МИНИСТЕРСТВО НАУКИ и высшего образования РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования «МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ»

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

филиал «РКТ» МАИ в г. Химки Московской области

**Специальность 09.02.03 «Программирование в компьютерных системах»**

**ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ**

**ПМ.02 «Разработка и администрирование баз данных**»

**Студент**

**Группы МП-32-17\_Самуйлов Иван Александрович \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)**

**Руководитель**

**практики от организации \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)**

**Руководитель**

**практики от филиала Шумаев А.Ю. / ( \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ )**

**2020г.**

**ПРОГРАММА ПРОИЗВОДСТВЕННОЙ ПРАКТИКИ**

По специальности 09.02.03 «Программирование в компьютерных системах»

|  |  |  |
| --- | --- | --- |
| Дата | Наименование выполняемых работ | Количество дней практики |
| 11.06.20 | Ознакомление с программой практики. Прохождение инструктажа по технике безопасности и охране труда, изучение внутреннего распорядка организации и правил работы. Разработка задания на производственную практику. | 4 |
| 12.06.20 | Описание структуры сети, в которой может функционировать разработанная база данных. | 4 |
| 13.06.20 | Описание сетевого оборудования необходимого для работы с базой данных. | 4 |
| 15.06.20 | Описание программного обеспечения необходимого для работы с базой данных по сети. | 4 |
| 16.06.20-18.06.20 | Разработка клиентских интерфейсов для клиент-серверных приложений. | 16 |
| 20.06.20 | Создание инфокоммуникационной системы.  Выполнение сетевых настроек для взаимодействия с СУБД. | 4 |
| 22.06.20-24.06.20 | Создание концептуальной модели данных.  Создание логической модели данных.  Создание физической модели данных.  Описание механизмов обеспечения целостности базы данных. | 16 |
| 25.06.20-01.07.20 | Описание СУБД представленной базы данных и ее возможностей. Внесение различных данных в базу данных.  Структурирование запросов базы данных.  Определение методов создания хранимых процедур и триггеров. Создание хранимых процедур и триггеров базы данных. | 50 |
| 02.07.20-03.07.20 | Определение способов управления правами пользователей.  Описание распределения прав пользователей и управления ими в базе данных. | 12 |
| 03.07.20-04.07.20 | Определение методов создания и синхронизации реплик базы данных. Описание существующих механизмов репликации в базе данных. Определение методов создания резервных копий базы данных. Описание существующих механизмов резервного копирования в базе данных. | 12 |
| 06.07.20 | Составление отчёта по практике | 6 |
| 07.07.20 | Подготовка к защите отчёта по практике. | 6 |
| 08.07.20 | Итоговая аттестация по производственной практике - зачёт | 6 |

Руководитель практики от филиала «РКТ» МАИ преподаватель

Шумаев А.Ю.

Дата \_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(подпись)*

**АТТЕСТАЦИОННЫЙ ЛИСТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ**

студент \_\_\_\_\_\_\_\_\_Самуйлов Иван Александрович\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*ФИО*

обучающийся на 3-ем курсе по специальности СПО 09.02.03 «Программирование в компьютерных системах»

успешно прошел производственную практику по профессиональному модулю   
**ПМ.02 «Разработка и администрирование баз данных»**

в объеме 144 часа с 11.06.2020г. по 08.07.2020г.

в организации филиала «РКТ» МАИ

Виды и качество выполнения работ

|  |  |  |
| --- | --- | --- |
| Виды работ, выполненных обучающимся(ейся) во время практики | Объем работ | Качество выполнения работ в соответствии с технологией и (или) требованиями организации, в которой проходила практика |
| Ознакомление с программой практики. Прохождение инструктажа по технике безопасности и охране труда, изучение внутреннего распорядка организации и правил работы. Разработка задания на производственную практику. | 4 |  |
| Описание структуры сети, в которой может функционировать разработанная база данных. | 4 |  |
| Описание сетевого оборудования необходимого для работы с базой данных. | 4 |  |
| Описание программного обеспечения необходимого для работы с базой данных по сети. | 4 |  |
| Разработка клиентских интерфейсов для клиент-серверных приложений. | 16 |  |
| Создание инфокоммуникационной системы.  Выполнение сетевых настроек для взаимодействия с СУБД. | 4 |  |
| Создание концептуальной модели данных.  Создание логической модели данных.  Создание физической модели данных.  Описание механизмов обеспечения целостности базы данных. | 16 |  |
| Описание СУБД представленной базы данных и ее возможностей. Внесение различных данных в базу данных.  Структурирование запросов базы данных.  Определение методов создания хранимых процедур и триггеров. Создание хранимых процедур и триггеров базы данных. | 50 |  |
| Определение способов управления правами пользователей.  Описание распределения прав пользователей и управления ими в базе данных. | 12 |  |
| Определение методов создания и синхронизации реплик базы данных. Описание существующих механизмов репликации в базе данных. Определение методов создания резервных копий базы данных. Описание существующих механизмов резервного копирования в базе данных. | 12 |  |
| Составление отчёта по практике | 6 |  |
| Подготовка к защите отчёта по практике. | 6 |  |
| Итоговая аттестация по производственной практике - зачёт | 6 |  |

Руководитель практики от филиала «РКТ» МАИ Шумаев А.Ю.

|  |  |  |
| --- | --- | --- |
| Дата\_\_\_\_\_\_\_\_\_\_\_\_\_ | *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*  *подпись* | *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*  *Расшифровка подписи* |

**ДНЕВНИК** **ПРОХОЖДЕНИЯ ПРАКТИКИ**

|  |  |  |
| --- | --- | --- |
| Дата | Наименование выполняемых работ | Подпись  руководителя |
| 11.06.20 | Ознакомление с программой практики. Прохождение инструктажа по технике безопасности и охране труда, изучение внутреннего распорядка организации и правил работы. Разработка задания на производственную практику. |  |
| 12.06.20 | Описание структуры сети, в которой может функционировать разработанная база данных. |  |
| 13.06.20 | Описание сетевого оборудования необходимого для работы с базой данных. |  |
| 15.06.20 | Описание программного обеспечения необходимого для работы с базой данных по сети. |  |
| 16.06.20-18.06.20 | Разработка клиентских интерфейсов для клиент-серверных приложений. |  |
| 20.06.20 | Создание инфокоммуникационной системы.  Выполнение сетевых настроек для взаимодействия с СУБД. |  |
| 22.06.20-24.06.20 | Создание концептуальной модели данных.  Создание логической модели данных.  Создание физической модели данных.  Описание механизмов обеспечения целостности базы данных. |  |
| 25.06.20-01.07.20 | Описание СУБД представленной базы данных и ее возможностей. Внесение различных данных в базу данных.  Структурирование запросов базы данных.  Определение методов создания хранимых процедур и триггеров. Создание хранимых процедур и триггеров базы данных. |  |
| 02.07.20-03.07.20 | Определение способов управления правами пользователей.  Описание распределения прав пользователей и управления ими в базе данных. |  |
| 03.07.20-04.07.20 | Определение методов создания и синхронизации реплик базы данных. Описание существующих механизмов репликации в базе данных. Определение методов создания резервных копий базы данных. Описание существующих механизмов резервного копирования в базе данных. |  |
| 06.07.20 | Составление отчёта по практике |  |
| 07.07.20 | Подготовка к защите отчёта по практике. |  |
| 08.07.20 | Итоговая аттестация по производственной практике - зачёт |  |

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 7](#_Toc53922442)

[1. GITHUB 8](#_Toc53922443)

[1.1. О системе контроля версий 8](#_Toc53922444)

[1.2. Локальные системы контроля версий 8](#_Toc53922445)

[1.3. Централизованные системы контроля версий 9](#_Toc53922446)

[1.4. Распределённые системы контроля версий 10](#_Toc53922447)

[2. ОСНОВНЫЕ ОПЕРАЦИИ GITHUB 12](#_Toc53922448)

[2.1. Репозиторий 12](#_Toc53922451)

[2.2. Лес git’а 12](#_Toc53922452)

[2.3. Ветвление 14](#_Toc53922453)

[2.4. Создание новой ветки 14](#_Toc53922454)

[2.5. Переключение между ветками 14](#_Toc53922455)

[2.6. Слияние веток 14](#_Toc53922456)

[2.7. Фичи git’а 15](#_Toc53922457)

[3 BOOTSTRAP 16](#_Toc53922458)

[3.1 Определение 16](#_Toc53922459)

[3.2 Как подключить Bootstrap 17](#_Toc53922460)

[3.3 Фреймворк 18](#_Toc53922461)

[3.4 Использование 21](#_Toc53922462)

[ЗАКЛЮЧЕНИЕ 22](#_Toc53922463)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 23](#_Toc53922464)

# ВВЕДЕНИЕ

Практическая выполнена в соответствии с рабочей программой ПМ.02 «Разработка и администрирование баз данных». В работе представлены основные этапы анализа и разработки предметной области «Разработка и администрирование базы данных», навыки работы с GitHub и SQLite.

База данных играет важную роль в большинстве предметных областей. Благодаря динамической природе автоматизированных отраслей сейчас приложения требуют некоторых механизмов хранения, доступа и изменения данных. Поскольку важность баз данных стремительно растёт, реляционные системы управления базами данных набирают свою популярность.

В качестве программного обеспечения для создания интерфейса выбран редактор Notepad++. А для написания кода выбраны: язык гипертекстовой разметки HTML, язык таблиц стилей CSS и для работы с базами данных PHP. Использован веб-интерфейс phpmyadmin и СУБД SQLite.

В процессе выполнения проекта должен быть освоен ряд следующих профессиональных компетенций: ознакомление со структурой Git, реализация базы данных в конкретной системе управления базами данных. Получение знаний и опыта в области в технологии СУБД «SQLite» Также мною будут решаться вопросы администрирования базы данных и реализовываться методы и технологии её защиты.

# GITHUB

Эта глава о том, как начать работу с Git. Вначале изучим основы систем контроля версий, затем перейдём к тому, как запустить Git на вашей ОС и окончательно настроить для работы. В конце главы вы уже будете знать, что такое Git и почему им следует пользоваться, а также получите окончательно настроенную для работы систему.

* 1. О системе контроля версий

Что такое «система контроля версий» и почему это важно? Система контроля версий — это система, записывающая изменения в файл или набор файлов в течение времени и позволяющая вернуться позже к определённой версии. Для контроля версий файлов в этой книге в качестве примера будет использоваться исходный код программного обеспечения, хотя на самом деле вы можете использовать контроль версий практически для любых типов файлов.

Если вы графический или web-дизайнер и хотите сохранить каждую версию изображения или макета (скорее всего, захотите), система контроля версий (далее СКВ) — как раз то, что нужно. Она позволяет вернуть файлы к состоянию, в котором они были до изменений, вернуть проект к исходному состоянию, увидеть изменения, увидеть, кто последний менял что-то и вызвал проблему, кто поставил задачу и когда и многое другое. Использование СКВ также значит в целом, что, если вы сломали что-то или потеряли файлы, вы спокойно можете всё исправить. В дополнение ко всему вы получите всё это без каких-либо дополнительных усилий.

* 1. Локальные системы контроля версий

Многие люди в качестве метода контроля версий применяют копирование файлов в отдельную директорию (возможно даже, директорию с отметкой по времени, если они достаточно сообразительны). Данный подход очень распространён из-за его простоты, однако он невероятно сильно подвержен появлению ошибок. Можно легко забыть, в какой директории вы находитесь, и случайно изменить не тот файл или скопировать не те файлы, которые вы хотели.

Для того, чтобы решить эту проблему, программисты давным-давно разработали локальные СКВ с простой базой данных, которая хранит записи о всех изменениях в файлах, осуществляя тем самым контроль ревизий.



Рисунок 1. «Локальный контроль версий»

Одной из популярных СКВ была система RCS, которая и сегодня распространяется со многими компьютерами. RCS хранит на диске наборы патчей (различий между файлами) в специальном формате, применяя которые она может воссоздавать состояние каждого файла в заданный момент времени.

* 1. Централизованные системы контроля версий

Следующая серьёзная проблема, с которой сталкиваются люди, — это необходимость взаимодействовать с другими разработчиками. Для того, чтобы разобраться с ней, были разработаны централизованные системы контроля версий (ЦСКВ). Такие системы, как CVS, Subversion и Perforce, используют единственный сервер, содержащий все версии файлов, и некоторое количество клиентов, которые получают файлы из этого централизованного хранилища. Применение ЦСКВ являлось стандартом на протяжении многих лет.



Рисунок 2. «Централизованный контроль версий»

Такой подход имеет множество преимуществ, особенно перед локальными СКВ. Например, все разработчики проекта в определённой степени знают, чем занимается каждый из них. Администраторы имеют полный контроль над тем, кто и что может делать, и гораздо проще администрировать ЦСКВ, чем оперировать локальными базами данных на каждом клиенте.

Несмотря на это, данный подход тоже имеет серьёзные минусы. Самый очевидный минус — это единая точка отказа, представленная централизованным сервером. Если этот сервер выйдет из строя на час, то в течение этого времени никто не сможет использовать контроль версий для сохранения изменений, над которыми работает, а также никто не сможет обмениваться этими изменениями с другими разработчиками. Если жёсткий диск, на котором хранится центральная БД, повреждён, а своевременные бэкапы отсутствуют, вы потеряете всё — всю историю проекта, не считая единичных снимков репозитория, которые сохранились на локальных машинах разработчиков. Локальные СКВ страдают от той же самой проблемы: когда вся история проекта хранится в одном месте, вы рискуете потерять всё.

* 1. Распределённые системы контроля версий

Здесь в игру вступают распределённые системы контроля версий (РСКВ). В РСКВ (таких как Git, Mercurial, Bazaar или Darcs) клиенты не просто скачивают снимок всех файлов (состояние файлов на определённый момент времени) — они полностью копируют репозиторий. В этом случае, если один из серверов, через который разработчики обменивались данными, умрёт, любой клиентский репозиторий может быть скопирован на другой сервер для продолжения работы. Каждая копия репозитория является полным бэкапом всех данных.



Рисунок 3. «Распределённый контроль версий»

Более того, многие РСКВ могут одновременно взаимодействовать с несколькими удалёнными репозиториями, благодаря этому вы можете работать с различными группами людей, применяя различные подходы единовременно в рамках одного проекта. Это позволяет применять сразу несколько подходов в разработке, например, иерархические модели, что совершенно невозможно в централизованных системах.

## ОСНОВНЫЕ ОПЕРАЦИИ GITHUB

Начнем, естественно, с загрузки. Надеемся, какая у вас операционная система, вы знаете. И сразу предупредим новичков: не путайте git и GitHub — это разные вещи. Нас интересует именно git, а GitHub (или ему подобные сервисы вроде Bitbucket или GitLab) — это по сути хостинг для проектов, использующих git.

2. 1. Репозиторий

Итак, вот у вас уже есть git. Теперь нужно создать хранилище версий для него. Запомните, это хранилище называется репозиторий (англ. repository) — при случае можете вставить где-нибудь это словечко. В зависимости от того, какая у вас оболочка, соответствующей командой создайте новую директорию, откройте ее (в командной строке, она же оболочка, а не проводником или чем-то подобным) и выполните:

git init

Все, локальный репозиторий в этой папке создан. То, что здесь сейчас хранится, будет бекапом, поэтому, чтобы его не испортить, создадим рабочую копию (англ. check out) локальной версии:

git clone [url]

Где [url] — это путь до клонируемого репозитория. Мы разбираем сейчас случай, когда вы создаете рабочую копию собственного репозитория, поэтому в качестве [url] здесь вам нужно указать путь до директории, для которой мы выполняли git init.

Но если вы крутой чувак и уже работаете с удаленным сервером, то вот такая команда будет для вас в самый раз:

git clone username@host:/path/to/repository

* 1. Лес git’а

Немного теории. Git в своей работе управляет тремя структурами, которые называются деревьями. Первое — это рабочая директория, в ней хранятся файлы, с которыми вы прямо сейчас работаете. Ну, она ж рабочая, логично. Второе — это Index, этакий чек-поинт, который позволяет вам вносить изменения и ничего не портить. А третье — это HEAD, который указывает на последний сделанный вами коммит. (Чтобы вы не запутались в терминологии: коммит (англ. commit) — это сохранение состояния проекта в репозиторий. Короче, считайте, новая версия.)

Так вот, чтобы вы не заблудились в этих трех соснах, запомните две крутые команды: add и commit. Они позволят вашей работе спокойно бродить по git’у, сохраняясь, куда надо. Если вы придумали что-то гениальное и тут же внесли изменение в рабочую копию проекта, то не спешите сразу коммитить! Сначала испытайте в Index’е, для этого выполните:

git add [имя\_файла]

если вы внесли изменение только в один файл, или

git add \*

если вы хорошо потрудились поменяли сразу кучу исходников. Изменения положительны? Хорошо потестили? Тогда скорее коммитить:

git commit -m "Commit message"

Вы, конечно же, плохой парень и не оставляете комментариев в коде. Но git — другое дело. Не ленитесь оставлять поясняющие сообщения: будьте уверены, вам хватит других проблем, кроме как разбираться, что же поменялось в этом коммите по сравнению с прошлой версией. Поясняющая картинка:

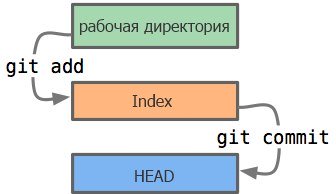
[](https://s3.tproger.ru/uploads/2016/06/EXcwUk86PWs.jpg)

Рисунок 4. «Git add и commit»

Теперь файл(-ы) прочно обосновались в HEAD вашей рабочей локальной копии. Оттуда их не выгнать, но в вашем удаленном репозитории их все еще нет. Давайте сунем их еще и туда! Используйте:

git push origin master

Только вместо master напишите название нужной ветки. Ах да, вы же еще не знаете, что такое ветки. Ну ладно, пока что запомните это место, а когда прочитаете про ветвление, вернетесь сюда.

Ах да, для крутых чуваков, работающих с серверами (как раз тут уместно говорить про GitHub, например), команда будет такой:

git remote add origin [сервер]

* 1. Ветвление

По-английски эта штука зовется branching — лучше как следует вникните в этот вопрос и почитайте про ветвление подробнее, я вас с ним только познакомлю. Ветвление используется для одновременной и независимой разработки разных фич (ну, или накопления большего количества багов, ведь исходного кода становится больше). Основной веткой является master — она появляется при создании репозитория. Другие ветки — это песочницы, когда достаточно в них наиграетесь, слейте в единое целое в master. Сейчас поясню, как это делается.

* 1. Создание новой ветки

Вот вы решили проработать какую-нибудь новую фичу. Создайте для нее новую ветку:

git checkout -b [новая\_ветка]

Ах да, фантазия-то у вас, наверное, работает на полную катушку, ну да поумерьте её в деле именования веток: назвать ветку можно только именем, допустимым для переменной в вашем любимом языке.

* 1. Переключение между ветками

Надо сделать перерыв в работе с этой фичей и переключиться на другую ветку? Используйте (если работаете с локальным репозиторием, то указывать его имя не обязательно):

git checkout [репозиторий]/[ветка]

Ну, а если вы уже совсем не хотите с ней работать, то удалите ее совсем:

git branch -d [ветка]

Со своей веткой вы можете творить любые непотребства: ее никто не увидит, пока вы сами ее не пропушите в удаленный репозиторий командой:

git push origin [ветка]

* 1. Слияние веток

Чтобы слить ветку в ту, с которой вы сейчас работаете, используйте:

git merge [ветка]

Но, понятное дело, это все приводит к конфликтам. И это реально проблема. Так что попробуйте исправлять все ручками прямо в директории с репозиторием. Только потом не забудьте пометить, что вы их «слили»:

git add [имя\_файла]

Кстати, ветки можно сравнить:

git diff [одна\_ветка] [другая\_ветка]

Так, теперь приступим к более решительным действиям. Будем обновлять свой репозиторий в соответствии с самым свежим коммитом. Сделать это очень просто (а вот вернуть обратно не очень, поэтому трижды подумайте, прежде чем совершать эту ужасную ошибку):

git pull

Я, конечно, понимаю, что вы слишком круты, чтобы оставлять какие-либо пометки на будущее — все держите в голове — но все-таки рекомендую вам оставлять тэги. И это не моя выдумка, так делают многие:

git tag [tag] [первые\_десять\_символов\_соответствующего\_коммита]

Вы не знаете, какие первые символы у имени нужного коммита? Не беда, смотрите в историю репозитория — его лог:

git log

Там есть куча разных параметров для использования этой полезной штуковины, ну да погуглите их сами. Ах да, кстати, мы уже писали как-то про то как сделать git log более информативным.

* 1. Фичи git’а

Если вы ленивый, и вам не охота по-трупрогерски все писать в оболочке своей ОСи, то можете использовать GUI git’а:

gitk

В источнике найдете еще кучу других GUI-шек.  
Если вам стандартный вывод git’а кажется скучным, раскрасьте его:

git config color.ui true

Ну, и есть еще такая штука — интерактивное индексирование. Когда у вас будет уже достаточно большой проект, то ужать представление index’а в log’е можно будет так:

git add -i

# 3 BOOTSTRAP

## 3.1 Определение

Bootstrap - это свободный набор инструментов для создания сайтов и веб-приложений. Включает в себя HTML- и CSS-шаблоны оформления для типографики, веб-форм, кнопок, меток, блоков навигации и прочих компонентов веб-интерфейса, включая JavaScript-расширения.

Основные инструменты:

• Сетки — заранее заданные размеры колонок, которые можно сразу же использовать, например, ширина колонки 140 px относится к классу span2 (col-md-2 в третьей версии фреймворка), который можно использовать в CSS-описании документа.

• Шаблоны — фиксированный или резиновый шаблон документа.

• Типографика — описания шрифтов, определение некоторых классов для шрифтов, таких как код, цитаты и т. п.

• Медиа — предоставляет некоторое управление изображениями и видео.

• Таблицы — средства оформления таблиц, вплоть до добавления функциональности сортировки.

• Формы — классы для оформления форм и некоторых событий, происходящих с ними.

• Навигация — классы оформления для панелей, вкладок, перехода по страницам, меню и панели инструментов.

• Алерты — оформление диалоговых окон, подсказок и всплывающих окон.

Преимущества и недостатки фреймворка Bootstrap:

• Высокая скорость создания качественной адаптивной вёрстки даже начинающими веб-разработчиками (достигается это благодаря использованию готовых компонентов, созданных профессионалами);

• Кроссбраузерность и кроссплатформеннось (корректное отображение и работа сайта во всех поддерживаемых этим фреймворком браузерах и операционных системах);

• Наличие большого количество готовых хорошо продуманных компонентов, протестированных огромным сообществом веб-разработчиков на различных устройствах;

• Возможность настройки под свой проект, достигается это посредством изменения SCSS переменных и использования Bootstrap миксинов (можно изменить количество колонок, цвета, радиуса скруглений углов элементов, отступы между колонками и многое другое);

• Низкий порог вхождения; для работы с фреймворком не обязательно иметь глубокие знания по HTML, CSS, JavaScript и jQuery (достаточно знать только основы вышеперечисленных технологий);

• Наличие хорошо продуманного дизайна компонентов и согласованности (в Bootstrap все компоненты выполнены в едином стиле);

• Наличие огромного сообщества, большого количества статей, рецептов и видеоматериалов; всё это при желании поможет не только хорошо разобраться в фреймворке, но и найти ответы практически на любые вопросы.

## 3.2 Как подключить Bootstrap

Зайдите на официальный сайт фреймворка - https://getbootstrap.com/ и отметьте все пункты, которые понадобятся для работы. Дополнительные возможности могут быть добавлены/удалены при необходимости. После того как вы нажмете на кнопку «Compile and Download» («Компилировать и Загрузить»), скачайте архив и распакуйте его.

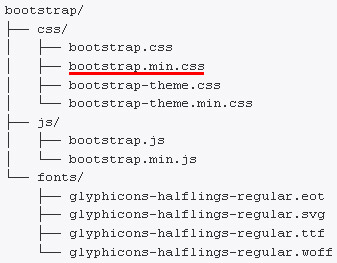


Рисунок 5. «Подключение»

Для базовых вещей вам потребуется всего один файл. Он подчеркнут красным цветом. Сохраните его в папке с вашим CSS и подключите в области head на странице.

<link rel='stylesheet' href='/styles/bootstrap.min.css' type='text/css' media='all'>

Иконочный шрифт находится в папке fonts (шрифты), и, если он вам действительно нужен, убедитесь в том, чтобы он остался в этой же папке. Сохраните структуру папок, чтобы все работало корректно.

Иконочный шрифт позволит вам забыть о раздражающих спрайтах для иконок и очень сильно упростит вашу жизнь. Единственное ограничение заключается в том, что к одной иконке можно применить только один цвет.

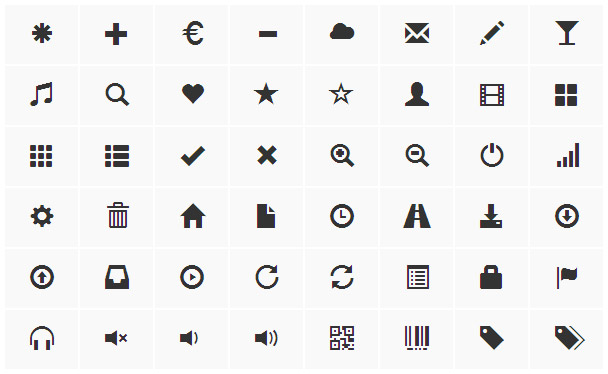


Рисунок 6. «Иконочный шрифт»

Вам предоставляется выбор из 200 иконок, и вы можете добавить их на ваш веб-сайт следующим образом: <span class="glyphicon glyphicon-star"></span>.

## 3.3 Фреймворк

­— Шаблоны Bootstrap:

Шаблоны в Bootstrap позволяют вам изменять уже модифицированные элементы под ваши нужды. Многие разработчики предлагают использовать их собственные шаблоны (платные или бесплатные). Подключаются шаблоны Bootstrap очень просто: после подключения самого Bootstrap вы просто добавляете вызов CSS шаблона.

— Содержимое фреймворка:

Если вы остановили свой выбор на Bootstrap, то данный фреймворк позволит вам существенно сэкономить время разработки фронтенд части проекта благодаря большому количеству готовых компонентов. Позже мы рассмотрим основные компоненты, которыми пользуются почти всех фронтенд разработчики. Следует здесь отметить, что Bootstrap — это, так сказать, набор из трех фреймворков: CSS/HTML, JS компоненты и иконочный шрифт.

— Сетка:

Наличие сетки является базовым требованием для хорошего макета. Сетка — это мощный инструмент для расположения блочного контента и вложенных элементов. С помощью префиксов вы можете указать, каким образом должны отображаться блоки, в зависимости от типа устройства, на котором просматривается веб-сайт.

Например, класс «col-xs-» будет использоваться для мобильных телефонов с шириной экрана менее 768 пикселей, а класс «col-lg-» — для устройств с шириной экрана более 1170 пикселей. Bootstrap разделяет ширину родительского блока на 12 равных блоков, которые мы можем использовать. Некоторые блоки могут комбинироваться, чтобы получить, например, три колонки: две 25% «col-lg-3» и одна 50% «col-lg-6».

Визуально страница может быть представлена в любом желаемом виде. На рисунке 5 представлен пример блоков.

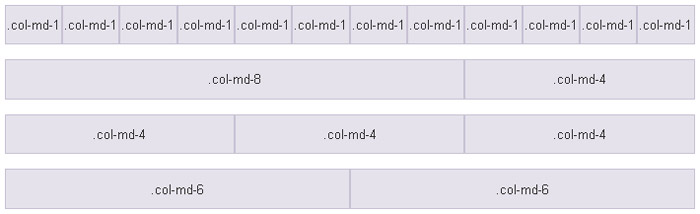


Рисунок 7. «Пример блоков»

Если вы выйдете за пределы 12 блоков, то оставшаяся часть будет отображена ниже, под другими блоками, поэтому будьте внимательны.

* Типографика:

В дополнение к возможности изменять блоки на макете и структуру страницы Bootstrap позволяет вам форматировать параметры шрифта: абзацы, цитаты, заголовки, подзаголовки, разные размеры текста, вставки кода и т.д. В большинстве случаев вам не нужно изменять настройки текста по умолчанию, потому что все внешние отступы, заголовки, межстрочные интервалы (интерлиньяж) и т.д. уже тщательно подобраны.

Большое внимание уделяется семантике: основной заголовок может быть задан в виде тега <h1>heading</h1>, но также и в виде <div class=»h1″>heading</div> – оба варианта будут выглядеть одинаково, но второй может быть использован любое количество раз на странице. На рисунке 6 представлены все различия заголовков.

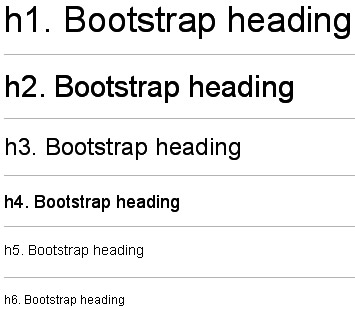


Рисунок 8. «Heading»

Если вы хотите попробовать поработать с Bootstrap, но не хотите сразу разбираться во всех «тонкостях», то начните с малого: создайте текстовую страничку с какими-то новостями. Просто проведите линии под заголовками и подзаголовками, вставьте изображения, добавьте какие-нибудь цитаты, списки и наслаждайтесь тем, как легко данный фреймворк справляется с данными задачами.

* Оповещения (алерты):

Любое оповещение может быть представлено в 4 стандартных форматах: положительный, информационный, предупреждающий, отрицательный. Пример таких оповещений представлен на рисунке 6.

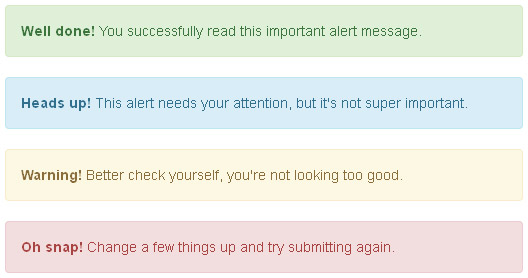


Рисунок 9. «Оповещения»

## 3.4 Использование

При разработке базы данных необходимо было выбрать инструмент Bootstrap, использовался редактор Sublime Text 3. Для базы данных по предметной области «Магазин детских игрушек», в которой главным действующим лицом является администратор магазина детских игрушек, был создан интерфейс с возможностью просмотра, удаления, добавления и обновления записей. С помощью Bootstrap мы выбрали один из подходов к организации HTML. Я создал директорию для общего проекта и скопировал в нее файлы Bootstrap, сохранив исходную структуру директорий. Поместив свои собственные HTML-файлы на самый верхний уровень, а файлы CSS, JavaScript в соответствующие поддиректории Bootstrap (с именами css, js соответственно). Общая структура изображена на рисунке 10.

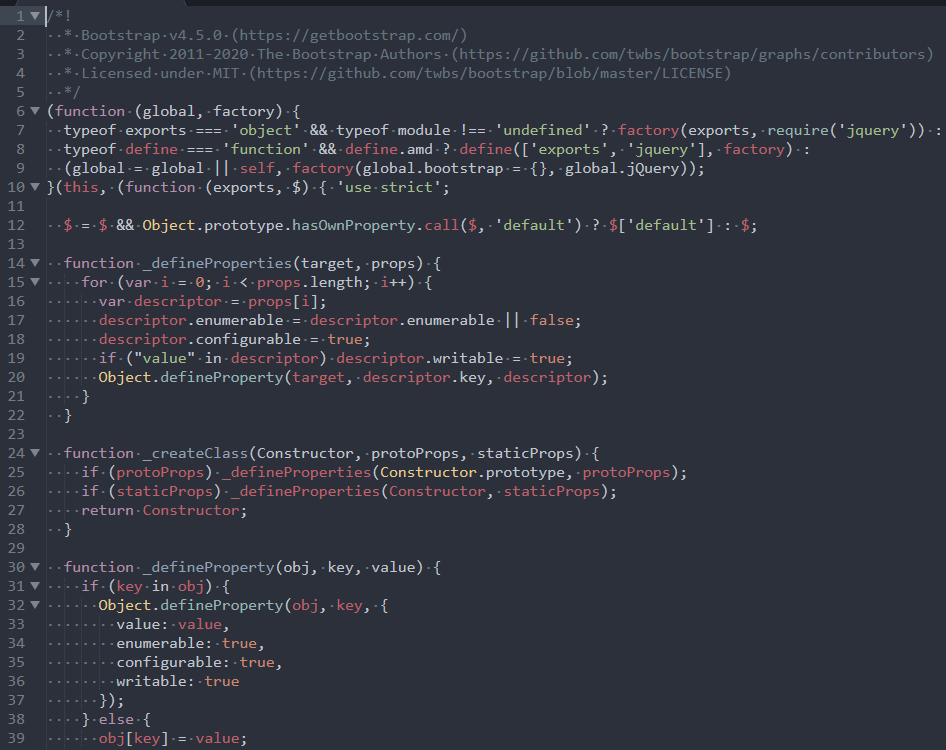


Рисунок 10 – «Bootstrap»

# ЗАКЛЮЧЕНИЕ

Мы рассмотрели большинство основных способов настройки клиента и сервера Git с тем, чтобы он был максимально соответствовал вашим рабочим процессам и проектам.

Мы узнали о всевозможных настройках, атрибутах файлов и о перехватчиках событий, а также рассмотрели пример настройки сервера с соблюдением политики. Теперь вам должно быть по плечу заставить Git подстроиться под практически любой тип рабочего процесса, который только можно вообразить.

В результате выполнения и изучения bootstrap фреймворка были применены знания и освоены профессиональные компетенций. Были закреплены полученные навыки и знания с фреймворком bootstrap. В двух словах в нём заложены основные css правила, которые облегчают вёрстку, очень прост в освоение при детальном изучение.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Github: [Электронный ресурс]/ https://tproger.ru/translations/difference-between-git-and-github - статья в интернете.
2. Bootstrap: [Электронный ресурс]/ https://webformyself.com/znakomstvo-s-bootstrap - статья в интернете.
3. Bootstrap документация на русском [Электронный ресурс]/ https://bootstrap-4.ru/ - статься в интернете.