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\_\_\_\_\_\_\_\_\_Сенькин Д.А.\_\_ / ( \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ )**

**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 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Сенькин Даниил Алексеевич\_\_\_\_\_\_\_\_\_\_\_

*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

[СОДЕРЖАНИЕ 6](#_Toc54187513)

[ВВЕДЕНИЕ 7](#_Toc54187514)

[1. GITHUB 8](#_Toc54187515)

[1.1. О системе контроля версий 8](#_Toc54187516)

[1.2. Локальные системы контроля версий 8](#_Toc54187517)

[1.3. Централизованные системы контроля версий 9](#_Toc54187518)

[1.4. Распределённые системы контроля версий 10](#_Toc54187519)

[2. ОСНОВНЫЕ ОПЕРАЦИИ GITHUB 12](#_Toc54187520)

[2.1. Репозиторий 12](#_Toc54187523)

[2.2. Лес git’а 12](#_Toc54187524)

[2.3. Ветвление 14](#_Toc54187525)

[2.4. Создание новой ветки 14](#_Toc54187526)

[2.5. Переключение между ветками 14](#_Toc54187527)

[2.6. Слияние веток 15](#_Toc54187528)

[2.7. Фичи git’а 15](#_Toc54187529)

[3. Текстовый редактор Atom 16](#_Toc54187530)

[ЗАКЛЮЧЕНИЕ 18](#_Toc54187531)

[ПРИЛОЖЕНИЯ 19](#_Toc54187532)

# 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 работы с текстовым редактором «Atom».

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 Atom. 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 Jquery.

# GITHUB

* 1. О системе контроля версий

Git — это набор консольных утилит, которые отслеживают и фиксируют изменения в файлах (чаще всего речь идет об исходном коде программ, но вы можете использовать его для любых файлов на ваш вкус). С его помощью вы можете откатиться на более старую версию вашего проекта, сравнивать, анализировать, сливать изменения и многое другое. Этот процесс называется контролем версий.

Системой контроля версий является программный продукт, который запоминает все ваши модификации данных и, при необходимости, позволяет выполнить откат. А также дополнительной функцией является возможность определить, кто внес различные изменения в документ.

* 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://tproger.ru/events/croc-it-summer-school/?utm_source=in_text)

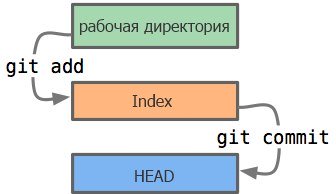


Рисунок 4 - Git add и commit

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

# Текстовый редактор Atom

Atom (в прошлом Atomicity) - бесплатный текстовый редактор с открытым исходным кодомдля Linux, macOS, Windows с поддержкой плагинов, написанных на Node.js, и встраиваемых под управлением Git. Большинство плагинов имеют статус свободного программного обеспечения, разрабатываются и поддерживаются сообществом.

Atom основан на Electron (ранее известный как Atom Shell) - фреймворке кросс-платформенной разработки с использованием Chromium и io.js. Редактор написан на CoffeeScript и LESS. Версия 1.0 была выпущена 25 июня 2015 г. На рисунке 5 представлен интерфейс текстового редактора Atom.

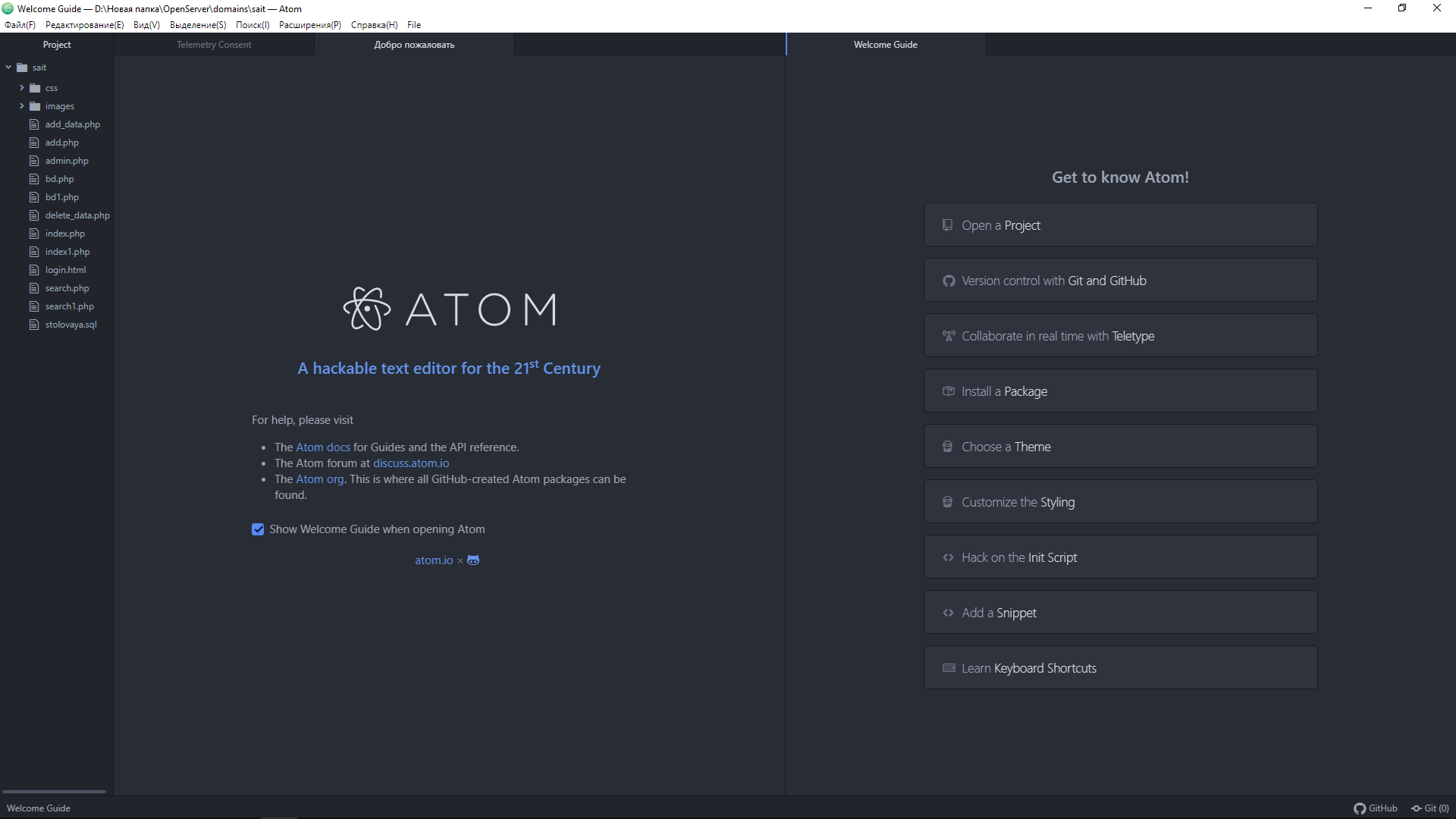


Рисунок 5 - Интерфейс текстового редактора Atom

Преимущества данного текстового редактора заключаются в том что для удобства написания разметки или кода можно устанавливать специальные аддоны которые в дальнейшем будут способствовать написанию. Также в нем присутствует система контроля версии Git и возможность загрузки проекта на GitHub.

Также в него встроена система распознавания текста для того чтобы предлагать пользователю тот или иной тег для упрощения и ускорения написания. На рисунках 6-7 представлен пример распознавания текста.

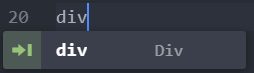


Рисунок 6 - Пример распознавания текста

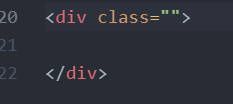


Рисунок 7 - Пример распознавания текста

В текстовом редакторе Atom реализована система сохранения последних проектов и открытых файлов для удобства. На рисунке 8 представлена система сохранения последних проектов и открытых файлов.

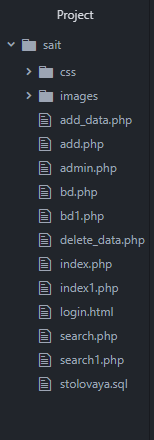


Рисунок 1 - Система сохранения последних проектов и открытых файлов.

# 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.

Текстовый редактор Atom является удобным инструментом для написания html, php, css, js файлов. Также в следствии его эксплуатации были открыты разные функции по типу: размещение проекта на GitHub, установка различных аддонов для упрощения и ускорения написания скриптов и кодов.

# ПРИЛОЖЕНИЯ

ПРИЛОЖЕНИЕ А

<?php

session\_start();

if($\_GET['do'] == 'logout'){

unset($\_SESSION['admin']);

session\_destroy();

}

if($\_SESSION['admin'] != "admin"){

header("Location: login.php");

exit;

}

?>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

<meta name="description" content="">

<meta name="author" content="">

<title>toy-shop</title>

<!-- Bootstrap core CSS -->

<link href="vendor/bootstrap/css/bootstrap.min.css" rel="stylesheet">

<!-- Custom styles for this template -->

<link href="css/shop-homepage.css" rel="stylesheet">

</head>

<body>

<!-- Navigation -->

<nav class="navbar navbar-expand-lg navbar-dark bg-dark fixed-top">

<div class="container">

<a class="navbar-brand" href="#">РњР°РіР°Р·РёРЅ РёРіСЂСѓС€РµРє</a>

<button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarResponsive" aria-controls="navbarResponsive" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarResponsive">

<ul class="navbar-nav ml-auto">

<li class="nav-item active">

<a class="nav-link" href="#">Р“Р»Р°РІРЅР°СЏ

<span class="sr-only">(current)</span>

</a>

</li>

<li class="nav-item">

<a class="nav-link" href="akcii.php">РђРєС†РёРё</a>

</li>

<li class="nav-item">

<a class="nav-link" href="prt.php">РќР°С€Рё РїР°СЂС‚РЅРµСЂС‹</a>

</li>

<li class="nav-item">

<a class="nav-link" href="company.html">Рћ РєРѕРјРїР°РЅРёРё</a>

</li>

</ul>

<script type="text/javascript">

function Input(){

login\_ok = false;

user\_name = "";

password = "";

user\_name = prompt("Р›РѕРіРёРЅ","");

user\_name = user\_name.toLowerCase();

password = prompt("РџР°СЂРѕР»СЊ","");

password = password.toLowerCase();

if (user\_name=="123" && password=="123") {

login\_ok = true;

window.location = "admin.php";

}

if (login\_ok==false) alert("РќРµРІРµСЂРЅС‹Р№ Р»РѕРіРёРЅ РёР»Рё РїР°СЂРѕР»СЊ!");

}

</script>

<form>

<input type="button" value="РђРґРјРёРЅ РїР°РЅРµР»СЊ" onClick="Input()">

</form>

<a class="nav-link" href="index.php?do=logout">Р’С‹Р№С‚Рё</a>

</div>

</div>

</nav>

<!-- Page Content -->

<div class="container">

<div class="row">

<?php

$mysqli = mysqli\_init();

if (!$mysqli) {

die('mysqli\_init Р·Р°РІРµСЂС€РёР»Р°СЃСЊ РїСЂРѕРІР°Р»РѕРј');

}

if (!$mysqli->options(MYSQLI\_INIT\_COMMAND, 'SET AUTOCOMMIT = 0')) {

die('РЈСЃС‚Р°РЅРѕРІРєР° MYSQLI\_INIT\_COMMAND Р·Р°РІРµСЂС€РёР»Р°СЃСЊ РїСЂРѕРІР°Р»РѕРј');

}

if (!$mysqli->options(MYSQLI\_OPT\_CONNECT\_TIMEOUT, 5)) {

die('РЈСЃС‚Р°РЅРѕРІРєР° MYSQLI\_OPT\_CONNECT\_TIMEOUT Р·Р°РІРµСЂС€РёР»Р°СЃСЊ РїСЂРѕРІР°Р»РѕРј');

}

if (!$mysqli->real\_connect('localhost', 'root', 'root', 'toy-store')) {

die('РћС€РёР±РєР° РїРѕРґРєР»СЋС‡РµРЅРёСЏ (' . mysqli\_connect\_errno() . ') ' . mysqli\_connect\_error());

}

?>

<?php

$products = array();

if ($result = $mysqli->query('SELECT \* FROM products')) {

while($tmp = $result->fetch\_assoc()) {

$products[] = $tmp;

}

$result->close();

}

?>

<div class="row">

<?php foreach($products AS $product) {?>

<div class="col-sm-4 col-lg-4 col-md-4">

<div class="thumbnail">

<img src="<?php echo $product['img'];?>" alt="">

<div class="caption">

<h4><?php echo $product['title'];?></h4>

<h4 class="pull-right">в‚Ѕ<?php echo $product['price'];?></h4>

<p> <img src="https://icons.iconarchive.com/icons/hopstarter/soft-scraps/24/Button-Info-icon.png">

<?php echo $product['intro'];?></p>

</div>

<div class="ratings">

<p class="pull-right"> <img src="https://icons.iconarchive.com/icons/double-j-design/origami-colored-pencil/24/blue-talk-icon.png">

<?php echo $product['reviews\_count'];?> РѕС‚Р·С‹РІРѕРІ</p>

<p>

</p>

</div>

</div>

</div>

<?php } ?>

</div>

</div>

<!-- /.col-lg-3 -->

<div class="col-lg-9">

<div id="carouselExampleIndicators" class="carousel slide my-4" data-ride="carousel">

<ol class="carousel-indicators">

<li data-target="#carouselExampleIndicators" data-slide-to="0" class="active"></li>

<li data-target="#carouselExampleIndicators" data-slide-to="1"></li>

<li data-target="#carouselExampleIndicators" data-slide-to="2"></li>

</ol>

<div class="carousel-inner" role="listbox">

<div class="carousel-item active">

<img class="d-block img-fluid" src="https://blog.fromjapan.co.jp/en/wp-content/uploads/2019/06/AnimalCrossing\_Banner-1.png" alt="First slide">

</div>

<div class="carousel-item">

<img class="d-block img-fluid" src="http://www.jeux-de-societe.info/wp-content/uploads/2019/11/Casse-tetes-en-bois.jpeg" alt="Second slide">

</div>

<div class="carousel-item">

<img class="d-block img-fluid" src="https://fyi.extension.wisc.edu/farmstress/files/2019/04/toy-tractor.jpg" alt="Third slide">

</div>

</div>

<a class="carousel-control-prev" href="#carouselExampleIndicators" role="button" data-slide="prev">

<span class="carousel-control-prev-icon" aria-hidden="true"></span>

<span class="sr-only">Previous</span>

</a>

<a class="carousel-control-next" href="#carouselExampleIndicators" role="button" data-slide="next">

<span class="carousel-control-next-icon" aria-hidden="true"></span>

<span class="sr-only">Next</span>

</a>

</div>

<div class="row">

<div class="col-lg-4 col-md-6 mb-4">

<div class="card h-100">

<a href="#"><img class="card-img-top" src="https://birtop.ru/wp-content/uploads/2019/08/2b20a4868e85ff41eb967ea3037047ff.jpg" alt=""></a>

<div class="card-body">

<h4 class="card-title">

<a href="#">РњР°С„РёСЏ</a>

</h4>

<h5>600 в‚Ѕ</h5>

<p class="card-text">РљР»Р°СЃСЃРёС‡РµСЃРєР°СЏ РЅР°СЃС‚РѕР»СЊРЅР°СЏ РёРіСЂР°. РџСЃРёС…РѕР»РѕРіРёС‡РµСЃРєР°СЏ РєРѕРјР°РЅРґРЅР°СЏ РёРіСЂР° СЃ РєСЂРёРјРёРЅР°Р»СЊРЅРѕ-РґРµС‚РµРєС‚РёРІРЅС‹Рј СЃСЋР¶РµС‚РѕРј.</p>

</div>

<div class="card-footer">

<small class="text-muted">&#9733; &#9733; &#9733; &#9734; &#9734;</small>

</div>

</div>

</div>

<div class="col-lg-4 col-md-6 mb-4">

<div class="card h-100">

<a href="#"><img class="card-img-top" src="http://league.org.ru/images/stories/smeshariki\_veselye\_starty-5-s.jpg" alt=""></a>

<div class="card-body">

<h4 class="card-title">

<a href="#">РЎРјРµС€Р°СЂРёРєРё</a>

</h4>

<h5>2.500 в‚Ѕ</h5>

<p class="card-text">РќР°СЃС‚РѕР»СЊРЅР°СЏ РёРіСЂР° РґР»СЏ РІСЃРµР№ СЃРµРјСЊРё! Р›СЋР±РёРјС‹Рµ РїРµСЂСЃРѕРЅР°Р¶Рё РїСЂСЏРјРѕ Сѓ С‚РµР±СЏ РґРѕРјР°. Р РµРєРѕРјРµРЅРґСѓРµРј РёРіСЂР°С‚СЊ РЅРµ РїРѕР·РґРЅРµРµ 20:00.</p>

</div>

<div class="card-footer">

<small class="text-muted">&#9733; &#9733; &#9734; &#9734; &#9734;</small>

</div>

</div>

</div>

<div class="col-lg-4 col-md-6 mb-4">

<div class="card h-100">

<a href="#"><img class="card-img-top" src="https://nintendobserver.com/wp-content/uploads/2018/09/CI\_NSwitch\_SuperSmashBrosUltimate\_HardwareBundle\_image950w-700x400.png" alt=""></a>

<div class="card-body">

<h4 class="card-title">

<a href="#">Nintendo Switch</a>

</h4>

<h5>22.990 в‚Ѕ</h5>

<p class="card-text">Р“РёР±СЂРёРґРЅР°СЏ РёРіСЂРѕРІР°СЏ РєРѕРЅСЃРѕР»СЊ(РїСЂРёСЃС‚Р°РІРєР°) РґР»СЏ РІСЃРµС… РІРѕР·СЂР°СЃС‚РѕРІ Рё РёРіСЂРѕРІС‹С… РїСЂРµРґРїРѕС‡С‚РµРЅРёР№. </p>

</div>

<div class="card-footer">

<small class="text-muted">&#9733; &#9733; &#9733; &#9733; &#9734;</small>

</div>

</div>

</div>

</div>

<!-- /.row -->

</div>

<!-- /.col-lg-9 -->

</div>

<!-- /.row -->

</div>

<!-- /.container -->

<!-- Footer -->

<footer class="py-5 bg-dark">

<div class="container">

<p class="m-0 text-center text-white">Copyright &copy; РњР°РіР°Р·РёРЅ РёРіСЂСѓС€РµРє 2020</p>

</div>

<!-- /.container -->

</footer>

<!-- Bootstrap core JavaScript -->

<script src="vendor/jquery/jquery.min.js"></script>

<script src="vendor/bootstrap/js/bootstrap.bundle.min.js"></script>

</body>

</html>