References

1. Demidova A.V., Druzhinina O.V., Jacimovic M, Masina O.N., Mijajlovic N. (2018). Synthesis and analysis of multidimensional mathematical models of population dynamics.*Proceedings of the Selected Papers of the 10th International Congress on Ultra Modern Telecommunications and Control Systems ICUMT* (Moscow, Russia, November 5–9, 2018). New York: IEEE Xplore Digital Library. IEEE Catalog Number CFP 1863G-USB. Pp. 361–366.
2. Lotka A. (1925).Elements of physical ecology. Baltimora: Williams and Wilkins.
3. Bazykin A.D. (2003). Nonlinear dynamics of interacting populations [*Nelinejnaya dinamika vzaimodejstvuyushchih populyacij*]. Moscow - Izhevsk: Institute of Computer Research.
4. Vol'terra V. (1976). The mathematical theory of the struggle for existence [*Matematicheskaya teoriya bor'by za sushchestvovanie*]. Moscow: Science.
5. AlexandrovA.Yu., PlatonovA.V., StarkovV.N., StepenkoN.A. (2017). Mathematical modeling and research of the stability of biological communities [*Matematicheskoe modelirovanie i issledovanie ustojchivosti biologicheskih soobshchestv*]. St. Petersburg: «Doe».
6. Druzhinina O.V., Masina O.N. (2009). Research methods for the stability and controllability of fuzzy and stochastic dynamical systems [*Metody issledovaniya ustojchivosti i upravlyaemosti nechetkih i stohasticheskih dinamicheskih sistem*]. Moscow: CC RAS.
7. Pykh Yu.A. (1983). Equilibrium and stability in models of population dynamics [*Ravnovesie i ustojchivost' v modelyah populyacionnoj dinamiki*]. Moscow: Science.
8. Tarova E.D. (2018). A qualitative study of a four-dimensional model of population dynamics using the first Lyapunov parameter [*Kachestvennoe issledovanie chetyrekhmernoj modeli populyacionnoj dinamiki s pomoshch'yu pervogo metoda Lyapunova*] // Nonlinear World. V. 3. Pp. 17-24.
9. SvirezhevYu.M., LogofetD.O. (1978). Sustainability of biological communities [*Ustojchivost' biologicheskih soobshchestv*]. Moscow: Science.

УДК 004.432

О РАЗРАБОТКЕ СЕРВЕР-ПРИЛОЖЕНИЯ СИСТЕМЫ ЭЛЕКТРОННОЙ КОММУНИКАЦИИ

Дмитрий Игоревич Максимов
старший преподаватель
timonpm@mail.ru
г. Елец

Елецкий государственный университет им. И.А. Бунина

Аннотация. Современные информационные технологии открывают большие возможности для коммуникации людей. Среди них можно выделить электронную почту, чаты, блоги, социальные сети, месенджеры. Важным звеном данных технологий являются сервер-приложения, обеспечивающие хранение, обработку данных и взаимодействие пользователей. В статье рассматриваются принципы разработки таких приложений. Рассматриваемое приложение может применяться для реализации систем электронной коммуникации пользователей. В качестве языка программирования использован язык PHP.

Ключевые слова: программирование, сервер-приложение, технология клиент-сервер, электронная коммуникация, PHP.

При проектировании системы электронной коммуникации необходимо выделить основные составляющие и выявить взаимосвязи между ними (рис. 1).

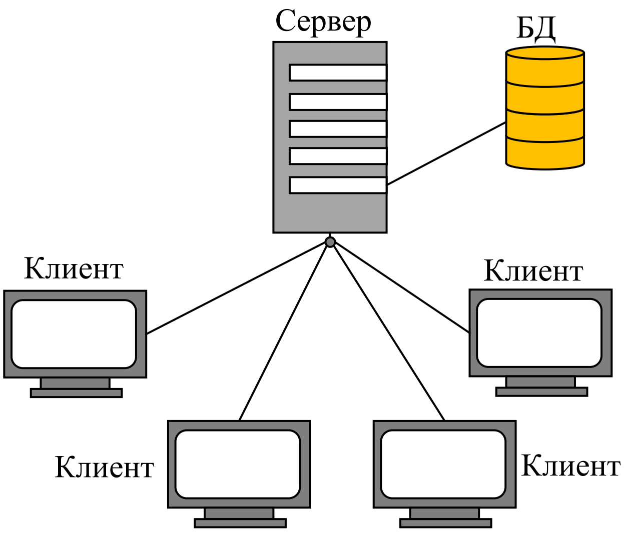

Рис. 1. Структура системы электронной коммуникации

Все данные, которые получает клиент, находятся на сервере. При отправке клиентом сообщения какому-либо клиенту его получает сервер, а уже потом переправляет его адресату. Так выглядит классическая схема архитектуры «клиент-сервер».

Таким образом, в рассматриваемой системе центральное место занимает сервер, а точнее установленное на нем сервер-приложение. В его обязанности входит работа с базами данных, передача и получение сообщений, оповещение клиентов.

Сервер-приложение представляет собой систему взаимосвязанных процессов, выполняющихся на сервере и обеспечивающую подключение и работу клиентов.

Рассматриваемое приложение имеет главный процесс и набор дочерних (рис. 2).

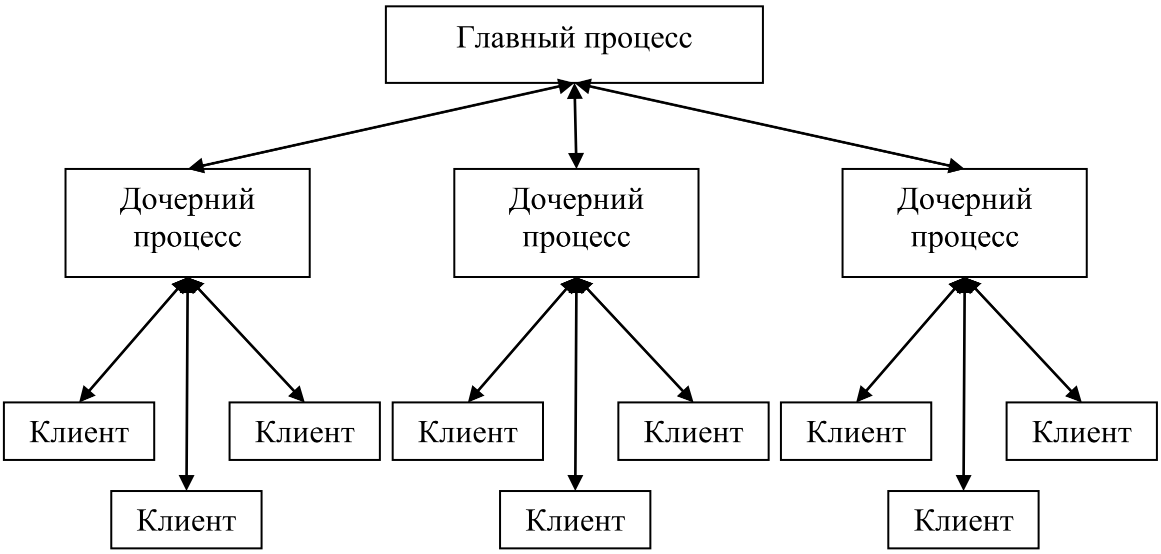

Рис. 2. Схема взаимодействия процессовсервер-приложения

Между главным и дочерними процессами устанавливаются связи при помощи парных сокетов, а между дочерними процессами и клиентами посредством TCP-соединений.

Количество дочерних процессов устанавливается в настройках сервер-приложения и зависит от предполагаемого количества клиентов. Каждый дочерний процесс может принимать до 1024 клиентов. Классы процессов реализуют интерфейс процесса, содержащий методы запуска и обработки сигналов (листинг 1).

```
interface IProcess {
  public function run();
  public function signalHandler($signo, $info = null);
}
class ProcessMaster implements IProcess {
  protected $isRun;
  protected $childSockets;

  public function __construct($childSockets = null){
        $this->childSockets = $childSockets;
        $this->isRun = false;
  }
  ...
}
interface INotification {
  public function attach(Colleague $client);
  public function detach(Colleague $client);
  public function detachAll();
  public function notify(ARequest $message);
}
class ChildProcess implements IProcess, INotification, IBaseAcceptable {
  protected $isRun;
  protected $clients;
  protected $serverSocket;
  protected $masterSocket;
  protected $bases;

  public function __construct($config, $serverSocket, $masterSocket) {
        $this->clients = array();
        $this->serverSocket = $serverSocket;
        $this->masterSocket = $masterSocket;
        $this->config = $config;
        $this->isRun = false;
  }
  ...
}
```

Листинг 1. Интерфейсыпроцессов

Основным манизмом взаимодействия сервер-приложения и клиентов является механизм сообщений. Под сообщением понимается некоторый блок данных, который может быть передан по установленному соединению. Сообщение должно иметь заранее определенный формат.

В базовом классе сообщений реализовано преобразование в JSON формат, а также на основе шаблона проектирования «Мост» подключается обработчик сообщения в виде статического поля (листинг 2).

```
abstract class ARequest implements JsonSerializable {
  protected $from;
  protected $content;
  protected $id;
  protected static $handler = null;

  public function jsonSerialize() {
```

```
            return
            [
                  'id' => $this->id,
                  'handler' =>"",
                  'from' => $this->from,
                  'content' => $this->content
            ];
      }
      public function __construct($id, $content, $from) {
            $this->id = $id;
            $this->from = $from;
            $this->content = $content;
      }
      public function getHandler() {
            return self::$handler;
      }
    }
    interface IHandler {
      public function execute(ARequest $message, $sender);
      public function create();
      public function toClientData(ARequest $message);
    }
```

Листинг 2.Базовый класс сообщения

Разработанный таким образом базовый класс сообщений позволяет добавлять в систему различные типы сообщений с возможностью их обработки на сервере.

Обмен сообщениями между клиентами и дочерними процессами реализовано на основе шаблона проектирования «Посредник», что позволяет использовать различные типы сообщений и выполнять их обработку и маршрутизацию (листинг 1, 3).

```
    abstract class Colleague {
      protected $notificator;

      public function __construct(INotification $notificator = null,
$stream = null) {
            $this->notificator = $notificator;
      }
      public function __destruct() {
            $this->notificator = null;
      }
      public function setNotificator(INotification $notificator = null) {
            $this->notificator = $notificator;
      }

      public abstract function notify(ARequest $message);
    }
    class Client extends Colleague {
      protected $id;
      protected $authorized;
      protected $socket;

      public function __construct(INotification $master = null, $stream =
null) {
            parent::__construct($master);
            $this->socket = $stream;
            $this->id = -1;
```

```
            $this->authorized = AUTHORIZED_UNSET;
        }
        ...
    }
```

Листинг 3. Класс клиента

Для передачи сообщений через TCP-соединение происходит разбиение сообщения на части фиксированного размера и их упаковка в байт-код (рис. 3, листинг 4).

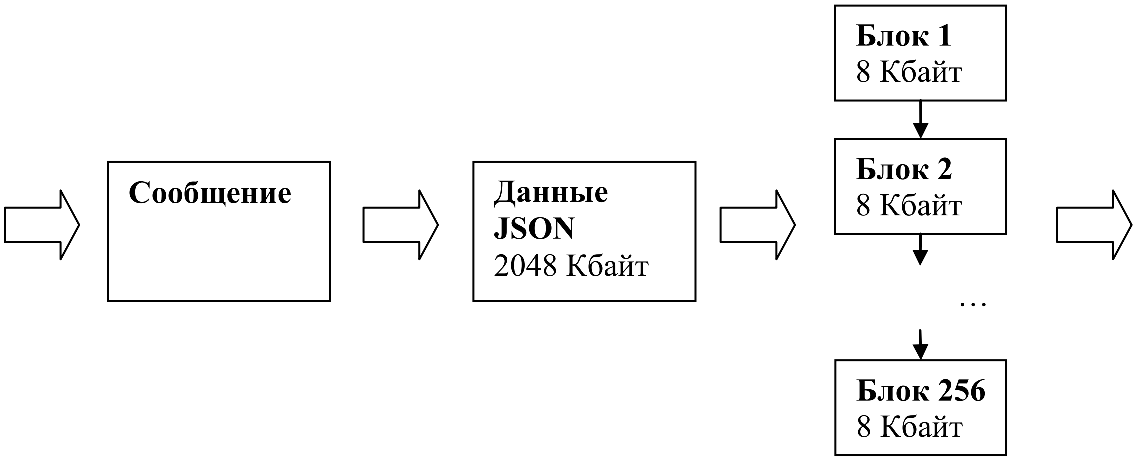

Рис. 3. Упаковка и разбиение сообщения

```
interfaceIPack {
    publicstaticfunctionpack($data);
    public static function unpack($data);
}
interface IStream {
    public static function fromStream($stream);
    public static function toStream($stream, ARequest $message);
}
```

Листинг 4. Интерфейсы классов упаковки и передачи

Использование механизма упаковки и потоков передачи позволяет передавать через соединение сообщения любых типов в одном формате.

Взаимодействие с базами данных представлено в виде базового класса и интерфейса подключения баз данных к серверу, что позволяет реализовывать подключение различных типов баз данных в процессе работы сервера (листинг 5).

```
abstract class Base {
    protected $config;
    protected $base;

    public function __construct($config) {
            $this->config = $config;
            $this->base = null;
    }
    public abstract function connect();
```

```
    public abstract function disconnect();
    public abstract function query($query);
    ...
}
interface IBaseAcceptable {
    public function attachBase(Base $base);
    public function detachBase(Base $base);
    public function detachAllBases();
}
```

Листинг 5. Базовый класс баз данных

Сервер-приложение представляет собой «PHP-демона» и работает в фоновом режиме (листинг 6).

```
class Demon{
    protected $handler;
    protected $pidFile;
    protected $logPath;

    public function __construct(IProcess $handler)
    {
          $this->handler = $handler;
          $this->pidFile = "pid.pid";
          $this->logPath = dirname(__FILE__);
    }
    ...
}
class Server implements IProcess, IBaseAcceptable {
    protected $config;
    protected $bases;

    public function __construct($config) {
          $this->config = $config;
    }
    ...
}
```

Листинг 6. Класс сервера

При запуске приложения происходит создание объекта «демона», который получает заранее настроенный объект сервера. «Демон» сохраняет идентификатор процесса в файл и ведет логирование.

Таким образом, разработанная система может обслуживать неограниченное количество клиентов. Используемый механизм сообщений позволяет добавлять различные типы сообщений в систему и корректно их обрабатывать. В систему можно динамически подключать различные базы данных. Реализация в виде «демона» освобождает интерфейс сервера и возвращает управление администратору операционной системы.

Список литературы

1. Маркин А.В. Основы web-программирования на PHP: учебное пособие / А.В. Маркин, С.С. Шкарин. М.: Диалог-МИФИ, 2012.
2. Матяш С.А. Корпоративные информационные системы: учебное пособие. М.; Берлин: Директ-Медиа, 2015

3. Пирамидин А. Учебник PHP. URL: https://phpclub.ru/manrus (дата обращения: 13.01.2019)
4. Рыбальченко М.В. Архитектура информационных систем: учебное пособие / М.В. Рыбальченко; Министерство образования и науки Российской Федерации, Южный федеральный университет. Таганрог: Издательство Южного федерального университета, 2015. Ч. 1.
5. Шабашов В.Я. Организация доступа к данным из PHP приложений для различных СУБД: учебное пособие по дисциплине «Web-программирование». М.; Берлин: Директ-Медиа, 2019.

ABOUT DEVELOPMENT OF THE SERVER-APPLICATION FOR ELECTRONIC COMMUNICATION SYSTEM

D.I. Maksimov
senior lecturer
timonpm@mail.ru
Yelets

Bunin Yelets State University

Abstract. Modern information technologies offer great opportunities for people to communicate. Among them are e-mail, chat rooms, blogs, social networks, instant messengers. An important part of the data are server applications that store and process user data. The article presents the principles for developing such applications. The considered application can be used for the implementation of electronic communication systems of users. The programming language is PHP.

Keywords: programming, server-application, client-server technology, electronic communication, PHP.

References

1. Markin, A.V. (2012). Basics of PHP web programming: tutorial [*Osnovy web-programmirovaniya na PHP: uchebnoye posobiye*]. Moscow: Dialogue-MEPI.
2. Matyash, S.A. (2015). Corporate information systems: a tutorial [*Korporativnyye informatsionnyye sistemy: uchebnoye posobiye*]. Moscow; Berlin: Direct Media.
3. Pyramidin, A., Book PHP [Electronic resource] [*Uchebnik PHP*]. URL: https://phpclub.ru/manrus
4. Rybalchenko, M.V. (2015). Information Systems Architecture: study guide [*Arkhitektura informatsionnykh sistem: uchebnoye posobiye*] / M.V. Rybalchenko; Ministry of Education and Science of the Russian Federation, Southern Federal University. Taganrog: Publishing House of the Southern Federal University.
5. Shabashov, V.Ya. (2019). Organization of access to data from PHP applications for various DBMS: a tutorial on the discipline "Web-programming" [*Organizatsiya dostupa k dannym iz PHP prilozheniy dlya razlichnykh SUBD: uchebnoye posobiye po distsipline «Web-programmirovaniye»*]. Moscow; Berlin: Direct Media.