

Южно-Уральский
государственный
университет

Национальный
исследовательский
университет

КРОК

Основы РНР. Взаимодействие с СУБД

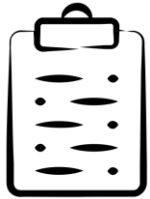
КРОК

Челябинск, ул. Карла Маркса, д. 38

Смирнов Анатолий
Технический менеджер

Антонов Сергей
Тимлид группы разработчиков,
старший инженер-разработчик

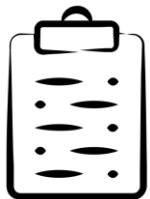
Кузнецов Сергей
Инженер-разработчик



- ORM (Object-Relational Mapping) - технология программирования, которая связывает СУБД с концепциями ООП, создавая «виртуальную объектную базу данных». Можно писать запросы к БД через классы и методы.

- Абстрактный пример:

```
user = Users
      .where('id', 3)
      .get('surname')  →  select surname
                        from users
                        where id = 3
```



Плюсы ORM:

- Не зависит от СУБД.
- Легче переносить и сопровождать.

Минусы ORM:

- Не всегда прозрачный результат работы.
- Может наблюдаться снижение скорости работы с СУБД.
- Запросы не всегда могут быть построены оптимальным образом.
- На каждом ЯП, фреймворке и библиотеке своя реализация.



Для работы с СУБД PostgreSQL в PHP используется библиотека pdo_pgsql:

~/PHP/php.ini

```
939 ;extension=oci8_12c ; Use with Oracle Database 12c Instant Client
940 ;extension=oci8_19 ; Use with Oracle Database 19 Instant Client
941 ;extension=odbc
942 ;extension=openssl
943 ;extension=pdo_firebird
944 ;extension=pdo_mysql
945 ;extension=pdo_oci
946 ;extension=pdo_odbc
947 extension=pdo_pgsql
948 ;extension=pdo_sqlite
949 ;extension=pgsql
950 ;extension=shmop
```



- Отвечает за работу с СУБД в PHP в основном отвечает класс PDO.

```
new PDO(<Строка подключения>, <пользователь>, <пароль>, <опции>);
```



Для PostgreSQL строка подключения выглядит следующим образом:

- `pgsql:host=<адрес/IP носта>;port=<порт>;dbname=<наименование БД>`

Например:

- `pgsql:host=localhost;port=5432;dbname=postgres`



Для пользователя postregs с паролем 1 подключение будет выглядеть следующим образом:

```
$dsn = "pgsql:host=localhost;port=5432;dbname=postgres";
```

```
$pdo = new PDO($dsn, 'postgres', '1');
```

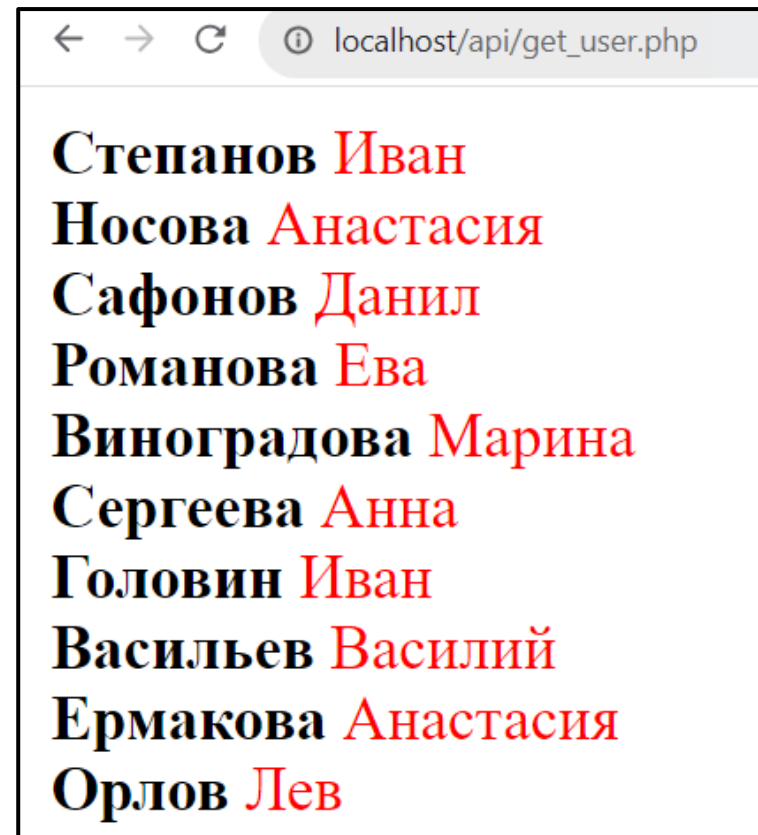
Выполним запрос в СУБД при помощи метода `query()` и выведем данные при помощи `fetch()`:

```
<?php
    $dsn = "pgsql:host=localhost;port=5432;dbname=postgres;";
    $pdo = new PDO($dsn, 'postgres', '1');

    $sql = "select surname, firstname from users";
    $stmt = $pdo->query($sql);

    while ($row = $stmt->fetch()) {
        $surname = $row['surname'];
        $firstname = $row['firstname'];

        echo "<b>$surname</b>
              <span style=\"color:red\">$firstname</span><br/>";
    }
?>
```



Выполним запрос в СУБД и выведем данные при помощи `fetchAll()`:

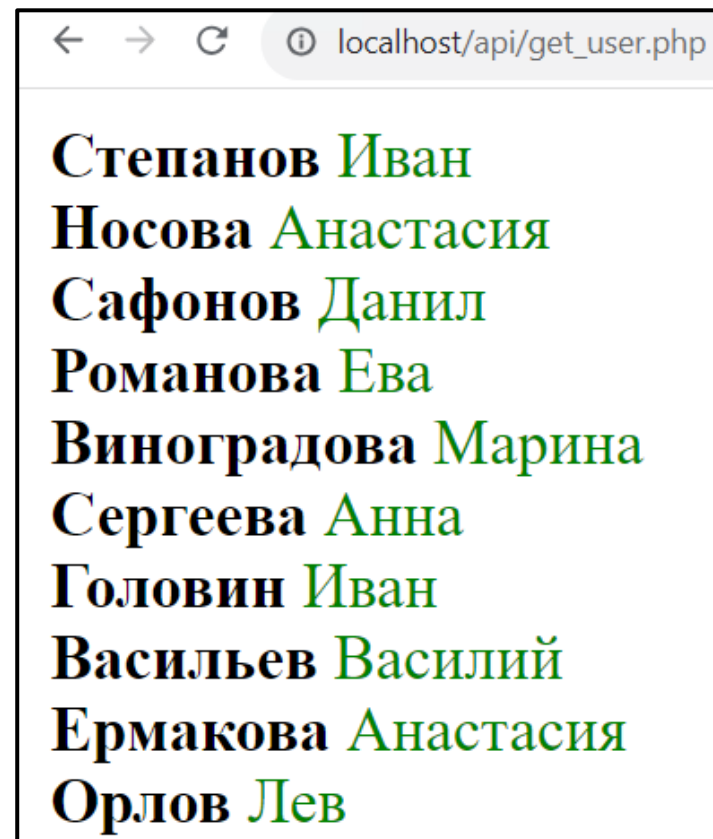
```
<?php
$dsn = "pgsql:host=localhost;port=5432;dbname=postgres;";
$pdo = new PDO($dsn, 'postgres', '1');

$sql = "select surname, firstname from users";
$stmt = $pdo->query($sql);

$results = $stmt->fetchAll();

foreach ($results as $row) {
    $surname = $row['surname'];
    $firstname = $row['firstname'];

    echo "<b>$surname</b>
        <span style=\"color:green\">$firstname</span><br/>";
}
?>
```



Запросы с параметрами. SQL-инъекция

Будем отдавать значение из БД по GET-запросу через параметры URL:

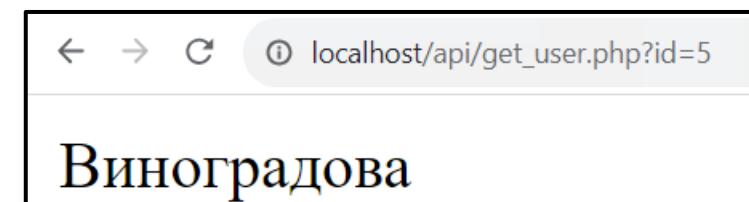
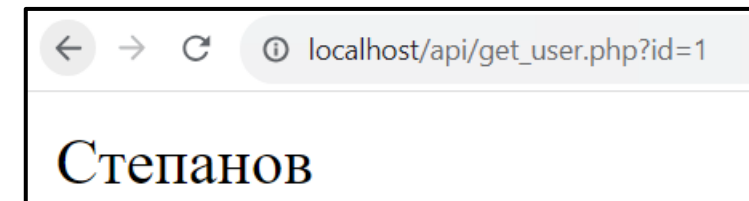
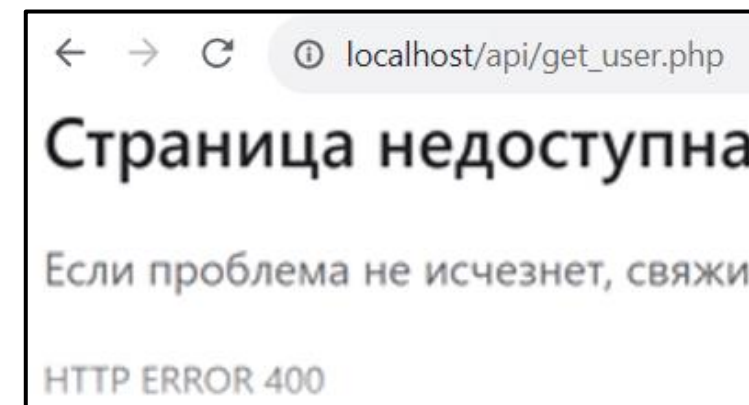
```
<?php
    if (!array_key_exists('id', $_GET)) {
        http_response_code(400);
        die();
    }

    $dsn = "pgsql:host=localhost;port=5432;dbname=postgres;";
    $pdo = new PDO($dsn, 'postgres', '1');

    $sql = "select surname from users where id = {$_GET['id']}";

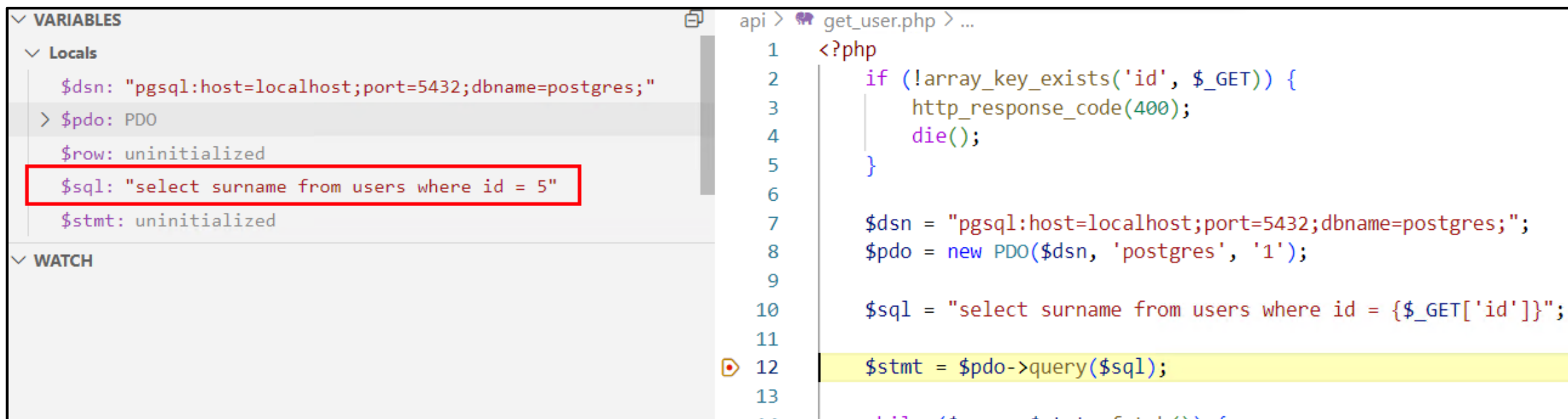
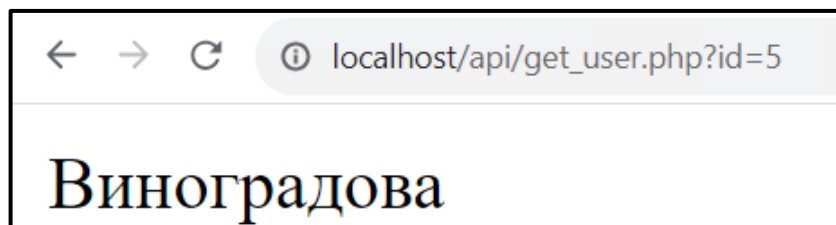
    $stmt = $pdo->query($sql);

    while ($row = $stmt->fetch()) {
        echo $row['surname']."<br/>";
    }
?>
```



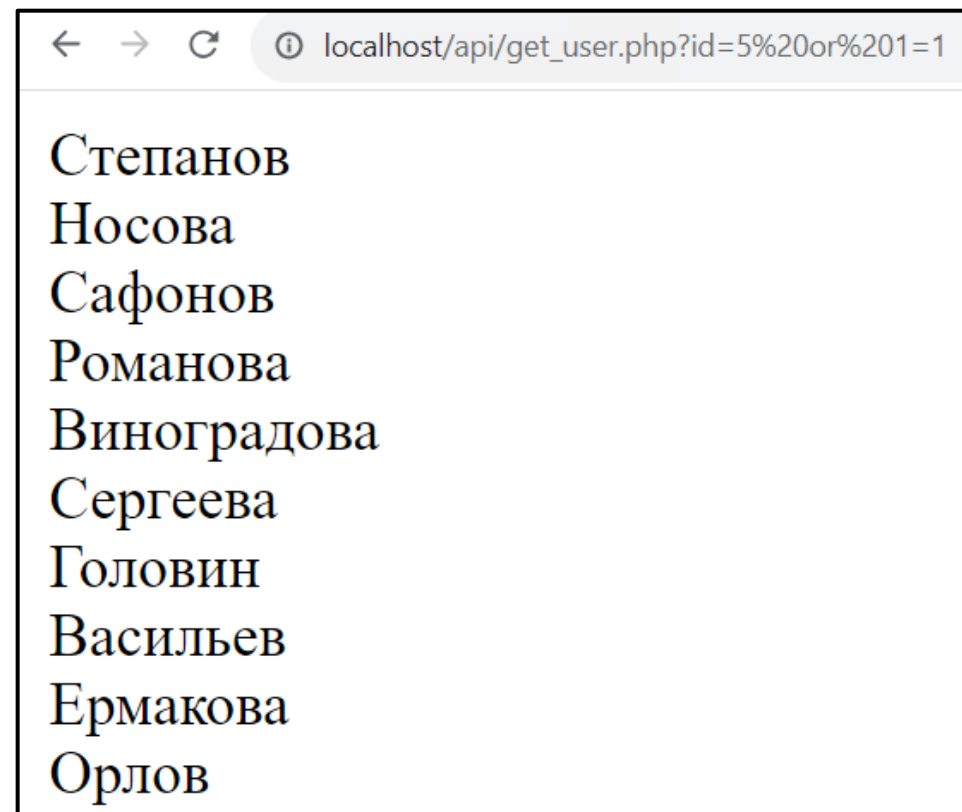
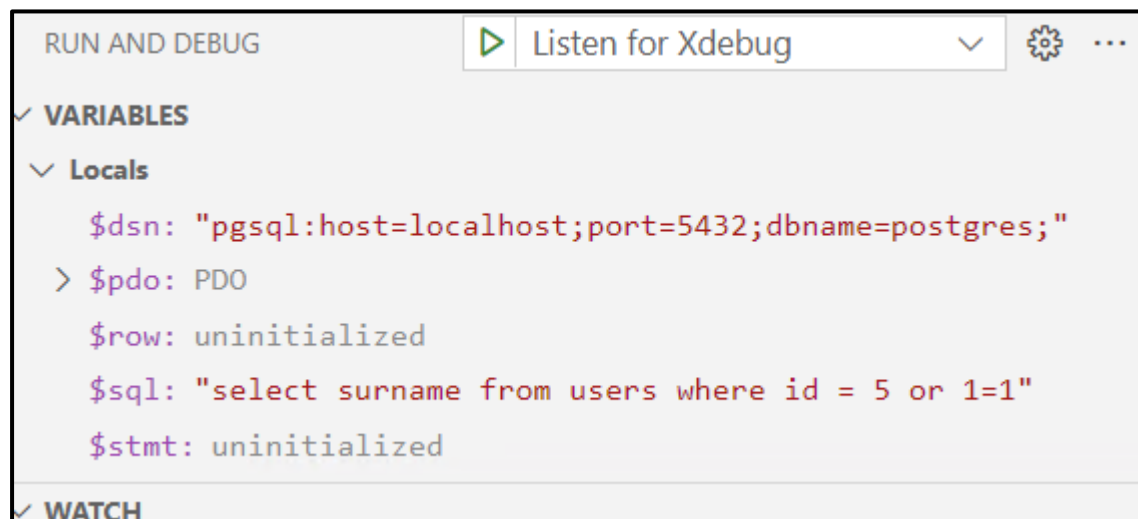
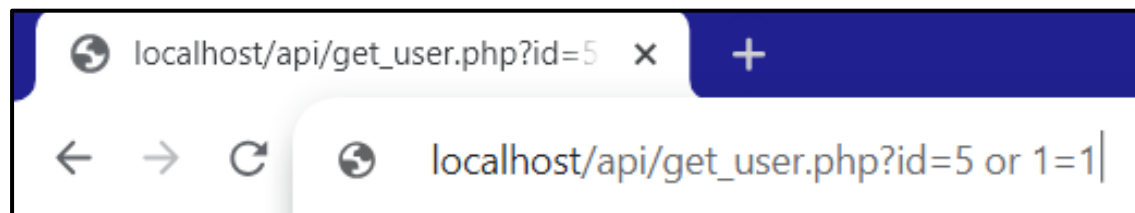


Уязвимое место при конкатенации строки и построении SQL-запроса:



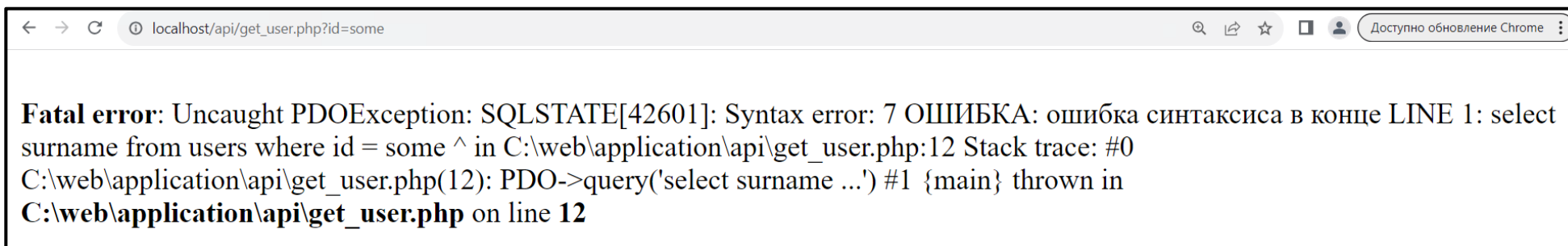


SQL-инъекция – это уязвимость веб-безопасности, которая позволяет злоумышленнику вмешиваться в запросы.





Не нужно выводить пользователю сообщения с ошибкой от СУБД, чтобы злоумышленник не узнал структуру запроса:

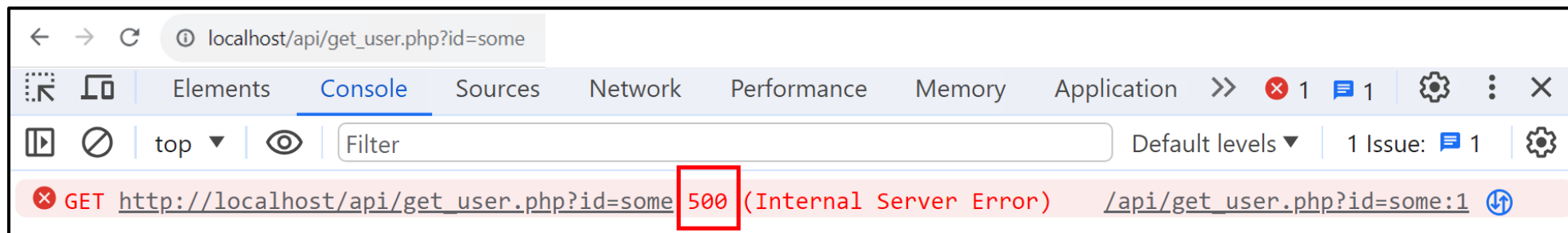


```
$stmt = $pdo->query($sql);
```

Обработка исключений



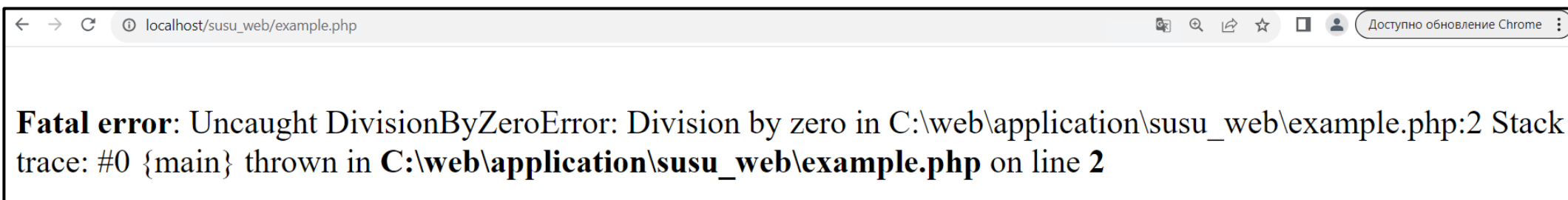
```
try {  
    $stmt = $pdo->query($sql);  
} catch (Exception $e) {  
    http_response_code(500);  
    die();  
}
```





Отлавливать ошибки можно заключив блок кода в try {} catch(<>){}:

```
<?php
    echo 1 / 0;
?>
```



```
<?php
try {
    $x = 1 / 0;
    echo $x;
} catch (Throwable $e) {
    echo '<h1>Ошибка!</h1>';
    echo '<div style="width: 300px; padding: 10px; background: lightcoral">'.$e.'</div>';
}
?>
```



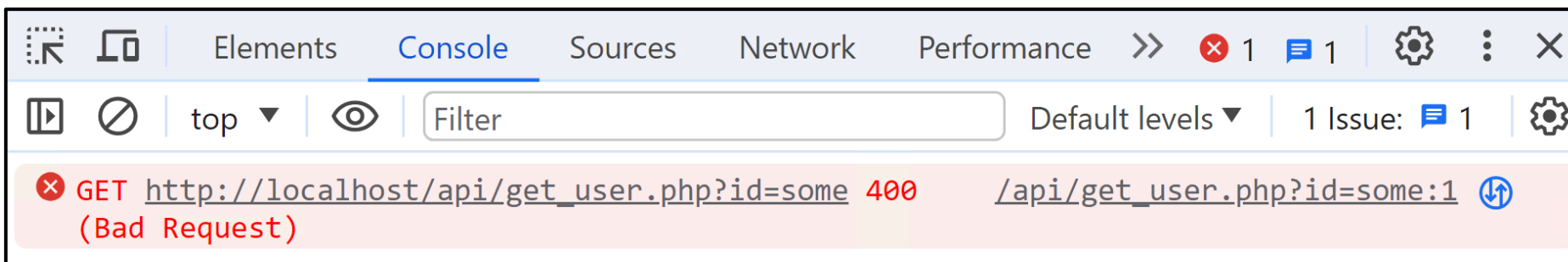


Стараться валидировать входные параметры там, где это ВОЗМОЖНО:

```
if (!array_key_exists('id', $_GET)) {  
    http_response_code(400);  
    die();  
}
```



```
if (!array_key_exists('id', $_GET) || !is_numeric($_GET['id'])) {  
    http_response_code(400);  
    die();  
}
```

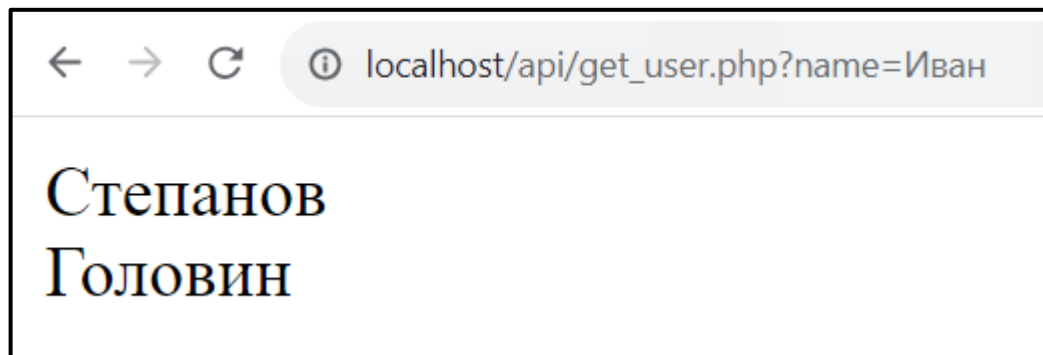


Но не все входные параметры принимают численные значения:

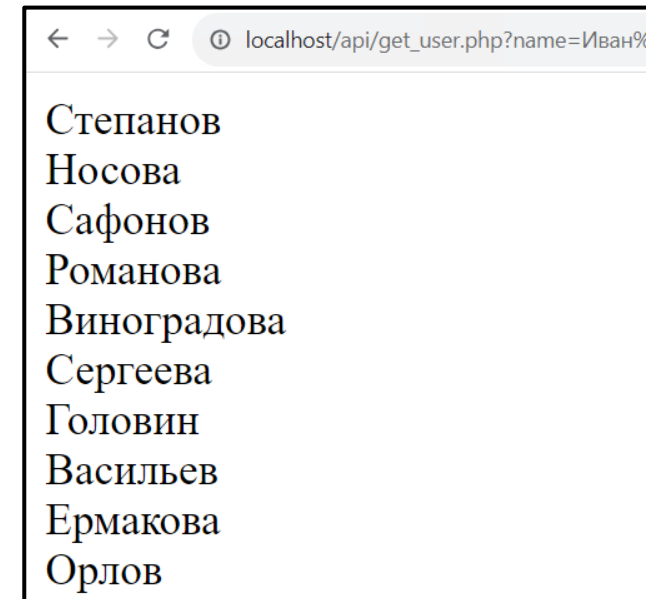
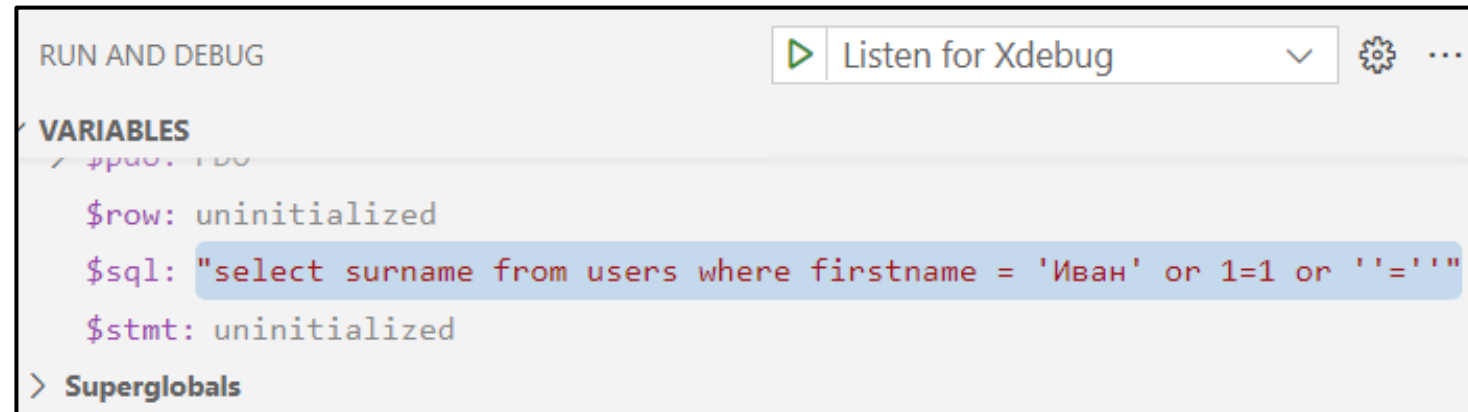
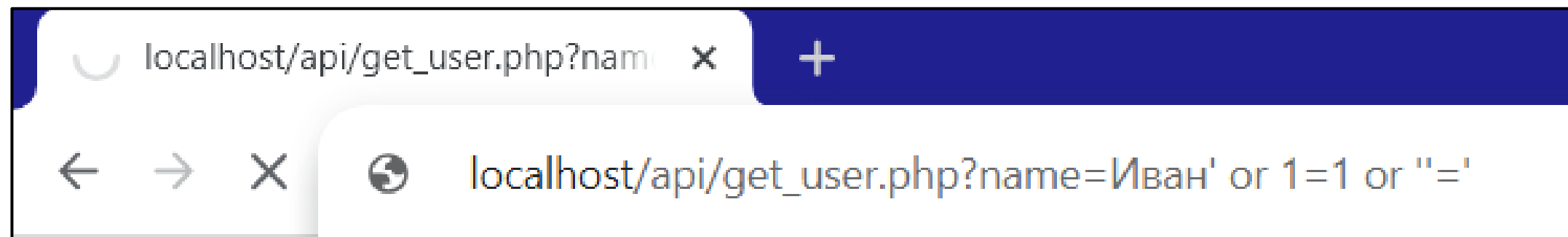
```
<?php
    $dsn = "pgsql:host=localhost;port=5432;dbname=postgres;";
    $pdo = new PDO($dsn, 'postgres', '1');

    $sql = "select surname from users where firstname = '{$_GET['name']}'";
    $stmt = $pdo->query($sql);

    while ($row = $stmt->fetch()) {
        echo $row['surname']."<br/>";
    }
?>
```



```
firstname = '{$_GET['name']}'
```





Использование подготовленного запроса с автоматической экранировкой входных параметров:

```
$sql = "select surname from users where firstname = '{$_GET['name']}'";  
$stmt = $pdo->query($sql);
```



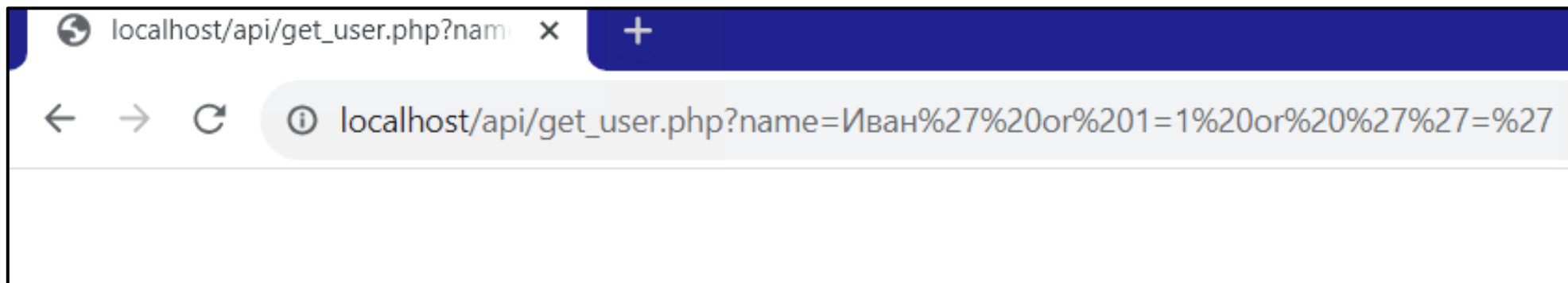
```
$stmt = $pdo->prepare("select surname from users where firstname = :name");  
$stmt->bindParam(':name', $_GET['name']);  
$stmt->execute();
```

The screenshot displays a development environment with two main sections. On the left, the 'VARIABLES' panel is expanded, showing the state of variables. A red rectangle highlights the entry for '\$stmt: PDOStatement', which has a 'queryString' property set to 'select surname from users where firstname = :name'. Below this, 'Superglobals' and 'User defined constants' are also visible. On the right, the code editor shows the source code for 'get_user.php'. The code includes database connection details, PDO instantiation, and the preparation and execution of a SQL query. The line '\$stmt->execute();' is highlighted in yellow, corresponding to the state shown in the variables panel.



Теперь при любом некорректном параметре он будет восприниматься как значение параметра, а не часть SQL-кода.

Значения с `firstname = Иван' or 1=1 or ''='` нет, запрос не вернет данных:





Было:

```
<?php
    if (!array_key_exists('id', $_GET)) {
        http_response_code(400);
        die();
    }

    $dsn = "pgsql:host=localhost;
port=5432;dbname=postgres;";

    $pdo = new PDO($dsn, 'postgres', '1');

    $sql = "select surname from users
        where id = {$_GET['id']}";

    $stmt = $pdo->query($sql);

    while ($row = $stmt->fetch()) {
        echo $row['surname']."<br/>";
    }

?>
```



Стало:

```
<?php
    // Валидация
    if (!array_key_exists('id', $_GET) || !is_numeric($_GET['id'])) {
        http_response_code(400);
        die();
    }

    $dsn = "pgsql:host=localhost;port=5432;dbname=postgres;";
    $pdo = new PDO($dsn, 'postgres', '1');

    try { // Обработка ошибок
        $stmt = $pdo->prepare("select surname from users where id = :id");
        $stmt->bindParam(':id', $_GET['id']); // Экранирование параметров
        $stmt->execute();
    } catch (Exception $e) {
        http_response_code(500);
        die();
    }

    while ($row = $stmt->fetch()) {
        echo $row['surname']."<br/>";
    }

?>
```



Общие рекомендации:

- Отлавливать все ошибки со стороны СУБД и не выводить клиенту лишнего.
- Проводить валидацию входных параметров (тип данных, мин. и макс. длина строки, допустимые значения и т.д.).
- Не вставлять параметры в строку запроса напрямую.
- Убедиться, что ваша библиотека для взаимодействия с СУБД экранирует передаваемые параметры.



Чтобы узнать, сколько строчек было удалено, можно воспользоваться функцией `rowCount()`:

```
<?php
$dsn = "pgsql:host=localhost;port=5432;
       dbname=postgres;";

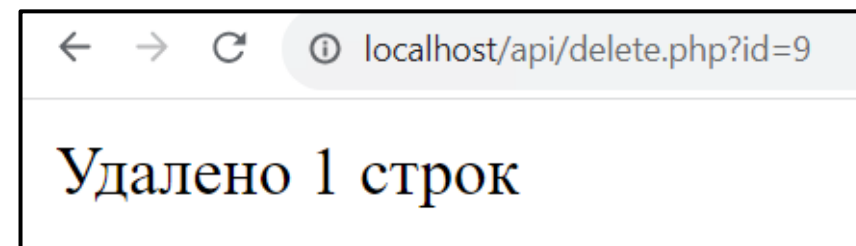
$pdo = new PDO($dsn, 'postgres', '1');

$stmt = $pdo->prepare("delete from users
                      where id = :id");
$stmt->bindParam(':id', $_GET['id']);
$stmt->execute();

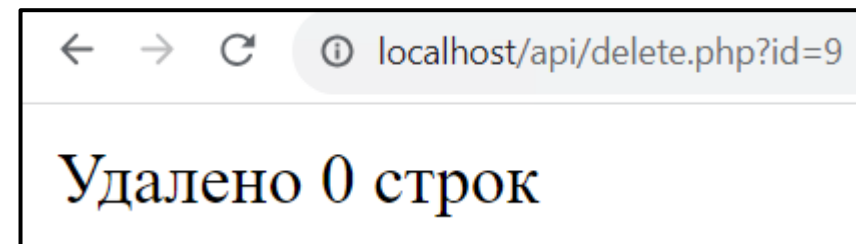
$count = $stmt->rowCount();
echo "Удалено $count строк";

?>
```

Первый раз



Второй раз





- В теге **form** можно задать путь до обработчика на стороне сервера в атрибуте **action**.
- Если в атрибуте **method** указать **"get"**, то данные в PHP будут доступны в массиве **\$_GET**. Если **"post"**, то в массиве **\$_POST**.
- Оправка данных методом **POST** добавляет заголовок **Content-Type: application/x-www-form-urlencoded** или **multipart/form-data**.

```
<meta charset="utf-8"/>
```

```
<form action="/api/create_form.php" method="post">
```

```
Имя: <input name="firstname"><br>
```

```
Фамилия: <input name="surname"><br>
```

```
Отчество: <input name="lastname"><br>
```

```
Кол-во полетов: <input name="flights"><br>
```

```
<input type="submit" value="Добавить">
```

```
</form>
```

← → ↻ ⓘ localhost/index.html

Имя:

Фамилия:

Отчество:

Кол-во полетов:



В ассоциативном массиве параметры доступны под теми именами, которые они имеют в атрибутах **input** в тегах **name**:

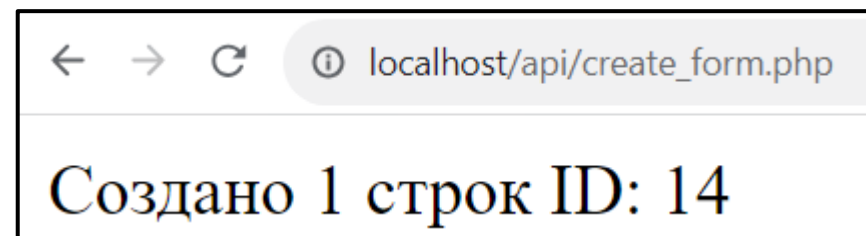
```
<?php
$dsn = "pgsql:host=localhost;port=5432;dbname=postgres;";
$pdo = new PDO($dsn, 'postgres', '1');

$sql = "insert into users (surname, firstname,
                           lastname, flights)
       values (:sn, :fn, :ln, :fl)";

$stmt = $pdo->prepare($sql);
$stmt->bindParam(':sn', $_POST['surname']);
$stmt->bindParam(':fn', $_POST['firstname']);
$stmt->bindParam(':ln', $_POST['lastname']);
$stmt->bindParam(':fl', $_POST['flights']);
$stmt->execute();

echo "Создано ". $stmt->rowCount() . " строк\n";
echo "ID: ". $pdo->lastInsertId();

?>
```

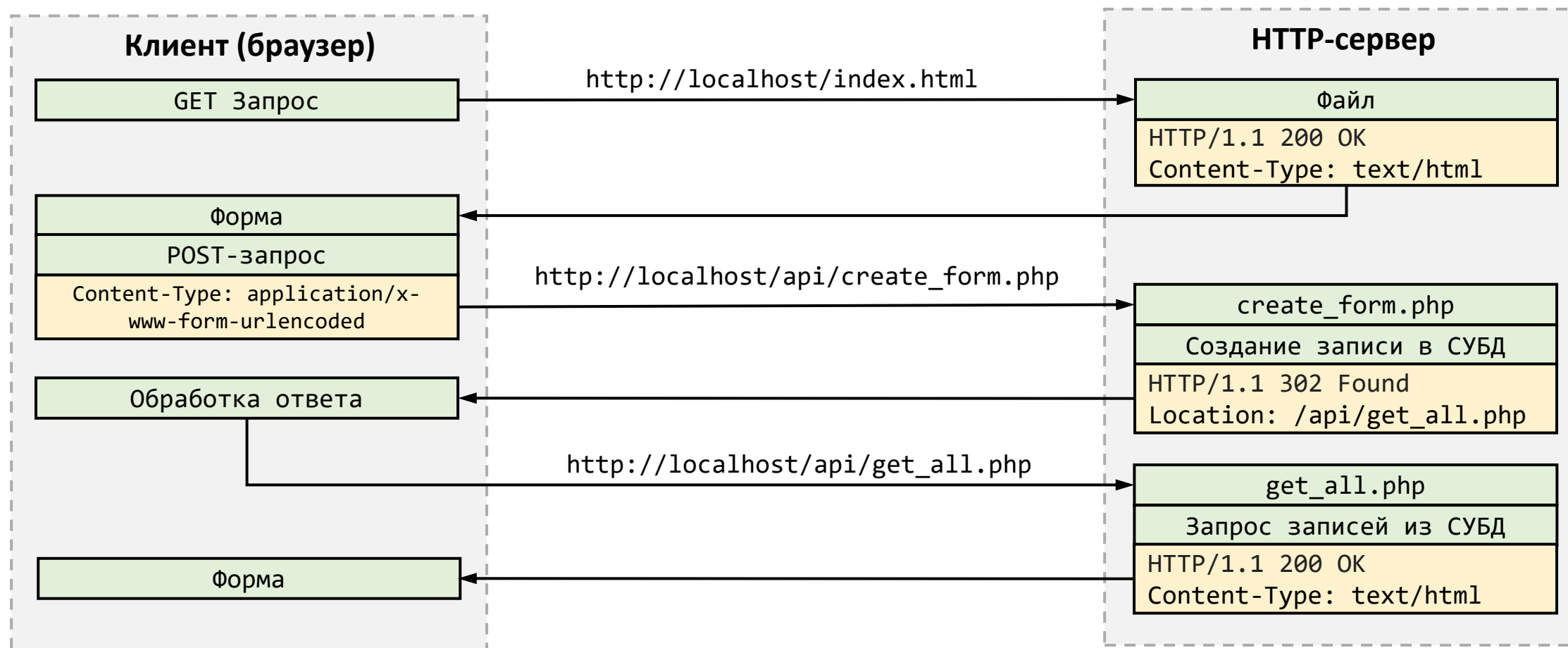


select * FROM users where id = 14				
Output Сообщения Notifications				
id	surname	firstname	lastname	flights
[PK] integer	character var	character var	character var	integer
14	1	Tecr	2	1000



После добавления или изменения записи часто удобнее перенаправить пользователя на другую страницу, если операция прошла успешно.

```
<?php
...
header("Location: /api/get_all.php");
?>
```





Перенаправление возвращает код 302 и браузер автоматически переходит на адрес, указанный в заголовке:

localhost/index.html

Имя:

Фамилия:

Отчество:

Кол-во полетов:



Elements Console Sources **Network** Performance Memory

Filter ☐ Preserve log ☐ Disable cache No throttling ☐ Invert ☐ Hide data URLs ☐ Hide extension URLs **All**

50 ms 100 ms 150 ms 200 ms 250 ms 300 ms 350 ms

Name	Status	Type	Initiator
create_form.php	302	document / Redirect	Other
get_all.php	200	document	create_form.php

localhost/api/get_all.php

- 1) Степанов Иван
- 2) Носова Анастасия
- 3) Сафонов Данил
- 4) Романова Ева
- 5) Виноградова Марина
- 6) Сергеева Анна
- 7) Головин Иван
- 11) Тест Олег
- 12) Смирнов Олег
- 13) 1 Тест
- 14) 1 Тест
- 15) 2 1
- 16) 222 111



- JSON (JavaScript Object Notation) — это формат передачи данных, который используется при взаимодействии веб-сервера и браузера.
- Сегодня JSON — практически стандарт передачи данных в Интернете.



Общие правила создания JSON:

- данные записаны в виде пар ключ:значение;
- данные разделены запятыми;
- объект находится внутри фигурных скобок {};
- массив находится внутри квадратных скобок [].



Особенности:

- В JSON-формате используют двойные кавычки. Кавычки не нужны для значений, не являющихся строкой, — чисел, массивов, булевых значений.
- Одна лишняя или пропущенная запятая или скобка могут привести к сбою работы JSON-файла.
- JSON-формат не поддерживает комментарии. Добавление комментария вызовет ошибку.



- JSON — это компактный и легкочитаемый формат.
- Почти у всех языков есть инструменты для чтения и генерации данных JSON.

```
{
  "Header": {
    "Authorization": "25dc1a8a-642c"
  },
  "Body": {
    "UpdateHomeCallRequest" : {
      "idHomeCallRequest": 29974,
      "homeCallStatus": 3,
      "guid": "C9015DAA-1F92-4043-82A3-E395F5E483F0"
    },
    "Data": [1, 2, 3]
  }
}
```



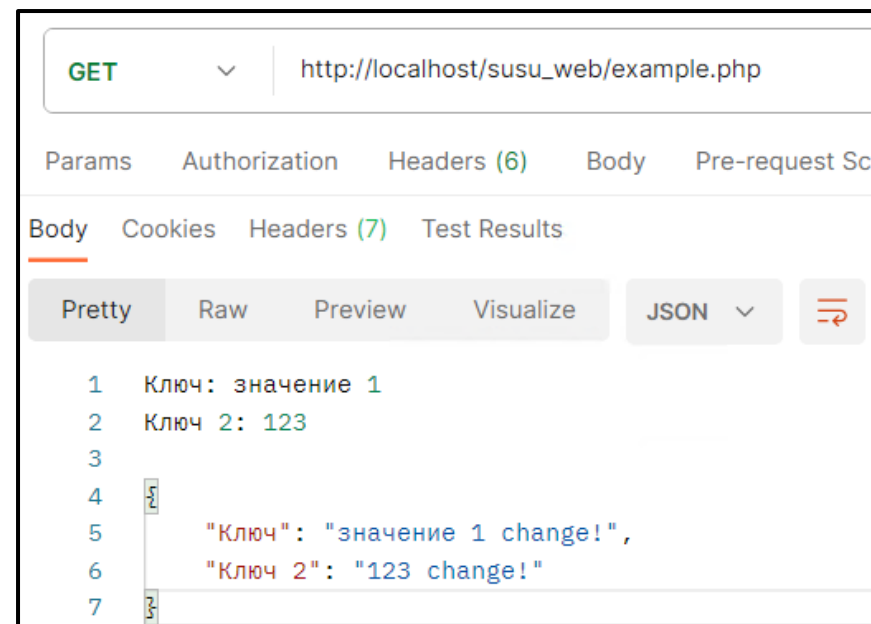
- Строка в ассоциативный массив: `json_decode(строка, <признак ассоциативного массива>)`.
- Ассоциативный массив в строку: `json_encode(массив, Флаги)`.
- `JSON_UNESCAPED_UNICODE` – флаг поддержки символов юникода.

```
<?php
$json_str = '{
    "Ключ": "значение 1",
    "Ключ 2": 123
}';
$json_arr = json_decode($json_str, true);

foreach($json_arr as $key => $value) {
    echo "$key: $value \n";
    $json_arr[$key] = $value . ' change!';
}

echo "\n" . json_encode($json_arr, JSON_UNESCAPED_UNICODE);

?>
```



Добавление записи через JSON



Добавление записи при помощи метода POST и тела запроса в формате JSON:

```
<?php
$postBodyStr = file_get_contents('php://input'); // POST
$postBody = json_decode($postBodyStr, true);

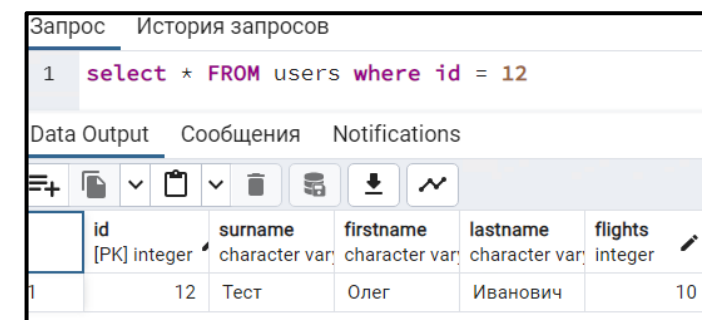
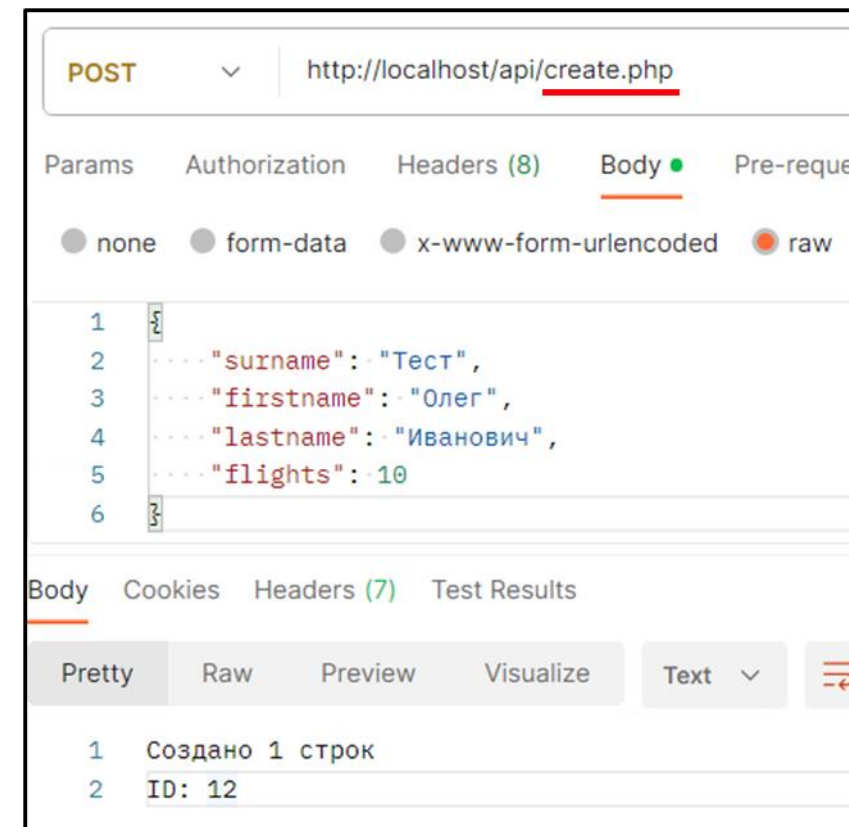
$dsn = "pgsql:host=localhost;port=5432;dbname=postgres;";
$pdo = new PDO($dsn, 'postgres', '1');

$sql = "insert into users
        (surname, firstname, lastname, flights)
        values (:sn, :fn, :ln, :fl)";

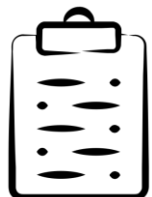
$stmt = $pdo->prepare($sql);
$stmt->bindParam(':sn', $postBody['surname']);
$stmt->bindParam(':fn', $postBody['firstname']);
$stmt->bindParam(':ln', $postBody['lastname']);
$stmt->bindParam(':fl', $postBody['flights']);
$stmt->execute();

echo "Создано ". $stmt->rowCount() . " строк\n";
echo "ID: ". $pdo->lastInsertId();

?>
```



Изменение записи через JSON



Изменение записи при помощи метода POST и тела запроса в формате JSON:

```
<?php
$postBodyStr = file_get_contents('php://input'); // POST
$postBody = json_decode($postBodyStr, true);

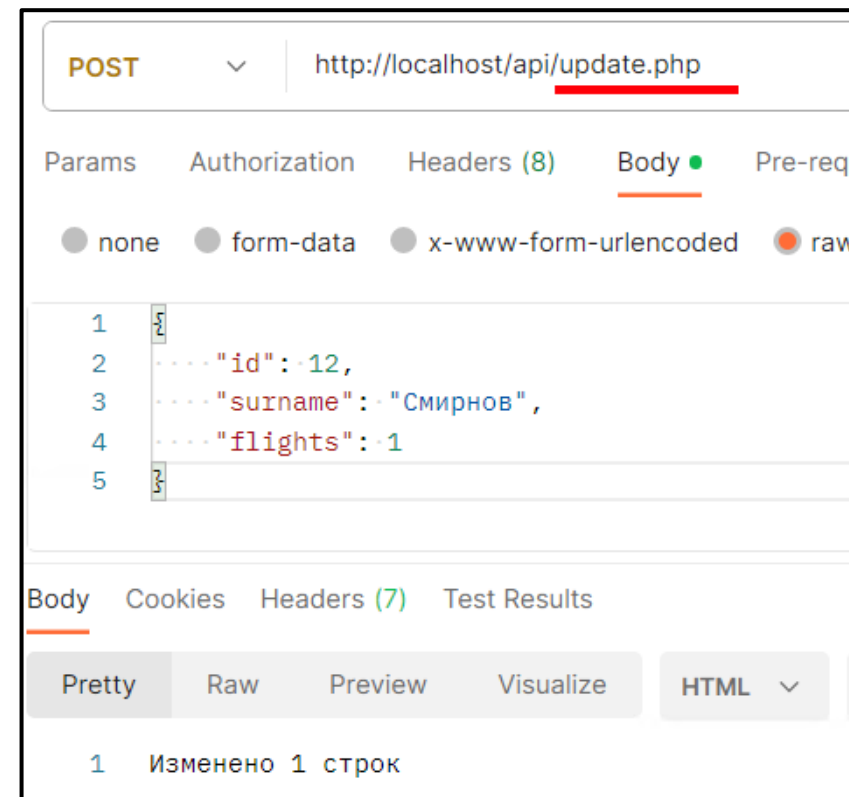
$dsn = "pgsql:host=localhost;port=5432;dbname=postgres;";
$pdo = new PDO($dsn, 'postgres', '1');

$sql = "update users set
        surname = :sn,
        flights = :fl
        where id = :id";

$stmt = $pdo->prepare($sql);
$stmt->bindParam(':id', $postBody['id']);
$stmt->bindParam(':sn', $postBody['surname']);
$stmt->bindParam(':fl', $postBody['flights']);
$stmt->execute();

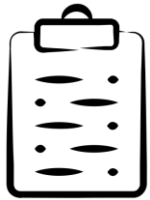
echo "Изменено ". $stmt->rowCount() . " строк\n";

?>
```

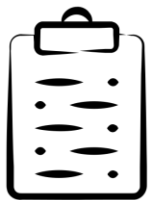


select * FROM users where id = 12

Output					Сообщения	Notifications
id	surname	firstname	lastname	flights		
[PK] integer	character var	character var	character var	integer		
12	Смирнов	Олег	Иванович	1		



- JSON может содержать очень сложную иерархию данных.
- Form Data лучше для передачи бинарных файлов от клиента серверу.
- Form Data можно отправлять без JavaScript (JS). Для отправки JSON необходимо преобразовывать данные к формату JSON перед отправкой при помощи JS.



- JSON для сложных форм.
- Form Data для простых форм (в простых приложениях) и бинарных файлов.



Для подключения одного файла PHP в другой используется функция `include(<путь до файла>)`.

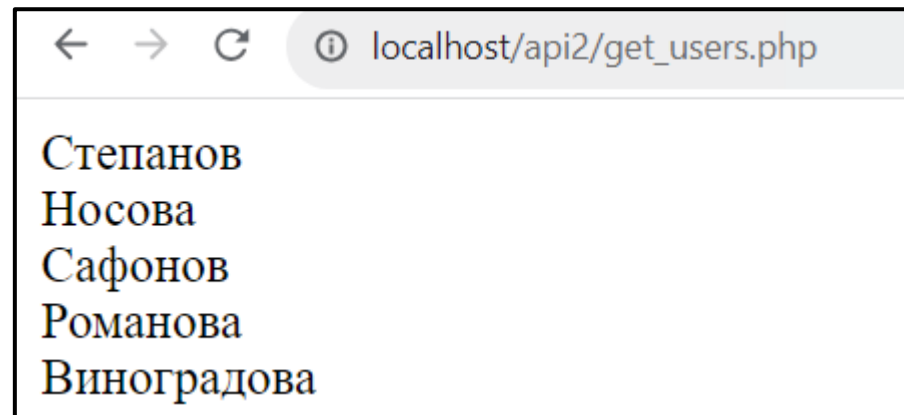
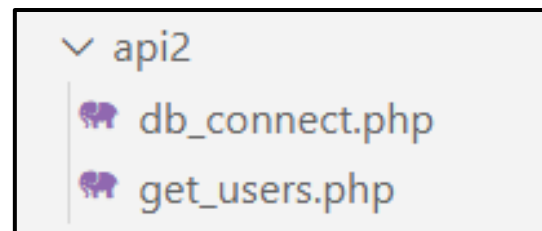
```
<?php // db_connect.php
    $dsn = "pgsql:host=localhost;
    port=5432;dbname=postgres";
    $pdo = new PDO($dsn, 'postgres', '1');
?>
```

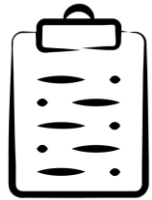
```
<?php // get_users.php
    include('./db_connect.php');

    $stmt = $pdo->prepare("select surname
                           from users
                           limit 5");

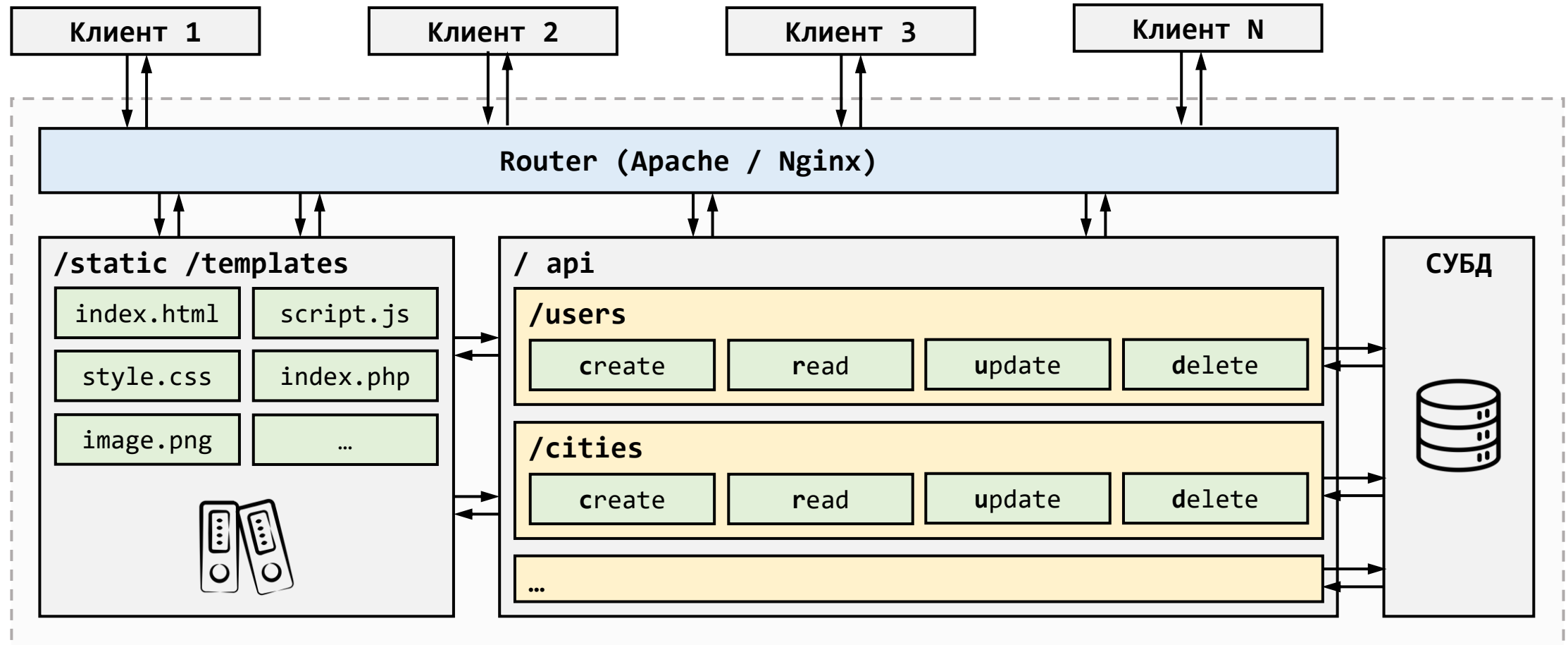
    $stmt->execute();

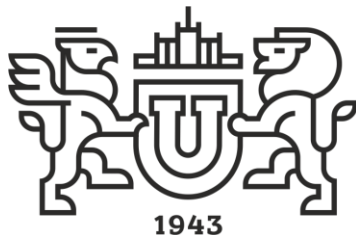
    while ($row = $stmt->fetch()) {
        echo $row['surname']."<br/>";
    }
?>
```





- CRUD - одна из самых фундаментальных концепций в веб-разработке.
- CRUD (Create, Read, Update и Delete) представляет собой четыре основные операции, выполняемые при управлении данными.





Южно-Уральский
государственный
университет

Национальный
исследовательский
университет

КРОК

Спасибо за внимание!



КРОК

Челябинск, ул. Карла Маркса, д. 38