|  |
| --- |
| 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**

**ОТЧЁТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ**

**Практика по получению профессиональных умений и опыта профессиональной деятельности**

**Тема практики**: «Создание инструментов для работы с измеряемыми величинами на Javascript»

приказ Университета о направлении на практику от «05» февраля 2021 г. 296-С

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

**ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ НА ПРОИЗВОДСТВЕННУЮ ПРАКТИКУ**

**Практика по получению профессиональных умений и опыта профессиональной деятельности**

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

**Бузыкину Игорю Валерьевичу**

**Место и время практики:** РТУ МИРЭА, кафедра МОСИТ, с 09 февраля 2021 г. по 05 апреля 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 PostgreSQL, Bootstrap, HTML, CSS, TypeScript

2.2. Практически выполнить: создание формы для ввода информации в БД

2.3. Ознакомиться: с принципами создания пользовательского интерфейса

**3. ДОПОЛНИТЕЛЬНОЕ ЗАДАНИЕ:** изучить возможности 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.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Заведующий кафедрой: | | | | | |
| «09» февраля 2021 г. | | | *(подпись)* | | (Головин С.А.) |
| **СОГЛАСОВАНО:** | |  |  | |  |
| Руководитель практики от кафедры  «09» февраля 2021 г. | | | *(подпись)* | | (Данилкин Ф.А.) |
| Задание получил | |  |  | |  |
| «09» февраля 2021 г. | |  | *(подпись)* | | (Бузыкин И.В.) |
| **Проведенные инструктажи:** |  | | |  | |
| Охрана труда: |  | | | «09» февраля 2021 г. | |
| Инструктирующий | *(подпись)* | | | Данилкин Ф.А., кафедра МОСИТ | |
| Инструктируемый | *(подпись)* | | | Бузыкин И.В. | |
|  |  | | |  | |
| Техника безопасности: |  | | | «09» февраля 2021 г. | |
| Инструктирующий | *(подпись)* | | | Данилкин Ф.А., кафедра МОСИТ | |
| Инструктируемый | *(подпись)* | | | Бузыкин И.В. | |
|  |  | | |  | |
| Пожарная безопасность: |  | | | «09» февраля 2021 г. | |
| Инструктирующий | *(подпись)* | | | Данилкин Ф.А., кафедра МОСИТ | |
| Инструктируемый | *(подпись)* | | | Бузыкин И.В. | |
|  | | | |  | |
| С правилами внутреннего распорядка ознакомлен: | | | | «09» февраля 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 | 09.02.2021 | Выбор темы практики. Оформление задания на практику и рабочего графика. Прохождение инструктажей |  |
| 1-2 | 10.02.2021-20.02.2021 | Изучить принципы работы PostgreSQL, Bootstrap, HTML, CSS, TypeScript. |  |
| 3-4 | 22.02.2021-6.03.2021 | Проанализировать способы и правила различных величин системы СИ |  |
| 5-6 | 9.03.2021-  20.03.2021 | Проанализировать особенности разработки веб-клиента с использованием базы данных PostgreSQL |  |
| 7-8 | 22.03.2021-  04.04.2021 | Изучить возможности Docker и Git. Реализовать форму для заполнения данных |  |
| 9 | 05.04.2021 | Оформление отчета по ПД, зачет |  |

**Согласовано:**

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

Руководитель практики от  
кафедры \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ Данилкин Ф.А./

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

**ОТЧЁТ**

**по производственной практике**

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

Бузыкина Игоря Валерьевича

1. Практику проходил с 02.02.2021 г. по 07.04.2021 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)*

«07» апреля 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

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

«\_\_» апреля 2021

**Отчет проверил:**

**Руководитель практики от Университета**

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

**Оглавление**

[Введение 2](#_Toc68639973)

[Описание требований к системе 3](#_Toc68639974)

[Инструменты и технологии для разработки клиентской части сайта 5](#_Toc68639975)

[CSS/HTML 5](#_Toc68639976)

[TypeScript 7](#_Toc68639977)

[PostgreSQL 9](#_Toc68639978)

[Docker 16](#_Toc68639979)

[Git 21](#_Toc68639980)

[Проектирование системы 23](#_Toc68639981)

[Предложение о решении задач 24](#_Toc68639982)

[Заключение 25](#_Toc68639983)

[Список информационных источников 27](#_Toc68639984)

# 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 реализовать программный продукт, позволяющий пользователю взаимодействовать с интерфейсом, позволяющим вносить, редактировать, взаимодействовать с данными в БД.

# Описание требований к системе

**Требования к функциям системы**

Система должна предоставлять пользователю удобный интерфейс для работы со сложными данных.

**Требование к техническому обеспечению**

• Серверная часть;

• Клиентская часть.

Технические требования к серверной части:

• Процессор Intel Core 2 поколения или выше;

• Оперативная память 1 Гб;

• Место на жестком диске 3 Гб.

Точные технические характеристики сервера будут уточнены после завершения системы и обширного тестирования.

Технические требования к клиентской части:

• Intel Pentium 4 или более поздней версии с поддержкой SSE2;

• Свободное место на диске 350 Мб;

• Оперативная память 512 Мб.

**Требования к программному обеспечению**

На серверной части должны быть установлены:

• Операционная система Ubuntu 19 и выше

• Веб-сервер Apache 2.4.41 и выше;

• СУБД PostgreSQL 13.2 и выше;

На клиентской части должен быть установлен один из нижеперечисленных браузеров:

• Google Chrome;

• Firefox;

• Safari.

**Требования к информационному обеспечению**

Все данные системы должны храниться в структурированном виде под управлением реляционной СУБД. Исключения составляют файлы данных, предназначенные для просмотра и скачивания (изображения, видео, документы и т. п.). Такие файлы сохраняются в docker контейнере, а в БД размещаются ссылки на них.

Наполнение различных сайтов, функционирование которых поддерживается одной и той же инсталляцией системы, должно храниться под управлением единой СУБД.

**Требования к лингвистическому обеспечению**

Интерфейс пользователя представлен на русском языке.

# Инструменты и технологии для разработки клиентской части сайта

## 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.

## PostgreSQL

На данный момент рынок имеет огромное количество готовых продуктов для реализации БД. Abris базируется на PostgreSQL из-за ряда причин:

Модель данных

PostgreSQL не просто реляционная, а объектно-реляционная СУБД. Это даёт ему некоторые преимущества над другими SQL базами данных с открытым исходным кодом, такими как MySQL, MariaDB и Firebird.

Фундаментальная характеристика объектно-реляционной базы данных — это поддержка пользовательских объектов и их поведения, включая типы данных, функции, операции, домены и индексы. Это делает Постгрес невероятно гибким и надежным. Среди прочего он умеет создавать, хранить и извлекать сложные структуры данных. В некоторых примерах ниже вы увидите вложенные и составные конструкции, которые не поддерживаются стандартными РСУБД.

Структуры и типы данных

Существует обширный список типов данных, которые поддерживает Постгрес. Кроме числовых, с плавающей точкой, текстовых, булевых и других ожидаемых типов данных (а также множества их вариаций), PostgreSQL может похвастаться поддержкой uuid, денежного, перечисляемого, геометрического, бинарного типов, сетевых адресов, битовых строк, текстового поиска, xml, json, массивов, композитных типов и диапазонов, а также некоторых внутренних типов для идентификации объектов и местоположения логов. Справедливости ради стоит сказать, что MySQL, MariaDB и Firebird тоже имеют некоторые из этих типов данных, но только Постгрес поддерживает их все.

Сетевые адреса

PostgreSQL обеспечивает хранение разных типов сетевых адресов. Тип данных CIDR (бесклассовая маршрутизация интернет-домена, Classless Internet Domain Routing) следует соглашению для сетевых адресов IPv4 и IPv6.

Также для хранения сетевых адресов доступен тип данных INET, используемый для IPv4 и IPv6 хостов, где подсети являются необязательными. для идентификации оборудования, таких как 08-00-2b-01-02-03.

У MySQL и MariaDB тоже есть INET функции для конвертации сетевых адресов, но они не предоставляют типы данных для внутреннего хранения сетевых адресов. У Firebird тоже нет типов для хранения сетевых адресов.

Многомерные массивы

Поскольку Постгрес — это объектно-реляционная база данных, массивы значений могут храниться для большинства существующих типов данных. Сделать это можно путём добавления квадратных скобок к спецификации типа данных для столбца или с помощью выражения ARRAY. Размер массива может быть задан, но это необязательно. Давайте рассмотрим меню праздничного пикника для демонстрации использования массивов:

MySQL, MariaDB, и Firebird так не умеют. Чтобы хранить такие массивы значений в традиционных реляционных базах данных, придется использовать обходной путь и создавать отдельную таблицу со строками для каждого из значений массива.

Геометрические данные

Геоданные быстро становятся основным требованием для многих приложений. PostgreSQL уже давно поддерживает множество геометрических типов данных, таких как точки, линии, круги и многоугольники. Один из этих типов – PATH, он состоит из множества последовательно расположенных точек и может быть открытым (начальная и конечная точки не связаны) или закрытым (начальная и конечная точки связаны).

Расширение PostGIS для PostgreSQL дополняет существующие свойства геометрических данных вспомогательными пространственными типами, функциями, операторами и индексами. Оно обеспечивает поддержку местоположения и поддерживает как растровые, так и векторные данные. Оно также обеспечивает совместимость с множеством сторонних геопространственных инструментов (защищённых авторским правом и с открытым исходным кодом) для отображения, отрисовки и работы с данными.

Стоит заметить, что в MySQL 5.7.8 и в MariaDB, начиная с версии 5.3.3, были добавлены расширения типов данных для поддержки стандарта географической информации OpenGIS. Эта версия MySQL и последующие версии MariaDB предлагают хранение типов данных, аналогичное штатным геоданным Постгреса. Тем не менее, в MySQL и MariaDB значения данных сначала должны быть сконвертированы в геометрический формат простыми командами перед тем, как будут вставлены в таблицу. Firebird на данный момент не поддерживает геометрические типы данных.

Поддержка JSON

Поддержка JSON в PostgreSQL позволяет перейти к хранению schema-less данных в SQL базе данных. Это может быть полезно, когда структура данных требует определённой гибкости: например, если в процессе разработки структура всё ещё меняется или неизвестно, какие поля будет содержать объект данных.

Тип данных JSON обеспечивает проверку корректности JSON, который позволяет использовать специализированные JSON операторы и функции, встроенные в Постгрес для выполнения запросов и манипулирования данными. Также доступен тип JSONB — двоичная разновидность формата JSON, у которой пробелы удаляются, сортировка объектов не сохраняется, вместо этого они хранятся оптимальным образом, и сохраняется только последнее значение для ключей-дубликатов. JSONB обычно является предпочтительным форматом, поскольку требует меньше места для объектов, может быть проиндексирован и обрабатывается быстрее, так как не требует повторного синтаксического анализа.

В MySQL 5.7.8 и MariaDB 10.0.1 была добавлена поддержка встроенных объектов JSON. Но, хотя существует множество функций и операторов для JSON, которые теперь доступны в этих базах данных, они не индексируются так, как JSONB в PostgreSQL. Firebird пока что не присоединился к тренду и поддерживает объекты JSON только в виде текста.

Создание нового типа

Если вдруг так случится, что обширного списка типов данных Постгреса вам окажется недостаточно, вы можете использовать команду CREATE TYPE, чтобы создать новые типы данных, такие как составной, перечисляемый, диапазон и базовый.

Поскольку они не являются объектно-реляционными, MySQL, MariaDB и Firebird не предоставляют такую мощную функциональность.

Размеры данных

PostgreSQL может обрабатывать много данных. Текущие опубликованные ограничения перечислены ниже:

|  |  |
| --- | --- |
| Максимальный размер базы данных | Неограничен |
| Максимальный размер таблицы | 32 TB |
| Максимальный размер строки | 1.6 TB |
| Максимальный размер поля | 1 GB |
| Максимальное количество строк в таблице | Неограничено |
| Максимальное количество столбцов в таблице | 250-1600 в зависимости от типа столбца |
| Максимальное количество индексов в таблице | Неограничено |

Таблица 1. Ограничения данных PostgreSQL

В Compose мы автоматически масштабируем инсталляцию, чтобы не приходилось волноваться о росте количества данных. Но, как известно любому администратору баз данных, стоит с опаской относиться к слишком большим и неограниченным возможностям.

Для сравнения, MySQL и MariaDB печально известны ограничением размера строк в 65 535 байт. Firebird также предлагает всего лишь 64Кб в качестве максимального размера строки. Обычно объём данных ограничивается максимальным размером файлов операционной системы. Поскольку PostgreSQL умеет хранить табличные данные в множестве файлов меньшего размера, он может обойти это ограничение. Но стоит отметить, что слишком большое количество файлов может негативно сказаться на производительности. MySQL и MariaDB поддерживают большее количество столбцов в таблице (до 4,096 в зависимости от типа данных) и большие индивидуальные размеры таблицы, чем PostgreSQL, но необходимость превысить существующие ограничения Постгреса возникает лишь в крайне редких случаях.

Целостность данных

Постгрес стремится соответствовать стандарту ANSI-SQL:2008, отвечает требованиям ACID (атомарность, согласованность, изолированность и надежность) и известен своей ссылочной и транзакционной целостностью. Первичные ключи, ограничивающие и каскадные внешние ключи, уникальные ограничения, ограничения NOT NULL, проверочные ограничения и другие функции обеспечения целостности данных дают уверенность, что только корректные данные будут сохранены.

MySQL и MariaDB больше работают на то, чтобы соответствовать стандарту SQL с движками таблиц InnoDB/XtraDB. Теперь они предлагают опцию STRICT с использованием режимов SQL, которая устанавливает проверки корректности используемых данных. Несмотря на это, в зависимости от того, какой режим вы используете, недостоверные и даже урезанные без вашего ведома данные могут быть вставлены или созданы при обновлении. Ни одна из этих баз данных сейчас не поддерживает CHECK ограничения. Кроме того, у них существует множество особенностей в отношении ограничений ссылочной целостности по внешним ключам. В дополнение к вышесказанному целостность данных может существенно пострадать в зависимости от выбранного движка хранения. MySQL (и fork MariaDB) не делают секрета из того, что променяли целостность и соответствие стандартам на скорость и эффективность.

У Постгреса множество возможностей. Созданный с использованием объектно-реляционной модели, он поддерживает сложные структуры и широкий спектр встроенных и определяемых пользователем типов данных. Он обеспечивает расширенную ёмкость данных и заслужил доверие бережным отношением к целостности данных. Эти особенности привлекли разработчиков Abris и проект базируется на PostgreSQL, как на одном из самых удобных и гибких СУБД на рынке.

## Docker

Современные технологии позволяют в разы ускорить производство продуктов, одна из таких технологий – Docker. Что такое докер?  
Докер — это открытая платформа для разработки, доставки и эксплуатации приложений. Docker разработан для более быстрого выкладывания ваших приложений. С помощью docker можно отделить приложение от инфраструктуры и обращаться с инфраструктурой как управляемым приложением. Docker помогает выкладывать код быстрее, быстрее тестировать, быстрее выкладывать приложения и уменьшить время между написанием кода и запуска кода. Docker делает это с помощью легковесной платформы контейнерной виртуализации, используя процессы и утилиты, которые помогают управлять и выкладывать ваши приложения.

Изображение выглядит как текст, монитор, внутренний, экран

Автоматически созданное описание

Рисунок 1. Docker-dashboard интерфейс

В своем ядре docker позволяет запускать практически любое приложение, безопасно изолированное в контейнере. Безопасная изоляция позволяет запускать на одном хосте много контейнеров одновременно. Легковесная природа контейнера, который запускается без дополнительной нагрузки гипервизора, позволяет добиваться больше от железа.  
  
Платформа и средства контейнерной виртуализации могут быть полезны в следующих случаях:

* упаковывание приложения (и так же используемых компонент) в docker контейнеры;
* раздача и доставка этих контейнеров командам для разработки и тестирования;
* выкладывания этих контейнеров на продакшены, как в дата центры, так и в облака.

Для чего можно использовать использовать docker?

Быстрое выкладывание приложений

Docker прекрасно подходит для организации цикла разработки. Docker позволяет разработчикам использовать локальные контейнеры с приложениями и сервисами. Что в последствии позволяет интегрироваться с процессом постоянной интеграции и выкладывания (continuous integration and deployment workflow).  
  
Например, разработчики пишут код локально и делятся своим стеком разработки (набором docker образов) с коллегами. Когда они готовы, отравляют код и контейнеры на тестовую площадку и запускают любые необходимые тесты. С тестовой площадки они могут оправить код и образы на продакшен.

Более простое выкладывание и разворачивание

Основанная на контейнерах docker платформа позволят легко портировать полезную нагрузку. Docker контейнеры могут работать на локальной машине, как реальной, так и на виртуальной машине в дата центре, так и в облаке.

Портируемость и легковесная природа docker позволяет легко динамически управлять нагрузкой. Можно использовать docker, чтобы развернуть или погасить приложение или сервисы. Скорость docker позволяет делать это почти в режиме реального времени.

Высокие нагрузки и больше полезных нагрузок

Docker легковесен и быстр. Он предоставляет устойчивую, рентабельную альтернативу виртуальным машинам на основе гипервизора. Он особенно полезен в условиях высоких нагрузок, например, при создания собственного облака или платформа-как-сервис (platform-as-service). Но он так же полезен для маленьких и средних приложений, когда вам хочется получать больше из имеющихся ресурсов.

Главные компоненты Docker

Docker состоит из двух главных компонентов:

* Docker: платформа виртуализации с открытым кодом;
* Docker Hub: наша платформа-как-сервис для распространения и управления docker контейнерами.

Архитектура Docker

Docker использует архитектуру клиент-сервер. Docker клиент общается с демоном Docker, который берет на себя тяжесть создания, запуска, распределения ваших контейнеров. Оба, клиент и сервер могут работать на одной системе, вы можете подключить клиент к удаленному демону docker. Клиент и сервер общаются через сокет или через RESTful API.

Docker-демон  
Как показано на диаграмме, демон за пускается на хост-машине. Пользователь не взаимодействует с сервером на прямую, а использует для этого клиент.

Docker-клиент  
Docker-клиент, программа docker — главный интерфейс к Docker. Она получает команды от пользователя и взаимодействует с docker-демоном.

Образы  
Docker-образ — это read-only шаблон. Например, образ может содержать операционку Ubuntu c Apache и приложением на ней. Образы используются для создания контейнеров. Docker позволяет легко создавать новые образы, обновлять существующие, или вы можете скачать образы, созданные другими людьми. Образы — это компонента сборки docker-а.

Реестр  
Docker-реестр хранит образы. Есть публичные и приватные реестры, из которых можно скачать либо загрузить образы. Публичный Docker-реестр — это Docker Hub. Там хранится огромная коллекция образов. Как вы знаете, образы могут быть созданы вами или вы можете использовать образы, созданные другими. Реестры — это компонента распространения.

Контейнеры  
Контейнеры похожи на директории. В контейнерах содержится все, что нужно для работы приложения. Каждый контейнер создается из образа. Контейнеры могут быть созданы, запущены, остановлены, перенесены или удалены. Каждый контейнер изолирован и является безопасной платформой для приложения. Контейнеры — это компонента работы.

Докер – мощная утилита, на которой базируется Abris из-за функционала, описанного выше.

## 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).

# Проектирование системы

Модель бизнес-процесса проекта и ее декомпозиция в нотации IDEF0 представлены на рисунках 12 и 13 соответственно.

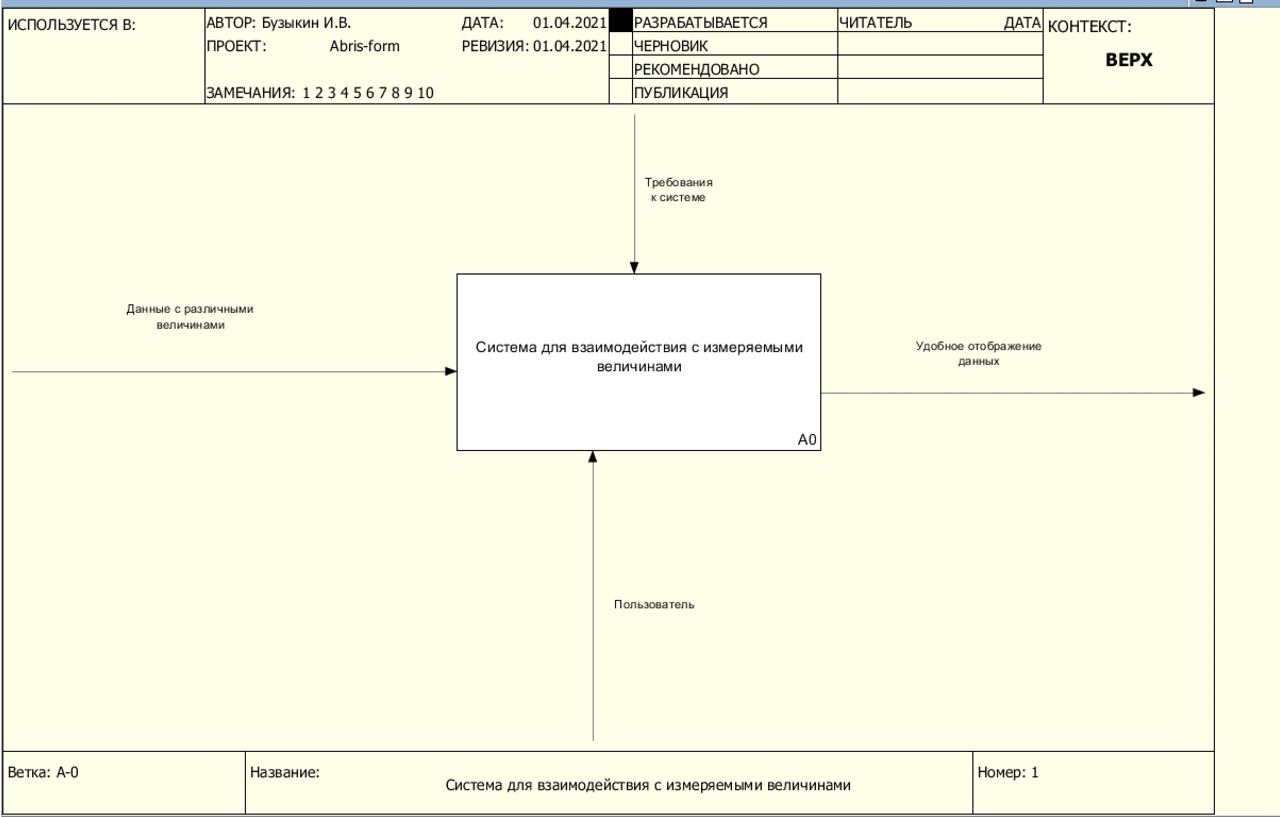


Рисунок 2. Модель бизнес-процесса проекта

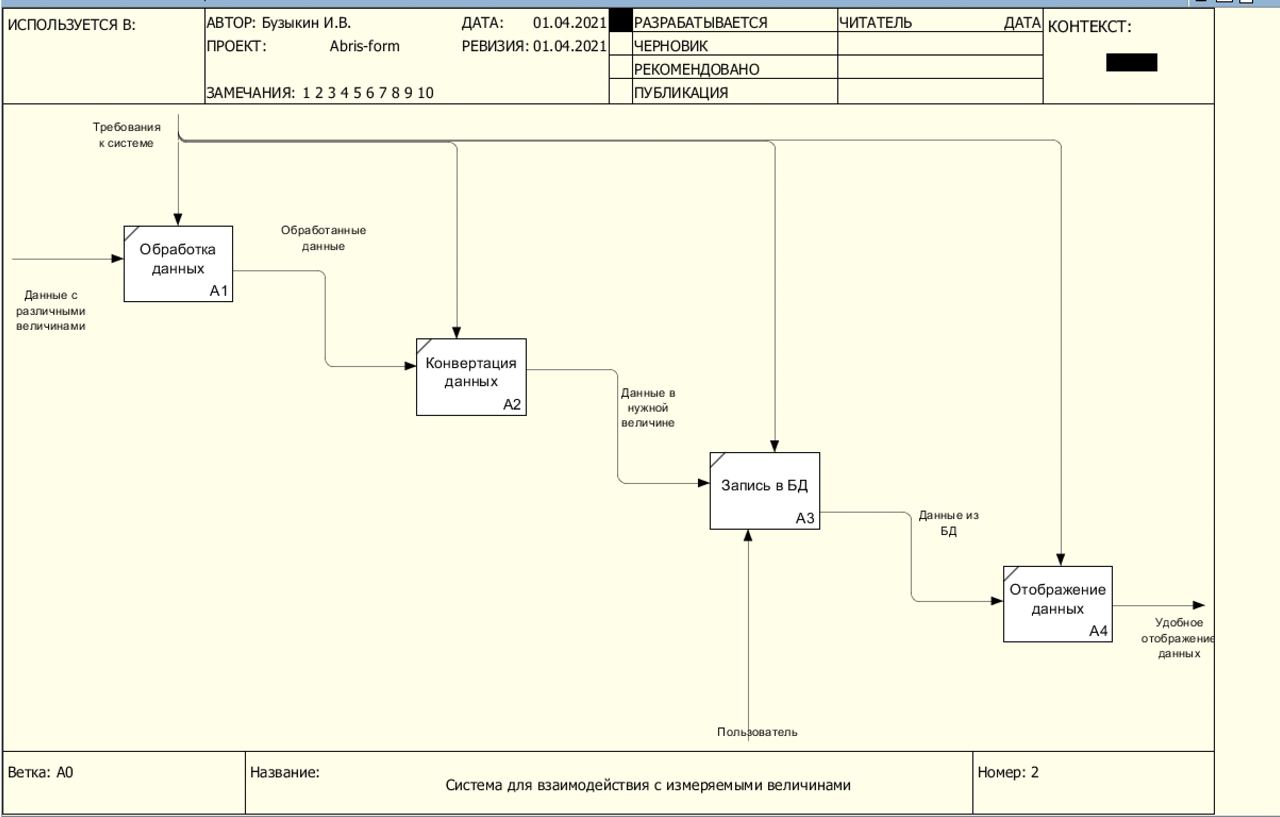


Рисунок 3. Декомпозиция модели бизнес-процесса проекта

## Предложение о решении задач

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

* в качестве виртуализации платформы используется Docker контейнер Abris
* для frontend-разработки использовать язык разметки HTML и язык описания внешнего вида документа с применением библиотеки bootstrap;
* база данных, выбранная платформой - реляционная, СУБД – PostgreSQL
* Обязательное использование GIT
* Для создания автоконвертера использовать JS и TS

# Заключение

В данной практической работе были рассмотрены инструменты разработки веб-приложения, проведено проектирование информационной модели системы. Далее, на основе проведенного проектирования была разработана форма заполнения и настроена платформа на локальном компьютере

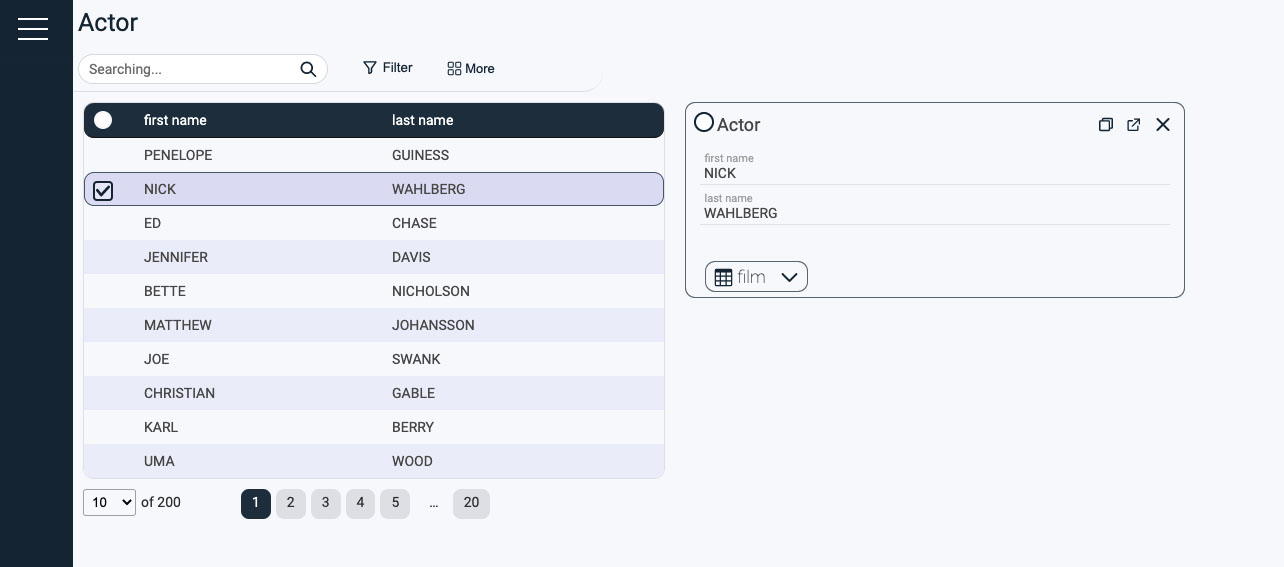


Рисунок 4. Интерфейс проекта

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 5. Дерево проекта

# Список информационных источников

1. Официальный сайт Docker – [Электронный источник] https://www.docker.com/ (дата обращения: 10.03.2021)
2. Базы данных – [Электронный источник] https://www.oracle.com/ru/database/what-is-database/ (дата обращения: 10.03.2021)
3. Abris platform – [Электронный источник] https://abrisplatform.com/ (дата обращения: 10.03.2021)
4. Docker – [Электронный источник] https://habr.com/ru/post/253877/ (дата обращения: 10.03.2021)
5. Что такое веб-интерфейс – [Электронный источник] https://semantica.in/blog/veb-interfejs.html (дата обращения: 10.03.2021)
6. Создание Web-приложений – [Электронный источник] https://www.frolov-lib.ru/books/rusedit/web\_development/ch01.html (дата обращения: 10.03.2021)
7. Чем PostgreSQL лучше других SQL баз данных – [Электронный источник] https://habr.com/ru/post/282764/ (дата обращения: 10.03.2021)
8. Эволюция веб-приложений – [Электронный источник] https://habr.com/ru/post/218215/ (дата обращения: 10.03.2021)
9. Крокфорд Дуглас, «Как устроен JavaScript» / Москва: «Питер», 2019 – 304с