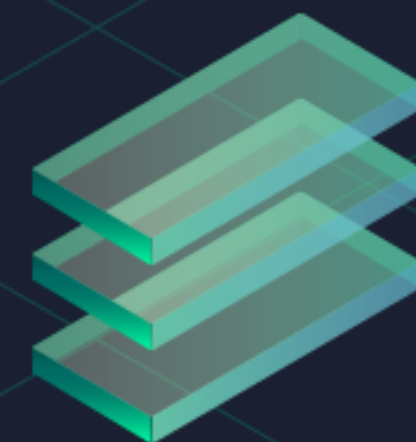




Раздел 6:

Базы данных



План



План

- Хранение пользовательских данных



План

- Хранение пользовательских данных
- SQL vs NoSQL



План

- Хранение пользовательских данных
- SQL vs NoSQL
- MongoDB



План

- Хранение пользовательских данных
- SQL vs NoSQL
- MongoDB
- Node.js Mongo Driver



План

- Хранение пользовательских данных
- SQL vs NoSQL
- MongoDB
- Node.js Mongo Driver
- Mock



Где можно хранить данные юзера



Где можно хранить данные юзера

- В браузере



Где можно хранить данные юзера

- В браузере
- На сервере:



Где можно хранить данные юзера

- В браузере
- На сервере:
 - в файлике на диске



Где можно хранить данные юзера

- В браузере
- На сервере:
 - в файлике на диске
 - в программе для хранения данных — базе данных



Реляционная база данных

база данных, основанная на реляционной модели данных. Структурированная база данных, которая обычно представляется как набор связанных между собой таблиц



Данные

```
[  
  {  
    "name": "Дамблдор",  
    "colorCoat": "rgb(241, 43, 107)",  
    "colorEyes": "yellow",  
    "colorFireball": "#5ce6c0"  
  },  
  {  
    "name": "Волдеморт",  
    "colorCoat": "rgb(215, 210, 55)",  
    "colorEyes": "black",  
    "colorFireball": "#30a8ee"  
  }  
]
```



Имя	Возраст	Цвет плаща	Цвет глаз
Пендальф	40	серый	rgb(77,22,12)
Радагаст	32	зелёный	rgb(228,2,228)
Гарри Поттер	17	чёрный	rgb(0,0,0)
Баба Яга	99	прозрачный	rgb(255,255,0)



Имя	Возраст	Цвет плаща	Цвет глаз
Пендальф	40 строка	серый	rgb(77,22,12)
Радагаст	32	зелёный	rgb(228,2,228)
Гарри Поттер	17	чёрный	rgb(0,0,0)
Баба Яга	99	прозрачный	rgb(255,255,0)



Имя	Возраст	Цвет плаща	Цвет глаз
Пендальф	40	серый	rgb(77,22,12)
Радагаст	21	зелёный	rgb(228,2,228)
Гарри Поттер	17	чёрный	rgb(0,0,0)
Баба Яга	99	прозрачный	rgb(255,255,0)

строка

столбец



Имя	Возраст	Цвет плаща	Цвет глаз
Пендальф	40	серый	rgb(77,22,12)
Радагаст	32	зеленый	rgb(228,2,228)
Гарри Поттер	17	чёрный	rgb(0,0,0)
Баба Яга	99	прозрачный	rgb(255,255,0)

таблица (связь)

столбец

строка



Имя игрока	Аватар	Персонаж			
Игорь	happy.png	Радагаст			
Евгений	image.jpg	Пендальф			
Петр	smile.gif	Баба Яга			
Николай	avatar.bmp	Имя	Возраст	Цвет плаща	Цвет глаз
		Пендальф	40	серый	rgb(77,22,12)
		Радагаст	32	зелёный	rgb(228,2,228)
		Гарри Поттер	17	чёрный	rgb(0,0,0)
		Баба Яга	99	прозрачный	rgb(255,255,0)



SQL (Structured Query Language)

язык запросов, применяемый для управления
данными в реляционной базе данных



РСУБД (RDBMS)

- Oracle Database
- Microsoft SQL Server
- MySQL (Oracle Corporation)
- IBM DB2
- IBM Informix
- SAP Sybase Adaptive Server Enterprise
- SAP Sybase IQ
- Teradata



SQL



SQL

— Плюсы



SQL

- Плюсы

- Универсальность



SQL

- Плюсы
 - Универсальность
 - Структурированность



SQL

- Плюсы
 - Универсальность
 - Структурированность
 - Гарантия целостности данных



SQL

- Плюсы
 - Универсальность
 - Структурированность
 - Гарантия целостности данных
- Минусы



SQL

- Плюсы
 - Универсальность
 - Структурированность
 - Гарантия целостности данных
- Минусы
 - Плохая масштабируемость



SQL

- Плюсы
 - Универсальность
 - Структурированность
 - Гарантия целостности данных
- Минусы
 - Плохая масштабируемость
 - Реляционная модель данных



SQL

- Плюсы
 - Универсальность
 - Структурированность
 - Гарантия целостности данных
- Минусы
 - Плохая масштабируемость
 - Реляционная модель данных
 - Неоправданная сложность



SQL

- Плюсы
 - Универсальность
 - Структурированность
 - Гарантия целостности данных
- Минусы
 - Плохая масштабируемость
 - Реляционная модель данных
 - Неоправданная сложность
 - Закрытость



NoSQL

термин, обозначающий ряд подходов, направленных на реализацию хранилищ баз данных, имеющих существенные отличия от моделей, используемых в традиционных реляционных СУБД с доступом к данным средствами языка SQL



Виды NoSQL



Виды NoSQL

- Хранилище типа «ключ-значение» (Key-value store)



Виды NoSQL

- Хранилище типа «ключ-значение» (Key-value store)
- Объектные базы данных (Object Database)



Виды NoSQL

- Хранилище типа «ключ-значение» (Key-value store)
- Объектные базы данных (Object Database)
- Документориентированные базы данных (Document store)



Виды NoSQL

- Хранилище типа «ключ-значение» (Key-value store)
- Объектные базы данных (Object Database)
- Документоориентированные базы данных (Document store)
- Графоориентированные базы данных (Graph database)



Виды NoSQL

- Хранилище типа «ключ-значение» (Key-value store)
- Объектные базы данных (Object Database)
- Документоориентированные базы данных (Document store)
- Графоориентированные базы данных (Graph database)
- прочие нереляционные базы данных



NoSQL



NoSQL

— Плюсы



NoSQL

- Плюсы
 - Масштабируемость



NoSQL

- Плюсы
 - Масштабируемость
 - Отсутствие реляционной модели



NoSQL

- Плюсы
 - Масштабируемость
 - Отсутствие реляционной модели
 - Бесплатность и открытость исходного кода



NoSQL

- Плюсы
 - Масштабируемость
 - Отсутствие реляционной модели
 - Бесплатность и открытость исходного кода
 - Легковесность



NoSQL

- Плюсы
 - Масштабируемость
 - Отсутствие реляционной модели
 - Бесплатность и открытость исходного кода
 - Легковесность
- Минусы



NoSQL

- Плюсы
 - Масштабируемость
 - Отсутствие реляционной модели
 - Бесплатность и открытость исходного кода
 - Легковесность
- Минусы
 - Отсутствие фиксированной модели данных



NoSQL

- Плюсы

- Масштабируемость
- Отсутствие реляционной модели
- Бесплатность и открытость исходного кода
- Легковесность

- Минусы

- Отсутствие фиксированной модели данных
- Целостность данных может не гарантироваться в каждый момент времени



MongoDB

документоориентированная система управления
базами данных (СУБД) с открытым исходным кодом,
не требующая описания схемы таблиц



mongoDB®

www.mongodb.com/



Особенности MongoDB



Особенности MongoDB

- JSON-подобное хранилище (документ)



Особенности MongoDB

- JSON-подобное хранилище (документ)
- Масштабируемость (scaling)



Особенности MongoDB

- JSON-подобное хранилище (документ)
- Масштабируемость (scaling)
- Доступность (high availability)



Особенности MongoDB

- JSON-подобное хранилище (документ)
- Масштабируемость (scaling)
- Доступность (high availability)
- Продвинутое техники индексации (текст, координаты, даты)



Особенности MongoDB

- JSON-подобное хранилище (документ)
- Масштабируемость (scaling)
- Доступность (high availability)
- Продвинутое техники индексации (текст, координаты, даты)
- Eventually consistency



Особенности MongoDB

- JSON-подобное хранилище (документ)
- Масштабируемость (scaling)
- Доступность (high availability)
- Продвинутое техники индексации (текст, координаты, даты)
- Eventually consistency
- Файловое хранилище



Система Управления Базами Данных

Приложение, которое предоставляет
фиксированный API для работы с базой данных.



Занимается непосредственно хранением данных на
конкретном сервере (серверах)



[Atlas](#)[Community Server](#)[Enterprise Server](#)[Ops Manager](#)[Compass](#)[Connector for BI](#)[Current Release](#) | [Previous Releases](#) | [Development Releases](#)

Current Stable Release (3.6.3)

02/22/2018: [Release Notes](#) | [Changelog](#)
Download Source: [tgz](#) | [zip](#)

 [Windows](#) [Linux](#) [OSX](#)

Version:

OS X 10.7+ 64-bit w/SSL x64 ▾

Package Manager:

[Instructions for installing with Homebrew](#)

Binary: [Installation Instructions](#) | [All Version Binaries](#)

 [DOWNLOAD \(tgz\)](#)

https://fastdl.mongodb.org/osx/mongodb-osx-ssl-x86_64-3.6.3.tgz

Deploy a free cluster in the cloud with MongoDB Atlas, our database service that gets you up and running in minutes.

www.mongodb.com/download-center?jmp=nav#community

Запуск MongoDB сервера



Запуск MongoDB сервера

- Создать папку, где будут храниться данные



Запуск MongoDB сервера

- Создать папку, где будут храниться данные
- Указать порт:
--port 27017



Запуск MongoDB сервера

- Создать папку, где будут храниться данные
- Указать порт:
--port 27017
- Указать путь до папки:
--dbpath=./data



Запуск MongoDB сервера

- Создать папку, где будут храниться данные
- Указать порт:
--port 27017
- Указать путь до папки:
--dbpath=./data
- Запустить сервер:
mongod --port 27017 --dbpath=./data



Клиент для работы с базой данных



Клиент для работы с базой данных

— Подключается к БД



Клиент для работы с базой данных

- Подключается к БД
- Следит за подключением к базе данных



Клиент для работы с базой данных

- Подключается к БД
- Следит за подключением к базе данных
- Работает по внутреннему протоколу работы с БД



Клиент для работы с базой данных

- Подключается к БД
- Следит за подключением к базе данных
- Работает по внутреннему протоколу работы с БД
- Предоставляет удобный API для разработчика



Подключение

```
const {MongoClient} = require(`mongodb`);

const url = `mongodb://localhost:27017`;

const connectAndRead = async () => {
  const client = await MongoClient.connect(url);
  const db = client.db(`my-fancy-game`);

  const collection = db.collection(`wizards`);
  console.log(collection);

  client.close();
};

connectAndRead().catch((e) => {
  throw e;
});
```



Базовые понятия



Базовые понятия

- База данных (*database, db*) – один сервер баз данных может содержать несколько баз данных, обращение к конкретной базе происходит при помощи указания имени базы данных



Базовые понятия

- База данных (*database, db*) – один сервер баз данных может содержать несколько баз данных, обращение к конкретной базе происходит при помощи указания имени базы данных
- Коллекция (*collection*) – набор однотипных данных в БД (таблица), например – коллекция магов, коллекция фото, коллекция пользователей



Базовые понятия

- База данных (*database, db*) – один сервер баз данных может содержать несколько баз данных, обращение к конкретной базе происходит при помощи указания имени базы данных
- Коллекция (*collection*) – набор однотипных данных в БД (таблица), например – коллекция магов, коллекция фото, коллекция пользователей
- Курсор (*cursor*) – итератор, который позволяет ограничивать или управлять данными при их получении



Паттерн Итератор (Курсор)

```
class Cursor {  
  constructor(data) {  
    this.data = data;  
  }  
  
  skip(count) {  
    return new Cursor(this.data.slice(count));  
  }  
  
  limit(count) {  
    return new Cursor(this.data.slice(0, count));  
  }  
  
  count() {  
    return this.data.length;  
  }  
  
  toArray() {  
    return this.data;  
  }  
}
```



Базовые операции MongoDB

mongodb.github.io/node-mongodb-native/3.0/tutorials/crud/



Базовые операции MongoDB

– Чтение – `find()`



Базовые операции MongoDB

- Чтение – `find()`
 - Принимает на вход объект запроса в виде JSON



Базовые операции MongoDB

- Чтение — `find()`
 - Принимает на вход объект запроса в виде JSON
 - Возвращает Cursor с полученными данными



Базовые операции MongoDB

- Чтение — `find()`
 - Принимает на вход объект запроса в виде JSON
 - Возвращает Cursor с полученными данными
- Запись — `insertOne`, `insertMany`



Базовые операции MongoDB

- Чтение – `find()`
 - Принимает на вход объект запроса в виде JSON
 - Возвращает `Cursor` с полученными данными
- Запись – `insertOne`, `insertMany`
 - `insertOne` – принимает на вход объект и записывает его в коллекцию



Базовые операции MongoDB

- Чтение — `find()`
 - Принимает на вход объект запроса в виде JSON
 - Возвращает `Cursor` с полученными данными
- Запись — `insertOne`, `insertMany`
 - `insertOne` — принимает на вход объект и записывает его в коллекцию
 - `insertMany` — принимает на вход массив объектов и записывает их все в коллекцию по одному



Базовые операции MongoDB

- Чтение — `find()`
 - Принимает на вход объект запроса в виде JSON
 - Возвращает `Cursor` с полученными данными
- Запись — `insertOne`, `insertMany`
 - `insertOne` — принимает на вход объект и записывает его в коллекцию
 - `insertMany` — принимает на вход массив объектов и записывает их все в коллекцию по одному
 - Возвращает объект с результатом операции, если всё хорошо



Базовые операции MongoDB

- Чтение — `find()`
 - Принимает на вход объект запроса в виде JSON
 - Возвращает Cursor с полученными данными
- Запись — `insertOne`, `insertMany`
 - `insertOne` — принимает на вход объект и записывает его в коллекцию
 - `insertMany` — принимает на вход массив объектов и записывает их все в коллекцию по одному
 - Возвращает объект с результатом операции, если всё хорошо
 - Выкидывает ошибку, если запрос некорректен или произошла ошибка

mongodb.github.io/node-mongodb-native/3.0/tutorials/crud/



Базовые операции MongoDB

mongodb.github.io/node-mongodb-native/3.0/tutorials/crud/



Базовые операции MongoDB

– Обновление – `updateOne`, `updateMany`



Базовые операции MongoDB

- Обновление – `updateOne`, `updateMany`
 - Принимает на вход два объекта – один запрос, другой объект и параметры обновления



Базовые операции MongoDB

- Обновление – `updateOne`, `updateMany`
 - Принимает на вход два объекта – один запрос, другой объект и параметры обновления
 - Возвращает объект с результатом операции, если всё хорошо



Базовые операции MongoDB

- Обновление – `updateOne`, `updateMany`
 - Принимает на вход два объекта – один запрос, другой объект и параметры обновления
 - Возвращает объект с результатом операции, если всё хорошо
 - Выкидывает ошибку, если запрос некорректен или произошла ошибка



Базовые операции MongoDB

- Обновление — `updateOne`, `updateMany`
 - Принимает на вход два объекта — один запрос, другой объект и параметры обновления
 - Возвращает объект с результатом операции, если всё хорошо
 - Выкидывает ошибку, если запрос некорректен или произошла ошибка
- Удаление — `deleteOne`, `deleteMany`



Базовые операции MongoDB

- Обновление — `updateOne`, `updateMany`
 - Принимает на вход два объекта — один запрос, другой объект и параметры обновления
 - Возвращает объект с результатом операции, если всё хорошо
 - Выкидывает ошибку, если запрос некорректен или произошла ошибка
- Удаление — `deleteOne`, `deleteMany`
 - Принимает на вход объект запроса



Базовые операции MongoDB

- Обновление — `updateOne`, `updateMany`
 - Принимает на вход два объекта — один запрос, другой объект и параметры обновления
 - Возвращает объект с результатом операции, если всё хорошо
 - Выкидывает ошибку, если запрос некорректен или произошла ошибка
- Удаление — `deleteOne`, `deleteMany`
 - Принимает на вход объект запроса
 - Возвращает объект с результатом операции, если всё хорошо



Базовые операции MongoDB

- Обновление — `updateOne`, `updateMany`
 - Принимает на вход два объекта — один запрос, другой объект и параметры обновления
 - Возвращает объект с результатом операции, если всё хорошо
 - Выкидывает ошибку, если запрос некорректен или произошла ошибка
- Удаление — `deleteOne`, `deleteMany`
 - Принимает на вход объект запроса
 - Возвращает объект с результатом операции, если всё хорошо
 - Выкидывает ошибку, если запрос некорректен или произошла ошибка

mongodb.github.io/node-mongodb-native/3.0/tutorials/crud/



Объект запроса (selector)



Объект запроса (selector)

- Представляет собой шаблон объекта который мы ищем



Объект запроса (selector)

- Представляет собой шаблон объекта который мы ищем
- Например, `{name: `Пендальф`}` – следует читать как:



Объект запроса (selector)

- Представляет собой шаблон объекта который мы ищем
- Например, `{name: `Пендальф`}` — следует читать как:
 - Найди в коллекции объекты у которых есть поле *name*



Объект запроса (selector)

- Представляет собой шаблон объекта который мы ищем
- Например, `{name: `Пендальф`}` – следует читать как:
 - Найди в коллекции объекты у которых есть поле *name*
 - Отфилтруй среди них, те объекты, значение поля *name* которых равно *Пендальф*



Объект запроса (selector)

- Представляет собой шаблон объекта который мы ищем
- Например, `{name: `Пендальф`}` – следует читать как:
 - Найди в коллекции объекты у которых есть поле *name*
 - Отфильтруй среди них, те объекты, значение поля *name* которых равно *Пендальф*
- Поддерживает более сложные запросы



GridFS

Распределённая файловая система, сделанная
поверх MongoDB и встроенная прямо в СУБД
MongoDB



GridFS API



GridFS API

- Автоматически разбивает данные на части (chunks)



GridFS API

- Автоматически разбивает данные на части (chunks)
- Имеет очень просто интерфейс основанный на *fs.Stream*



GridFS API

- Автоматически разбивает данные на части (chunks)
- Имеет очень просто интерфейс основанный на *fs.Stream*
 - *openUploadStream* – создаёт *WriteStream* в который можно писать данные



GridFS API

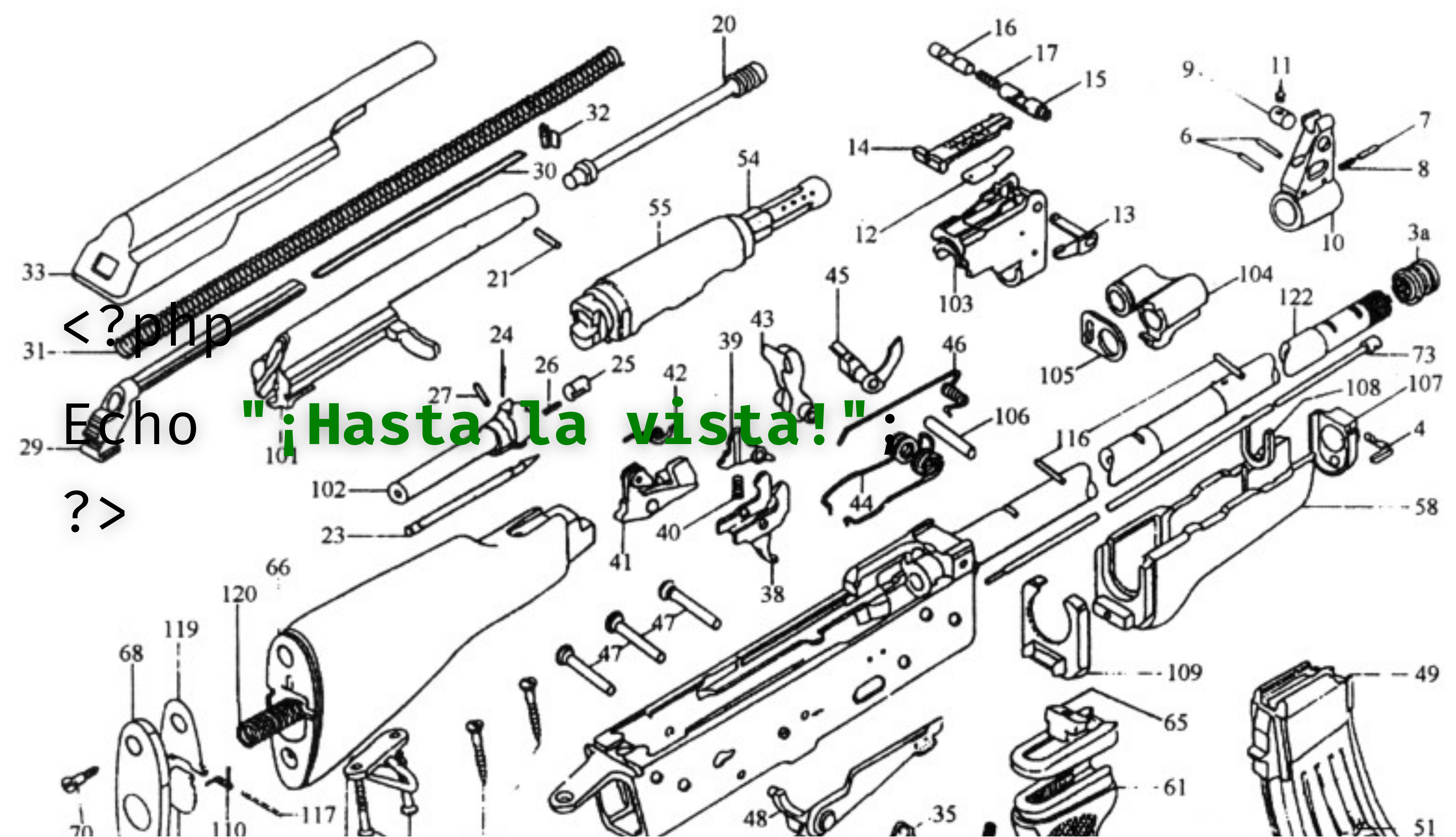
- Автоматически разбивает данные на части (chunks)
- Имеет очень просто интерфейс основанный на *fs.Stream*
 - *openUploadStream* – создаёт *WriteStream* в который можно писать данные
 - *openDownloadStreamByName* – создаёт *ReadStream* из которого можно прочитать данные



Mock

Техника позволяющая подменить настоящие данные, функции и объекты — искусственными для целей тестирования и проверки работоспособности кода





<?php

Echo

?>

"¡Hasta la vista!"