

**Протокол имплементации**

**Интернетный ресурс базы данных   
речных и озерных рыб водоемов РФ**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Проектная группа | -- | | | |
| Заказчик | Самостоятельный проект | | | |
| Проект | Концепция интернетного ресурса свободного доступа к ограниченной базе данных речных и озерных рыб водоемов РФ | | | |
| Руководитель проекта | Артур Мангус | | | |
| Участники проекта |  | | | |
| Документ | Протокол имплементации | | Количество страниц |  |
| Автор документа | Артур Мангус | | | |
| Создан | 18.12.2022 | | | |
| Последнее изменение | 18.12.2022 | | | |
| Статус обработки | X | В обработке  Представлен разработчикам и закзчику  Одобрен  Закрыт | | |

**История документа**

|  |  |  |  |
| --- | --- | --- | --- |
| **Версия** | **Дата** | **Автор изменения** | **Описание / замечание** |
| 0.1 | 18.12.2022 | Артур Мангус |  |

Оглавление

[1 Общее описание 3](#_Toc122292325)

[2 Макет с использованием CSS 3](#_Toc122292326)

[3 Cookie и JavaScript 5](#_Toc122292327)

[4 Реализация меню DropDown в CSS 6](#_Toc122292328)

[5 Реализация базы данных MySQL 7](#_Toc122292329)

[6 Работа с базой данных из PHP 7](#_Toc122292330)

[7 Подключение проекта к GitHub 7](#_Toc122292331)

[Приложения и ресурсы 9](#_Toc122292332)

Настоящий протокол направлен на анализ реализации приложения, с целью улучшения кода в последующих проектах.

# Общее описание

При реализации файла index.php и UpdatePage.php было принято очень верное решение использовать html\_entity\_decode всего один раз, пропуская через него весь код страницы.   
Текст index.php получился очень компактным и представляет из себя только три блока:

* Включение скриптов CSS и JavaScript
* Описание заголовка и мета данных
* HTML код страницы

Текст UpdatePage.php также очень компактный и включает в себя два блока:

* Обработка событий массивов $\_GET и присвоение значений массива $\_COOKIE
* HTML код страницы

Так как технология PHP предполагает создание страниц на стороне сервера, который в свою очередь возвращает полный код в формате HTML, для каждой отдельно сгенерированной страницы, то нет необходимости следовать каким-то особым требованиям в формировании кода страницы.

Для облегчения понимания кода был создан класс PageSubject, сводящий фрагменты макета.

Все классы, возвращающие код макета HTML, реализованы как статические. Использование динамических классов в данном случае не оправдано, так как размер используемых ресурсов памяти на стороне сервера для данной реализации ничтожен. В свою очередь вызов статических методов делает код более компактным и читаемым.

Реализация проводится в IDE PhpStorm. Программа автоматически переформатирует табуляцию, позволяющую лучше проанализировать глубину вложения. Для избегания автоматического форматирования кода необходимо встраивать после объявления класса строку с комментарием // @formatter:off

При тестировании страницы в браузере необходимо каждый раз вызывать обновление клавиатурным сокращением Ctrl+F5.

# Макет с использованием CSS

Реализация скриптов CSS основана на рекомендациях книги Макфарланда «Новая большая книга CSS» издательства O’Railly.

Основной макет построен на блоках div. Блоки выстраиваются автоматически по вертикали образуя макет QLinearVBox. Для организации блоков по горизонтали используется свойство display со значением flex. Необходимо проверить наследуется ли это свойство вложенными блоками. Если наследуется, то его необходимо переопределять со значением initial.

Для установки значений по умолчанию использован файл Main.css, в котором определены базовые установки для всех элементов макета. Так, например, для предотвращения копирования текста использован следующий код:

|  |
| --- |
| -webkit-touch-callout: none; */\* iOS Safari \*/* -webkit-user-select: none; */\* Safari \*/* -khtml-user-select: none; */\* Konqueror HTML \*/* -moz-user-select: none; */\* Old versions of Firefox \*/* -ms-user-select: none; */\* Internet Explorer/Edge \*/* user-select: none; */\*generic\*/* |

Для предотвращения нарушений цветового представления из-за специфических настроек браузера использована установка белого цвета фона по умолчанию:

|  |
| --- |
| body {  line-height: 1.2;  background-color: white; } |

Для выравнивания блоков по вертикали, относительно родительского элемента макета, использован следующий код:

|  |
| --- |
| position: relative; margin: auto; top: 0; right: 0; bottom: 0; left: 0; |

Для использования шрифтов, не включенных в список стандартных, используется файл Font.css с определением шрифтов. Файл импортируется в другие файлы css вызовом @import "Fonts.css";. Строки инициализации font-family: "Roboto Condensed"; font-size: 100%; включены в файл Main.css для всех элементов макета по умолчанию. Таким образом необходимо только менять стиль и размер шрифта. Для определения пользовательских шрифтов в файле Fonts.css использован следующий код:

|  |
| --- |
| @font-face {  font-family: 'Roboto Condensed';  src: url('../fonts/RobotoCondensed-Regular.eot');  src: local('Roboto Condensed'), local('RobotoCondensed-Regular'),  url('../fonts/RobotoCondensed-Regular.eot?#iefix') format('embedded-opentype'),  url('../fonts/RobotoCondensed-Regular.woff') format('woff'),  url('../fonts/RobotoCondensed-Regular.ttf') format('truetype');  font-weight: normal;  font-style: normal; } |

Для работы с изображениями использован рекомендуемый код:

|  |
| --- |
| background: url("../img/headerkontakt.jpg") no-repeat center center; |

При разработке кнопок с использованием файлов svg необходимо учитывать, что размер рабочей области документа изображения должен соответствовать конечному необходимому размеру, используемому в приложении. Все графические элементы изображения должны лежать в рабочей области документа. Для использования разных состояний иконки необходимо каждое из этих состояний сохранить в файле svg в определенных координатах. Состояние может быть определено классом class=“‘.$state.‘“ или подклассом .class:hover. Смена состояния осуществляется смещением координат:

|  |
| --- |
| background-position: left 0px top -38px; |

При работе с селекторами была отмечена значительная разница использования слитного и раздельного написания селекторов классов. Так слитное написание относится к самому элементу, при необходимости более точной идентификации (например обращение к общему классу подклассу и идентификатору .class.subclass#id), а раздельное обращается к дочерним элементам (например класс родителя класс дочернего элемента .parentClass .childClass). Свойство position со значением relativ задает координаты относительно родительского компонента. Значение absolut задает координаты относительно окна браузера.

При использовании рамок вокруг элементов макета необходимо не забывать добавлять свойство border-style: solid;.

# Cookie и JavaScript

Для запоминания выбора пользователя, использован массив глобальных переменных $\_COOCKIE. Все кнопки обрабатывают событие onClick=““ в файле NavicationControl.js. Запрос формируется из значений, полученных из глобальных переменных следующим образом (все строки кода релевантны):

|  |
| --- |
| function *setFamilyId*(control) {  if (document.cookie.length > 0) {   var contentID = *getCookieById*("contentID") + '&';  var pageID = *getCookieById*("pageID") + '&';  var viewID = *getCookieById*("viewID") + '&';  var familyID = 'familyID=' + control.getAttribute("name");  document.cookie = familyID;   var requestData = contentID + pageID + viewID + familyID;   var request = new XMLHttpRequest();  request.open('GET', './php/UpdatePage.php?' + requestData, false);  request.send(null);   if (request.status == 200) {  document.getElementById("mainContent").innerHTML = request.responseText;  }  } } |

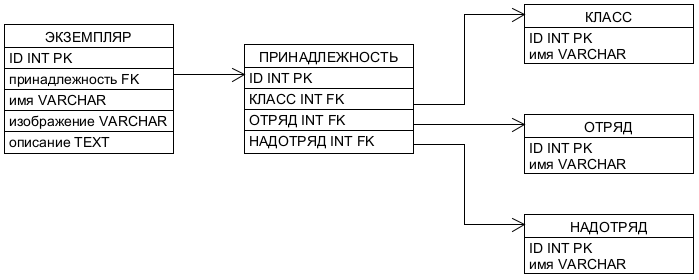
Получение глобальных переменных и их запись в JavaScript работает надежнее чем в php.

# Реализация меню DropDown в CSS

Пример простой реализации меню DropDown:

|  |
| --- |
| <style> /\* Style The Dropdown Button \*/ .dropbtn {   background-color: #4CAF50;   color: white;   padding: 16px;   font-size: 16px;   border: none;   cursor: pointer; }  /\* The container <div> - needed to position the dropdown content \*/ .dropdown {   position: relative;   display: inline-block; }  /\* Dropdown Content (Hidden by Default) \*/ .dropdown-content {   display: none;   position: absolute;   background-color: #f9f9f9;   min-width: 160px;   box-shadow: 0px 8px 16px 0px rgba(0,0,0,0.2);   z-index: 1; }  /\* Links inside the dropdown \*/ .dropdown-content a {   color: black;   padding: 12px 16px;   text-decoration: none;   display: block; }  /\* Change color of dropdown links on hover \*/ .dropdown-content a:hover {background-color: #f1f1f1}  /\* Show the dropdown menu on hover \*/ .dropdown:hover .dropdown-content {   display: block; }  /\* Change the background color of the dropdown button when the dropdown content is shown \*/ .dropdown:hover .dropbtn {   background-color: #3e8e41; } </style>  <div class="dropdown">   <button class="dropbtn">Dropdown</button>   <div class="dropdown-content">     <a href="#">Link 1</a>     <a href="#">Link 2</a>     <a href="#">Link 3</a>   </div> </div> |

# Реализация базы данных MySQL



Наиболее важным условием при создании таблиц базы данных, является устранение любого рода редундантности. Исходя из этого условия была создана таблица affiliation\_fish в которой представлены все возможные комбинации классов, отрядов и групп, для отдельно взятого случая.

|  |
| --- |
| CREATE DATABASE fishatlas;  USE fishatlas;  CREATE TABLE class\_fish (  id INT PRIMARY KEY,  name VARCHAR(1024) NOT NULL,  discription VARCHAR(2048)  );  INSERT INTO class\_fish ( id, name, discription) VALUES (1, 'Костные рыбы', '');  INSERT INTO class\_fish ( id, name, discription) VALUES (2, 'Хрящевые рыбы', '');  CREATE TABLE subclass\_fish (  id INT PRIMARY KEY,  name VARCHAR(1024) NOT NULL,  discription VARCHAR(2048)  );  INSERT INTO subclass\_fish ( id, name, discription) VALUES (1, 'Костно-хрящевые', '');  INSERT INTO subclass\_fish ( id, name, discription) VALUES (2, 'Двоякодышащие', '');  INSERT INTO subclass\_fish ( id, name, discription) VALUES (3, 'Кистеперые', '');  INSERT INTO subclass\_fish ( id, name, discription) VALUES (4, 'Костистые рыбы', '');  INSERT INTO subclass\_fish ( id, name, discription) VALUES (5, 'Акулы', '');  INSERT INTO subclass\_fish ( id, name, discription) VALUES (6, 'Скаты', '');  INSERT INTO subclass\_fish ( id, name, discription) VALUES (7, 'Химеры', '');  CREATE TABLE groups\_fish (  id INT PRIMARY KEY,  name VARCHAR(1024) NOT NULL,  discription VARCHAR(2048)  );  INSERT INTO groups\_fish ( id, name, discription) VALUES (0, '', '');  INSERT INTO groups\_fish ( id, name, discription) VALUES (1, 'Осетровые', '');  INSERT INTO groups\_fish ( id, name, discription) VALUES (2, 'Сельдеобразные', '');  INSERT INTO groups\_fish ( id, name, discription) VALUES (3, 'Лососеобразные', '');  INSERT INTO groups\_fish ( id, name, discription) VALUES (4, 'Карпообразные', '');  INSERT INTO groups\_fish ( id, name, discription) VALUES (5, 'Сомообразные', '');  INSERT INTO groups\_fish ( id, name, discription) VALUES (6, 'Трескообразные', '');  INSERT INTO groups\_fish ( id, name, discription) VALUES (7, 'Камбалообразные', '');  CREATE TABLE affiliation\_fish (  id INT PRIMARY KEY,  class INT,  subclass INT,  groups INT,  FOREIGN KEY (class) REFERENCES class\_fish (id),  FOREIGN KEY (subclass) REFERENCES subclass\_fish (id),  FOREIGN KEY (groups) REFERENCES groups\_fish (id)  );  INSERT INTO affiliation\_fish ( id, class, subclass, groups ) VALUES (1, 1, 1, 1);  INSERT INTO affiliation\_fish ( id, class, subclass, groups ) VALUES (2, 1, 2, 0);  INSERT INTO affiliation\_fish ( id, class, subclass, groups ) VALUES (3, 1, 3, 0);  INSERT INTO affiliation\_fish ( id, class, subclass, groups ) VALUES (4, 1, 4, 2);  INSERT INTO affiliation\_fish ( id, class, subclass, groups ) VALUES (5, 1, 4, 3);  INSERT INTO affiliation\_fish ( id, class, subclass, groups ) VALUES (6, 1, 4, 4);  INSERT INTO affiliation\_fish ( id, class, subclass, groups ) VALUES (7, 1, 4, 5);  INSERT INTO affiliation\_fish ( id, class, subclass, groups ) VALUES (8, 1, 4, 6);  INSERT INTO affiliation\_fish ( id, class, subclass, groups ) VALUES (9, 1, 4, 7);  INSERT INTO affiliation\_fish ( id, class, subclass, groups ) VALUES (10, 2, 5, 0);  INSERT INTO affiliation\_fish ( id, class, subclass, groups ) VALUES (11, 2, 6, 0);  INSERT INTO affiliation\_fish ( id, class, subclass, groups ) VALUES (12, 2, 7, 0);  CREATE TABLE ident (  id INT PRIMARY KEY AUTO\_INCREMENT,  affiliation INT,  name VARCHAR(1024) NOT NULL,  image VARCHAR(1024),  discription VARCHAR(2048),  FOREIGN KEY (affiliation) REFERENCES affiliation\_fish (id)  );  INSERT INTO ident (id, affiliation, name, image, discription) VALUES (  1, 6, 'Карась золотой (Carassius carassius)', './img/fishes/karas.jpg',  'Тело высокое умеренно сжатое с боков, с толстой спиной. Усиков нет. Килей на теле нет. У представителей рода зубы однорядные. У карася золотого 17-25. В полной боковой линии 32 - 35 крупных чешуй. Жаберных тычинок 23 - 25. Чешуйки на ощупь гладкие. Бока тела окрашены в медно-красный (особенно в темных торфяных водоемах), золотистые тона. Брюшко чуть-чуть светлее, спина заметна темнее. Плавники желтовато-красные с более темными концами. Длина тела до 40 см. Масса до 5 кг. В малокормных (олигатрофных) водоемах, например в лесных или тундровых озерах, есть тугорослые (карликовые) формы, длина тела которых не превышает 15 см, а масса 80 - 100 грамм. Наиболее обычная рыба страны, обитает в стоящих и медленнотекущих пресных водоемах.'  ); |

# Работа с базой данных из PHP

|  |
| --- |
| class DataDispatcher {  *// @formatter:off* static private *$dbDispatcher*;  static private *$errorMissage*;   static public function connectDB() : bool  {  $dsn = 'mysql:host=localhost; dbname=fishatlas';  $user = 'root';  $password = '';   try {  self::*$dbDispatcher* = new PDO($dsn, $user,$password);  self::*$dbDispatcher*->setAttribute(PDO::*ATTR\_ERRMODE*, PDO::*ERRMODE\_EXCEPTION*);  }  catch (PDOException $e) {  self::*$errorMissage* = "Connection failed: " . $e->getMessage();  return false;  }   return true;  }   static public function disconnectDB()  {  self::*$dbDispatcher* = null;  } } |

Команда UPDATE изменяет данные, уже находящиеся в таблице.

Чтобы изменить только некоторые строки, в запрос SQL с командой UPDATE следует ввести

предложение WHERE с логическим выражением, описывающим изменяемые строки. Таким образом,

изменения по такому запросу произойдут только в тех строках таблицы, которые совпадают с

условием, заданным в предложении WHERE.

|  |
| --- |
| UPDATE имя\_таблицы SET столбец\_N= значение\_N [,...] [WHERE id=‘X‘] |
| $db->exec("UPDATE dishes SET is\_spicy = 1 WHERE dish\_name = 'Eggplant with Chili Sauce'"); |

Команда DELETE удаляет строки из таблицы.

|  |
| --- |
| DELETE FROM имя\_таблицы [WHERE логическое\_выражение] |

## Безопасный ввод данных из формы

|  |
| --- |
| $stmt = $db->prepare('INSERT INTO dishes (dish\_name, price, is\_spicy) VALUES (?,?,?)');  $stmt->execute(array($\_POST['new\_dish\_name'], $\_POST['new\_price'], $\_POST('is\_spicy'])); |

## Извлечение информации из базы данных

Команда SELECT извлекает информацию из таблицы базы данных.

|  |
| --- |
| SELECT dish\_name, price FROM dishes WHERE price > 5.00  SELECT \* FROM dishes |
| $q = $db->query( **'SELECT dish\_name, price FROM dishes'** );  while ($row = $q->fetch()) {  print "$row[dish\_name], $row[price] \n";  } |

Следующий запрос возвращает сразу все запрошенные данные в одном массиве:

|  |
| --- |
| $data = $db->query( <SQL\_REQUEST> )->fetch();  echo $data[i]; |

Команда ORDER BY

Команда LIMIT

Команда DESC

Команда LIKE

# JQuery

Использование библиотеки JQuery позволяет делать поисковые запросы к базе данных с методом POST с помощью JavaScript, что в свою очередь делает код более защищенным от вмешательства взломщиков сайта.

## Установка JQuery на XAMPP

С сайта <https://jquery.com/download/> скачать сжатую версию в каталог проекта /js размешенного на XAMPP. /js/jquery.min.js   
Для обращения к библиотеке использовать следующий скрипт

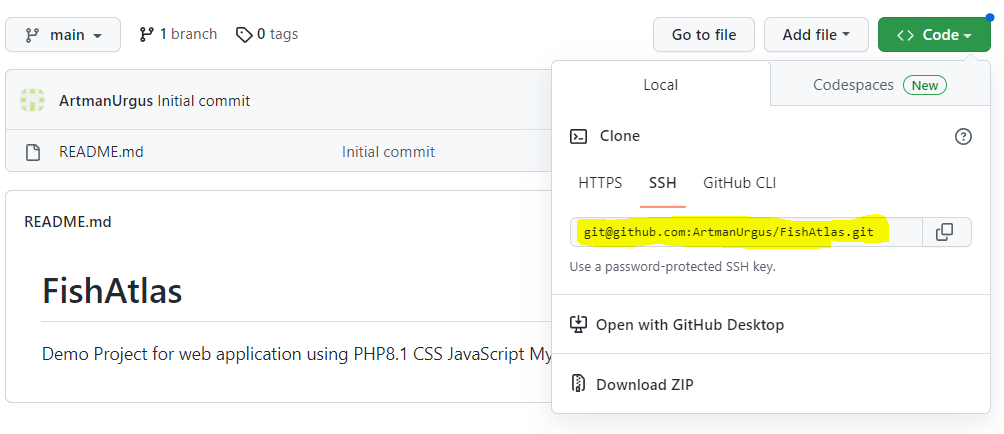
|  |
| --- |
| echo '<script src="./js/jquery.min.js"></script>'; |

Или ничего не скачивая вставить скрипт:

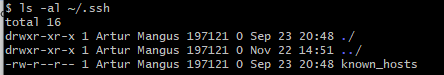
|  |
| --- |
| echo '<script src="http://code.jquery.com/jquery-3.6.3.min.js"></script>'; |

# Подключение проекта к GitHub

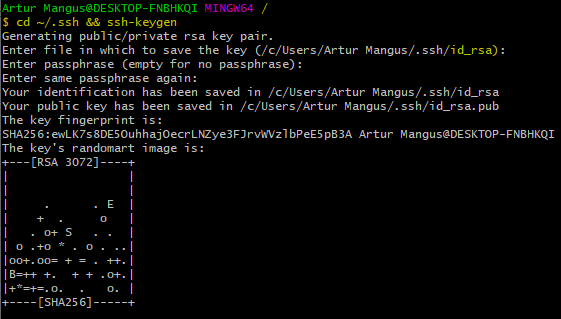
Для подключения к созданному репозиторию на ресурсе GitHub необходимо получить правильный путь соединения SSH, который обеспечивает наиболее качественное соединение с клонированным репозиторием. Для этого необходимо открыть вкладку Code и скопировать путь из вкладки SSH



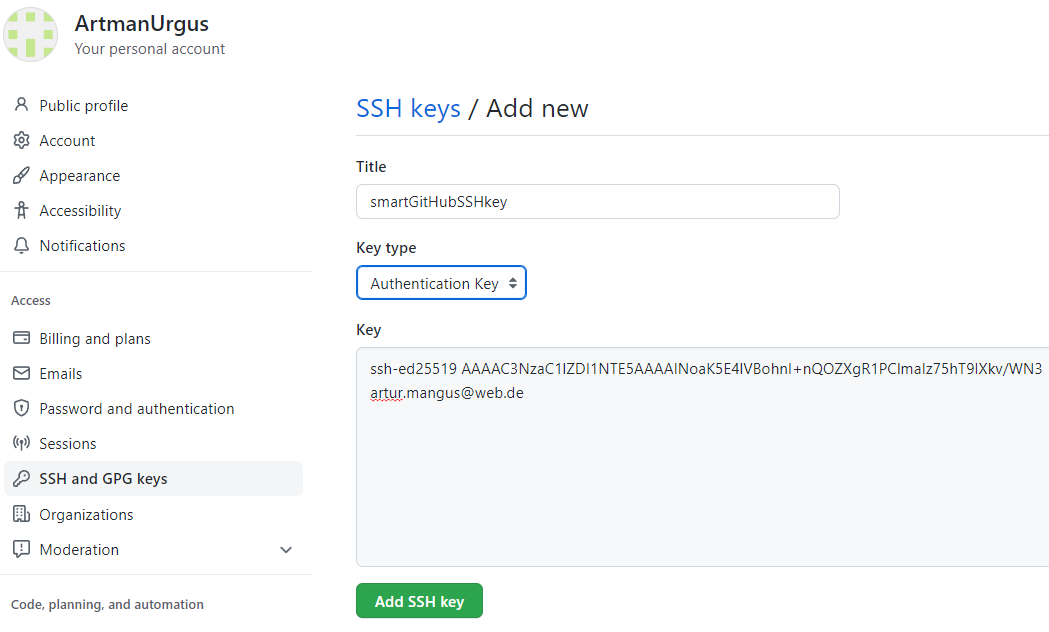
Для использования соединения необходимо создать ключ public SSH. Ключ создается из консоли git bash на локальном компьютере. Следующая команда проверяет наличие файлов ключей хранящихся на компьютере в пути C:\usres\<NAME>\.ssh



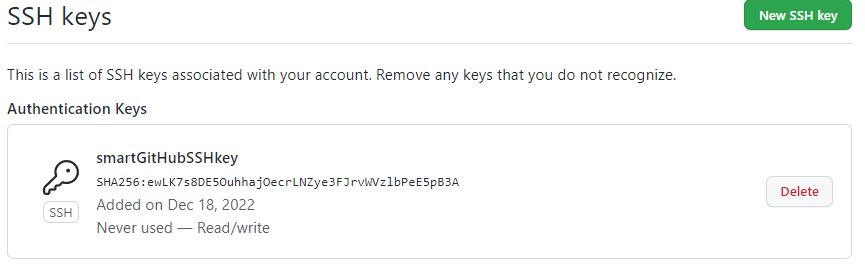
В данном примере файлы ключей отсутствуют. Для их создания необходимо выполнить команду:

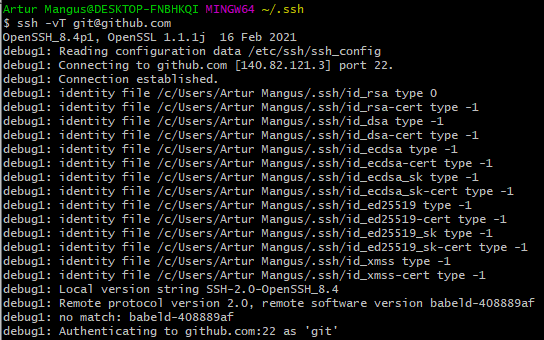


Команда генерирует файл id\_rsa. Файл необходимо открыть текстовым редактором и скопировать из него весь текст ключа. Текст ключа необходимо добавить в Settings->SSH and CGP keys.

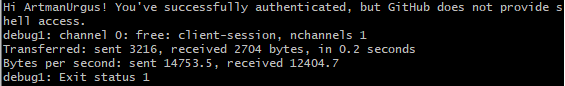


Результат добавления:



С помощью следующей команды необходимо проверить связь с сервером GitHub.   


Следующее сообщение подтверждает удачное соединение с сервером



Далее необходимо выбрать команду Clone в SmartGit. В поле ввода удаленной репозитории добавить SSH адрес репозитории и кликнуть продолжить. В открывшемся окне необходимо выбрать файл.

Далее необходимо установить GitHub Desktop для Windows и открыть в нем локальный проект.   
Локальную репозиторию можно автоматически разместить на GitHub.

# Приложения и ресурсы