Web Development Frameworks

LBAW . Databases and Web Applications
MIEIC, 2018/19 Edition

Sérgio Nunes
DEI, FEUP, U.Porto

Software Frameworks

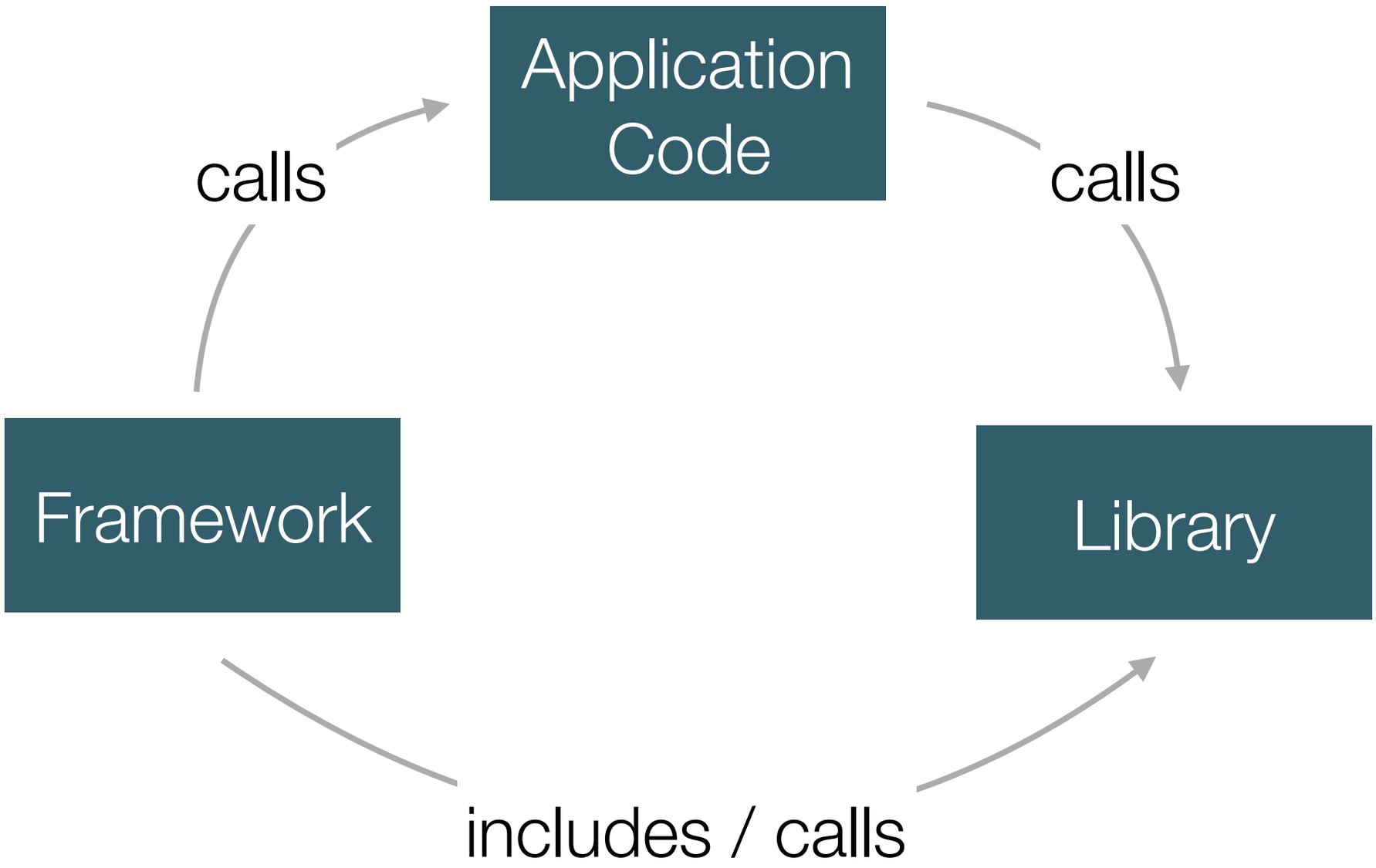
- Software frameworks provide a generic software foundation over which custom application-specific code can be written.

- Software frameworks often include multiple libraries, in addition to tools and rules on how to structure and use these components.

- Libraries are used by the application-specific code to support specific features.

- Frameworks control the application flow and call application-specific code.

Frameworks and Libraries



Why Frameworks

- **Advantages?**
  - Implementation speed
  - Tested, proven solutions
  - Access to expertise and off-the-shelf solutions
  - Maintenance (i.e. updates, patches)

- **Disadvantages?**
  - Reduced independence
  - Lower performance
  - Dependence on external entities
  - Technological lock-in

Choosing Frameworks

- **What to consider?**
  - Team expertise on language, libraries and framework
  - Existing code base
  - Licensing model
  - Maturity
  - Community support
  - …

Web Development Frameworks

- Web development frameworks are designed to support the development of web applications, providing generic and integrated solutions to common use cases.

- These frameworks provide tools, libraries, and rules to address common tasks. Integration is central in contrast to the use of individual libraries.

- Currently, there is a big and diverse ecosystem of libraries and frameworks to support both backend and frontend development.

- Examples of libraries: jQuery, Bootstrap, Twig, PEAR DB.

- Examples of frameworks: ReactJS, Vue.js, Ruby on Rails, Django, Laravel.

History

- In 1993, the Common Gateway Interface (CGI) was designed to enable the communication between browsers and applications, i.e. "programs as web pages". First popular web development libraries were designed to support CGI.

- During the 90s there was a strong development of libraries targeted at common use cases, e.g. outputting HTML (templating), accessing data, interface with mail, managing user input, etc.

- In the early 2000s, the first modern full-stack server-side frameworks for web development started to appear, e.g. Drupal, Ruby on Rails, Symfony, Django.

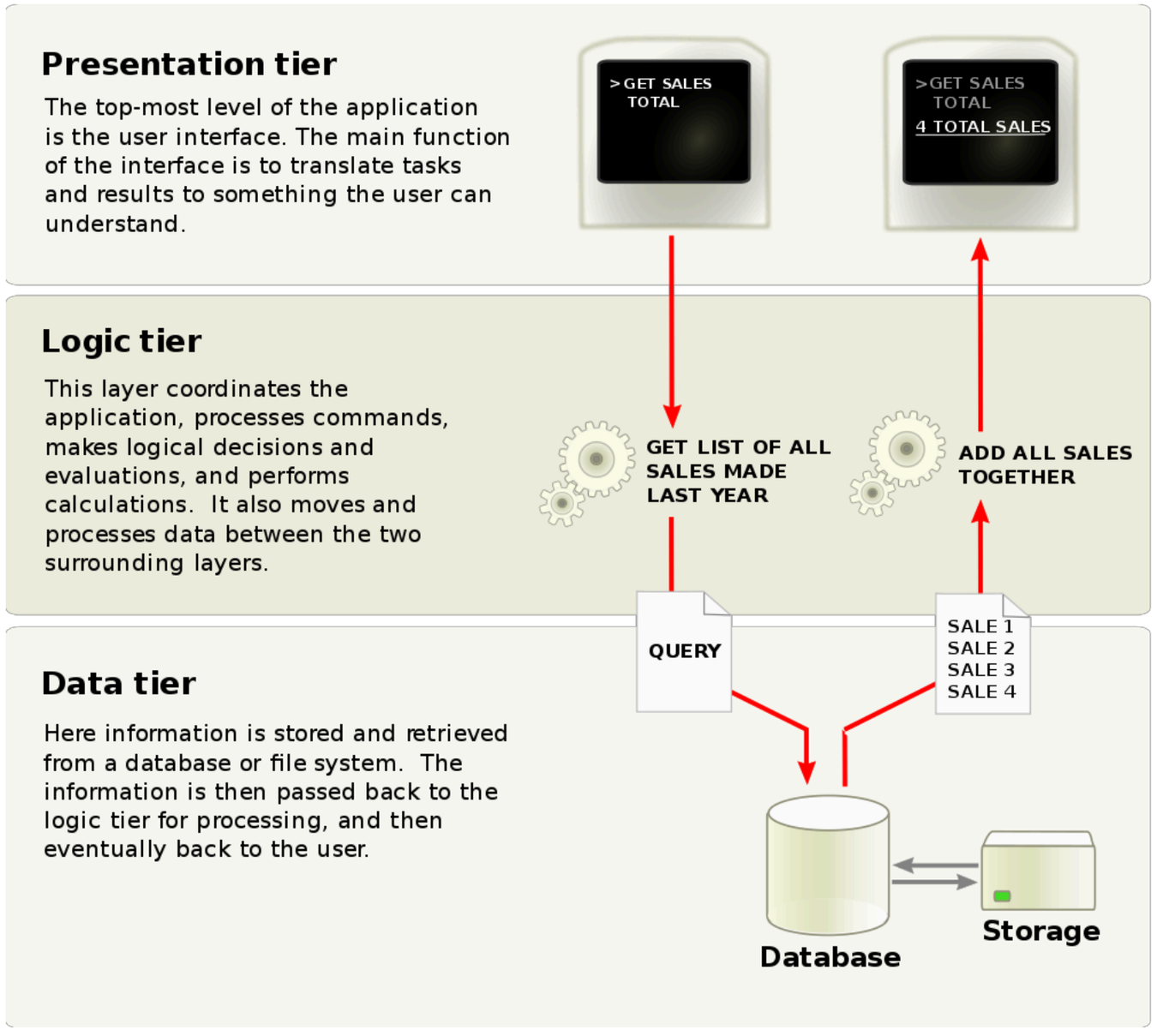
- The growth of client-side supported web applications led to the development of multiple frontend libraries, e.g. jQuery, Mustache.

- More recently, full-stack client-side frameworks for web development emerged, e.g. ReactJS, AngularJS, Vue.js.

Three-tier Architectures

- The Three-tier Architecture is a software architecture pattern commonly adopted in web applications.
    - Presentation tier — interface with the user; process user interactions; present views to the user.
    - Logic tier — coordinate the application; decide on the application flow; process data; move data between the two other layers.
    - Data tier — manage information; persist information; handle consistency; translate between physical models and conceptual model.

Three-tier Architecture



Source: https://en.wikipedia.org/wiki/Multitier_architecture

Common Problems in Web Development

- **Common problems in web applications development?**
  - Handle access requests.
  - Manage interface components.
  - Access and manipulate data.
  - Session and authentication management.
  - Access control management.
  - Validating inputs.
  - Error handling.
  - Interacting with email systems.
  - …

Server-side Frameworks

- Frameworks can be grouped in three types:
  - **Micro Frameworks** —focused on routing HTTP request to a callback, commonly used to implement HTTP APIs.
  - **Full-Stack Frameworks** — feature-full frameworks that includes routing, templating, data access and mapping, plus many more packages.
  - **Component Frameworks** — collections of specialized and single-purpose libraries that can be used together to make a a micro- of full-stack framework.

Framework Components

- Core components
  - Request Routing — Match incoming HTTP requests to code.
  - Template Engine — Structure and separate presentation from logic.
  - Data Access — Uniform data access, mapping and configuration.
- Common components
  - Security — Protection agains common web security attacks.
  - Sessions — Session management and configuration.
  - Error Handling — Capture and manage application-level errors.
  - Scaffolding — Quickly generate CRUD interfaces based on data model.
  - …

Request Routing

- Request routing maps HTTP access requests to specific functions.
- URL design is handled independently from application code using request routing.
- Clean URLs (aka "friendly URLs") are an important part of a web application: usability, seo, technology independence, etc.
- https://sigarra.up.pt/feup/pt/cur_geral.cur_view?pv_curso_id=742
- https://sigarra.up.pt/feup/pt/curso/mieic

Laravel Example

```php
Route::get('user/{id}', function ($id) {
    return 'User '.$id;
});
```

```php
Route::get('posts/{post}/comments/{comment}', function ($postId, $commentId) {
    //
});
```

```php
Route::get('user/{name}', function ($name) {
    //
})->where('name', '[A-Za-z]+');

Route::get('user/{id}', function ($id) {
    //
})->where('id', '[0-9]+');
```

Template Engines

- Many of the first libraries for web development were template engines.
- Focused on the separation between presentation code and logic code.
- There are many independent libraries. Frameworks either use existing libraries or develop their own system.
- Notable solutions: Smarty (PHP), Blade (PHP), Jinja (Python), Mustache (*).

Laravel Example (Blade)

```
<html>
    <head>
        <title>App Name - @yield('title')</title>
    </head>
    <body>
        @section('sidebar')
            This is the master sidebar.
        @show

        <div class="container">
            @yield('content')
        </div>
    </body>
</html>
```

```
@if (count($records) === 1)
    I have one record!
@elseif (count($records) > 1)
    I have multiple records!
@else
    I don't have any records!
@endif
```

```
Route::get('greeting', function () {
    return view('welcome', ['name' => 'Samantha']);
});
```

Data Access

- The data access layer can be managed with different levels of automatism and control.

- Data layer independence can be achieved using libraries that provide a uniform access to different technologies. Example: PHP PDO.

- A higher level of coupling can be achieved by providing access to data through a mapping between the underlying data structures and an object layer (ORM). Example: ActiveRecord (Laravel, RoR).

- This coupling between the data layer and the application’s data model can imply database migrations.

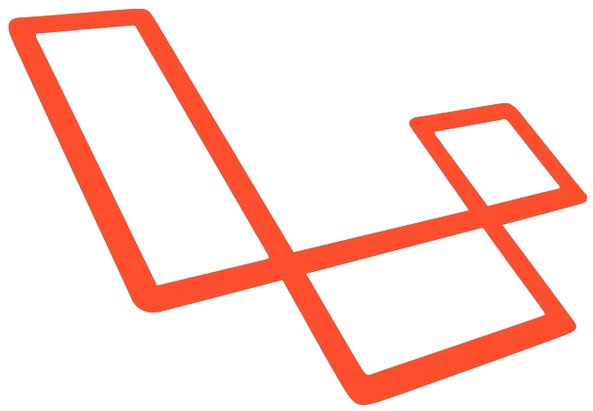
Laravel Example (Eloquent)

```php
$user = DB::table('users')->where('name', 'John')->first();


echo $user->name;
```

```php
$users = DB::table('users')->count();


$price = DB::table('orders')->max('price');
```

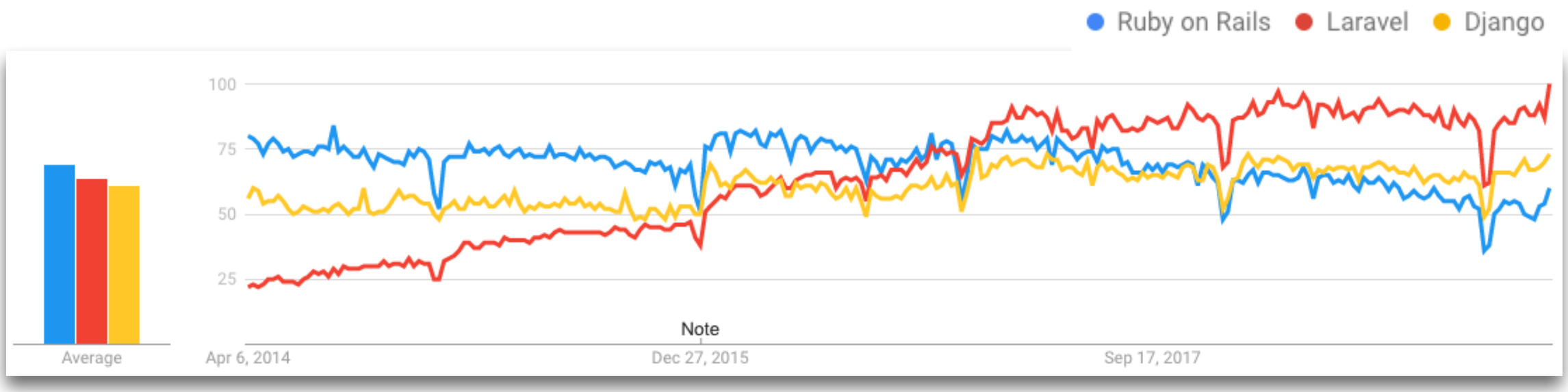
```php
$users = DB::table('users')
                    ->select(DB::raw('count(*) as user_count, status'))
                    ->where('status', '<>', 1)
                    ->groupBy('status')
                    ->get();
```

Notable Web Frameworks



https://en.wikipedia.org/wiki/Comparison_of_web_frameworks

Trends



https://trends.google.com/[...]

Client-side Frameworks

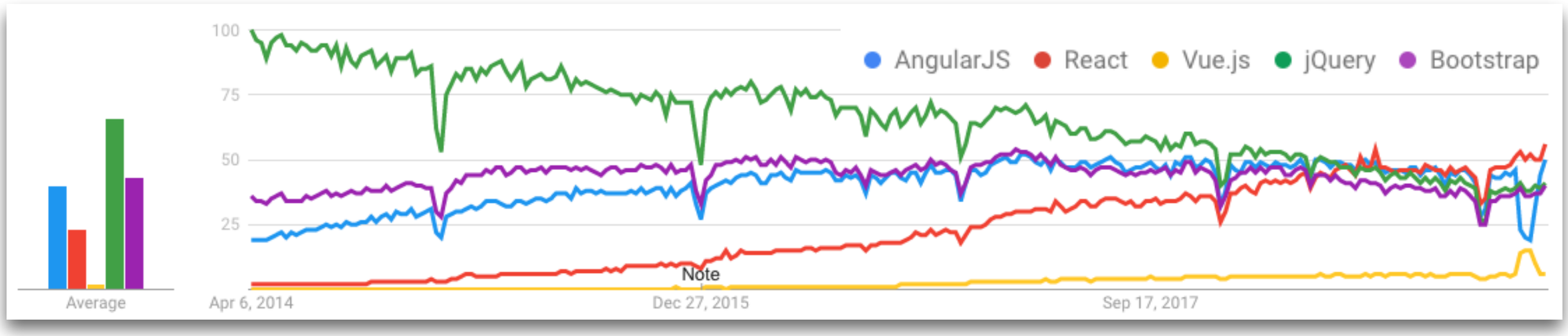
- Client-based web applications, rely on frontend technologies to manage both the Presentation and the Logic layers. In a nutshell, the current web page is dynamically manipulated instead of loading a new page.

- Single-page web applications (SPA) are an extreme example of this architecture. All application views are mapped to a single document.

- Frontend web frameworks typically adopt an MVC pattern (or variants) to support this architectural style, including: data-bindings, templates, routing.

- Most notable solutions: AngularJS (Google), React (Facebook), Vue.js.

Trends



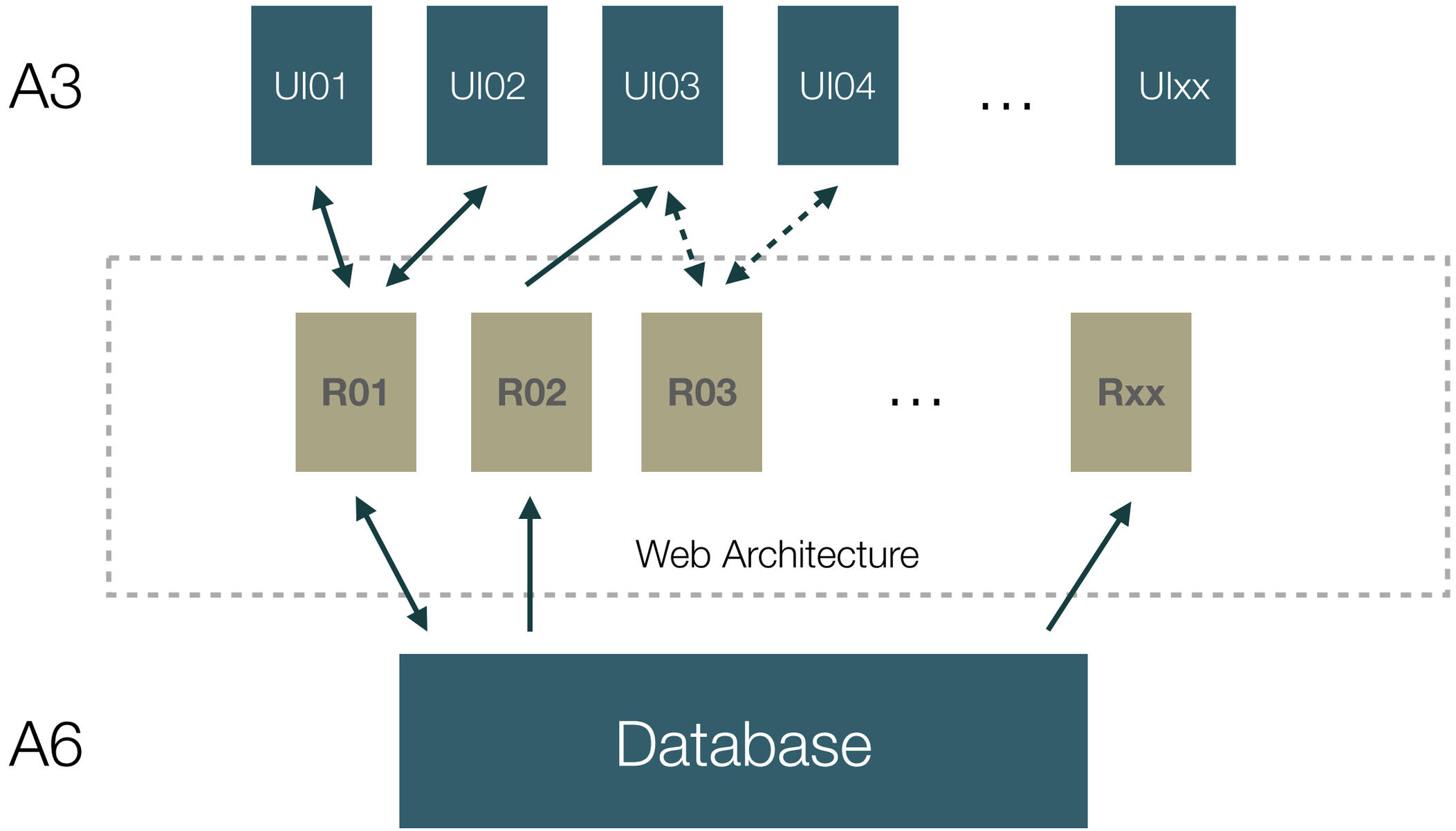
https://trends.google.com/[...]

Other Topics

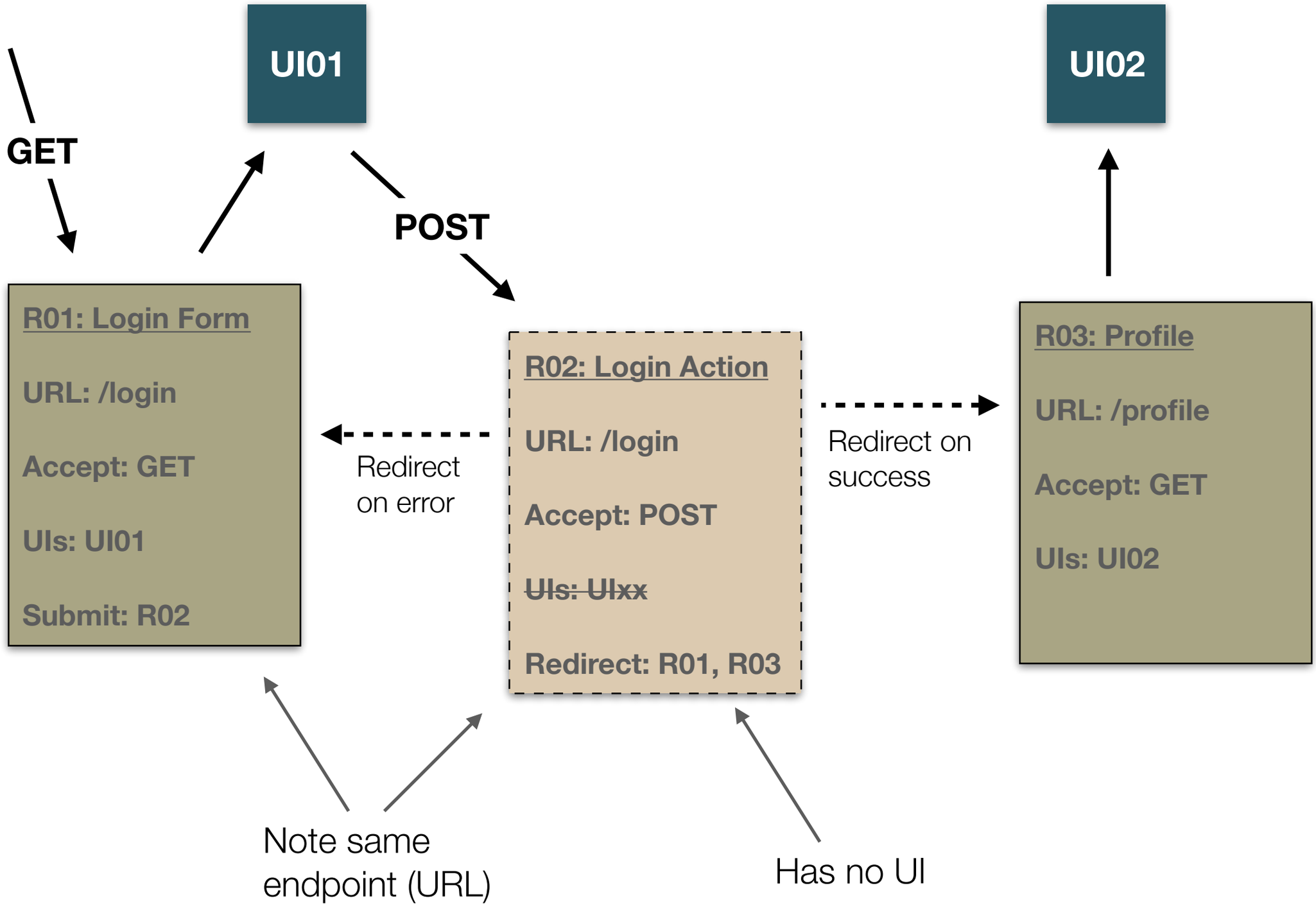
- Content Management Systems (e.g. WordPress)
- Static-site generators
- …

A7: Web Architecture

A7: Web Architecture



A7: View Action Pattern



A7: Web Architecture

- This artefact presents an overview of the web resources to implement, organized into modules. Also included in this artefact are the permissions used in the modules to establish the conditions of access to resources.

- For each module identified in the artefact, a catalogue of web resources that are part of the module, including possible APIs, must be presented. For each resource, the expected URL, HTTP methods and parameters supported as well as possible responses should be included. For resources with a user interface, a reference (link) is included to the corresponding graphical interface defined in A3 artefact. ~~For resources that need SQL, a reference is included to the corresponding SQL statement in A9 artefact.~~

A7: MediaLibrary - Overview

1. Overview

An overview of the web application to implement is presented in this section, where the modules are identified and briefly described. The web resources associated with each module are detailed in the individual documentation of each module.

M01: Authentication and Individual Profile	Web resources associated with user authentication and individual profile management, includes the following system features: login/logout, registration, credential recovery, view and edit personal profile information.
M02: Works	Web resources associated with library items, includes the following system features: items list and search, view and edit item details, and delete items.
M03: Reviews and Wish list	Web resources associated with reviews and wish list, includes the following system features: add review, list reviews and delete reviews; add items to wish list and remove items from the wish list.
M04: Loans	Web resources associated with loans, includes the following system features: view loans, add loans and edit loans.
M05: User Administration and Static pages	Web resources associates with user management, specifically: view and search users, delete or block user accounts, view and change user information, and view system access details for each user. Web resources with static content are associated with this module: dashboard, about, contact, services and faq.

A7: MediaLibrary - Permissions

2. Permissions

This section defines the permissions used in the modules to establish the conditions of access to resources.

PUB	Public	Users without privileges
USR	User	Authenticated users
OWN	Owner	User that are owners of the information (e.g. own profile, own itens)
ADM	Administrator	Administrators

A7: MediaLibrary - Modules

3. Modules

This section documents each web resource of each module, indicating the URL, HTTP methods, the possible parameters (mandatory or optional), body request, interfaces with the user (referring to the A3 artefact), or JSON responses in the event of AJAX call to the API.

Module M01: Authentication and Individual Profile

Endpoints of Module Authentication and Individual Profile

- **R101: Login Form** [`/login`]
- **R102: Login Action** [`/login`]
- **R103: Logout Action** [`/logout`]
- **R104: Register Form** [`/register`]
- **R105: Register Action** [`/register`]
- **R106: View Profile** [`/users/{id}`]
- **R107: Edit Profile Form** [`/users/{id}/edit`]
- **R108: Edit Profile Action** [`/users/{id}`]
- **R109: Password Reset Form** [`/password/reset`]
- **R110: Password Reset Action** [`/password/reset`]

A7: MediaLibrary - Resources

R104: Register Form

URL	`/register`
Description	Page with a form to register a new user account.
Method	GET
UI	UI07
SUBMIT	R105
Permissions	PUB

A7: MediaLibrary - Resources

R105: Register Action

URL	`/register`	
Description	This web resource inserts the user into the system. Redirects to the user profile page on success and the register form on failure.	
Method	POST	
Request Body	+name: string	Name
	+email: string	Email
	+password: string	Password
	?picture: file	Profile picture
Redirects	R106	Success
	R104	Error
Permissions	PUB	

A7: MediaLibrary - JSON

4. JSON/XML Types

The structure of the JSON formatted answers must be documented as illustrated below.

JSON202: Search Works: {work}[]


```
{
  "work": [
    {
      "id": "1",
      "title": "Rihanna - Unapologetic",
      "obs": "Good pop music album.",
      "year": "2012",
      "owner": "Joana Lima",
      "type": "MP3"
    },
    {
      "id": "15",
      "title": "Mr. Bean",
      "obs": "The best comedy movie.",
      "year": "1995",
      "owner": "Manuel Teixeira",
      "type": "DVD"
    }
  ]
}
```