Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

**ОТЧЕТ**

по производственной практике

Студент группы 450531: Городковец Ф.Н.

Руководитель практики

от университета: Кобяк И.П.

Руководитель практики

от предприятия: Никитин Е.Б.

МИНСК 2018

**СОДЕРЖАНИЕ**

[1.ЗАДАНИЕ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ 3](#__RefHeading___Toc2615_2133121204)

[2.СТРУКТУРА ПРЕДПРИЯТИя 3](#__RefHeading___Toc815_307119199)

[3.Обзор стека технологий 5](#__RefHeading___Toc817_307119199)

[3.1Drupal 5](#__RefHeading___Toc2621_2133121204)

[3.1.1PHP 7](#__RefHeading___Toc2623_2133121204)

[3.2JavaScript 8](#__RefHeading___Toc2627_2133121204)

[3.2.1Структура языка 8](#__RefHeading___Toc2629_2133121204)

[3.2.2Область применения 9](#__RefHeading___Toc2631_2133121204)

[3.3SQL 10](#__RefHeading___Toc2633_2133121204)

[3.3.1Описание 12](#__RefHeading___Toc2635_2133121204)

[3.3.2Операторы 12](#__RefHeading___Toc2637_2133121204)

[3.3.3Преимущества и недостатки 12](#__RefHeading___Toc2639_2133121204)

[4.Выполнение технического задания 13](#__RefHeading___Toc2647_2133121204)

[4.1Определение требований 13](#__RefHeading___Toc2649_2133121204)

[4.2Выполнение задания 14](#__RefHeading___Toc2700_1440254095)

[5.Знакомство с системами контроля версий 17](#__RefHeading___Toc2795_2133121204)

[6.Заключение 21](#__RefHeading___Toc2797_2133121204)

# ЗАДАНИЕ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ

Заданием по производственной практике является:

1. Ознакомление со структурой предприятия. Организация управления;
2. Получение задания от руководителя практики от предприятия;
3. Ознакомление со средой разработки;
4. Изучение системы контроля версий;
5. Выполнение задания;
6. Написание тестов по заданию;
7. Получение отзыва о задании от руководителя;
8. Исправление ошибок.

Место прохождения технологической практики — ЗАО «Итранзишэн». Руководитель практики от предприятия — Никитин Е.Б. B соответствии с указаниями руководителя технологической практики от вуза Кобяка И.П. были определены цели и сформулированы основные задачи.

Дата прибытия на практику – 08.06.2018.

Дата окончания практики – 02.07.2018.

Результатом прохождения практики является оценка руководителя практики, характеристика, а также отзыв руководителей практики от вуза.

# СТРУКТУРА ПРЕДПРИЯТИя

Itransition Group (Itransition) — международная ИТ-компания в области разработки программных решений и ИТ-консалтинга. Резидент Белорусского парка высоких технологий (ПВТ) и член научно-технологической ассоциации «Инфопарк». Входит в пятерку крупнейших экспортеров программного обеспечения из числа компаний-резидентов ПВТ.

Представительства Itransition находятся в США, Великобритании, Нидерландах, России, центр разработки — в Белоруссии.

Компания начала свою деятельность в составе многопрофильного холдинга «БелХард». В 1998 году Сергей Гвардейцев, работавший директором по разработке, создал подразделение из 10 Java-специалистов. В 2005 году подразделение выделилось в независимую компанию, став одним из первых резидентов ПВТ. В 2006 году компания была преобразована в холдинг.

* В 2008 году открыто представительство в России и Нидерландах; Itransition становится участником партнерской программы Microsoft. На сегодняшний день компания является золотым партнером с сертифицированными компетенциями «Порталы и совместная работа» (Collaboration and Content) и «Разработка программного обеспечения» (Application Development).
* В 2009 году открыт офис в Лондоне, Великобритания и первый американский офис в США, в Остине.
* В 2013 году департамент тестирования программного обеспечения Itransition выделился в дочернюю компанию A1QA.
* В 2014 году компания расширяет основные виды деятельности, включая в их список услуги по тестированию безопасности и технической защите информации.
* В 2015 году компания вошла в пятерку крупнейших разработчиков ПО по версии рейтингового агентства «Эксперт РА».
* В 2016 году Алексей Демичев становится генеральным директором компании. Открыт офис в Денвере, США.

Деятельность:

* ИТ-консалтинг
* Заказная разработка ПО
* Интеграция корпоративных приложений
* Тестирование и контроль качества
* Сопровождение и поддержка
* Формирование выделенных команд

Рейтинги и награды:

* С 2009 года Itransition ежегодно входит в рейтинг The Global Outsourcing Top 100 по мнению экспертов IAOP (International Association of Outsourcing Professionals). В 2016 году Itransition вошла в топ рейтинга в номинациях «Отзывы клиентов» и «Инновационные программы», а также вошла в ряды «Суперзвезд» за присутствие в списке на протяжении последних восьми лет.
* В 2015 году компания Deloitte включила Itransition в список самых динамично развивающихся компаний Северной Америки — Deloitte Technology Fast 500.
* Рейтинговое агентство «Эксперт РА» в 2013—2015 годах включило Itransition в список крупнейших российских компаний в области информационных и коммуникационных технологий, а также в Топ-30 разработчиков программного обеспечения. В 2015 году Itransition вошла в пятерку крупнейших разработчиков.
* По данным «Коммерсанта», по итогам 2013—2015 годов Itransition входит в список крупнейших ИТ-компаний России, а также в список крупнейших разработчиков программного обеспечения.
* Компания является лауреатом первой премии «HR-бренд Беларусь 2014», двукратным победителем номинации «Социально ответственный бренд» конкурса «Бренд года», организатором самого значимого проекта КСО Беларуси за 2016 год в номинации «Взаимодействие с местными сообществами» по версии фонда «Идея».

Образовательная деятельность

* С 2005 года работает учебный центр Itransition, ежегодно обучая более шестисот студентов технологиям Java, .NET, Ruby, PHP, SAP и тестированию ПО.

Компания активно сотрудничает с ведущими техническими вузами Беларуси. На сегодняшний день Itransition оборудованы 12 учебных лабораторий в Белорусском национальном техническом университете (БНТУ), Белорусском государственном университете (БГУ), Белорусском государственном университете информатики и радиоэлектроники (БГУИР).

# Обзор стека технологий

## Drupal

Drupal (Дру́пал) — система управления содержимым (CMS), используемая также как каркас для веб-приложений (CMF), написанная на языке PHP и использующая в качестве хранилища данных реляционную базу данных (поддерживаются MySQL, PostgreSQL и другие). Drupal является свободным программным обеспечением, защищённым лицензией GPL, и развивается усилиями энтузиастов со всего мира.

После того, как Drupal был создан, очевидным вопросом стал выбор и создание эмблемы проекта. Участники сообщества считали, что она должна быть связана с каплей или водой. Первоначально эмблема Drupal изображалась как капля в круге, со временем пришла идея сделать эмблему в виде карикатурной капли с лицом. Стивен Виттенс (Steven Wittens) создал трёхмерную каплю, но идея не пошла в массы главным образом потому, что трёхмерный объект трудно печатать, редактировать и т. д.

Текущая эмблема системы придумана Кристьяном Янсеном (Kristjan Jansen), она представляет собой каплю, изображающую лицо, глаза показываются как две соединённые капли в виде знака бесконечности и выражают в символической форме бесконечные возможности.

Архитектура Drupal позволяет применять его для построения различных типов сайтов — от блогов и новостных сайтов до информационных архивов или социальных сетей. Имеющуюся по умолчанию функциональность можно увеличивать подключением дополнительных расширений — «модулей» в терминологии Drupal. Наиболее важные функции, предоставляемые Drupal «из коробки»:

* единая категоризация всех видов содержимого (таксономия) — от форумных сообщений до блогов и новостных статей;
* широкий набор свойств при построении рубрикаторов: плоские списки, иерархии, иерархии с общими предками, синонимы, родственные категории;
* вложенность категорий любой глубины;
* поиск по содержимому сайта, в том числе поиск по таксономии и пользователям;
* разграничение доступа пользователей к материалам (ролевая модель);
* динамическое построение меню;
* поддержка XML-форматов;
* авторизация через OpenID;
* символьные осмысленные URL (иначе «человеко-понятные» — ЧПУ);
* переводы интерфейса сайта на разные языки, а также поддержка ведения разноязычного контента;
* возможность создания сайтов с пересекающимся содержимым (например общей базой пользователей или общими настройками);
* раздельные конфигурации сайта для различных виртуальных хостов (мультисайтинг), в том числе собственные наборы модулей и тем оформления для каждого подсайта;
* уведомления о выходящих обновлениях модулей/

Система поддерживает локализацию интерфейса на более чем 100 языкаx (однако не все переводы являются полными). Имеется поддержка русского языка. В Drupal используется собственный механизм для локализации с хранением переводов в базе данных, наравне с остальным содержимым сайта. Импорт и экспорт переводов сайта осуществляется в виде po-файлов (формат используемый библиотекой gettext).

Полная локализация содержимого поддерживается сторонними модулями, которые не входят в поставку Drupal, но могут быть скачаны с сайта разработчиков.

В Drupal предлагается гибкая схема организации структуры сайта на основе таксономии. Таксономия — механизм, позволяющий создавать произвольное количество тематических категорий для содержимого сайта и ассоциировать их с модулями, обеспечивающими ввод и вывод информации. Категории могут представлять плоские или иерархические списки, либо сложные структуры, где элемент может иметь несколько «родителей» и несколько дочерних элементов. С помощью подобной схемы одними и теми же модулями возможна организация различных вариантов структуризации содержимого. Например, легко создаётся сквозной список «ключевых слов» для всех документов сайта и т. п.

Другая парадигма появилась с созданием в Drupal расширения Content Construction Kit (CCK). CCK позволяет дополнять документы новыми полями различных типов — от полей ввода URL и email, до полей хранения и отображения мультимедийных файлов. Также посредством дополнительных модулей к CCK (например Node reference) можно организовать связи между документами, не используя механизм таксономии.

### PHP

PHP (Hypertext Preprocessor — «PHP: препроцессор гипертекста») — скриптовый язык общего назначения, интенсивно применяемый для разработки веб-приложений. В настоящее время поддерживается подавляющим большинством хостинг-провайдеров и является одним из лидеров среди языков, применяющихся для создания динамических веб-сайтов.

Язык и его интерпретатор (Zend Engine) разрабатываются группой энтузиастов в рамках проекта с открытым кодом. Проект распространяется под собственной лицензией, несовместимой с GNU GPL.

В области веб-программирования, в частности серверной части, PHP — один из популярных сценарных языков (наряду с JSP, Perl и языками, используемыми в ASP.NET).

Популярность в области построения веб-сайтов определяется наличием большого набора встроенных средств для разработки веб-приложений[10]. Основные из них:

* автоматическое извлечение POST и GET-параметров, а также переменных окружения веб-сервера в предопределённые массивы;
* взаимодействие с большим количеством различных систем управления базами данных (MySQL, MySQLi, SQLite, PostgreSQL, Oracle (OCI8), Oracle, Microsoft SQL Server, Sybase, ODBC, mSQL, IBM DB2, Cloudscape и Apache Derby, Informix, Ovrimos SQL, Lotus Notes, DB++, DBM, dBase, DBX, FrontBase, FilePro, Ingres II, SESAM, Firebird / InterBase, Paradox File Access, MaxDB, Интерфейс PDO), Redis;
* автоматизированная отправка HTTP-заголовков;
* работа с HTTP-авторизацией;
* работа с cookies и сессиями;
* работа с локальными и удалёнными файлами, сокетами;
* обработка файлов, загружаемых на сервер;
* работа с XForms.

В настоящее время PHP используется сотнями тысяч разработчиков. Согласно рейтингу корпорации TIOBE, базирующемся на данных поисковых систем, в мае 2016 года PHP находился на 6 месте среди языков программирования. К крупнейшим сайтам, использующим PHP, относятся Facebook, Wikipedia и др.

Входит в LAMP — распространённый набор программного обеспечения для создания и хостинга веб-сайтов (Linux, Apache, MySQL, PHP).

В 2014 году было проведено голосование, по результатам которого следующая версия получила название PHP 7. Выход новой версии планировался в середине октября 2015 года. В марте 2015 года Zend представили инфографику в которой описаны основные нововведения PHP 7.

3 декабря 2015 года было объявлено о выходе PHP версии 7.0.0.

Новая версия основывается на экспериментальной ветви PHP, которая изначально называлась phpng (PHP Next Generation — следующее поколение), и разрабатывалась с упором на увеличение производительности и уменьшение потребления памяти. В новой версии добавлена возможность указывать тип возвращаемых из функции данных, добавлен контроль передаваемых типов для скалярных данных, а также новые операторы.

## JavaScript

JavaScript обычно используется как встраиваемый язык для программного доступа к объектам. Наиболее широкое применение находит в [браузерах](http://ru.wikipedia.org/wiki/Браузер) как язык сценариев для придания[интерактивности](http://ru.wikipedia.org/wiki/Интерактивность)[веб-страницам](http://ru.wikipedia.org/wiki/Веб-страница).

[объекты первого класса](http://ru.wikipedia.org/wiki/Объект_первого_класса)

На JavaScript оказали влияние многие языки, при разработке была цель сделать язык похожим на Java, но при этом лёгким для использования. Языком JavaScript не владеет какая-либо компания или организация, что отличает его от ряда языков программирования, используемых в веб-разработке.

Название «JavaScript» является зарегистрированным [товарным знаком](http://ru.wikipedia.org/wiki/Товарный_знак) компании.

### Структура языка

Структурно JavaScript можно представить в виде объединения трёх чётко различимых друг от друга частейю

* ядро (ECMAScript);
* объектная модель браузера ([Browser Object Model или BOM](http://ru.wikipedia.org/w/index.php?title=Browser_Object_Model&action=edit&redlink=1) );
* объектная модель документа ([Document Object Model или DOM](http://ru.wikipedia.org/wiki/Document_Object_Model)).

Если рассматривать JavaScript в отличных от браузера окружениях, то объектная модель браузера и объектная модель документа могут не поддерживаться.

Объектную модель документа иногда рассматривают как отдельную от JavaScript сущность, что согласуется с определением DOM как независимого от языка интерфейса документа. В противоположность этому ряд авторов находят BOM и DOM тесно взаимосвязанными.

ECMAScript не является браузерным языком и в нём не определяются методы ввода и вывода информации. Это, скорее, основа для построения скриптовых языков. Спецификация ECMAScript описывает типы данных, инструкции, ключевые и [зарезервированные](http://ru.wikipedia.org/wiki/Зарезервированное_слово)слова, объекты, не ограничивая авторов производных языков в расширении их новыми составляющими.

Объектная модель браузера — браузер-специфичная часть языка, являющаяся прослойкой между ядром и объектной моделью документа. Основное предназначение объектной модели браузера — управление окнами браузера и обеспечение их взаимодействия. Каждое из окон браузера представляется объектом window, центральным объектом DOM. Объектная модель браузера на данный момент не стандартизирована, однако спецификация находится в разработке.

Помимо управления окнами, в рамках объектной модели браузера, браузерами обычно обеспечивается поддержка следующих сущностей.

* Управление;
* поддержка задержки в исполнении кода и зацикливания с задержкой;
* системные диалоги;
* управление адресом открытой страницы;
* управление информацией о браузере;
* управление информацией о параметрах;
* ограниченное управление историей просмотра страниц;
* Объектная модель документа.

Объектная модель документа — [интерфейс программирования приложений](http://ru.wikipedia.org/wiki/Интерфейс_программирования_приложений) для HTML и [XML](http://ru.wikipedia.org/wiki/XML)-документов. Согласно DOM, документ (например, веб-страница) может быть представлен в виде дерева объектов, обладающих рядом свойств, которые позволяют производить с ним различные манипуляции:

* генерация и добавление узлов;
* получение узлов;
* изменение узлов;
* изменение связей между узлами;
* удаление узлов.

### Область применения

JavaScript используется в клиентской части веб-приложений: клиент-серверных программ, в котором клиентом является браузер, а сервером — веб-сервер, имеющих распределённую между сервером и клиентом логику. Обмен информацией в веб-приложениях происходит по сети. Одним из преимуществ такого подхода является тот факт, что клиенты не зависят от конкретной операционной системы пользователя, поэтому веб-приложения являются кроссплатформенными сервисами.

JavaScript используется в, популярном подходе к построению интерактивных пользовательских интерфейсов веб-приложений, заключающемся в «фоновом» асинхронном обмене данными браузера с веб-сервером. В результате, при обновлении данных веб-страница не перезагружается полностью и интерфейс веб-приложения становится быстрее, чем это происходит при традиционном подходе (без применения AJAX).

JavaScript широко используется в[браузерных операционных системах](http://ru.wikipedia.org/wiki/WebOS). Так, например, исходный код [IndraDesktop WebOS](http://ru.wikipedia.org/w/index.php?title=IndraDesktop_WebOS&action=edit&redlink=1) на 75 % состоит из JavaScript, код браузерной операционной системы[IntOS](http://ru.wikipedia.org/w/index.php?title=IntOS&action=edit&redlink=1) — на 70 %. Доля JavaScript в исходном коде [eyeOS](http://ru.wikipedia.org/wiki/EyeOS)— 5 %, однако и в рамках этой операционной системы JavaScript играет важную роль, участвуя в визуализации на клиенте и являясь необходимым механизмом для коммуницирования клиента и сервера.

Пользовательские скрипты в браузере — это программы, написанные на JavaScript, выполняемые в браузере пользователя при загрузке страницы. Они позволяют автоматически заполнять формы, переформатировать страницы, скрывать нежелательное содержимое и встраивать желательное для отображения содержимое, изменять поведение клиентской части веб-приложений, добавлять элементы управления на страницу и т. д.

Для управления пользовательскими скриптами в [Mozilla Firefox](http://ru.wikipedia.org/wiki/Mozilla_Firefox) используется расширение; и предоставляют средства поддержки пользовательских скриптов и возможности для выполнения ряда скриптов Greasemonkey.

Приложения, написанные на JavaScript, могут исполняться на серверах, использующих [Java](http://ru.wikipedia.org/wiki/Java) 6 и более поздних версий. Это обстоятельство используется для построения серверных приложений, позволяющих обрабатывать JavaScript на стороне сервера.

Помимо Java 6, существует ряд платформ, использующих существующие движки (интерпретаторы) JavaScript для исполнения серверных приложений. (Как правило, речь идёт о повторном использовании движков, ранее созданных для исполнения кода JavaScript в браузерах WWW.)

## SQL

SQL (structured query language — «структурированный язык запросов») — декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных.

SQL является прежде всего информационно-логическим языком, предназначенным для описания, изменения и извлечения данных, хранимых в реляционных базах данных. SQL можно назвать языком программирования, при этом он не является тьюринг-полным, но вместе с тем стандарт языка спецификацией SQL/PSM предусматривает возможность его процедурных расширений.

Изначально SQL был основным способом работы пользователя с базой данных и позволял выполнять следующий набор операций:

* создание в базе данных новой таблицы;
* добавление в таблицу новых записей;
* изменение записей;
* удаление записей;
* выборка записей из одной или нескольких таблиц (в соответствии с заданным условием);
* изменение структур таблиц.

Со временем SQL усложнился — обогатился новыми конструкциями, обеспечил возможность описания и управления новыми хранимыми объектами (например, индексы, представления, триггеры и хранимые процедуры) — и стал приобретать черты, свойственные языкам программирования.

При всех своих изменениях SQL остаётся единственным механизмом связи между прикладным программным обеспечением и базой данных. В то же время современные СУБД, а также информационные системы, использующие СУБД, предоставляют пользователю развитые средства визуального построения запросов.

Каждое предложение SQL — это либо запрос данных из базы, либо обращение к базе данных, которое приводит к изменению данных в базе. В соответствии с тем, какие изменения происходят в базе данных, различают следующие типы запросов:

Запросы на создание или изменение в базе данных новых или существующих объектов (при этом в запросе описывается тип и структура создаваемого или изменяемого объекта):

* запросы на получение данных;
* запросы на добавление новых данных (записей);
* запросы на удаление данных;
* обращения к СУБД.

Основным объектом хранения реляционной базы данных является таблица, поэтому все SQL-запросы — это операции над таблицами. В соответствии с этим, запросы делятся на:

* запросы, оперирующие самими таблицами;
* запросы, оперирующие с отдельными записями или наборами записей.

Каждая таблица описывается в виде перечисления своих полей (столбцов таблицы) с указанием:

* типа хранимых в каждом поле значений;
* связей между таблицами (задание первичных и вторичных ключей);
* информации, необходимой для построения индексов.

Запросы первого типа в свою очередь делятся на запросы, предназначенные для создания в базе данных новых таблиц, и на запросы, предназначенные для изменения уже существующих таблиц. Запросы второго типа оперируют со строками, и их можно разделить на запросы следующего вида:

* вставка новой строки;
* изменение значений полей строки или набора строк;
* удаление строки или набора строк.

Самый главный вид запроса — это запрос, возвращающий (пользователю) некоторый набор строк, с которым можно осуществить одну из трёх операций:

* просмотреть полученный набор;
* изменить все записи набора;
* удалить все записи набора.

Таким образом, использование SQL сводится, по сути, к формированию всевозможных выборок строк и совершению операций над всеми записями, входящими в набор.

### Описание

Язык SQL представляет собой совокупность:

* операторов,
* инструкций,
* вычисляемых функций.

### Операторы

Согласно общепринятому мнению, операторы (и другие зарезервированные слова) в SQL обычно делятся на:

* Операторы определения данных (Data Definition Language, [DDL](http://ru.wikipedia.org/wiki/Data_Definition_Language)):
  1. [CREATE](http://ru.wikipedia.org/wiki/Create_(SQL))создает объект БД (саму базу, таблицу, [представление](http://ru.wikipedia.org/wiki/Представление_(базы_данных)), пользователя и т. д.),
  2. [ALTER](http://ru.wikipedia.org/w/index.php?title=Alter_(SQL)&action=edit&redlink=1)изменяет объект,
  3. [DROP](http://ru.wikipedia.org/w/index.php?title=Drop_(SQL)&action=edit&redlink=1)удаляет объект;
* Операторы манипуляции данными (Data Manipulation Language, [DML](http://ru.wikipedia.org/wiki/Data_Manipulation_Language)):
  1. [SELECT](http://ru.wikipedia.org/wiki/Select_(SQL))считывает данные, удовлетворяющие заданным условиям,
  2. [INSERT](http://ru.wikipedia.org/wiki/Insert_(SQL))добавляет новые данные,
  3. [UPDATE](http://ru.wikipedia.org/wiki/Update_(SQL))изменяет существующие данные,
  4. [DELETE](http://ru.wikipedia.org/wiki/Delete_(SQL))удаляет данные;
* Операторы доступа к данным (Data Control Language):
  1. [GRANT](http://ru.wikipedia.org/w/index.php?title=Grant_(SQL)&action=edit&redlink=1)предоставляет разрешения на операции с объектом;
  2. [REVOKE](http://ru.wikipedia.org/w/index.php?title=Revoke_(SQL)&action=edit&redlink=1)отзывает ранее выданные разрешения;
  3. [DENY](http://ru.wikipedia.org/w/index.php?title=Deny_(SQL)&action=edit&redlink=1)задает запрет, имеющий приоритет над разрешением;
* Операторы управления [транзакциями](http://ru.wikipedia.org/wiki/Транзакция_(информатика)) (Transaction Control Language, [TCL](http://ru.wikipedia.org/wiki/TCL_(SQL))):
  1. [COMMIT](http://ru.wikipedia.org/wiki/Commit_(SQL))применяет транзакцию;
  2. [ROLLBACK](http://ru.wikipedia.org/wiki/Rollback_(SQL))откатывает все изменения, сделанные в контексте транзакции;
  3. [SAVEPOINT](http://ru.wikipedia.org/w/index.php?title=Savepoint_(SQL)&action=edit&redlink=1)делит транзакцию на более мелкие участки.

### Преимущества и недостатки

Несмотря на наличие диалектов и различий в синтаксисе, в большинстве своём тексты SQL-запросов, содержащие DDL и DML, могут быть достаточно легко перенесены из одной СУБД в другую. Существуют системы, разработчики которых изначально ориентировались на применение по меньшей мере нескольких СУБД. Естественно, что при применении некоторых специфичных для реализации возможностей такой переносимости добиться уже очень трудно.

Наличие стандартов и набора тестов для выявления совместимости и соответствия конкретной реализации SQL общепринятому стандарту только способствует «стабилизации» языка. Правда, стоит обратить внимание, что сам по себе стандарт местами чересчур формализован и раздут в размерах (например, базовая часть стандарта SQL: 2003 состоит из более 1300 страниц текста).

С помощью SQL программист описывает только то, какие данные нужно извлечь или модифицировать. То, каким образом это сделать, решает СУБД непосредственно при обработке SQL-запроса. Однако не стоит думать, что это полностью универсальный принцип — программист описывает набор данных для выборки или модификации, однако ему при этом полезно представлять, как СУБД будет разбирать текст его запроса. Чем сложнее сконструирован запрос, тем больше он допускает вариантов написания, различных по скорости выполнения, но одинаковых по итоговому набору данных.

Создатели реляционной модели данных[Эдгар Кодд](http://ru.wikipedia.org/wiki/Кодд,_Эдгар), и их сторонники указывают на то, что SQL не является истинно реляционным языком. В частности, они указывают на следующие проблемы SQL:

* повторяющиеся строки;
* неопределённые значения (nulls);
* явное указание порядка колонок слева направо;
* колонки без имени и дублирующиеся имена колонок;
* отсутствие поддержки свойства «=»;
* использование указателей;
* высокая избыточность.

# Выполнение технического задания

## Определение требований

Разработать модуль для Drupal 7, реализующий следующее.

Реализовать возможность уведомлять администратора/модератора об опечатках в контенте (читатель выделяет фрагмент текста, жмёт Ctrl+Enter, появляется попап как на orphus.ru и т.д.).

Функциональность должна подвязываться к полям типа rich text (включаться/выключаться для поля в настройках поля).

Глобальные настройки модуля:

* email, на который высылаются жалобы (по умолчанию email админа сайта);
* макс. длина выделяемого текста, разрешённая для жалобы (т.е. для бОльшего фрагмента попап не появится);
* текст, выводимый в попапе жалобы;
* суточный лимит суммарного количества жалоб с одного IP (либо неограниченное количество).

Частные настройки (для каждого поля по отдельности, если частная настройка для поля задана, она перекрывает глобальную):

* email, на который высылаются жалобы (по умолчанию email админа сайта);
* макс. длина выделяемого текста, разрешённая для жалобы (т.е. для бОльшего фрагмента попап не появится);
* текст, выводимый в попапе жалобы.

JavaScript должен корректно отрабатывать при условии, что на странице может быть >1 поля, контролируемого модулем, эти поля могут иметь разный текст для popup.

Нужно реализовать страницу в админке, отображающую список жалоб.

В email о жалобе (а так же на соответствующей странице админки) должны присутствовать:

* тип сущности (нода, etc)
* id сущности, ссылка на её редактирование
* время жалобы
* название (человеческое и машинное) поля, на которое жалуются
* фрагмент текста, на который жалуются
* UID/юзернейм (если залогинен) и IP-адрес юзера-жалобщика

Постарайся сделать максимальный упор на красивое решение и соответствие стандартам кодирования под Drupal 7. По мере готовности логически завершённых частей коммить и пуш в свой битбакет, поделишься потом ссылкой на репозиторий.

## Выполнение задания

Что бы начать выполнение задания нужно было выполнить несколько предварительных этапов:

* настроить рабочее окружение;
* уставить «чистый» Drupal на хост машину и настроить его;
* создать репозиторий модуля для разработки;

После этого началась разработка страницы с настройкам в панели администратора, там должны были быть такие поля, как:

* email, на который высылаются жалобы (по умолчанию email админа сайта)
* макс. длина выделяемого текста, разрешённая для жалобы (т.е. для бОльшего фрагмента попап не появится)
* текст, выводимый в попапе жалобы
* суточный лимит суммарного количества жалоб с одного IP (либо неограниченное количество)

Эти же настройки будут глобальными для всех полей, в которых будут настройки данного модуля, но в каждом поле можно будет их переопределить.

В связи с особенностями системы Drupal большая часть функциональности выполнялась в файле с расширением «.module» в так называемых хуках(hook). Хук это функция которую можно описать в своём модуле и после она будет выполнена в процессе подготовки и загрузки страницы.

Пример хука — hook\_form\_FORM\_ID\_alter() - позволяет получить доступ к уже существующей форме на странице и переопределить её свойства. Таких хуков множество и всё выполняются в определённой последовательности.

Но кроме того можно определять и свои функции, но их имена должны следовать правилам drupal для модулей.

После глобальных настроек, нужно было добавить к стандартным настройкам drupal для полей, свои настройки, для этого использовались различные хуки для переопределения форм, виджетов и пр. Это пожалуй была самая сложная часть выполнения.

Кроме того на странице администратора должна быть таблица со всем отчетами об ошибках с возможностью их удаления. Это было сделано в отдельном шаблоне.

После того как создали все настройки на странице администратора, перешли к созданию front-end части. В этой части нужно было создать шаблоны для popup. Для корректной работы нужно было создать для него стили и js код.

Для контроля за правильностью работы модуля созданы несколько тестов для контроля основных функций.

После выполнения задания, ссылка на удалённый репозиторий была отослана руковолителю практики для проверки и получения отзыва, после которого были выполнены правки и доработки.

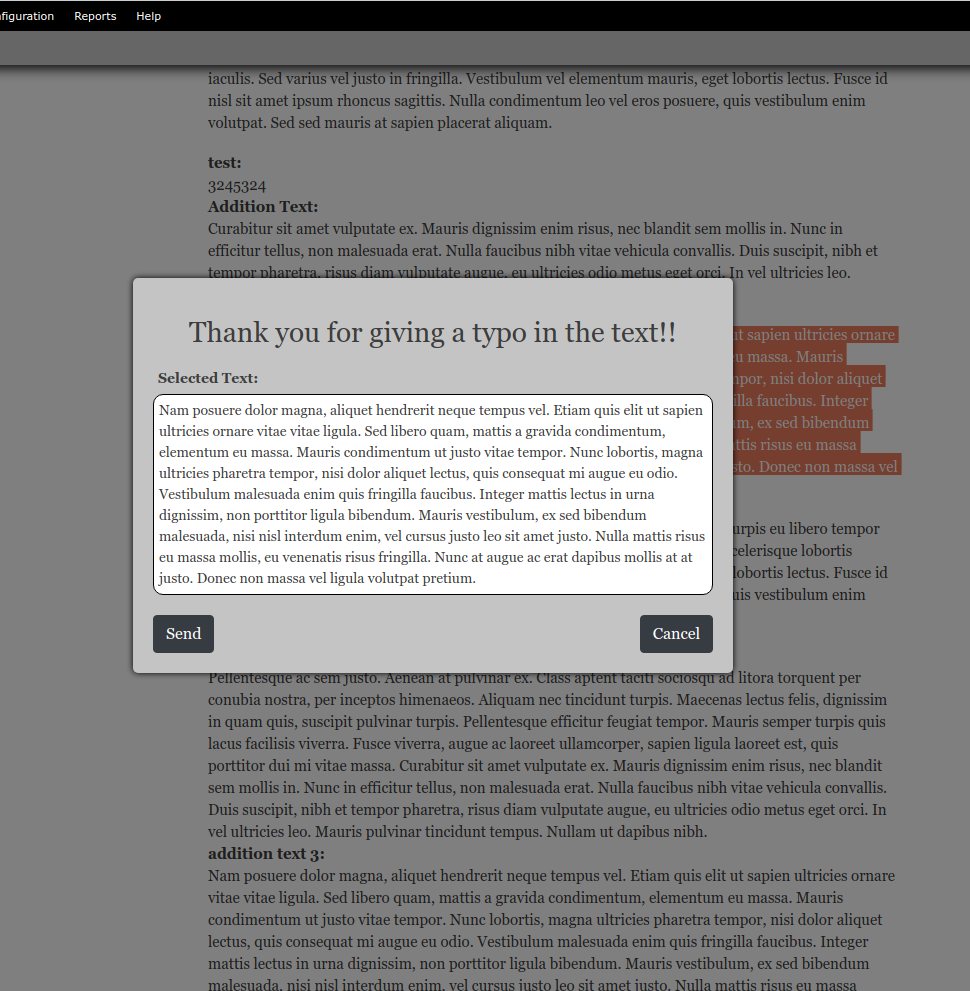
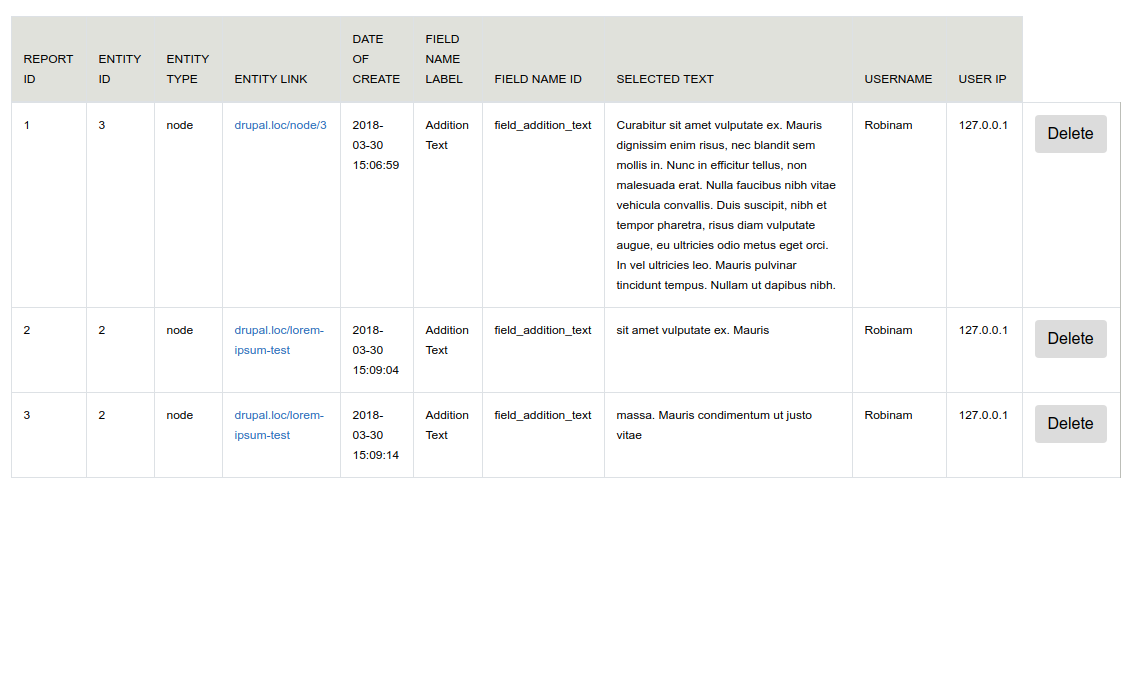
Ниже преведены несколько примеров работы модуля.

Рис. 1 Попап модуля

Рис. 2 Таблица с отчётами

# Знакомство с системами контроля версий

Для разработки модуля использовалась система контроля версий Git, с репозиторием на github. Ниже приведёно её описание.

Git (произн. «гит») — распределённая система управления версиями. Проект был создан Линусом Торвальдсом для управления разработкой ядра Linux, первая версия выпущена 7 апреля 2005 года. На сегодняшний день его поддерживает Джунио Хамано.

Среди проектов, использующих Git — ядро Linux, Swift, Android, Drupal, Cairo, GNU Core Utilities, Mesa, Wine, Chromium, Compiz Fusion, FlightGear, jQuery, PHP, NASM, MediaWiki, DokuWiki, Qt, ряд дистрибутивов Linux.

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. Например, Cogito является именно таким примером оболочки к репозиториям Git, а StGit использует Git для управления коллекцией исправлений (патчей).

Git поддерживает быстрое разделение и слияние версий, включает инструменты для визуализации и навигации по нелинейной истории разработки. Как и Darcs, BitKeeper, Mercurial, Bazaar и Monotone[en], Git предоставляет каждому разработчику локальную копию всей истории разработки, изменения копируются из одного репозитория в другой.

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 репозитории 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 репозиторий хранится в подкаталоге с названием «.git» в корневом каталоге рабочей копии дерева файлов, хранящегося в репозитории. Любое файловое дерево в системе можно превратить в репозиторий 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).

Нижний уровень git является так называемой контентно-адресуемой файловой системой. Инструмент командной строки git содержит ряд команд по непосредственной манипуляции этим репозиторием на низком уровне. Эти команды не нужны при нормальной работе с git как с системой контроля версий, но нужны для реализации сложных операций (ремонт поврежденного репозитория и так далее), а также дают возможность создать на базе репозитория git своё приложение.

Для каждого объекта в репозитории вычисляется SHA-1-хеш, и именно он становится именем файла, содержащего данный объект в каталоге .git/objects. Для оптимизации работы с файловыми системами, не использующими деревья для каталогов, первый байт хеша становится именем подкаталога, а остальные — именем файла в ней, что снижает количество файлов в одной директории (ограничивающий фактор производительности на таких устаревших файловых системах).

Все ссылки на объекты репозитория, включая ссылки на один объект, находящийся внутри другого объекта, являются SHA-1-хешами.

Кроме того, в репозитории существует каталог refs, который позволяет задать читаемые человеком имена для каких-то объектов Git. В командах Git оба вида ссылок — читаемые человеком из refs, и нижележащие SHA-1 — полностью взаимозаменяемы.

В классическом обычном сценарии в репозитории git есть три типа объектов — файл, дерево и «коммит». Файл есть какая-то версия какого-то пользовательского файла, дерево — совокупность файлов из разных поддиректорий, «коммит» (англ. commit — фиксация) — дерево и некая дополнительная информация (например, родительские коммиты, а также комментарий).

В репозитории иногда производится сборка мусора, во время которой устаревшие файлы заменяются на «дельты» между ними и актуальными файлами (то ест, актуальная версия файла хранится неинкрементально, инкременты используются только для возврата к предыдущим версиям), после чего данные «дельты» складываются в один большой файл, к которому строится индекс. Это снижает требования по ёмкости хранения.

Репозиторий Git бывает локальный и удалённый. Локальный репозиторий — это подкаталог .git, создается (в пустом виде) командой git init и (в непустом виде с немедленным копированием содержимого родительского удаленного репозитория и простановкой ссылки на родителя) командой git clone.

Практически все обычные операции с системой контроля версий, такие, как коммит и слияние, производятся только с локальным репозиторием. Удалённый репозиторий можно только синхронизировать с локальным как «вверх» (push), так и «вниз» (pull).

Наличие полностью всего репозитория проекта локально у каждого разработчика дает Git ряд преимуществ перед SVN. Так, например, все операции, кроме push и pull, можно осуществлять без наличия Интернет-соединения.

Очень мощной возможностью git являются ветви, реализованные куда более полно, чем в SVN: по сути, ветвь git есть не более чем именованная ссылка, указывающая на некий коммит в репозитории (используется подкаталог refs). Коммит без создания новой ветви всего лишь передвигает эту ссылку на себя, а коммит с созданием ветви — оставляет старую ссылку на месте, но создает новую на новый коммит, и объявляет её текущей. Заменить локальные девелоперские файлы на набор файлов из иной ветви, тем самым перейдя к работе с ней — так же тривиально.

Также поддерживаются суб-репозитории с синхронизацией текущих ветвей в них.

Команда push передаёт все новые данные (те, которых ещё нет в удаленном репозитории) из локального репозитория в репозиторий удаленный. Для исполнения этой команды необходимо, чтобы удалённый репозиторий не имел новых коммитов в себя от других клиентов, иначе push завершается ошибкой, и придётся делать pull и слияние.

Команда pull — обратна команде push. В случае, если одна и та же ветвь имеет независимую историю в локальной и в удалённой копии, pull немедленно переходит к слиянию.

Слияние в пределах разных файлов осуществляется автоматически (все это поведение настраивается), а в пределах одного файла — стандартным трёхпанельным сравнением файлов. После слияния нужно объявить конфликты как разрешенные.

Результатом всего этого является новое состояние в локальных файлах у того разработчика, что осуществил слияние. Ему нужно немедленно сделать коммит, при этом в данном объекте коммита в репозитории окажется информация о том, что коммит есть результат слияния двух ветвей и имеет два родительских коммита.

Кроме слияния, Git поддерживает ещё операцию перемещения (англ. rebase. Эта операция есть получение набора всех изменений в ветви А, с последующим их «накатом» на ветвь B. В результате ветвь B продвигается до состояния AB. В отличие от слияния, в истории ветви AB не останется никаких промежуточных коммитов ветви A (только история ветви B и запись о самом rebase, это упрощает интеграцию крупных и очень крупных проектов).

Также Git имеет временный локальный индекс файлов. Это — промежуточное хранилище между собственно файлами и очередным коммитом (коммит делается только из этого индекса). С помощью этого индекса осуществляется добавление новых файлов (git add добавляет их в индекс, они попадут в следующий коммит), а также коммит не всех изменённых файлов (коммит делается только тем файлам, которым был сделан git add). После git add можно редактировать файл далее, получатся три копии одного и того же файла — последняя, в индексе (та, что была на момент git add), и в последнем коммите.

Имя ветви по умолчанию: master. Имя удалённого репозитория по умолчанию, создаваемое git clone во время типичной операции «взять имеющийся проект с сервера себе на машину»: origin.

Таким образом, в локальном репозитории всегда есть ветвь master, которая есть последний локальный коммит, и ветвь origin/master, которая есть последнее состояние удалённого репозитория на момент завершения исполнения последней команды pull или push.

Команда fetch (частичный pull) — берёт с удаленного сервера все изменения в origin/master, и переписывает их в локальный репозиторий, продвигая метку origin/master.

Если после этого master и origin/master разошлись в стороны, то необходимо сделать слияние, установив master на результат слияния (команда pull есть fetch+merge). Далее возможно сделать push, отправив результат слияния на сервер и установив на него origin/master.

# Заключение

В результате прохождения технологической практики были решены задачи, поставленные руководителем практики от предприятия ЗАО «Итранзишэн»: изучение стека технологий (Drupal 7, PHP ,JavaScript,SQL). Написан модуль для Drupal. Проведено тестирование модуля.

Была изучена необходимая техническая документация, патентные и литературные источники, действующие стандарты, технические условия, положения и инструкции по разработке, наладке и эксплуатации, методы и технологии проектирования.

В соответствии с программой практики и индивидуальным заданием был осуществлен сбор, анализ, систематизация информации по теме задания и его непосредственное выполнение.

Также был подготовлен настоящий отчет и дневник с характеристикой руководителя от предприятия.

В ходе работы были получены навыки работы на предприятии, командной работы, общения с заказчиком. В ходе прохождения практики была изучена система контроля версий Git.