Web Development Frameworks

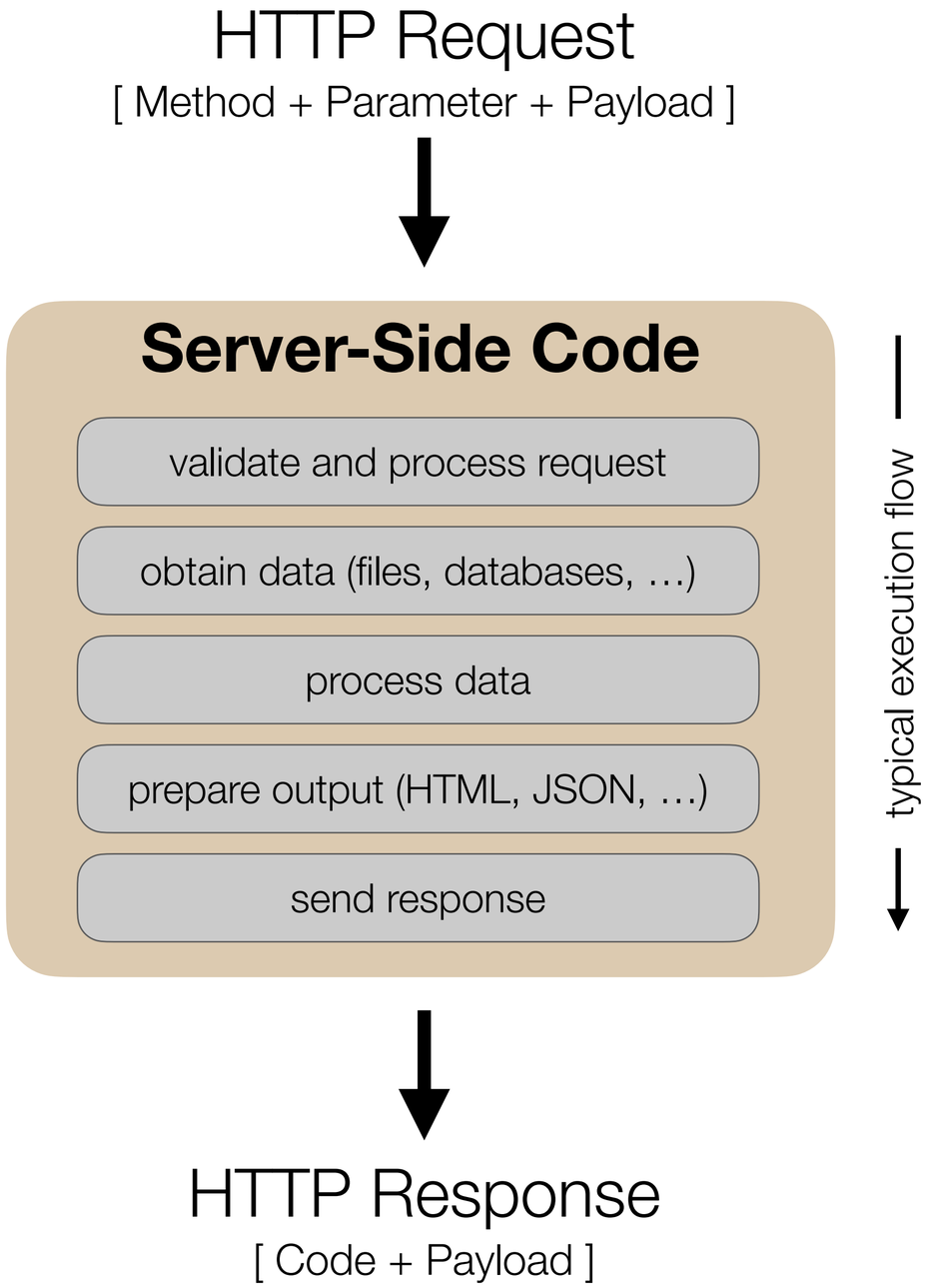
Databases and Web Applications Laboratory (LBAW)
Bachelor in Informatics Engineering and Computation (L.EIC)

Sérgio Nunes
Dept. Informatics Engineering
FEUP · U.Porto

Outline

- Software frameworks
- History of web frameworks
- Typical components in web frameworks
- Server-side frameworks

Server-Side Web Development



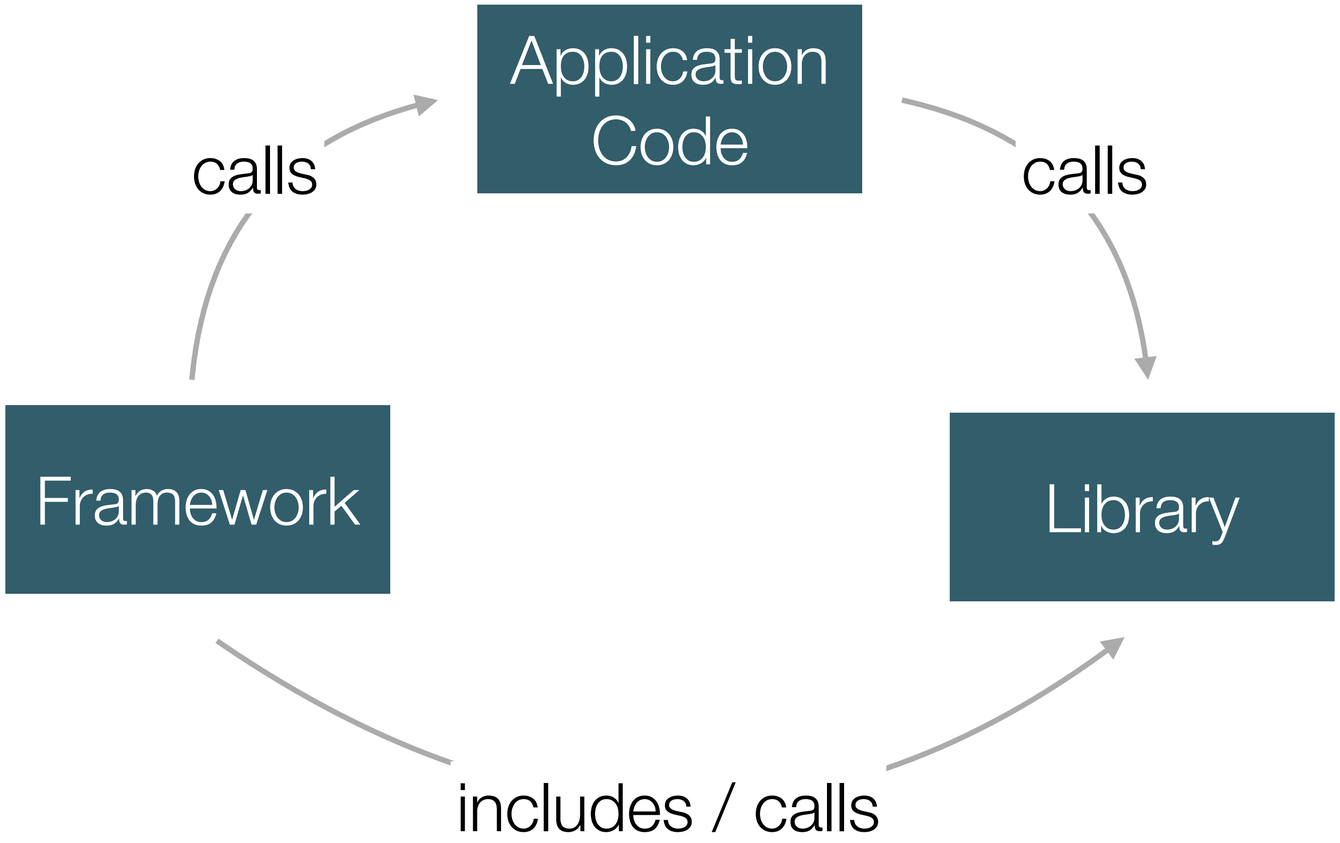
Web Development Frameworks

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.

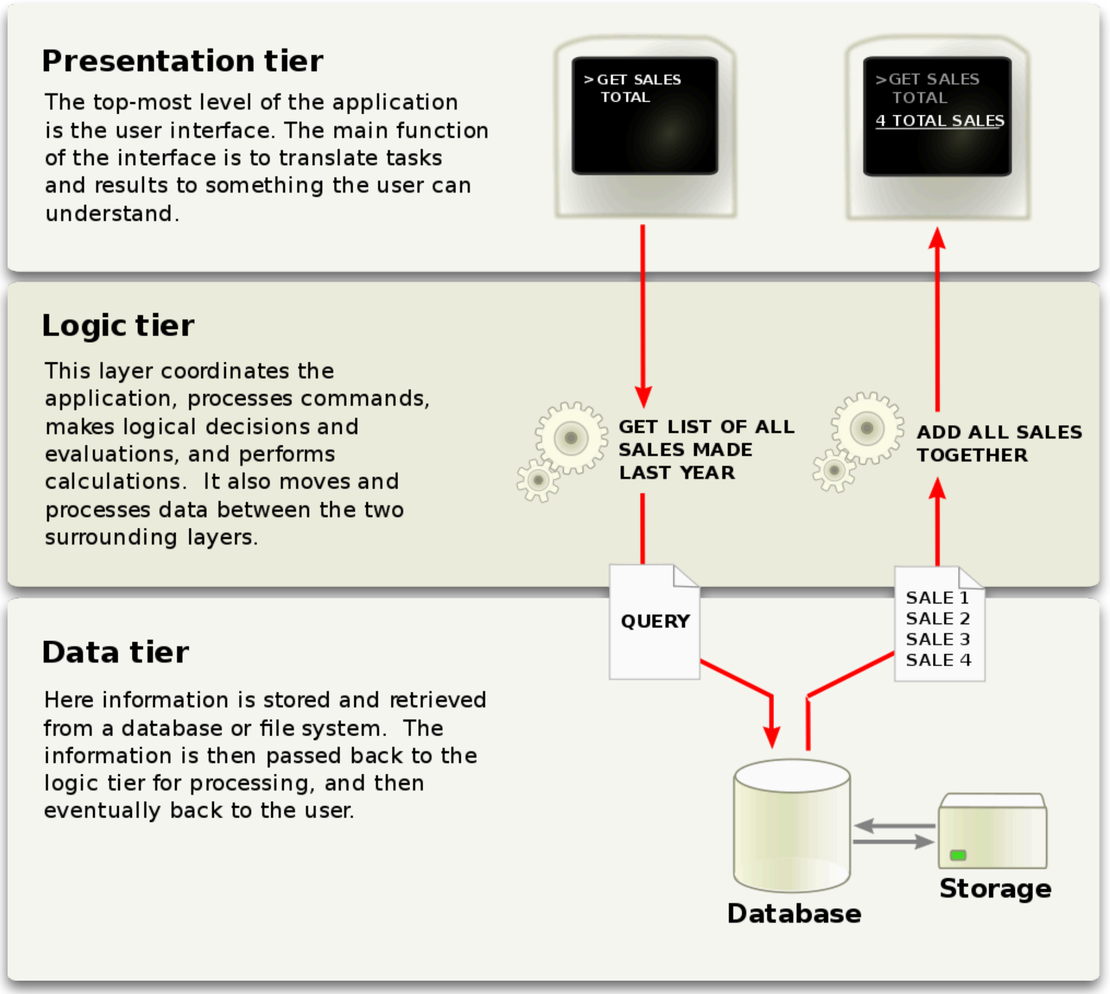
History

- 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.
- 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 user 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 Routing Example

```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]+');

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

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

```php
Route::get('/user/{name?}', function ($name = null) {
    return $name;
});
```

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 Templating 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('/', function () {
    return view('greeting', ['name' => 'Finn']);
});
```

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.

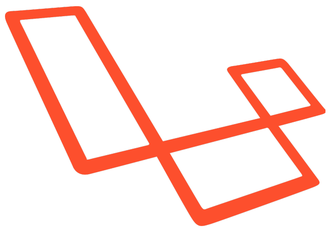
Laravel ORM Example (Eloquent)

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

return $user->email;
```

```php
$count = Flight::where('active', 1)->count();

$max = Flight::where('active', 1)->max('price');
```

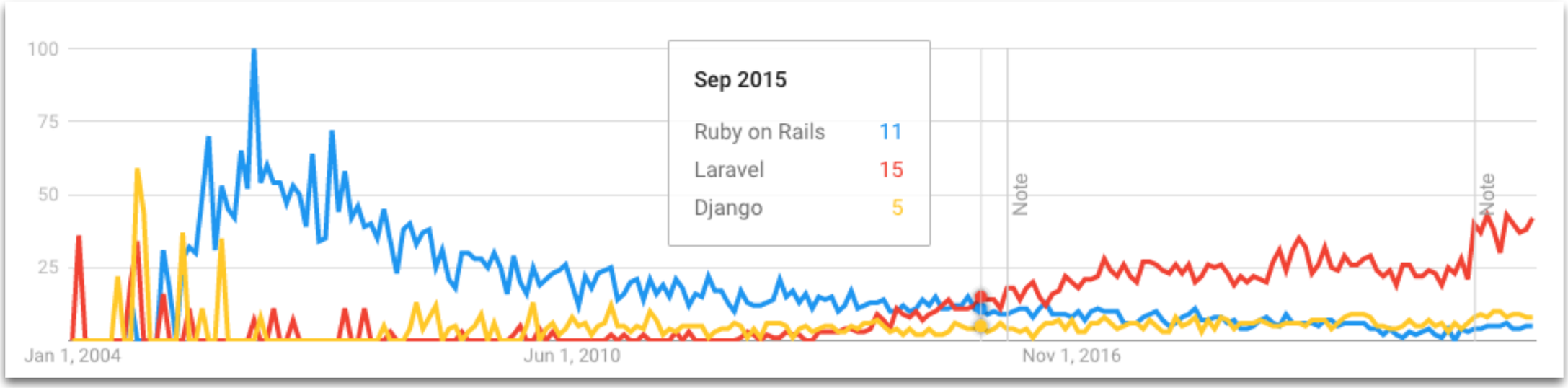
```php
$users = DB::table('users')
            ->select(DB::raw('count(*) as user_count, status'))
            ->where('status', '<>', 1)
            ->groupBy('status')
            ->get();
```

Notable Web Frameworks



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

Trends


100
75
50
25
Jan 1, 2004
Jun 1, 2010
Nov 1, 2016
Note
Note
Sep 2015
Ruby on Rails 11
Laravel 15
Django 5

Client-side Frameworks

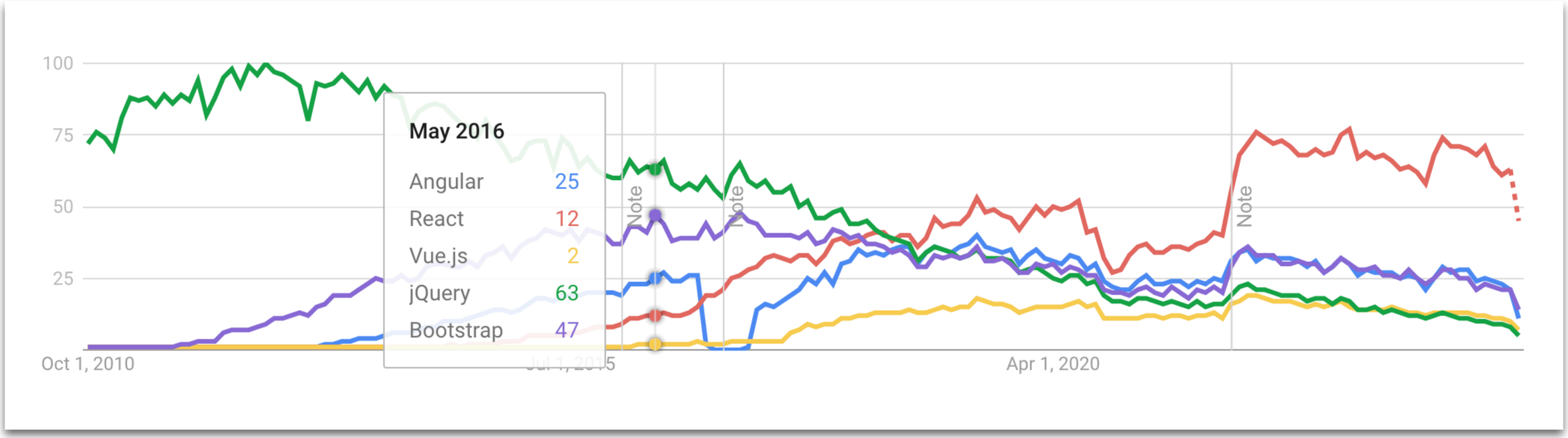
- Client-based web applications, rely on frontend technologies to manage both the Presentation and the Logic layers. In a nutshell, the current web page is dynamically manipulated instead of loading a new page.

- 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), ReactJS (Facebook), Vue.js.

Trends


100
75
50
25
Oct 1, 2010
May 2016
Angular 25
React 12
Vue.js 2
jQuery 63
Bootstrap 47
Note
Note
Note
Apr 1, 2020

Laravel

Laravel Overview

- A free open-source PHP backend web framework first released in 2011.
- Version 11.x is the current stable version (Feb, 2024).
  - We will be **using Laravel 10 in LBAW**.
- Follows the model-view-controller architectural pattern.
- Strong community and documentation.
  - laravel.com
  - laracasts.com

Laravel Concepts

- **Routes**, define how requests for URLs are handled.
- **Models**, using Eloquent ORM, tables are mapped to models supporting insert, update and delete data.
- **Views**, using Blade template engine, views are defined for presentation.

- **Artisan** is Laravel's command line interface (CLI),
  - E.g., create new application, create model skeleton, serve application, etc.

Laravel in LBAW

- template-laravel includes a sample app - Thingy!
  - https://git.fe.up.pt/lbaw/template-laravel
- A8: Quickstart guide describes how to go from,
  - wireframes (A3) +
  - database (A6) +
  - web architecture (A7),
  - to a working vertical prototype.

Quickstart guide workflow

- Model > Controllers > Routes > Blade

- Start by creating the models to interact with the database.
  - Typically one model per each table.
- Define the associations between models.
- Implement the controllers logic (application logic, validation, authorization, etc).
- Define the web resources endpoints.
- Implement the HTML views using Blade.

Other Topics

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

References

- MDN, Server-side web frameworks, developer.mozilla.org/docs/Learn/Server-side/First_steps/Web_frameworks
- MDN, Understanding client-side JavaScript frameworks, developer.mozilla.org/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks