

COMERÇ ELECTRÒNIC

PRA 1 - CREACIÓ D'UNA BOTIGA ONLINE BÀSICA

JUAN RAMÓN GAVILANES SANCHEZ

ÍNDICE

Contexto.....	2
Puesta en marcha del proyecto en local.....	3
1.- Clonamos el repositorio.....	3
2.- Instalamos dependencias php para poder usar SAIL.....	3
3.- Configura el acceso a datos de la aplicación.....	4
4.- Arranca entorno de desarrollo, ejecuta migraciones y seeders.....	5
5.- Comprobar entorno de desarrollo.....	6
Punto 1. Diseño e implementación de la base de datos de la tienda.....	7
Punto 2. Creación de la estructura básica en HTML/PHP de la tienda.....	9
Página de inicio con las categorías disponibles.....	9
Página de categorías.....	10
Página de producto.....	11
Carro de la compra.....	12
Página de compra.....	13
Página de administración.....	15
Punto 3. Gestión de la cesta.....	17
Punto 4. Gestión del proceso de compra online.....	18
Compra usuario invitado.....	19
Compra usuario registrado.....	20
Punto 5. Panel de administración para consultar pedidos realizados.....	21
Bibliografía.....	23

Contexto

Para la utilización de la práctica, he usado el framework de PHP llamado **Laravel** en su versión 11.

Aprovecho la práctica para aprender a trabajar con este framework, ya que próximamente vamos a utilizarlo en el trabajo.

He simulado una tienda de compra de videojuegos retro, llamada **RetroStore**

He alojado el código del proyecto en el repositorio público de GitHub <https://github.com/jrgavilanes/uoc-store>.

Se puede interactuar con una versión en vivo de la práctica en la dirección <https://janrax.es/>

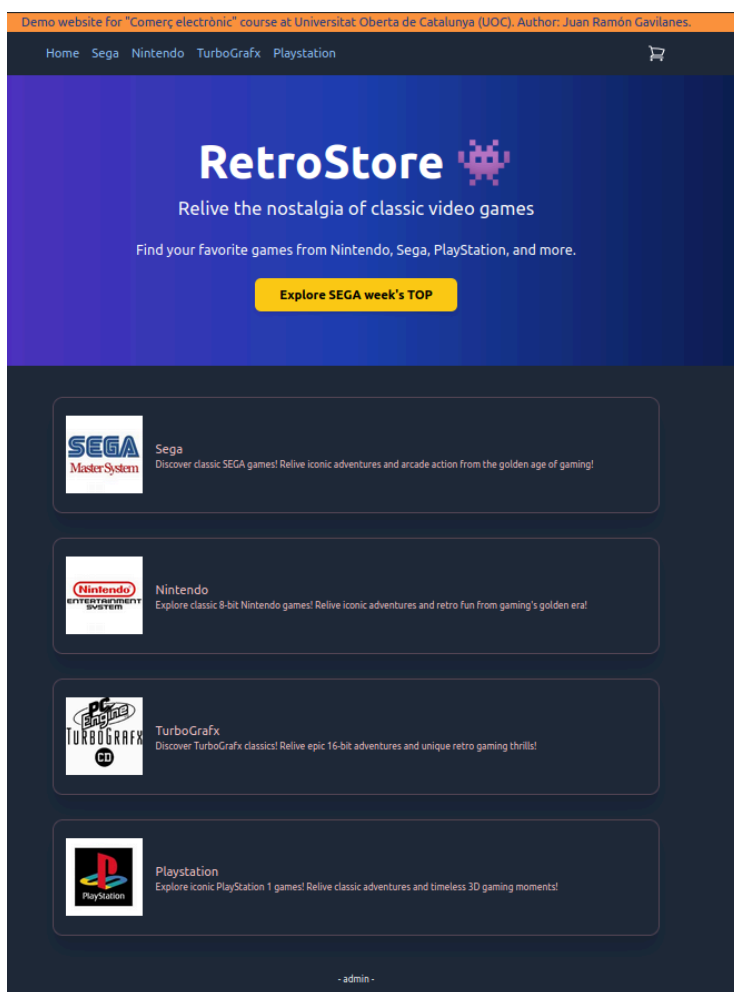


Figura 1: Frontpage de la tienda

Puesta en marcha del proyecto en local

He desarrollado el proyecto en un sistema **Linux Mint**, utilizando contenedores **Docker**.

Las únicas dependencias que debemos tener en nuestro sistema, serán **git**, **docker** y **docker-compose**. Para explotar la base de datos MySQL utilizo **DBeaver CE**.

1.- Clonamos el repositorio

```
git clone git@github.com:jrgavilanes/uoc-store.git .
```

```
janrax@janrax-Legion-5-15ACH6H:~/Escritorio/uoc-store-local$ git clone git@github.com:jrgavilanes/uoc-store.git .
Clonando en '.'...
remote: Enumerating objects: 501, done.
remote: Counting objects: 100% (501/501), done.
remote: Compressing objects: 100% (321/321), done.
remote: Total 501 (delta 229), reused 415 (delta 144), pack-reused 0 (from 0)
Recibiendo objetos: 100% (501/501), 611.72 KiB | 168.00 KiB/s, listo.
Resolviendo deltas: 100% (229/229), listo.
janrax@janrax-Legion-5-15ACH6H:~/Escritorio/uoc-store-local$ ls -ltr
total 468
-rwxrwxr-x 1 janrax janrax 2086 nov 10 09:58 README.md
drwxrwxr-x 2 janrax janrax 4096 nov 10 09:58 documentation
-rwxrwxr-x 1 janrax janrax 3224 nov 10 09:58 docker-compose.yml
drwxrwxr-x 5 janrax janrax 4096 nov 10 09:58 database
drwxrwxr-x 2 janrax janrax 4096 nov 10 09:58 config
-rwxrwxr-x 1 janrax janrax 298158 nov 10 09:58 composer.lock
-rwxrwxr-x 1 janrax janrax 2041 nov 10 09:58 composer.json
drwxrwxr-x 3 janrax janrax 4096 nov 10 09:58 bootstrap
-rwxrwxr-x 1 janrax janrax 350 nov 10 09:58 artisan
drwxrwxr-x 5 janrax janrax 4096 nov 10 09:58 app
drwxrwxr-x 5 janrax janrax 4096 nov 10 09:58 resources
drwxrwxr-x 4 janrax janrax 4096 nov 10 09:58 public
-rwxrwxr-x 1 janrax janrax 80 nov 10 09:58 postcss.config.js
-rwxrwxr-x 1 janrax janrax 1121 nov 10 09:58 phpunit.xml
-rwxrwxr-x 1 janrax janrax 102291 nov 10 09:58 package-lock.json
-rwxrwxr-x 1 janrax janrax 407 nov 10 09:58 package.json
-rwxrwxr-x 1 janrax janrax 263 nov 10 09:58 vite.config.js
drwxrwxr-x 4 janrax janrax 4096 nov 10 09:58 tests
-rwxrwxr-x 1 janrax janrax 243 nov 10 09:58 tailwind.config.js
drwxrwxr-x 6 janrax janrax 4096 nov 10 09:58 storage
drwxrwxr-x 2 janrax janrax 4096 nov 10 09:58 routes
```

2.- Instalamos dependencias php para poder usar SAIL

```
docker run --rm -v $(pwd):/app composer install
```

```
janrax@janrax-Legion-5-15ACH6H:~/Escritorio/uoc-store-local$ docker run --rm -v $(pwd):/app composer install
Installing dependencies from lock file (including require-dev)
Verifying lock file contents can be installed on current platform.
Package operations: 111 installs, 0 updates, 0 removals
- Downloading symfony/finder (v7.1.4)
- Downloading symfony/polyfill-mbstring (v1.31.0)
- Downloading symfony/var-dumper (v7.1.5)
- Downloading psr/log (3.0.2)
- Downloading maximebf/debugbar (v1.23.2)
- Downloading voku/portable-ascii (2.0.1)
- Downloading symfony/polyfill-php80 (v1.31.0)
- Downloading symfony/polyfill-ctype (v1.31.0)
- Downloading phpoption/phpoption (1.9.3)
- Downloading graham-campbell/result-type (v1.1.3)
- Downloading vlucas/phpdotenv (v5.6.1)
```

...

```
- Installing wireui/wireui (v2.1.3): Extracting archive
 0/111 [>-----] 0%
48/111 [=====>-----] 43%
74/111 [=====>-----] 66%
92/111 [=====>-----] 82%
108/111 [=====>-----] 97%
111/111 [=====] 100%
Generating optimized autoload files
> Illuminate\Foundation\ComposerScripts::postAutoloadDump
> @php artisan package:discover --ansi
```

```
INFO Discovering packages.
```

```
barryvdh/laravel-debugbar ..... DONE
laravel/sail ..... DONE
laravel/tinker ..... DONE
livewire/livewire ..... DONE
nesbot/carbon ..... DONE
nunomaduro/collision ..... DONE
nunomaduro/termwind ..... DONE
wireui/heroicons ..... DONE
wireui/wireui ..... DONE
```

80 packages you are using are looking for funding.
Use the `composer fund` command to find out more!

3.- Configura el acceso a datos de la aplicación

```
cp .env.example .env && vi .env
```

```
janrax@janrax-Legion-5-15ACH6H:~/Escritorio/uoc-store-local$ ls -a
.  .artisan  composer.lock  docker-compose.yml  .env  .git  package.json  postcss.config.js  resources  tailwind.config.js  vite.config.js
.. bootstrap  config  documentation  .env.example  .gitattributes  package-lock.json  public  routes  tests
app  composer.json  database  .editorconfig  .env.testing  phpunit.xml  README.md  storage  vendor
janrax@janrax-Legion-5-15ACH6H:~/Escritorio/uoc-store-local$ cp .env.example .env
janrax@janrax-Legion-5-15ACH6H:~/Escritorio/uoc-store-local$ cat .env
APP_NAME=Laravel
APP_ENV=local
APP_KEY=base64:qoccXY9hqpBls90MJnJqN9L0rAZFFp7ze1DEiuc3XxQ=
APP_DEBUG=true
APP_TIMEZONE=UTC
APP_URL=http://localhost

APP_LOCALE=en
APP_FALLBACK_LOCALE=en
APP_FAKER_LOCALE=en_US

APP_MAINTENANCE_DRIVER=file
# APP_MAINTENANCE_STORE=database

BCRYPT_ROUNDS=12

LOG_CHANNEL=stack
LOG_STACK=single
LOG_DEPRECATIONS_CHANNEL=null
LOG_LEVEL=debug

DB_CONNECTION=mysql
DB_HOST=mysql
DB_PORT=3306
DB_DATABASE=uoc-store
DB_USERNAME=sail
DB_PASSWORD=password

SESSION_DRIVER=database
SESSION_LIFETIME=120
SESSION_ENCRYPT=false
SESSION_PATH=/
SESSION_DOMAIN=null
```

4.- Arranca entorno de desarrollo, ejecuta migraciones y seeders.

```

janrax@janrax-Legion-5-15ACH6H:~/Escritorio/uoc-store-local$ ./vendor/bin/sail up
Creating network "uoc-store-local_sail" with driver "bridge"
Creating volume "uoc-store-local_sail-mysql" with local driver
Creating volume "uoc-store-local_sail-redis" with local driver
Creating volume "uoc-store-local_sail-meilisearch" with local driver
Creating uoc-store-local_selenium_1 ... done
Creating uoc-store-local_meilisearch_1 ... done
Creating uoc-store-local_mysql_1 ... done
Creating uoc-store-local_mailpit_1 ... done
Creating uoc-store-local_redis_1 ... done
Creating uoc-store-local_laravel.test_1 ... done
Attaching to uoc-store-local_selenium_1, uoc-store-local_redis_1, uoc-store-local_mailpit_1, uoc-store-local_meilisearch_1, uoc-store-local_mysql_1, uoc-store-local_laravel.test_1
meilisearch_1 | 888b d888 d8b 888 d8b 888
meilisearch_1 | 8888b d8888 Y8P 888 Y8P
meilisearch_1 | 88888b.d88888 888 888
meilisearch_1 | 888Y888888P888 d88b 888 888 888 d8888b d88b 888888 888d888 d8888b 88888b.
meilisearch_1 | 888 Y888P 888 d8P Y8b 888 888 888 88K d8P Y8b "88b 888P" d88P" 888 "88b
meilisearch_1 | 888 Y8P 888 888888888 888 888 888 "Y8888b. 888888888 d888888 888 888 888 888
meilisearch_1 | 888 " 888 Y8b. 888 888 888 X88 Y8b. 888 888 888 Y88b. 888 888
meilisearch_1 | 888 888 "Y8888 888 888 888 888888P' "Y8888 "Y888888 888 "Y8888P 888 888
meilisearch_1 |
meilisearch_1 | Config file path: "none"
meilisearch_1 | Database path: ".data.ms"
meilisearch_1 | Server listening on: "http://0.0.0.0:7700"

janrax@janrax-Legion-5-15ACH6H:~/Escritorio/uoc-store-local$ ./vendor/bin/sail artisan migrate
INFO: Preparing database.
Creating migration table ..... 23.33ms DONE
INFO: Running migrations.
0001_01_01_000000 create_users_table ..... 104.86ms DONE
0001_01_01_000001 create_cache_table ..... 39.39ms DONE
0001_01_01_000002 create_jobs_table ..... 100.90ms DONE
2024_10_09_071124 create_categories_table ..... 51.85ms DONE
2024_10_09_071125 create_products_table ..... 136.84ms DONE
2024_10_09_071312 create_orders_table ..... 86.63ms DONE
2024_10_09_072203 create_order_lines_table ..... 129.86ms DONE

janrax@janrax-Legion-5-15ACH6H:~/Escritorio/uoc-store-local$ ./vendor/bin/sail artisan db:seed
INFO: Seeding database.
Database\Seeders\CategorySeeder ..... RUNNING
Database\Seeders\CategorySeeder ..... 18 ms DONE
Database\Seeders\ProductSeeder ..... RUNNING
Database\Seeders\ProductSeeder ..... 96 ms DONE
Database\Seeders\OrderSeeder ..... RUNNING
Database\Seeders\OrderSeeder ..... 33 ms DONE

```

```
./vendor/bin/sail up
```

```
./vendor/bin/sail npm run dev
```

Desde otro terminal, ejecutamos la migraciones y seeders para que cree las tablas y las cargue con datos de prueba.

```
./vendor/bin/sail artisan migrate
```

```
./vendor/bin/sail artisan db:seed
```

Este comando crea el usuario con permisos administrativos **admin@test.com**, y dos usuarios registrados: **user1@test.com** y **user2@test.com**. La contraseña de los tres es **password**.

5.- Comprobar entorno de desarrollo

Ya podemos acceder a la base de datos que hayamos configurado, por defecto, la base de datos será **uoc-store**, user: **sail**, clave: **password**

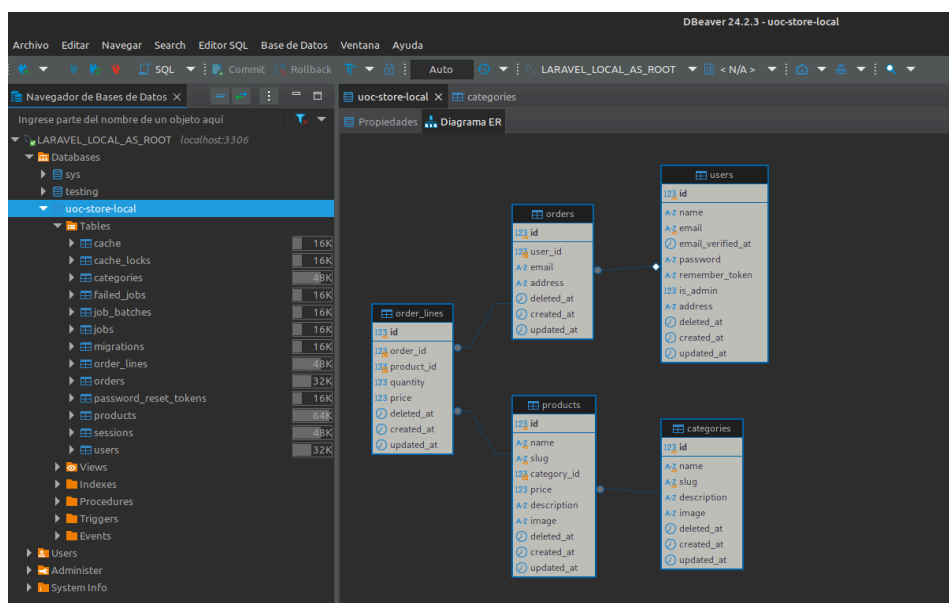


Figura 2: Accediendo a la base de datos desde DBeaver

Si accedemos a **localhost** desde un navegador, ya veremos la tienda en nuestro entorno de desarrollo.

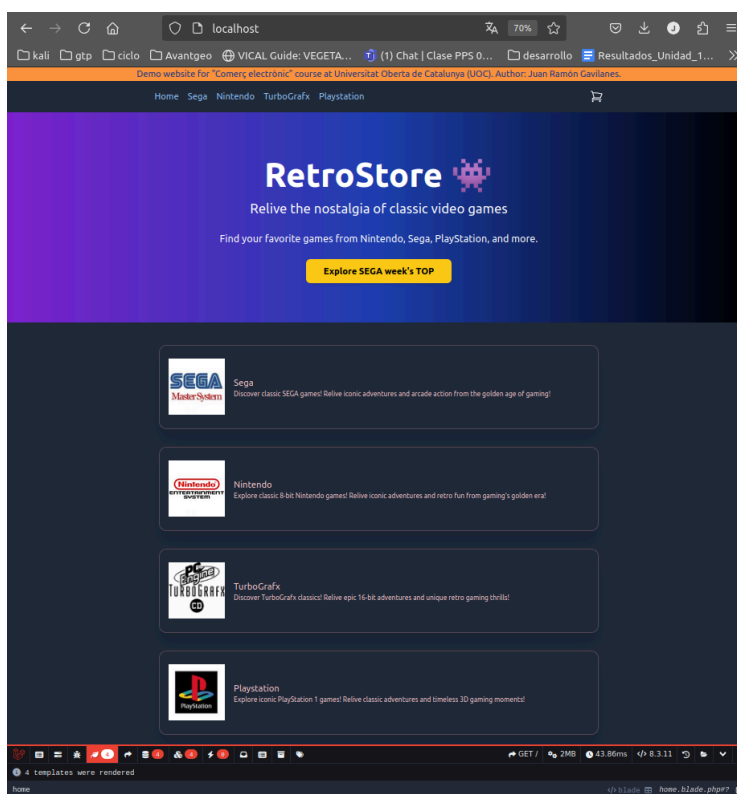


Figura 3: Tienda web desde nuestro entorno de desarrollo

Punto 1. Diseño e implementación de la base de datos de la tienda.

Se puede acceder al *dump* de la base de datos desde el siguiente enlace: [backup.sql](#)

Hay que tener en cuenta que el framework Laravel introduce tablas propias para gestionar cache, colas de trabajo, migraciones, sesiones, etc.

Las tablas de trabajo que hemos creado son las siguientes:



Figura 4: Tablas de trabajo de la tienda web

Users(id, name, **email**, email_verified_at, password, remember_token, address, is_admin, deleted_at, created_at, updated_at)

Orders(id, user_id, email, address, deleted_at, created_at, updated_at)

- Relación: **Users(id)** -> **Orders(user_id)**

Order_lines(id, order_id, product_id, quantity, price, created_at, updated_at)

- Relación: **Orders(id)** -> **Order_lines(order_id)**
- Relación: **Products(id)** -> **Order_lines(product_id)**

Products(id, name, slug, category_id, price, description, image, deleted_at, created_at, updated_at)

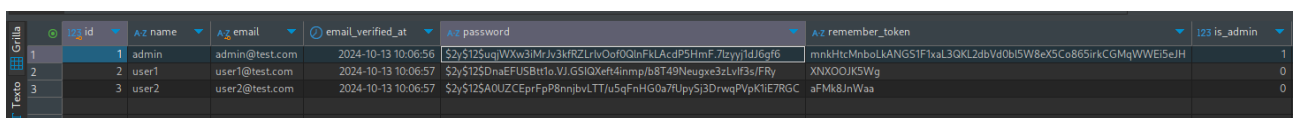
- Relación: **Categories(id)** -> **Products(category_id)**

Categories(id, name, slug, description, image, deleted_at, created_at, updated_at)

He simplificado al máximo la estructura de tablas con respecto a la mostrada en el tema 3, para optimizarla al máximo con los requisitos actuales del frontend.

Como indicamos en el apartado anterior, podemos acceder a la base de datos que hayamos configurado, por defecto, la base de datos será **uoc-store**, user: **sail**, clave: **password**.

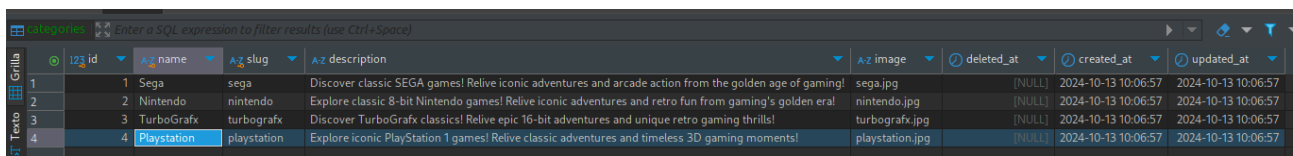
También hemos creado tres usuarios: **admin@test.com** con permisos administrativos, y dos usuarios registrados: **user1@test.com** y **user2@test.com**. La contraseña de los tres es **password**.



	id	name	email	email_verified_at	password	remember_token	is_admin
1	1	admin	admin@test.com	2024-10-13 10:06:56	\$2y\$12\$uqWxw3MrJv3KfRZLrlyOoF0QlnFKAcDP5HmF.7zyj1dJ6gf6	mnhHtcMnboLkANG5tF1xL3QKL2dbVd0b5W8eX5Co865irkCGMqWWE5eJH	1
2	2	user1	user1@test.com	2024-10-13 10:06:57	\$2y\$12\$DnaEFU5Btt1a.VJ.G5IQXef4inmp/b8T49Neugxe3zLvIf3s/FRy	XNKO0JK5Wg	0
3	3	user2	user2@test.com	2024-10-13 10:06:57	\$2y\$12\$A0UzCEprFpP8nnjvLTT/u5qFnHG0a7UpYj3DrwqPvPkiE7RGC	aFMk8JnWaa	0

Figura 5: Usuarios por defecto

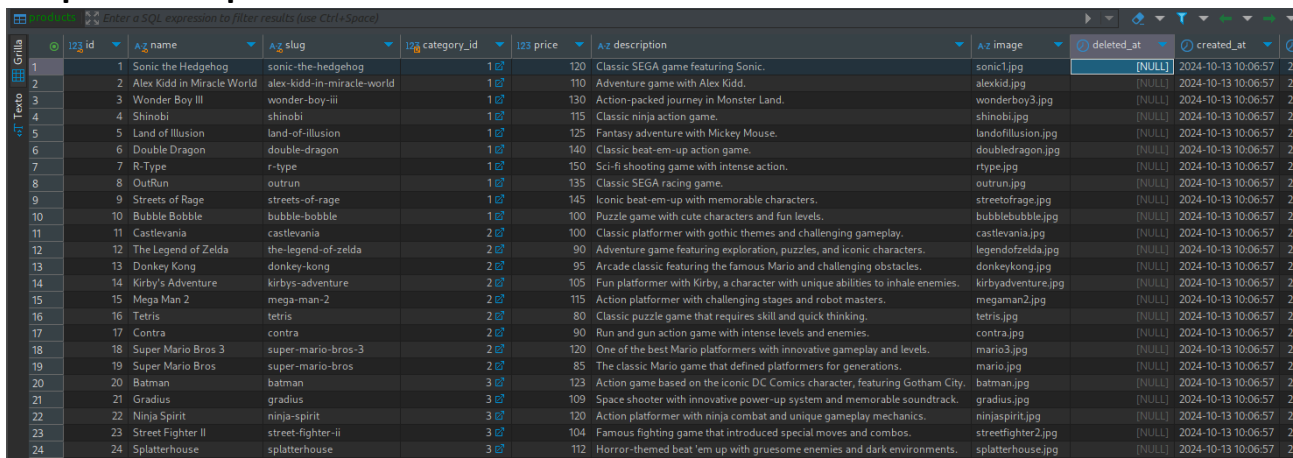
Las categorías creadas por defecto son las siguientes



	id	name	slug	description	image	deleted_at	created_at	updated_at
1	1	Sega	sega	Discover classic SEGA games! Relive iconic adventures and arcade action from the golden age of gaming!	sega.jpg	[NULL]	2024-10-13 10:06:57	2024-10-13 10:06:57
2	2	Nintendo	nintendo	Explore classic 8-bit Nintendo games! Relive iconic adventures and retro fun from gaming's golden era!	nintendo.jpg	[NULL]	2024-10-13 10:06:57	2024-10-13 10:06:57
3	3	TurboGrafx	turbografx	Discover TurboGrafx classics! Relive epic 16-bit adventures and unique retro gaming thrills!	turbografx.jpg	[NULL]	2024-10-13 10:06:57	2024-10-13 10:06:57
4	4	Playstation	playstation	Explore iconic PlayStation 1 games! Relive classic adventures and timeless 3D gaming moments!	playstation.jpg	[NULL]	2024-10-13 10:06:57	2024-10-13 10:06:57

Figura 6: Categorías por defecto

Los productos por defecto



	id	name	slug	category_id	price	description	image	deleted_at	created_at
1	1	Sonic the Hedgehog	sonic-the-hedgehog	1	120	Classic SEGA game featuring Sonic.	sonic1.jpg	[NULL]	2024-10-13 10:06:57
2	2	Alex Kidd in Miracle World	alex-kidd-in-miracle-world	1	110	Adventure game with Alex Kidd.	alexkid.jpg	[NULL]	2024-10-13 10:06:57
3	3	Wonder Boy III	wonder-boy-iii	1	130	Action-packed journey in Monster Land.	wonderboy3.jpg	[NULL]	2024-10-13 10:06:57
4	4	Shinobi	shinobi	1	115	Classic ninja action game.	shinobi.jpg	[NULL]	2024-10-13 10:06:57
5	5	Land of Illusion	land-of-illusion	1	125	Fantasy adventure with Mickey Mouse.	landofillusion.jpg	[NULL]	2024-10-13 10:06:57
6	6	Double Dragon	double-dragon	1	140	Classic beat-em-up action game.	doubledragon.jpg	[NULL]	2024-10-13 10:06:57
7	7	R-Type	r-type	1	150	Sci-fi shooting game with intense action.	rtype.jpg	[NULL]	2024-10-13 10:06:57
8	8	OutRun	outrun	1	135	Classic SEGA racing game.	outrun.jpg	[NULL]	2024-10-13 10:06:57
9	9	Streets of Rage	streets-of-rage	1	145	Iconic beat-em-up with memorable characters.	streetofrage.jpg	[NULL]	2024-10-13 10:06:57
10	10	Bubble Bobble	bubble-bobble	1	100	Puzzle game with cute characters and fun levels.	bubblebubble.jpg	[NULL]	2024-10-13 10:06:57
11	11	Castlevania	castlevania	2	100	Classic platformer with gothic themes and challenging gameplay.	castlevania.jpg	[NULL]	2024-10-13 10:06:57
12	12	The Legend of Zelda	the-legend-of-zelda	2	90	Adventure game featuring exploration, puzzles, and iconic characters.	legendofzelda.jpg	[NULL]	2024-10-13 10:06:57
13	13	Donkey Kong	donkey-kong	2	95	Arcade classic featuring the famous Mario and challenging obstacles.	donkeykong.jpg	[NULL]	2024-10-13 10:06:57
14	14	Kirby's Adventure	kirbys-adventure	2	105	Fun platformer with Kirby, a character with unique abilities to inhale enemies.	kirbyadventure.jpg	[NULL]	2024-10-13 10:06:57
15	15	Mega Man 2	mega-man-2	2	115	Action platformer with challenging stages and robot masters.	megaman2.jpg	[NULL]	2024-10-13 10:06:57
16	16	Tetris	tetris	2	80	Classic puzzle game that requires skill and quick thinking.	tetris.jpg	[NULL]	2024-10-13 10:06:57
17	17	Contra	contra	2	90	Run and gun action game with intense levels and enemies.	contra.jpg	[NULL]	2024-10-13 10:06:57
18	18	Super Mario Bros 3	super-mario-bros-3	2	120	One of the best Mario platformers with innovative gameplay and levels.	mario3.jpg	[NULL]	2024-10-13 10:06:57
19	19	Super Mario Bros	super-mario-bros	2	85	The classic Mario game that defined platformers for generations.	mario.jpg	[NULL]	2024-10-13 10:06:57
20	20	Batman	batman	3	123	Action game based on the iconic DC Comics character, featuring Gotham City.	batman.jpg	[NULL]	2024-10-13 10:06:57
21	21	Gradius	gradius	3	109	Space shooter with innovative power-up system and memorable soundtrack.	gradius.jpg	[NULL]	2024-10-13 10:06:57
22	22	Ninja Spirit	ninja-spirit	3	120	Action platformer with ninja combat and unique gameplay mechanics.	ninjaspirit.jpg	[NULL]	2024-10-13 10:06:57
23	23	Street Fighter II	street-fighter-ii	3	104	Famous fighting game that introduced special moves and combos.	streetfighter2.jpg	[NULL]	2024-10-13 10:06:57
24	24	Splatterhouse	splatterhouse	3	112	Horror-themed beat 'em up with gruesome enemies and dark environments.	splatterhouse.jpg	[NULL]	2024-10-13 10:06:57

Figura 7: Productos por defecto

Punto 2. Creación de la estructura básica en HTML/PHP de la tienda.

Página de inicio con las categorías disponibles

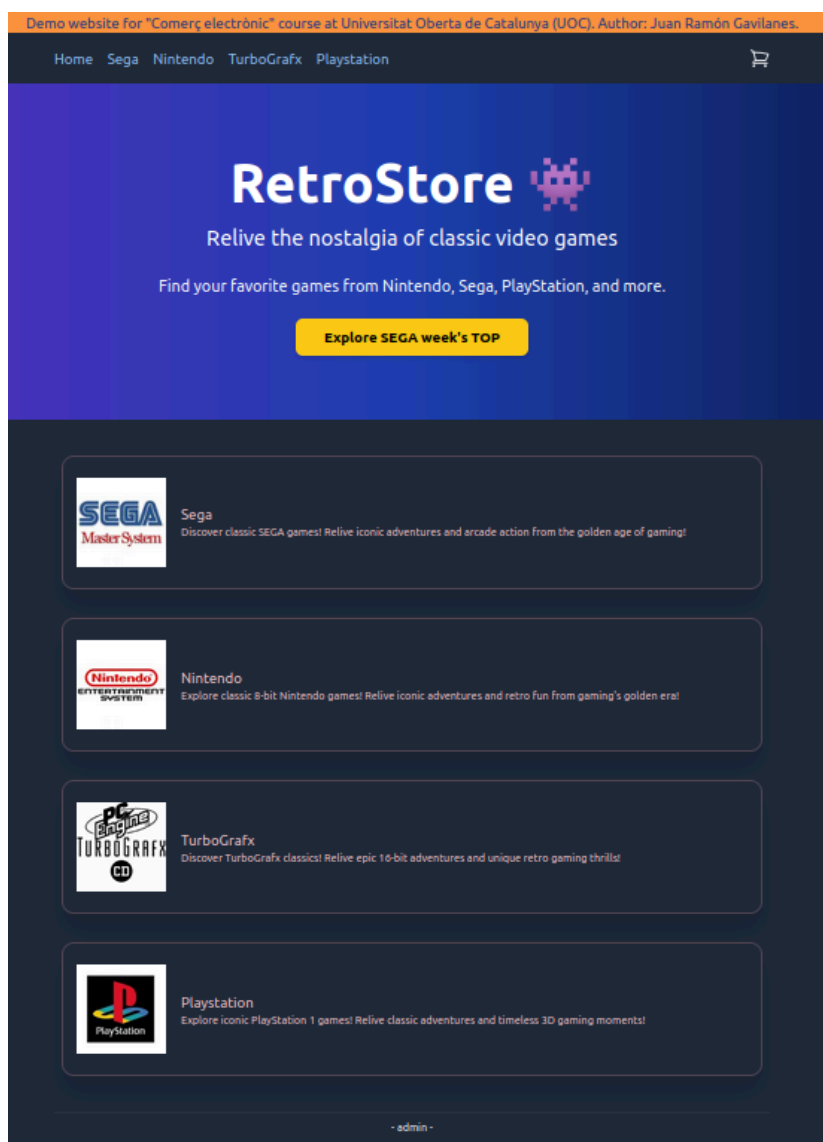


Figura 8: Página de inicio con las categorías disponibles

Página de categorías

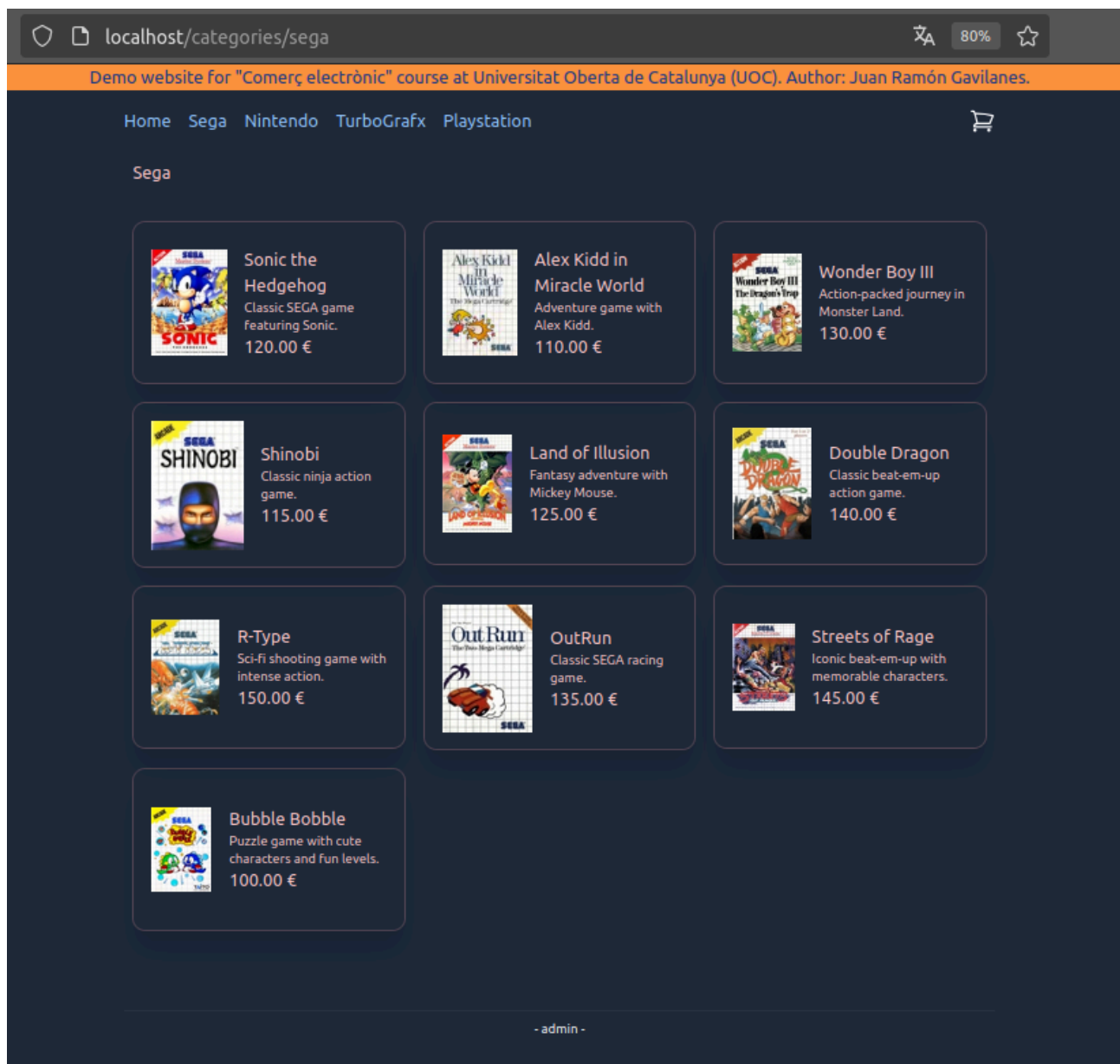


Figura 9: Página de categorías

Página de producto

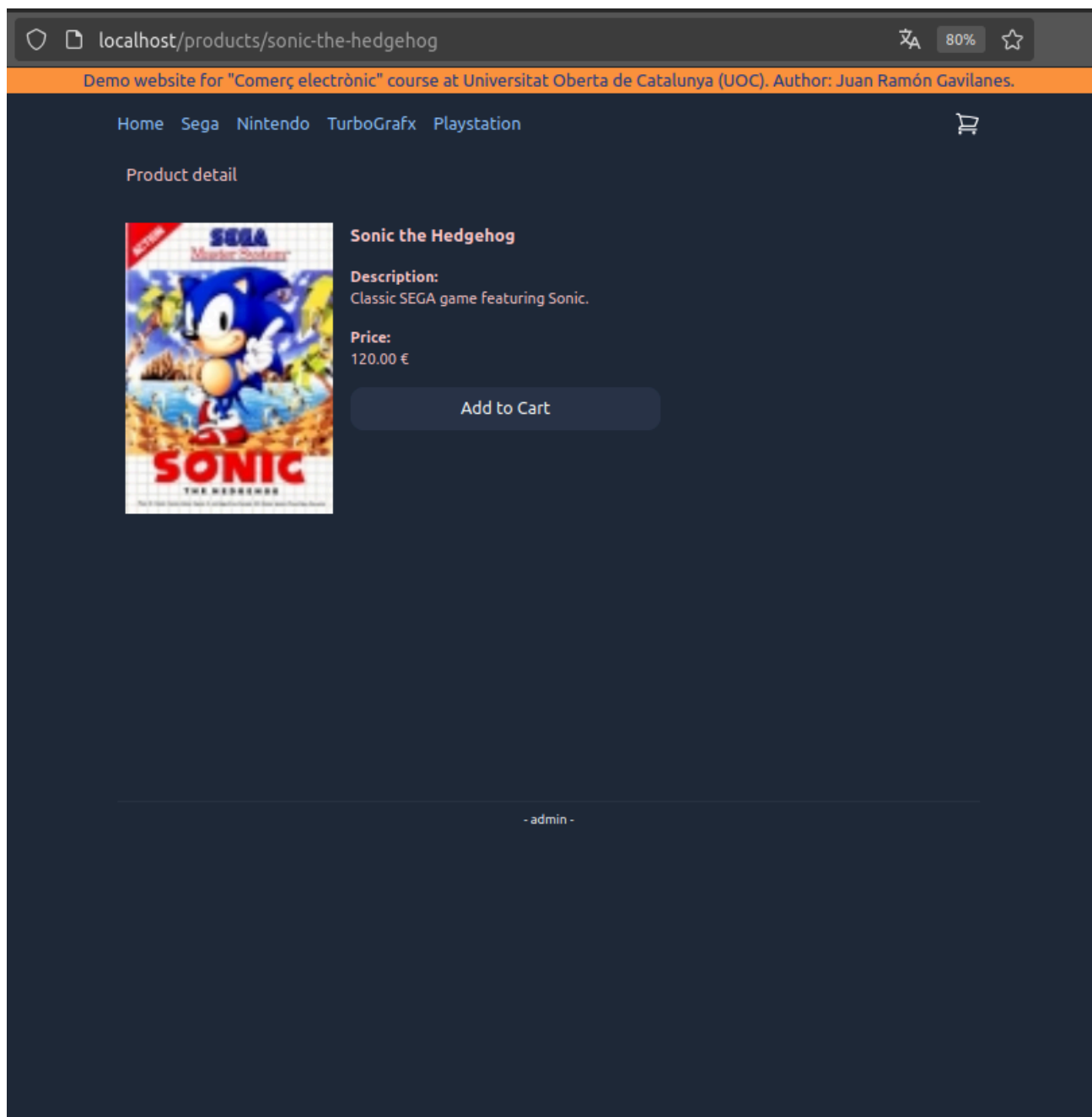


Figura 10: Página de producto

Carro de la compra

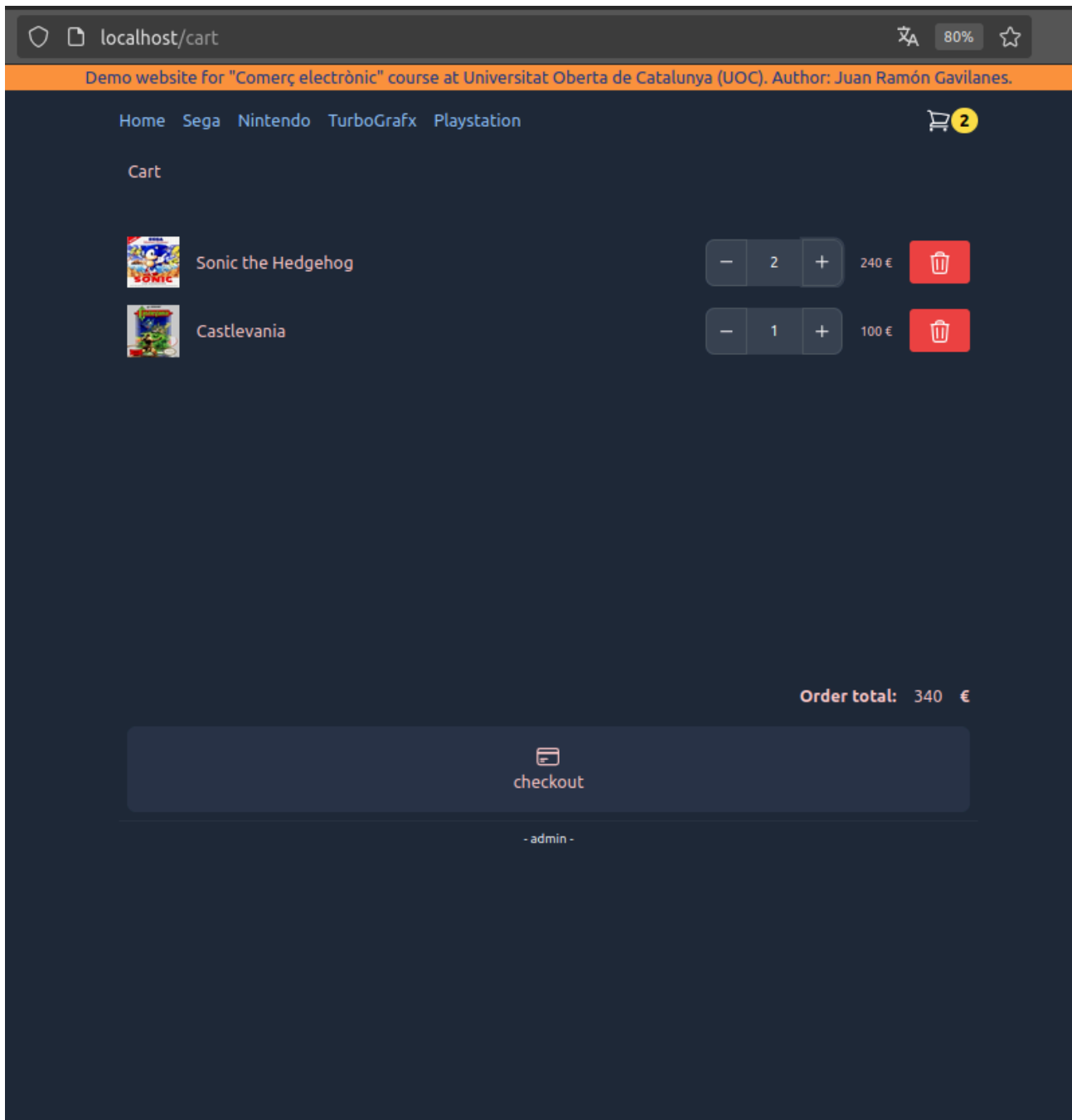


Figura 11: Web del carrito

Página de compra

localhost/checkout

Demo website for "Comerç electrònic" course at Universitat Oberta de Catalunya (UOC). Author: Juan Ramón Gavilanes.

Home Sega Nintendo TurboGrafx Playstation

2

Checkout

Guest users

Guest email

Guest address

Checkout

Registered users

Name

Email

Password

Address

Checkout

- admin -

Figura 12: Página de compra

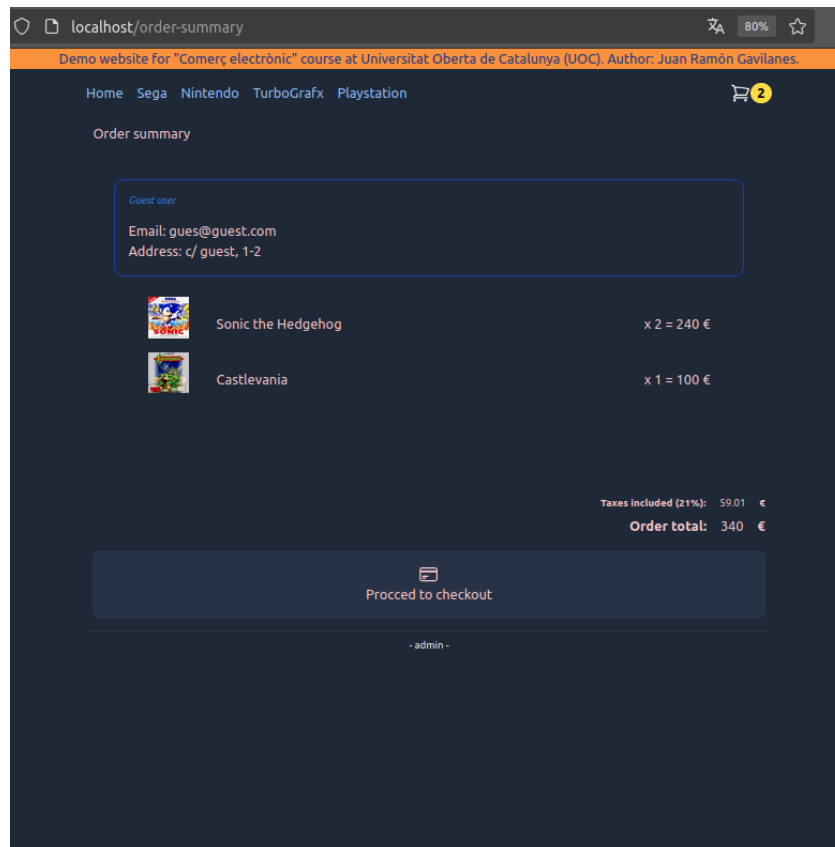
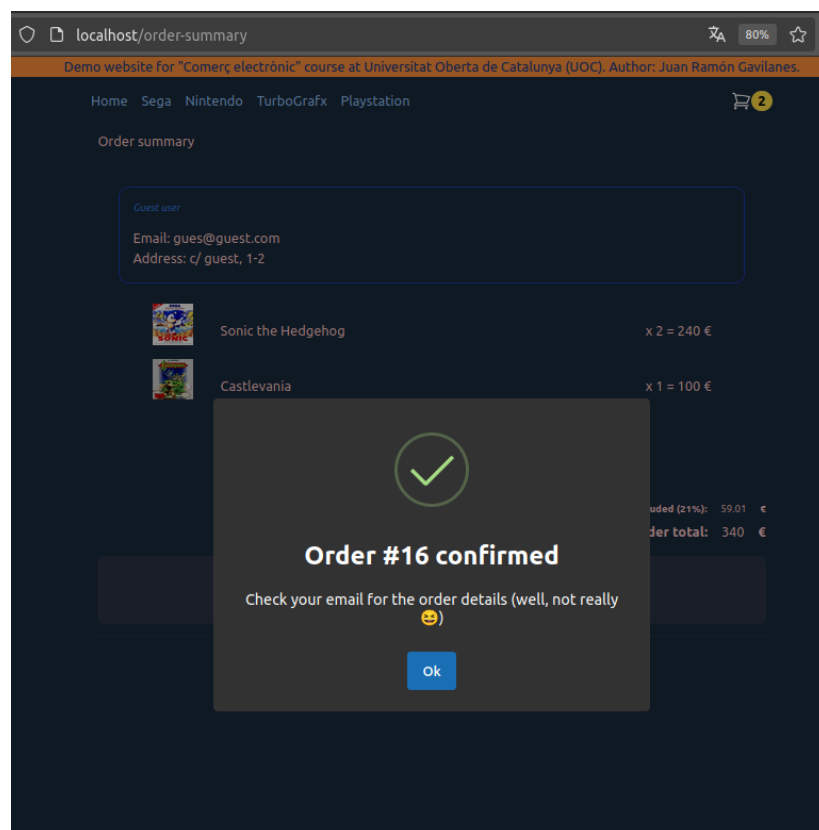


Figura 13: Pàgina de compra proceed to checkout

Figura 14: Confirmación de la compra.
Tras aceptar, vacía carrito y vuelve a pantalla principal.

Página de administración

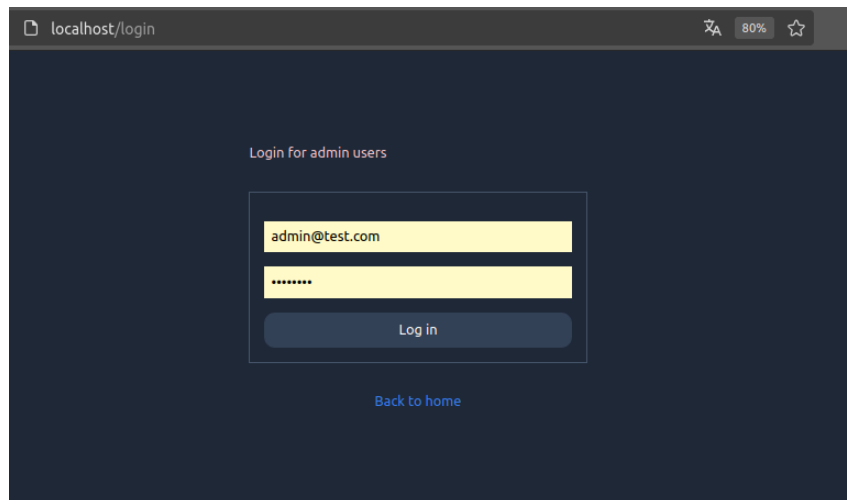


Figura 15: Acceso a página de administración

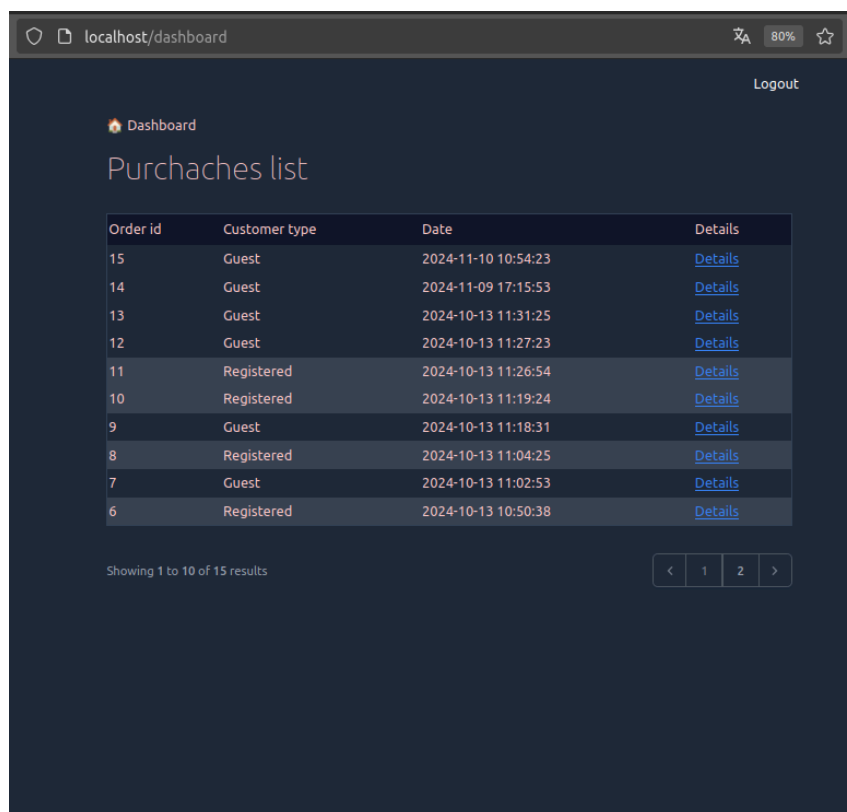


Figura 16: Lista de compras

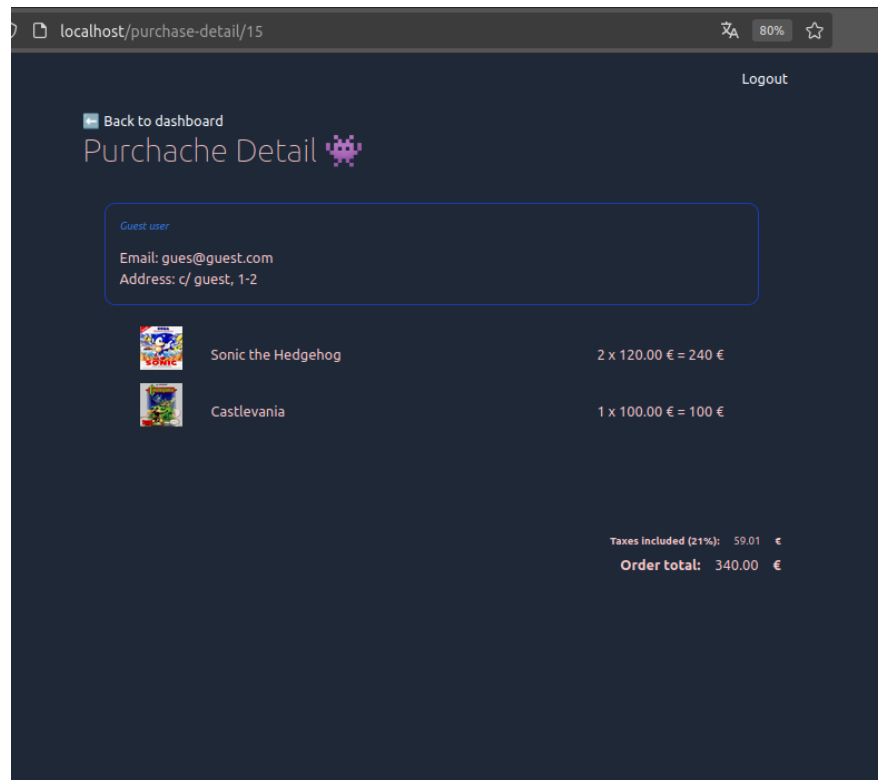


Figura 17: Detalle de la compra

Punto 3. Gestión de la cesta.

He conseguido que se pueda interactuar con el carrito desde cualquier pantalla del cliente. Por ejemplo, si se añade un producto, el contador de productos se actualiza en tiempo real.

La gestión del carrito de la compra la he llevado a cabo de forma híbrida entre el cliente y el servidor. Por un lado, he utilizado sessionStorage del navegador con javascript para interactuar de forma ágil con las cantidades y los productos.

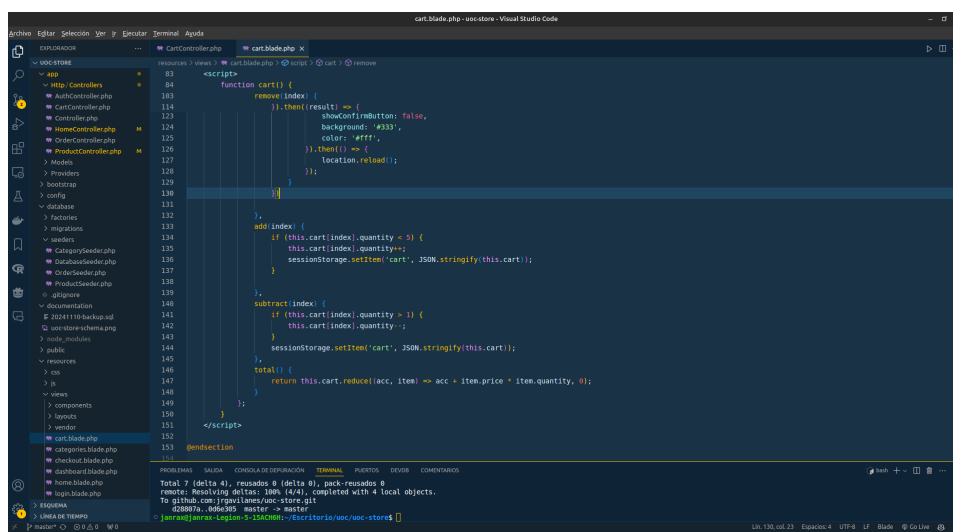


Figura 18: Fragmento de gestión de carrito por parte del cliente

Por otro lado, en el backend he controlado mediante sesiones los datos personales del cliente como el email, dirección, etc.

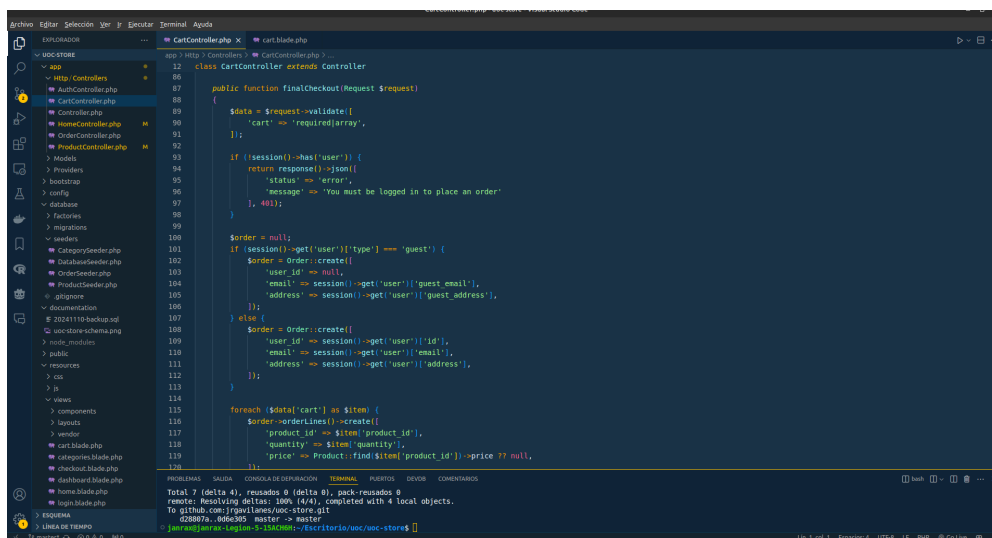


Figura 19: Fragmento de gestión de carrito por parte del servidor

Punto 4. Gestión del proceso de compra online.

Estas serían las tablas involucradas en el proceso de compra.

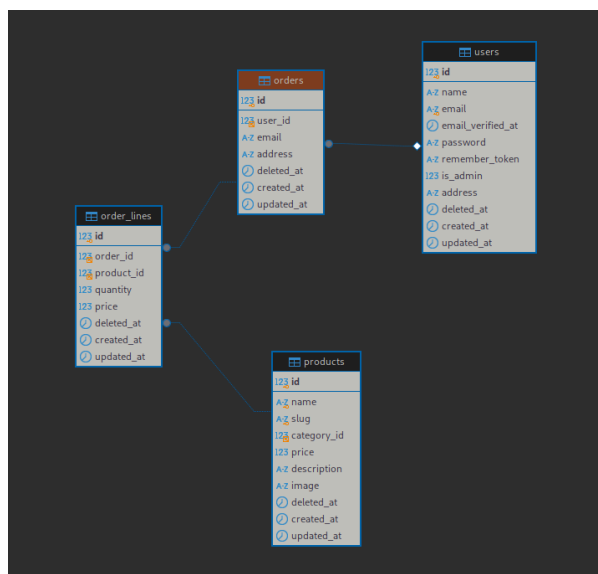


Figura 20: Tablas que intervienen en el proceso de compra.

```
-- uoc-store .orders definition
--
CREATE TABLE `orders` (
  `id` bigint unsigned NOT NULL AUTO_INCREMENT,
  `user_id` bigint unsigned DEFAULT NULL,
  `email` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `address` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `deleted_at` datetime DEFAULT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `orders_user_id_foreign` (`user_id`),
  CONSTRAINT `orders_user_id_foreign` FOREIGN KEY (`user_id`) REFERENCES `users` (`id`) ON DELETE RESTRICT
) ENGINE=InnoDB AUTO_INCREMENT=18 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

Figura 21: El campo user_id de la tabla órdenes puede ser nulo. De esta manera podremos aceptar usuarios no registrados.

Compra usuario invitado

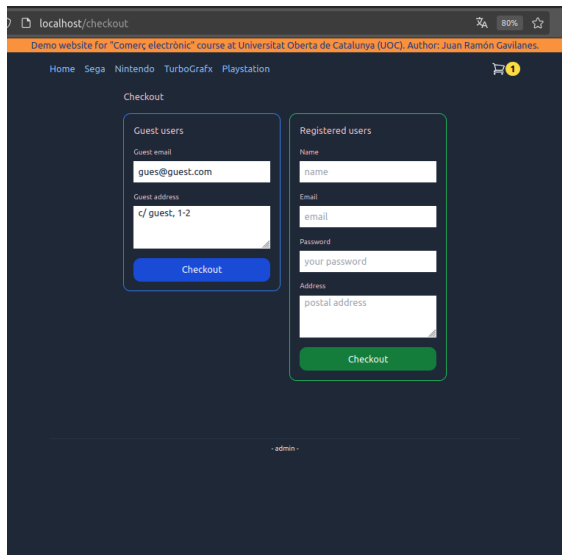


Figura 22: Compra realizada por usuario invitado 1/3

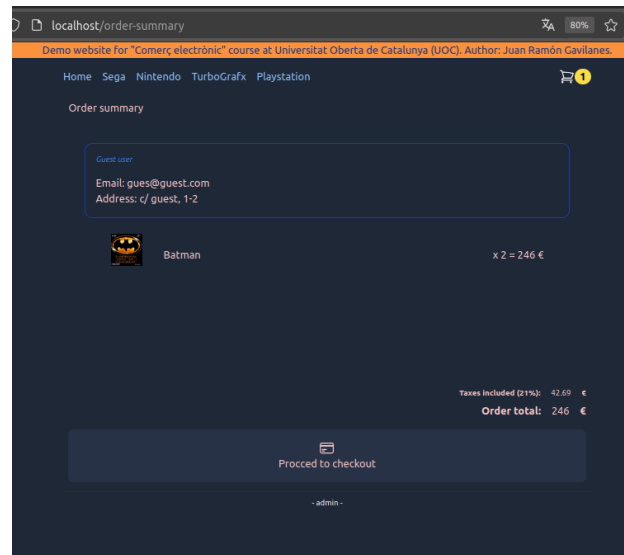


Figura 23: Compra realizada por usuario invitado 2/3.
Al no estar registrado, el campo nombre no se muestra

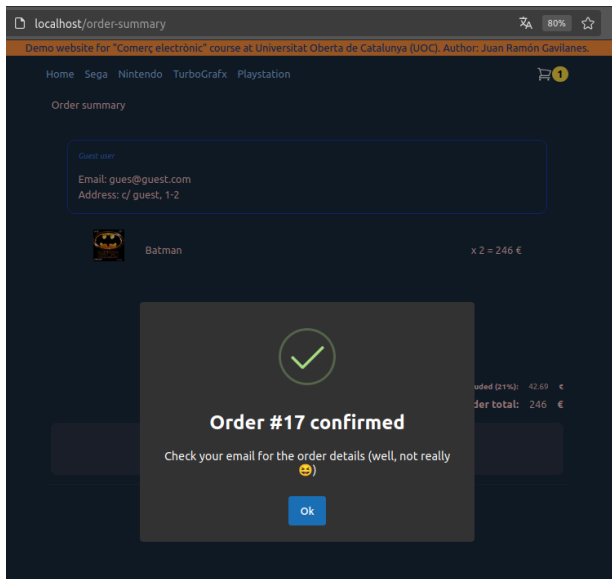


Figura 24: Compra realizada por usuario invitado 3/3

id	user_id	email	address	deleted_at	created_at	updated_at
17	[NULL]	gues@guest.com	c/ guest, 1-2	[NULL]	2024-11-10 11:39:28	2024-11-10 11:39:28

Figura 25: no se asocia a ningún usuario registrado.
Sólo se guarda el email y la dirección indicadas

id	user_id	product_id	quantity	price	deleted_at	created_at	updated_at
23	17	20	2	123	[NULL]	2024-11-10 11:39:28	2024-11-10 11:39:28

Figura 26: Detalle del pedido.
Es igual tanto para usuarios registrados como invitados.

Compra usuario registrado

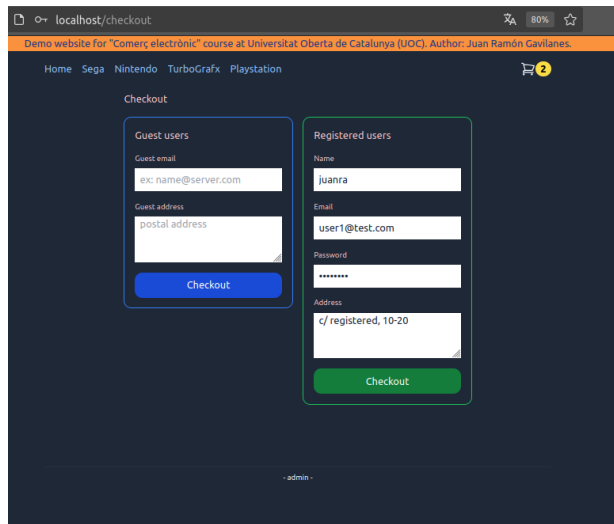


Figura 27: Compra realizada por usuario registrado

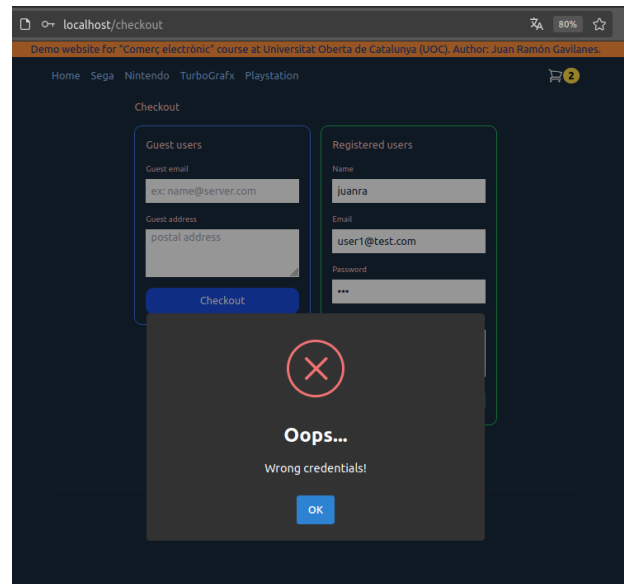


Figura 28: Compra realizada por usuario registrado
Con credenciales erróneas

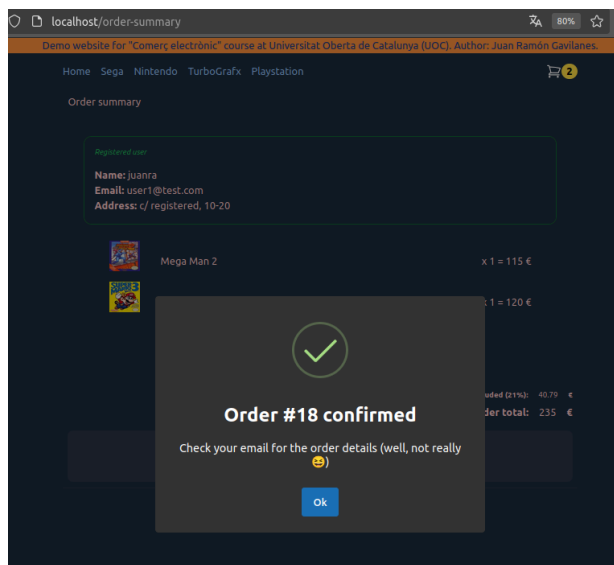


Figura 29: Compra realizada por usuario registrado.
Observa como se muestra el nombre.

	id	user_id	email	address	deleted_at	created_at	updated_at
1	17	(NULL)	guest@server.com	c/ guest, 1-2	(NULL)	2024-11-10 11:39:28	2024-11-10 11:39:28
2	18	2	user1@test.com	c/ registered, 10-20	(NULL)	2024-11-10 11:59:18	2024-11-10 11:59:18

Figura 30: La nueva compra se asocia al usuario 2
Se guarda la dirección indicada

	id	order_id	product_id	quantity	price	deleted_at	created_at	updated_at
1	28	17	15	2	123	(NULL)	2024-11-10 11:39:28	2024-11-10 11:39:28
2	29	18	15	1	115	(NULL)	2024-11-10 11:59:18	2024-11-10 11:59:18
3	30	18	18	1	120	(NULL)	2024-11-10 11:59:18	2024-11-10 11:59:18

Figura 26: Detalle del pedido.
Es igual tanto para usuarios registrados como invitados.

Punto 5. Panel de administración para consultar pedidos realizados.

Para entrar en el panel de administración, el usuario debe loguearse, y tratarse de un usuario con permisos de administración. Debe tener el campo `is_admin = TRUE` en la tabla `users`.

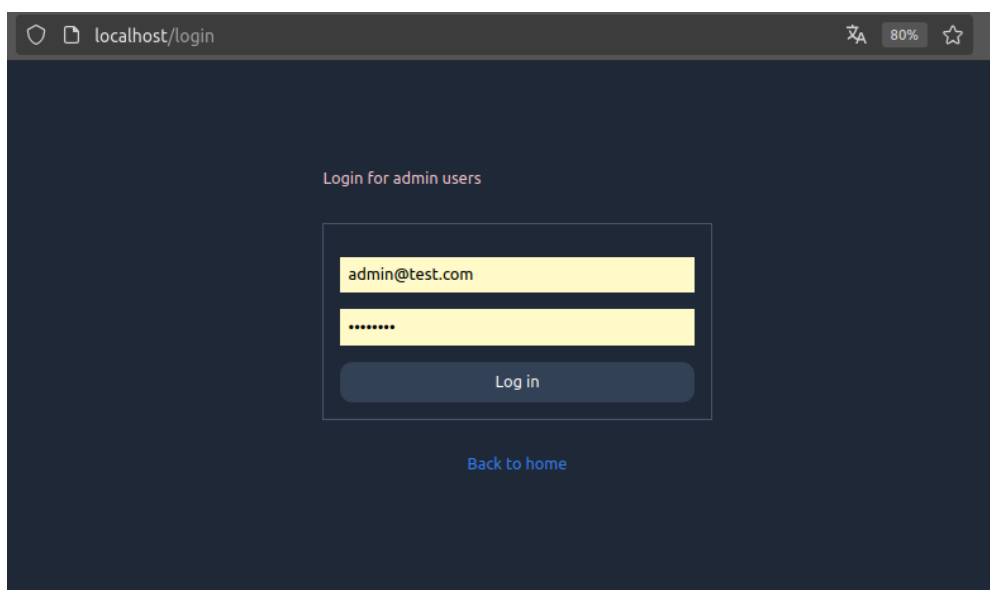


Figura 27: Login usuario administrador.

Recuerda que en nuestro datos, para este usuario la contraseña es **password**

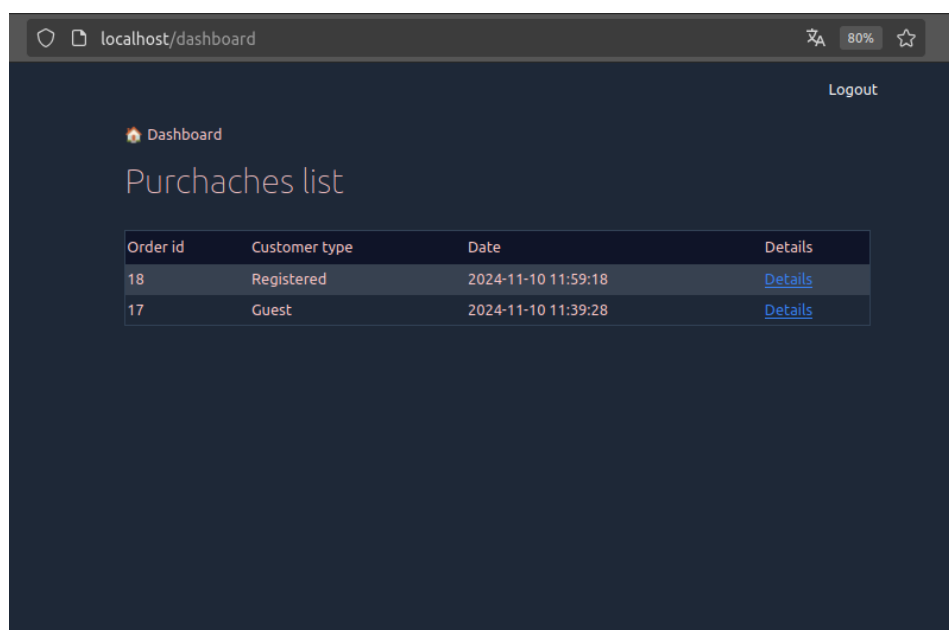


Figura 28: Listado de compras realizadas

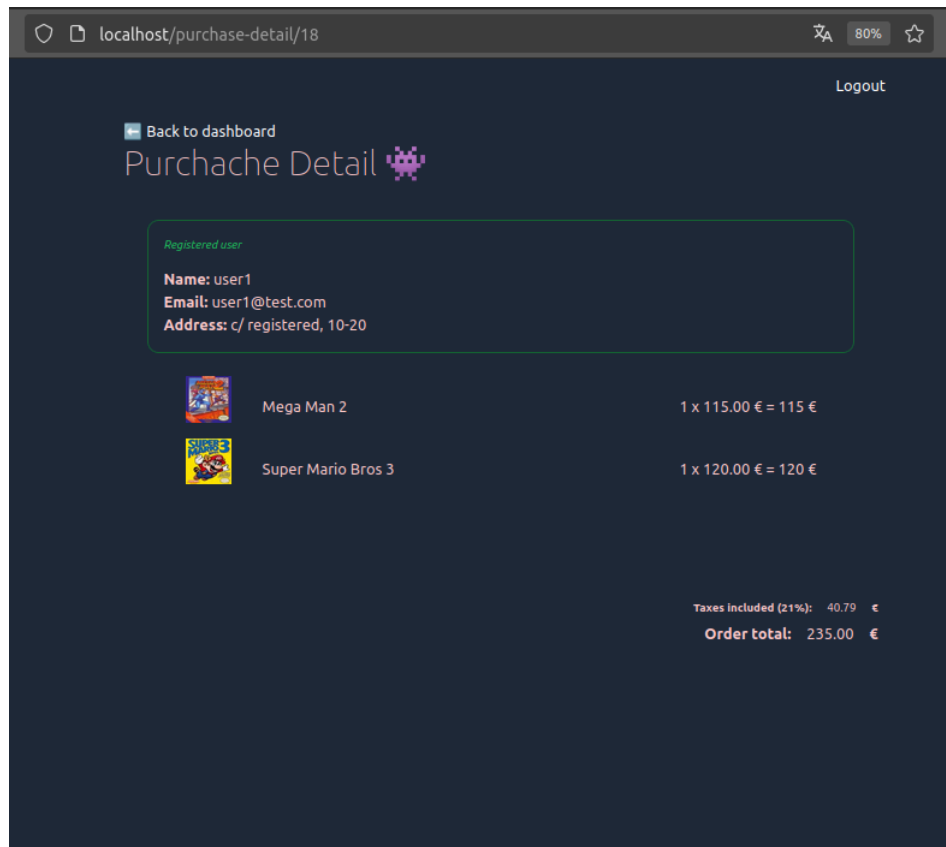


Figura 29: Detalle compra de usuario registrado

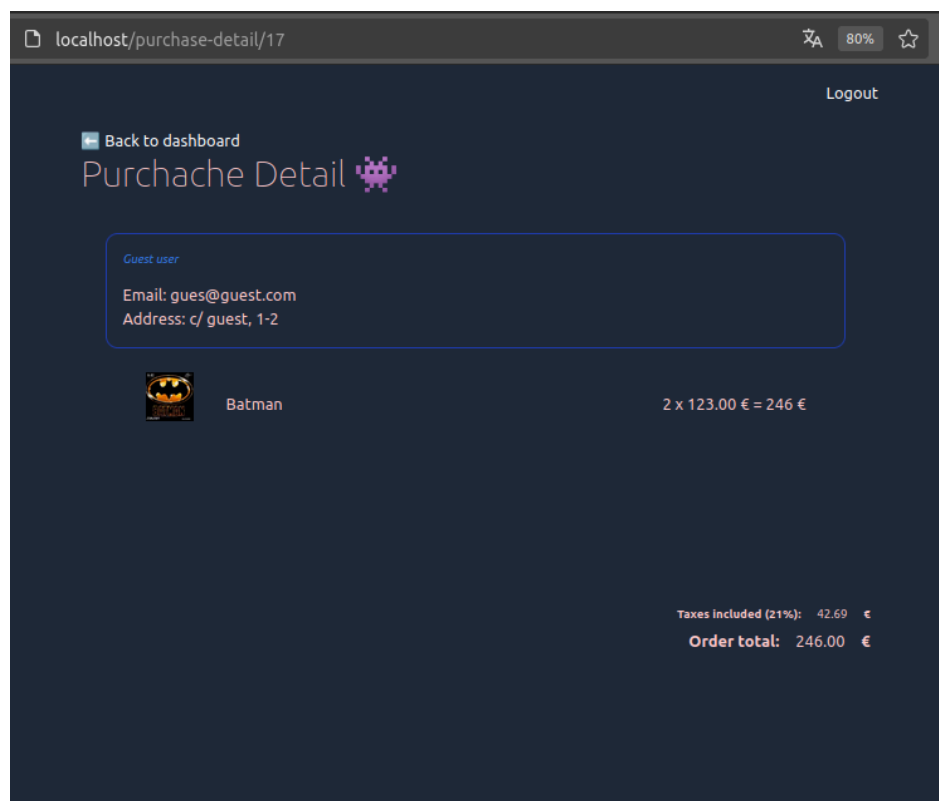


Figura 30: Detalle compra de usuario invitado

Bibliografía

Laravel. (2024). *Documentación oficial de Laravel*. Recuperado de <https://laravel.com/docs>

Docker. (2024). *Documentación oficial de Docker*. Recuperado de <https://docs.docker.com>

Docker. (2024). *Documentación de Docker Compose*. Recuperado de <https://docs.docker.com/compose>

PHP.net. (2024). *Manual de PHP*. Recuperado de <https://www.php.net/manual/es/>

PHP: The Right Way. (s.f.). *PHP: The Right Way*. Recuperado de <https://phptherightway.com>

DBeaver. (2024). *Documentación oficial de DBeaver*. Recuperado de <https://dbeaver.com/>