Practical test

UpWork - Software developer

Table of contents

Introduction

Thank you for taking the time to complete our practical test. This example project is a simplified order processing application, focusing on using the DataTables library (see links at the end of the project description).

Requirements

Please make sure the code follows the PSR-2 coding style and the Laravel naming conventions: GitHub - alexeymezenin/laravel-best-practices: Laravel best practices
Please use descriptive class, variable, method, etc. names, do not shorten the names. Comment your code as necessary.
Don’t worry about the design, it will not matter when assessing your test.

Application preparation

1. Create a fresh Laravel application
2. Install the necessary php packages: “yajra/laravel-datatables-oracle”, “silber/bouncer v1.0.0-rc.5”, “spatie/laravel-activitylog”
3. Install the necessary javascript libraries: datatables.net-bs4
4. Create the following models and migration files. Use the data type you think it’s best for the purpose.
    - users (already exist in the default Laravel installation): id, name, email
    - customers: id, name, email, timestamps
    - products: id, name, price, in_stock (indicates whether the product is in stock or out of stock), timestamps
    - orders: id, invoice_number, total_amount, status (can be “new” or “processed”), timestamps

    - order_items: id, order_id, product_id, quantity
5. Create the following database seeders (use fake data):
    - Create three users (users who have access to the application, not customers). Use the `silber/bouncer` package to assign these roles to the user:
        - administrator has access to every page
        - user-manager: has access to users only
        - shop-manager: has access to products and orders only
    - Create 200 customers
    - Create 100 products
    - Create 50 orders (status either “new” or “processed”) with at least one item

Pages

Use DataTables on all pages to display the data in a table-like format. Data must be provided via ajax requests.

Customers

- “user-manager” permission required to access this page
- show list of users (name, email and registered date - `created_at` in UK date format)
- ability to search for customer name or email (default search input for DataTables)
- ability to order the list by customer name

Products

- “shop-manager” permission required to access this page
- show list of products (name, price, in_stock)
- ability to search for the product’s name (default search input for DataTables)
- ability to toggle the visibility of products that are in stock, out of stock or both

Orders

- “shop-manager” permission required to access this page
- show list of orders (customer name, total amount, status (“new” or “processed”))
- ability to order the list by order date
- ability to search for the customer’s name (default search input for DataTables)
- add ability to view the order details: either clicking on the row or adding a new column with a button
    - when the button is clicked, open a new tab showing the customer’s name and the products they ordered (don’t worry about the looks, it’s just the data that matters)
    - log this event using the activitylog package. The activitylog entry should look like this: “XY (logged in user’s name) processed the order: [insert order_id here]”

Final steps

Upload the source code to GitHub or BitButcket (or any other service you may prefer) so we can take a good look. :)

Resources

- GitHub - fzaninotto/Faker: Faker is a PHP library that generates fake data for you
- GitHub - spatie/laravel-activitylog: Log activity inside your Laravel app
- GitHub - JosephSilber/bouncer: Eloquent roles and abilities.
- DataTables | Table plug-in for jQuery
- Laravel Datatables
- GitHub - yajra/laravel-datatables: jQuery DataTables API for Laravel 4|5

Examples

Example controller method for responding to DataTables ajax requests:

```
/**
 * Handling DataTables ajax requests
 *
 * @return \Yajra\DataTables\DataTables
 */
public function datatables()
{
    $users = User::select(["name", "email", "date_of_birth"]);

    return datatables($users)
        ->editColumn('date_of_birth', function ($user) {
            return $user->date_of_birth->format('d/m/Y');
        })
        ->make(true);
}
```

Example javascript to display a table using DataTables:

```
$(document).ready(function() {
    var usersTable = $('.datatable').DataTable({
        ajax: "{{ route('users.datatables') }}",
        columns: [
            {data: 'name'},
            {data: 'email'},
            {data: 'date_of_birth', searchable: false}
        ]
    });

    $('#DataTables_Table_0_filter label input').focus();
});
```