|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **Компьютерные системы и сети**

**ОТЧЕТ ПО ПРЕДДИПЛОМНОЙ ПРАКТИКЕ**

Студент Большаков Павел Андреевич

*фамилия, имя, отчество*

Группа ИУ6-82

Тип практики Производственная

Название предприятия ЗАО «КРОК Инкорпорейтед»

Студент **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Большаков П. А.**

*подпись, дата фамилия, и.о.*

Руководитель практики **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Фомин М. М.**

*подпись, дата фамилия, и.о.*

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. *г.*

**1.Введение**

Цель практики: исследование технологий разработки промышленных систем на языке Java.

**2.Теоретическая часть**

Современная разработка на языке Java в корпоративных масштабах использует ряд прилегающих к языку Java и необходимых для полноценного процесса разработки технологий, фреймворков языков программирования необходимых для разработки клиентской части приложения,возаимодействия с БД и решения сторонних задач системы.

В компании КРОК существует устоявшийся стек технологий при разработке приложений корпоративного масштаба, в который входят также технологии созданные внутри компании, такие как фреймворки JXFW и WEBCLIENТ.

Современные приложения корпоративного масштаба в большинстве случаев предполагают моделирование бизнес-процессов и реализацию бизнес-логики, которые также требуют использования технологий позволяющих внедрение данных компонентов в структуру приложение и последующие контроль и мониторинг их исполнения. Одной из таких технологий является CAMUNDA BPMN.

**2.1. Spring Framework**

**Spring Framework** (или коротко **Spring**) — универсальный [фреймворк](https://ru.wikipedia.org/wiki/%D0%A4%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA) [с открытым исходным кодом](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BA%D1%80%D1%8B%D1%82%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) для [Java](https://ru.wikipedia.org/wiki/Java)-платформы. Также существует [форк](https://ru.wikipedia.org/wiki/%D0%A4%D0%BE%D1%80%D0%BA) для платформы [.NET Framework](https://ru.wikipedia.org/wiki/.NET_Framework), названный Spring.NET[[2]](https://ru.wikipedia.org/wiki/Spring_Framework#cite_note-2).

Первая версия была написана [Родом Джонсоном](https://ru.wikipedia.org/w/index.php?title=%D0%94%D0%B6%D0%BE%D0%BD%D1%81%D0%BE%D0%BD,_%D0%A0%D0%BE%D0%B4&action=edit&redlink=1), который впервые опубликовал её вместе с изданием своей книги «Expert One-on-One Java EE Design and Development»[[3]](https://ru.wikipedia.org/wiki/Spring_Framework#cite_note-3) ([Wrox Press](https://ru.wikipedia.org/wiki/Wrox_Press), [октябрь](https://ru.wikipedia.org/wiki/%D0%9E%D0%BA%D1%82%D1%8F%D0%B1%D1%80%D1%8C) [2002 года](https://ru.wikipedia.org/wiki/2002_%D0%B3%D0%BE%D0%B4)).

Фреймворк был впервые выпущен под лицензией [Apache 2.0 license](https://ru.wikipedia.org/wiki/Apache_License) в июне 2003 года. Первая стабильная версия 1.0 была выпущена в марте [2004](https://ru.wikipedia.org/wiki/2004). Spring 2.0 был выпущен в октябре 2006, Spring 2.5 — в ноябре 2007, Spring 3.0 в декабре 2009, и Spring 3.1 в декабре 2011. Текущая версия — 5.1.2.

Несмотря на то, что Spring не обеспечивал какую-либо конкретную модель программирования, он стал широко распространённым в Java-сообществе главным образом как альтернатива и замена модели [Enterprise JavaBeans](https://ru.wikipedia.org/wiki/EJB). Spring предоставляет бо́льшую свободу Java-разработчикам в проектировании; кроме того, он предоставляет хорошо документированные и лёгкие в использовании средства решения проблем, возникающих при создании приложений корпоративного масштаба.

Между тем, особенности ядра Spring применимы в любом Java-приложении, и существует множество расширений и усовершенствований для построения [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) на [Java Enterprise платформе](https://ru.wikipedia.org/wiki/J2EE). По этим причинам Spring приобрёл большую популярность и признаётся разработчиками как стратегически важный фреймворк.

**Spring** обеспечивает решения многих задач, с которыми сталкиваются Java-разработчики и организации, которые хотят создать информационную систему, основанную на платформе [Java](https://ru.wikipedia.org/wiki/Java). Из-за широкой функциональности трудно определить наиболее значимые структурные элементы, из которых он состоит. **Spring** не всецело связан с платформой [Java Enterprise](https://ru.wikipedia.org/wiki/J2EE), несмотря на его масштабную интеграцию с ней, что является важной причиной его популярности.

**Spring**, вероятно, наиболее известен как источник расширений (features), нужных для эффективной разработки сложных бизнес-приложений вне тяжеловесных программных моделей, которые исторически были доминирующими в промышленности. Ещё одно его достоинство в том, что он ввел ранее неиспользуемые функциональные возможности в сегодняшние господствующие методы разработки, даже вне платформы Java.

Этот фреймворк предлагает последовательную модель и делает её применимой к большинству типов приложений, которые уже созданы на основе платформы Java. Считается, что **Spring** реализует модель разработки, основанную на лучших стандартах индустрии, и делает её доступной во многих областях Java.

**Spring** может быть рассмотрен как коллекция меньших фреймворков или фреймворков во фреймворке. Большинство этих фреймворков может работать независимо друг от друга, однако они обеспечивают большую функциональность при совместном их использовании. Эти фреймворки делятся на структурные элементы типовых комплексных приложений:

* [**Inversion of Control**](https://ru.wikipedia.org/wiki/Inversion_of_Control)**-контейнер**: конфигурирование компонентов приложений и управление жизненным циклом Java-объектов.
* **Фреймворк**[**аспектно-ориентированного программирования**](https://ru.wikipedia.org/wiki/%D0%90%D1%81%D0%BF%D0%B5%D0%BA%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%D0%BE%D0%B5_%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%D0%B5): работает с функциональностью, которая не может быть реализована возможностями [объектно-ориентированного программирования](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%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%D0%BE%D0%B5_%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%D0%B5) на Java без потерь.
* **Фреймворк доступа к данным**: работает с [системами управления реляционными базами данных](https://ru.wikipedia.org/wiki/%D0%A1%D0%A3%D0%A0%D0%91%D0%94) на Java-платформе, используя [JDBC](https://ru.wikipedia.org/wiki/JDBC)- и [ORM](https://ru.wikipedia.org/wiki/ORM)-средства и обеспечивая решения задач, которые повторяются в большом числе Java-based environments.
* **Фреймворк управления**[**транзакциями**](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D0%B7%D0%B0%D0%BA%D1%86%D0%B8%D1%8F_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)): координация различных [API](https://ru.wikipedia.org/wiki/API) управления транзакциями и инструментарий настраиваемого управления транзакциями для объектов Java.
* **Фреймворк**[**MVC**](https://ru.wikipedia.org/wiki/Model-view-controller): каркас, основанный на [HTTP](https://ru.wikipedia.org/wiki/HTTP) и [сервлетах](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%BB%D0%B5%D1%82), предоставляющий множество возможностей для расширения и настройки (*customization*).
* **Фреймворк удалённого доступа**: конфигурируемая передача Java-объектов через сеть в стиле [RPC](https://ru.wikipedia.org/wiki/Remote_procedure_call), поддерживающая [RMI](https://ru.wikipedia.org/wiki/RMI), [CORBA](https://ru.wikipedia.org/wiki/CORBA), [HTTP](https://ru.wikipedia.org/wiki/HTTP)-based протоколы, включая [web-сервисы](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%BB%D1%83%D0%B6%D0%B1%D0%B0) ([SOAP](https://ru.wikipedia.org/wiki/SOAP)).
* **Фреймворк**[**аутентификации**](https://ru.wikipedia.org/wiki/%D0%90%D1%83%D1%82%D0%B5%D0%BD%D1%82%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F)**и**[**авторизации**](https://ru.wikipedia.org/wiki/%D0%90%D0%B2%D1%82%D0%BE%D1%80%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F): конфигурируемый инструментарий процессов аутентификации и авторизации, поддерживающий много популярных и ставших индустриальными стандартами протоколов, инструментов, практик через дочерний проект [Spring Security](https://ru.wikipedia.org/wiki/Spring_Security) (ранее известный как [Acegi](https://ru.wikipedia.org/wiki/Acegi)).
* **Фреймворк удалённого управления**: конфигурируемое представление и управление Java-объектами для локальной или удалённой конфигурации с помощью [JMX](https://ru.wikipedia.org/wiki/JMX).
* **Фреймворк работы с сообщениями**: конфигурируемая регистрация объектов-слушателей сообщений для прозрачной обработки сообщений из [очереди сообщений](https://ru.wikipedia.org/wiki/%D0%9E%D1%87%D0%B5%D1%80%D0%B5%D0%B4%D1%8C_%D1%81%D0%BE%D0%BE%D0%B1%D1%89%D0%B5%D0%BD%D0%B8%D0%B9) с помощью [JMS](https://ru.wikipedia.org/wiki/Java_Message_Service), улучшенная отправка сообщений по стандарту JMS API.
* [**Тестирование**](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F): каркас, поддерживающий классы для написания модульных и интеграционных тестов.

**2.2. Hybernate**

**Hibernate** — самая популярная реализация спецификации [JPA](https://ru.wikipedia.org/wiki/Java_Persistence_API), предназначенная для решения задач объектно-реляционного отображения ([ORM](https://ru.wikipedia.org/wiki/ORM)). Распространяется [свободно](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%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) на условиях [GNU Lesser General Public License](https://ru.wikipedia.org/wiki/GNU_Lesser_General_Public_License).

Целью Hibernate является освобождение разработчика от значительного объёма сравнительно низкоуровневого программирования при работе в объектно-ориентированных средствах в реляционной базе данных. Разработчик может использовать Hibernate как в процессе проектирования системы классов и таблиц «с нуля», так и для работы с уже существующей [базой данных](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85).

Библиотека не только решает задачу связи классов Java с таблицами базы данных (и типов данных Java с типами данных [SQL](https://ru.wikipedia.org/wiki/SQL)), но и также предоставляет средства для автоматической генерации и обновления набора таблиц, построения запросов и обработки полученных данных и может значительно уменьшить время разработки, которое обычно тратится на ручное написание [SQL](https://ru.wikipedia.org/wiki/SQL)- и [JDBC](https://ru.wikipedia.org/wiki/JDBC)-кода. Hibernate автоматизирует генерацию SQL-запросов и освобождает разработчика от ручной обработки результирующего набора данных и преобразования объектов, максимально облегчая перенос (портирование) приложения на любые базы данных SQL.

Hibernate обеспечивает прозрачную поддержку сохранности данных (*persistence*) для «[POJO](https://ru.wikipedia.org/wiki/Plain_Old_Java_Object)» (то есть для стандартных Java-объектов); единственное строгое требование для сохраняемого класса — наличие [конструктора по умолчанию](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%82%D0%BE%D1%80_%D0%BF%D0%BE_%D1%83%D0%BC%D0%BE%D0%BB%D1%87%D0%B0%D0%BD%D0%B8%D1%8E) (без параметров). Для корректного поведения в некоторых приложениях требуется также уделить внимание методам equals() и hashCode().

Mapping (сопоставление, проецирование) Java-классов с таблицами базы данных осуществляется с помощью конфигурационных [XML](https://ru.wikipedia.org/wiki/XML)-файлов или [Java-аннотаций](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%BD%D0%BE%D1%82%D0%B0%D1%86%D0%B8%D1%8F_(Java)). При использовании файла XML Hibernate может генерировать скелет [исходного кода](https://ru.wikipedia.org/wiki/%D0%98%D1%81%D1%85%D0%BE%D0%B4%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4) для классов длительного хранения. В этом нет необходимости, если используется аннотация. Hibernate может использовать файл XML или аннотации для поддержки [схемы базы данных](https://ru.wikipedia.org/wiki/%D0%A1%D1%85%D0%B5%D0%BC%D0%B0_%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85).

Обеспечиваются возможности по организации отношения между классами «[один-ко-многим](https://ru.wikipedia.org/w/index.php?title=%D0%9E%D0%B4%D0%B8%D0%BD-%D0%BA%D0%BE-%D0%BC%D0%BD%D0%BE%D0%B3%D0%B8%D0%BC&action=edit&redlink=1)» и «[многие-ко-многим](https://ru.wikipedia.org/w/index.php?title=%D0%9C%D0%BD%D0%BE%D0%B3%D0%B8%D0%B5-%D0%BA%D0%BE-%D0%BC%D0%BD%D0%BE%D0%B3%D0%B8%D0%BC&action=edit&redlink=1)». В дополнение к управлению связями между объектами Hibernate также может управлять [рефлексивными отношениями](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D1%84%D0%BB%D0%B5%D0%BA%D1%81%D0%B8%D0%B2%D0%BD%D0%BE%D0%B5_%D0%BE%D1%82%D0%BD%D0%BE%D1%88%D0%B5%D0%BD%D0%B8%D0%B5), где объект имеет связь «один-ко-многим» с другими экземплярами своего собственного [типа данных](https://ru.wikipedia.org/wiki/%D0%A2%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85).

Hibernate поддерживает отображение пользовательских типов значений. Это делает возможными такие сценарии:

* Переопределение типа по умолчанию SQL, Hibernate выбирает при отображении столбца свойства.
* Проецирование [перечисляемого типа](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D1%87%D0%B8%D1%81%D0%BB%D1%8F%D0%B5%D0%BC%D1%8B%D0%B9_%D1%82%D0%B8%D0%BF) Java на поле БД, будто они являются обычными свойствами.
* Проецирование одного свойства в несколько колонок.

Коллекции объектов данных, как правило, хранятся в виде коллекций Java-объектов, таких, как набор (Set) и список (List). Поддерживаются [обобщенные классы](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D0%BE%D0%B1%D1%89%D0%B5%D0%BD%D0%BD%D0%BE%D0%B5_%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%D0%B5) (Generics), введеные в Java 5. Hibernate может быть настроен на «ленивые» (отложенные) загрузки коллекций. Отложенные загрузки является вариантом по умолчанию, начиная с Hibernate 3.

Связанные объекты могут быть настроены на *каскадные* операции. Например, родительский класс Album (музыкальный альбом) может быть настроен на каскадное сохранение и/или удаление своего потомка Track. Это может сократить время разработки и обеспечить [целостность](https://ru.wikipedia.org/wiki/%D0%A1%D1%81%D1%8B%D0%BB%D0%BE%D1%87%D0%BD%D0%B0%D1%8F_%D1%86%D0%B5%D0%BB%D0%BE%D1%81%D1%82%D0%BD%D0%BE%D1%81%D1%82%D1%8C). Функция проверки изменения данных (*dirty checking*) позволяет избежать ненужной записи действий в базу данных, выполняя SQL-обновление только при изменении полей персистентных объектов.

Успех библиотеки Hibernate подтолкнул [JCP](https://ru.wikipedia.org/wiki/JCP) к разработке спецификации JDO, ставшей одной из стандартных технологий ORM на платформе JavaEE. Также Hibernate совместима с JSR-220/317 и предоставляет стандартные средства JPA.

Hibernate может использоваться как в самостоятельных приложениях [Java](https://ru.wikipedia.org/wiki/Java), так и в программах [Java EE](https://ru.wikipedia.org/wiki/Java_EE), выполняемых на сервере (например, [сервлет](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%BB%D0%B5%D1%82) или компоненты [EJB](https://ru.wikipedia.org/wiki/EJB)). Также он может включаться как дополнительная возможность к другим языкам программирования. Например, [Adobe](https://ru.wikipedia.org/wiki/Adobe_Systems) интегрировал Hibernate в девятую версию [ColdFusion](https://ru.wikipedia.org/wiki/ColdFusion) (запускаемый на серверах с поддержкой приложений [J2EE](https://ru.wikipedia.org/wiki/J2EE)) с уровнем абстракции новых функций и синтаксиса, приложенных к [CFML](https://ru.wikipedia.org/w/index.php?title=CFML&action=edit&redlink=1).

**2.3. JXFW**

**Croc JXFW** - программная платформа Java Extendable Framework.

Платформа для разработки корпоративных автоматизированных систем Java eXtendable FrameWork (jXFW) предоставляет разработчикам полный набор инструментов для создания автоматизированных систем в соответствии с критериями заказчиков.

Платформа jXFW создана на открытых стандартах и технологиях. Это снижает риски встраивания систем в уже действующую инфраструктуру и защищает инвестиции. В основе платформы jXFW стек технологий [Java](http://www.tadviser.ru/index.php/%D0%9F%D1%80%D0%BE%D0%B4%D1%83%D0%BA%D1%82:Java).

В основе архитектуры решений на платформе jXFW действуют принципы Model-driven engineering (MDE). В центре решения модель предметной области, на основе которой автоматически генерируются ключевые сервисы приложений. С этим подходом сокращаются сроки разработки первой версии прикладного решения, обеспечивается гибкость внесения изменений.

Интеграционный модуль платформы снижает затраты на настройку подключения созданного решения к внешним системам и интеграционным шинам.

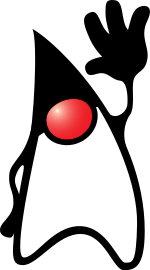
В интерфейсе пользователя использованы возможности HTML5. Клиентские приложения работают, как в браузере, так и в виде Windows-приложений или на мобильных устройствах под управлением iOS и Android.

Заложенные в архитектуру платформы принципы позволяют расширять её возможности посредством подключения дополнительных модулей. С течением времени состав базовых компонентов пополняется поддержкой других видов генераторов отчетов, СУБД и NoSql, ECM -систем и других возможностей.

**2.4. Java**

**Java** — сильно типизированный [объектно-ориентированный язык программирования](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%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), разработанный компанией [Sun Microsystems](https://ru.wikipedia.org/wiki/Sun_Microsystems) (в последующем приобретённой компанией [Oracle](https://ru.wikipedia.org/wiki/Oracle)). Приложения Java обычно [транслируются](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D1%81%D0%BB%D1%8F%D1%82%D0%BE%D1%80) в специальный [байт-код](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B9%D1%82-%D0%BA%D0%BE%D0%B4), поэтому они могут работать на любой компьютерной архитектуре с помощью [виртуальной Java-машины](https://ru.wikipedia.org/wiki/Java_Virtual_Machine). Дата официального выпуска — 23 мая 1995 года. На 2019 год Java — один из самых популярных языков программирования.

Программы на Java [транслируются](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D1%81%D0%BB%D1%8F%D1%82%D0%BE%D1%80) в [байт-код Java](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B9%D1%82-%D0%BA%D0%BE%D0%B4_Java), выполняемый [виртуальной машиной Java](https://ru.wikipedia.org/wiki/Java_Virtual_Machine) (JVM) — программой, обрабатывающей байтовый код и передающей инструкции оборудованию как [интерпретатор](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BF%D1%80%D0%B5%D1%82%D0%B0%D1%82%D0%BE%D1%80).

[](https://commons.wikimedia.org/wiki/File:Duke_(Java_mascot)_waving.svg?uselang=ru)

Достоинством подобного способа выполнения программ является полная независимость байт-кода от [операционной системы](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0) и [оборудования](https://ru.wikipedia.org/wiki/%D0%90%D0%BF%D0%BF%D0%B0%D1%80%D0%B0%D1%82%D0%BD%D0%B0%D1%8F_%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B0), что позволяет выполнять Java-приложения на любом устройстве, для которого существует соответствующая виртуальная машина. Другой важной особенностью технологии Java является гибкая система безопасности, в рамках которой исполнение программы полностью контролируется виртуальной машиной. Любые операции, которые превышают установленные полномочия программы (например, попытка несанкционированного доступа к данным или соединения с другим компьютером), вызывают немедленное прерывание.

**2.6. WEBCLIENT**

**Croc WEBCLIENT** – это фреймворк, который используется в связке с JXFW для генерации клиентской части кода, взаимодействия со стороны клиента с бд и реализации графического интерфейса.

При работе с фреймворком используются языки программирования javascript и typescript, а также hbs-файлы для верстки шаблонов страниц и less-файлы для описания стилей.

**2.7 Vaadin**

**Vaadin** — [свободно распространяемый](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%D0%BD%D0%BE_%D1%80%D0%B0%D1%81%D0%BF%D1%80%D0%BE%D1%81%D1%82%D1%80%D0%B0%D0%BD%D1%8F%D0%B5%D0%BC%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) [фреймворк](https://ru.wikipedia.org/wiki/%D0%A4%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA) для создания [RIA-веб-приложений](https://ru.wikipedia.org/wiki/Rich_Internet_Application), разрабатываемый одноимённой финской компанией. В отличие от [библиотек](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0_(%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%D0%B5)) на [Javascript](https://ru.wikipedia.org/wiki/Javascript) и специфических [плагинов](https://ru.wikipedia.org/wiki/%D0%9F%D0%BB%D0%B0%D0%B3%D0%B8%D0%BD) для [браузеров](https://ru.wikipedia.org/wiki/%D0%91%D1%80%D0%B0%D1%83%D0%B7%D0%B5%D1%80), Vaadin предлагает [сервер](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))-ориентированную [архитектуру](https://ru.wikipedia.org/wiki/%D0%90%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F), базирующуюся на [Java Enterprise Edition](https://ru.wikipedia.org/wiki/J2EE). Использование JEE позволяет выполнять основную часть логики приложения на стороне сервера, тогда как технология [AJAX](https://ru.wikipedia.org/wiki/AJAX), используемая на стороне браузера, позволяет [интерактивно](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%B0%D0%BA%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D1%81%D1%82%D1%8C) взаимодействовать с пользователем, не отставая от аналогичных десктоп-приложений. Для отображения элементов [пользовательского интерфейса](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F) и взаимодействия с [сервером на стороне клиента](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B8%D0%B5%D0%BD%D1%82-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80) Vaadin использует [Google Web Toolkit](https://ru.wikipedia.org/wiki/Google_Web_Toolkit).

Использование [Java](https://ru.wikipedia.org/wiki/Java) как единственного [языка 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) при создании [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) и [веб-контента](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%82%D0%B5%D0%BD%D1%82) — одна из наиболее значимых функций в Vaadin. [Фреймворк](https://ru.wikipedia.org/wiki/%D0%A4%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA) использует [событийную модель](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D0%B1%D1%8B%D1%82%D0%B8%D0%B9%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%D0%BE%D0%B5_%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%D0%B5) и определенные элементы пользовательского интерфейса, [виджеты](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%B4%D0%B6%D0%B5%D1%82), что делает её очень близкой к модели разработки настольных приложений на Java с использованием [HTML](https://ru.wikipedia.org/wiki/HTML) и Javascript.

Организация [модели данных](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) и виджетов позволяет отображать в браузере большие объёмы данных без значительной загрузки [оперативной памяти](https://ru.wikipedia.org/wiki/%D0%9E%D0%97%D0%A3) и без дополнительных действий со стороны [разработчика](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D1%87%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%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F). Использование [Google Web Toolkit](https://ru.wikipedia.org/wiki/Google_Web_Toolkit) для отображения страниц с результатами поиска и обработки действий пользователя (наподобие терминального клиента). Так как Google Web Toolkit функционирует только на стороне клиента, Vaadin добавляет дополнительную [валидацию](https://ru.wikipedia.org/wiki/%D0%92%D0%B0%D0%BB%D0%B8%D0%B4%D0%B0%D1%86%D0%B8%D1%8F) данных на стороне сервера: это решает проблемы безопасности, связанные с возможностью подмены данных или кода Javascript. Соответственно, при изменении и повреждении данных, поступающих от браузера, сервер, определив это, не пропускает запросы.

Расширяемость обеспечивается возможностью использования дополнительных виджетов, написанных для GWT, а также кастомизации при помощи [CSS](https://ru.wikipedia.org/wiki/CSS). Однако стандартное приложение, создаваемое на Vaadin, не требует [программирования](https://ru.wikipedia.org/wiki/%D0%9F%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%D0%B5) именно на GWT и последующей компиляции GWT-компилятором, если только разработчик не добавляет в проект нестандартные виджеты.

**2.8 Camunda**

Camunda – это платформа для критически важных бизнес-процессов. Данную систему используют крупные компании во всем мире, где важна масштабируемость, надёжность, производительность. Данная платформа при использовании совместно с Java-приложениями позволяет инстанцировать, отслеживать, конфигурировать, моделировать и управлять бизнес-процессами.

**3. Практическая часть**

**3.1. Постановка задачи**

* Изучить технологии применимые при разработке современных систем корпоративного масштаба на языке Java.
* Изучить возможности различных инструментов применимых при разработке на данной платформе.
* Изучить применяемые архитектурные решения при построении крупных систем корпоративного масштаба, паттерны проектирования, возможные подходы к проектировке.

**3.2 Выполнение**

В ходе выполнения задания была разработана система c использованием всех вышеописанных технологий и подходов Code First и Domain Driven Design. Были изучены патерны проектирования, такие как Façade, DTO, Factory, Builder, Singleton и пр.

Было рассмотрено множество систем с различными подходами к разработке, был освоен подход TDD и несколько современных методологий разработки, например Agile.

Далее приведены некоторые листинги кода.

Класс пользователя описаный на языке Xtend:

package ru.croc.dgp.ugd.domain  
  
import java.util.Set  
import javax.persistence.Column  
import javax.persistence.JoinColumn  
import javax.persistence.JoinTable  
import javax.persistence.Table  
import org.hibernate.annotations.FetchMode  
import org.hibernate.annotations.Fetch  
import ru.croc.ctp.jxfw.core.domain.meta.XFWElementLabel  
import ru.croc.ctp.jxfw.core.domain.meta.persistence.XFWManyToMany  
import ru.croc.ctp.jxfw.core.domain.meta.persistence.XFWManyToOne  
import ru.croc.ctp.jxfw.core.generator.meta.XFWObject  
import ru.croc.dgp.ugd.domain.classifier.Company  
import ru.croc.dgp.ugd.domain.classifier.Department  
import ru.croc.dgp.ugd.domain.classifier.Position  
import ru.croc.dgp.ugd.domain.classifier.Rule  
import javax.persistence.FetchType  
import ru.croc.ctp.jxfw.core.domain.meta.XFWDefaultValue  
  
@XFWObject  
@XFWElementLabel("Пользователь")  
@Table(name = "user\_table")  
class User {  
  
 @XFWElementLabel("Идентификатор СУДИР")  
 @Column(nullable = false)  
 String sudirId  
  
 @XFWElementLabel("Фамилия")  
 @Column(nullable = false)  
 String lastName  
  
 @XFWElementLabel("Имя")  
 @Column(nullable = false)  
 String firstName  
  
 @XFWElementLabel("Отчество")  
 @Column(nullable = true)  
 String middleName  
  
 @XFWElementLabel("Организация")  
 @XFWManyToOne(optional = false)  
 Company company  
  
 @XFWElementLabel("Чиновник")  
 @XFWDefaultValue("false")  
 @Column(nullable = false)  
 Boolean isOfficial  
  
 @XFWElementLabel("Подразделение")  
 @XFWManyToOne(optional=true)  
 Department department  
  
 @XFWElementLabel("Должность")  
 @XFWManyToOne(optional=true)  
 Position position  
  
 @XFWElementLabel("e-mail")  
 @Column(nullable = true)  
 String email  
  
 @XFWElementLabel("Телефон")  
 @Column(nullable = true)  
 String phone  
  
 @XFWManyToMany(fetch = FetchType.EAGER)  
 @XFWElementLabel("Доступные полномочия")  
 @Fetch(FetchMode.SUBSELECT)  
 @JoinTable(  
 name = "user\_rules",  
 joinColumns = @JoinColumn(name = "user\_id", referencedColumnName = "id"),  
 inverseJoinColumns = @JoinColumn(name = "rule\_code", referencedColumnName = "code")  
 )  
 @JoinColumn(nullable = true)  
 Set<Rule> rules  
  
 @XFWElementLabel("Логин")  
 String login  
  
 @XFWElementLabel("Пароль")  
 String password  
  
 def getFullName() {  
 var str = ""  
 str += lastName;  
 if (firstName !== null) {  
 str += (" " + firstName.charAt(0) + ".")  
 }  
 if (middleName !== null) {  
 str += (" " + middleName.charAt(0) + ".")  
 }  
 return str  
 }  
  
}  
  
@XFWObject(persistence = TRANSIENT)  
public class PrincipalUser {  
  
 String login;  
  
 User user;  
  
}

Пример скрипта из клиентской части:

**import** $ = **require**("jquery");  
**import** core = **require**("core");  
**import** xconfig = **require**("xconfig");  
**import** model = **require**("app/domain/model-ext");  
**import** DataFacadeSmart = **require**("lib/interop/DataFacadeSmart");  
**import** BackendInterop = **require**("lib/interop/BackendInterop");  
**import** DataStoreFactory = **require**("lib/data/DataStoreFactory");  
**import** AppNavMenu = **require**("lib/ui/menu/AppNavMenu");  
**import** SystemMenu = **require**("lib/ui/menu/SystemMenu");  
**import** AppToolbar = **require**("lib/ui/AppToolbar");  
**import** templateApp = **require**("xhtmpl!app/ui/templates/app.hbs");  
**import** appToolbarTemplate = **require**("xhtmpl!app/ui/templates/appToolbar.hbs");  
**import** systemMenuTemplate = **require**("xhtmpl!app/ui/templates/menu/SystemMenu.hbs");  
**import** resources = **require**("i18n!app/nls/resources");  
  
**import** "app/iconProvider";  
**import** "app/default-menus";  
**import** "app/default-parts";  
**import** "app/parts";  
**import** "app/area-default";  
**import** "app/area-printing";  
**import** "app/hbs-helpers";  
  
**import** "app/ui/my\_account/objects/ObjectActionsPart"  
**import** "app/ui/my\_account/projects/ProjectActionsPart"  
**import** "app/ui/application/ApplicationParts"  
**import** "app/ui/application/AddDocumentActionsPart"  
**import** "app/ui/application/DocumentListForAppViewer"  
**import** "app/ui/application/TaskDecisionListForAppViewer"  
**import** "app/ui/application/ApplicationShortViewer"  
**import** "app/ui/application/ApplicationPrintingForm"  
**import** "app/ui/application/ApplicationSetExecutiveDepartmentPart"  
**import** "app/ui/application/ApplicationRegistration"  
**import** "app/ui/application/ApplicationSetDeputyOrAssignee"  
**import** "app/ui/application/ApplicationChecking"  
**import** "app/ui/application/ApplicationConsideration"  
**import** "app/ui/application/PerformanceIndicatorListForAppConsideration"  
**import** "app/ui/application/ApplicationPreparingDecision"  
**import** "app/ui/application/StatusOfFillingApplicationDecisionPart"  
**import** "app/ui/bestReleasedProject/BestReleasedProjectParts"  
**import** "app/ui/bestReleasedProject/BRPApplicationEditor"  
**import** "app/ui/document/DocumentParts"  
**import** "app/ui/company/CompanyParts"  
**import** "app/ui/bestReleasedProject/CapitalConstructionObjectViewer"  
**import** "app/ui/my\_account/tasks/TaskList"  
**import** "app/ui/my\_account/requests/RequestList"  
**import** "app/ui/my\_account/objects/CCOList"  
**import** "app/ui/my\_account/projects/ProjectList"  
**import** "app/ui/my\_account/MyAccountPart"  
**import** {myActionsPart} **from** "./ui/myActionsMenuPart";  
//import "app/ui/main\_area/main\_area"  
  
**import** "modules/security/module-security"  
**import** "modules/files/module-files"  
**import** "lib/ui/slick/SlickInlineEditAddon"  
  
**import** lang = core.lang;  
**import** Area = core.composition.Area;  
**import** {PrincipalUser, User} **from** "./domain/model";  
  
**if** (!xconfig) {  
 **let** err = "No xconfig global object found. Unable to proceed.";  
 alert(err);  
 **throw new** Error(err);  
}  
  
**class** Application **extends** core.Application {  
 appToolbar: AppToolbar;  
 sysMenu: SystemMenu;  
 navMenu: core.ui.AppNavMenu;  
 model: **typeof** model;  
  
 **constructor**() {  
 **super**(xconfig, {template: templateApp});  
 **this**.model = model;  
 }  
  
 createDataFacade() {  
 **return** DataStoreFactory  
 .create(xconfig.appName, /\*db version:\*/ 1, model.meta)  
 .then(**function** (store) {  
 **return new** DataFacadeSmart(  
 **new** BackendInterop(xconfig),  
 **null**, // eventPublisher, Application will initialize it while running  
 store,  
 {  
 // *TODO: cacheManager: new CacheManager()* }  
 );  
 });  
 }  
  
 preinitialize() {  
 **let** that = **this**;  
  
 // зарегаю сейчас, потому что он нужен в шапке, а до этого момента парты не регались  
 **this**.registerPart("MyActions", myActionsPart);  
  
 that.appToolbar = **new** AppToolbar(that, {  
 template: appToolbarTemplate,  
 authMenu: **false**,  
 langMenu: **false**,  
 onlineBeacon: **false** //affix: false  
 });  
 that.sysMenu.setTemplate(systemMenuTemplate);  
 }  
  
 initialize() {  
 **let** that = **this**;  
  
 that.eventPublisher.subscribe("security.login", **function** (args) {  
 **let** uow = that.createUnitOfWork();  
 uow.getCurrentUser().then(**function**(principalUser) {  
 uow.load(User, principalUser.id, {preloads: ["rules", "company"]}).done(**function**(args){  
 that["currentUser"] = args;  
 });  
 });  
 });  
  
 $('.logout').click(  
 **function**(){  
 that.dataFacade.logout()  
 .then(  
 **function**(){  
 that.areaManager.getActiveArea().activateState("");  
 /\*that.dataFacade.login()  
 .then(  
 function(){  
 that.areaManager.getActiveArea().activateState("");  
 }  
 );\*/  
 });  
 }  
 );  
  
 // top navigation menu (switching areas):  
 // NOTE: AppNavToolbar cannot be created in preinitialize and rendered in template as it depends on AreaManager which is initialized after template rendered  
 //that.navMenu = new AppNavMenu(that.areaManager);  
 //that.appToolbar.appNavMenu(that.navMenu);  
 }  
}  
  
**export** = Application;

Пример Hbs-шаблона:

<form onsubmit="**return false**">  
 <div class="gray-container custom-pe">  
 {{#**with viewModel**}}  
 <div style="display: flex; flex-direction: row;">  
 <div style="flex: 1 1 100%;">  
 <div class="container">  
 <div class="row">  
 <div class="{{**get-col-class** 12}}">  
 <span class="main-region\_\_title">Заявка на участие в ежегодном конкурсе</br>«Лучший реализованный проект в области строительства»</span>  
 </div>  
 </div>  
 <div class="row">  
 <div class="{{**get-col-class** 12}}">  
 <span class="step-title">Шаг 2. Данные организации, представившей проект на конкурс</span>  
 <div class="tile">  
 <div class="row">  
 <div class="{{**get-col-class** 8}}">  
 <div class="blue-title" style="margin-top: 0;">Общие данные</div>  
  
 {{!-- наименование, адрес организации --}}  
 <div class="row margin-top-16px">  
 <div class="{{**get-col-class** 6}}">  
 <span class="prop\_\_label\_\_bold">Наименование</span>  
 {{#**if representative**}}  
 <div class="prop\_\_value">{{**representative**.**company**.**name**}}</div>  
 {{**else**}}  
 <div class="prop\_\_value empty">Нет данных</div>  
 {{/**if**}}  
 </div>  
 <div class="{{**get-col-class** 6}}">  
 <span class="prop\_\_label\_\_bold">Адрес организации</span>  
 {{#**if representative**}}  
 <div class="prop\_\_value">не знаю что тут</div>  
 {{**else**}}  
 <div class="prop\_\_value empty">Нет данных</div>  
 {{/**if**}}  
 </div>  
 </div>  
 <div class="blue-title" style="margin-top: 0;"></div>  
  
 {{!-- контактный тел организации, контактное лицо --}}  
 <div class="row margin-top-16px">  
 <div class="{{**get-col-class** 6}}">  
 <span class="prop\_\_label\_\_bold">Контактный телефон организации</span>  
 <div class="row">  
 {{#**if companyContactNumber**}}  
 <div class="{{**get-col-class** 11}}">  
 <div id="{{**push-id**}}" >{{**pe name**="companyContactNumber" **target**=":pop-id"}}</div>  
 </div>  
 <div class="{{**get-col-class** 1}}">  
 {{**info-for-prefilled-pe**}}  
 </div>  
 {{**else**}}  
 <div class="{{**get-col-class** 12}}">  
 <div id="{{**push-id**}}" >{{**pe name**="companyContactNumber" **target**=":pop-id"}}</div>  
 </div>  
 {{/**if**}}  
 </div>  
 </div>  
 <div class="{{**get-col-class** 6}}">  
 <span class="prop\_\_label\_\_bold">Контактное лицо по вопросам участия в конкурсе:</span>  
 <div class="row">  
 {{#**if contactPerson**}}  
 <div class="{{**get-col-class** 11}}">  
 <div id="{{**push-id**}}" >{{**pe name**="contactPerson" **target**=":pop-id" }}</div>  
 </div>  
 <div class="{{**get-col-class** 1}}">  
 {{**info-for-prefilled-pe**}}  
 </div>  
 {{**else**}}  
 <div class="{{**get-col-class** 12}}">  
 <div id="{{**push-id**}}" >{{**pe name**="contactPerson" **target**=":pop-id"}}</div>  
 </div>  
 {{/**if**}}  
 </div>  
 </div>  
 </div>  
  
 {{!-- тел контактного лица, email контактного лица --}}  
 <div class="row margin-top-16px" >  
 <div class="{{**get-col-class** 6}}">  
 <span class="prop\_\_label\_\_bold">Телефон контактного лица</span>  
 <div class="row">  
 {{#**if contactPersonNumber**}}  
 <div class="{{**get-col-class** 11}}">  
 <div id="{{**push-id**}}" >{{**pe name**="contactPersonNumber" **target**=":pop-id"}}</div>  
 </div>  
 <div class="{{**get-col-class** 1}}">  
 {{**info-for-prefilled-pe**}}  
 </div>  
 {{**else**}}  
 <div class="{{**get-col-class** 12}}">  
 <div id="{{**push-id**}}" >{{**pe name**="contactPersonNumber" **target**=":pop-id"}}</div>  
 </div>  
 {{/**if**}}  
 </div>  
 </div>  
 <div class="{{**get-col-class** 6}}">  
 <span class="prop\_\_label\_\_bold">Email контактного лица</span>  
 <div class="row">  
 {{#**if contactPersonEmail**}}  
 <div class="{{**get-col-class** 11}}">  
 <div id="{{**push-id**}}" >{{**pe name**="contactPersonEmail" **target**=":pop-id"}}</div>  
 </div>  
 <div class="{{**get-col-class** 1}}">  
 {{**info-for-prefilled-pe**}}  
 </div>  
 {{**else**}}  
 <div class="{{**get-col-class** 12}}">  
 <div id="{{**push-id**}}" >{{**pe name**="contactPersonEmail" **target**=":pop-id"}}</div>  
 </div>  
 {{/**if**}}  
 </div>  
 </div>  
 </div>  
  
 <div class="blue-title margin-top-32px" >Форма участия организации (физического лица) в проекте, выдвигаемом на конкурс:</div>  
 {{#**with representative**}}  
 <div class="brpcontainer">  
 <div class="container1">  
 <h3>Наименовании организации</h3>  
 {{#**with this**.**company**}}  
 <p>{{**this**.**name**}}</p>  
 {{/**with**}}  
 </div>  
 <div class="container2">  
 <h3>Форма участия</h3>  
 {{#**with this**.**role**}}  
 <p>{{**this**.**name**}}</p>  
 {{/**with**}}  
 </div>  
 </div>  
 {{/**with**}}  
  
 <div class="blue-title margin-top-32px" >Другие участники проекта</div>  
 <div class="row">  
 <div class="{{**get-col-class** 12}}">  
 {{**render ..**/**this**.**editor**.**options**.**constructionParticipantList**}}  
 </div>  
 </div>  
 </div>  
 </div>  
 </div>  
 <div>  
 <button class="white-button margin-top-24px" {{**command-bind ..**/**this**.**editor**.**options**.**commands**.**toStep editor**=**..**/**this**.**editor stepNumber**="1"}} >  
 <span class="white-button\_\_text" >Назад</span>  
 </button>  
 <button class="blue-button margin-top-24px" {{**command-bind ..**/**this**.**editor**.**options**.**commands**.**toStep3 editor**=**..**/**this**.**editor** }} >  
 <span class="blue-button\_\_text" >ДАЛЕЕ</span>  
 </button>  
 <button class="colorless-button margin-top-24px" {{**command-bind ..**/**this**.**editor**.**commands**.**save editor**=**..**/**this**.**editor**}} >  
 <span class="colorless-button\_\_text" >Сохранить</span>  
 </button>  
 </div>  
 </div>  
 </div>  
 </div>  
 </div>  
 <div style="flex: 0 0 325px; margin-left: 32px;"></div>  
 </div>  
 {{/**with**}}  
 </div>  
</form>

Пример СSS-файла:

@import "fonts.less";  
@import "../../../../node\_modules/@croc/webclient/dist/lib/ui/styles/.variables.less";  
@import "../../../../node\_modules/@croc/webclient/dist/lib/ui/styles/.mixins.less";  
@import ".variables-override.less";  
  
html, body {  
 margin: 0;  
 padding: 0;  
 height: 100%;  
 background-color: @color-tone;  
}  
  
body {  
 font-family: @font-family, 'PT Sans', sans-serif;  
}  
  
.x-areas-container {  
 padding: 0 **!important**;  
 .x-area {  
 background-color: @color-content;  
 .x-region {  
 padding: 0 **!important**;  
 }  
 }  
}  
  
.clearfloat {  
 clear: both;  
}  
  
.\_textAlign\_center {  
 text-align: center;  
}  
  
.footer-stub {  
 height: 50px;  
}  
  
@headerHeight: 80px;  
@pageMinWidth: 1768px;  
  
.header {  
 @logoWidth: 248px;  
 @userInfoWidth: 350px;  
 display: flex;  
 flex-wrap: wrap;  
 width: 100%;  
 min-width: @pageMinWidth;  
 height: @headerHeight;  
 background-color: #ffffff;  
 &\_\_logo,  
 &\_\_user-info {  
 height: @headerHeight;  
 }  
 &\_\_search,  
 &\_\_calendar,  
 &\_\_my-events {  
 height: @headerHeight - 32px;  
 }  
 &\_\_logo {  
 display: flex;  
 justify-content: flex-start;  
 align-items: center;  
 flex: 0 0 @logoWidth;  
 max-width: @logoWidth;  
 padding: 0 20px 0;  
 }  
 &\_\_center {  
 display: flex;  
 flex-wrap: nowrap;  
 min-width: 1168px;  
 width: calc(100% - 248px - 350px);  
 padding: 16px 40px 16px 24px;  
 }  
 &\_\_search {  
 flex: 0 0 45%;  
 max-width: 45%;  
 }  
 &\_\_calendar {  
 margin-left: auto;  
 flex: 0 0 15%;  
 max-width: 15%;  
 }  
 &\_\_my-events {  
 flex: 0 0 15%;  
 max-width: 15%;  
 }  
 &\_\_user-info {  
 flex: 0 0 @userInfoWidth;  
 max-width: @userInfoWidth;  
 padding: 16px 28px 16px 23px;  
 }  
}  
  
.header-logo {  
 display: block;  
}  
  
.header-search {  
 &\_\_input {  
 display: block;  
 width: 100%;  
 height: 48px;  
 border-radius: 24px;  
 background-color: #f1f3f6;  
 border: none;  
 padding: 0 24px 0;  
 outline: none;  
 appearance: text-field;  
 &::-webkit-input { // Стили placeholder  
 font-size: 14px;  
 line-height: 16px;  
 color: rgba(#7f8ba5, .73);  
 }  
 &::-moz { // Стили placeholder  
 font-size: 14px;  
 line-height: 16px;  
 color: rgba(#7f8ba5, .73);  
 }  
 &:-moz { // Стили placeholder  
 font-size: 14px;  
 line-height: 16px;  
 color: rgba(#7f8ba5, .73);  
 }  
 &:-ms-input { // Стили placeholder  
 font-size: 14px;  
 line-height: 16px;  
 color: rgba(#7f8ba5, .73);  
 }  
 &::-ms-clear { // Скрытие крестика  
 display: none;  
 width : 0;  
 height: 0;  
 }  
 &::-ms-reveal { // Скрытие крестика  
 display: none;  
 width : 0;  
 height: 0;  
 }  
 &::-webkit-search-decoration,  
 &::-webkit-search-cancel-button,  
 &::-webkit-search-results-button,  
 &::-webkit-search-results-decoration { // Скрытие крестика  
 display: none;  
 }  
 }  
}  
  
.header-user-info {  
 display: flex;  
 justify-content: space-between;  
 border-left: 1px solid #dae2f3;  
 border-bottom: 1px solid #dae2f3;  
 &\_\_dropdown {  
 display: flex;  
 align-items: center;  
 cursor: pointer;  
 }  
 &\_\_photo {  
 width: 48px;  
 height: 48px;  
 img {  
 display: block;  
 width: 100%;  
 height: 100%;  
 object-fit: cover;  
 object-position: 50% 50%;  
 }  
 }  
 &\_\_button {  
 margin-left: -10px;  
 }  
 &\_\_notify {  
 position: relative;  
 > .x-popupview {  
 right: 0;  
 }  
 }  
}  
  
.header-notify {  
 display: flex;  
 justify-content: center;  
 align-items: center;  
 width: 48px;  
 height: 48px;  
 padding: 0;  
 background-color: rgba(#6e93e0, .15);  
 border: none;  
 border-radius: 50%;  
 outline: none;  
 cursor: pointer;  
 img, svg {  
 width: 24px;  
 height: 24px;  
 }  
}  
  
/\* Большие кнопки \*/  
.btn-bpm {  
 position: relative;  
 display: inline-block;  
 height: 48px;  
 padding-right: 22px;  
 padding-left: 22px;  
 background-color: transparent;  
 border: none;  
 border-radius: 24px;  
 outline: none **!important**;  
 cursor: pointer;  
 text-align: center;  
 white-space: nowrap;  
 img, svg {  
 position: absolute;  
 top: 50%;  
 transform: translateY(-50%);  
 }  
 span {  
 font-family: 'Roboto';  
 font-size: 14px;  
 font-weight: 700;  
 line-height: 44px;  
 color: #667278;  
 }  
 &\_border {  
 border-width: 2px;  
 border-style: solid;  
 border-color: #667278;  
 span {  
 line-height: 44px;  
 }  
 }  
 &\_color {  
 &\_blue {  
 &.btn-bpm\_border {  
 border-color: #6e93e0;  
 }  
 span {  
 color: #6e93e0;  
 }  
 }  
 }  
 &\_icon {  
 &\_left {  
 padding-left: 51px;  
 img, svg {  
 left: 18px;  
 }  
 }  
 &\_right {  
 padding-right: 51px;  
 img, svg {  
 right: 18px;  
 }  
 }  
 }  
}  
  
.menu-region {  
 width: @menu-region-width;  
}  
  
// цвета текста  
.blue-text {  
 color: @blue-text;  
}  
.dark-mint-text {  
 color: @dark-mint  
}  
  
.inline-block {  
 display: inline-block;  
}  
  
// стили чекбоксов  
input[type='checkbox']{  
 margin: 0;  
 &:checked{  
 &:before {  
 content: url(../../../content/img/checkbox-checked.svg);  
 }  
 }  
}  
  
[data-tooltip-text] {  
 &.tooltip-for-prefilled-pe {  
 &::before {  
 width: 270px;  
 }  
 }  
 position: relative;  
 &::before {  
 content: attr(data-tooltip-text);  
 display: none;  
 position: absolute;  
 bottom: calc(100% + 12px);  
 left: 50%;  
 transform: translateX(-50%);  
 padding: 16px;  
 background-color: #ffffff;  
 border-radius: 8px;  
 box-shadow: 0 25px 50px 0 rgba(0, 38, 116, 0.2);  
 max-width: 270px;  
 width: 100%;  
 font-size: 14px;  
 font-weight: 400;  
 line-height: 16px;  
 color: #525f66;  
 z-index: 9;  
 }  
 &::after {  
 content: '';  
 display: none;  
 width: 0;  
 height: 0;  
 position: absolute;  
 bottom: 100%;  
 left: 50%;  
 transform: translateX(-50%);  
 border: 12px solid #fff;  
 border-left-color: transparent;  
 border-right-color: transparent;  
 border-bottom: none;  
 z-index: 9;  
 }  
 &:hover {  
 &::before {  
 display: block;  
 }  
 &::after {  
 display: block;  
 }  
 }  
}  
  
// стили ссылки сброса фильтров списков  
.reset-filter {  
 cursor: pointer;  
 text-decoration: underline;  
 font-size: 13px;  
 margin-top: -47px;  
 margin-right: 86px;  
 float: right;  
}  
  
// заголовок контента главного региона  
.main-region {  
 &\_\_title {  
 font-size: 28px;  
 font-weight: bold;  
 color: #58647b;  
 margin-bottom: 34px;  
 margin-top: 6px;  
 display: inline-block;  
 }  
 &\_\_section-title{  
 font-size: 20px;  
 font-weight: bold;  
 color: #58647b;  
 margin-bottom: 34px;  
 margin-top: 6px;  
 display: inline-block;  
 }  
}  
  
.x-editor-page {  
 padding: 0 **!important**;  
 form {  
 padding: 0 **!important**;  
 }  
}  
  
.tile {  
 border-radius: 8px;  
 background-color: white;  
 box-shadow: 0 25px 50px -25px rgba(0, 38, 116, 0.28);  
 padding: 24px 24px;  
}  
  
.blue-button, button[name='yes'] {  
 border-radius: 8px **!important**;  
 background-color: @soft-blue **!important**;  
 color: @white **!important**;  
 border: 0 **!important**;  
 margin-right: 32px;  
 padding: 13px 24px **!important**;  
 &:hover {  
 background-color: #78a2f5 **!important**;  
 }  
 &\_\_text, span {  
 display: block;  
 }  
}  
  
.white-button {  
 border-radius: 8px **!important**;  
 background-color: @white **!important**;  
 color: @gunmetal **!important**;  
 border: 1px solid @cloudy-blue **!important**;  
 margin-right: 32px;  
 padding: 8px 16px **!important**;  
 &:hover {  
 color: @soft-blue **!important**;  
 border: 1px solid #78a2f5 **!important**;  
 box-shadow: 0 2px 4px 0 rgba(110, 147, 224, 0.4);  
 }  
 &\_\_text, span {  
 display: block;  
 }  
}  
  
.colorless-button, button[name='no'] {  
 border-radius: 8px **!important**;  
 background-color: transparent **!important**;  
 border: 0 **!important**;  
 color: @gunmetal **!important**;  
 margin-right: 32px;  
 padding: 8px 16px **!important**;  
 &:hover {  
 }  
 &\_\_text, span {  
 display: block;  
 }  
}  
  
// в форме подачи заявления, подготовки решения об утв ПД, подачи заявки ЛРП скрою вкладки  
.application-editor, .application-preparing-decision, .BRP-application-editor {  
 .x-editor-tabs {  
 display: none;  
 }  
}  
  
// стили модальных диалоговых окон  
.modal {  
 display: table;  
 height: 100%;  
 width: 100%;  
 .modal-dialog {  
 display: table-cell;  
 vertical-align: middle;  
 min-width: 560px;  
 .modal-content {  
 margin: 0 auto;  
 height: inherit;  
 width: inherit;  
 border-radius: 8px;  
 .modal-header {  
 display: none; // пока без заголовков  
 background-color: white;  
 border-bottom: 0;  
 border-radius: 8px;  
 padding: 32px 32px 0 32px;  
 .modal-btn-close {  
 display: none;  
 }  
 h4 {  
 font-size: 20px;  
 font-weight: bold;  
 }  
 }  
 .modal-body {  
 padding: 32px 32px 16px 32px **!important**; //16px 32px 16px 32px !important; // пока без заголовков  
 }  
 .modal-footer {  
 background-color: white **!important**;  
 border-top: 0 **!important**;  
 padding: 16px 32px 32px 32px;  
 border-radius: 8px;  
 }  
 .pull-right {  
 float: left **!important**;  
 }  
 .x-menu-item-default {  
 font-weight: normal **!important**;  
 }  
 .blue-button, button[name='yes'] {  
 padding: 10px 16px **!important**;  
 }  
 }  
 }  
}  
  
.disabled{  
 a {  
 opacity: 0.65;  
 }  
}  
  
.menu-actions {  
 &\_\_ico {  
 content: url(../../../content/img/more.svg);  
 &:hover {  
 content: url(../../../content/img/more-blue.svg);  
 }  
 }  
 &\_\_action-item {  
 height: 40px;  
 padding: 10px 0;  
 }  
 &\_\_svg {  
 margin-right: 16px;  
 }  
 .dropdown-menu {  
 border-radius: 8px;  
 a {  
 color: @gunmetal **!important**;  
 &:hover {  
 background-color: @pale-grey-two **!important**;  
 }  
 }  
 }  
}  
  
// стили для свойств объектов  
.prop {  
 &\_\_label {  
 &\_\_bold {  
 font-weight: bold;  
 }  
 font-size: 13px;  
 font-weight: bold;  
 color: @blue-grey;  
 margin-top: 24px;  
 &\_\_no-margin-top {  
 margin-top: 0;  
 }  
 &\_\_gun-metal-15px {  
 color: @gunmetal;  
 font-size: 15px;  
 }  
 }  
 &\_\_value {  
 .x-pe-viewonly {  
 font-style: normal **!important**;  
 }  
 &.empty {  
 color: @light-grey-blue;  
 }  
 .x-pe{  
 padding: 0 **!important**;  
 }  
 line-height: 2em;  
 padding: 0 **!important**;  
 font-size: 14px;  
 font-style: normal **!important**;  
 color: @gunmetal;  
 }  
}  
  
.step-title {  
 font-size: 20px;  
 font-weight: bold;  
 color: @slate;  
 margin: 16px 0 24px 0;  
 display: block;  
}  
  
.blue-title {  
 font-size: 18px;  
 font-weight: bold;  
 margin-top: 40px;  
 color: @soft-blue;  
 padding: 7px 0;  
 border-bottom: 1px solid @light-blue-grey;  
}  
  
.applicant\_\_avatar {  
  
}  
  
.applicant\_\_fio-container {  
 vertical-align: top;  
 width: calc(100% - 56px);  
}  
  
.applicant\_\_fio-value {  
 margin-left: 8px;  
 font-size: 16px;  
 font-weight: bold;color: @slate;  
 div {  
 font-style: normal **!important**;  
 margin-top: 2px;  
 }  
}  
  
.blue-gray-color {  
 color: @blue-grey **!important**;  
}  
  
.margin-left-8 {  
 margin-left: 8px;  
}  
  
// все ссылки  
a {  
 color: @soft-blue;  
 &:hover {  
 text-decoration: underline;  
 color: @soft-blue;  
 cursor: pointer;  
 }  
}  
  
// заголовок для правой плитки в процессинге заявлений  
.app-title-right-block {  
 font-size: 20px;  
 font-weight: bold;  
 color: @soft-blue;  
}  
  
  
// стили для списков с опцией "показать все"  
.collapsing-list{  
 &.collapse {  
 overflow: hidden;  
 }  
 &.collapse{  
 &:not(.in) {  
 display: block;  
 height: auto **!important**;  
 max-height: 1251px **!important**;  
 }  
 &.in {  
 overflow: visible;  
 }  
 }  
 &.collapsing {  
 min-height: 1251px;  
 }  
}  
  
a[data-toggle="collapse"] {  
 text-decoration: underline;  
 text-decoration-style: dotted;  
 display: block;  
 padding: 10px 0 10px 32px;  
 background-color: white;  
 border-radius: 0 0 8px 8px;  
 margin-top: -4px;  
 &.collapsed:after {  
 content: 'Развернуть';  
 }  
 &:not(collapsed):after {  
 content: 'Свернуть';  
 }  
}  
  
.gray-container {  
 .x-list {// список на основе slick-grid плохо работает с flex, поэтому задам ширину явно  
 width: calc(100vw - 598px - 90px);  
 }  
 .low-width-list {  
 width: calc(100% - 340px);  
 .x-list {// список на основе slick-grid плохо работает с flex, поэтому задам ширину явно  
 width: calc(100vw - 600px - 90px - 340px); // список с доп местом справа  
 }  
 }  
}  
  
.complex-project-category {  
 font-size: 13px;  
 font-weight: normal;  
 font-style: normal;  
 font-stretch: normal;  
 line-height: 1.23;  
 letter-spacing: normal;  
 color: @blue-grey  
}  
  
.complex-project-finance-source {  
 font-size: 14px;  
 font-weight: 600;  
 line-height: 2;  
 color: @gunmetal  
}  
  
// класс ячейки списка, контент внутри которой по середине  
.flex-cell {  
 display: flex;  
 flex-direction: column;  
 justify-content: center;  
 width: 100%;  
 height: 100%;  
 &\_\_a{  
 white-space: nowrap;  
 text-overflow: ellipsis;  
 overflow-x: hidden;  
 display: block;  
 }  
 &\_\_div {  
 white-space: nowrap;  
 text-overflow: ellipsis;  
 overflow-x: hidden;  
 }  
}  
  
// стили для блока контекстного меню объекта списка  
.actions {  
 .menu-actions {  
 position: absolute;  
 top: 5px;  
 right: 20px;  
 ul {  
 position: relative **!important**;  
 }  
 }  
}  
  
.margin-top-32px {  
 margin-top: 32px;  
}  
  
.margin-top-24px {  
 margin-top: 24px;  
}  
  
.margin-top-16px {  
 margin-top: 16px;  
}  
  
// горизонтальная серая линия  
.horizontal-gray-1px-line {  
 border-top: 1px solid @light-blue-grey;  
}  
  
// подсказка в контролле массовой загрузки файлов  
.multibinary-info {  
 width: 14px;  
 height: 14px;  
 background-color: @light-blue-grey;  
 border-radius: 50%;  
 float: right;  
 &\_\_content {  
 margin-top: -2px;  
 &::before {  
 width: 270px;  
 }  
 }  
}  
  
// класс для row что внутри него все колонки были одной высоты  
.equal{  
 display: -webkit-box;  
 display: -moz-box;  
 display: -ms-flexbox;  
 display: -webkit-flex;  
 display: flex;  
 flex:1 0 auto;  
 div[class\*='col-'] {  
 display: -webkit-box;  
 display: -moz-box;  
 display: -ms-flexbox;  
 display: -webkit-flex;  
 display: flex;  
 }  
}  
  
.hidden {  
 display: none;  
}

**4. Заключение**

В ходе прохождения практики были рассмотрены современные подходы в разработке больших систем. Изучены вышеописанные технологии.

**5. Список литературы**

1.Сайт КРОК <https://www.croc.ru>

2. Документация Spring <https://spring.io>

3. Документация JXFW, WEBCLIENT хранится на внутренних конфиденциальных ресурсах.

4. Документация Vaadin <https://vaadin.com/framework>

5. Документация Hybernate <http://hibernate.org>