основе Spring и предоставляет их разработчику посредством так называемых **starter**-пакетов (spring-boot-starter-web, spring-boot-starter-data-jpa и т.д.

**Starter**-пакеты представляют собой набор удобных дескрипторов зависимостей, которые можно включить в свое приложение. Это позволит получить универсальное решение для всех, связанных со Spring технологий, избавляя программиста от лишнего поиска примеров кода и загрузки из них требуемых дескрипторов зависимостей (пример таких дескрипторов и стартовых пакетов будет показан ниже)

Например, если вы хотите начать использовать Spring Data JPA для доступа к базе данных, просто включите в свой проект зависимость **spring-boot-starter-data-jpa** и все будет готово (вам не придется искать совместимые драйверы баз данных и библиотеки Hibernate)

Если вы хотите создать Spring web-приложение, просто добавьте зависимость **spring-boot-starter-web**, которая подтянет в проект все библиотеки, необходимые для разработки Spring MVC-приложений, таких как **spring-webmvc**, **jackson-json**, **validation-api** и **Tomcat**

Другими словами, **Spring Boot** собирает все общие зависимости и определяет их в одном месте, что позволяет разработчикам просто использовать их, вместо того, чтобы изобретать колесо каждый раз, когда они создают новое приложение

Следовательно, при использовании **Spring Boot**, файл **pom.xml** содержит намного меньше строк, чем при использовании его в Spring-приложениях

Обратитесь к [документации](https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#using-boot-starter), чтобы ознакомиться со всеми **Spring Boot starter-пакетами**

**Автоматическая конфигурация**

Второй превосходной возможностью **Spring Boot** является автоматическая конфигурация приложения

После выбора подходящего **starter**-пакета, **Spring Boot** попытается автоматически настроить Spring-приложение на основе добавленных вами **jar**-зависимостей

Например, если вы добавите **Spring-boot-starter-web**, Spring Boot автоматически сконфигурирует такие зарегистрированные бины, как **DispatcherServlet**, **ResourceHandlers**, **MessageSource**

Если вы используете **spring-boot-starter-jdbc**, **Spring Boot** автоматически регистрирует бины **DataSource**, **EntityManagerFactory**, **TransactionManager** и считывает информацию для подключения к базе данных из файла **application.properties**

Если вы не собираетесь использовать базу данных, и не предоставляете никаких подробных сведений о подключении в ручном режиме, Spring Boot автоматически настроит базу в памяти, без какой-либо дополнительной конфигурации с вашей стороны (при наличии H2 или HSQL библиотек)

Автоматическая конфигурация может быть полностью переопределена в любой момент с помощью пользовательских настроек

**Встроенная поддержка сервера приложений — контейнера сервлетов**  
Каждое Spring Boot web-приложение включает встроенный web-сервер. Посмотрите на [список](https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#getting-started-system-requirements-servlet-containers) контейнеров сервлетов, которые поддерживаются "из коробки"

Разработчикам теперь не надо беспокоиться о настройке контейнера сервлетов и развертывании приложения на нем. Теперь приложение может запускаться само, как исполняемый jar-файл с использованием встроенного сервера

Если вам нужно использовать отдельный HTTP-сервер, для этого достаточно исключить зависимости по умолчанию. Spring Boot предоставляет отдельные starter-пакеты для разных HTTP-серверов

Создание автономных web-приложений со встроенными серверами не только удобно для разработки, но и является допустимым решением для приложений корпоративного уровня и становится все более полезно в мире микросервисов. Возможность быстро упаковать весь сервис (например, аутентификацию пользователя) в автономном и полностью развертываемом артефакте, который также предоставляет API — делает установку и развертывание приложения значительно проще

**Hibernate**



Рисунок 15. Логотип Hibernate

ORM-решением для языка [Java](http://ru.wikipedia.org/wiki/Java), является технология [Hibernate](http://ru.wikipedia.org/wiki/Hibernate_(%D0%B1%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0)), которая не только заботится о связи Java классов с таблицами базы данных (и типов данных Java в типы данных [SQL](http://ru.wikipedia.org/wiki/Sql)), но также предоставляет средства для автоматического построения запросов и извлечения данных и может значительно уменьшить время разработки, которое обычно тратится на ручное написание SQL и [JDBC](http://ru.wikipedia.org/wiki/Jdbc) кода. Hibernate генерирует SQL вызовы и освобождает разработчика от ручной обработки результирующего набора данных и конвертации объектов, сохраняя приложение портируемым во все SQL базы данных.

**Front-end часть**

**HTML**



Рисунок 16. Логотип HTML

**HTML** (от [англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *HyperText Markup Language* — «язык [гипертекстовой](https://ru.wikipedia.org/wiki/%D0%93%D0%B8%D0%BF%D0%B5%D1%80%D1%82%D0%B5%D0%BA%D1%81%D1%82) разметки») — стандартизированный язык разметки документов во [Всемирной паутине](https://ru.wikipedia.org/wiki/%D0%92%D1%81%D0%B5%D0%BC%D0%B8%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D0%B0%D1%83%D1%82%D0%B8%D0%BD%D0%B0). Большинство [веб-страниц](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D1%82%D1%80%D0%B0%D0%BD%D0%B8%D1%86%D0%B0" \o "Веб-страница) содержат описание разметки на языке HTML (или [XHTML](https://ru.wikipedia.org/wiki/XHTML)). Язык HTML интерпретируется [браузерами](https://ru.wikipedia.org/wiki/%D0%91%D1%80%D0%B0%D1%83%D0%B7%D0%B5%D1%80); полученный в результате интерпретации форматированный текст отображается на экране монитора компьютера или мобильного устройства.

Язык HTML до 5-й версии определялся как приложение [SGML](https://ru.wikipedia.org/wiki/SGML) (стандартного обобщённого языка разметки по стандарту [ISO](https://ru.wikipedia.org/wiki/ISO) 8879). Спецификации HTML5 формулируются в терминах [DOM](https://ru.wikipedia.org/wiki/Document_Object_Model) (объектной модели документа).

Язык [XHTML](https://ru.wikipedia.org/wiki/XHTML) является более строгим вариантом HTML, он следует синтаксису [XML](https://ru.wikipedia.org/wiki/XML) и является приложением языка XML в области разметки гипертекста.

Во всемирной паутине HTML-страницы, как правило, передаются браузерам от сервера по протоколам [HTTP](https://ru.wikipedia.org/wiki/HTTP) или [HTTPS](https://ru.wikipedia.org/wiki/HTTPS), в виде простого текста или с использованием [шифрования](https://ru.wikipedia.org/wiki/%D0%A8%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5).

**Bootstrap**



Рисунок 17. Логотип Bootstrap

**Bootstrap** (также известен как **Twitter Bootstrap**[[3]](https://ru.wikipedia.org/wiki/Bootstrap_(%D1%84%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA)" \l "cite_note-_e02fd5ade744736e-3)[[4]](https://ru.wikipedia.org/wiki/Bootstrap_(%D1%84%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA)#cite_note-_8b1a178567366f5a-4)[[5]](https://ru.wikipedia.org/wiki/Bootstrap_(%D1%84%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA)#cite_note-_9ea25a04bd658533-5)) — свободный набор инструментов для создания [сайтов](https://ru.wikipedia.org/wiki/%D0%A1%D0%B0%D0%B9%D1%82) и [веб-приложений](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5" \o "Веб-приложение). Включает в себя [HTML](https://ru.wikipedia.org/wiki/HTML)- и [CSS](https://ru.wikipedia.org/wiki/CSS)-шаблоны оформления для [типографики](https://ru.wikipedia.org/wiki/%D0%A2%D0%B8%D0%BF%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D0%BA%D0%B0" \o "Типографика), веб-форм, кнопок, меток, блоков навигации и прочих компонентов веб-интерфейса, включая [JavaScript](https://ru.wikipedia.org/wiki/JavaScript)-расширения.

**Apache FreeMarker**



Рисунок 18. Логотип Apache FreeMarker

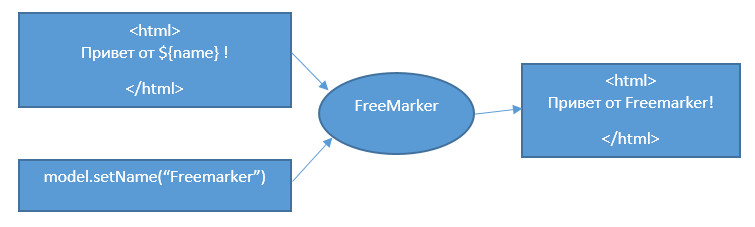
Apache FreeMarker — это механизм шаблонов: библиотека Java для генерации текстового вывода (HTML-страницы, xml, файлы конфигурации, исходный код и.т.д. На вход подается шаблон, например html в котором есть специальные выражения, подготавливаются данные соответствующие этим выражением, а Freemarker динамически вставляет эти данные и получается динамически заполненный документ.  
  


Рисунок 19. Схема взаимодействия Apache FreeMarker

## Реализация Back-End части.

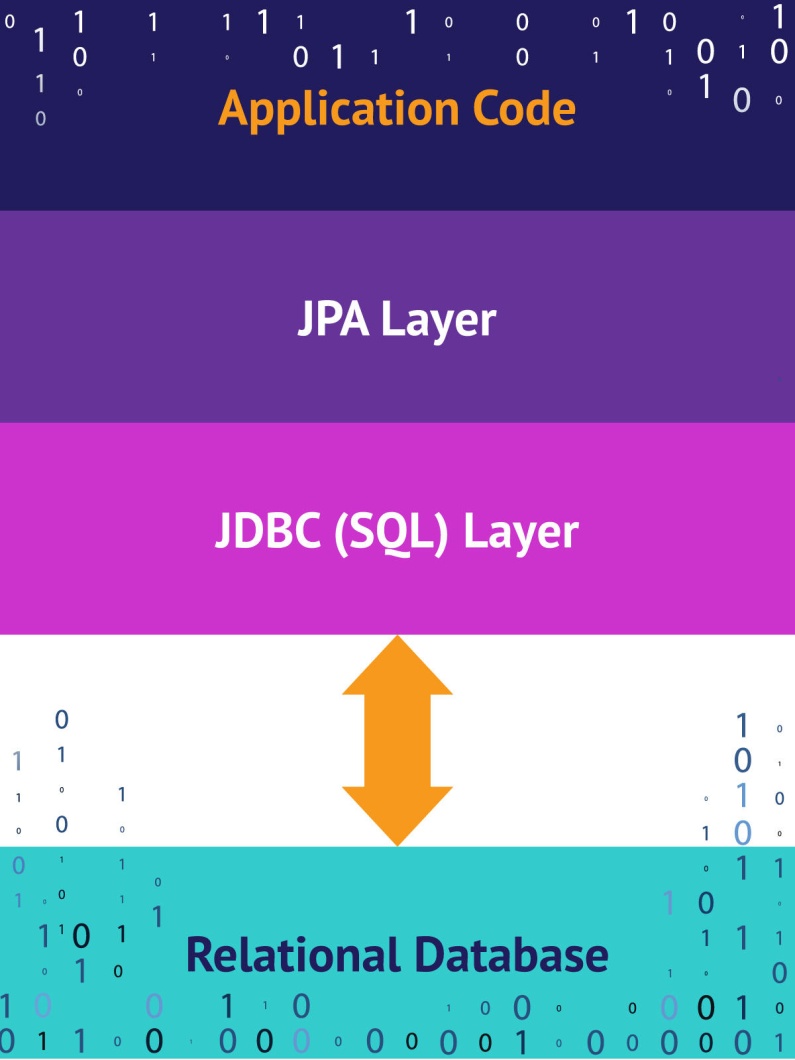


Рисунок 20. Упрощенная схема проекта

Данный проект реализован как Spring Boot проект, с использованием JPA технологии. В данном проекте используется реализация JPA – Hibernate.

**JPA** – это технология, обеспечивающая объектно-реляционное отображение простых JAVA объектов и предоставляющая API для сохранения, получения и управления такими объектами.

Серверную часть сервиса можно разделить на следующие модули:

Sql-скрипты для инициализации базы данных, а так же система контроля версии БД.

Доменные сущности для реализации представления таблиц по технологии JPA(hibertane).

DAO -репозитории для реализации API управления сущностями.

Rest-контроллеры, отвечаю за реализацию rest-запросов к сервису, а так же хранят в себе бизнес-логику.

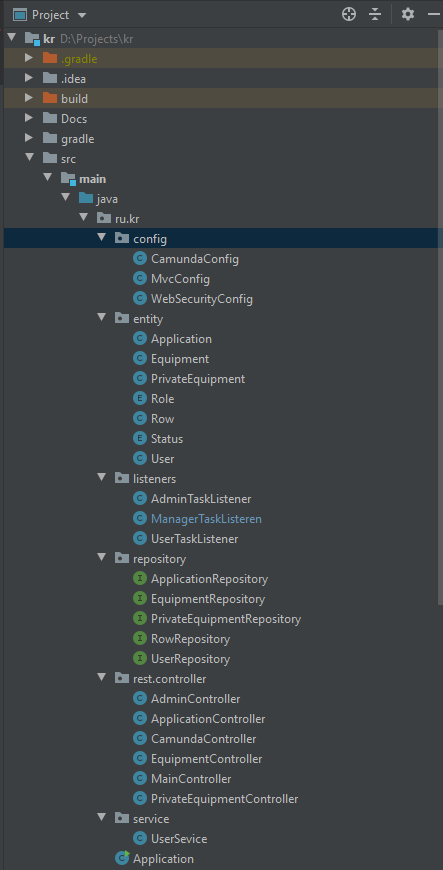


Рисунок 21. Структура проекта

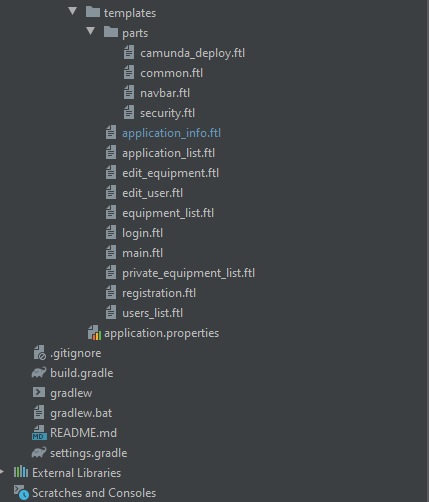


Рисунок 22.Структура проекта

**Sql-скрипты для инициализации базы данных, а так же система контроля версии БД.**

В папке resources/db лежит все необходимое для правильной работоспособности Liquibase, а так же корректного наката скриптов.

V0001\_INIT\_DATABASE.sql – файл инициализации БД.

changelog-master.xml – файл Liquibase с перечнем sql-скриптов для наката.

В реализация данного модуля в большей степени задействован пакет Spring Data JPA.

Spring Data — дополнительный удобный механизм для взаимодействия с сущностями базы данных, организации их в репозитории, извлечение данных, изменение, в каких то случаях для этого будет достаточно объявить интерфейс и метод в нем, без имплементации.

**Доменные сущности для реализации представления таблиц по технологии JPA и DAO -репозитории для реализации API управления сущностями.**

Все сущности находятся в пакете ru.kr.entity. Репозитории же лежат в ru.kr.repository.

При проектировании информационной системы выявляются некоторые слои, которые отвечают за взаимодействие различных модулей системы. Соединение с базой данных является одной из важнейшей составляющей приложения. Всегда выделяется часть кода, модуль, отвечающающий за передачу запросов в БД и обработку полученных от неё ответов. В общем случае, определение [Data Access Object](https://ru.wikipedia.org/wiki/Data_Access_Object)описывает его как прослойку между БД и системой. **DAO** абстрагирует сущности системы и делает их отображение на БД, определяет общие методы использования соединения, его получение, закрытие и (или) возвращение в **Connection Pool**.

Для создания доменных сущностей используется hibernate annotations.

Аннотации являются мощным иструментом  для предоставления метаданных, а также намного нагляднее при чтении нашего кода другим разработчиком

Обязательными аннотациями являются следующие:

**@Entity**

Эта аннотация указывает Hibernate, что данный класс является сущностью (entity bean). Такой класс должен иметь конструктор по-умолчанию (пустой конструктор).

**@Table**

С помощью этой аннотации мы говорим Hibernate,  с какой именно таблицей необходимо связать (map) данный класс. Аннотация **@Table**имеет различные аттрибуты, с помощью которых мы можем указать *имя таблицы, каталог, БД и уникальность столбцов в таблец БД*.

**@Id**

С помощью аннотации **@Id**мы указываем *первичный ключ (Primary Key)* данного класса.

**@GeneratedValue**

Эта аннотация используется вместе с аннотацией **@Id** и определяет такие паметры, как **strategy** и **generator**.

**@Column**

Аннотация **@Column**определяет к какому столбцу в таблице БД относится конкретное поле класса (аттрибут класса).

Наиболее часто используемые аттрибуты аннотации **@Column**такие:

* **name**Указывает имя столбца в таблице
* **unique**Определяет, должно ли быть данноезначение уникальным
* **nullable**Определяет, может ли данное поле быть NULL, или нет.
* **length**Указывает, какой размер столбца (например колчиство символов, при использовании String).

DAO -репозитории для реализации API управления сущностями реализованы через интерфейс Spring Repository.

Основное понятие в Spring Data — это репозиторий. Это несколько интерфейсов которые используют JPA Entity для взаимодействия с ней. Так например интерфейс  
public interface CrudRepository<T, ID extends Serializable> extends Repository<T, ID>  
обеспечивает основные операции по поиску, сохранения, удалению данных (CRUD операции)

T save(T entity);

Optional findById(ID primaryKey);

void delete(T entity);

и др. операции.  
  
Есть и другие абстракции, например PagingAndSortingRepository.  
Т.е. если того перечня что предоставляет интерфейс достаточно для взаимодействия с сущностью, то можно прямо расширить базовый интерфейс для своей сущности, дополнить его своими методами запросов и выполнять операции. Сейчас я покажу коротко те шаги что нужны для самого простого случая (не отвлекаясь пока на конфигурации, ORM, базу данных).

Запросы к сущности можно строить прямо из имени метода. Для этого используется механизм префиксов find…By, read…By, query…By, count…By, и get…By, далее от префикса метода начинает разбор остальной части. Вводное предложение может содержать дополнительные выражения, например, Distinct. Далее первый By действует как разделитель, чтобы указать начало фактических критериев. Можно определить условия для свойств сущностей и объединить их с помощью And и Or.

В документации определен весь перечень, и правила написания метода. В качестве результата могут быть сущность T, Optional, List, Stream. В среде разработки, например в Idea, есть подсказка для написания методов запросов.

**Rest-контроллеры.**

Все котроллеры расположены в пакете ru.kr.rest.controllers.

Разделены они по принципу - для обработки каждой сущности или модуля свой контроллер.

REST (Representational state transfer) – это стиль архитектуры программного обеспечения для распределенных систем, таких как World Wide Web, который, как правило, используется для построения веб-служб. Термин REST был введен в 2000 году Роем Филдингом, одним из авторов HTTP-протокола. Системы, поддерживающие REST, называются RESTful-системами.  
В общем случае REST является очень простым интерфейсом управления информацией без использования каких-то дополнительных внутренних прослоек. Каждая единица информации однозначно определяется глобальным идентификатором, таким как URL. Каждая URL в свою очередь имеет строго заданный формат.

REST быстро стал де-факто стандартом для создания web-сервисов, поскольку таким они легки для создания и обработки.

Существует большое обсуждение о том, как REST вписывается в мир микросервисов, но в этом уроке просто описывается создание RESTful сервисов.

Почему REST? Из [REST на практике](https://www.amazon.com/gp/product/0596805829?ie=UTF8&tag=martinfowlerc-20&linkCode=as2&camp=1789&creative=9325&creativeASIN=0596805829), заимствуется фраза [Martin Flower](https://martinfowler.com/" \t "_blank): "Понятие того, что web является доказательством очень масштабируемой распределенной системы, которая работает действительно хорошо и мы можем брать идеи, исходя из того, чтобы создавать системы более легко." Я думаю, что это довольно веская причина: REST охватывает указания самого web, его архитектуры и его приемуществ.

Какие преимущества? Главным образом все те, какие несет в себе HTTP как платформа сама по себе. Безопасность приложений(шифрование и аутентификация), для которой известно некоторое количество решений. Кеширование встроено в протокол. Маршрутизация, через DNS, гибкая и хорошо известная система уже поддерживается повсеместно.

REST, несмотря на повсеместность использования, не является стандартом, как таковой, а подходом, стилем, ограниечением HTTP протокола. Его реализация может различаться в стиле, подходе. Для клиента такого API это может быть разочарованием. Качество REST сервисов дико варьируется.

Dr. Leonard Richardson собрал воедино модель, которая объясняет различные уровни соответствующих понятий REST и сортирует их. Она описывает 4 уровня, начиная с **level 0**. Martin Fowler [очень хорошо написал про модель](https://martinfowler.com/articles/richardsonMaturityModel.html)

**Level 0**: Swamp of POX - это уровень, где мы просто используем HTTP как транспорт. Вы можете вызвать SOAP технологию. Она использует HTTP, но как транспорт. Стоит также отметить, что вы можете также использовать SOAP [поверх чего-то подобного JMS](https://www.w3.org/TR/soapjms/) без HTTP. SOAP, соответветственно не является RESTful. Он всего лишь HTTP-aware.

**Level 1**: Resources - на этом уровне сервисы могут использовать HTTP URI для отличия между сущностями в системе. К примеру, вы можете направить запросы на /customers, /users и т.д. XML-RPC является примером **Level 1** технологии: он использует HTTP и он может использовать URI для различия точек выхода. В конечном счете, несмотря на то, что XML-RPC не является RESTful, он использует HTTP как транспорт для чего-нибудь ещё(удаленный вызов процедур).

**Level 2**: HTTP Verbs - это уровень, на котором вы хотите быть. Если вы делаете все неправильно с Spring MVC, то вы скорее всего все ещё остановитесь тут. На этом уровне сервисы используют преимущества нативных HTTP возможностей, таким как заголовки, коды статуса, определенные URI и другие. Отсюда мы начнем наше путешествие.

**Level 3**: Hypermedia Controls - это заключительный уровень, к которому мы стремимся. Гипермедиа как практическое применение [HATEOAS](https://en.wikipedia.org/wiki/HATEOAS) ("HATEOAS" является сокращением от "Hypermedia as the Engine of Application State") шаблона проектирования. Гипермедиа продлевает жизнь сервису, отделяя клиента сервиса от необходимости глубокого знания платформы и топологии сервиса. Она описывает REST сервисы. Сервис может ответить на вопросо о том, какой выл вызов и когда. Мы рассмотрим более глубже в дальнейшем.

Для маршрутизации запросов в Spring Framework используется аннотация **@RequestMapping** с указанием HTTP-метода при помощи свойства method или более простые аннотации вроде  
**@GetMapping**, **@PostMapping**, **@DeleteMapping**и т.д.

**Система авторизации и аутентификации.**

В данном проекте, слоем авторизации и аутентификации, был выбран Spring Sequrity.



Рисунок 23. Логотип Spring Security

Spring Security это Java/JavaEE framework, предоставляющий механизмы построения систем аутентификации и авторизации, а также другие возможности обеспечения безопасности для корпоративных приложений, созданных с помощью Spring Framework. Проект был начат Беном Алексом (Ben Alex) в конце 2003 года под именем «Acegi Security», первый релиз вышел в 2004 году. Впоследствии проект был поглощён Spring'ом и стал его официальным дочерним проектом. Впервые публично представлен под новым именем Spring Security 2.0.0 в апреле 2008 года.

Ключевые объекты контекста Spring Security:

* SecurityContextHolder, в нем содержится информация о текущем контексте безопасности приложения, который включает в себя подробную информацию о пользователе(Principal) работающем в настоящее время с приложением. По умолчанию SecurityContextHolder используетThreadLocal для хранения такой информации, что означает, что контекст безопасности всегда доступен для методов исполняющихся в том же самом потоке. Для того что бы изменить стратегию хранения этой информации можно воспользоваться статическим методом класса SecurityContextHolder.setStrategyName(String strategy). Более подробно [SecurityContextHolder](http://docs.spring.io/autorepo/docs/spring-security/3.0.x/apidocs/org/springframework/security/core/context/SecurityContextHolder.html)
* SecurityContext, содержит объект Authentication и в случае необходимости информацию системы безопасности, связанную с запросом от пользователя.
* Authentication представляет пользователя (Principal) с точки зрения Spring Security.
* GrantedAuthority отражает разрешения выданные пользователю в масштабе всего приложения, такие разрешения (как правило называются «роли»), например ROLE\_ANONYMOUS, ROLE\_USER, ROLE\_ADMIN.
* UserDetails предоставляет необходимую информацию для построения объекта Authentication из DAO объектов приложения или других источников данных системы безопасности. Объект UserDetailsсодержит имя пользователя, пароль, флаги: isAccountNonExpired, isAccountNonLocked, isCredentialsNonExpired, isEnabled и Collection — прав (ролей) пользователя.
* UserDetailsService, используется чтобы создать UserDetails объект путем реализации единственного метода этого интерфейса

Так же была написана кастомная сущность User, для использования представления пользователя в Spring Security. В данной сущности были добавлены новые поля. Одним из них является поле роли, для разграничения доступа к ресурсам системы.

**BPMN-процесс и UserTask-листенеры для работы и интеграции с движком Camunda.**

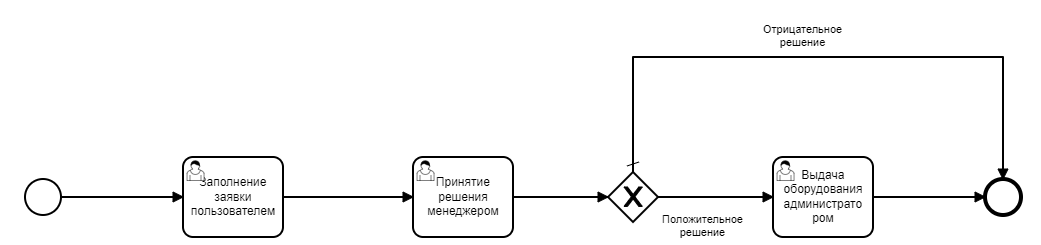


Рисунок 24. Схема BPMN-процесса системы

Данный процесс был разработан в редакторе BPMN процессов Camunda Modeler. В данном случае, используется имплементация задач – User Task.

За каждой задачей закреплен свой обработчик, написанный на Java. Так называемые «Листенеры».

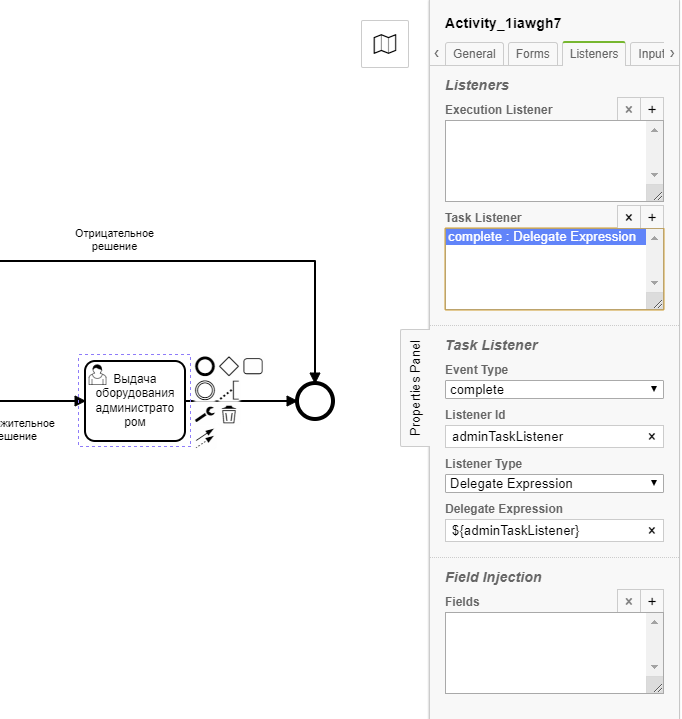


Рисунок 25. Пример указания листенера.

На рисунке можно увидеть пример указания листенера для задачи.

Листенер используется для выполнения пользовательской логики на Java или выражения при возникновении определенного события, связанного с задачей. Его можно добавить только в определение процесса как дочерний элемент пользовательской задачи.

Листенеры имею широкий набор событий, связанных с задачей:

* Создание задачи
* Назначение задачи на пользователя
* Выполнение задачи
* Удаление задачи

В данном случае используется событие, которое происходит при выполнении задачи.

Сам же листенер выглядит следующим образом:

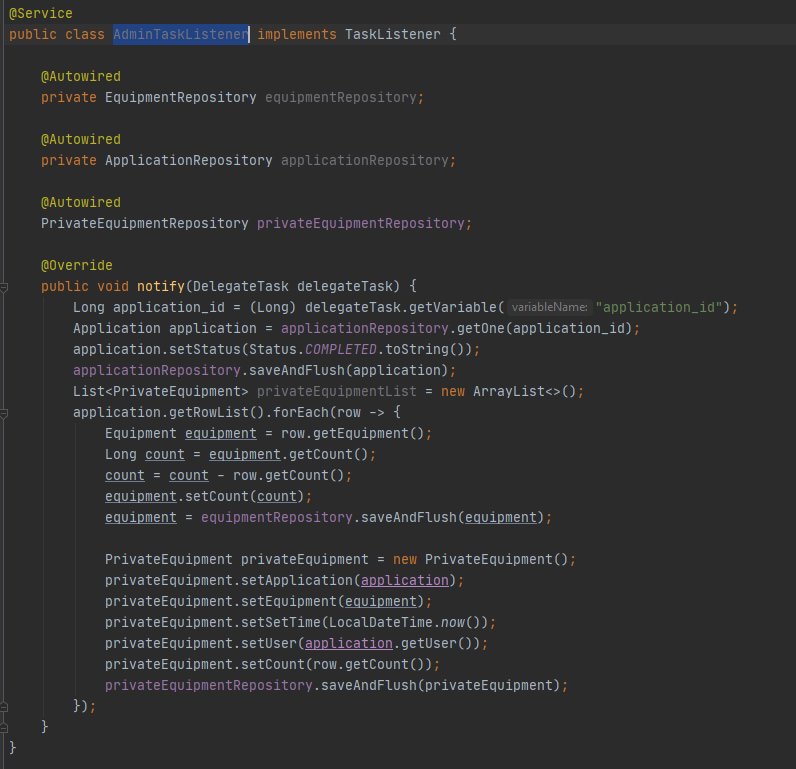


Рисунок 26. Скриншот листенера AdminTaskListener

Это класс, имплементирующий интерфейс TaskListener.

Данный класс всегда должен реализовывать метод notify.

В него уже и помещается нужная разработчику логика. В данной имплементации проиходят преобразования данных по заявке.

## Реализация Front-End части.

В реализации ui учавствуют следующие элементы:

* Freemarker – шаблонизатор, для наполнения html-заготовок данными.
* Bootstrap – подключаемая библиотека с готовым набором решений для реализации пользовательского интерфейса.
* HTML-файлы, заготовки с разметкой freemarker и подключенной библиотекой Bootstrap .

Все файлы шаблонов лежат в папке resources/templates.

**Страница главного экрана сервиса**

Файл main.ftl

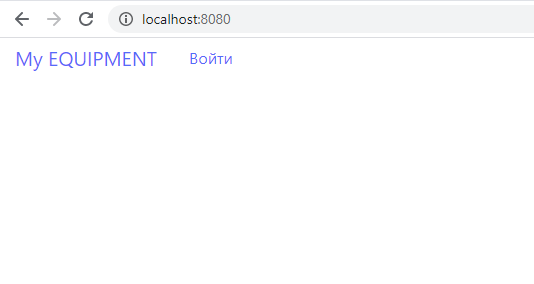


Рисунок 27. Главная страница системы

**Страница входа в систему**

Файл login.ftl

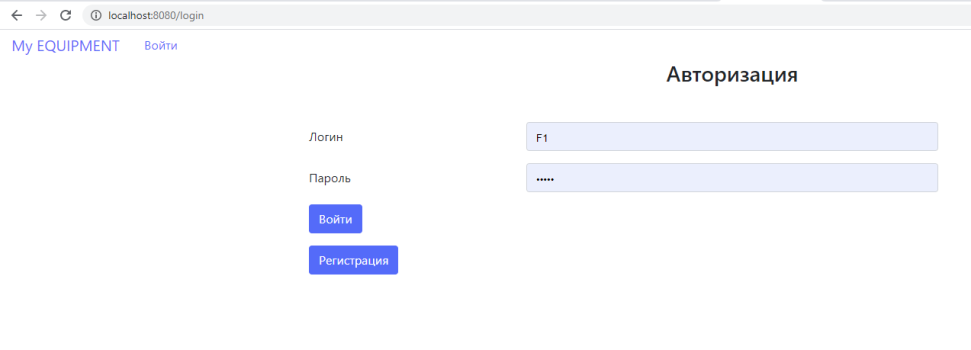


Рисунок 28. Страница входа в систему

**Страница регистрации**

Файл registration.ftl

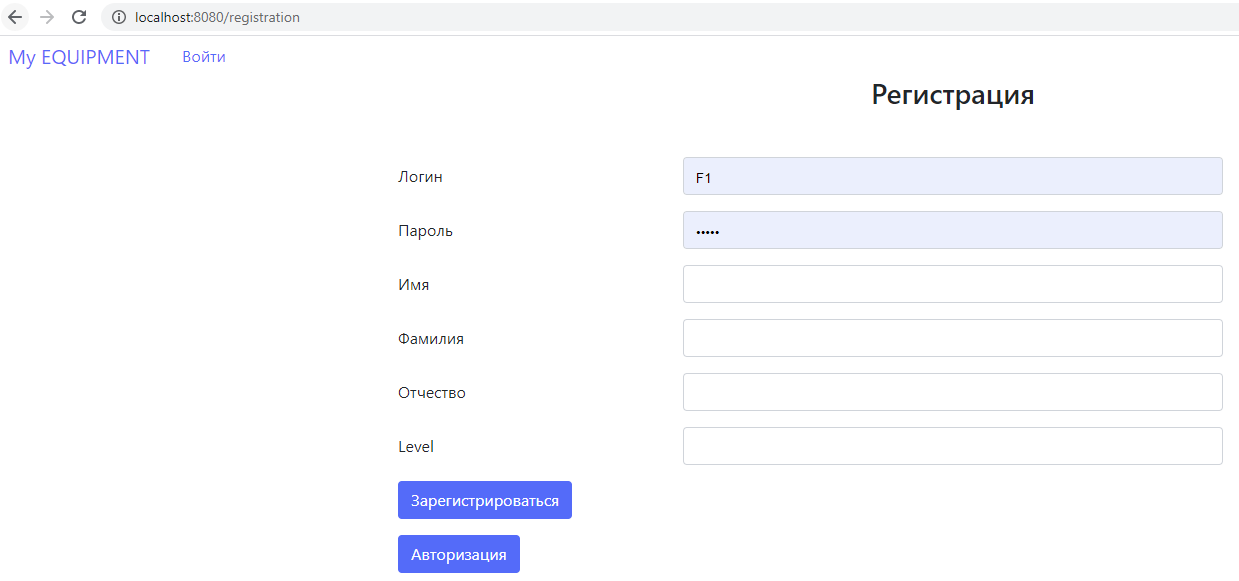


Рисунок 29. Страница Регистрации

**Верхняя панель навигации**

Файл navbar.ftl

Реализован переход на главную страницу, просмотра списка домов, просмотра списка квартир, просмотра списка комнат.

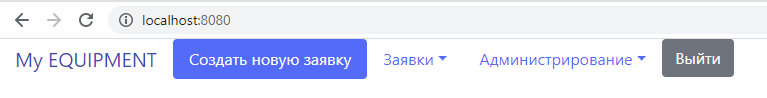


Рисунок 30. Панель навигации

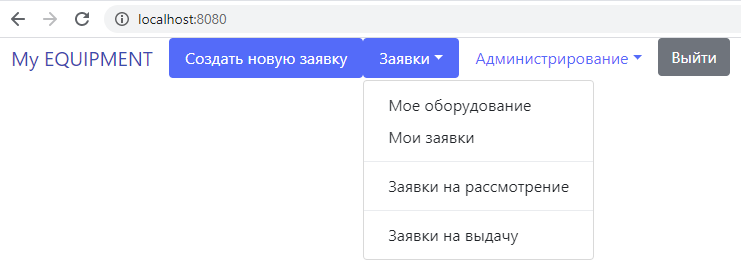


Рисунок 31. Панель навигации. Заявки

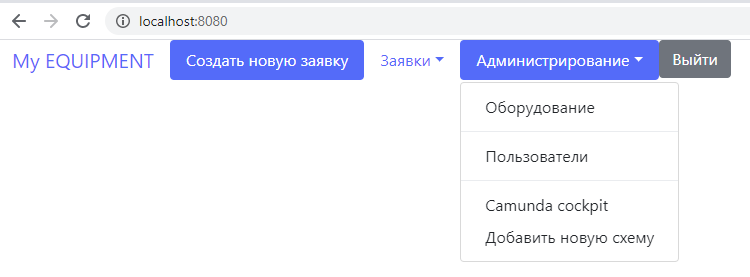


Рисунок 32. Панель навигации. Администрирование

В данной панели реализован функционал разграничения доступа для разных ролей.

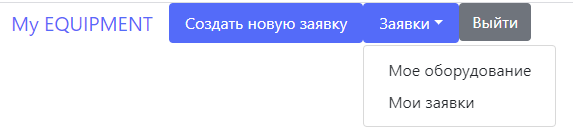


Рисунок 33. Панель навигации. Обычный пользователь

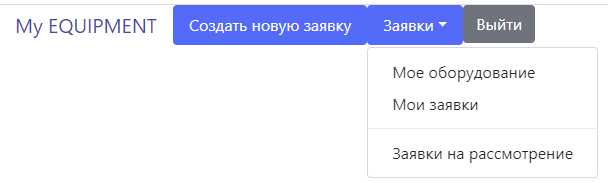


Рисунок 34. Панель навигации. Менеджер

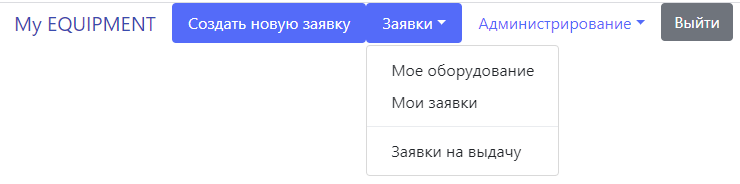


Рисунок 35. Навигационная панель. Администратор

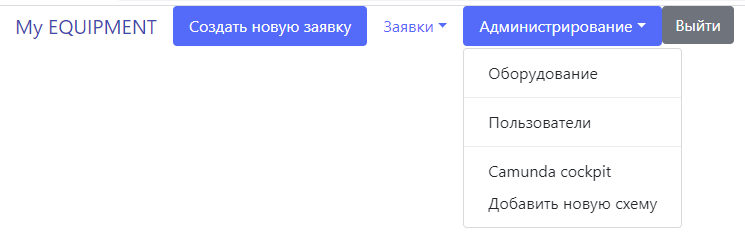


Рисунок 36. Навигационная панель. Администратор

**Страница списка пользователей**

Файл users\_list.ftl

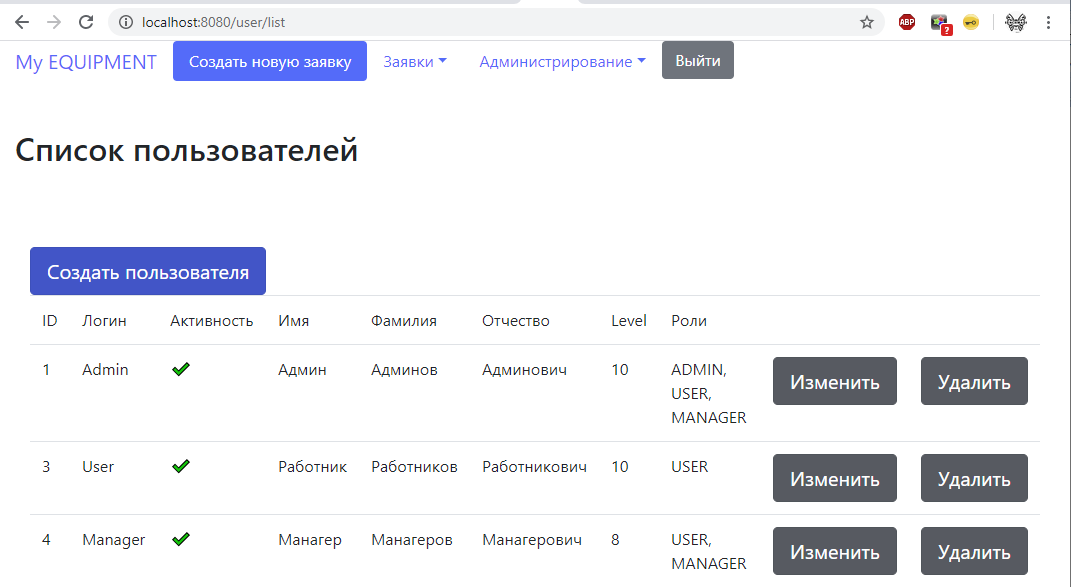


Рисунок 37. Страница списка пользователей

**Страница редактирования пользователя**

Файл edit\_user.ftl

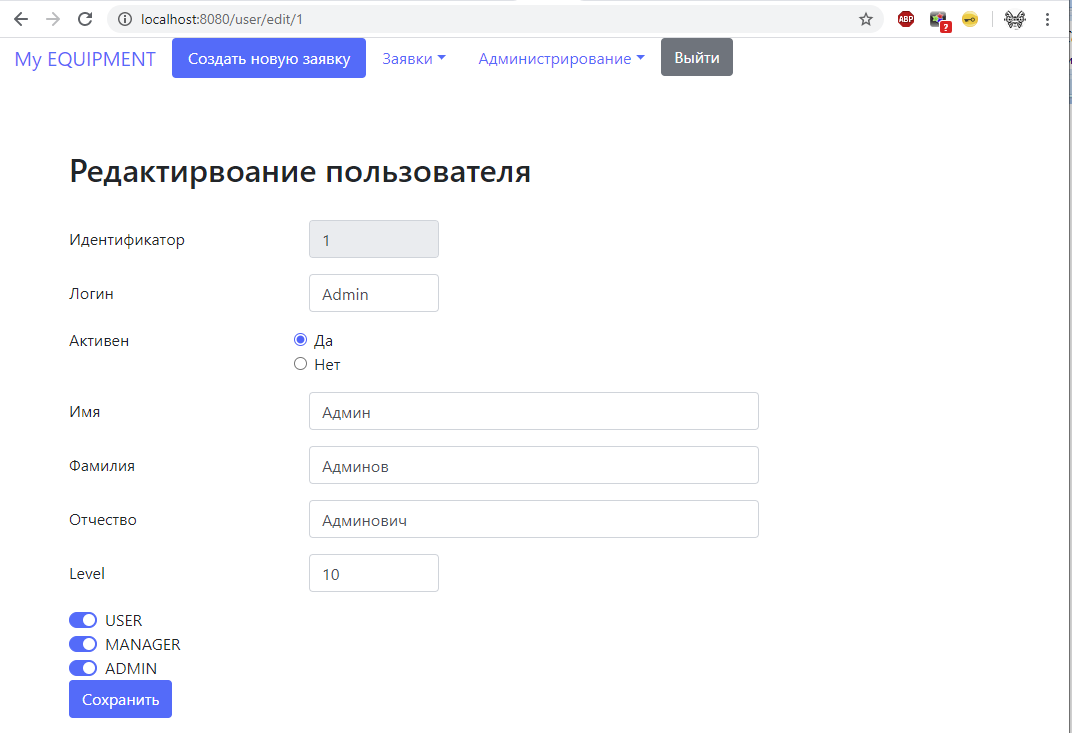


Рисунок 38. Страница редактирования пользователя

**Страница списка оборудования**

Файл equipment\_list.ftl

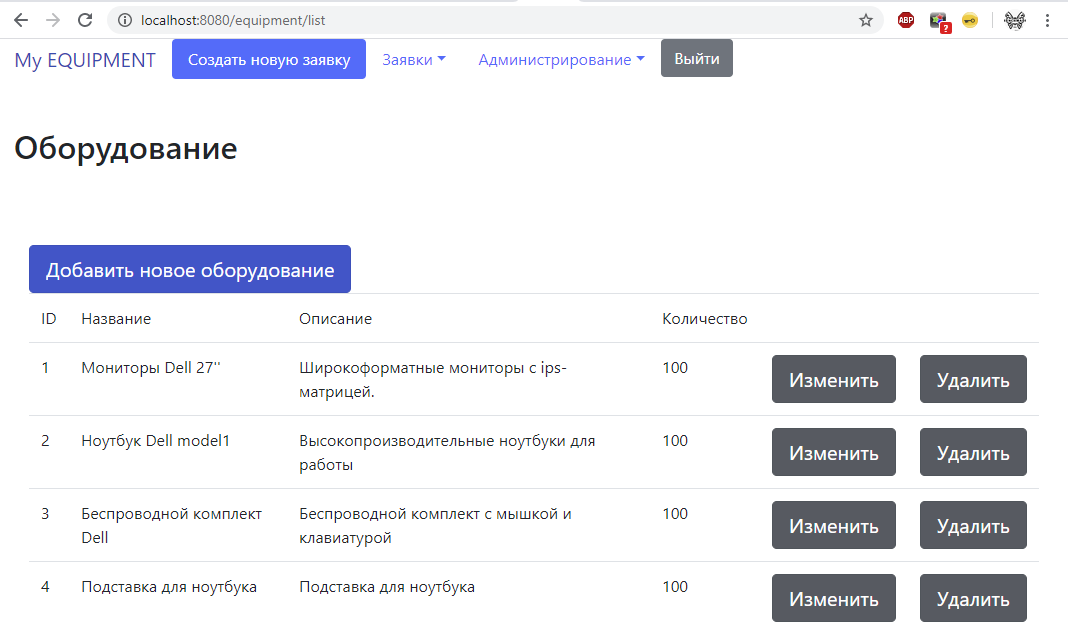


Рисунок 39. Страница списка оборудования

**Страница добавления или редактирования оборудования**

Файл edit\_equipment.ftl

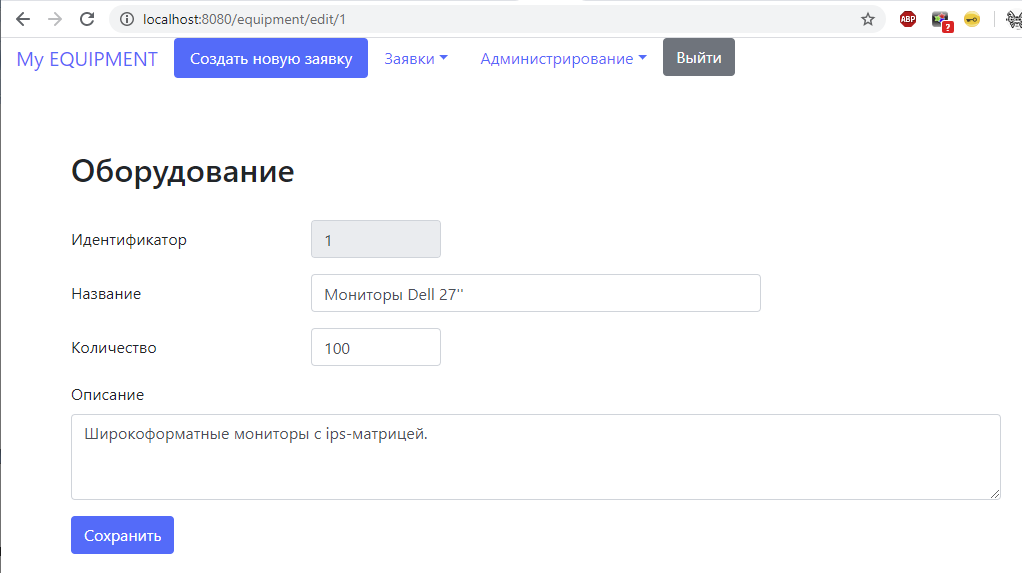


Рисунок 40. Редактирования информации об оборудовании

**Страница загрузки схем BPMN-процессов в движок Camunda**

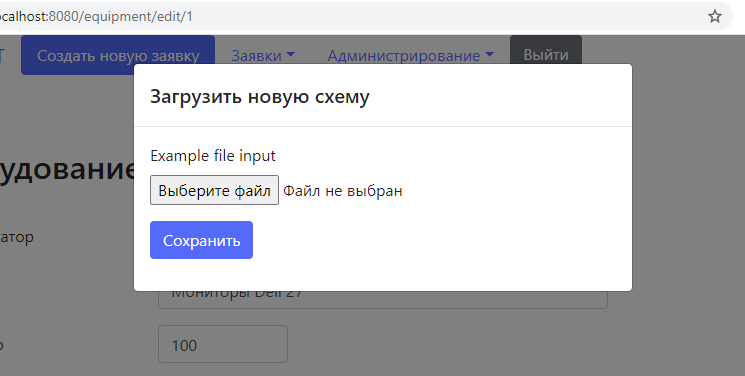


Рисунок 41. Загрузка bpmn-схемы

**Страница создания и редактирования заявки**

Файл application\_info.ftl

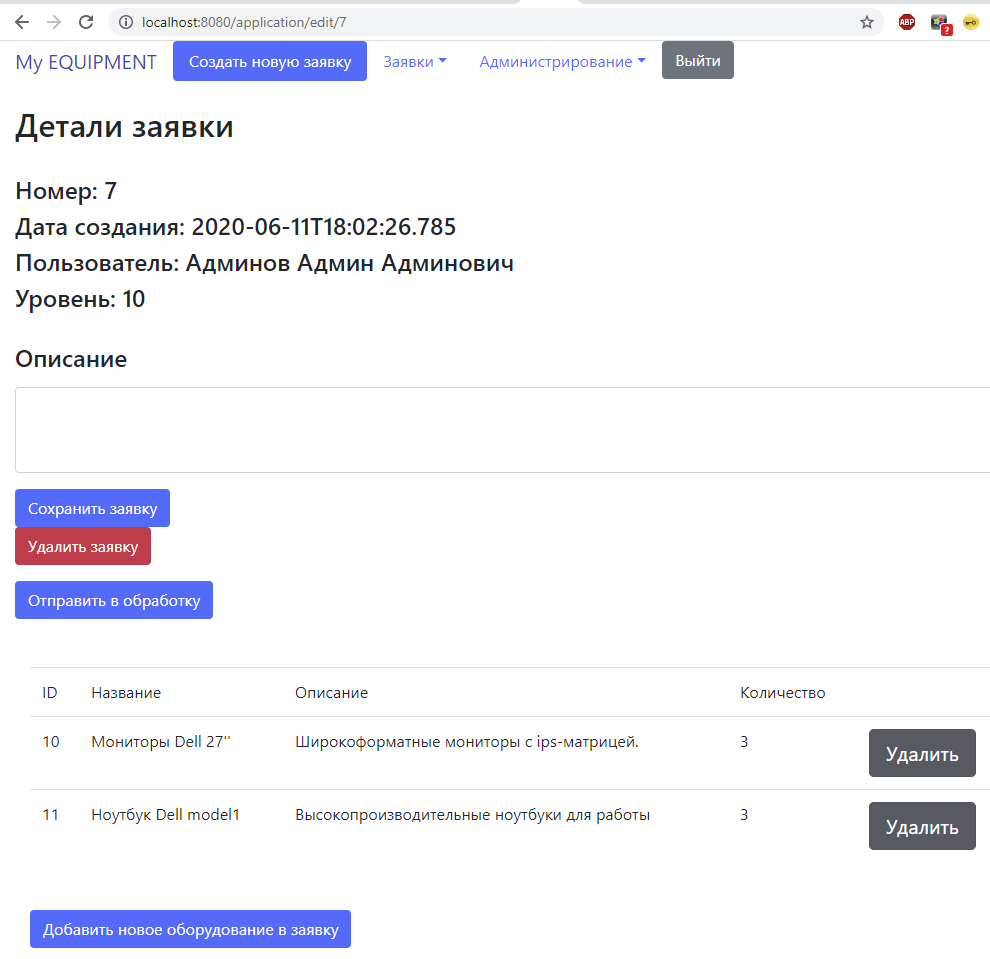


Рисунок 42. Создание и редактирование заявк

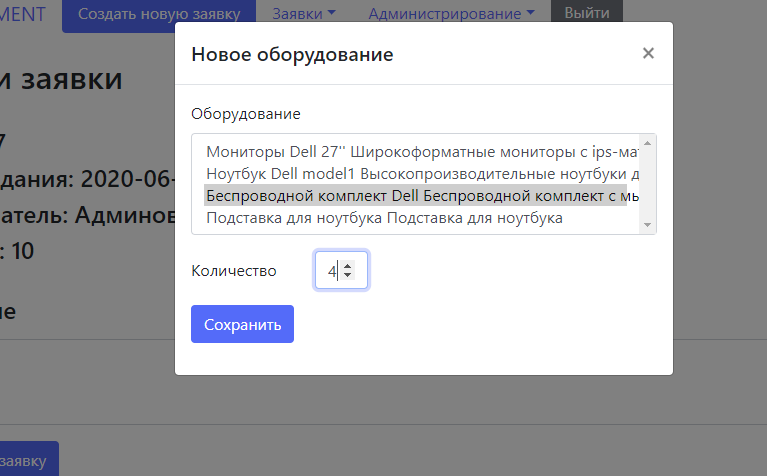


Рисунок 43. Модальное окно добавления оборудования в заявку

**Страница списка заявок (Мои заявки / Заявки на рассмотрении / Заявки на выдачу)**

Файл application\_list.ftl

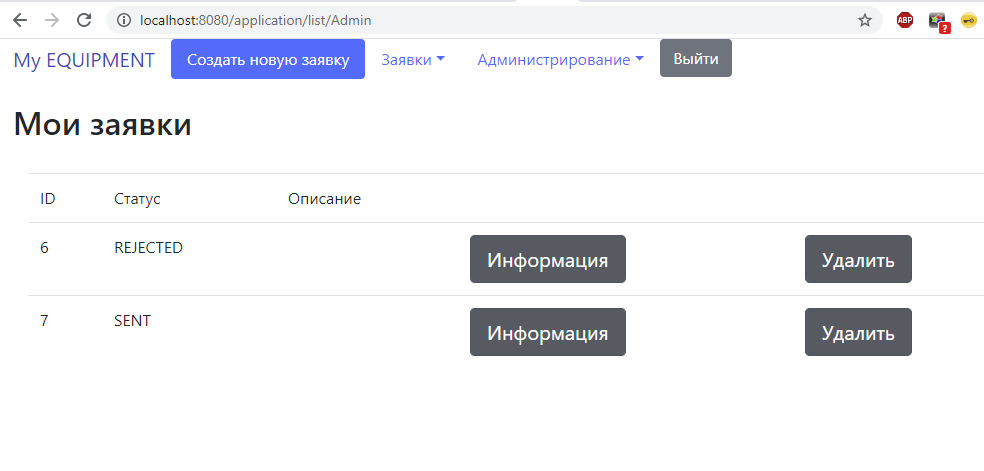


Рисунок 44. Список заявок

**Страница рассмотрения заявки менеджером**

Файл application\_info.ftl

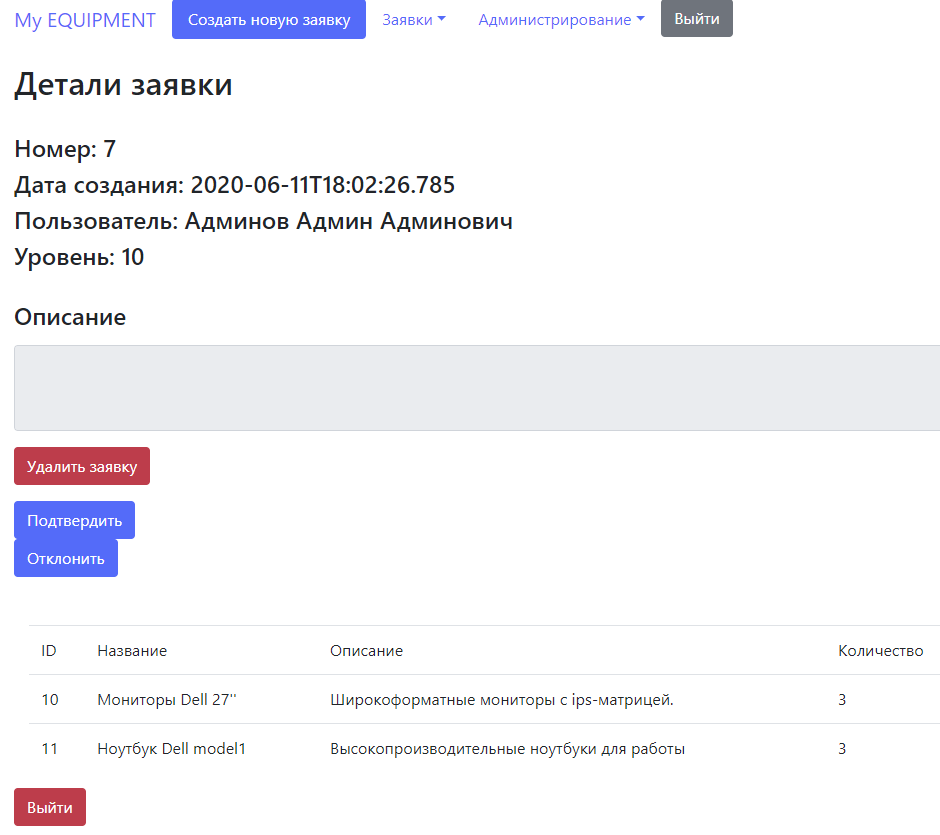


Рисунок 45. Страница рассмотрения заявки менеджером

**Страница рассмотрения заявки администратором**

Файл application\_info.ftl

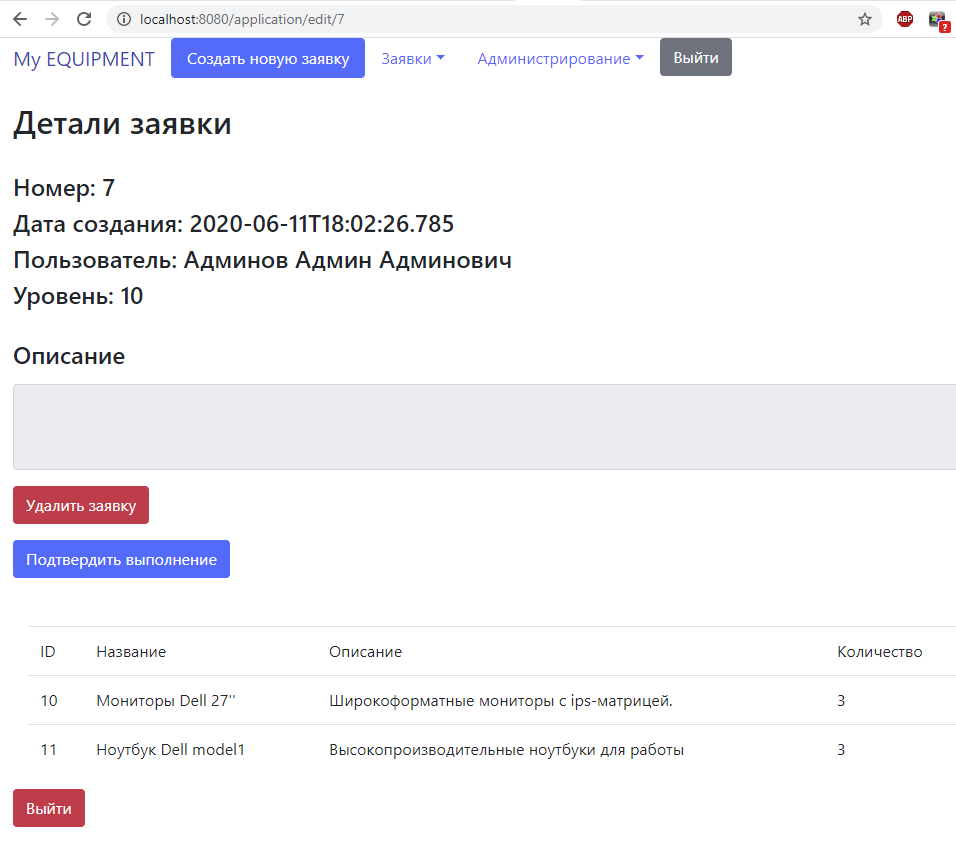


Рисунок 46. Страница рассмотрения заявки администратором

**Страница со списком оборудования, закрепленным за работником**

Файл equipment\_list.ftl

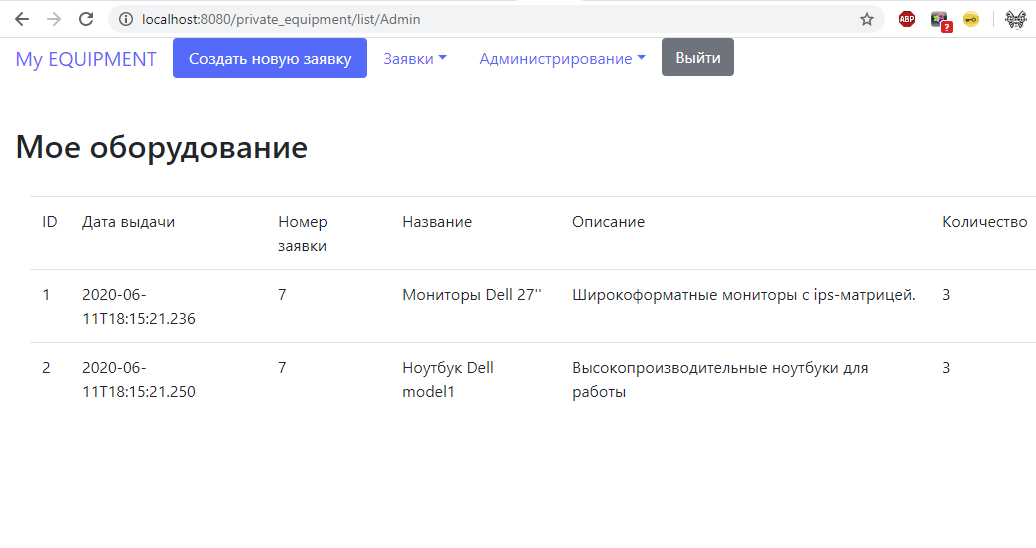


Рисунок 47. Список оборудования, закрепленного за сотрудником

**Camunda Tool**

Так же в составе основного пакета библиотек Camunda Engine, поставляется пакет модулей по администрированию:

* Camunda Cockpit
* TaskList
* Admin

В данном случае особый интерес вызывает модуль Camunda Cockpit.

Данный модуль так же имеет сервер-клиентскую архитектуру.

Попасть в Camunda Cockpit можно через панель навигации в разделе администрирования. Данный модуть так же имеет отдельную систему аутентификации, но для доступа к этому ресурсу все равно потребуется общая аутентификация пользователей.

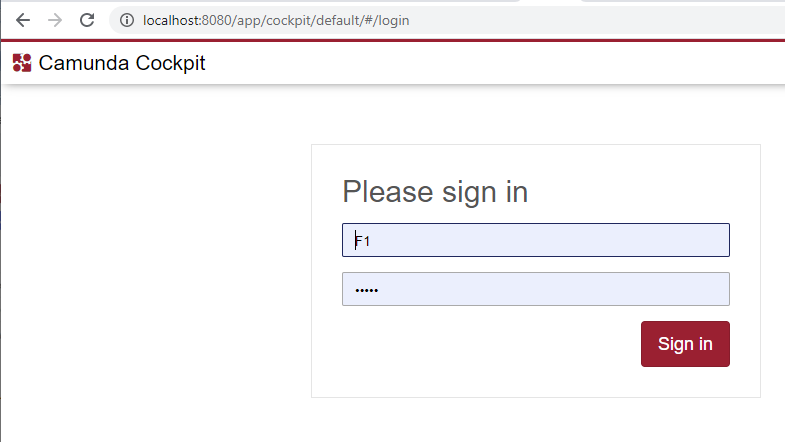


Рисунок 48. Форма авторизации Camunda Cockpit

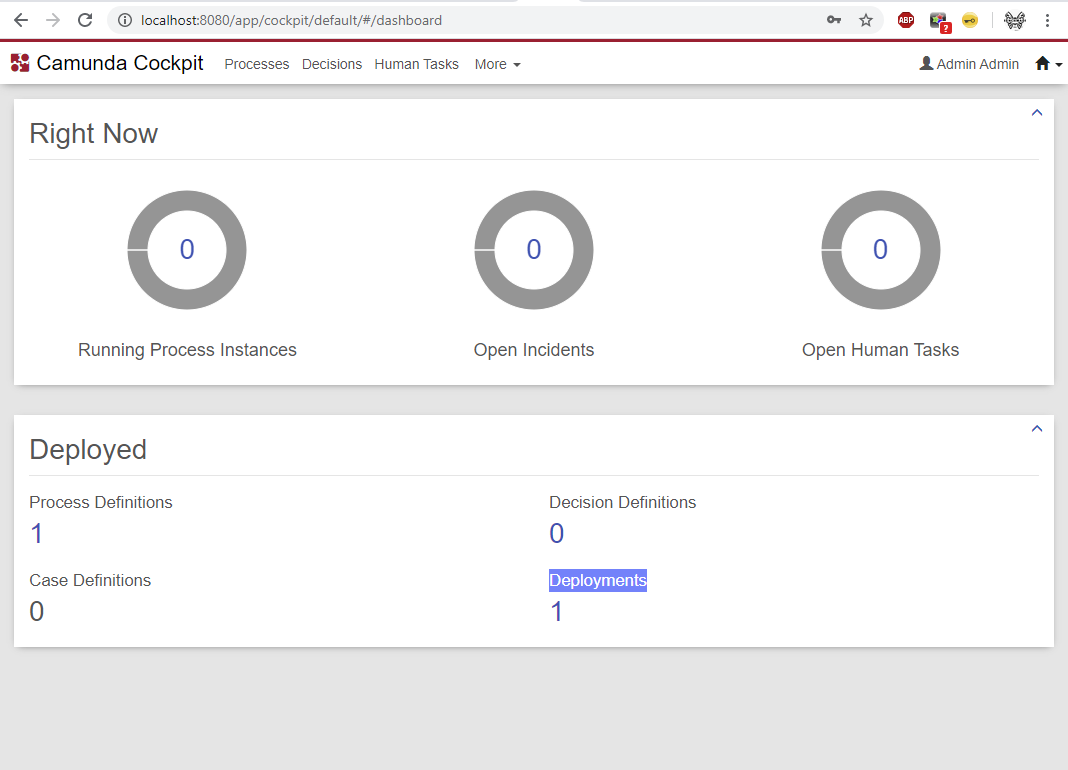


Рисунок 49. Главный экран модуля Cockpit

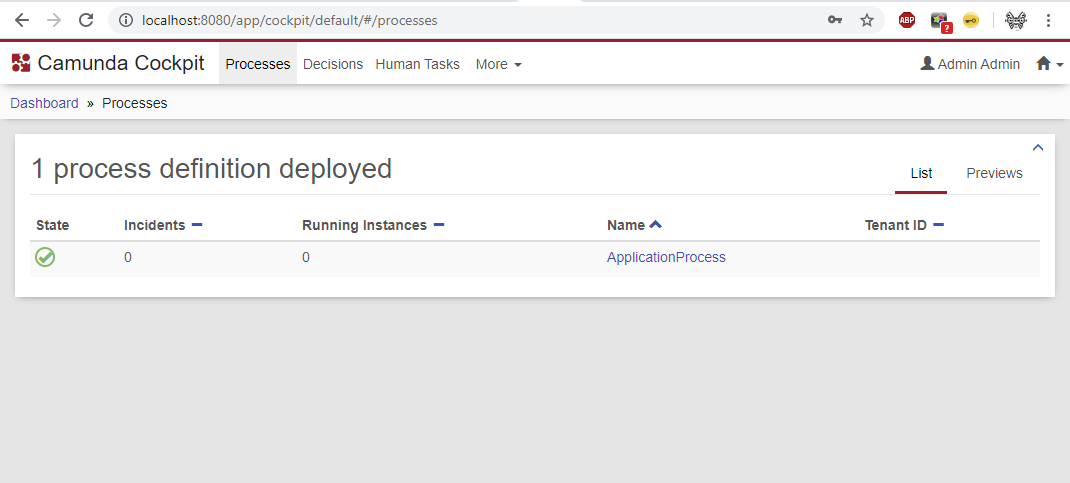


Рисунок 50. Экран списка всех существующих процессов

На вкладке процессов, мы можем увидет все загруженные процесс и количество из запущенных имплементаций

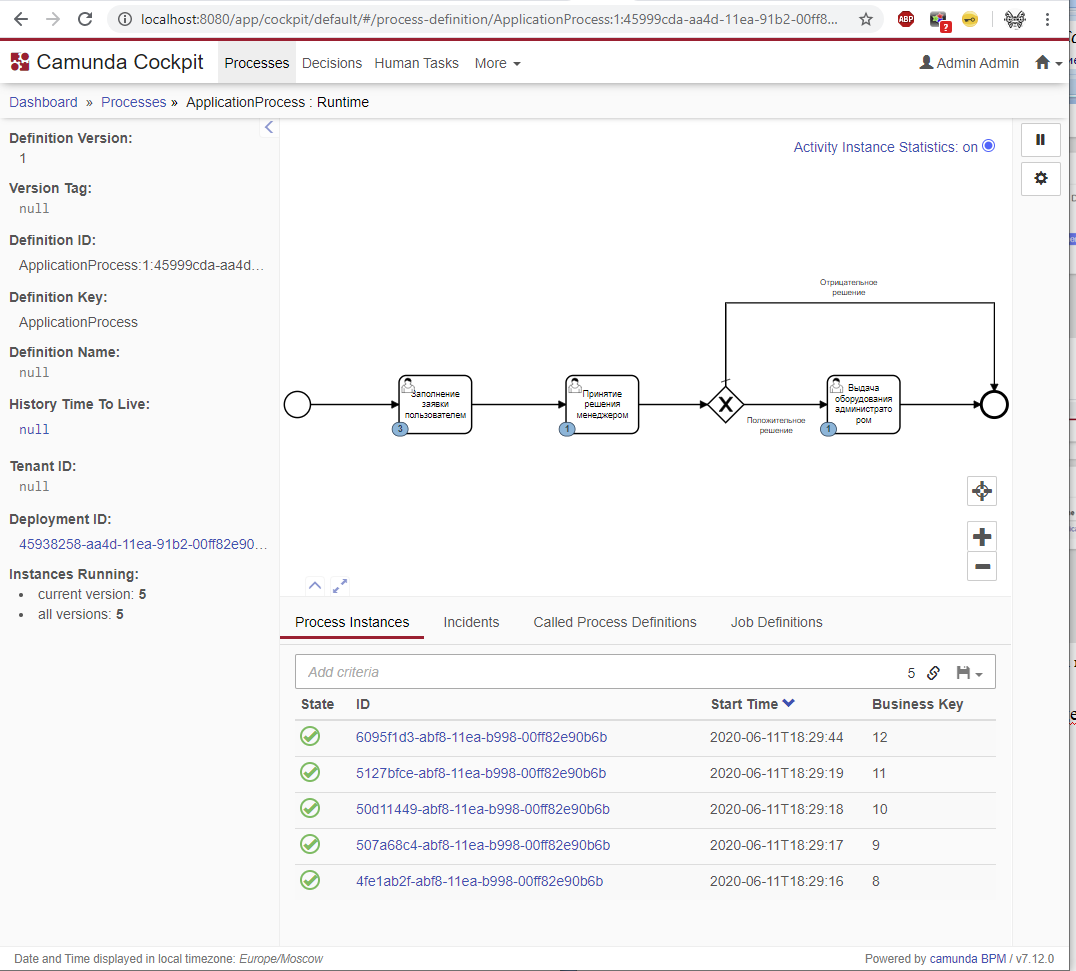


Рисунок 51. Информация о процессе

При открытии интересующего нас процесса, мы можем детально увидеть информацию о самом процессе, всех его запущенных имплементациях а так же о активных задачах.

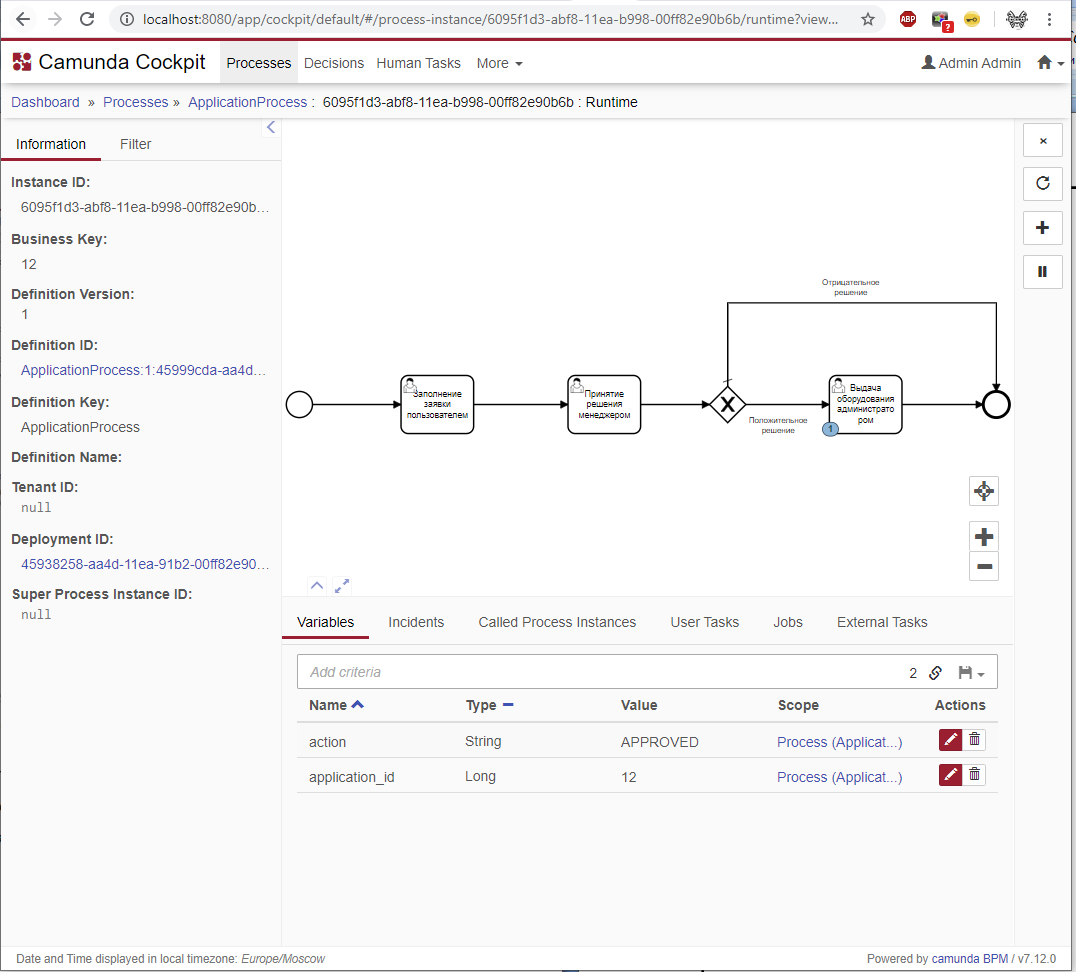


Рисунок 52. Детали имплементации процесса

Так же, если выбрать интересующий наш инстанс процесса, можно увидеть актуальный набор переменных. Информацию о статусе процесса и на какой задаче он сейчас находится.

Изм.

Лист

№ докум.

Подпись

Дата

Лист

24

Заключение

# Заключение

В ходе выполнения данной курсовой работы была разработана информационная система «Автоматизация деятельности по учету и согласованию выдачи оборудования сотрудникам для ООО "АКСЕНЧЕР", г. Тверь», которая обеспечивает информационную поддержку деятельности системных администраторов в компании ООО «АКСЕНЧЕР».

Преимущество использования автоматизированной системы состоит в том, что затрачивается меньше времени на обработку заявок от сотрудников на получение или замену оборудования. Автоматизируется механизм согласования нового оборудования и закуп. Появляетсяудобная система генерации отчетов. Так же имеется возможно гибкой настройки бизнеспроцесса согласования и выдачи оборудования.

Изм.

Лист

№ докум.

Подпись

Дата

Лист

26

Источники

# Источники

1. <https://www.atlassian.com/ru/software/jira> - Официальный сайт Atlassian

2. <https://www.easyprojects.net/ru/> - официальный сайт Easy Projects .NET

3. <http://www.paydox.ru/Help/Documentation.asp?l=ru> – официальный сайт PayDox

4. <https://www.trinion.org/blog/chto-takoe-sistemy-upravleniya-proektami-prednaznachenie-i-primenenie> - статья про системы управления проектами