DevCode

Proyecto web PHP mediante el framework Laravel.

Inicio

Proyecto web `devCode`

- Basado en PHP.
- Uso del framework `Laravel`.

Usuarios de prueba:

- user@example.org - secret
- admin@example.org - secret

Descripción

Este proyecto es la continuación de la primera parte de las prácticas de **Ingeniería Web** del curso 2017-2018. Como recordatorio lo que se pretendía conseguir era llevar a cabo un proceso de ingeniería inversa sobre un sitio web en el que básicamente se pretende conseguir:

- Seleccionar un subconjunto de funcionalidades de la página web lo suficientemente abarcable.
- Describir de manera breve y clara el subconjunto seleccionado en forma de un modelo conceptual WebML.
- La propuesta presentada debe estar compuesta por al menos:
    - Un modelo de datos, que representará las diferentes tablas de datos, los campos de cada tabla y las relaciones entre ellas.
    - Un modelo de hipertexto con al menos una vista pública y otra privada. De hecho este modelo describe una vista del sitio y está compuesto a su vez, por dos modelos, modelo de composición que representa las páginas de un hipertexto y su contenido. Por otra parte también tenemos un modelo de navegación que representa los enlaces entre las páginas y sus elementos de contenido.

Con todo lo anterior y como punto de partida en la segunda parte de la práctica se nos pide que hagamos un diseño del sistema a partir del sitio web escogido, que planifiquemos el desarrollo del proyecto a partir de los modelos y diagramas diseñados en la práctica 1. Todo esto acompañado de su correspondiente implementación utilizando las herramientas, metodologías, patrones y demás tecnología necesaria para el desarrollo de la página web.En resumidas cuentas lo que se nos pide es:

- Diseño del sistema
    - Un Wireframe que nos proporcioné una visión general del sistema por perfiles de usuario,
    - Mockups detallados de las pantallas principales.
    - Otros diagramas que nos puedan ayudar.
- Planificación
    - Dividir el proyecto en iteraciones o establecer alguna planificación temporal adecuada.
    - Seleccionar una metodología de desarrollo.
    - Documentar todo el proceso.
- Requisitos mínimos de implementación
    - Registro de usuarios
    - Inicio y cierre de sesión, control de seguridad.
    - Web de back-office
    - Página principal y mínimo dos niveles de navegación completa con las operaciones necesarias implementadas.

Descripción de la web

El sistema escogido se llama DevCode https://devcode.la/ y es una plataforma online que permitirá a cualquier interesado en la programación y desarrollo web a recibir nociones fundamentales que le permitirán iniciarse en el mundo de los desarrolladores.

Devcode oferta una gran selección de cursos (HTML, Javascript, PHP, bootsrap, etc), blogs, tutoriales. Cada uno de los cuales cuenta con temario y ejercicos para poner en práctica los conocimientos adquiridos a lo largo del curso.Los temas están compuesto por videos y recursos externos para dar soporte al contenido.Los cursos finalizan con un proyecto en donde poner en práctica todo lo aprendido. Además los cursos incluyen exámenes (uno por curso).

Para adquirir alguno de los curso solo tenemos que inscribirnos creando una cuenta gratuita, siendo posible acceder a los vídeos que componen cada uno de los cursos. El objetivo es ofrecer una plataforma con vídeos interactivos que muestre también la posibilidad de realizar proyectos y permita desarrollar ejercicios, teniendo el código disponible para entender mejor el tema que se está tratando.

Para acceder a todo el contenido, Devcode ofrece tres tipos de suscripciones, plan anual y plan mensual. En ambos podemos acceder a contenido premium.

Descripción tecnológica

Laravel

Laravel es un framework de código abierto para desarrollar aplicaciones y servicios web con PHP 5 y PHP 7. El objetivo de Laravel es el de ser un framework que permita el uso de una sintaxis refinada y expresiva para crear código de forma sencilla, evitando el “código espagueti” y permitiendo multitud de funcionalidades. Aprovecha todo lo bueno de otros frameworks y utiliza las características de las últimas versiones de PHP.La mayor parte de su estructura está formada por dependencias, especialmente de Symfony, lo que implica que el desarrollo de Laravel dependa también del desarrollo de sus dependencias.Alguna de sus características:

- Sistema de ruteo, también RESTful
- Blade, Motor de plantillas
- Eloquent ORM
- Basado en Composer
- Soporte para el caché
- Soporte para MVC
- Usa componentes de Symfony

Instalación

Inicialmentes empezamos instalando la herramienta **composer** para la administración de dependencias en PHP. Le permite declarar las bibliotecas de las que depende el proyecto y las administrará.Laravel utiliza Composer para administrar las dependencias. Es por esto que que debemos asegurarnos tenerlo instalado antes de usar Laravel.

```
composer global require "laravel/installer"
```

Una vez instalado laravel podemos empezar a trabajar en nuestro proyecto. Con el siguiente comando crearemos una nueva instalación de laravel en el directorio especificado y que contendrá todas las dependencias necesarias.

```
laravel new DevCode
```

También podemos crear el proyecto con el siguiente comando.

```
composer create-project --prefer-dist laravel/laravel DevCode
```

Para desplegar el proyecto podemos usar el servidor de desarrollo incorporado en PHP con

```
php artisan serve
```

Alternativamente podemos instalar otro servidor a través de la herramienta xampp.

XAMPP es un entorno de desarrollo del lenguaje PHP que incluye un servidor web. Ofrece todo lo que necesitas para crear y publicar una página web: un servidor web Apache, la base de datos MariaDB con soporte MySQL, software de desarrollo PHP, y soporte de Perl.

Bootstrap

Bootstrap es un framework basado en HTML, CSS y JavaScript para crear webs responsive, es decir, 100% adaptables a todo tipo de dispositivos móviles.Es una herramienta de código abierto y que gracias a su diseño de 12 columnas conseguimos adaptar nuestro contenido a los distintos tamaños de pantalla. Además soporta las versiones de HTML y CSS, posee un código limpio y optimizado para que tu web cargue lo más rápido posible. Y es compatible con los navegadores más populares como safari, chrome, firefox, explorer y opera.

Ventajas de Laravel

Trabajando con laravel como framework de base en el desarrollo de los proyectos web obtenemos :

- Reducción de costos y tiempos en el desarrollo y mantenimiento.
- Curva de aprendizaje relativamente Baja (en comparación con otros framework Php).
- Flexible y adaptable no solo al MVC Tradicional (Modelo vista controlador) sino que para reducir código propone usar "Routes with clousures"
- Buena y abundante documentación sobre todo en el sitio oficial y otros foros.
- Es modular y con una amplio sistemas de paquetes y drivers con el que se puede extender la funcionalidad de forma fácil, robusta y segura.
- Hace que el manejo de los datos en Laravel no sea complejo; mediante Eloquent (que es un ORM basado en el patrón active record) la interacción con las bases de datos es totalmente orientada a objetos, siendo compatible con la gran mayoría de las bases de datos del mercado actual y facilitando la migración de nuestros datos de una forma fácil y segura. Otro punto es que permite la creación de consultas robustas y complejas.
- Facilita el manejo de ruteo de nuestra aplicación como así también la generación de url amigables y control de enlaces auto–actualizables lo que hace mas fácil el mantenimiento de un sitio web.
- El sistema de plantillas **Blade de Laravel**, trae consigo la generación de mejoras en la parte de presentación de la aplicación como la generación de plantillas más simples y limpias en el código y además incluye un sistema de cache que las hace más rápidas, lo que mejora el rendimiento de la aplicación.
- También cuenta con una herramienta de interfaces de líneas de comando llamada Artisan que me permite programar tareas programadas como por ejemplo ejecutar migraciones, pruebas programadas, etc.

Problemas encontrados

Peticiones POST

Surge un problema a la hora de hacer una petición POST. Por seguridad, Laravel indica que debe indicarse como cabecera un token especial llamado `CSRF Protection` como se indica en su wiki

Para ello, debe añadirse como HEADER de la web:

```
// File: /resources/views/layout.blade.php
<meta name="csrf-token" content="{{ csrf_token() }}" />
```

Una vez se tiene dicho token se podrá utilizar en una petición AJAX mediante JQUERY:

```
headers: {
            'X-CSRF-TOKEN': $('meta[name="csrf-token"]').attr('content')
        },
```

En caso se querer probar la API desde herramientas externas, como por ejemplo, POSTMan debe eliminarse el control de dicho token en laravel, para ello, se debe comentar la siguiente línea:

```
// File: /app/Http/Kernel.php
            \App\Http\Middleware\VerifyCsrfToken::class,
```

Permitiendo una llamada POST desde POSTMan si es requerido.

Clases no encontradas

Si se descarga una rama o se utilizan cambios o nuevas clases de otras ramas e indica que alguna clase no existe, se debe ejecutar:

```
$ php artisan optimize
```

Rutas no válidas

Dependiendo de la máquina donde se ejecute y su configuración a veces existen problemas con las rutas indicando:

```
Not Found

The requested URL /devCode/public/cursos was not found on this server.
```

Para arreglarlo, se deberá utilizar en la URL el parámetro index.php:

```
http://127.0.0.1/devCode/public/index.php/cursos/
```

Columna updated_at

Por defecto, laravel añade dicha columna a las tablas, si no ha sido añadida, en el modelo se debe añadir:

```
public $timestamps = false;
```

Asociaciones oneToMany

Para asignar una variable en la clase Many-To-One, en este caso, por ejemplo, tutorial, se debe asociar un objeto del tipo de la relación como se indica a continuación:

```
$tutorial->author()->associate(Author::find(9));
```

Puesta en marcha

El proyecto una vez descargado se deberán seguir los siguientes pasos:

- Ser copiado en la carpeta donde se situe el

  ```
  localhost
  ```

  de la máquina que lo ejecute.
  - Para este primer paso tan solo es necesario copiar la carpeta `/public`, por tanto, si se crea un `link` a dicha carpeta en la carpeta donde se situe el servidor `localhost` será suficiente.
- Ejecutar:

```
$ composer install
```

- Si aparecen los siguientes errores:
  - `Problema al abrir laravel.log`: Ejecutar: `$ sudo chmod -R 777 ./storage`
- Generar la Key utilizada:

```
$ php artisan key:generate
```

- Configurar la base de datos como indica en el siguiente enlace.
- Asegurarnos que la herramienta composer está instalado. Es esencial dado que un proyecto que depende de ciertas librerías desarrolladas por terceros, y a su vez, éstas librerías también dependen de otras, lo que hace Composer en este caso es averiguar que librerías deben instalarse; es decir, resuelve todas las dependencias indirectas y descarga automáticamente la versión correcta de cada paquete.
    - Normalmente estará ubicada en la carpeta `/vendor` dentro del directorio del proyecto.

Y ya está la web funcionado!

http://127.0.0.1/devCode/public/index.php

Usuarios de prueba:

- user@example.org - secret
- admin@example.org - secret

Ejemplos de ejecución

Ejemplos de vistas:

DevCode Cursos Tutoriales TV Perfil ¡Sé premium!

Usuario Prueba

Email: user@example.org

Registrado el: 01-01-1970

Cerrar sesión

Mis Cursos

Curso de MongoDB

Aprende a trabajar con MongoDB, la base de datos NoSQL más popular del mundo.

Ir al curso

Curso de React Router 4

Aprende React Router y facilita la escritura de aplicaciones webs SPA con React.

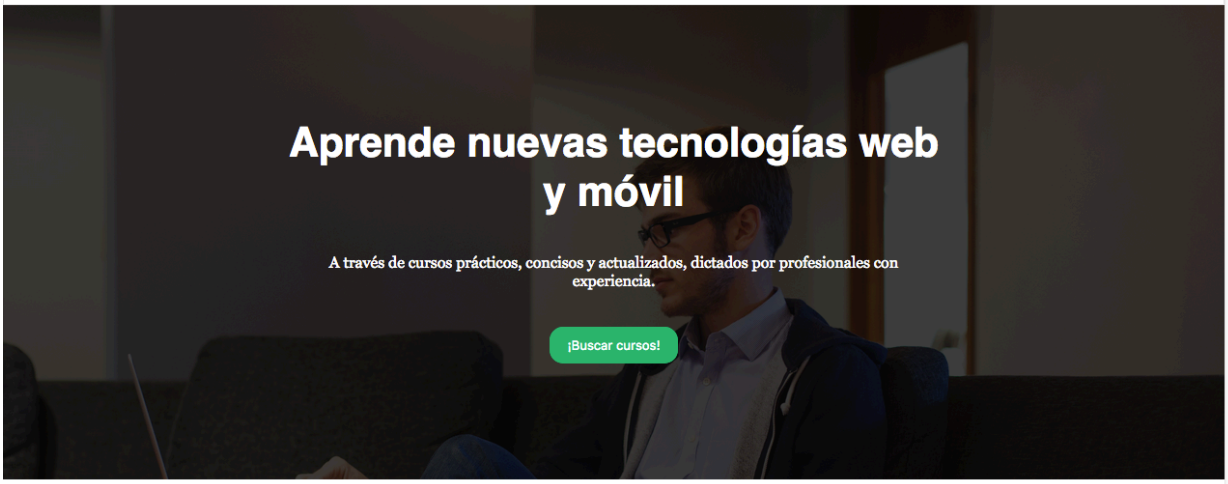
Ir al curso

Fundamentos de Laravel

Aprende a desarrollar en el framework de PHP de mayor crecimiento en la actualidad.

Ir al curso

DevCode Cursos Tutoriales TV Admin Perfil ¡Sé premium!



Curso de MongoDB

Aprende a trabajar con MongoDB, la base de datos NoSQL más popular del mundo.

Curso de React Router 4

Aprende React Router y facilita la escritura de aplicaciones webs SPA con React.

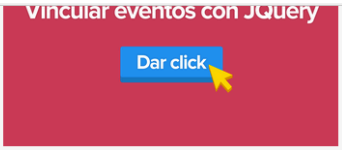
Fundamentos de Laravel

Aprende a desarrollar en el framework de PHP de mayor crecimiento en la actualidad.

Caché de lado del servidor con Express en NodeJS

Aprende la técnica de almacenar datos en memoria para mejorar el rendimiento de cualquier aplicación, móvil, web o de escritorio.

Julio Giampiere Grados Caballero



Vincular eventos a elementos con jQuery

Aprende a asociar eventos a tus elementos en tus aplicaciones y sitios web a través de jQuery.

Julio Giampiere Grados Caballero

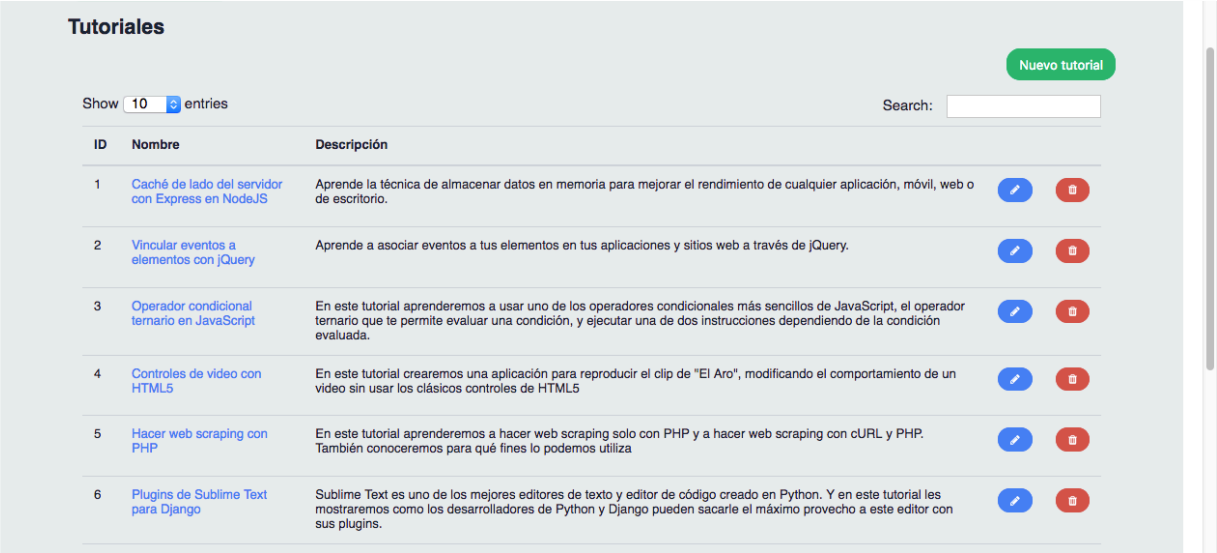
Operador condicional ternario en JavaScript

En este tutorial aprenderemos a usar uno de los operadores condicionales más sencillos de JavaScript, el operador ternario que te permite evaluar una condición, y ejecutar una de dos instrucciones dependiendo de la condición evaluada.

Julio Giampiere Grados Caballero

127.0.0.1/devCode/public/index.php/tutoriales/1

Administrar tutoriales



Posibles mejoras

Algunas mejoras a nivel funcional:

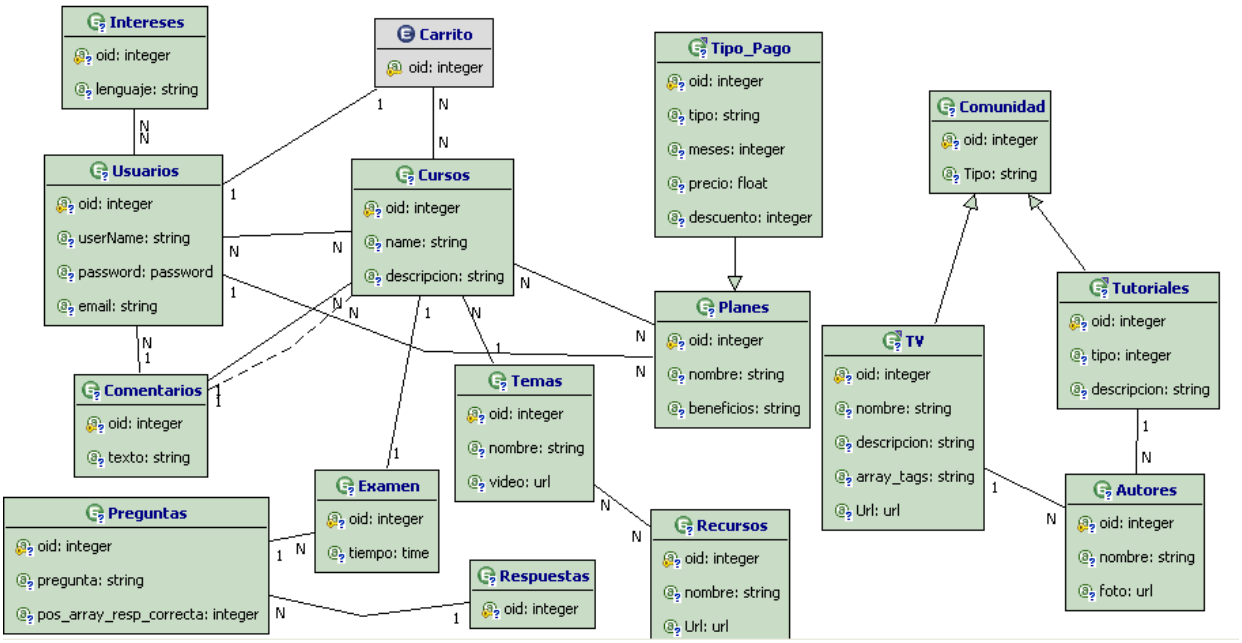
- Mejorar el catálogo de de cursos añadiendo una mayor variedad de los mismos
- Proporcionar contenido a los cursos y blogs (tutoriales y tv) con su respectivos temas, recursos, ejercicios y examen final.
- Dotar de mayor practicidad y funcionalidad al hecho de que los usuarios sean premium.
- Incorporar un sistema de discusiones para que los usuarios puedan compartir opiniones o plantear dudas a cerca de los cursos.
- Proporcionar al administrador de mayor maniobrabilidad para poder gestionar la web como se haría en un sistema real.

En cuanto a nivel de diseño algunas posibles mejoras podrían ser:

- **Singleton** para poder establecer un usuario global en el `Controller` el cuál sea establecido una sola vez.
- **Factory** para poder agrupar los tutoriales y tv en Comunidad ya que poseen demasiados atributos en común.

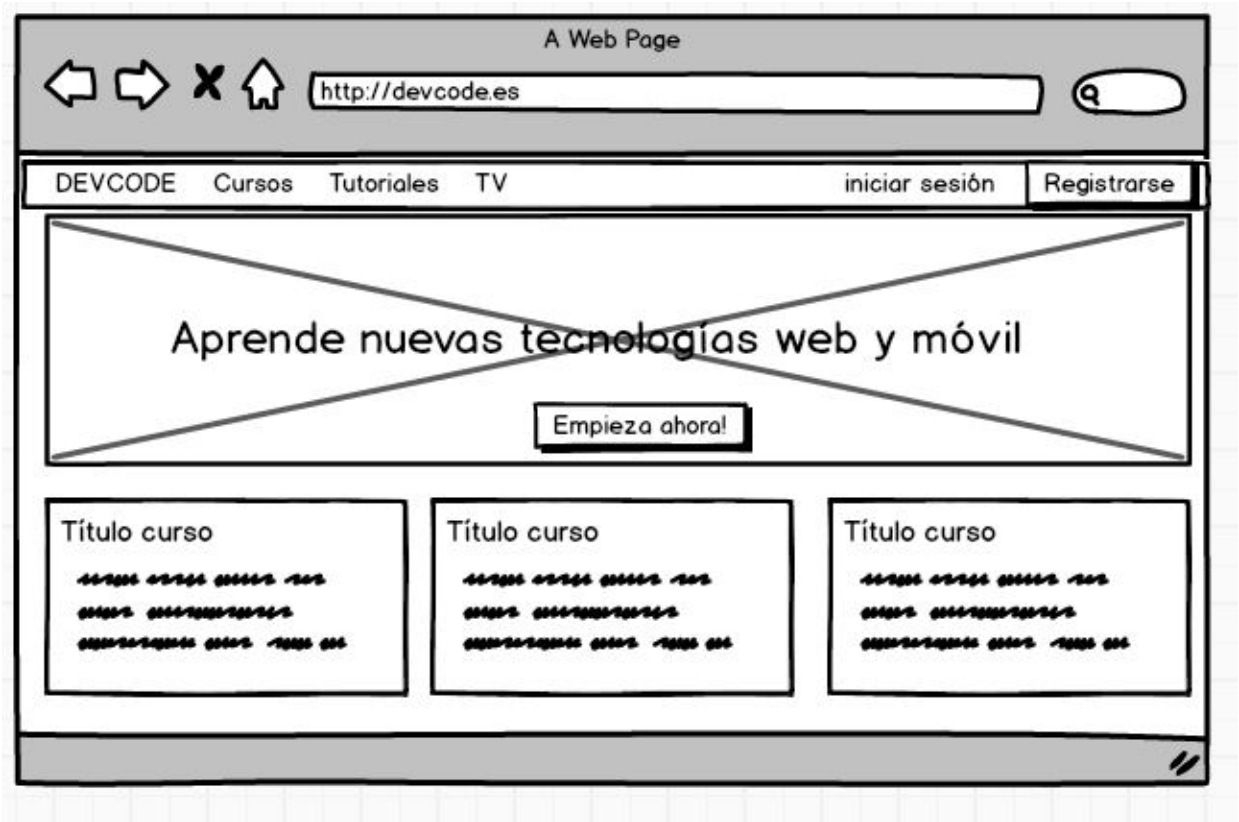
Iteración 1

Diagrama UML

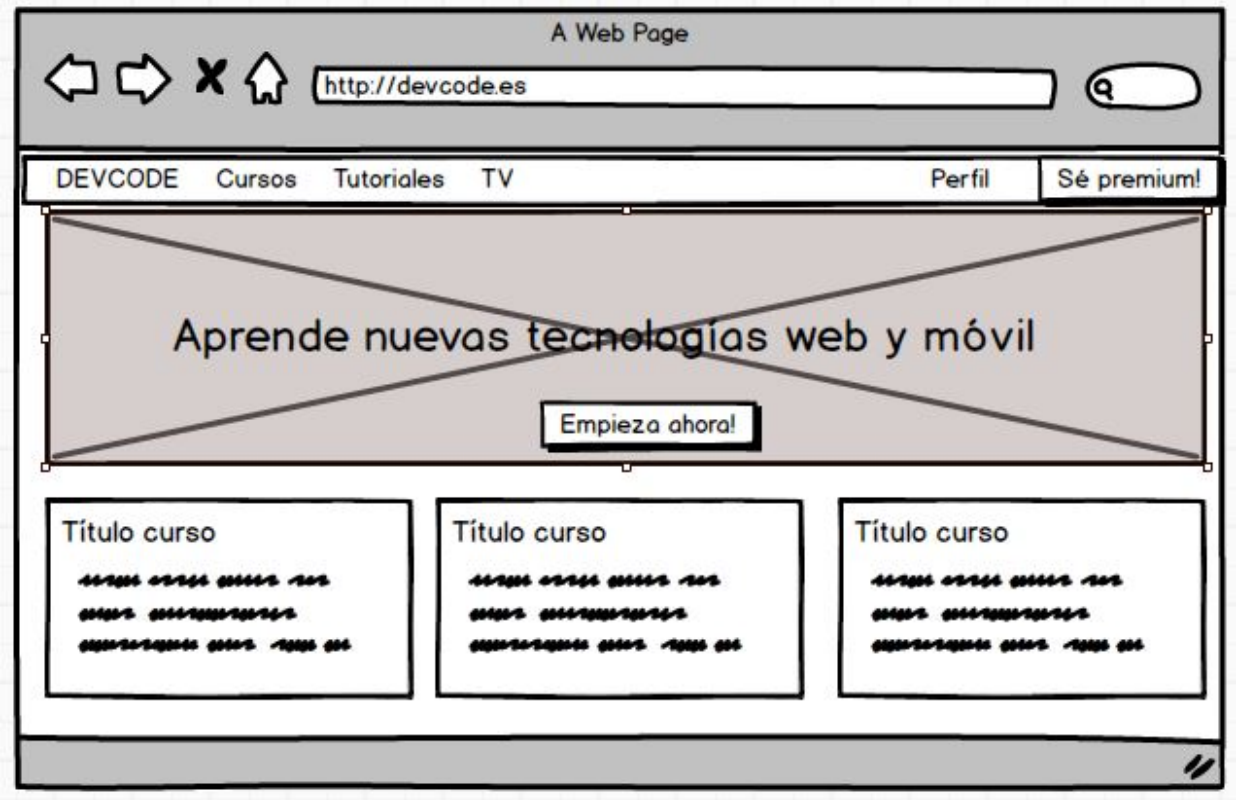


Mockups

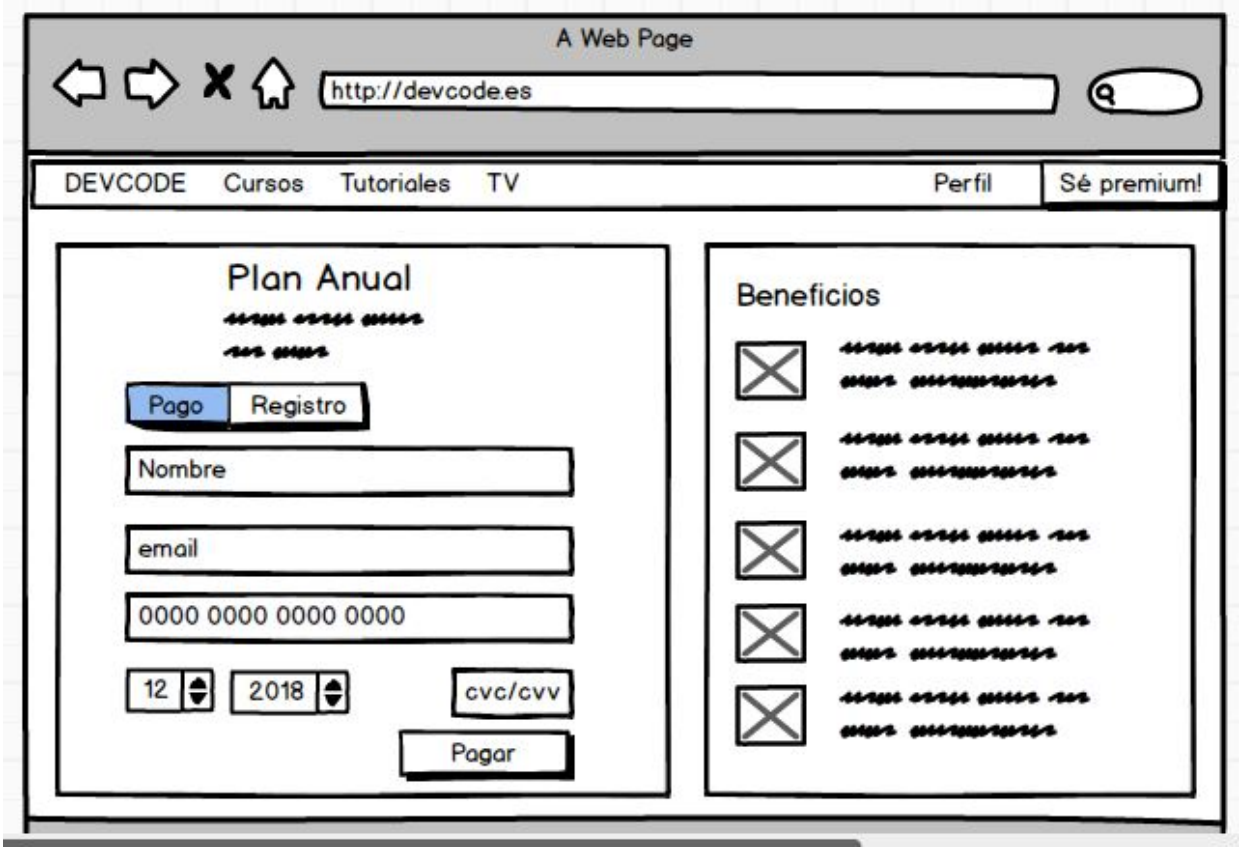
Pagina principal vista pública



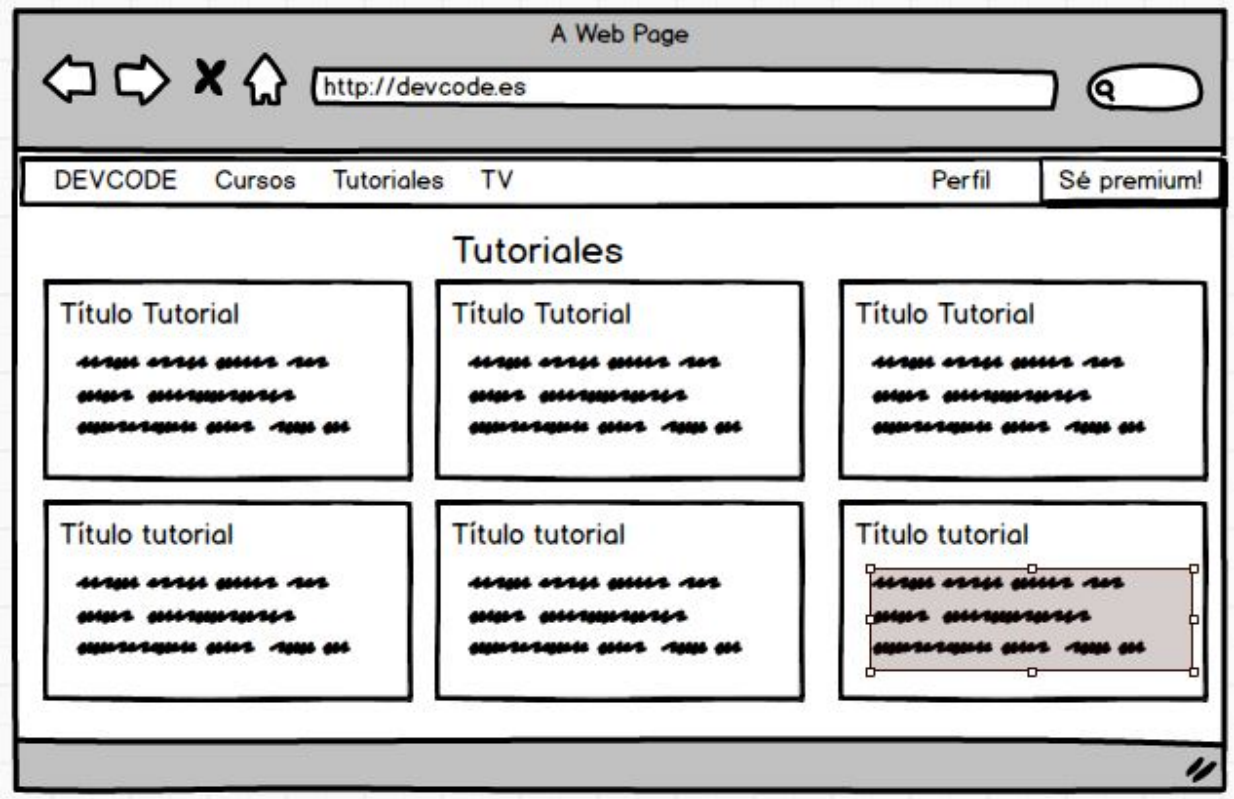
Pagina principal vista privada



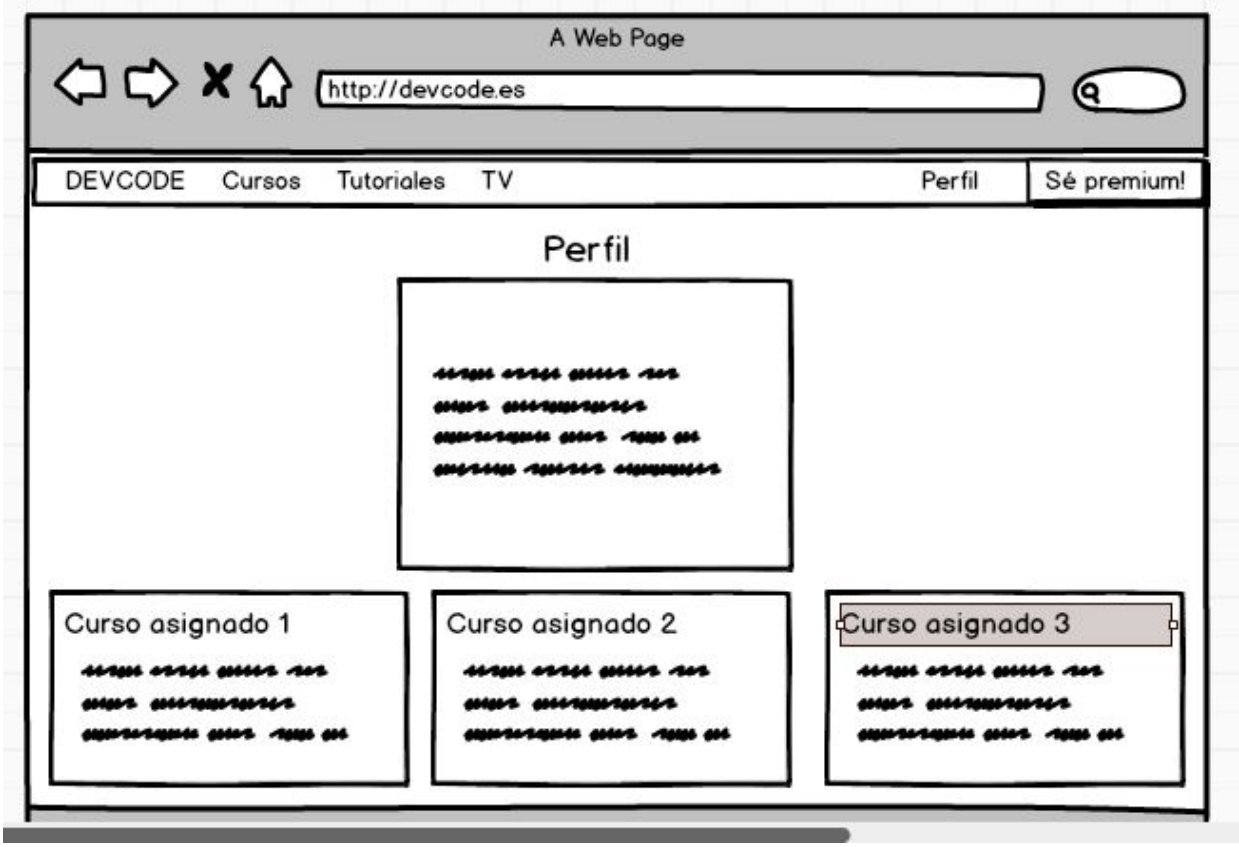
Vista pago



Vista tutoriales



Vista perfil



Estructura del proyecto

El proyecto contendrá la siguiente estructura:

Modelos

en la carpeta `app/Models`:

- Cada modelo utilizará el namespace `App\Models;`.
- Los controladores se basarán en los **modelos**, por tanto, por ejemplo, el método `login()` de `User` se realizará en `UserController`.
- Para el modelo `User` extendido de Authenticatable se deberá cambiar la siguiente línea:

```
// File: /config/auth.php
    'providers' => [
        'users' => [
            'driver' => 'eloquent',
            'model' => App\Models\User::class,
        ],
```

Views

Las vistas se dividirán en:

- Páginas en la carpeta `resources/views/pages`.
    - Vistas de acceso público en: `resources/views/pages/public`.
    - Vistas de acceso de usuario: `resources/views/pages/user`.
    - Vistas de administración: `resources/views/pages/admin`.
- Items en subcarpetas `resources/views/subcarpeta`.
- Vista principal + Vista admin principal en `resources/views/`.

Rutas

- Tendrán como nombre model.metodo.
- Seguirá un esquema APIRest, por tanto, las URL se tratarán como modelos y si se desea un usuario en concreto se utilizará:

```
/users/email/email@email.com
/users/id
/users/id/accionAUsuario
```

Estructura de la web

Estructura parte pública/privada

Las páginas serán imprimidas a través de los controladores mediante el nombre showPage, siendo Page el nombre de la página que se desea mostrar.

Las páginas tendrán siempre la siguiente estructura:

```
@extends('layout')
@section('page')
Cuerpo de la página
@stop
```

Por tanto, todas las páginas heredarán la estructura de `layout`, el cuál se encargará de importar las librerías y ficheros globales del website. En caso se querer añadir un script o estilo CSS en especial se realizará en el cuerpo de la página.

Layout como esqueleto de la web, por tanto, tiene la siguiente estructura:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>DevCode</title>
        <meta name="csrf-token" content="{{ csrf_token() }}" />
        scripts
    </head>
    <body>
        <header>
            contenido Header
        </header>
        <main>
            @yield('page')
        </main>
        <footer class="inner-body">
            contenido Footer
        </footer>
    </body>
</html>
```

Estructura administración

El panel de administración contará con una capa más, la cuál mostrará secciones que engloban todo el panel de administración.

Para ello, las páginas heredarán de `admin` y ya no lo harán de `layout`.

```
@extends('admin')
@section('adminPage')
```

El encargado de heredar de `layout` por tanto, será `admin`, éste como en el punto anterior, poseerá la herencia de `layout` y tendrá la sección `page`.

Por último, `admin` llamará a la sección `adminPage` para mostrar el cuerpo de la página del panel de administración que se encuentre el usuario.

Iteración 2

Base de datos

Instalación

En este caso se ha optado por una base de datos MySQL.

Se debe renombrar el fichero `.env.example` a `.env`.

En windows: `$ rename ".env.example" ".env"`.

En linux: `$ mv ".env.example" ".env"`.

Para la conexión de laravel con la base de datos se debe modificar el archivo `.env` indicando:

```
DB_CONNECTION = mysql
DB_HOST = 127.0.0.1
DB_PORT = 3306
DB_DATABASE = devCode
DB_USERNAME = username
DB_PASSWORD = password
```

Se puede crear una base de datos mediante la secuencia de comandos:

```
$ CREATE USER 'devcode'@'localhost' IDENTIFIED BY 'secret';
$ GRANT ALL PRIVILEGES ON *.* TO 'devcode'@'localhost' WITH GRANT OPTION;
```

Uso de la base de datos

Para crear una tabla en la base de datos se utilizarán las herramientas aportadas por el Framework, en este caso, se aportan varias herramientas concentradas en la carpeta `database` del proyecto.

En este caso, se utilizará una biblioteca de datos ya preparada para poder poner en marcha el proyecto con datos ficticios en la máxima brevedad posible.

Para añadir las tablas a la base de datos, que en este caso se ha optado por una base de datos MySQL se utilizará la herramienta `migrate`

```
$ php artisan migrate
```

Y para añadir datos a las tablas de la base de datos se utilizará la herramienta `db:seed`:

```
$ php artisan db:seed
```

Relaciones entre modelos

Muchos a muchos

Para esta relación se ha decidido crear una tabla auxiliar, la cuál posee 2 claves ajenas que apuntan a la clave primaria de las cada tabla a la qué se hace referencia.

Por tanto, la migración tendrá la siguiente estructura:

```
Nombre de tabla: 'tabla1_tabla2'

$table->integer('tabla1_id')->unsigned()->nullable();
$table->foreign('tabla1_id')->references('id')->on('tabla1')-
>onDelete('cascade');

$table->integer('tabla2_id')->unsigned()->nullable();
$table->foreign('tabla2_id')->references('id')->on('tabla2')-
>onDelete('cascade');
```

Al eliminar la fila de la tabla 1 o 2 se borrará automáticamente en `cascada` las relaciones que posea.

Al ser una relación muchos a muchos, hay que añadir una variable en el modelo que relacione ambas tablas, para ello, se deberá añadir en el modelo las siguientes líneas:

```
// Modelo tabla1
public function tabla2(){
   return $this->belongsToMany('App\Models\Tabla2', 'tabla1_tabla2');
}
```

Por tanto, a la hora de cargar el usuario en el controlador, al haberlo hecho global (user = tabla1):

```
//File: /app/Http/Controllers/Controller.php
$this->user = User::find(Auth::user()->id);
```

Se podrán acceder a sus corriespondientes relaciones con tabla2 (tabla2 = cursos) desde **cualquier** vista.

```
$user->cursos
```

Uno a muchos

Para realizar esta relación, al igual que en la relación muchos a muchos, se debe implentar en la migración:

```
Nombre de tabla: 'tabla1'

$table->integer('tabla2_id')->unsigned();
$table->foreign('tabla2_id')->references('id')->on('tabla2')-
>onDelete('cascade');
```

Además, se añade la clave ajena que apunte a la otra tabla indicando que se borre si es borrado la fila de `tabla2`.

Además en los modelos se debe añadir:

Modelo1:

```
public function datoTabla2(){
        return $this->belongsTo('App\Models\ModeloTabla2');
}
```

Modelo2:

```
public function datosTabla1(){
        return $this->hasMany('App\Models\ModeloTabla1');
}
```

Registro, Login y Logout

Se ha optado por utilizar la herramienta `Auth` proporcionada por laravel. Las contraseñas son cifradas mediante bcrypt y en caso de querer recordar la sesión se utilizará un `remember_token`.

Login

Para el inicio de sesión se pedirá el correo electrónico y contraseña, este formulario realizará una petición `Ajax` mediante `jQuery` desde la misma página.

Se controlarán los siguientes errores:

- Que se envíe una petición POST y no GET.
- Que exista un usuario que posea ambos datos.

El formulario como se ha descrito, se realiza mediante una petición POST utilizando `Ajax`, por tanto, no es necesario recargar la página.

Una vez enviado el formulario mediante una petición JSON, al comprobar que todo sea correcto se realizará:

```
// $request->input('remember-me') = false/true
Auth::attempt($userData, $request->input('remember-me'))
```

Almacenando la sesión del usuario.

Registro

Para crear un usuario son requeridos los siguientes datos:

- Nombre.
- Email.
- Contraseña.

Al igual que en el inicio de sesión, la petición se realizará en la misma página mediante el mismo método.

Se controlarán los siguientes errores:

- Que se envíe una petición POST y no GET.
- Que las contraseñas coincidan.
- Que no exista un usuario en la base de datos con dicho correo.

Al registrarse el controlador creará un usuario con los correspondientes datos, cifrando la contraseña, tras esto, ejecutará:

```
$user = new User();
[...]
$user->save();
```

Para almacenar el usuario en la base de datos.

Logout

Al igual que en los casos anteriores, realiza el mismo tipo de petición mediante `POST`, reenviando al usuario al `Home`.

Para cerrar la sesión el controlador ejecutará:

```
Auth::logout();
```

Peticiones

Las peticiones al servidor se harán mediante JSON, por tanto, si se desea realizar una peticion JQuery habrá que indicar:

```
dataType: 'JSON',
data: jsonAPasar
```

Para ello se formará el JSON mediante:

```
JSON.parse(JSON.stringify({'valName1': valName1Value, 'valName2':
valName2Value}));
```

El controlador recogerá los datos mediante:

```
$request->input('$request->input('email')')
```

Iteración 3

Notificaciones

En cuanto a mensajes que se puedan producir durante la interacción del usuario por el website se ha decidido por abstraer el método que produce dichas notificaciones, los métodos serán:

```
//FILE: /public/js/devCode.js
publicErrorMsg(errorMsg)
publicSuccessMsg(successMsg)
```

Estas funciones podrán llamarse desde cualquier lugar de la web que lo requiera indicando el mensaje. Se ha decidido separar para diferenciar claramente qué tipo de mensaje se desea mostrar al usuario.

Cada página se encargará de mostrar un elemento cuyo id será `notificationMsg` en el formato que corresponda.

Tipos de notificaciones

Se ha decantado por dos tipos:

- Mensajes de error.
- Mensajes indicando que todo ha funcionado correctamente.

Para esto, como los controladores solo tienen permitido devolver 2 variables en formato json: `response` o `error`, tan solo será necesario el control de dichas 2 respuestas.

Primer caso: Se ha producido un error.

En este caso, el controlador se encargará de devolver un json que contiene un error indicando el mensaje de dicho error, tras esto, será enviado al usuario a través de una notificación.

Segundo caso: Todo ha ido "OK".

En este caso, si todo ha funcionado correctamente siempre devolverá "OK" puesto que hay métodos que no necesitan devolver un texto concreto. En caso se querer indicar al usuario un mensaje, se indicará antes de enviar al usuario la notificación pertinente.

Rangos de usuario

Cada usuario tiene un rango:

- 0: Usuario normal (Por defecto).
- 1: Usuario premium.
- 2: Administrador.

Planes premium:

- 1: Anual.
- 2: Mensual.
- 3: Gratuito (Por defecto).

La principal funcionalidad del rango es permitir acceder al panel de administración del sitio web.Ésto es controlado en:

- Menú `/resources/views/menu/header-menu.blade.php`.
- Vista principal de administración `/resources/views/admin.blade.php`.

En este segundo caso se ha optado por ser controlado directamente en la vista para evitar el duplicamiento de código, en un principio debería ser comprobado en cada método.

Control de errores

Globales:

- Que se envíe una petición `POST` y no `GET` en caso de ser requerido.

- Que el objeto indicado como `id` no exista.

Tutorial

- Que sea un nuevo tutorial:
- Solo debe admitir `new` en la dirección URL.
- El nombre de un tutorial no se puede repetir.
- Los campos no estén vacíos.

Planes

- No se debe perder el rango de administrador al cambiar a cualquier plan.
- El rango del usuario debe corresponderse con el plan correspondiente.

Usuario

Formará parte del controlador `UserController`

- Redirigir al perfil si se ha iniciado sesión (login/registro), o al login en caso contrario.
- Registro
    - Que el email no exista ya en la base de datos.
- Login
    - Que el correo y contraseña sean correctos.