|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования **«МИРЭА − Российский технологический университет»**  **РТУ МИРЭА** |

**Институт информационных технологий (ИИТ)**

**Кафедра математического обеспечения и стандартизации информационных технологий**

**(МОСИТ)**

**ОТЧЁТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ**

**Преддипломная практика**

приказ Университета о направлении на практику от «08» апреля 2021 г. № 1663-С

|  |  |  |  |
| --- | --- | --- | --- |
| Отчет представлен к  рассмотрению:  Студент группы ИКБО-12-17 | «\_\_» мая 2021 | (подпись) | Бузыкин И.В. |
| Отчет утвержден.  Допущен к защите: |  |  |  |
| Руководитель практики от кафедры | «\_\_» мая 2021 | (подпись) | Данилкин Ф.А. |

Москва 2021 г.

|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования **«МИРЭА − Российский технологический университет»**  **РТУ МИРЭА** |

**Институт информационных технологий (ИИТ)**

**Кафедра математического обеспечения и стандартизации информационных технологий**

**(МОСИТ)**

**ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ НА ПРОИЗВОДСТВЕННУЮ ПРАКТИКУ**

**Преддипломная практика**

**Студенту 4 курса учебной группы ИКБО-12-17**

**Бузыкину Игорю Валерьевичу**

**Место и время практики:** РТУ МИРЭА, кафедра МОСИТ, с 20 апреля 2021 г. по 17 мая 2021 г.

**Должность на практике:**  студент

**1. ЦЕЛЕВАЯ УСТАНОВКА:** Получение первичных профессиональных умений и опыта профессиональной деятельности по теме: «Создание инструментов для работы с измеряемыми величинами».

**2. СОДЕРЖАНИЕ ПРАКТИКИ:**

2.1. Изучить: принципы работы HTML, TypeScript, KnockoutJS

2.2. Практически выполнить: создать прототип формы для ввода конвертируемых значений

2.3. Ознакомиться: сборщиком проектов Parcel

**3. ДОПОЛНИТЕЛЬНОЕ ЗАДАНИЕ:** изучить работу GIT в крупных проектах и командах

**4. ОРГАНИЗАЦИОННО-МЕТОДИЧЕСКИЕ УКАЗАНИЯ:** пройти инструктаж по охране труда, технике безопасности, пожарной безопасности и правилам внутреннего распорядка Университета, в течение всей научно-исследовательской работы осуществлять устойчивое взаимодействие с научным руководителем.

|  |  |  |  |
| --- | --- | --- | --- |
| **СОГЛАСОВАНО:**  Заведующий кафедрой: | | | |
| «20» апреля 2021 г. | | *(подпись)* | (Головин С. А.) |
|  |  |  |  |
| Руководитель практики от кафедры  «20» апреля 2021 г. | | *(подпись)* | (Данилкин Ф.А.) |
| Задание получил |  |  |  |
| «20» апреля 2021 г. |  | *(подпись)* | (Бузыкин И.В.) |

|  |  |  |
| --- | --- | --- |
| **Проведенные инструктажи:** |  |  |
| Охрана труда: |  | «20» апреля 2021 г. |
| Инструктирующий | *(подпись)* | Данилкин Ф. А., д.т.н., профессор кафедры МОСИТ |
| Инструктируемый | *(подпись)* | Бузыкин И.В. |
|  |  |  |
| Техника безопасности: |  | «20» апреля 2021 г. |
| Инструктирующий | *(подпись)* | Данилкин Ф. А., д.т.н., профессор кафедры МОСИТ |
| Инструктируемый | *(подпись)* | Бузыкин И.В. |
|  |  |  |
| Пожарная безопасность: |  | «20» апреля 2021 г. |
| Инструктирующий | *(подпись)* | Данилкин Ф. А., д.т.н., профессор кафедры МОСИТ |
| Инструктируемый | *(подпись)* | Бузыкин И.В. |
|  | |  |
| С правилами внутреннего распорядка ознакомлен: | | «20» апреля 2021 г. |
|  | *(подпись)* | Бузыкин И.В. |

|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования **«МИРЭА − Российский технологический университет»**  **РТУ МИРЭА** |

**РАБОЧИЙ ГРАФИК ПРОВЕДЕНИЯ  
ПРОИЗВОДСТВЕННОЙ ПРАКТИКИ**

студента Бузыкина И.В. 4 курса группы ИКБО-12-17 очной формы обучения, обучающегося по направлению подготовки 09.03.04 «Программная инженерия», профиль «Системная и программная инженерия»

|  |  |  |  |
| --- | --- | --- | --- |
| **Неделя** | **Сроки**  **выполнения** | **Этап** | **Отметка о выполнении** |
| 1 | 20.04.2021 | Подготовительный этап, включающий в себя организационное собрание |  |
| 1-2 | 21.04.2021-01.05.2021 | Проведение анализа средств разработки клиентской части |  |
| 2-3 | 02.05.2021-06.05.2021 | Разработка конвертера |  |
| 3-4 | 07.05.2021-  11.05.2021 | Разработка клиентской части приложения |  |
| 4 | 12.05.2021-  13.05.2021 | Отладка взаимодействия интерфейса и конвертера |  |
| 4 | 14.05.2021 | Представление руководителю предварительной версии отчета с обеспечением согласованности материала по всем его частям |  |
| 4 | 17.05.2021 | Подготовка окончательной версии отчета по практике |  |

Руководитель практики от  
кафедры \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ Данилкин Ф.А., д.т.н., профессор /

Обучающийся \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ Бузыкин И.В. /

**Согласовано:**

Заведующий кафедрой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ Головин С. А., д.т.н., профессор /

**ОТЧЁТ**

**по производственной практике**

**студента 4 курса учебной группы ИКБО-12-17 института Университета**

Бузыкина Игоря Валерьевича

1. Практику проходил с 20.04.2021 г. по 17.05.2021 г.. в\_\_ФГБОУ ВО «МИРЭА – Российский

технологический университет», на кафедре математического обеспечения и стандартизации информационных технологий, студент

*(место прохождения практики и должность)*

2. Задание на практику выполнил

в полном объеме

*(указать: в полном объеме или частично)*

Не выполнены следующие задания:

---------------

*(указать также причины невыполнения)*

Подробное содержание выполненной на практике работы и достигнутые результаты: Проведен анализ предметной области, а также составлен отчет \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

Предложения по совершенствованию организации и прохождения практики:

предложений нет

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Бузыкин И.В.)

*(подпись)*

«17» мая 2021

Заключение руководителя практики

Приобрел следующие профессиональные навыки: студент продемонстрировал профессиональные умения и навыки, знание и понимание прикладной области, задач, требующих решения в прикладной области, современные подходы и средства решения прикладных задач разных классов, умение находить и работать с различными источниками информации по профессиональной деятельности, структурировать отчет с учетом тематики исследования

Бузыкин И.В. проявил себя как: дисциплинированный ответственный специалист: соблюдал сроки календарного графика практики, регулярно отчитывался о проделанных этапах работ; за срок прохождения практики не получил ни одного замечания - проявляет инициативу, четко и в определенные сроки выполняет задания; в любой ситуации уважителен в общении с другими

«17» мая 2021

**Отчет проверил:**

**Руководитель практики от Университета**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(Данилкин Ф.А.)

*(подпись)*

**Оглавление**

[Введение 2](#_Toc72152199)

[Инструменты и технологии для разработки клиентской части сайта 3](#_Toc72152200)

[CSS/HTML 3](#_Toc72152201)

[TypeScript 5](#_Toc72152202)

[Git 7](#_Toc72152203)

[Knockout 8](#_Toc72152204)

[Parcel 10](#_Toc72152205)

[Проектирование системы 12](#_Toc72152206)

[Предложение о решении задач 12](#_Toc72152207)

[Реализация конвертера 13](#_Toc72152208)

[Реализация графического интерфейса (viewModel) 17](#_Toc72152209)

[Заключение 20](#_Toc72152210)

[Список информационных источников 22](#_Toc72152211)

# Введение

В современном мире все информационные системы взаимодействуют с базами данных (далее - БД) и не все пользователи этих информационных систем знакомы с программированием, поэтому необходимо предоставить пользователям интерфейс, который позволяет взаимодействовать с БД.

Abris — это платформа разработки приложений для создания веб-интерфейсов для БД PostgreSQL. Платформу Abris можно использовать для быстрого создания приложений с удобными формами через декларативное описание SQL.

На рынке есть достаточное количество продуктов, позволяющих взаимодействовать с БД, но все они имеют определенные ограничения в вопросах управления данными с измеряемыми величинами

Таким образом, программный продукт, реализующий возможность взаимодействия с БД для простых пользователей и позволяющий без особых усилий переводить одни величины в другие, существенно облегчает и ускоряет взаимодействие с большим объемом информации для предприятий, в которых используются информационные системы.

Целью практической является анализ методов взаимодействия пользователя, не владеющего навыками программирования и администрирования, с базой данных и возможность без особых усилий конвертировать сложные данные в различные величины. По результатам анализа необходимо ознакомиться с технологиями, позволяющие реализовать рабочий прототип, для взаимодействия пользователя с интерфейсом, позволяющим вносить, редактировать, взаимодействовать с данными в БД.

После ознакомления реализовать прототип.

# Инструменты и технологии для разработки модуля клиентской части платформы

## CSS/HTML

Рука об руку с JavaScript идут CSS и HTML - вместе они составляют святую троицу разработки веб-интерфейса. HTML (Hyper Text Mark Up Language) - это язык веб-браузеров - с помощью которых сделаны сайты. CSS (каскадные таблицы стилей) заставляет их выглядеть стильно и со вкусом. Для разработчиков веб-интерфейсов очень важно знать эти инструменты от начала и до конца. Они также окажутся полезными для backend разработчиков.

Стили являются удобным, практичным и эффективным инструментом при вёрстке веб-страниц и оформления текста, ссылок, изображений и других элементов. Несмотря на явные плюсы применения стилей, рассмотрим все преимущества CSS, в том числе и незаметные на первый взгляд.

Достоинств у такого решения было несколько:

1. Разграничение кода и оформления

Идея о том, чтобы код HTML был свободен от элементов оформления вроде установки цвета, размера шрифта и других параметров. В идеале, веб-страница должна содержать только теги логического форматирования, а вид элементов задаётся через стили. При подобном разделении работа над дизайном и версткой сайта может вестись параллельно.

1. Разное оформление для разных устройств

С помощью стилей можно определить вид веб-страницы для разных устройств вывода: монитора, принтера, смартфона, планшета и др. Например, на экране монитора отображать страницу в одном оформлении, а при её печати — в другом. Эта возможность также позволяет скрывать или показывать некоторые элементы документа при отображении на разных устройствах.

1. Расширенные по сравнению с HTML способы оформления элементов

В отличие от HTML стили имеют гораздо больше возможностей по оформлению элементов веб-страниц. Простыми средствами можно изменить цвет фона элемента, добавить рамку, установить шрифт, определить размеры, положение и многое другое.

1. Ускорение загрузки сайта

При хранении стилей в отдельном файле, он кэшируется и при повторном обращении к нему извлекается из кэша браузера. За счёт кэширования и того, что стили хранятся в отдельном файле, уменьшается код веб-страниц и снижается время загрузки документов. Кэшем называется специальное место на локальном компьютере пользователя, куда браузер сохраняет файлы при первом обращении к сайту. При следующем обращении к сайту эти файлы уже не скачиваются по сети, а берутся с локального диска. Такой подход позволяет существенно повысить скорость загрузки веб-страниц.

1. Единое стилевое оформление множества документов

Сайт это не просто набор связанных между собой документов, но и одинаковое расположение основных блоков, и их вид. Применение единообразного оформления заголовков, основного текста и других элементов создает преемственность между страницами и облегчает пользователям работу с сайтом и его восприятие в целом. Разработчикам же использование стилей существенно упрощает проектирование дизайна.

1. Централизованное хранение

Стили, как правило, хранятся в одном или нескольких специальных файлах, ссылка на которые указывается во всех документах сайта. Благодаря этому удобно править стиль в одном месте, при этом оформление элементов автоматически меняется на всех страницах, которые связаны с указанным файлом. Вместо того чтобы модифицировать десятки HTML-файлов, достаточно отредактировать один файл со стилем и оформление нужных документов сразу же поменяется.

## TypeScript

TypeScript — это язык программирования, в котором исправлены многие недостатки JavaScript. Код на TypeScript выглядит почти так же, как и код на JS, и, если есть опыт frontend-разработки, изучить TypeScript достаточно просто. Особенно учитывая, что можно писать JS-код прямо в TS-скриптах.

Код на TypeScript компилируется в JS и подходит для разработки любых проектов под любые браузеры — тем более что можно выбрать версию JS, в которую будет компилироваться код.

TypeScript — проект с открытым исходным кодом, поэтому он очень быстро развивается. Многое, что появляется в TS, позже переходит и в JavaScript: например, let и const, стрелочные функции и так далее.

Необходимо разобрать основные преимущества TS перед JS:

* Многие проблемы в JavaScript появляются из-за динамической типизации и в целом странного поведения типов данных. В TypeScript типизация статическая, что избавляет от множества проблем. Есть числовой тип, строковый, логический и другие.
* И в JS, и в TS есть поддержка объектно-ориентированного программирования: классы, объекты, наследование. Однако TypeScript шагнул чуть дальше и использует больше возможностей ОПП. В том числе, например, интерфейсы.

Также есть и другие возможности:

* определение полей в конструкторе;
* преобразование типов;
* абстрактные классы;
* обобщение.

В будущем всё это может появиться и в JavaScript, но браузеры начнут поддерживать такие возможности ещё очень нескоро.

Недостатки TS:

Разработчики любят этот язык, а некоторые крупные проекты уже переходят на него. Например, популярный фреймворк Angular.JS. Но этого всё равно недостаточно, чтобы он стал таким же востребованным, как JavaScript. Это связано с тем, что разработка веб-приложения на TypeScript стоит дороже и отнимает больше времени.

Особенно если необходимо использовать какую-нибудь библиотеку или фреймворк, которые не портированы на TS. В этом случае разработчикам придётся самостоятельно описывать сигнатуры (указывать типы данных) всех функций и методов — достаточно длительный процесс, учитывая размеры современных библиотек.

Также порог входа в TypeScript выше — чтобы использовать его преимущества, важно знать типы данных и объектно-ориентированное программирование.

## Git

Git — распределённая система управления версиями.

Система спроектирована как набор программ, специально разработанных с учётом их использования в сценариях. Это позволяет удобно создавать специализированные системы контроля версий на базе Git или пользовательские интерфейсы.

Удалённый доступ к репозиториям Git обеспечивается git-демоном, SSH- или HTTP-сервером. TCP-сервис git-daemon входит в дистрибутив Git и является наряду с SSH наиболее распространённым и надёжным методом доступа. Метод доступа по HTTP, несмотря на ряд ограничений, очень популярен в контролируемых сетях, потому что позволяет использовать существующие конфигурации сетевых фильтров.

Ядро Git представляет собой набор утилит командной строки с параметрами. Все настройки хранятся в текстовых файлах конфигурации. Такая реализация делает Git легко портируемым на любую платформу и даёт возможность легко интегрировать Git в другие системы (в частности, создавать графические git-клиенты с любым желаемым интерфейсом).

Репозиторий Git представляет собой каталог файловой системы, в котором находятся файлы конфигурации репозитория, файлы журналов, хранящие операции, выполняемые над репозиторием, индекс, описывающий расположение файлов, и хранилище, содержащее собственно файлы. Структура хранилища файлов не отражает реальную структуру хранящегося в репозиторием файлового дерева, она ориентирована на повышение скорости выполнения операций с репозиторием. Когда ядро обрабатывает команду изменения (неважно, при локальных изменениях или при получении патча от другого узла), оно создаёт в хранилище новые файлы, соответствующие новым состояниям изменённых файлов. Существенно, что никакие операции не изменяют содержимого уже существующих в хранилище файлов.

По умолчанию репрозиторий хранится в подкаталоге с названием «.git» в корневом каталоге рабочей копии дерева файлов, хранящегося в репрозитории. Любое файловое дерево в системе можно превратить в репрозиторий git, отдав команду создания репозитория из корневого каталога этого дерева (или указав корневой каталог в параметрах программы). Репозиторий может быть импортирован с другого узла, доступного по сети. При импорте нового репозитория автоматически создаётся рабочая копия, соответствующая последнему зафиксированному состоянию импортируемого репозитория (то есть не копируются изменения в рабочей копии исходного узла, для которых на том узле не была выполнена команда commit).

## Knockout

KnockoutJS представляет собой библиотеку на языке JavaScript, которая реализует паттерн MVVM (Model - View - ViewModel) и предназначена для создания функциональных пользовательских интерфейсов на JavaScript и HTML.

Первый релиз KnockoutJS вышел в июле 2010 года. Автором фреймворка является работник Microsoft Стив Сандерсон (Steve Sanderson). Хотя официально библиотека не является продуктом компании Microsoft.

По некоторым данным в середине 2016 года KnockoutJS использовался более чем на 41000 сайтов.

Официальным сайтом библиотеки является <http://knockoutjs.com/>, где можно, собственно, загрузить файл библиотеки, найти сопроводительные справочные материалы и посмотреть прочую сопутствующую информацию.

Основная концепция библиотеки заключается в том, что библиотека позволяет автоматически отслеживать изменения в интерфейсе и в соответствии с изменениями в элементах интерфейса синхронизировать используемый источник данных. Иначе говоря, KnockoutJS позволяет простой подход к использованию привязки данных.

KnockoutJS имеет следующие особенности:

* Элегантное отслеживание зависимостей: библиотека автоматически обновляет элементы пользовательского интерфейса при изменениях в привязанном источнике данных
* Декларативный способ привязки: библиотека позволяет декларативным способом в коде HTML привязывать элементы пользовательского интерфейса к источнику данных
* Расширяемость: при необходимости можно легко добавить кастомное поведение к уже имеющимся возможностям библиотеки

Кроме того, библиотека обладает рядом дополнительных преимуществ:

* Она написана на чистом JavaScript и поэтому работает с любой технологией серверного уровня и на множестве клиентов
* Для ее добавления не надо вносить существенных изменений в существующий код веб-страниц
* Компактность и легковесность - она весит всего 13 кБ после сжатия
* Поддержка большинства браузеров - от самых последних до довольно старых (IE 6+, Firefox 2+, Chrome, Safari, Edge)

Поскольку библиотека реализует паттерн MVVM, то применение Knockout вовлекает три компонента:

* представление (view) - код HTML-элементов, к которым привязываются данные
* модель (model) - сами привязываемые данные
* ViewModel - является посредником между представлением и данными, осуществляя между ними привязку

## Parcel

Практика подразумевает под собой создание «Прототипа» формы для работы с величинами, без интеграции её в платформы Abris, чтобы быстро и удобно реализовать такой прототип, необходимо использовать сборщик проекта. В практики выбор остановился на Parcel.

[Parcel](https://parceljs.org/) — маленький и быстрый бандлер, позиционируется как решение для маленьких проектов. Согласно официальной документации, имеет следующие плюсы:

**Быстрая сборка**

Parcel использует worker process для многопоточной сборки, а также имеет свой файловый кэш для быстрой пересборки при последующих изменениях.

**Собирает все ваши ассеты**

Из коробки имеется поддержка ES6, TypeScript, CoffeeScript, HTML, SCSS, Stylus, raw-файлов. Плагины не требуются.

**Автоматические преобразования**

Весь код автоматически проходит через Babel, PostCSS, PostHTML — подхватываются при необходимости из node\_modules.

**Разделение кода без лишней конфигурации**

Используя динамический import(), Parcel разделяет бандл для возможности быстрой начальной загрузки точки входа в приложение

**Горячая перезагрузка**

Типичный хот-релоад без конфигурации — сохраняете изменения и они автоматически применяются в браузере.

**Дружелюбный вывод ошибок**

При ошибке подсвечивается кусок кода, в котором она произошла.

Механика работы

У нас есть сущность — Asset. Ассет — это любой файл. Механика работы такова: реализуется интерфейс, который предоставляет логику для превращения файла в AST, разрешения всех зависимостей, применения нужных трансформаций и генерирования итогового кода.

Дальше в дело вступает Packager. Упаковщик склеивает ассеты в итоговый бандл. Это происходит после обработки и успешного построения дерева. Упаковщики регистрируются на основе типа файлов.

# Проектирование системы

## Предложение о решении задач

Проанализировав статистику по популярности использования технологий и имеющиеся технические ресурсы для разработки, предлагаю следующее решение:

* для frontend-разработки использовать язык разметки HTML, а также Typescript в связке с библиотекой KnockoutJS
* Использовать систему контроля версий GIT
* Для создания конвертера величин использовать Typescript

## Реализация конвертера

Было принято решение написать алгоритм перевода величин на языке TypeScript.

Конвертер представляет собой класс (ConvertedValue), внутри которого есть некоторые свойства и методы, которые я подробно опишу.

Свойства класса, с которыми взаимодействует пользователь:

* valueType – значение, которое устанавливает пользователь при инициализации объекта класса
* convertedSystemType – тип значения, которое передается строкой (прим. “kilograms”)

Свойства класса, для работы конвертера:

* valueTypeIsJSON – параметр, отвечающий за то, какое значение мы предаем (просто строка с типом или JSON)
* valueTypeAllJsonString - если в конвертер передан JSON, с типом данных, а не строка, то сюда перезаписывается то, что пользователь передал в valueType в виде JSON
* convertedSystemType – система величин, в которой можно конвертировать значение (в этот параметр определяется то, что мы конвертируем, скорость, время и т. д.)
* avalibleConvertRules - массив с доступными переводами в той системе, которая установлена в convertedSystemType (distance -> kilometers, meters, ...)

Методы класса, с которыми взаимодействует пользователь:

* getValue() – получить свойство valueType
* getValueType() – получить свойство convertedSystemType
* getValueAndType() – получить массивом свойства valueType и convertedSystemType
* getAvalibleConvertRules() – получить свойство avalibleConvertRules
* convert() – основной метод – вызывается для переконвертирования valueType

Методы класса, для работы конвертера:

* detectJson() - функция проверяет, что передает пользователь в параметр "valueType" - строку с типом или JSON. Если передаваемый valueType является JSON, то valueTypeIsJSON устанавливается, как true, далее в valueTypeAllJsonString – перезаписывается содержимое valueType, а в сам valueType перезаписывается тип значения, который был упомянут в JSON
* detectConvertedValueType() - определяем, к какой из величин относится convertedSystemType, и записываем доступные переводы в avalibleConvertRules
* rewriteMainData() - перезаписать данные в объекте класса
* convertMass() – конвертирование массы (метод вызывается автоматически в convert)
* convertDistance() - конвертирование дистанции (метод вызывается автоматически в convert)
* convertVolume() - конвертирование объем (метод вызывается автоматически в convert)
* convertArea() - конвертирование площадь (метод вызывается автоматически в convert)
* convertSpeed() - конвертирование скорость (метод вызывается автоматически в convert)
* convertTemperature() - конвертирование температуры (метод вызывается автоматически в convert)
* convertTime() - конвертирование времени (метод вызывается автоматически в convert)

Описание работы: создается экземпляр класса ConvertedValue, в конструктор которого передается значение и тип значения строкой или JSON. (В конвертер встроена защита от ввода мусора, если пропустить значение или указать тип значения, который не существует – будет выведена ошибка). После инициализации экземпляра можно вызвать метод convert, в который передается строкой тип значения и свойства valueType и convertedSystemType будут изменены, согласно передаваемому типу значения. После этого можно вызвать один из методов экземпляра, для получения интересующих свойств.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 1. Работа конвертера

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2. Работа конвертера

## Реализация графического интерфейса (viewModel)

Для реализации графического интерфейса необходимо выводить пользователю поле для ввода значения, селектор типов значения, а также блок, для округления дробных значений, состоящий из кнопки и поля ввода

Создадим HTML структуру и в теги создаваемых элементов устанавливаем атрибут data-bind, определяемый библиотекой knockout:

**Изображение выглядит как текст

Автоматически созданное описание**

Рисунок 3. Структура HTML

Поле для ввода значения

Свойства, определяемые атрибутом data-bind в теге <input>:

* valueField – свойство для отслеживания вводимого в него значения
* {change: valueFieldChangeEvent} – свойство отслеживает изменение поля
* valueFieldEnabler – свойство отслеживает HTML атрибут disabled

Выпадающий список для выбора типа значения

Свойства, определяемые атрибутом data-bind в теге <select>:

* availableOperationsArray – свойство отвечает за элементы, находящиеся в выпадающем списке <select>
* selectedValueType – свойство отслеживает выбранный элемент из выпадающего списка <select>
* {change: optionsChangeEvent} – свойство отслеживает изменение поля
* selectEnabler – свойство отслеживает HTML атрибут disabled

Блок для округления значений

Свойства, определяемые атрибутом data-bind в теге <input>:

* roundField – свойство для отслеживания вводимого в него значения
* roundFieldEnabler – свойство отслеживает HTML атрибут disabled
* roundFieldVisible - свойство отслеживает HTML атрибут visibility

Свойства, определяемые атрибутом data-bind в теге <button>:

* roundButtonClick – свойство отслеживает нажатие кнопки
* roundButtonEnabler – свойство отслеживает HTML атрибут disabled
* roundButtonVisible - свойство отслеживает HTML атрибут visibility

Теперь мы можем отслеживать все элементы, необходимые для реализации интерфейса.

Напишем обработчики событий для элементов формы

**Изображение выглядит как текст

Автоматически созданное описание**

Рисунок 4. Структура HTML

# Заключение

В данной практической работе были рассмотрены инструменты разработки веб-приложения, проведено проектирование информационной модели системы. Далее, на основе проведенного проектирования была разработана форма для конвертируемых данных.

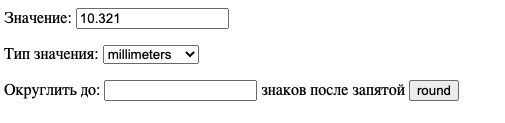


Рисунок 5. Интерфейс формы

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 6. Селектор типов

Изображение выглядит как текст, черный, снимок экрана

Автоматически созданное описание

Рисунок 7. Дерево проекта

# Список информационных источников

1. Официальный сайт Typescript – [Электронный источник] https://www.typescriptlang.org/ (дата обращения: 15.05.2021)
2. Официальный сайт KnockoutJS – [Электронный источник] https://knockoutjs.com/ (дата обращения: 15.05.2021)
3. Официальный сайт Parcel – [Электронный источник] https://ru.parceljs.org/ (дата обращения: 15.05.2021)
4. Введение в KnockoutJS - [Электронный источник] <https://metanit.com/web/knockout/1.1.php> (дата обращения: 15.05.2021)
5. Abris platform – [Электронный источник] https://abrisplatform.com/ (дата обращения: 15.05.2021)
6. Что такое веб-интерфейс – [Электронный источник] https://semantica.in/blog/veb-interfejs.html (дата обращения: 15.05.2021)
7. Создание Web-приложений – [Электронный источник] https://www.frolov-lib.ru/books/rusedit/web\_development/ch01.html (дата обращения: 15.05.2021)
8. Эволюция веб-приложений – [Электронный источник] https://habr.com/ru/post/218215/ (дата обращения: 15.05.2021)
9. Крокфорд Дуглас, «Как устроен JavaScript» / Москва: «Питер», 2019 – 304с