

Pràctica 1 de Comerç Electrònic

Presentació

Aquesta Pràctica se centra en els conceptes apresos en els diferents mòduls de l'assignatura, realitzant el desenvolupament d'una botiga online bàsica a través de PHP/MySQL. Per dur-la a terme, cal treballar en un ordinador amb l'entorn d'Apache XAMPP, on es desplegarà la solució.

Competències

- Capacitat de comunicació escrita en l'àmbit acadèmic i professional.
- Capacitat per adaptar-se a les tecnologies i als futurs entorns actualitzant les competències professionals.
- Capacitat per analitzar un problema en el nivell d'abstracció adequat a cada situació i aplicar les habilitats i coneixements adquirits per abordar-lo i resoldre'l.
- Capacitat per aplicar les tècniques específiques de tractament, emmagatzematge i administració de dades.
- Capacitat per proposar i avaluar diferents alternatives tecnològiques per a resoldre un problema concret.

Objectius

- Creació d'una botiga online bàsica amb HTML, CSS, PHP i MySQL.

Recursos

Bàsics

- Mòduls didàctics de l'assignatura.
- Informació proporcionada pel consultor al tauler de l'aula virtual.

Complementaris

- Ordinador amb connexió a Internet i navegador web.

Criteris d'avaluació

La pràctica té com a objectiu familiaritzar-se amb la programació web per crear una botiga online amb els punts bàsics de tot comerç electrònic.

Per importància de cara a la nota, es podria repartir de la següent manera:

- Realització del punt 1) i punt 2) de la pràctica (50%)
- Realització del punt 3) (20%)
- Realització del punt 4) (15%)
- Realització del punt 5) (15%)

En l'avaluació no és estrictament necessari completar tots els punts per aprovar. No obstant, s'ha de tenir en compte que els punts 1 i 2 són un requisit necessari per poder fer amb èxit la resta dels punts.

Format i data de lliurament

La solució s'ha de lliurar en un arxiu comprimit .zip o .rar anomenat **CognomsNom_CE_Practica1** que contindrà els diferents arxius de solució de la pràctica (arxius php, css, base de dades en format .sql, etc.) i un document PDF amb l'estat de la pràctica.

El lliurament ha de ser com a màxim el **dimecres dia 7 de desembre a les 24:00h**, a la bústia de lliurament d'activitats de l'Aula Virtual de l'assignatura.

Descripció de la pràctica a realitzar

Enunciat

En aquest document es fa una descripció de la pràctica 1 de l'assignatura de Comerç Electrònic. La pràctica consisteix en la implementació d'una botiga virtual bàsica a partir de diferents punts i es basa en la teoria que s'ha vist en el mòdul "Gestió de la Informació" dels mòduls didàctics de l'assignatura.

Introducció

Per desenvolupar la nostra botiga virtual necessitarem els següents elements tecnològics:

- **Sistema operatiu:** es pot treballar amb qualsevol sistema operatiu (Windows / MacOS / Linux) sobre el qual s'instal·larà l'aplicació XAMPP (inclou servidor web Apache, MySQL i PHP).

Per tal de començar amb la instal·lació de les diferents eines per realitzar la pràctica, es facilita un petit guió d'instal·lació d'aquestes, disponible a través de l'apartat de Recursos d'aprenentatge de l'Aula Virtual.

L'eina XAMPP, disponible per a Windows, MacOS i Linux, es pot descarregar en aquest enllaç: <https://www.apachefriends.org/es/index.html>

Es recomana descarregar i utilitzar la versió 7.4.28 amb **PHP 7.4.28** per a posterior compatibilitat amb la pràctica 2. Per a això, es pot descarregar aquesta versió 7.4.28 des d'aquest enllaç: <https://www.apachefriends.org/download.html>

Consideracions generals:

- No es pot fer ús de CMS com PrestaShop, Woocommerce, Magento o similars. Ha de ser una programació bàsica però feta des de zero mitjançant HTML/CSS i PHP.
- Es poden utilitzar frameworks de PHP tipus Laravel o Symfony, però no està dins de l'abast d'aquesta pràctica aprendre a usar-los.

Descripció dels punts

En aquest apartat es descriuen els punts que conformen la pràctica.

1. Disseny i implementació de la base de dades de la botiga.
2. Creació de l'estructura bàsica en HTML / PHP de la botiga.
3. Gestió de la cistella de la compra.
4. Gestió del procés de compra online.
5. Creació d'un panell d'administració per a consulta de les comandes.

Punt 1: disseny i implementació de la base de dades de la botiga.

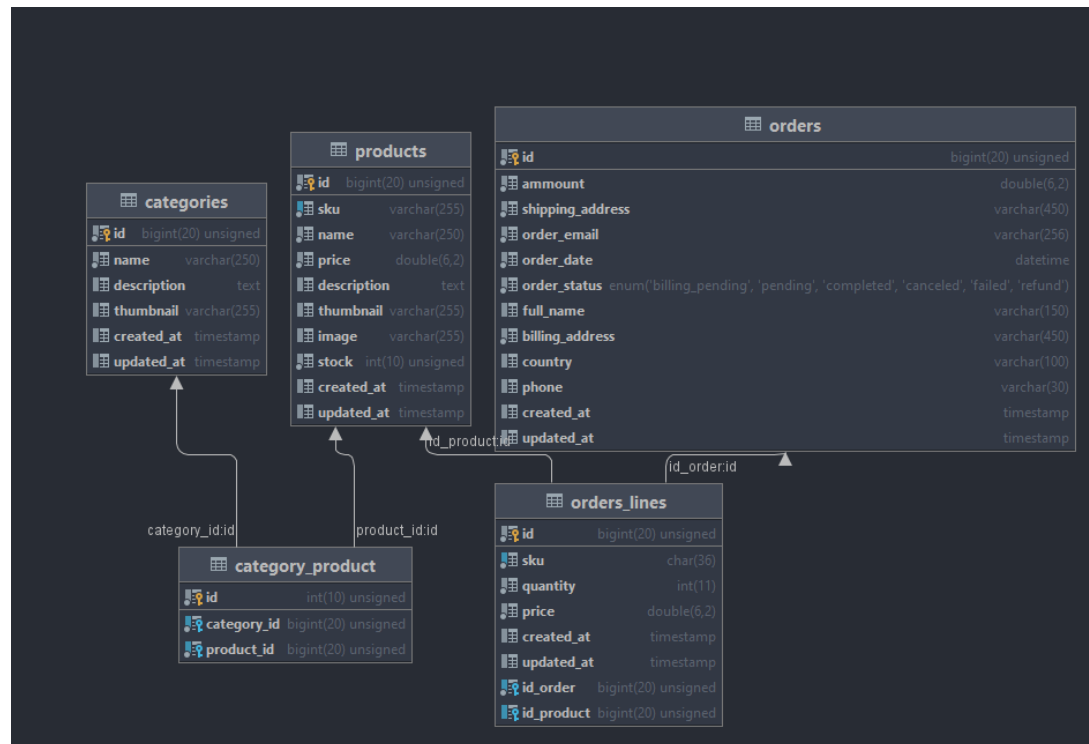
S'hauran de crear les taules relacionals necessàries de la botiga online usant el phpMyAdmin directament o mitjançant la importació d'un fitxer .sql amb l'estructura de la mateixa.

S'haurà d'identificar les dades necessàries a emmagatzemar en les taules de la base de dades i crear aquesta en base a aquesta informació i les seves relacions.

També serà necessària la creació d'un usuari i contrasenya per accedir a la base de dades mitjançant els arxius PHP corresponents.

Es pot utilitzar de referència l'estructura de la base de dades del tema 3 de teoria, punt 2.2.1, encara que no cal implementar totes les opcions que apareixen com a estats de comanda, idiomes o facturació.

Aquest el model relacional que he dissenyat (al directori ./docs es troba el UML y el SQL):



A continuació explicaré amb més detall cada taula i la estratègia a seguir.

(Aquest model relacional ha sigut auto generat per el ORM Eloquent de Laravel a traves de les meves definicions escrites en PHP, aquestes “migracions” es troben al directori ./database/migrations). Es per això que es recomanable crear/recrear el entorn de la base de dades amb les eines que proporciona el framework (Explicació al fitxer README.MD del projecte php), i no executar directament el fitxer .SQL del directori ./docs

A la taula productes guardarem:

- El seu id auto incremental
- El SKU unic del producte (Unique)
- El nom
- El preu
- Un descripció
- Una fotografia del producte
- L'estoc del producte
- El timestamp del producte

Raw SQL:

```
create or replace table products
(
    id bigint unsigned auto_increment
      primary key,
    sku varchar(255) not null,
    name varchar(250) not null,
    price double(6,2) not null,
    description text null,
    thumbnail varchar(255) null,
    image varchar(255) null,
    stock int unsigned default 0 not null,
    created_at timestamp null,
    updated_at timestamp null,
    constraint products_sku_unique
      unique (sku)
)
collate=utf8mb4_unicode_ci;
```

Migració Eloquent (ORM Laravel):

```
class CreateProductsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('products', function (Blueprint $table) {
            $table->bigIncrements('id');
            $table->string('sku')->unique();
            $table->string('name', 'length: 250');
            $table->float('price', 'total: 6, places: 2');
            $table->text('description')->nullable();
            $table->string('thumbnail', 'length: 255')->nullable();
            $table->string('image', 'length: 255')->nullable();
            $table->integer('stock', 'autoIncrement: false, unsigned: true')->default('value: 0');
            $table->timestamps();
        });
    }
}
```

A la taula Categories guardarem:

- El seu id auto incremental
- El nom de la categoria
- Una descripció de la categoria
- Una imatge de la categoria (no s'utilitza)
- El timestamp de la categoria

Raw SQL:

```
create or replace table categories
(
    id bigint unsigned auto_increment
    primary key,
    name varchar(250) not null,
    description text null,
    thumbnail varchar(255) null,
    created_at timestamp null,
    updated_at timestamp null
)

collate=utf8mb4_unicode_ci;
```

Migració Eloquent (ORM Laravel):

```
class CreateCategoriesTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('categories', function (Blueprint $table) {
            $table->bigIncrements('id');
            $table->string('name', 250);
            $table->text('description')->nullable();
            $table->string('thumbnail', 255)->nullable();
            $table->timestamps();
        });
    }
}
```

La relació entre Productes y Categoria es ManyToMany es a dir un producte es pot associar en 0 o mes Categories i una categoria por tenir 0 mes productes associats, per resoldre aquesta relació es crea la taula category_product on cada tupla serà la combinació única de id_categoria i id_producte.

Raw SQL:

```
create or replace table category_product
(
    id int unsigned auto_increment
        primary key,
    category_id bigint unsigned not null,
    product_id bigint unsigned not null,
    constraint category_product_category_id_product_id_unique
        unique (category_id, product_id),
    constraint category_product_category_id_foreign
        foreign key (category_id) references categories (id)
        on update cascade on delete cascade,
    constraint category_product_product_id_foreign
        foreign key (product_id) references products (id)
        on update cascade on delete cascade
)
collate=utf8mb4_unicode_ci;
```

Si un producte o categoria s'elimina, totes les tuples que els continguin s'esborraran en cascada, eliminant la relació entre ells.

Migració Eloquent (ORM Laravel):

```
class CreateCategoryProductTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        // https://appdividend.com/2022/01/21/laravel-many-to-many-relationship/
        Schema::create( table: 'category_product', function (Blueprint $table) {
            $table->increments( column: 'id');

            $table->foreignId( column: 'category_id')
                ->constrained( table: 'categories')
                ->onUpdate( action: 'cascade')
                ->onDelete( action: 'cascade');

            $table->foreignId( column: 'product_id')
                ->constrained( table: 'products')
                ->onUpdate( action: 'cascade')
                ->onDelete( action: 'cascade');

            $table->unique(['category_id', 'product_id']);
        });
    }
}
```

A la taula Orders (Comandes) guardarem:

- El seu id auto incremental
- El preu total de la comanda
- La direcció de enviament
- El email del client
- La data de comanda
- El estat de la comanda
- Nom del client
- Direcció de facturació
- País del client
- Telèfon del client
- Timestamp de la comanda

Raw SQL:

```
create or replace table orders
(
  id bigint unsigned auto_increment
    primary key,
  ammount double(6,2) not null,
  shipping_address varchar(450) not null,
  order_email varchar(256) not null,
  order_date datetime not null,
  order_status enum('BILLING_PENDING', 'PENDING', 'COMPLETED', 'CANCELED', 'FAILED', 'REFUND') default 'PENDING' not null,
  full_name varchar(150) null,
  billing_address varchar(450) not null,
  country varchar(100) null,
  phone varchar(30) null,
  created_at timestamp null,
  updated_at timestamp null
)
collate=utf8mb4_unicode_ci;
```

Migració Eloquent (ORM Laravel):

```
class CreateOrdersTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('orders', function (Blueprint $table) {
            $table->bigIncrements('id');
            $table->float('ammount', total: 6, places: 2);
            $table->string('shipping_address', length: 450);
            $table->string('order_email', length: 256);
            $table->dateTime('order_date')->default(DB::raw('CURRENT_TIMESTAMP'));
            $table->enum('order_status',
                ['BILLING_PENDING',
                 'PENDING',
                 'COMPLETED',
                 'CANCELED',
                 'FAILED',
                 'REFUND']
            )->default('PENDING');
            $table->string('full_name', length: 150)->nullable();
            $table->string('billing_address', length: 450);
            $table->string('country', length: 100)->nullable();
            $table->string('phone', length: 30)->nullable();
            $table->timestamps();
        });
    }
}
```


A la taula de Orders_Lines (Línia Comanda) guardarem:

- El seu Id auto incremental
- El SKU únic del producte associat a la línia de la comanda
- Unitats del producte comprat associat a la línia de la comanda
- Preu de la línia (preu producte*unitat comprades a la línia)
- Timestamp de la línia
- Claus foranies que associa la línia amb una comanda y un producte

Estratègia:

- S'estableixen restriccions UNIQUE per fer que una línia sols pugui estar en una comanda determinada y vinculi un únic producte.
- Si una comanda s'elimina s'esborren totes les línies associades en cascada
- Si un producte s'elimina, per tal de mantenir un històric de comandades amb el seu desglossat de línia, la clau forania que apunta al producte es posa en NULL en cascada

Raw SQL:

```
create or replace table orders_lines
(
  id bigint unsigned auto_increment
    primary key,
  sku char(36) not null,
  quantity int not null,
  price double(6,2) not null,
  created_at timestamp null,
  updated_at timestamp null,
  id_order bigint unsigned not null,
  id_product bigint unsigned null,
  constraint orders_lines_id_order_id_product_unique
    unique (id_order, id_product),
  constraint orders_lines_sku_id_order_unique
    unique (sku, id_order),
  constraint orders_lines_id_order_foreign
    foreign key (id_order) references orders (id)
    on update cascade on delete cascade,
  constraint orders_lines_id_product_foreign
    foreign key (id_product) references products (id)
    on update cascade on delete set null
)
collate=utf8mb4_unicode_ci;
```

Migració Eloquent (ORM Laravel):

```
class CreateOrdersLinesTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('orders_lines', function (Blueprint $table) {
            $table->bigIncrements('id');
            $table->string('sku');
            $table->integer('quantity');
            $table->float('price', ['total' => 6, 'places' => 2]);
            $table->timestamps();

            $table->foreignId('id_order')
                ->constrained('orders')
                ->onUpdate('cascade')
                ->onDelete('cascade');

            $table->foreignId('id_product')->nullable()
                ->constrained('products')
                ->onUpdate('cascade')
                ->onDelete('set null');

            $table->unique(['id_order', 'id_product']);
            $table->unique(['sku', 'id_order']);
        });
    }
}
```

Punt 2: creació de l'estructura bàsica en HTML/PHP de la botiga.

Una estructura bàsica vàlida podria ser la següent (s'inclouen tots els punts demanats a la pràctica):

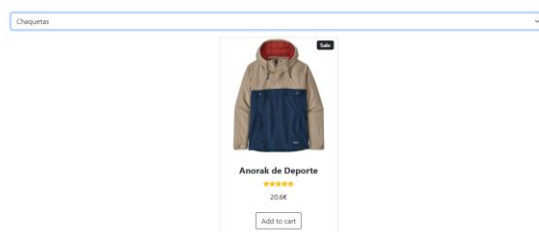
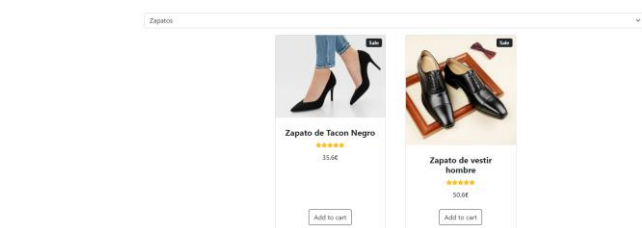
- Pàgina d'inici: es mostren les categories. Per exemple, en una botiga de calçat, les categories podrien ser: botes, sabates, sandàlies, etc.
- Pàgina de categories: llistat dels productes d'aquesta categoria.

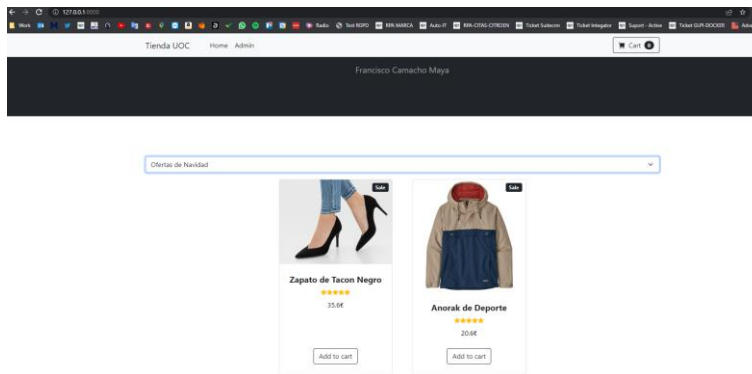
En la meua versió he unit els 2 punts anteriors, la pagina principal "127.0.0.1:8000/"

Mostra un menú de selecció amb totes les categories (carregades asincronament amb JS) y una llista del productes de la categoria seleccionada (Carregats asincronament amb JS):

Fitxers implicats:

- `app/Http/Controllers/HomeController.php`
- `resources/views/home/home.blade.php`
- `resources/views/product/productItem.blade.php`
- `resources/views/base/*.blade.php` (Vistes reutilitzables en las vistas principals)
- `app/Http/Controllers/CategoryController.php`
- `public/js/category/category.js`
- `routes/web.php`
- `routes/api.php`





(Aquí es pot veure com un producte per estar a una mateixa categoria)

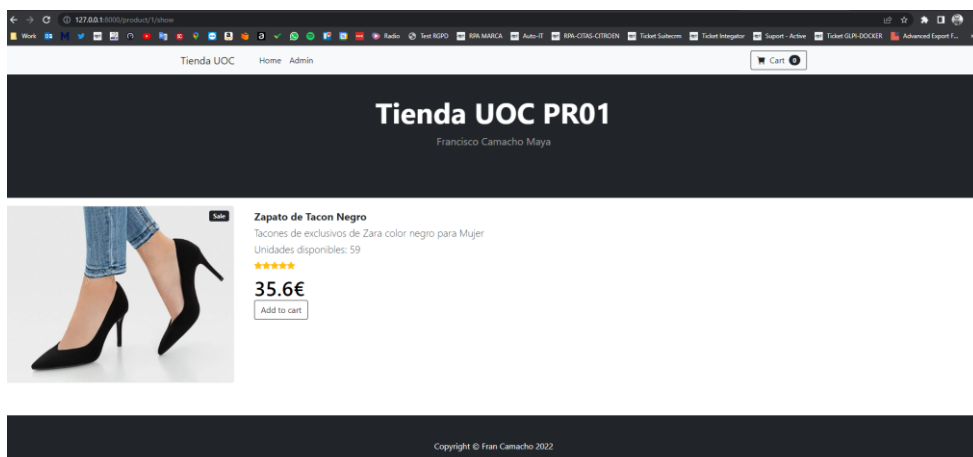
- Pàgina de producte: informació del producte i opció d'afegir a la cistella de la compra.

Es pot accedir a la pagina detallada de un producte clicant a la imatge del mateix en la vista principal de la botiga.

La url per veure el detall de un producte es :
 “127.0.0.1:8080/product/<id_producto>/show”

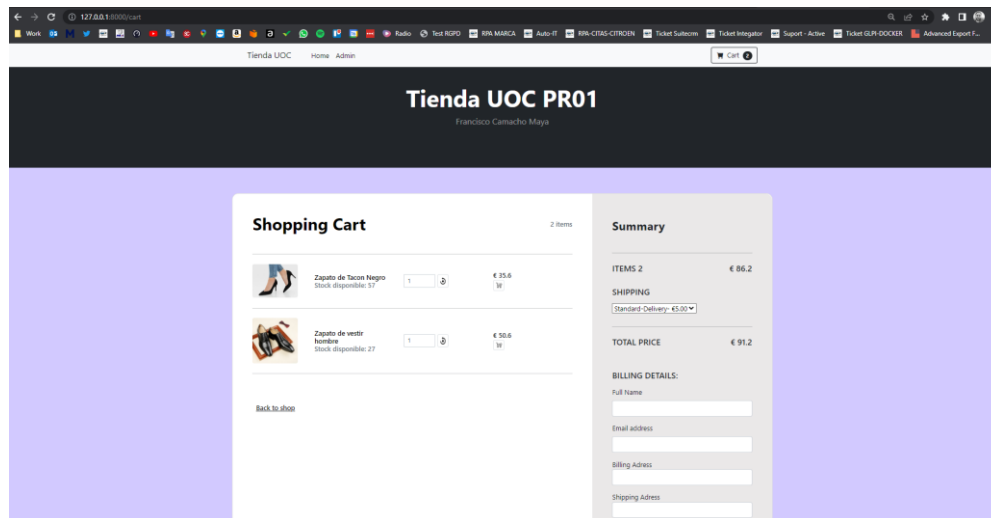
Fitxers implicats:

- app/Http/Controllers/ProductController.php
- resources/views/product/productdetail.blade.php
- resources/views/base/*.blade.php (Vistes reutilitzables en las vistas principals)
- routes/web.php
- routes/api.php



- Web de la cistella: es mostra el contingut de la cistella actual i es permet la compra.
- Pàgina de compra: resum de la cistella i formulari per realitzar la comanda.

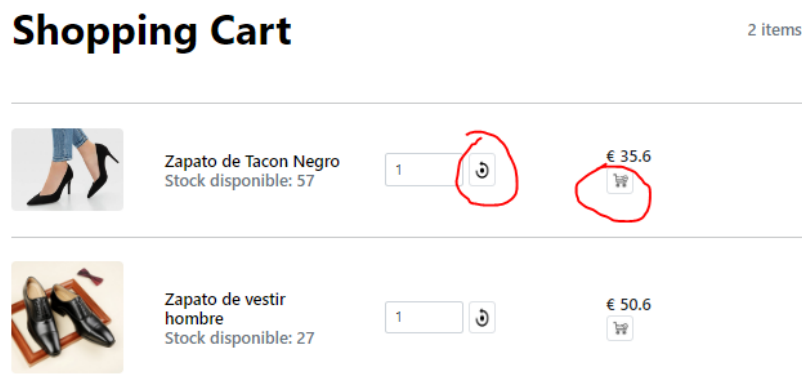
Es pot accedir a la pagina de la cistella clicant a la imatge la cistella disponible en la barra de navegació de la pagina o a la url: "127.0.0.1:8080/cart"



En la meua versió he unit els 2 punts anteriors en una sola pagina, es pot afegir un producte des de qualsevol punt de la pagina, així com accedir a la cistella.

Les unitats que es volen comprar en cada producte es controla des de la web de la cistella, es controla el stock disponible i no es pot posar mes del que hi ha.

A la cistella trobem els controls necessaris per actualitzar i esborrar un producte dins de la cistella (recalculem la cistella asincronament via JS).



Els preus de la cistella i l'import total es re calcula correctament si modifiquem la cistella amb els seus controls.

També tenim el formulari de compra:

Summary

ITEMS 2
€ 86.2

SHIPPING

Standard-Delivery- €5.00

TOTAL PRICE
€ 91.2

BILLING DETAILS:

Full Name

Email address

Billing Address

Shipping Address

Country

Phone

Checkout

Al clicar sobre el boto checkout es crea un registre a la taula Orders i per cada element de la cistella es crea un registre a la taula orders_lines.

```
public function create(Request $request, CartService $cartService): JsonResponse
{
    $cart = $cartService->getCart();
    $order = new Order();
    $order->fill($request->all())->save();
    $order = $order->refresh();

    foreach ($cart as $itemCart){
        $orderLine = new OrderLine([
            "sku" => $itemCart->product->sku,
            "quantity" => $itemCart->quantity,
            "price" => $itemCart->product->price,
        ]);
        $orderLine->product()->associate($itemCart->product);
        $orderLine->order()->associate($order);
        $orderLine->save();
    }

    $cartService->clearCart();

    return new JsonResponse(['order'=>$order, 'message'=>"Compra realizada correctamente"], status: 201);
}
```

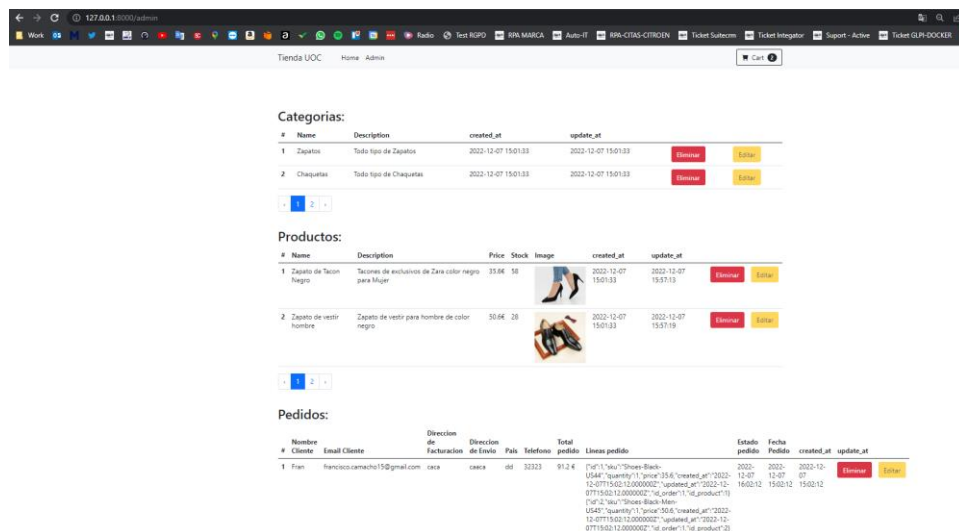
Fitxers implicats:

- app/Http/Controllers/ProductController.php
- app/Http/Controllers/OrderController.php
- public/js/cart/cart.js
- resources/views/cart/cart.blade.php
- resources/views/base/*.blade.php (Vistes reutilitzables en las vistas principals)
- routes/web.php
- routes/api.php

- Pàgina d'administració: visualització dels detalls de les comandes emmagatzemades.

Per accedir a la pagina de administració de la botiga hem de accedir a :

“127.0.0.1:8080/admin” esta protegit via HTTP-BASIC el user es : admin@admin.com i el password es: “password” aquesta informació la treu de la taula “users” taula interna que crea laravel per guardar el usuaris operatius de la aplicació web.



The screenshot shows the admin interface of a web application. It has a top navigation bar with 'Tienda UOC', 'Home', and 'Admin'. A 'Cart' icon is on the right. The main content area is divided into three sections: 'Categorías', 'Productos', and 'Pedidos'.

Categorías:

#	Name	Description	created_at	update_at		
1	Zapatos	Todo tipo de Zapatos	2022-12-07 15:01:33	2022-12-07 15:01:33	Eliminar	Editar
2	Chaquetas	Todo tipo de Chaquetas	2022-12-07 15:01:33	2022-12-07 15:01:33	Eliminar	Editar

Productos:

#	Name	Description	Price	Stock	Image	created_at	update_at		
1	Zapato de Tacón Negro	Tacones de exclusivos de Zara color negro para Mujer	35.00	50		2022-12-07 15:01:33	2022-12-07 15:01:33	Eliminar	Editar
2	Zapato de vestir hombre	Zapato de vestir para hombre de color negro	30.00	20		2022-12-07 15:01:33	2022-12-07 15:01:33	Eliminar	Editar

Pedidos:

Numero	Nombre	Email Cliente	Direccion de Facturacion	Direccion de Envio	País	Telefono	Total pedido	Lineas pedido	Estado pedido	Fecha Pedido	created_at	update_at		
1	Fran	francisco.camacho15@gmail.com	oeca	oeca	idit	33323	912.4	[{"id":1,"sku":"Shoes-Black-US44","quantity":1,"price":35.0,"created_at":"2022-12-07T15:02:12.000000Z","updated_at":"2022-12-07T15:02:12.000000Z","id_pedido":1,"id_producto":1}, {"id":2,"sku":"Shoes-Black-Men-US43","quantity":1,"price":30.0,"created_at":"2022-12-07T15:02:12.000000Z","updated_at":"2022-12-07T15:02:12.000000Z","id_pedido":1,"id_producto":2}]	2022-12-07 15:02:12	2022-12-07 15:02:12	2022-12-07 15:02:12	2022-12-07 15:02:12	Eliminar	Editar

No m'ha donat temps de presentar correctament les línies de les comandes en la taula de comandes de la vista i esta renderitzat el array de línies en JSON.

Tampoc m'ha donat temps a fer les vistes per afegir y editar nous objectes (La API Rest esta feta però no he pogut per temps fer els formularis via JS).

El que si es pot fer es eliminar objectes de la botiga.

Es pot veure que les taules estan paginades, he posat una paginació petita de 2 elements per pagina perquè es pugui veure aquesta funcionalitat amb pocs registres.

Fitxers implicats:

- app/Http/Controllers/AdminController.php
- public/js/admin/admin.js
- resources/views/base/*.blade.php (Vistes reutilitzables en las vistes principals)
- routes/web.php
- routes/api.php

Rutes de la API REST de les entitats (Les rutes amb el middleware “api”):

```
PS C:\Users\fcamacho\PhpstormProjects\PRO1-TiendaOnline> php artisan route:list
```

Domain	Method	URI	Name	Action	Middleware
	GET HEAD	/	home	App\Http\Controllers\HomeController@index	web
	GET HEAD	admin	admin	App\Http\Controllers\AdminController@index	web
	POST	api/category		App\Http\Controllers\CategoryController@create	api
	GET HEAD	api/category		App\Http\Controllers\CategoryController@getAll	api
	GET HEAD	api/category/{id}		App\Http\Controllers\CategoryController@get	api
	PUT	api/category/{id}		App\Http\Controllers\CategoryController@update	api
	DELETE	api/category/{id}		App\Http\Controllers\CategoryController@delete	api
	GET HEAD	api/category/{id}/products		App\Http\Controllers\CategoryController@getProductsByCategory	api
	GET HEAD	api/order		App\Http\Controllers\OrderController@getAll	api
	DELETE	api/order/{id}		App\Http\Controllers\OrderController@delete	api
	GET HEAD	api/order/{id}		App\Http\Controllers\OrderController@get	api
	GET HEAD	api/product		App\Http\Controllers\ProductController@getAll	api
	POST	api/product		App\Http\Controllers\ProductController@create	api
	PUT	api/product/{id}		App\Http\Controllers\ProductController@update	api
	DELETE	api/product/{id}		App\Http\Controllers\ProductController@delete	api
	GET HEAD	api/product/{id}		App\Http\Controllers\ProductController@get	api
	GET HEAD	api/product/{id}/category		App\Http\Controllers\ProductController@getCategoryByProduct	api
	GET HEAD	cart	cart	Closure	web
	POST	order	order.create	App\Http\Controllers\OrderController@create	web
	GET HEAD	product/{id}/addProductToCart	addProductToCart	App\Http\Controllers\ProductController@addProductToCart	web
	GET HEAD	product/{id}/removeProductCart	removeProductCart	App\Http\Controllers\ProductController@removeProductCart	web
	GET HEAD	product/{id}/show	product.show	App\Http\Controllers\ProductController@showDetail	web
	GET HEAD	sanctum/csrf-cookie		Laravel\Sanctum\Http\Controllers\CsrfCookieController@show	web

Si es va al controlador de cada entitat (/app/Http/Controllers) es pot veure el codi de la API rest de les entitats y si vols provar-la post fer servir Postman.

Així mateix, en aquest punt es poden desenvolupar les opcions de disseny de CSS i JavaScript que es consideri necessari per al codi.

Nota: en aquest punt es pot fer ús de qualsevol plantilla HTML + CSS existent o de la biblioteca Bootstrap per facilitar la feina.

Punt 3: gestió de la cistella.

Creació de la lògica en PHP necessària per a la gestió d'una cistella de la compra online bàsica.

Com a punts a desenvolupar de cara a obtenir una major nota, es pot considerar la realització de les següents característiques:

- Accés a la cistella des de totes les pàgines de la botiga online.
- Gestió de les quantitats d'un mateix producte: si s'afegeix diverses vegades el mateix producte, el sistema podria recalculer la quantitat d'aquest producte en lloc d'afegir-lo unitàriament.
- Opció d'eliminar productes de la cistella carret.

Tots aquests punts els he explicat en el punt on explico la vista de la cistella i resum de compra.

La meua cistella la he implementat com a Servei es dir una classe de PHP que s'instancia una única vegada al executar el servidor web (Patró Singleton) i es consumit per els meus controladors, en el meu cas el controlador app/Http/Controllers/ProductController.php

En els endpoints “addProductToCart” i “removeProductToCart” s'injecta la dependència del servei de la cistella.

La cistella es basa en Sessions de PHP, per defecte Laravel guarda les sessions PHP al directori (storage/framework/sessions).

Es pot veure el codi de la cistella a ([app/Service/CartService.php](#))

```
class CartService
{
    private array $cart = [];

    public function __construct()
    {
        if(!session()->has("SESSION_CART")) {
            session()->put("SESSION_CART", $this->cart);
            session()->save();
        }
        else{
            $this->cart = session( 'key' "SESSION_CART");
        }
    }
}
```

```
public function getCart(){
    return $this->cart;
}

public function isProductInCart(int $productId)
{
    $key = null;
    if (count($this->cart)===0) return null;

    foreach($this->cart as $key => $value){
        if ($value->productId == $productId){
            $key = $key;
        }
    }

    return $key;
}
```

```
public function addProductCart(int $productId, int $quantity): bool{
    /** @var Product $product */
    $product = Product::query()->find($productId);
    if ($product){
        // Si product esta en el carro recupero el stock del item del carro en la base de datos y
        // le resto la cantidad requerida finalmente
        $key = $this->isProductInCart($productId);
        if($key != null){
            $finalStock = (($product->stock+$this->cart[$key]->quantity)-$quantity);
            $product->update([
                'stock'=>($finalStock)
            ]);
            $product = $product->refresh();
            $this->cart[$key]->product = $product;
            $this->cart[$key]->quantity = $quantity;
        }else{
            $product->update([
                'stock'=>$product->stock-$quantity
            ]);
            $product = $product->refresh();

            $itemCart = (object)[
                'productId'=>$productId,
                'product'=>$product,
                'quantity' => $quantity
            ];

            $this->cart[] = $itemCart;
        }
        session()->put("SESSION_CART", $this->cart);
        session()->save();

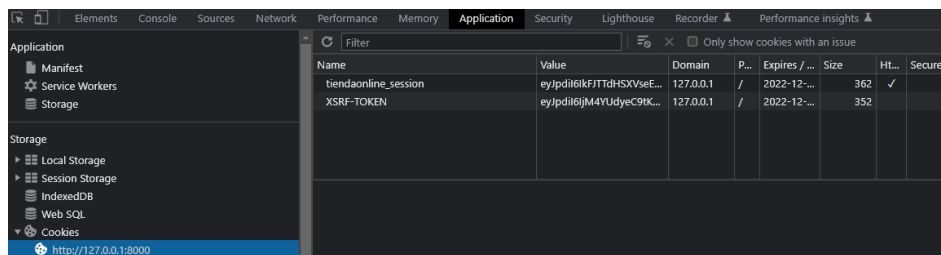
        return true;
    }else{
        return false;
    }
}
```

```
public function removeProductCart(int $productId): bool{
    /** @var Product $product */
    $product = Product::query()->find($productId);
    if($product){
        $key = $this->isProductInCart($productId);
        if($key != null){
            $product->update([
                'stock'=>$product->stock+$this->cart[$key]->quantity
            ]);
            $product->refresh();

            unset($this->cart[$key]);
            session()->put("SESSION_CART", $this->cart);
            session()->save();
            return true;
        }
    }

    return false;
}

public function clearCart(){
    $this->cart = [];
    session()->put("SESSION_CART", $this->cart);
    session()->save();
}
```

Valoració Personal:

En general ha sigut una PR divertida de fer, jo no soc programador de PHP (Soc Backend Developer Python), amb aquesta PR he tingut la oportunitat de fer servir un framework full stack on no tenia massa practica (vaig utilitzar Symfony fa molts anys al Cicle Formatiu).

Tota la part de Backend (API, Base de dades, Control de sessions..etc) com es normal no ha sigut un gran problema ja que ja faig servir aquestes tecnologies a diari, el que mes m'ha costat es la part de les vistes i la potencia que el template engine de Laravel (Blade) posseeix. M'ha deixat impressionat i el trobo molt superior a altres alternatives basades en python com pot ser Jinja2.

Per altre banda també he de dir que el ORM de PHP Eloquent es una obra de enginyeria i possiblement sigui superior al ORM de Django o SQLAlchemy (ORM que actualment utilitzo).

La documentació de Laravel es molt completa y fàcil de llegir.

Si he de dir alguna cosa dolenta de aquesta PR es la poca informació que es dona respecte a el llenguatge PHP, crec que en aquest grau no hi ha cap assignatura que prepari al alumne per utilitzar un llenguatge interpretat per fer una aplicació web i es dona per sentat molts conceptes com el de les cookies i les sessions. (No trobo suficient el recursos que doneu a la assignatura al respecte).

