|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | МИНОБРНАУКИ РОССИИ | | | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  РТУ МИРЭА | | |   Институт Информационных технологий | |
|  | |
| Кафедра Математического обеспечения и стандартизации информационных технологий | |
|  | |
|  | |

|  |  |
| --- | --- |
| **Автореферат** | |
| **по дисциплине** | |
| **«**Стандартизация системной и программной инженерии**»**  **Тема: «**Разработка инструментов на Javascript для работы с измеряемыми величинами**»** | |
|  | |
| Выполнил студент группы ИКБО-12-17 | Бузыкин И.В. |
| Научный руководитель | Данилкин Ф. А. |

Москва 2020

Оглавление

[Введение 3](#_Toc68290189)

[Актуальность 3](#_Toc68290190)

[Цель работы 3](#_Toc68290191)

[Новизна 3](#_Toc68290192)

[Глава 1 4](#_Toc68290193)

[Постановка задачи 4](#_Toc68290194)

[Целевая функция 4](#_Toc68290195)

[Требования 4](#_Toc68290196)

[Перечень задач выпускной квалификационной работы 5](#_Toc68290197)

[Анализ предметной области 5](#_Toc68290198)

[Сравнительная оценка своей разработки 30](#_Toc68290199)

[Перечень основных применяемых стандартов касающейся темы ВКР. 31](#_Toc68290200)

[Глава 2 32](#_Toc68290201)

[Проектирование системы 32](#_Toc68290202)

[Предложение о решении задач 33](#_Toc68290203)

[Глава 3 34](#_Toc68290204)

[Экспериментальные результаты 34](#_Toc68290205)

[Выводы 36](#_Toc68290206)

[Информационные источники 37](#_Toc68290207)

# Введение

## Актуальность

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 (далее – сложные данные)

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.

## Цель работы

Целью выпускной квалификационной работы 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.

## Новизна

Научная новизна выпускной квалификационной работы заключается в реализации системы, которая позволяет взаимодействовать с БД, не держа в голове меры различных величин и прочую информацию, не касающуюся работы непосредственно БД. В качестве операционной системы может выступать любая ОС, поддерживающая технологию Docker.

Проанализировав существующие решения, было выявлено, что на рынке недостаточно продуктов, с такими возможностями, в результате чего можно судить о научной новизне работы.

# Глава 1

## Постановка задачи

Необходимо реализовать модуль формы, позволяющий указывать тип измеряемой величины пользователю, а также автоматическую конвертацию, при переключении меры величины с уже введенными значениями.

## Целевая функция

Реализация данной ВКР позволит пользователям платформы быстро перенести информацию в БД, не задумываясь о мерах величин, которые вносятся, а далее уже отображать её в той мере, которая требуется.

## Требования

**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.

## Перечень задач выпускной квалификационной работы

Задачи выпускной квалификационной работы:

1. Анализ предметной области.

2. Изучение принципов работы PostgreSQL, Docker, jQuery, Bootstrap, HTML, CSS, TypeScript.

3. Проектирование системы.

4. Написание формы для взаимодействия с таблицами БД.

5. Реализация модуля управления типом данных, передаваемых в таблицы.

6. Реализация модуля отображения типа данных в таблицах.

7. Реализация удобного изменения (конвертации) данных внутри таблицы и формы.

8. Тестирование работы системы.

# Анализ предметной области

Измеряемые величины. Анализ использования различных величин в жизни

В мире существует огромное количество разных величин. Разные единицы измерения одной и той же величины возникают по разным причинам. Необходимо разобраться, почему так произошло и как реализовать удобное взаимодействие пользователя с бд.

Первая причина – историческая.

Когда-то давно, когда люди только начали придумывать единицы измерения, придумывали их независимо в разных городах и странах. Ничего удивительного, что они получились разными. Более того, одна и та же единица могла со временем менять своё значение.

Часто это случалось, например, с мерами веса, ведь они был связаны с денежными единицами. Древнегреческая драхма была монетой, содержащей одну драхму (по весу) серебра. В разное время в разных полисах чеканились разные драхмы, содержащие от 3.7 до 8.6 граммов серебра. Соответственно, и драхма как мера веса тоже была разной.

Вторая причина - удобство использования.

Единицы измерения должны быть удобными. Для измерения роста человека, например, удобно использовать сантиметры, для расстояний между городами - километры, для расстояний до звёзд - парсеки или световые годы. Можно, конечно, написать, что расстояние от Земли до Солнца равно 149 597 870 700 метров, или что рост человека равен 0.0000000000117 астрономических единиц. Только неудобно с такими числами работать - слишком много лишних знаков. Поэтому в разных профессиональных областях часто используются свои единицы, удобные именно для этих областей.

Примеры самых распространённых величин:

|  |  |
| --- | --- |
| 1 т = 1000 кг  1 т = 10 ц  1 ц = 100 кг  1 кг = 1000 г | 1 век = 100 лет  1 год = 365 / 366 суток  1 сутки = 24 часа  1 час = 60 минут  1 минута = 60 секунд |
| 1 км = 1000 м  1 м = 10 дм = 100 см  1 м = 1000 мм  1 дм = 10 см = 100 мм  1 см = 10 мм | 1 м2 = 100 дм2  1 м2 = 10000 см2  1 дм2 = 100 см2  1 а = 100 м2  1 га = 100 а  1 км2 = 100 га |

Таблица 1. Примеры величин, реализуемых в конвертере

Основной темой ВКР является – реализация удобного ввода разных величин и последующим изменением. Огромное количество величин достаточно сложно запомнить пользователю, так как эту информацию нужно держать в голове, так же никто не отменяет «человеческого фактора» - человек просто может ошибиться в переводе одной величины в другую, поэтому необходимо реализовать возможность автоматического перерасчета значения при изменении величины.

Веб-интерфейс.

Так как одной из задач ВКР является проектировка удобного отображения данных, необходимо разобраться, что такое пользовательский веб-интерфейс, чем он отличается от интерфейса и почему в нашей работе необходимо вообще его использовать.

Компьютер - сложное техническое устройство, которое управляется при помощи сложных команд, поэтому доступ к большим вычислительным мощностям был у тех людей, которые обладали техническими знаниями. Так как главной задачей компьютера являлась возможность облегчить работу с большим объемом данных, необходимо было изобрести интуитивно-понятного посредника между машиной и человеком, не обладающим техническими знаниями в области информационной технологии. В 80-х годах 3 компании поставили перед собой задачу, придумать способы взаимодействия пользователя с компьютером, без сложных и специфических знаний. Xerox, Apple и Microsoft являются компаниями, которые придумали то, что в современном мире зовется интерфейсом.

Последний вариант интерфейса часто называют пользовательским. По виду взаимодействия с вычислительной техникой его разделяют на такие виды.

* Графический пользовательский интерфейс – самый распространенный на сегодняшний день способ взаимодействия пользователя с операционными системами и прикладным программным обеспечением. Отличается интуитивностью, простотой восприятия и ввода информации. Для взаимодействия с таким интерфейсом часто достаточно обычной компьютерной мыши.
* Жестовый интерфейс можно назвать продолжением развития графического интерфейса. В нем управление элементами взаимодействия с операционной системой и программами используются сенсорные экраны, стилусы, графические планшеты и другие передовые технологии.

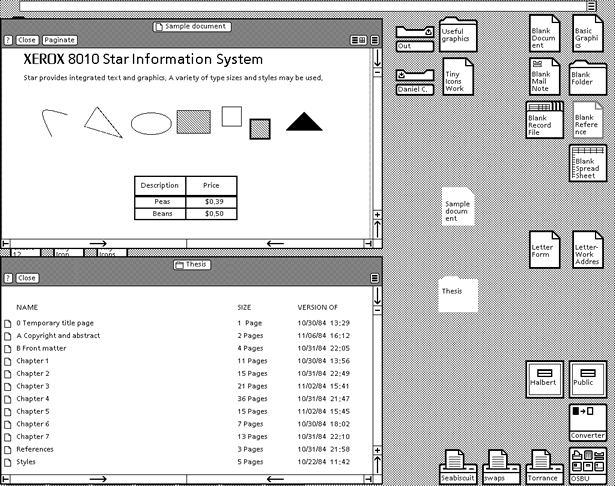


Рисунок 1. Xerox 8010 Star (1981)

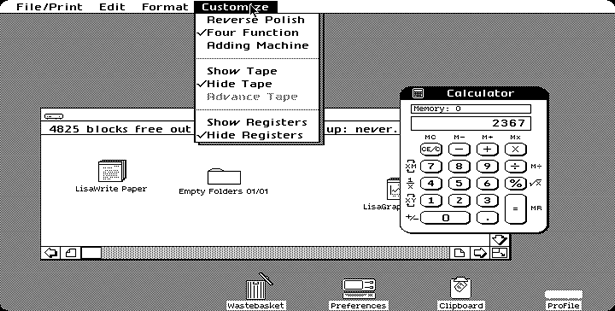


Рисунок 2. Apple Lisa Office System 1 (1983)

Все разработки и решения, придуманные в 80х годах были направлены на то, чтобы упростить опыт использования машин для простых людей.

Ключевая особенность Abris это следование концепции - чем проще, тем лучше для простого пользователя, и именно поэтому в ВКР будет реализован веб-интерфейс, который так же будет позволять людям без технических знаний удобно взаимодействовать с БД.

С развитием интернета стало популярно реализовывать, так называемые веб-приложения, чем и является платформа Abris, необходимо разобраться, что такое веб интерфейс.

Веб интерфейс – это среда взаимодействия пользователя и программы или приложения, запущенной на удаленном сервере.

Чаще всего web-interface применяется для работы с различными онлайн сервисами: начиная с электронной почти и заканчивая системами веб-аналитики. В некоторых случаях веб-интерфейс называется «Личным кабинетом» на каком-либо сайте, но не во все личные кабинеты – интерфейсы. Разберем понятие по частям. Приставка «веб» означает, что элемент работает удаленно от компьютера пользователя, на локальном или интернет-сервере. Взаимодействие с сервисом при этом происходит через «интерфейс» специальную графическую оболочку, состоящую из кнопок, окон, полей заполнения или любых других элементы. Разберем на примерах:

Веб-интерфейс электронной почты

Это самый распространенный случай применения web interface. Примерно 15-20 лет назад, электронная почта работала только через программу-клиент, которую пользователь устанавливал на свой компьютер. Со временем функции этих приложений перенесли в веб – теперь достаточно зайти на страницу сервиса электронной почти и зайти в свой кабинет. Веб-интерфейс электронной почты умеет все то же самое, что и программа-клиент: принимать, отправлять, обрабатывать, перенаправлять и сортировать письма. Сначала их стали использовать такие сервисы как Яндекс, Майл.ру и GMail, а затем их стали подключились и небольшие корпоративные серверы.

В веб-интерфейсе почты есть папки для разной почты: исходящей, входящей, удаленной и нежелательной. Для получения доступа к почтовому интерфейсу потребуется введение пароля, который был указан пользователем в момент регистрации почтового ящика.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 3. Веб-интерфейс gmail

WEB-интерфейс модема или роутера

Любое управляемое сетевое оборудование — это отдельный компьютер и обладает собственной операционной системой. Но она выполняет узко специализированные функции, и позволяет соединиться с сетью провайдера для подключения к интернету.

В самом начале в целях управления ею применялись: протокол и командная строка Telnet. Но в результате развития оборудования, чтобы пользователь мог осуществлять более простой процесс управления, на модеме или роутере был сделан отдельный сервер с полезной для потребителя командной web-оболочкой.

Здесь также присутствует и основное меню, из которого можно настроить устройство. В настоящий момент все современные сетевые устройства с наличием управляющей функции обладают собственным web-интерфейсом управления. В качестве хорошего примера выступают коммутаторы, видео-камеры, модемы, а также маршрутизаторы Zyxel, D-Link, TP-Link, Asus и др.

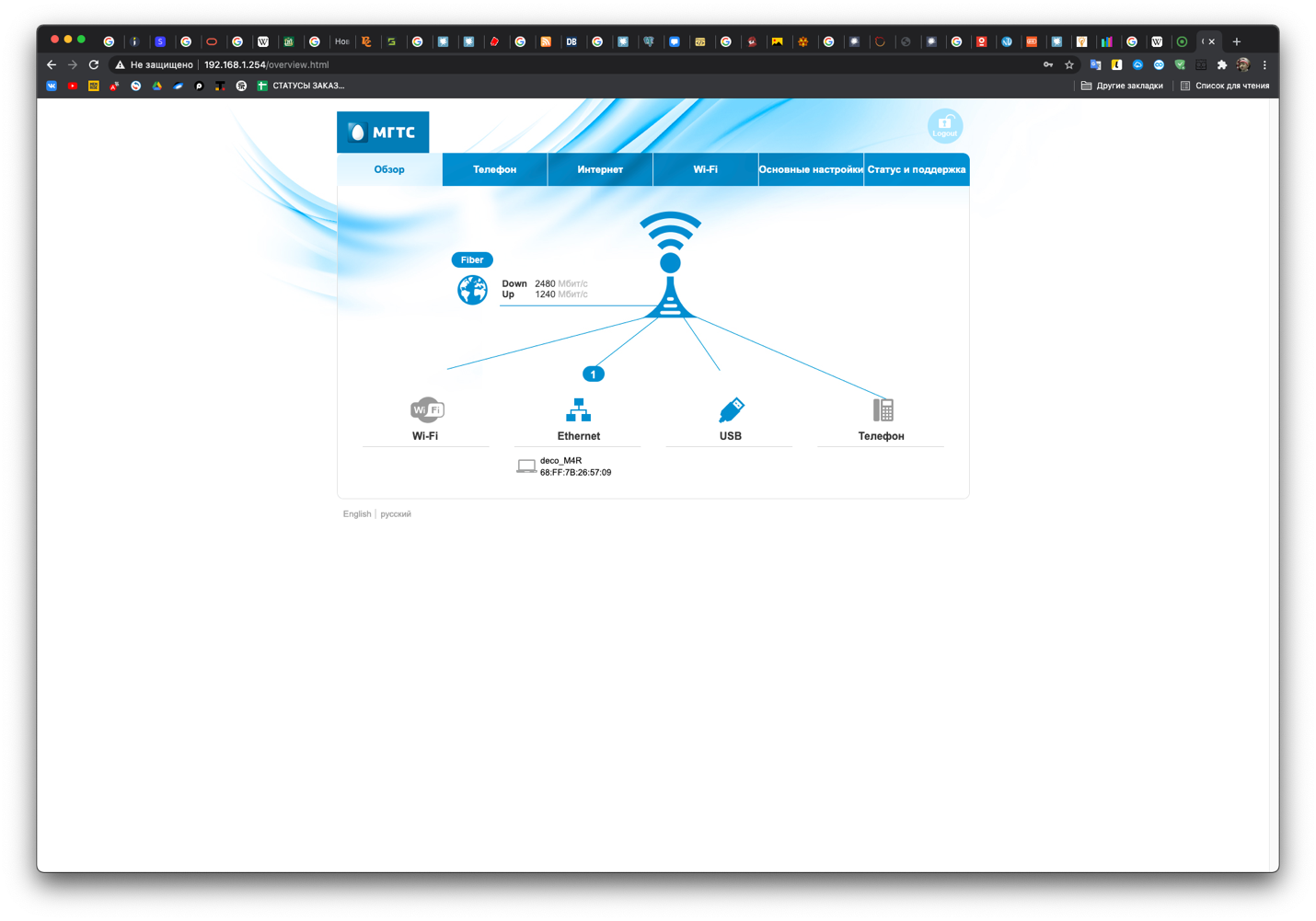


Рисунок 4. Веб-интерфейс модема

Веб-интерфейс хостинга

С ними обычно сталкиваются веб-мастера и пользователи, которые работают с сайтами: занимаются их созданием, продвижением, администрированием и др. Любой сайт в Интернете работает на конкретном www-сервере, который размещен на хостинге – специальной платформе, созданной под размещение сайтов. Эта платформа управляется с помощью веб-интерфейса, который называется “хостинг-панель”

У хостеров востребованы такие оболочки, как Plesk, CPanel, ISPmanager. Отдельные крупные компании для этих целей занимаются разработкой своего определенного набора программ. С помощью веб-интерфейса хостинга открывается доступ для мониторинга состояния сервера и управления его работой, для просмотра логов и сохранения дополнительных копий.

Изображение выглядит как текст, монитор, компьютер, снимок экрана

Автоматически созданное описание

Рисунок 5. Веб-интерфейс хостинга

Облачный веб-интерфейс

В последние 5-6 лет распространились облачные хранилища или просто «облака». После регистрации в сервисе пользователю бесплатно предоставляется место на удаленном жестком диске. Оно используется для хранения самой различной информации: документов, программ, фото, видео или музыкальных файлов. Она доступна из любой точки мира, где есть выход в Интернет. В том случае, когда бесплатно предоставляемого пространства становится недостаточно, то всегда можно докупить его дополнительный объем.

Облака становятся популярнее, потому что с ними удобно работать – нужные файлы доступны везде, где есть Wi-Fi, проводной интернет или покрытие мобильных сетей. В личном кабинете пользователя сервиса можно просматривать, создавать, удалять, копировать и редактировать файлы. Это и есть веб-интерфейс облака. В России чаще используются: Яндекс Диск, Google Drive, DropBox.

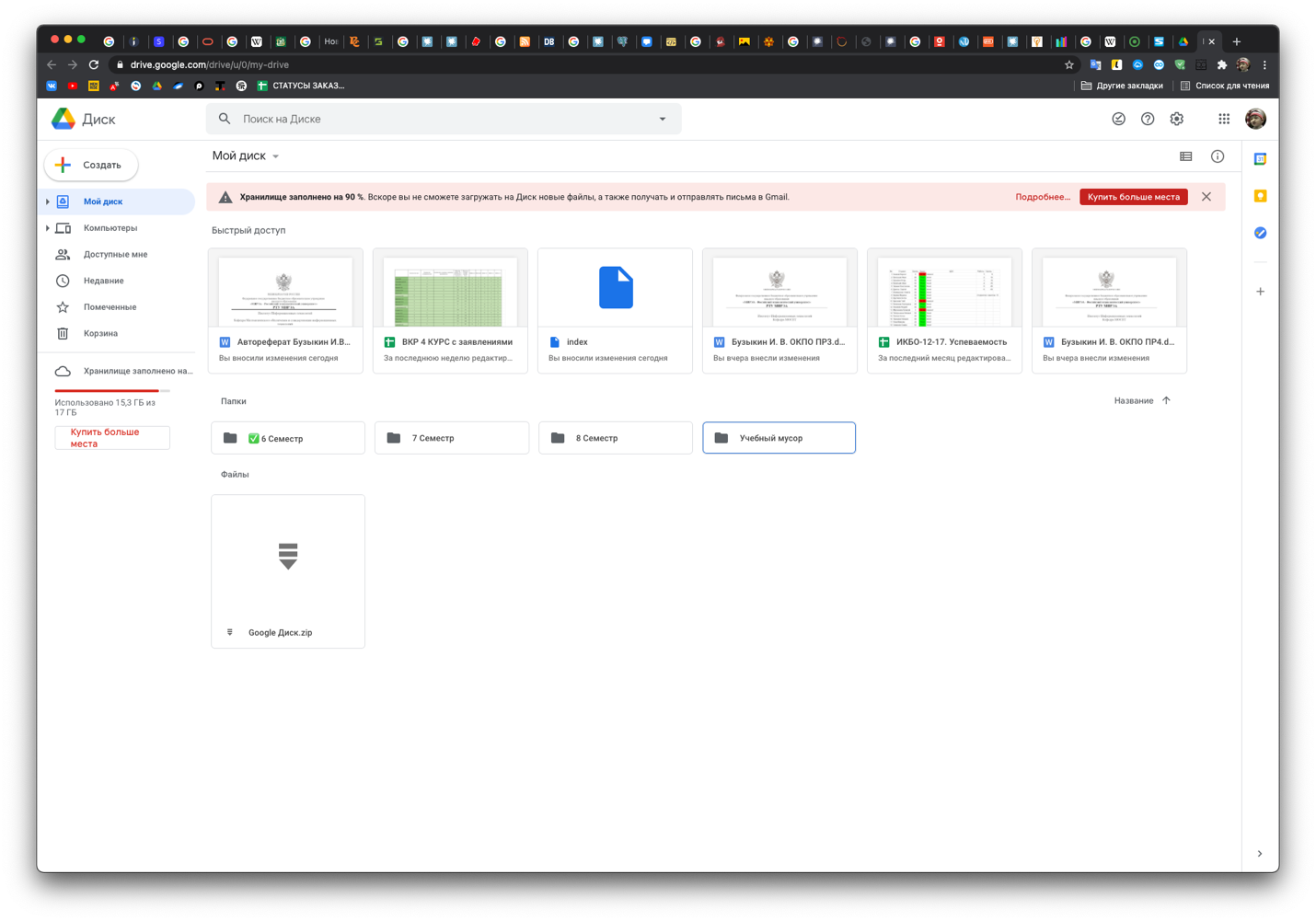


Рисунок 6. Веб-интерфейс Google drive

Таким образом, проведя анализ существующих технологий, можно прийти к выводу, что веб-приложение – это современное решение для удобного взаимодействия пользователя с системой. Нативное приложение в реализации платформы Abris не используется, так как там необходима поддержка большого кол-ва операционных систем, веб-приложение и технологии виртуализации Docker – позволяет единожды спроектированный интерфейс отображать на большом количестве устройств, под управлением разных ОС, что, в свою очередь, решает проблему времени производства продукта, так как писать нативные приложения дольше и финансово дороже.

**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.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

**Принципы построения интерфейсов**

Золотое сечение

Золотое сечение — это самая комфортная для глаза пропорция, форма, в основе построения которой лежит сочетание симметрии и золотого сечения, способствует наилучшему зрительному восприятию и появлению ощущения красоты и гармонии.

В математике пропорцией называют равенство двух отношений: a : b = с: d.

Отрезок прямой АВ можно разделить точкой С на две части следующими способами:

·        на две равные части АВ : АС = АВ : ВС,

·        на две неравные части в любом отношении (такие части пропорции не образуют);

·        таким образом, когда АВ: ВС = ВС: АС.

Последнее и есть золотое деление или деление отрезка в крайнем и среднем отношении.

Золотое сечение — это такое пропорциональное деление отрезка на неравные части, при котором весь отрезок так относится к большей части, как сама большая часть относится к меньшей; или другими словами, меньший отрезок так относится к большему, как больший ко всему.

а : b =  b : с или с : b = b : а.

Отрезки золотой пропорции выражаются бесконечной иррациональной дробью 0,618..., если с принять за единицу, а = 0,382. Отношение же отрезков а и b составляет 1,618.

Прямоугольник с таким отношением сторон стали называть *золотым прямоугольником*. Он также обладает интересными свойствами. Если от него отрезать квадрат, то останется вновь золотой прямоугольник. Этот процесс можно продолжать до бесконечности. А если провести диагональ первого и второго прямоугольника, то точка их пересечения будет принадлежать всем получаемым золотым прямоугольникам.

Есть и золотой треугольник (равнобедренный треугольник, у которого отношение длины боковой стороны к длине основания равняется 1,618), и золотой кубоид (прямоугольный параллелепипед с ребрами, имеющими длины 1,618, 1 и 0,618).

Золотое сечение не является искусственным явлением. Оно очень широко распространено в природе: золотое сечение можно найти в пропорциях тел многих растений и животных, а также морских раковин и птичьих яиц. Но наиболее впечатляющий пример "применения" природой принципа золотого сечения — человеческое тело. Оно целиком и его части (лицо, руки, кисти рук и т. п.) насквозь пронизаны пропорцией 1,618.

Принцип золотого сечения был открыт людьми еще в глубокой древности. Знаменитые египетские пирамиды в Гизе, например, основаны на пропорциях золотого сечения. Более молодые мексиканские пирамиды и античный храм Парфенон также содержат в себе пропорцию 1,618.

С развитием дизайна и технической эстетики действие закона золотою сечения распространилось на конструирование машин, мебели и т. д. Проектирование компьютерных интерфейсов — не исключение. Формы диалоговых окон и элементов управления, стороны которых относятся как 1,618, очень привлекательны для пользователей. Например, очень много восторгов у пользователей программы Chameleon Clock (http://www.softshape.com) вызывает такая, казалось бы, обыденная вещь, как вид диалогового окна Свойства. А все потому, что при его проектировании использовался именно принцип золотого сечения.

Кошелек Миллера

Этот принцип назван так в честь ученого-психолога Г. А. Миллера, который исследовал кратковременную память, проверяя выводы, сделанные ранее его коллегой, Г. Эббингаузом. Эббингауз пытался выяснить, сколько информации может запомнить человек без каких-либо специальных мнемонических приемов. Оказалось, что емкость памяти ограничена семью цифрами, семью буквами или названиями семи предметов. Это "магическое число" семь, служащее своего рода меркой памяти, и было проверено Миллером, который показал, что память действительно в среднем не может хранить более семи элементов; в зависимости от сложности элементов это число может колебаться в пределах от пяти до девяти.

Если необходимо в течение короткого времени сохранить информацию, включающую больше семи элементов, мозг почти бессознательно группирует эту информацию таким образом, чтобы число запоминаемых элементов не превышало предельно допустимого. Например, номер банковского счета 30 637 402 710, состоящий из одиннадцати элементов, будет, скорее всего, запоминаться как 30 63 740 27 10, т. е. как пять числовых элементов, или восемь слов (тридцать, шестьдесят, три, семьсот, сорок, двадцать, семь, десять).

Применяя принцип кошелька Миллера в дизайне интерфейсов, следует группировать элементы в программе (кнопки на панелях инструментов, пункты меню, закладки, опции на этих закладках и т. п.) с учетом этого правила— т. е. не более семи в группе, в крайнем случае — девяти. Взгляните, например, на главное окно программы-словаря ABBYY Lingvo 6.0: четырнадцать кнопок на верхней панели, между которыми нет ни одного разделителя, воспринимаются гораздо хуже, чем кнопки на панели внизу, которые разделены на группы.

Итак, принцип кошелька Миллера говорит о семи плюс-минус двух элементах. Но если взглянуть на программы, интерфейс которых совершенствовался годами (тот же Microsoft Word), то можно заметить, что число объектов (пунктов меню, кнопок на панелях инструментов) в группах доходит до шести-семи довольно редко, а в основном элементы сгруппированы по три-четыре объекта. Такие небольшие группы объектов наиболее хорошо воспринимаются взглядом пользователя, уже слегка утомленного сложными интерфейсами современных программ. Я думаю, при проектировании интерфейсов программ верхнюю границу кошелька Миллера — семь-девять элементов — нужно применять очень осторожно, стараясь обходиться группами, содержащими максимум пять объектов.

Принцип группировки

Согласно этому правилу, экран программы должен быть разбит на ясно очерченные блоки элементов, может быть, даже с заголовком для каждого блока. При этом группировка, естественно, должна быть осмысленной: как расположение элементов в группах, так и расположение самих групп друг от друга должны быть продуманы.

Примеров реализации этого принципа очень много: это уже упоминавшиеся при разговоре о кошельке Миллера пункты меню, кнопочные панели инструментов, а также сгруппированные по назначению флажки и переключатели, с помощью которых настраиваются параметры работы программы в диалоговых окнах Свойства, Настройка и т. п..

Бритва Оккама или KISS

Философский принцип, носящий название "Бритва Оккама", гласит: "Не множить сущности без надобности". Или, как говорят американцы, KISS ("Keep It Simple, Stupid" — "He усложняй, болван").

На языке интерфейсов это означает, что:

·        любая задача должна решаться минимальным числом действий;

·        логика этих действий должна быть очевидной для пользователя;

·        движения курсора и даже глаз пользователя должны быть оптимизированы.

Простым на первый взгляд требованиям из этого списка на самом деле не так уж легко следовать. Для проектирования сложного по своим функциям и простого для понимания интерфейса требуется немалые опыт, знания и особое чутье. Как пишет Лу Гринзоу: "Если и есть в мире что-то такое, что почти все программисты постоянно повторяют наизусть, как мантры, но при этом откровенно игнорируют, так это — принцип KISS'".

Принцип KISS перекликается с несколькими из эвристических правил Якоба Нильсена — "Эстетичный и минималистический дизайн", "Равенство между системой и реальным миром", "Понимание лучше, чем запоминание". KISS более универсален и применяется практически во всех сферах человеческой деятельности, в том числе и в области, очень близкой к теме книги—в программировании.

Видимость отражает полезность

Смысл этого принципа состоит в том, чтобы вынести самую важную информацию и элементы управления на первый план и сделать их легкодоступными пользователю, а менее важную — переместить, например, в меню

Этот вопрос уже немного затрагивался при разговоре о принципе Якоба Нильсена "Эстетичный и минималистический дизайн", правда, в привязке к отражению полезности.

Отличие принципа "Видимость отражает полезность" как раз и состоит в том, что интерфейс программы должен быть построен вокруг объектов, с которыми манипулирует пользователь, и отражать состояние текущего объекта. Реализацию этого принципа вы видите каждый раз, когда пользуетесь компьютером: контекстные панели инструментов в программах пакета Microsoft Office, которые меняются в зависимости от того, с какой частью программы (редактором, предварительным просмотром, рисованием и т. п.) и данный момент работает пользователь. Еще один пример — уже упоминавшееся меню Пуск в Windows ME и Windows 2000: по умолчанию в них видимы наиболее часто используемые, т. е. полезные для пользователя, пункты.

Умное заимствование

Заимствование широко распространенных приемов дизайна интерфейсов и удачных находок авторов конкурирующих программ позволяет резко сократить время обучения и повысить комфорт пользователя. При работе он будет использовать уже приобретенные навыки — этот вопрос затрагивает и принцип равенства между системой и реальным миром.

Заимствование чужих интерфейсных находок не является чем-то зазорным. Программы, лидирующие па рынке, являются неистощимым источником вдохновения для разработчиков более мелких программ, поразительно напоминающих легендарный Norton Commander: FAR, Volcov Commander, DOS Navigator, DISCo Commander

Базы данных. Популярные системы управления базами данных.

База данных — это упорядоченный набор структурированной информации или данных, которые обычно хранятся в электронном виде в компьютерной системе. База данных обычно управляется системой управления базами данных (СУБД). Данные вместе с СУБД, а также приложения, которые с ними связаны, называются системой баз данных, или, для краткости, просто базой данных.

Данные в наиболее распространенных типах современных баз данных обычно хранятся в виде строк и столбцов, формирующих таблицу. Этими данными можно легко управлять, изменять, обновлять, контролировать и упорядочивать. В большинстве баз данных для записи и запросов данных используется язык структурированных запросов (SQL).

SQL — это язык программирования, используемый в большинстве реляционных баз данных для запросов, обработки и определения данных, а также контроля доступа. SQL был разработан в IBM в 1970-х годах. Со временем у стандарта SQL ANSI появились многочисленные расширения, разработанные такими компаниями как IBM, Oracle и Microsoft. Хотя в настоящее время SQL все еще широко используется, начали появляться новые языки программирования запросов.

Базы данных значительно изменились с момента их появления в начале 1960-х годов. Исходными системами, которые использовались для хранения и обработки данных, были навигационные базы данных – например, иерархические базы данных (которые опирались на древовидную модель и допускали только отношение «один-ко-многим») и базы данных с сетевой структурой (более гибкая модель, допускающая множественные отношения). Несмотря на простоту, эти ранние системы были негибкими. В 1980-х годах стали популярными реляционные базы данных, в 1990-х годах за ними последовали объектно-ориентированные базы данных. Совсем недавно вследствие роста Интернета и возникновения необходимости анализа неструктурированных данных появились базы данных NoSQL. В настоящее время облачные базы данных и автономные базы данных открывают новые возможности в отношении способов сбора, хранения, использования данных и управления ими.

В истории вычислительной техники можно проследить развитие двух основных областей ее использования.

Первая область — применение вычислительной техники для выполнения численных расчетов, которые слишком долго или вообще невозможно производить вручную. Развитие этой области способствовало интенсификации методов численного решения сложных математических задач, появлению языков программирования, ориентированных на удобную запись численных алгоритмов, становлению обратной связи с разработчиками новых архитектур ЭВМ. Характерной особенностью данной области применения вычислительной техники является наличие сложных алгоритмов обработки, которые применяются к простым по структуре данным, объем которых сравнительно невелик.

Вторая область, которая непосредственно относится к нашей теме, — это использование средств вычислительной техники в автоматических или автоматизированных информационных системах.

Существует множество различных типов баз данных. Выбор наилучшей базы данных для конкретного проекта зависит от того, как разработчики намереваются использовать данные.

* **Реляционные базы данных.** Реляционные базы данных стали преобладать в 1980-х годах. Данные в реляционной базе организованы в виде таблиц, состоящих из столбцов и строк. Реляционная СУБД обеспечивает быстрый и эффективный доступ к структурированной информации.
* **Объектно-ориентированные базы данных.** Информация в объектно-ориентированной базе данных представлена в форме объекта, как в объектно-ориентированном программировании.
* **Распределенные базы данных.** Распределенная база данных состоит из двух или более частей, расположенных на разных серверах. Такая база данных может храниться на нескольких компьютерах.
* **Хранилища данных.** Будучи централизованным репозиторием для данных, хранилище данных представляет собой тип базы данных, специально предназначенной для быстрого выполнения запросов и анализа.
* **Базы данных NoSQL.** База данных NoSQL, или нереляционная база данных, дает возможность хранить и обрабатывать неструктурированные или слабоструктурированные данные (в отличие от реляционной базы данных, задающей структуру содержащихся в ней данных). Популярность баз данных NoSQL растет по мере распространения и усложнения веб-приложений.
* **Графовые базы данных.** Графовая база данных хранит данные в контексте сущностей и связей между сущностями.
* **Базы данных OLTP.** База данных OLTP — это база данных, предназначенная для выполнения бизнес-транзакций, выполняемых множеством пользователей.

Это лишь некоторые из десятков типов баз данных, используемых в настоящее время.

В реляционных базах данных информация представляется в виде двумерных таблиц. Каждая строка содержит запись с уникальным идентификатором, столбцы имеют атрибуты. Примерами реляционных систем управления баз данных являются MySQL, PostgreSQL, Oracle Database, Microsoft SQL Server.

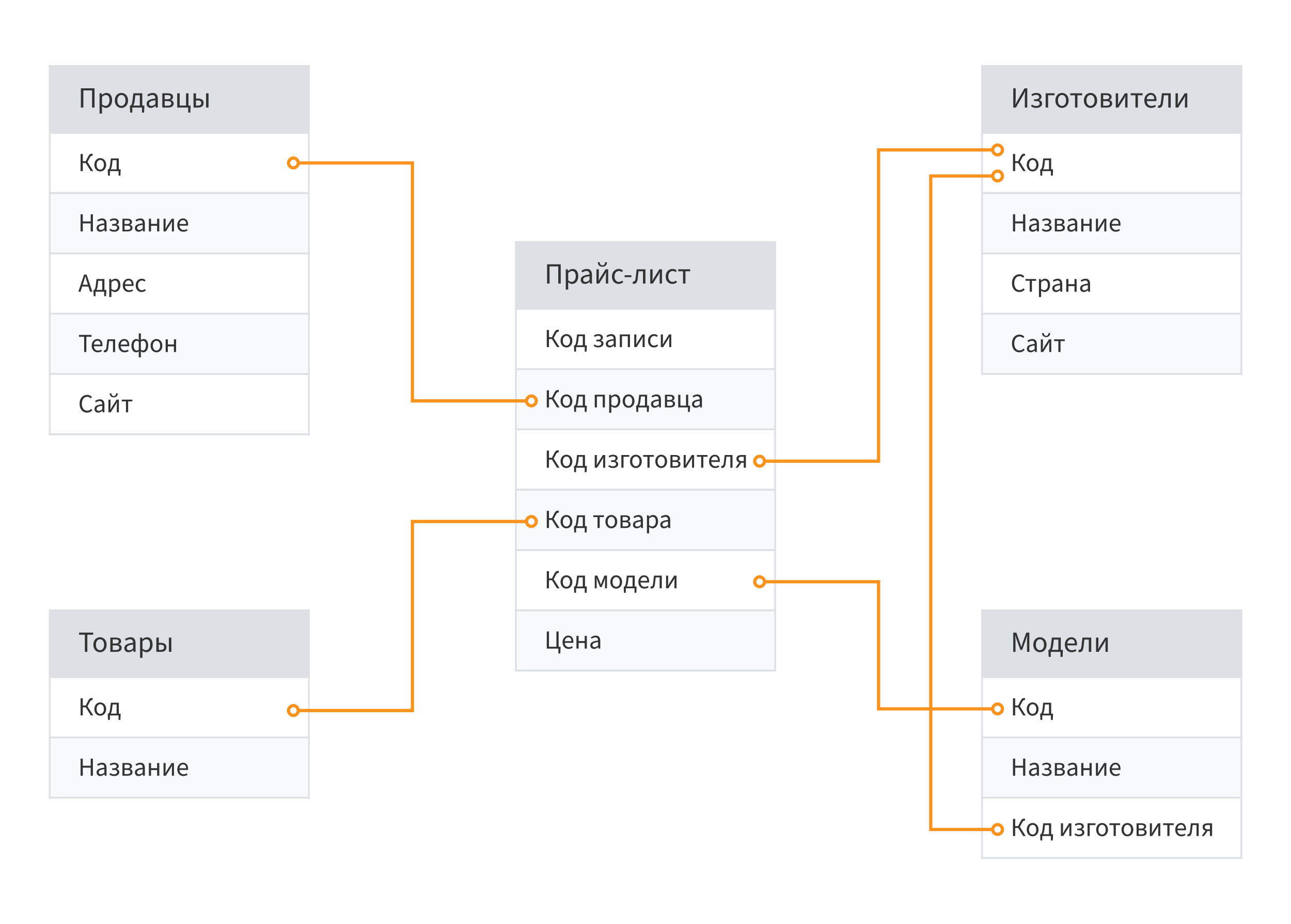


Рисунок 7. Реляционная БД

В отличие от реляционных баз данных в нереляционных структура данных форматируется под конкретные требования типа хранимых данных. К примеру, данные могут быть представлены как ключ-значение, семейство столбцов или являться графом. Примерами нереляционных систем управления баз данных являются Redis, MongoDB, Apache Cassandra, Neo4j.

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 2. 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.

Подведя итог, можно понять, что современный мир информационных технологий трудно представить себе без использования баз данных. Практически все системы в той или иной степени связаны с функциями долговременного хранения и обработки информации. Фактически информация становится фактором, определяющим эффективность любой сферы деятельности. Увеличились информационные потоки и повысились требования к скорости обработки данных, и теперь уже большинство операций не может быть выполнено вручную, они требуют применения наиболее перспективных компьютерных технологий. Любые административные решения требуют четкой и точной оценки текущей ситуации и возможных перспектив ее изменения. И если раньше в оценке ситуации участвовало несколько десятков факторов, которые могли быть вычислены вручную, то теперь таких факторов сотни и сотни тысяч и ситуация меняется не в течение года, а через несколько минут, и обоснованность принимаемых решений требуется большая, потому что и реакция на неправильные решения более серьезная, более быстрая и более мощная, чем раньше.

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. Docker контейнеры и виртуализация.

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 8. 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.

Как работает Docker?

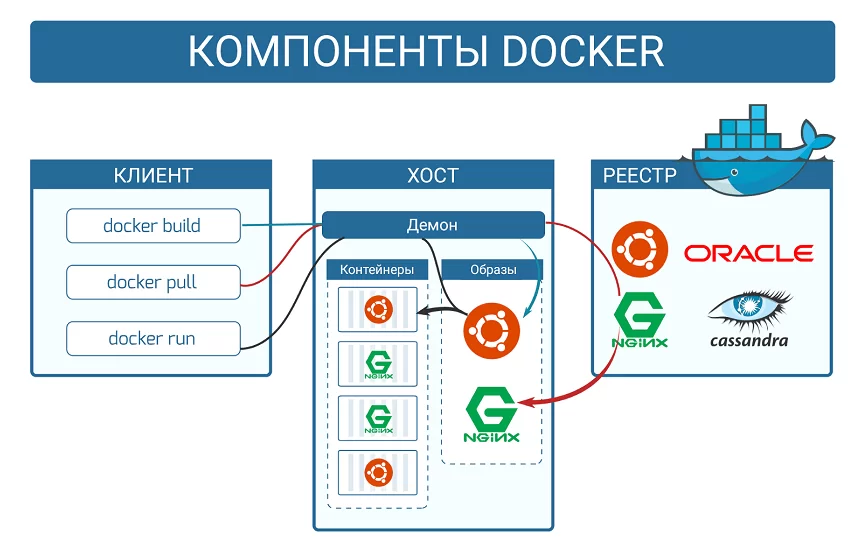


Рисунок 9. Компоненты Docker

Пока мы знаем, что:

* можем создавать образы, в которых находятся наши приложения;
* можем создавать контейнеры из образов, для запуска приложений;
* можем распространять образы через Docker Hub или другой реестр образов.

Как работает образ?

Мы уже знаем, что образ — это read-only шаблон, из которого создается контейнер. Каждый образ состоит из набора уровней. Docker использует union file system для сочетания этих уровней в один образ. Union file system позволяет файлам и директориями из разных файловых систем (разным ветвям) прозрачно накладываться, создавая когерентную файловую систему.  
  
Одна из причин, по которой docker легковесен — это использование таких уровней. Когда вы изменяете образ, например, обновляете приложение, создается новый уровень. Так, без замены всего образа или его пересборки, как вам возможно придётся сделать с виртуальной машиной, только уровень добавляется или обновляется. И вам не нужно раздавать весь новый образ, раздается только обновление, что позволяет распространять образы проще и быстрее.  
  
В основе каждого образа находится базовый образ. Например, ubuntu, базовый образ Ubuntu, или fedora, базовый образ дистрибутива Fedora. Так же вы можете использовать образы как базу для создания новых образов. Например, если у вас есть образ apache, вы можете использовать его как базовый образ для ваших веб-приложений.

Docker образы могут создаться из этих базовых образов, шаги описания для создания этих образов мы называем инструкциями. Каждая инструкция создает новый образ или уровень. Инструкциями будут следующие действия:

* запуск команды
* добавление файла или директории
* создание переменной окружения
* указания что запускать, когда запускается контейнер этого образа

Эти инструкции хранятся в файле Dockerfile. Docker считывает это Dockerfile, когда вы собираете образ, выполняет эти инструкции, и возвращает конечный образ.

Как работает docker реестр?

Реестр — это хранилище docker образов. После создания образа вы можете опубликовать его на публичном реестре Docker Hub или на вашем личном реестре.  
  
С помощью docker клиента вы можете искать уже опубликованные образы и скачивать их на вашу машину с docker для создания контейнеров.  
  
Docker Hub предоставляет публичные и приватные хранилища образов. Поиск и скачивание образов из публичных хранилищ доступно для всех. Содержимое приватных хранилищ не попадает в результат поиска. И только вы и ваши пользователи могут получать эти образы и создавать из них контейнеры.

Как работает контейнер?

Контейнер состоит из операционной системы, пользовательских файлов и метаданных. Как мы знаем, каждый контейнер создается из образа. Этот образ говорит docker-у, что находится в контейнере, какой процесс запустить, когда запускается контейнер и другие конфигурационные данные. Docker образ доступен только для чтения. Когда docker запускает контейнер, он создает уровень для чтения/записи сверху образа (используя union file system, как было указано раньше), в котором может быть запущено приложение.

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).

## Сравнительная оценка своей разработки

Проведя анализ предметной области, было обнаружено, что существующие конвертеры сложный величин не являются динамическими, что в разы затрудняет работу с ними.

Мой же проект реализован на принципах динамического обновления данных, поэтому нет необходимости каждый раз при ошибочной вводе обновлять страницу и вводить все данные заново.

## Перечень основных применяемых стандартов касающейся темы ВКР.

Основные применяемые стандарты:

* ISO/IEC/IEEE 12207 – регламентирует жизненный цикл ПО, базируется на таких процессах: организационные процессы, основные процессы ЖЦ ПО, вспомогательные процессы.
* Стандарт распространяется на автоматизированные системы (АС), используемые в различных видах деятельности (исследование, проектирование, управление и т. п.), включая их сочетания, создаваемые в организациях, объединениях и на предприятиях. ГОСТ 34.603-92 – Стандарт устанавливает виды испытаний АС и общие требования к их проведению.
* ГОСТ 34.601-90 – стандарт распространяется на автоматизированные системы (АС), используемые в различных видах деятельности (исследование, проектирование, управление и т. п.), включая их сочетания, создаваемые в организациях, объединениях и на предприятиях. Стандарт устанавливает стадии и этапы создания АС.

# Глава 2

## Проектирование системы

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

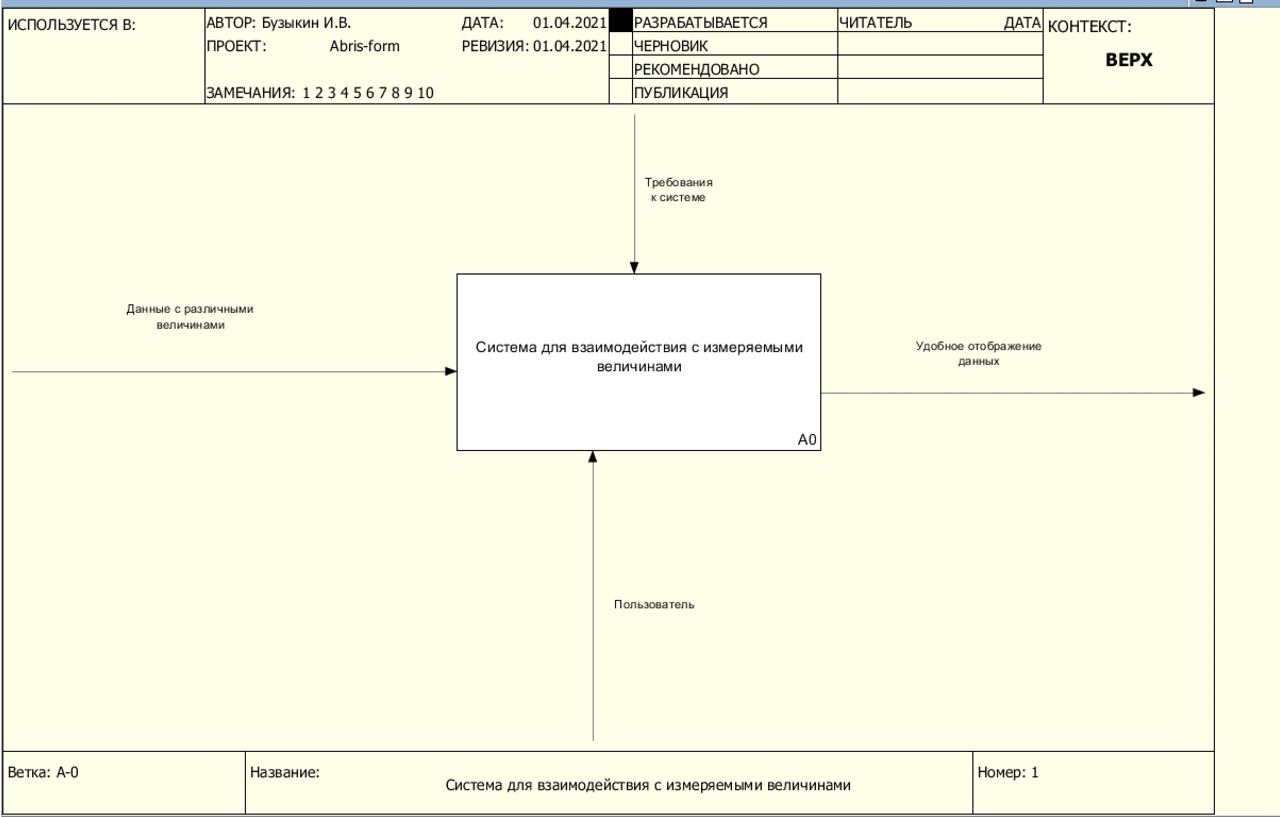


Рисунок 10. Модель бизнес-процесса проекта

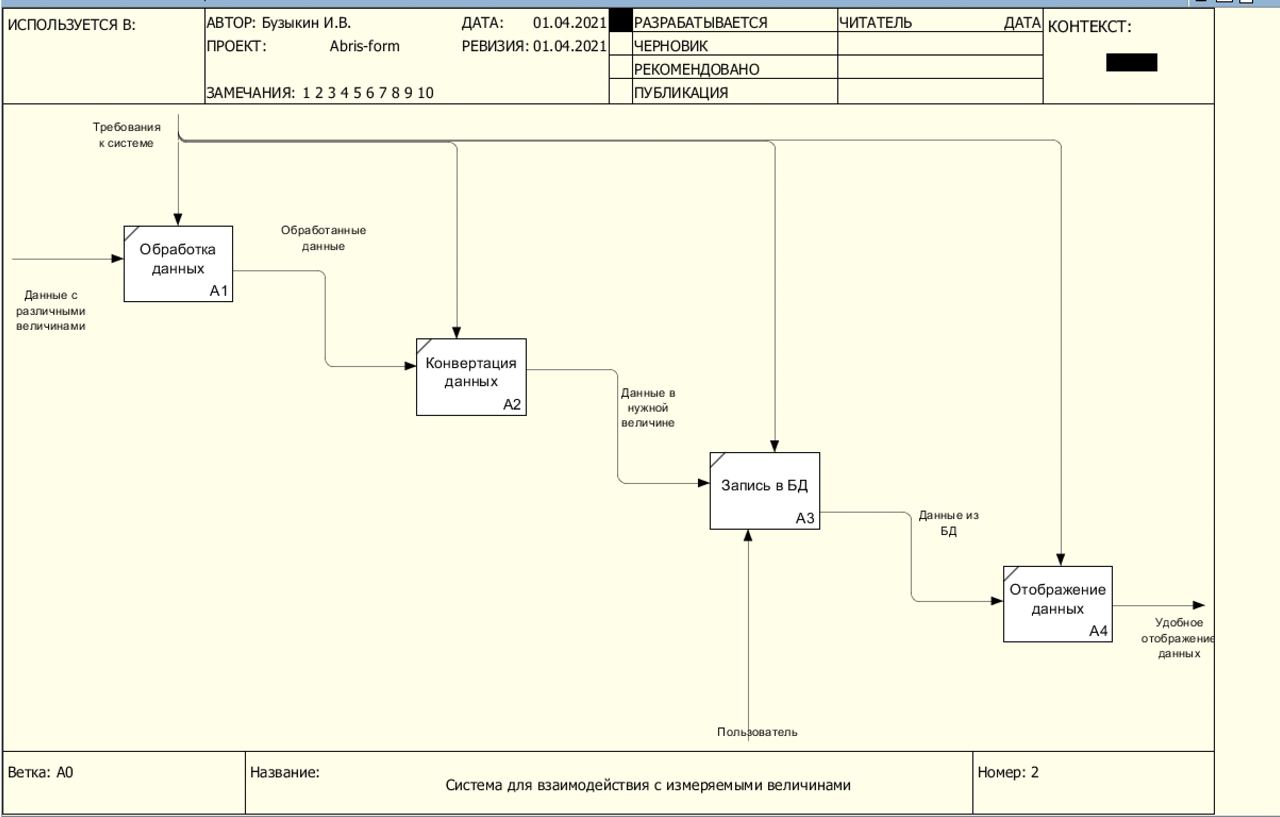


Рисунок 11. Декомпозиция модели бизнес-процесса проекта

## 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 выбраная платформой - реляционная, СУБД – PostgreSQL
* Обязательное использование GIT
* Для создания автоконвертера использовать JS и TS

# Глава 3

## Экспериментальные результаты

На текущий момент, разработана форма заполнения и настроена платформа на локальном компьютере

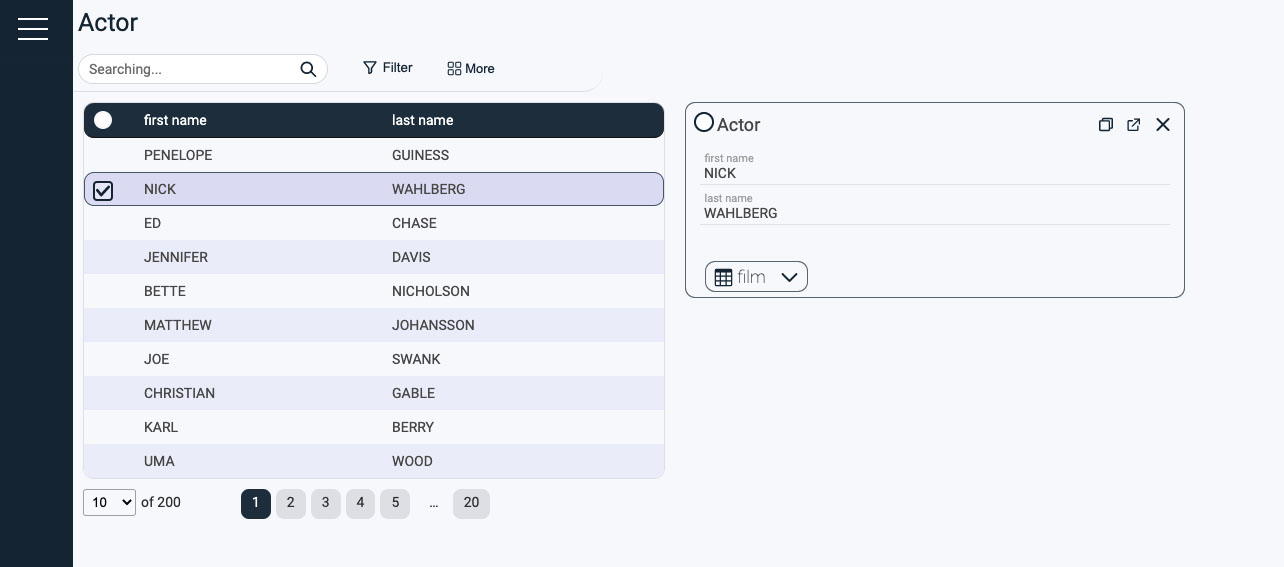


Рисунок 12. Интерфейс проекта

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 13. Дерево проекта

# Выводы

В результате выполнения выпускной квалификационной работы был создан модуль для взаимодействия с данными, у которых есть величины и интегрировал это в виде web-приложение Abris, которая предоставляет пользователям удобный интерфейс для взаимодействия с БД.

Были определены актуальность, цель, новизна и задачи выпускной квалификационной работы. Были проанализированы:

* Использование величин в повседневной жизни
* Базы данных. Популярные системы управления базами данных;
* Принципы построения интерфейсов;
* Необходимость использования Docker

# Информационные источники

1. Стандарт ISO/IEC/IEEE 12207 - [Электронный источник] <https://www.iso.org/standard/73446.html> (дата обращения: 10.03.2021).
2. ГОСТ 34.601-90 - [Электронный источник] <https://docs.cntd.ru/document/1200006921> (дата обращения: 10.03.2021).
3. Официальный сайт Docker – [Электронный источник] <https://www.docker.com/> (дата обращения: 10.03.2021)
4. Базы данных – [Электронный источник] https://www.oracle.com/ru/database/what-is-database/ (дата обращения: 10.03.2021)
5. Abris platform – [Электронный источник] <https://abrisplatform.com/> (дата обращения: 10.03.2021)
6. Docker – [Электронный источник] <https://habr.com/ru/post/253877/> (дата обращения: 10.03.2021)
7. Что такое веб-интерфейс – [Электронный источник] <https://semantica.in/blog/veb-interfejs.html> (дата обращения: 10.03.2021)
8. Создание Web-приложений – [Электронный источник] <https://www.frolov-lib.ru/books/rusedit/web_development/ch01.html> (дата обращения: 10.03.2021)
9. Чем PostgreSQL лучше других SQL баз данных – [Электронный источник] <https://habr.com/ru/post/282764/> (дата обращения: 10.03.2021)
10. Эволюция веб-приложений – [] <https://habr.com/ru/post/218215/> (дата обращения: 10.03.2021)