

# ИНСТРУКЦИЯ ПО РАБОТЕ С API TELEGRAM

## Введение

В проекте [https://github.com/vbatalov/kwork\\_fastdimerrr](https://github.com/vbatalov/kwork_fastdimerrr) находятся ключевые файлы:

1. composer.json – содержит необходимые пакеты для работы бота
2. index.php – подключает все необходимые файлы и содержит логику работы бота
3. database.php – содержит логику работы с Базой данных MySQL и конфигурацию для подключения.

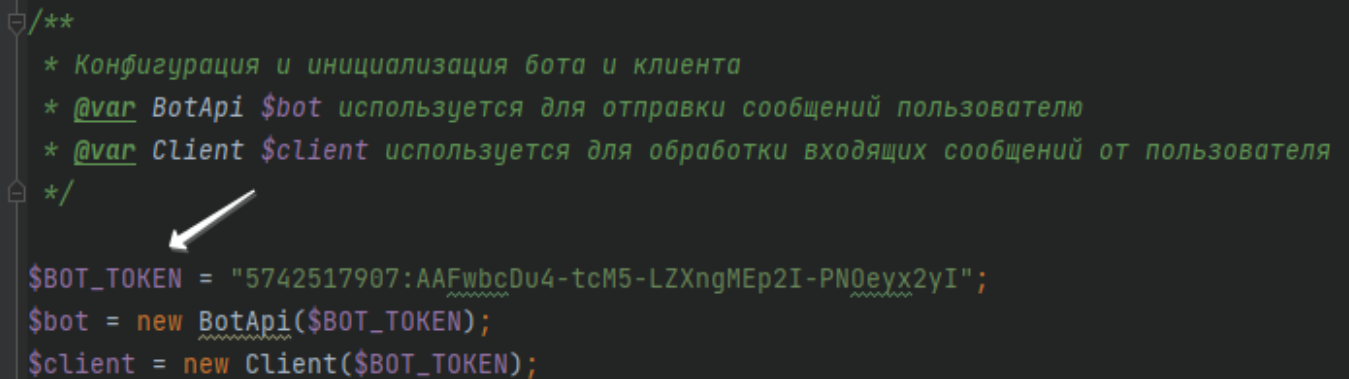
1. Для старта проекта необходимо загрузить архив с файлами на сервер **или** с помощью Терминала выполнить команду

- git clone [https://github.com/vbatalov/kwork\\_fastdimerrr](https://github.com/vbatalov/kwork_fastdimerrr) - скопирует проект

2. После загрузки проекта в каталог сервера, необходимо выполнить команду

- composer install - установит все зависимости.

3. Следующим шагом необходимо настроить конфигурацию бота в файле **index.php**



```
/**
 * Конфигурация и инициализация бота и клиента
 * @var BotApi $bot используется для отправки сообщений пользователю
 * @var Client $client используется для обработки входящих сообщений от пользователя
 */

$BOT_TOKEN = "5742517907:AAFwbcDu4-tcM5-LZXngMEp2I-PN0eyx2yI";
$bot = new BotApi($BOT_TOKEN);
$client = new Client($BOT_TOKEN);
```

В переменную \$BOT\_TOKEN внесите токен своего бота

4. Следующим шагом необходимо настроить базу данных

```
public function __construct()
{
    $this->db = new Manager();

    $this->db->addConnection([
        'driver' => 'mysql',
        'host' => 'localhost',
        'database' => 'test',
        'username' => 'root',
        'password' => 'q1w2e3!',
        'charset' => 'utf8',
        'collation' => 'utf8_unicode_ci',
        'prefix' => '',
    ]);

    $this->db->setAsGlobal();
}
```

Укажите имя базы данных, пользователя и пароль

5. Теперь вся конфигурация готова.

5.1 Регистрируем URL для бота. Перейдите по адресу [example.com/?register\\_bot](http://example.com/?register_bot)

5.2 Инициализируем базу данных. Таблица users будет создана автоматически. Необходимо перейти по адресу [example.com/?database\\_install](http://example.com/?database_install)

```

public function database_install()
{
    $this->db::schema()->dropIfExists( table: "users");

    return $this->db::schema()->create( table: 'users', function (Blueprint $table) {
        $table->integer( column: 'id')->autoIncrement();
        $table->string( column: 'cid');
        $table->string( column: 'first_name')->nullable();
        $table->string( column: 'last_name')->nullable();
        $table->string( column: 'phone')->nullable();
    });
}

```



database.php

За создание таблицы отвечает файл **database.php** и функция **database\_install()**

В результате будет создана таблица users, в которой будут следующие строки:

1. cid – Идентификатор пользователя Telegram
2. first\_name – Имя пользователя
3. last\_name – Фамилия пользователя
4. phone – Телефон пользователя

**Внимание!** Переход по адресу [example.com/?database\\_install](http://example.com/?database_install) полностью удалит таблицу users и создаст заново. Не используйте эту команду, когда у вас будут реальные данные.

```

public function addUser(string $cid, string $first_name, string $last_name)
{
    $this->db::table( table: "users")->updateOrInsert(
        [
            "cid" => $cid,
        ],
        [
            "first_name" => $first_name,
            "last_name" => $last_name,
        ]
    );
}

public function addPhone($cid, $phone)
{
    $this->db::table( table: "users")->updateOrInsert(
        [
            "cid" => $cid,
        ],
        [
            "phone" => $phone
        ]
    );
}
}

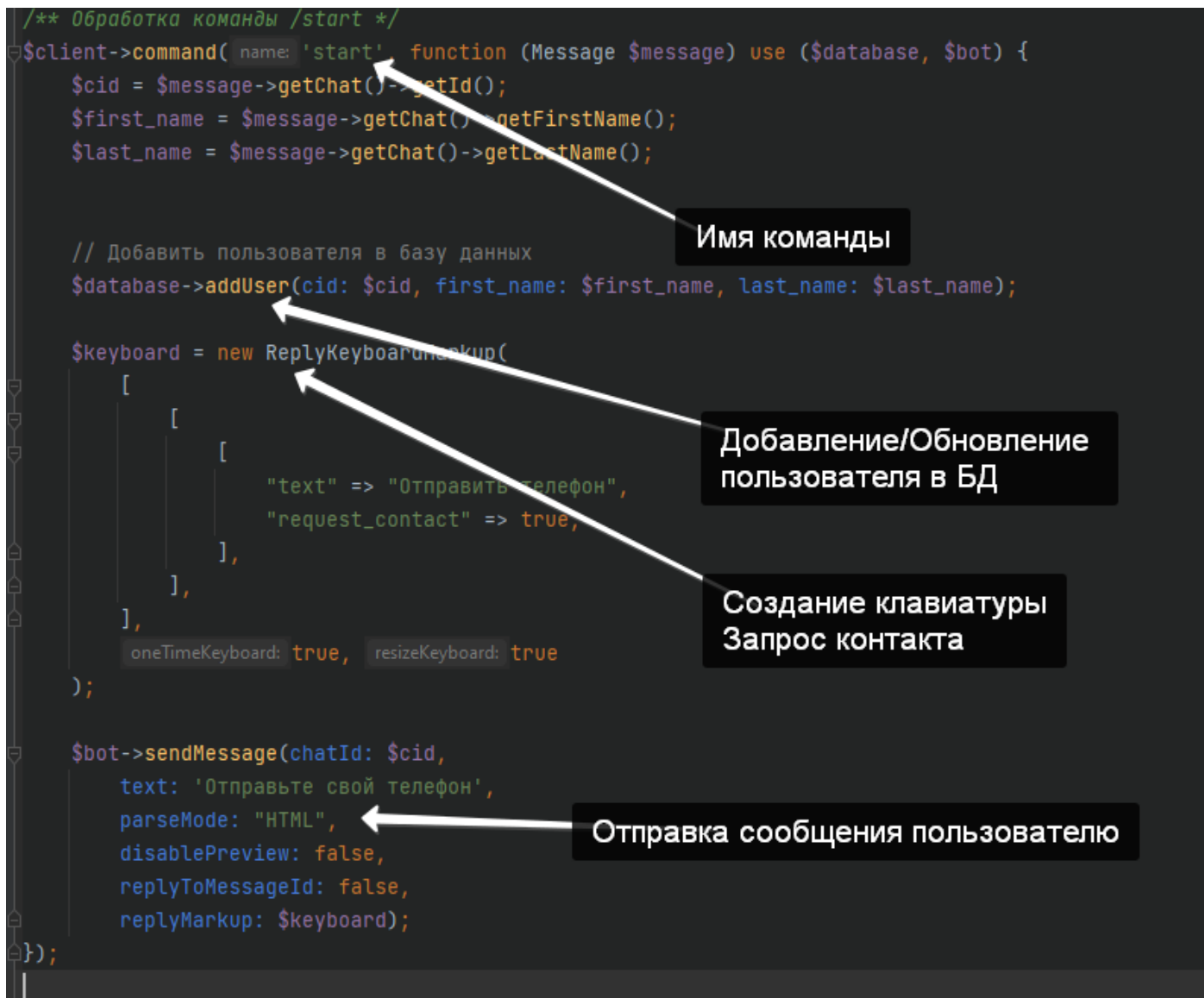
```



database.php

В классе Database так же доступны две функции:

- addUser — добавляет нового пользователя. При этом используется метод updateOrCreate, что позволяет при каждом нажатии /start в боте, добавить или обновить данные о пользователе в базе данных.
- addPhone — добавляет телефон к пользователю.



Файл `index.php` содержит логику работы бота.

**На рисунке показана обработка команды `/start`.**

Весь код на рисунке – это логика работы при нажатии `/start` пользователем.

```

// Обработка всех сообщений
$client->on(function (Update $update) use ($bot, $database) {
    /** @var Message $message содержит всю информацию о сообщении */
    $message = $update->getMessage();
    /** @var string $cid содержит информацию о ID пользователя */
    $cid = $message->getChat()->getId();
    /** @var \TelegramBot\Api\Types\Contact $contact содержит информацию о отправленном контакте */
    $contact = $message->getContact();

    /** Обработка полученного контакта */
    if (!empty($contact)) {
        // Проверка: Пользователь должен отправить СВОЙ контакт, а не любой другой из записной книги
        if ($contact->getUserId() == $cid) {
            $database->addPhone(cid: $cid, phone: $contact->getPhoneNumber());
            $bot->sendMessage(chatId: $cid, text: "Номер сохранен.");
        } else {
            $bot->sendMessage(chatId: $cid, text: "Вы отправили чужой контактный номер.");
        }
    } else {
        $bot->sendMessage(chatId: $cid, text: "Вы не отправили контакт");
    }
}, function () {
    return true;
});

```

## Обработка всех входящих сообщений.

Код на рисунке будет работать **ДЛЯ ВСЕХ** входящих сообщений, полученных от пользователя.

В этом блоке необходимо четко описать логику, как мы будем обрабатывать то или иное сообщение.

По вашему ТЗ в блоке показана обработка полученного контакта.

Возможные сценарии:

1. Пользователь отправляет просто сообщение – Бот отвечает «Вы не отправили контакт»
2. Пользователь отправляет контакт, **но чужой**. Бот отвечает «Вы отправили чужой контакт»
3. Если пользователь отправляет свой контакт, то мы добавляем телефон в базу данных и отвечаем = **Номер сохранен**

Если вы хотите получать от пользователя следующий ответ, например на вопрос «Из какого вы города?», необходимо для каждого пользователя вести учет Cookies файлов

в базе данных, чтобы мы всегда могли четко определить — на какой вопрос сейчас отвечает бот.