

## MEMORIA FINAL DE PROYECTO

PalomitasTime



## CICLO FORMATIVO DE GRADO SUPERIOR

**Desarrollo de Aplicaciones Web**

### AUTORES

Jorge Herrera  
Lorena Moreno  
Emanuel Suca

### TUTOR

José Luis Gallego

### COORDINADOR

José Luis Gallego



## **Licencia**

Esta obra está bajo una licencia Reconocimiento-Compartir bajo la misma licencia 3.0 España de Creative Commons. Para ver una copia de esta licencia, visite <http://creativecommons.org/licenses/by-sa/3.0/es/> o envíe una carta a Creative Commons, 171 Second Street, Suite 300, San Francisco, California 94105, USA.

## INDICE

1.. INTRODUCCIÓN.....	7
2.. ALCANCE DEL PROYECTO .....	8
2.1. ANÁLISIS PREVIO.....	9
3.. ESTUDIO DE VIABILIDAD.....	12
3.1.. ESTADO ACTUAL DEL SISTEMA.....	12
3.2.. RESUMEN DE REQUISITOS DEL CLIENTE .....	13
3.3.. POSIBLES SOLUCIONES.....	13
3.4.. SOLUCIÓN ELEGIDA.....	13
4.. ANÁLISIS.....	15
4.1.. MODELO DE DATOS.....	21
4.2.. DEFINICIÓN DE TABLAS.....	22
4.3.. Requisitos funcionales .....	38
4.4.. Requisitos no funcionales .....	40
5.. DISEÑO.....	42
5.1.. ESTRUCTURA DE LA APLICACIÓN.....	42
5.2.. Componentes del sistema / arquitectura de red.....	45
5.3.. Herramientas y tecnologías utilizadas. ....	48
6.. IMPLEMENTACIÓN.....	53
6.1.. Implementación del modelo de datos.....	53
6.2.. Carga de datos.....	53
6.3.. Configuraciones realizadas en el sistema .....	53
6.4.. Implementaciones de código realizadas.....	58
7.. PRUEBAS .....	63
7.1.. Casos de pruebas.....	63
8.. EXPLOTACIÓN.....	108
8.1.. Planificación .....	108
8.2.. Preparación para el cambio.....	108
8.3.. Manual de usuario .....	109
8.4.. Manual de administrador.....	115
8.5.. Implantación propiamente dicha .....	149
9..DEFINICIÓN DE PROCEDIMIENTOS DE CONTROL Y EVALUACIÓN .....	152
10.. CONCLUSIONES .....	158
11.. FUENTES.....	159
1.Legislación .....	159
11.1.Bibliografía .....	160

## **INDICE de tablas y figuras**

Tabla 1: Detalle de funcionalidades y prioridades (1/3)  
Tabla 2: Detalle de funcionalidades y prioridades (2/3)  
Tabla 3: Detalle de funcionalidades y prioridades (3/3)  
Tabla 4: Definición de la tabla administradores.  
Tabla 5: Definición de la tabla butacas.  
Tabla 6: Definición de la tabla butacas\_reservadas.  
Tabla 7: Definición de la tabla categorías.  
Tabla 8: Definición de la tabla facturas.  
Tabla 9: Definición de la tabla ingredientes.  
Tabla 10: Definición de la tabla ingrediente\_producto.  
Tabla 11: Definición de la tabla lineas\_venta  
Tabla 12: Definición de la tabla menus.  
Tabla 13: Definición de la tabla menu\_producto.  
Tabla 14: Definición de la tabla migrations.  
Tabla 15: Definición de la tabla password\_resets  
Tabla 16: Definición de la tabla peliculas.  
Tabla 17: Definición de la tabla plantillas\_sesiones.  
Tabla 18: Definición de la tabla plantilla\_sesion\_sesion\_vacia.  
Tabla 19: Definición de la tabla productos.  
Tabla 20: Definición de la tabla resenas.  
Tabla 21: Definición de la tabla salas.  
Tabla 22: Definición de la tabla sesiones.  
Tabla 23: Definición de la tabla sesion\_vacias.  
Tabla 24: Definición de la tabla users.  
Tabla 25: Definición de la tabla users\_verifications.  
Tabla 26: Cobertura de pruebas.

Figura 1. Error de la aplicación de Cinesa.  
Figura 2: Diagrama de casos de uso.  
Figura 3: Modelo de dominio.  
Figura 4: Diagrama de flujo de una SPA.  
Figura 5: Fichero .env.  
Figura 6: Fichero composer.json.  
Figura 7: Fichero package.json.  
Figura 8: Manual de usuario (1/12).  
Figura 9: Manual de usuario (2/12).  
Figura 10: Manual de usuario (3/12).  
Figura 11: Manual de usuario (4/12).  
Figura 12: Manual de usuario (5/12).  
Figura 13: Manual de usuario (6/12).  
Figura 14: Manual de usuario (7/12).  
Figura 15: Manual de usuario (8/12).  
Figura 16: Manual de usuario (9/12).  
Figura 17: Manual de usuario (10/12).  
Figura 18: Manual de usuario (11/12).  
Figura 19: Manual de usuario (12/12).  
Figura 20: Manual de administrador. Administradores (1/7).  
Figura 21: Manual de administrador. Administradores (2/7).

- Figura 22: Manual de administrador. Administradores (3/7).  
Figura 23: Manual de administrador. Administradores (4/7).  
Figura 24: Manual de administrador. Administradores (5/7).  
Figura 25: Manual de administrador. Administradores (6/7).  
Figura 26: Manual de administrador. Administradores (7/7).  
Figura 27: Manual de administrador. Cine (1/28).  
Figura 28: Manual de administrador. Cine (2/28).  
Figura 29: Manual de administrador. Cine (3/28).  
Figura 30: Manual de administrador. Cine (4/28).  
Figura 31: Manual de administrador. Cine (5/28).  
Figura 32: Manual de administrador. Cine (6/28).  
Figura 33: Manual de administrador. Cine (7/28).  
Figura 34: Manual de administrador. Cine (8/28).  
Figura 35: Manual de administrador. Cine (9/28).  
Figura 36: Manual de administrador. Cine (10/28).  
Figura 37: Manual de administrador. Cine (11/28).  
Figura 38: Manual de administrador. Cine (12/28).  
Figura 39: Manual de administrador. Cine (13/28).  
Figura 40: Manual de administrador. Cine (14/28).  
Figura 41: Manual de administrador. Cine (15/28).  
Figura 42: Manual de administrador. Cine (16/28).  
Figura 43: Manual de administrador. Cine (17/28).  
Figura 44: Manual de administrador. Cine (18/28).  
Figura 45: Manual de administrador. Cine (19/28).  
Figura 46: Manual de administrador. Cine (20/28).  
Figura 47: Manual de administrador. Cine (21/28).  
Figura 48: Manual de administrador. Cine (22/28).  
Figura 49: Manual de administrador. Cine (23/28).  
Figura 50: Manual de administrador. Cine (24/28).  
Figura 51: Manual de administrador. Cine (25/28).  
Figura 52: Manual de administrador. Cine (26/28).  
Figura 53: Manual de administrador. Cine (27/28).  
Figura 54: Manual de administrador. Cine (28/28).  
Figura 55: Manual de administrador. Tienda (1/23).  
Figura 56: Manual de administrador. Tienda (2/23).  
Figura 57: Manual de administrador. Tienda (3/23).  
Figura 58: Manual de administrador. Tienda (4/23).  
Figura 59: Manual de administrador. Tienda (5/23).  
Figura 60: Manual de administrador. Tienda (6/23).  
Figura 61: Manual de administrador. Tienda (7/23).  
Figura 62: Manual de administrador. Tienda (8/23).  
Figura 63: Manual de administrador. Tienda (9/23).  
Figura 64: Manual de administrador. Tienda (10/23).  
Figura 65: Manual de administrador. Tienda (11/23).  
Figura 66: Manual de administrador. Tienda (12/23).  
Figura 67: Manual de administrador. Tienda (13/23).  
Figura 68: Manual de administrador. Tienda (14/23).  
Figura 69: Manual de administrador. Tienda (15/23).  
Figura 70: Manual de administrador. Tienda (16/23).  
Figura 71: Manual de administrador. Tienda (17/23).

## **PalomitasTime**

- Figura 72: Manual de administrador. Tienda (18/23).
- Figura 73: Manual de administrador. Tienda (19/23).
- Figura 74: Manual de administrador. Tienda (20/23).
- Figura 75: Manual de administrador. Tienda (21/23).
- Figura 76: Manual de administrador. Tienda (22/23).
- Figura 77: Manual de administrador. Tienda (23/23).
- Figura 78: Implementación.
- Figura 79: Implementación.
- Figura 80: Implementación.
- Figura 81: Control de evaluación.
- Figura 82: Control de evaluación.
- Figura 83: Control de evaluación.
- Figura 84: Control de evaluación.
- Figura 85: Control de evaluación.
- Figura 86: Control de versiones.

## 1. INTRODUCCIÓN

Este documento recoge el trabajo realizado para el **módulo de Proyecto** del CFGS en Desarrollo de Aplicaciones Web.

Este módulo profesional complementa la formación establecida para el resto de los módulos profesionales que integran el título en las funciones de **análisis** del contexto, **diseño** o **desarrollo** del proyecto y organización de la **ejecución**.

## 2. ALCANCE DEL PROYECTO

El proyecto a desarrollar es una aplicación web para un cine en la que el usuario pueda acceder a la información de películas, compra de entradas con los menos pasos posibles y de forma intuitiva y, también, a los productos que ofrece el cine para consumir en sus salas durante la proyección de la película.

Es importante entender que esta aplicación tiene la finalidad de mejorar los servicios ofrecidos por un negocio, pero no es un negocio por sí misma. Por ejemplo, el negocio de Facebook es su página web (o, al menos originariamente), si no hay página web, no hay Facebook. En nuestro caso el negocio es el cine, su establecimiento e instalaciones, esta aplicación es un servicio adicional que ofrece el cine, pero el negocio (el cine) sigue existiendo sin necesidad de ninguna aplicación web.

La definición de esta aplicación se puede resumir en una tienda online de entradas de cine y productos.

El fin de este proyecto es:

- Reducir las colas en los mostradores de ventas de entradas del cine: los clientes pueden realizar la compra desde cualquier dispositivo con acceso a internet en cualquier momento.
- Reducir las colas en los accesos de las salas: al realizar la compra el cliente recibe, en la dirección de correo que introduzcan en su registro, un email con un código bidi u otro sistema que el cliente escoja. El acomodador podrá validar este código con los lectores electrónicos sin tener que revisar que la hora, la sesión y la sala son las correctas en cada entrada.
- Aumentar la venta de productos en las tiendas: muchas veces los clientes llegan al cine con el tiempo justo para comprar el menú de palomitas de siempre o, incluso, para entrar directamente en la sala sin pasar por la tienda. En la página web pueden ver sin límite de tiempo todos los productos que ofrece el cine y su precio. De esta forma los clientes pueden entrar en la tienda sabiendo qué artículos hay, y qué es lo que quieren tomar de forma rápida y eficaz. También pueden realizar la compra desde la aplicación directamente y recogerlo en la tienda.
- Reducir las colas en las tiendas de productos: Los clientes que realicen la compra de productos a través de la aplicación sólo tienen que mostrar el código del ticket recibido para recoger su compra. De este modo en la caja se reduce el tiempo de cobro por cliente al no ser necesario el proceso de pago por tarjeta ni en efectivo.

## 2.1. Análisis previo.

La ambición de este proyecto y la limitación de tiempo hacen que la probabilidad de no completarlo al 100% sea alta. En la siguiente tabla se establecen unos puntos mínimos que cumplir y sus prioridades, que hemos dividido en objetos de negocio. Entendemos como cliente a la empresa que compra o está interesada en nuestra aplicación, usuario al visitante de la parte pública de la aplicación (cartelera, restaurante, registro, etc.) y administrador como al usuario registrado que tiene acceso a la parte privada de la aplicación en la que se gestiona el contenido de la misma.

FUNCIONALIDAD		PRIORIDAD	MOTIVO
PELÍCULA / SESIONES	Mostrar	Alta	Esta es la funcionalidad más importante de la aplicación pues es su principal fin.
	Crear	Alta	Todas las semanas se estrenan películas, esta es la tarea que se realizará con más frecuencia.
	Modificar	Baja	La información de las películas es siempre la misma. La sinopsis, el título, la duración, etc. no cambian una vez que se ha estrenado. Añadimos esta funcionalidad para poder corregir erratas, pero esta información viene de una base de datos especializada y es poco probable que tenga errores.
	Borrar	Baja	Esta funcionalidad no la consideramos realmente necesaria, pues es más interesante mantener cierta información (aunque la película ya no esté en cartelera) para poder realizar estadísticas en un futuro.
RESEÑA	Crear	Media	El propósito de esta funcionalidad es obtener la interacción de los usuarios. No es necesaria para la finalidad principal de la página, pero creemos que puede hacerla más atractiva y aumentaría el número de visitas.
	Mostrar	Media	El usuario crea una reseña con el propósito de que sea vista.
	Modificar	Media	El usuario sólo puede realizar una reseña por película, pero puede modificarla si lo desea.
	Borrar	Media	La reseña pertenece al usuario, debería poder eliminarla si lo desea.
	Control	Baja	Con esta funcionalidad nos referimos a utilizar filtros y/o poder bloquear reseñas inapropiadas. Hemos establecido prioridad baja pues el uso o no de estos filtros y su nivel preferimos que sean establecidos por el cliente, por lo que serían desarrollados una vez realizada su venta, en el proceso de personalización y adaptación de la aplicación.

Tabla 1: Detalle de funcionalidades y prioridades (1/3)

FUNCIONALIDAD		PRIORIDAD	MOTIVO
SALA / BUTACA	Crear	Baja	Un cine tiene las salas que tiene, en muy contadas ocasiones se crean más.
	Mostrar	Alta	En esta funcionalidad hay que distinguir entre mostrar la sala a los usuarios en la aplicación y mostrar la sala a los administradores en el panel de administrador. En el primer caso optamos por mostrar gráficamente sólo la sala de la sesión que está consultando el usuario. En el segundo hemos considerado más cómodo mostrar la información de las salas en forma de tablas, así ocupan menos espacio y se puede ver más cantidad de información en la pantalla.
	Modificar	Media	Las probabilidades de modificar una sala son más altas que la de crear una nueva, por eso hemos aumentado un poco su prioridad, pero tampoco es frecuente que se realicen obras tan drásticas en las salas de cine como para ser necesario modificar la información que almacenamos sobre ellas.
	Modificar	Alta	En este caso si vemos importante que el administrador pueda bloquear butacas para impedir su venta, pues puede hacerlo tanto porque han sido vendidas para un evento como porque su estado no es apto para su venta y debe ser bloqueada hasta su reparación.
	Borrar	Baja	Al igual que crear salas, cerrarlas también es muy poco probable. En ese caso el administrador tiene la posibilidad de no crear sesiones en esa sala. Las salas sin sesiones nunca llegan al usuario.
	Productos	Crear	Esta es la funcionalidad que nos diferencia del resto de aplicaciones en este sector.
	Modificar	Alta	Aunque no haga con demasiada frecuencia, es importante poder modificar la información (sobre todo el precio) que se muestra de los productos.
	Mostrar	Alta	Obviamente, si el usuario no tiene acceso a los productos no podrá comprarlos.
	Borrar	Baja	Aunque un producto no esté en venta es interesante conservar su información para la realización de estadísticas.
	Menús	Crear	Esta funcionalidad también es importante, pero consideramos que es un punto que se puede ofrecer al cliente que compre la aplicación como adaptable a sus necesidades, por lo que nos limitaremos a tener una muestra de su funcionamiento, pero no una versión final.
	Modificar	Media	Los menús suelen variar con más frecuencia que los productos individuales, pero al tratarse de una muestra de funcionalidad nos limitaremos a realizarlo de forma sencilla.
	Mostrar	Media	Al igual que los productos, si el usuario no ve los menús, no puede comprarlos.
	Borrar	Baja	Recomendamos no usar esta funcionalidad y almacenar la información para estadísticas.

Tabla 1: Detalle de funcionalidades y prioridades (2/3)

FUNCIONALIDAD		PRIORIDAD	MOTIVO
Categorías	Crear	Alta	Es básico para una buena experiencia de usuario.
	Modificar	Media	Esta funcionalidad también puede ser personalizada a las necesidades de cada cliente, por eso realizaremos solo una muestra.
	Mostrar	Alta	Es básico para una buena experiencia de usuario.
	Borrar	Baja	Es interesante mantener para realizar estadísticas.
Ingredientes			Los ingredientes tienen la finalidad de informar sobre alérgenos. Esta información es obligatoria, pero consideramos que la forma de informarla debería establecerla el cliente (si lo quiere más detallado, sólo con iconos y leyenda, etc.), pues los requisitos mínimos que establece la ley varían en función del tipo de producto y establecimiento.
	Crear	Media	
	Modificar	Baja	Los alérgenos son siempre iguales.
	Mostrar	Media	La Ley de Información Alimentaria (Alérgenos), recogida en el Reglamento Europeo 1169/2011 establece que todas las empresas operadoras de colectividades tendrán que informar de los alérgenos que contengan sus platos.
	Borrar	Baja	Recomendamos no usar esta funcionalidad y almacenar la información para estadísticas.
Usuarios			Al poder registrarse, es posible ofrecer al usuario servicios que mejoren su experiencia como ofertas, promociones, servicio de fidelización... así como acceso a la información de todas sus compras y de los datos que almacenamos sobre él.
	Crear	Alta	
	Modificar	Alta	El usuario es dueño de sus datos y debe poder decidir sobre ellos
	Mostrar	Alta	El usuario debe poder acceder a sus datos.
	Borrar	Baja	Esta funcionalidad es adaptable a las necesidades del cliente. Como siempre, recomendamos que no se eliminen por completo los datos, en este caso sugerimos que el usuario pueda bloquear su cuenta pero no eliminarla del todo. La decisión final la tomará el cliente pues depende de la política de privacidad y uso de datos que quieran ofrecer.
Admin			De forma 'hard-code' se genera una cuenta de 'super-administrador' que no se puede eliminar. Sólo este 'super-administrador' puede crear nuevas cuentas de administrador.
	Crear	Alta	
	Modificar	Alta	Es recomendable modificar con relativa frecuencia ciertos datos.
	Mostrar	Alta	Este apartado también lo establece el cliente. Recomendamos que sólo el 'super-administrador' pueda ver cierta información de los demás administradores.
	Borrar	Alta	Al ser una parte sensible, en este caso si recomendamos eliminar por completo los administradores y su información que no sea necesaria.

Tabla 1: Detalle de funcionalidades y prioridades (3/3)

### 3. ESTUDIO DE VIABILIDAD

En esta fase se considera si el proyecto se puede realizar teniendo en cuenta las circunstancias internas y externas de la empresa, las diferentes soluciones posibles y los recursos de los cuales se dispone.

Para ello se hace una valoración del estado actual del sistema y de los requisitos del cliente, se presentará un estudio de soluciones alternativas y la solución elegida por el cliente.

#### 3.1. Estado actual del sistema

Actualmente sólo las grandes cadenas de cine como Cinesa, Yelmo o Kinépolis disponen de una web propia para la venta de entradas. Unifican todos sus cines en la misma aplicación (y normalmente no funcionan de forma eficiente).

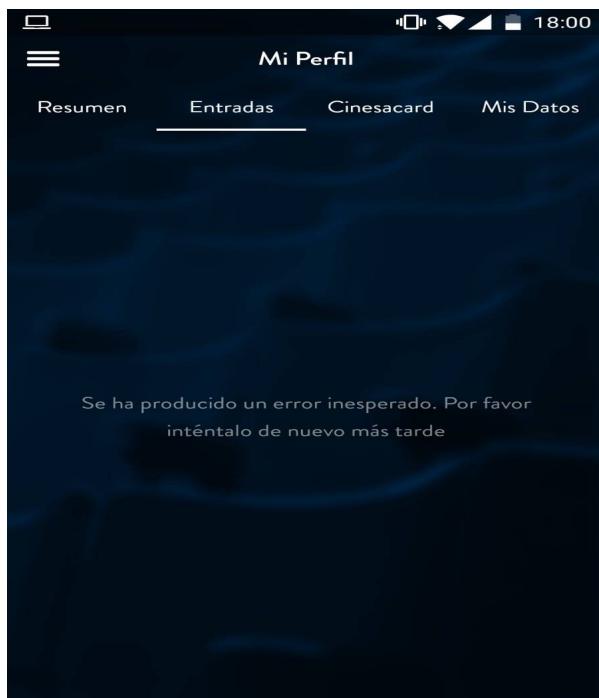


Figura 1. Error de la aplicación de Cinesa.

La aplicación de Cinesa da un error al mostrar tus entradas compradas online, nunca ha funcionado este servicio desde su salida en septiembre del 2017.

Los cines independientes como Callao pueden tener una página web en la que muestran su cartelera y eventos, pero la venta de sus entradas se realiza por intermediarios como [www.reservaentradas.com](http://www.reservaentradas.com) o [www.sensacine.com](http://www.sensacine.com).

En ningún caso, estas aplicaciones incluyen la venta de productos comestibles para el cine. Los llamados *candys* (tiendas dentro del cine) son los que distribuyen palomitas, golosinas,

bebidas, perritos, nachos, etc. Entre sus mayores desventajas incluyen el abarrotamiento momentáneo de los clientes durante los minutos anteriores al comienzo de una sesión. Esto se traduce en colas muy largas, mucho personal encargándose de la misma tarea durante un breve lapso de su jornada (cobrando los productos y limpiando el espacio de palomitas y bebida derramadas) y frustración para el cliente y los trabajadores.

Obviamente al no mostrar los productos en sus aplicaciones o webs, tampoco se pueden comprar los mismos con antelación.

### **3.2. Resumen de requisitos del cliente**

El cliente busca una forma de vender sus entradas de forma online, eliminando los intermediarios que encarecen su producto a causa de comisiones. A su vez se pretende agilizar de algún modo la venta de los productos comestibles que tienen en su *candy*.

La venta de entradas online precisa del bloqueo de butacas durante la compra para no generar duplicidades. Por ello necesita una forma de integrarse con la aplicación interna de venta de entradas que ya posean en las taquillas físicas.

La venta de productos comestibles necesitaría una forma de gestionarlos de manera online, mostrando los menús de los que disponen en su tienda física.

### **3.3. Posibles soluciones**

Las soluciones actuales pasan por el uso de webs intermedias para la venta de entradas y ningún tipo de servicio externo para la venta de productos comestibles, a excepción de la contratación de más personal durante los estrenos más populares que provoquen mayores colas en los *candys*.

### **3.4. Solución elegida**

Se realizará una *single page application*, una web personalizada para el cine, que incluya, además de los próximos estrenos, una tienda online de productos consumibles en la sala. En cuanto a los estrenos, los usuarios obtendrán información de las películas (sinopsis, género e información sobre la dirección y los actores más relevantes). Además, los usuarios podrán generar reseñas y opiniones sobre la película, de forma que se genere una comunidad alrededor de la aplicación que la enriquezca.

En cuanto a los productos, se mostrará una selección apropiada para su consumo en el cine (palomitas, perritos, nachos, etc.) divididos en menús o compra simple. Así, estos se adquieren con antelación y estarán disponibles para su recogida en caja sin necesidad de

### **PalomitasTime**

sacar la cartera, eliminando las esperas que se generan al cobrar en tarjeta o efectivo y reduciendo estrés al consumidor y al personal del cine.

## 4. ANÁLISIS

Para poder sacar el máximo partido a esta aplicación es necesario que el cine disponga de aparatos de lectura de códigos y un software interno compatible con la aplicación para que no se produzcan problemas de reservas dobles de butacas.

Si la empresa posee un dominio, este debe disponer de servidor web y de correo. En caso contrario nosotros podemos encargarnos de contratarlo incluyendo los costes en la facturación.

### Diagrama de casos de uso.

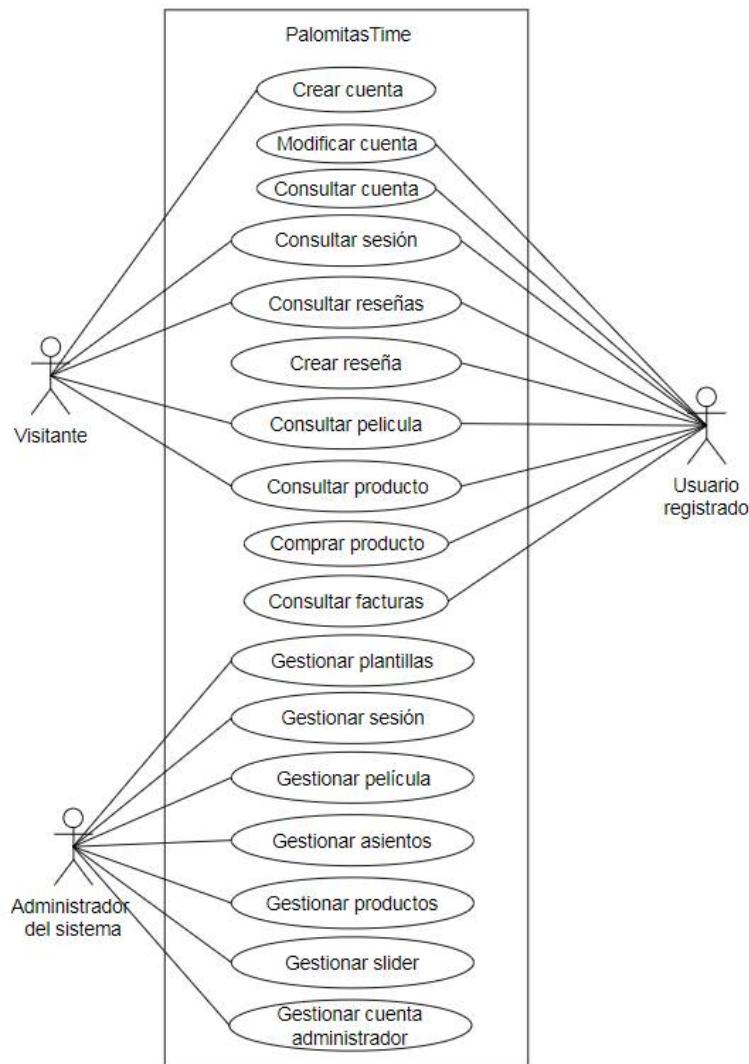


Figura 2: Diagrama de casos de uso.

## **PalomitasTime**

### Caso de uso (UC001).

Escenario de éxito: El visitante crea una cuenta.

Actores: Visitante no registrado.

Precondición: El visitante no debe estar autenticado en el sistema.

1. El visitante indica al sistema que quiere registrarse.
2. El sistema muestra el formulario de registro.
3. El visitante rellena el formulario de registro.
4. El sistema muestra un resumen con los datos introducidos.
5. El visitante confirma los datos.
6. El sistema registra los datos introducidos.

Extensiones:

- 3a. La contraseña introducida no cumple los requisitos de seguridad.
  1. El sistema indica que la contraseña introducida no es válida.
  2. El visitante introduce una nueva contraseña.
  3. Se repiten los puntos 1 y 2 de esta extensión hasta que la contraseña sea válida.
  4. Continúa en el punto 4.
- 3b. La dirección de email introducida ya está registrada.
  1. El sistema indica que el email introducido ya está en uso.
  2. El visitante introduce una nueva dirección de email.
  3. Se repiten los puntos 1 y 2 de esta extensión hasta que la dirección de email sea correcta.
  4. Continúa en el punto 4.
- 4a. El visitante quiere modificar alguno de los datos introducidos.
  1. El visitante indica al sistema que quiere modificar los datos.
  2. El sistema muestra el formulario con los datos introducidos anteriormente por el usuario.
  3. El visitante modifica los datos deseados.
  4. Continúa en el punto donde comenzó esta extensión.

### Caso de uso (UC002).

Escenario de éxito: El cliente compra una entrada.

Actores: Usuario registrado.

Precondición: El usuario está autenticado en el sistema.

1. El sistema muestra todas las películas disponibles en el día de la consulta y posteriores.
2. El usuario selecciona la película deseada.
3. El sistema muestra la sala con las butacas disponibles de la primera sesión disponible de la película.
4. El usuario selecciona las butacas deseadas.
5. El sistema muestra un resumen con la información de la película y butacas seleccionadas.
6. El usuario confirma los datos.
7. El sistema muestra el formulario de pago.
8. El usuario realiza las operaciones solicitadas por el gestor de pagos.
9. El sistema muestra la confirmación del pago.

Extensiones:

- 3a. El usuario quiere comprar entradas para un día distinto.
  1. El usuario selecciona otra fecha.

2. El sistema muestra la sala y butacas disponibles para la primera sesión de ese día.
3. Continúa en el punto 5.
- 4b. El usuario quiere cambiar la sesión.
  1. El usuario selecciona otra sesión.
  2. El sistema muestra la sala y butacas disponibles de la sesión solicitada.
  3. Continúa en el punto 5.
- 7a. El usuario quiere comprar algún producto.
  1. El usuario continúa en el caso de uso UC003.

Caso de uso (UC003).

Escenario de éxito: El cliente compra un producto.

Actores: Usuario registrado.

Precondición: El usuario está autenticado en el sistema.

1. El usuario indica al sistema que quiere comprar un producto.
2. El sistema muestra los productos disponibles.
3. El usuario selecciona el producto deseado e indica la cantidad.
4. El sistema añade el producto al carro de la compra.
5. El usuario indica al sistema que quiere consultar su carro de la compra.
6. El sistema muestra el carro de la compra del usuario.
7. El usuario confirma el carro de la compra.
8. El sistema muestra los productos deseados y las entradas reservadas.
9. El usuario indica al sistema que quiere realizar el pago.
10. El sistema redirecciona al gestor de pagos.
11. El usuario realiza las operaciones solicitadas por el gestor de pagos.
12. El sistema muestra la confirmación del pago.

Extensiones:

- 4a. El usuario quiere comprar más productos.
  1. Repetir los pasos 3 y 4 tantas veces como productos quiere comprar el usuario.
  2. Continúa en el punto 5.
- 6a. El usuario quiere eliminar productos de su carro de la compra.
  1. El usuario selecciona la opción de borrar del producto que quiere eliminar.
  2. El sistema elimina el producto del carro de la compra.
  3. Repetir los puntos 1 y 2 de esta extensión con todos los productos que el usuario quiera eliminar.
  4. Continúa en el punto 7.
- 6b. El usuario quiere cambiar la cantidad de un producto del carro de la compra.
  1. El usuario cambia la cantidad del producto deseado.
  2. El sistema calcula y muestra el nuevo precio.
  3. Repetir los puntos 1 y 2 de esta extensión con todos los productos que el usuario quiera eliminar.
  4. Continúa en el punto 7.
- 6c. El usuario quiere continuar comprando.
  1. El usuario indica al sistema que quiere comprar más productos.
  2. Continúa en el punto 2.
- 6d. El usuario quiere eliminar su carro de la compra.
  1. El usuario indica que quiere eliminar el carro completo de la compra.
  2. El sistema elimina el carro de la compra.

3. Continúa en el punto 2.

Caso de uso (UC004).

Escenario de éxito: El usuario incluye una reseña.

Actores: Usuario registrado.

Precondición: El usuario está autenticado en el sistema.

1. El usuario selecciona una película.
2. El sistema muestra la información de la película.
3. El usuario indica al sistema que quiere añadir una reseña.
4. El sistema muestra el formulario de reseñas.
5. El usuario rellena y confirma el formulario.
6. El sistema registra y muestra la reseña.

Extensiones:

2a. El usuario quiere modificar una reseña escrita.

1. El sistema muestra la reseña escrita sobre esa película y la opción de modificar.
2. El usuario indica al sistema que quiere modificar la reseña.
3. Continúa en el punto 4.

2b. El usuario quiere borrar una reseña escrita.

1. El sistema muestra la reseña escrita sobre esa película y la opción de borrar.
2. El usuario indica al sistema que quiere borrar la reseña.
3. El sistema pide confirmación.
4. El usuario confirma la acción.
5. El sistema borra la reseña.

Caso de uso (UC005).

Escenario de éxito: El usuario quiere modificar su cuenta.

Actores: Usuario registrado.

Precondición: El usuario está autenticado en el sistema.

1. El usuario indica al sistema que quiere ver su perfil.
2. El sistema muestra los datos que el usuario puede modificar.
3. El usuario actualiza los datos.
4. El sistema pide confirmación.
5. El usuario confirma la modificación.

Extensiones:

- 3a. El usuario introduce algún dato que ya existe
  1. El sistema indica que el dato introducido ya está en uso.
  2. El usuario introduce una nuevo dato.
  3. Se repiten los puntos 1 y 2 de esta extensión hasta que el dato sea correcto.
  4. Continúa en el punto 4.

Caso de uso (UC006).

Escenario de éxito: El usuario administrador incluye una película en el sistema.

Actores: Usuario administrador.

Precondición: El usuario debe estar previamente autenticado como administrador.

1. El administrador indica al sistema que quiere incluir una nueva película.
2. El sistema muestra el formulario para buscar información sobre la película.
3. El administrador introduce el título de la película.

4. El sistema muestra una lista de películas con nombre similar y su fecha de estreno.
5. El usuario indica al sistema que película quiere registrar.
6. El sistema muestra los datos de la película seleccionada.
7. El usuario confirma los datos de la película.
8. El sistema registra los datos introducidos e incluye la película registrada.

Extensiones:

- 6a. El administrador quiere modificar alguno de los datos mostrados.
  1. El administrador modifica los datos deseados.
  2. Continúa en el paso 7.
- 5b. La película introducida ya existe en el sistema.
  1. El sistema indica al administrador que la película que quiere introducir ya existe.
  2. Se vuelve al paso número 2.

#### Caso de uso (UC007).

Escenario de éxito: El usuario administrador incluye una nueva sesión en el sistema.

Actores: Usuario administrador.

Precondición: El usuario debe estar previamente autenticado como administrador.

1. El administrador indica al sistema que quiere incluir una nueva sesión.
2. El sistema muestra el formulario para el registro de la nueva sesión.
3. El administrador rellena el formulario de registro de la nueva sesión.
4. El sistema muestra los datos introducidos por el administrador.
5. El administrador confirma los datos.
6. El sistema registra los datos introducidos e incluye el horario registrado.

Extensiones:

- 3a. La sala está ocupada para esa sesión.
  1. El sistema indica al administrador que la sesión que quiere introducir ya existe.
  2. Se vuelve al paso número 2.
- 4a-5a. El administrador quiere modificar alguno de los datos introducidos.
  1. El administrador indica al sistema que quiere modificar los datos.
  2. El sistema muestra el formulario con los datos introducidos anteriormente por el administrador.
  3. Se vuelve al paso 3.

#### Caso de uso (UC008).

Escenario de éxito: El usuario administrador actualiza el estado de un asiento.

Actores: Usuario administrador.

Precondición: El usuario debe estar previamente autenticado como administrador.

1. El administrador selecciona la sala.
2. El sistema muestra todos los asientos de dicha sala.
3. El administrador actualiza el estado del asiento deseado.
4. El sistema pide confirmación.
5. El administrador confirma.
6. El sistema registra el estado.

#### Caso de uso (UC009).

Escenario de éxito: El usuario administrador incluye un producto en el sistema.

Actores: Usuario administrador.

## **PalomitasTime**

Precondición: El usuario debe estar previamente autenticado como administrador.

1. El administrador indica al sistema que quiere incluir un nuevo producto.
2. El sistema muestra el formulario para el registro del nuevo producto.
3. El administrador rellena el formulario de registro del nuevo producto.
4. El sistema muestra los datos introducidos por el administrador.
5. El administrador confirma los datos.
6. El sistema registra los datos introducidos e incluye el producto registrado.

Extensiones:

- 4a-5a. El administrador quiere modificar alguno de los datos introducidos.
  1. El administrador indica al sistema que quiere modificar los datos.
  2. El sistema muestra el formulario con los datos introducidos anteriormente por el administrador.
  3. Se vuelve al paso 3.
- 5b. El producto introducido ya existe en el sistema.
  1. El sistema indica al administrador que el producto que quiere introducir ya existe.
  2. Se vuelve al paso número 2.

### Caso de uso (UC010).

Escenario de éxito: El administrador borra un producto del sistema.

Actores: Usuario administrador.

Precondición: El usuario debe estar previamente autenticado como administrador.

1. El administrador indica al sistema que desea borrar un producto.
2. El sistema muestra una lista de los productos ordenados por un filtro.
3. El administrador indica el producto deseado.
4. El sistema muestra los datos del producto a borrar.
5. El administrador confirma que quiere borrar el producto.
6. El sistema borra el producto.

Extensiones:

- 4a. El administrador se equivocó a la hora de indicar el producto.
  1. El sistema descarta la operación.
  2. Se vuelve al paso 2.
- 6a. El administrador no quería borrar el producto que indicó.
  1. El sistema deshace el borrado volviendo a incluir el producto.
  2. Se vuelve al paso 2.

## 4.1.. MODELO DE DATOS

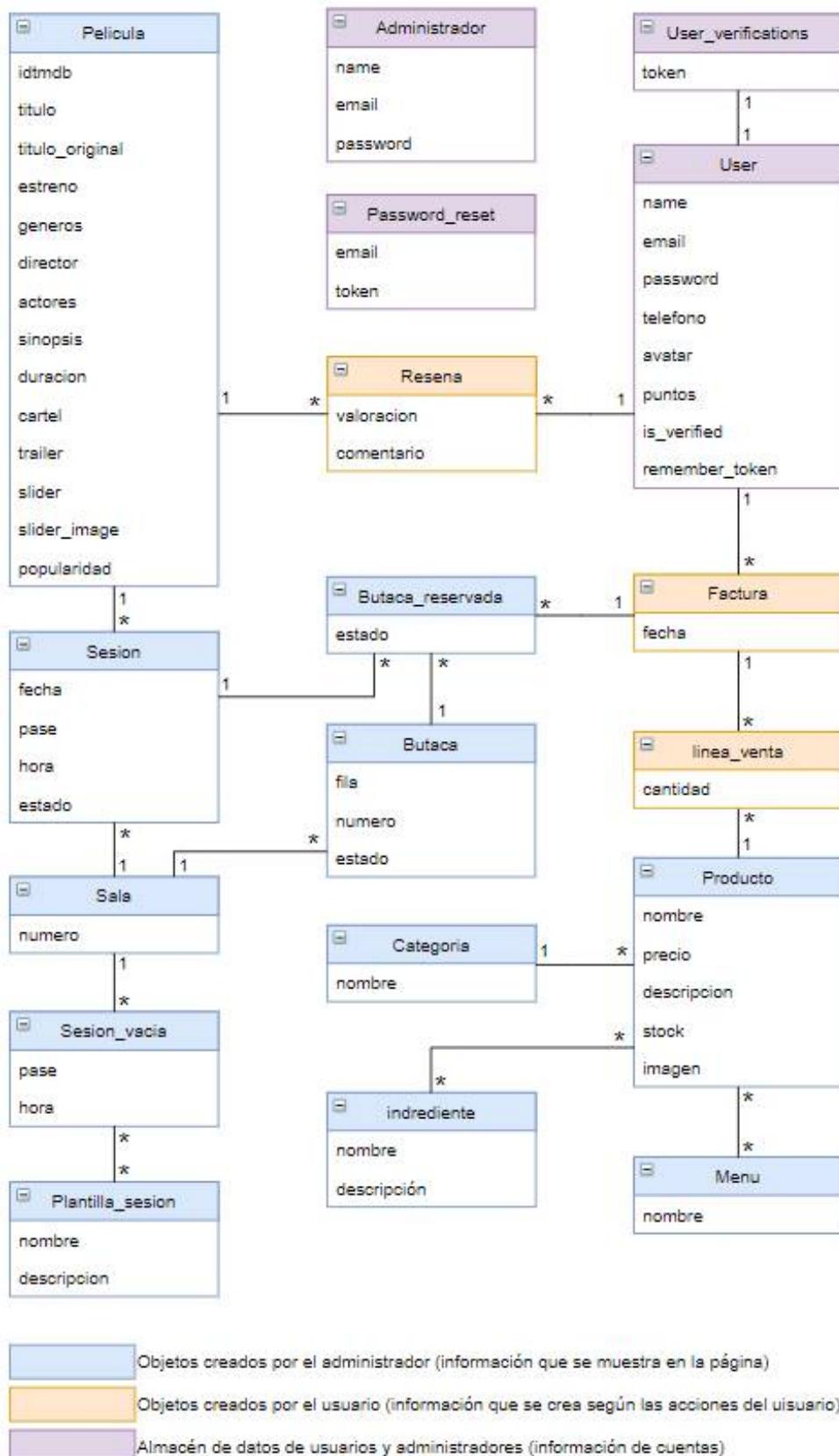


Figura 3: Modelo de dominio.

## 4.2.. DEFINICIÓN DE TABLAS

Definición de tablas ordenadas por orden alfabético.

### Tabla administradores

<b>administradores</b>				
Almacena los datos de los administradores registrados. El administrador con valor de id 1 se crea automáticamente al desplegar la aplicación. Este administrador se considera como súper administrador y no puede ser eliminado. Todos sus datos, a excepción del id pueden ser modificados.				
CAMPO	TIPO	DESCRIPCIÓN	RESTRICCIONES	NULLABLE
id	int(10)	Identificador del administrador. Auto-generado	Primary Key	no
name	varchar(191)	Nombre del administrador.		no
email	varchar(191)	Email del administrador. Utilizado en el login.	Unique	no
password	varchar(191)	Clave del administrador. Encriptado.		no
rememberToken	varchar(100)	Token de sesión.		si
created_at	timestamp	Fecha de creación del administrador.		si
updated_at	timestamp	Fecha de modificación del administrador		si

Tabla 4: Definición de la tabla administradores.

Tabla butacas

butacas				
Contiene la información de las butacas de las salas.				
CAMPO	TIPO	DESCRIPCIÓN	RESTRICCIONES	NULLABLE
id	int(10)	Identificador de la butaca	primary Key	no
fila	int(10)	Fila en la que está la butaca.		no
numero	int(10)	Número de la butaca		no
estado	int(10)	Estado de la butaca	0 = libre, 1 = bloqueada	no
sala_id	int(10)	Identificador de la sala en la que está la butaca.	foreign key SALA	no
created_at	timestamp	Fecha de registro de la butaca.		si
updated_at	timestamp	Fecha de modificación de los datos de la butaca		si

Tabla 5: Definición de la tabla butacas.

Tabla butacas\_reservadas

<b>butacas_reservadas</b>				
Información de las facturas generadas por las ventas a través de la web.				
CAMPO	TIPO	DESCRIPCIÓN	RESTRICCIONES	NULLABLE
id	int(10)	Identificador de la butaca reservada.	primary Key	no
estado	int(10)	Estado de la butaca reservada.	0 = Libre, 1= Vendida, 2= Reservada	no
sesion_id	int(10)	Identificador de la sesión en la que se ha reservado la butaca.	foreign key SALA	no
butaca_id	int(10)	Identificador de la butaca física que se ha reservado.	foreign key BUTACA	no
factura_id	int(10)	Identificador de la factura donde se ha cobrado la reserva.	foreign key FACTURA	si
created_at	timestamp	Fecha de creación de la butaca reservada.		si
updated_at	timestamp	Fecha de modificación de la butaca reservada.		si

Tabla 6: Definición de la tabla butacas\_reservadas.

Tabla categorías

<b>categorías</b>				
Información de las categorías en las que se clasifican los productos.				
CAMPO	TIPO	DESCRIPCIÓN	RESTRICCIONES	NULLABLE
id	int(10)	Identificador de la categoría.	primary Key	no
nombre	string(191)	Nombre de la categoría.		no
created_at	timestamp	Fecha de creación de la categoría.		si
updated_at	timestamp	Fecha de modificación de la categoría		si

Tabla 7: Definición de la tabla categorías.

Tabla facturas

facturas				
Información de las facturas generadas por las ventas a través de la web.				
CAMPO	TIPO	DESCRIPCIÓN	RESTRICCIONES	NULLABLE
id	int(10)	Identificador de la factura.	primary Key	no
fecha	date	Fecha de emisión de la factura.		no
usuario_id	int(10)	Identificador del usuario propietario de la factura.	foreign key USER	no
created_at	timestamp	Fecha de creación de la factura.		sí
updated_at	timestamp	Fecha de modificación de la factura.		sí

Tabla 8: Definición de la tabla facturas.

Tabla ingredientes

ingredientes				
Información sobre los ingredientes que contienen los productos para información sobre alérgenos.				
CAMPO	TIPO	DESCRIPCIÓN	RESTRICCIONES	NULLABLE
id	int(10)	Identificador del ingrediente.	primary Key	no
nombre	varchar(191)	Nombre del ingrediente.	unique	no
descripcion	varchar(191)	Descripción del ingrediente.		no
created_at	timestamp	Fecha de creación del ingrediente.		sí
updated_at	timestamp	Fecha de modificación del ingrediente.		sí

Tabla 9: Definición de la tabla ingredientes.

Tabla ingrediente\_producto

ingrediente_producto				
Tabla intermedia que relaciona los ingredientes y los productos.				
CAMPO	TIPO	DESCRIPCIÓN	RESTRICCIONES	NULLABLE
producto_id	Int(10)	Identificador del producto que contiene el ingrediente.	foreign key PRODUCTO	no
ingrediente_id	Int(10)	Identificador del ingrediente.	foreign key INGREDIENTE	no

Tabla 10: Definición de la tabla ingrediente\_producto.

Tabla lineas\_venta

lineas_venta				
Información de las líneas de venta que contienen las facturas.				
CAMPO	TIPO	DESCRIPCIÓN	RESTRICCIONES	NULLABLE
id	int(10)	Identificador de la línea de venta.	primary Key	no
producto_id	int(10)	Identificador del producto comprado.	foreign key PRODUCTO	no
factura_id	int(10)	Identificador de la factura donde se cobra el producto.	foreign key FACTURA	no
created_at	timestamp	Fecha de creación de la línea de venta.		si
updated_at	timestamp	Fecha de modificación de la línea de venta.		si

Tabla 11: Definición de la tabla lineas\_venta

Tabla menus

menus				
Información sobre los menús.				
CAMPO	TIPO	DESCRIPCIÓN	RESTRICCIONES	NULLABLE
id	int(10)	Identificador del menú.	primary Key	no
nombre	varchar(191)	Nombre del menú.	unique	no
created_at	timestamp	Fecha de creación del menú.		si
updated_at	timestamp	Fecha de modificación del menú.		si

Tabla 12: Definición de la tabla menus.

Tabla menu\_producto

menu_producto				
Tabla intermedia que relaciona cada menú con los productos que contiene.				
CAMPO	TIPO	DESCRIPCIÓN	RESTRICCIONES	NULLABLE
producto_id	Int(10)	Identificador del producto.	foreign key PRODUCTO	no
menu_id	Int(10)	Identificador del menú que contiene el producto.	foreign key MENU	no

Tabla 13: Definición de la tabla menu\_producto.

Tabla migrations

migrations				
Creada por Laravel para registrar las migraciones de las tablas y sus actualizaciones				
CAMPO	TIPO	DESCRIPCIÓN	RESTRICCIONES	NULLABLE
id	int(10)	Identificador de la migración.	primary Key	no
migration	varchar(191)	Nombre del fichero de migración.		no
batch	int(10)	Lote de ejecución de la migración.		no

Tabla 14: Definición de la tabla migrations.

Tabla password\_resets

migrations				
Creada por Laravel para registrar las migraciones de las tablas y sus actualizaciones				
CAMPO	TIPO	DESCRIPCIÓN	RESTRICCIONES	NULLABLE
id	int(10)	Identificador de la migración.	primary Key	no
migration	varchar(191)	Nombre del fichero de migración.		no
batch	int(10)	Lote de ejecución de la migración.		no

Tabla 15: Definición de la tabla password\_resets

Tabla peliculas

peliculas				
Almacena los datos de las películas. Los datos se extraen de la API 'The Movie Database'. Algunos campos pueden haber sido modificados por el administrador.				
CAMPO	TIPO	DESCRIPCIÓN	RESTRICCIONES	NULLABLE
id	int(10)	Identificador de la película.	primary Key	no
idtmdb	int(10)	Identificador de la película en TMDB	unique	no
titulo	varchar(191)	Título en español de la película.		no
titulo_original	varchar(191)	Título original de la película.		si
estreno	date	Fecha de estreno de la película.		si
generos	varchar(191)	Géneros cinematográficos a los que pertenece la película.		si
director	varchar(191)	Director de la película		si
actores	varchar(300)	Los cinco principales actores de la película.		si
sinopsis	varchar(1500)	Sinopsis de la película.		si
duracion	int(10)	Duración en minutos de la película.		si
cartel	varchar(191)	Enlace a la imagen del cartel en TMDB de la película.		si
trailer	varchar(191)	Enlace al vídeo del tráiler en YouTube de la película.		si
slider	boolean	Si se muestra en el slider de la página.	default 0	no
slider_image	varchar(191)	Enlace a la imagen del slider en TMDB de la película.		si
popularidad	decimal(10,6)	Puntos de popularidad de la película.	default 0	no
created_at	timestamp	Fecha de creación del registro.		si
updated_at	timestamp	Fecha de modificación del registro.		si

Tabla 16: Definición de la tabla peliculas.

Tabla plantillas\_sesiones

<b>plantillas_sesiones</b>				
Almacena los datos de las plantillas registradas para programar sesiones				
CAMPO	TIPO	DESCRIPCIÓN	RESTRICCIONES	NULLABLE
id	int(10)	Identificador de la plantilla.	primary Key	no
nombre	varchar(250)	Nombre de la plantilla.		no
descripcion	varchar(250)	Descripción de la plantilla.	default ''	no
created_at	timestamp	Fecha de creación de la plantilla.		si
updated_at	timestamp	Fecha de modificación de la plantilla.		si

Tabla 17: Definición de la tabla plantillas\_sesiones.

Tabla plantilla\_sesion\_sesion\_vacia

<b>plantilla_sesion_sesion_vacia</b>				
Tabla intermedia que relaciona cada menú con los productos que contiene.				
CAMPO	TIPO	DESCRIPCIÓN	RESTRICCIONES	NULLABLE
plantilla_sesion_id	int(10)	Identificador de la plantilla	foreign key PLANTILLA	no
sesion_vacia_id	int(10)	Identificador de la sesión	foreign key SESION	no

Tabla 18: Definición de la tabla plantilla\_sesion\_sesion\_vacia.

Tabla productos

productos				
Información de los productos registrados.				
CAMPO	TIPO	DESCRIPCIÓN	RESTRICCIONES	NULLABLE
id	int(10)	Identificador del producto.	primary Key	no
nombre	varchar(191)	Nombre del producto		no
precio	decimal(5,2)	Precio del producto		no
imagen	varchar(191)	Ruta de la imagen del producto.		no
categoría_id	int(10)	Categoría en la que se clasifica el producto.	foreign key	no
created_at	timestamp	Fecha de creación del producto.		si
updated_at	timestamp	Fecha de modificación del producto.		si

Tabla 19: Definición de la tabla productos.

Tabla reseñas

reseñas				
Almacena las reseñas creadas por los usuarios registrados				
CAMPO	TIPO	DESCRIPCIÓN	RESTRICCIONES	NULLABLE
id	int(10)	Identificador de la reseña.	primary Key	no
valoracion	int(10)	Puntuación que el usuario ha dado sobre la película.		no
comentario	varchar(191)	Mensaje del usuario sobre la película		no
user_id	int(10)	Identificador del usuario propietario de la reseña.	foreign key USER	no
pelicula_id	int(10)	Identificador de la película a la que se refiere la reseña.	foreign key PELICULA	no
created_at	timestamp	Fecha de creación de la reseña.		si
updated_at	timestamp	Fecha de modificación de la reseña.		si

Tabla 20: Definición de la tabla reseñas.

Tabla salas

salas				
Almacena los datos de las salas del cine.				
CAMPO	TIPO	DESCRIPCIÓN	RESTRICCIONES	NULLABLE
id	int(10)	Identificador de la sala.	primary Key	no
numero	int(10)	Número de la sala.	unique	no
created_at	timestamp	Fecha de registro de la sala.		si
updated_at	timestamp	Fecha de modificación de los datos de la sala.		si

Tabla 21: Definición de la tabla salas.

Tabla sesiones

sesiones				
Información de las sesiones programadas.				
CAMPO	TIPO	DESCRIPCIÓN	RESTRICCIONES	NULLABLE
id	int(10)	Identificador de la sesión.	primary Key	no
fecha	date	Día en el que está programada la sesión		no
pase	int(10)	Número del pase de la sesión		no
hora	time	Hora a la que está programada la sesión.		no
estado	int(10)	Estado de la sesión.	0 = Inactiva, 1 = Activa	no
pelicula_id	int(10)	Película que se proyecta en la sesión.	foreign key PELICULA	no
sala_id	int(10)	Sala en la que se realiza la sesión.	foreign key SALA	no
created_at	timestamp	Fecha de creación de la sesión.		sí
updated_at	timestamp	Fecha de modificación de la sesión.		sí

Tabla 22: Definición de la tabla sesiones.

Tabla sesion\_vacia

sesion_vacias				
Almacena los datos de las plantillas registradas para programar sesiones				
CAMPO	TIPO	DESCRIPCIÓN	RESTRICCIONES	NULLABLE
id	int(10)	Identificador de la sesión vacía.	primary Key	no
pase	int(11)	Número de pase en el que se da la sesión.		no
hora	time	Hora de la sesión.		no
sala_id	int(10)	Sala de la sesión.	foreign key SALA	no
created_at	timestamp	Fecha de creación de la sesión.		si
updated_at	timestamp	Fecha de modificación de la sesión.		si

Tabla 23: Definición de la tabla sesion\_vacias.

Tabla users

users				
Almacena los datos de los registrados.				
CAMPO	TIPO	DESCRIPCIÓN	RESTRICCIONES	NULLABLE
id	int(10)	Identificador del usuario. Autogenerado.	Primary Key	no
name	varchar(191)	Nombre del usuario.		no
email	varchar(191)	Email del usuario. Utilizado en el login	Unique	no
password	varchar(191)	Clave del usuario. Encriptado		no
telefono	int(10)	Número de teléfono del usuario	Unique	no
avatar	varchar(191)	Enlace a la imagen que el usuario usa como avatar.	Default= imagen genérica	no
puntos	int(10)	Puntos acumulados por el usuario en sus compras.	Default=0	no
is_verified	boolean	Es false cuando el usuario se registra. Actualiza a true cuando verifica su email. Hasta que no lo verifica no puede loguear.		no
remember_token	varchar(100)	Token de sesión.		si
created_at	timestamp	Fecha de registro del usuario.		si
updated_at	timestamp	Fecha de modificación del usuario.		si

Tabla 24: Definición de la tabla users.

Tabla users\_verifications

user_verifications				
CAMPO	TIPO	DESCRIPCIÓN	RESTRICCIONES	NULLABLE
id	int(10)	Identificador del registro.	primary Key	no
user_id	int(10)	Identificador del usuario.	foreign key USER	no
token	varchar(191)	Token del usuario.		no

Tabla 25: Definición de la tabla users\_verifications.

### **4.3.. Requisitos funcionales**

Son aquellos que determinan qué tareas tiene que hacer el sistema.

#### **Independizar usuarios registrados de la aplicación de administradores.**

Por seguridad y rapidez los usuarios de la aplicación son independientes de los usuarios administradores, cada parte tiene su acceso independiente y en ningún momento se cruzan.

#### **Restringir la creación de nuevos administradores.**

Sólo el súper administrador puede crear nuevas cuentas de administrador, no existe formulario de registro si no está autenticado con esta cuenta.

#### **Restringir el acceso a ciertos datos de los administradores y su manipulación.**

Sólo el súper administrador puede ver toda la información (excepto contraseñas) de los demás administradores y modificarla. El resto de administradores sólo puede ver cierta información, sin saber nunca quién es el súper administrador, y no puede modificar ningún dato que no pertenezca a su cuenta.

#### **Registrar películas de forma eficaz.**

La información registrada de las películas se extrae de una base de datos de cine, así el administrador pierde el mínimo tiempo posible en llenar todos los campos requeridos para que la información de la película sea completa. Puede modificar esta información si lo desea.

#### **Control de eliminación de películas registradas.**

Para evitar errores y problemas, un administrador no puede borrar una película que tiene sesiones activas. Se informa de esto si intenta hacerlo.

#### **Gestionar slider de cabecera.**

El slider muestra las películas más destacadas. El administrador selecciona las películas que quiere que aparezcan hasta un máximo de 3.

#### **Registrar valoraciones de los usuarios sobre las películas.**

Sólo los usuarios registrados pueden opinar sobre una película y sólo pueden dejar una valoración por película. Pueden modificar sus valoraciones y borrarlas.

#### **Mostrar las sesiones de una película.**

Al entrar en la compra de entradas de una película el usuario puede ver un formulario con las fechas y sesiones de esa película.

**Mostrar y seleccionar butacas de una sesión.**

Al seleccionar la sesión el usuario debe poder ver las todas las butacas de la sala donde se proyecta la película y su estado (ocupada, reservada o libre). También podrá seleccionar las butacas que desee siempre que estén libres y sean contiguas en la misma fila.

**Mostrar y seleccionar productos.**

El usuario puede ver todos los productos registrados organizados por categorías. Puede indicar la cantidad que desea de un producto. Igualmente, si desea cancelar la compra de un producto o modificar la cantidad, puede hacerlo desde el carrito de la compra.

**Desbloquear butacas que se han reservado pero no se han comprado.**

Pasado un tiempo, si el usuario no ha realizado la compra de las butacas seleccionadas, estas deben liberarse para que otro usuario pueda comprarlas.

**Mostrar detalles de la compra antes de realizar el pago.**

Antes de realizar el pago el usuario puede ver en detalle todo lo que va a comprar, tanto entradas como productos.

**Crear plantillas de sesiones.**

Con el fin de agilizar el proceso de registro de sesiones, el administrador puede crear plantillas en las que establece unas horas por cada pase y sala. La plantilla puede tener todas las sesiones con una hora establecida o sólo algunas.

En cualquier momento el administrador puede modificar los datos de una plantilla.

**Crear sesiones.**

El administrador puede crear sesiones a partir de plantillas. De esta forma sólo tiene que seleccionar la película que se va a proyectar en esa sesión, las horas están establecidas según la plantilla seleccionada.

Si la hora de una sesión de la plantilla es errónea el administrador puede modificarla en la planificación de sesiones sin que modifique la plantilla.

**Crear una nueva sala.**

El administrador puede crear una nueva sala. En la primera versión de este proyecto sólo puede indicar el número de sala, teniendo en cuenta que no pueden existir dos salas con el mismo número. El número de filas y de butacas por fila está bloqueado. La personalización de estos datos se contempla en una extensión de este proyecto.

Al crear la sala, se generan todas las butacas de esa sala.

PalomitasTime

### **Bloquear y desbloquear butacas.**

El administrador puede bloquear las butacas de una sala. Al bloquear las butacas de una sala se bloquean las butacas de todas las sesiones.

### **Crear y modificar categorías de productos.**

El administrador puede crear y modificar hasta cuatro categorías para los productos que muestra en la página.

### **Crear y modificar productos.**

El administrador puede registrar nuevos productos en la aplicación. Para que estos productos sean visibles para el usuario, se les debe asignar una categoría. Un producto sólo puede pertenecer a una categoría.

### **Crear y modificar menús.**

El administrador puede crear nuevos menús que se mostrarán en la aplicación. Los menús sólo se pueden formar con productos que ya están registrados.

## **4.4.. Requisitos no funcionales**

Son propiedades o cualidades que el sistema debe cumplir.

### **Reducir lo máximo posible el número de pasos necesarios para realizar una compra.**

El usuario puede acceder a la compra de butacas con un solo click. Por defecto se muestra la primera sesión disponible, pero en esa misma ventana puede cambiar el día y la hora sin tener que retroceder ni repetir pasos.

Con otro click más el usuario pasa directamente al pago de las entradas.

### **Diseño *mobile first* para la aplicación destinada al usuario.**

Debido a que cada vez con más frecuencia los usuarios acceden a las aplicaciones desde dispositivos móviles, la aplicación destinada a los usuarios tiene un diseño *mobile first* en el que se prioriza su maquetación adaptada a pantallas pequeñas. Por consiguiente, el diseño es responsivo.

Una vez cargada la página principal, el intercambio de información entre el cliente y el servidor se realiza mediante mensajes en formato JSON. De esta forma se consigue que la respuesta sea más rápida y eficiente incluso si la conexión del cliente no es de buena calidad. También minimiza el consumo de datos para el cliente.

**Diseño atractivo y sencillo del panel de administración.**

Normalmente este tipo de aplicaciones tiene un aspecto sobrio e, incluso, poco intuitivo porque van destinados al personal que ya conoce la aplicación.

El diseño del panel de administración de PalimitasTime tiene un diseño más moderno manteniendo y sencillo manteniendo la profesionalidad necesaria para el entorno en el que se utiliza y facilitando al administrador que entra por primera vez su uso.

En este caso el diseño es responsive y *web first*, ya que el entorno donde se utiliza esta parte de la aplicación es laboral. En este entorno se suele utilizar ordenadores o portátiles, además que se intenta mostrar la mayor cantidad de información posible en una sola ventana y esto es complicado de compatibilizar con el diseño *mobile first*.

**Tiempo real en la selección de butacas.**

Cuando un usuario selecciona una butaca para reservarla, esa butaca queda bloqueada para todos los demás usuarios que estén en la web, de modo que los usuarios no pueden nunca seleccionar la misma butaca de una sesión.

De igual modo para si el usuario desbloquea la butaca.

**Tiempo real en el registro y modificación de reseñas.**

Si un usuario crea una reseña, todos los demás usuarios que estén en la aplicación reciben esa reseña y la ven en tiempo real. Los cambios realizados sobre esa reseña también son recibidos por los demás usuarios.

## PalomitasTime

### 5.. DISEÑO

En esta fase se realiza una aproximación al diseño tecnológico de la solución.

#### 5.1.. ESTRUCTURA DE LA APLICACIÓN

Se trata del desarrollo de una Single Page App (SPA de ahora en adelante). Es un tipo de aplicación web donde todas las pantallas se muestran en la misma página, sin recargar el navegador. Las SPA surgieron de la necesidad de mejorar la experiencia de usuario y el flujo de trabajo. Google, Facebook o Trello utilizan SPA's.

Podemos definir a una SPA como una aplicación web que se ejecuta en una sola página, dando al usuario una sensación de estar usando una aplicación de escritorio. El usuario navega a través de la página no de la forma tradicional de enlaces entre componentes de la página, sino con un sistema de uso del HTML, AJAX o JavaScript e incluso combinando todas ellas haciendo que se actualice únicamente lo que ve el usuario, no la página entera.

Además, una SPA puede funcionar si se pierde la conexión con internet y puede contar con apartados que funcionen en tiempo real.

Un diagrama de flujo de una SPA sería el siguiente:

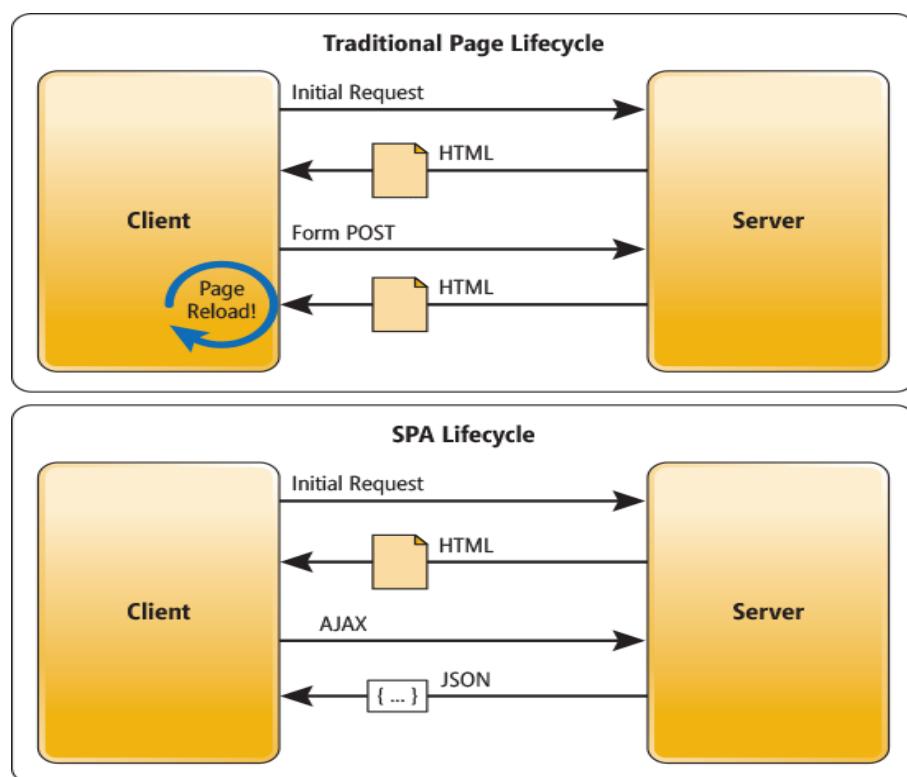


Figura 4: Diagrama de flujo de una SPA.

## PalomitasTime

Ventajas de usar una SPA:

- Reduce las fases de trabajo en la arquitectura al trabajar sobre una misma página, reduciendo con ello la cantidad de código que hay que escribir.
- Simplifica el análisis de la web dado que sólo analizamos una página.
- Facilita el diseño general de la página dado que es único de la página, haciendo que podamos centrarnos en los detalles de la página, o poder probar varios tipos de diseño para la página.

Desventajas de usar una SPA:

- Al tratarse de una única página es fundamental centrarse en su diseño.
- Pierdes oportunidades de mejorar tu tráfico a través del long-text ya que tienes menos información.
- El contenido suele estar supeditado al diseño general de la página lo que puede provocar que no se pueda subir cierto contenido a la página.

Puntualizaciones:

- Generalmente la seguridad de los sitios SPA es menor dado que JavaScript no es un lenguaje compilado, y está abierto a código malicioso proveniente de usuarios malintencionados, aunque existen medios para aumentar la seguridad.
- La optimización SEO puede ser menos efectiva en el contenido de las páginas SPA debido a que los rastreadores SEO dependen directamente del contenido estable en las diferentes páginas, aunque actualmente hay rastreadores de SPA que facilitan el trabajo.

A continuación procedemos a enumerar las diferentes tecnologías que hemos usado, adjuntando una breve descripción de en qué consisten, y las explicaremos en profundidad en el apartado 'Herramientas y tecnologías utilizadas'.

En lo que a la parte front se refiere hemos utilizado:

1. Node.js: Es un entorno de ejecución para JavaScript construido con el motor de JavaScript V8 de Chrome.
2. Yarn: Es un administrador de dependencias relativamente nuevo, creado por Facebook y Google, open source, cuya principal característica es la rapidez. Cualquier paquete que existe en npm es compatible con yarn, al ser este también un package manager.
3. Bulma: Es un framework CSS de código abierto basado en Flexbox y utilizado por más de 100.000 desarrolladores.

## PalomitasTime

4. Flexbox: Fue diseñado como un modelo unidimensional de layout, y como un método que pueda ayudar a distribuir el espacio entre los ítems de una interfaz y mejorar las capacidades de alineación.
5. Vue.js: Es un framework progresivo para construir interfaces de usuario. En nuestro caso lo hemos utilizado por su fácil integración en aplicaciones tipo SPA. Como complemento a Vue hemos usado Vue router, Algolia search y Pusher.
6. Vue router: cliente las rutas son fake cuando se usan desde el navegador, pero cuando tu escribes la dirección en la barra del url, realmente Laravel por debajo hace una petición al servidor para renderizar esa página con Vue.
7. Algolia: Es un proveedor “search-as-a-service” que ayuda a los desarrolladores a implementar una experiencia de búsqueda intuitiva, que permite mostrar los resultados mientras escribes, lo que posibilita la búsqueda de contenido en tiempo real. Laravel Scout es una herramienta que nos ayudará con Algolia.
8. JQuery: Es una librería de JavaScript. Se rigen por el lema ‘Write less, do more’ que lo dice todo. Nos permite agregar funciones JavaScript a nuestra página sin la necesidad de conocer a fondo el lenguaje. Es de código abierto y es muy cómodo de utilizar.

En cuanto a la parte back podemos distinguir:

1. Pusher: Es un servicio online que encapsula la implementación de websockets. Es utilizado generalmente para acciones en tiempo real.
2. Websockets: Es una técnica que establece una conexión de socket TCP largamente mantenida en cliente y servidor.
3. JWT: JSON Web Token es un conjunto de medios de seguridad para peticiones HTTP para transferir ciertos datos entre un cliente y un servidor de forma codificada.
4. TMBD: Es una base de datos de películas y TV construida por la comunidad, que llevan añadiendo datos y elementos desde el año 2008. Nosotros nos centramos en el uso de su API.
5. Laravel Mix: Esta basado en Webpack. Nos provee de una API que compila y minifica el código Cue.js y CSS, además de JavaScript. que hace por debajo es usar Webpack, quien compila y minifica el código Vue y el CSS.

## 5.2.. Componentes del sistema / arquitectura de red

### SERVIDOR WEB

Los servidores web son los que hacen posible el Web hosting, es decir, la posibilidad de alquilar un espacio en un servidor para almacenar los datos de nuestro sitio web. La principal función de un servidor Web es almacenar los archivos de un sitio y emitirlos por Internet para poder ser visitado por los usuarios. Básicamente, un servidor Web es una gran computadora que guarda y transmite datos. Cuando un usuario entra en una página de Internet, su navegador se comunica con el servidor enviando y recibiendo datos que determinan qué es lo que ve en la pantalla. Por eso, decimos que los servidores Web están para almacenar y transmitir datos de un sitio según lo solicita el navegador de un visitante.

En nuestro proyecto hemos optado por DigitalOcean como servidor Web gracias a que da buenas características a un buen precio. Para utilizarlo es necesario crear una cuenta y añadir el dinero que se va a utilizar para alquilar el host. Ese dinero se va descontando como si se tratase de una hucha hasta que llegue a 0, momento en el que el host dejará de dar servicio y debemos o bien incluir más dinero o bien dejarlo estar.

Cuando creamos un droplet, ponen a nuestra disposición una gran variedad de distribuciones GNU/Linux, para poder instalarlas, Debian, Ubuntu, CentOS, cualquier *sabor* que os guste es el que podréis instalar. El droplet se pone en marcha en menos de 1 minuto, una vez funcionando nos enviarán un mail con la IP, el usuario y contraseña para poder acceder por SSH. Igualmente desde el panel web tenemos acceso a una consola en la que podremos gestionar nuestro servidor.

### SISTEMA GESTOR DE BASE DE DATOS

Un sistema gestor de base de datos (a partir de ahora SGDB) es, a groso modo, un sistema que permite la creación, gestión y administración de bases de datos, así como la elección y manejo de las estructuras necesarios para el almacenamiento y búsqueda de la información del modo más eficiente posible.

## **PalomitasTime**

En nuestro caso el SGBD utilizado ha sido MySQL. Como característica principal y que lo hace muy atractivo es que es gratuito. Además es muy sencillo de utilizar, únicamente sabiendo unos pocos comandos podemos utilizarlo (aunque en nuestro caso usaremos las funciones de Laravel para gestionar la BBDD), es rápida, posee buena seguridad y es compatible con varios SO dado que es de código abierto. Todas estas ventajas hacen que MySQL sea uno de los SGBD más usados en el mundo en lo que a bases de datos relacionales se refiere.

## **NAVEGADORES WEB**

Un navegador web es un software, aplicación o programa que permite acceder a la web interpretando el conjunto de archivos proporcionados por un sitio web para que el usuario sea capaz de visualizarlos. La funcionalidad básica de un navegador web es permitir la visualización de documentos de texto, generalmente escritos en lenguaje HTML o JSP. Además, permite visitar páginas web y hacer actividades en ella, es decir, enlazar un sitio con otro, imprimir, enviar y recibir correo, entre otras funcionalidades más. A esto se le llama navegar.

Actualmente existen numerosos navegadores web, aunque las preferencias de los usuarios están muy definidas, pero cabe destacar los siguientes:

- Google Chrome: Es actualmente el navegador más utilizado debido a su interfaz amigable, su rapidez a la hora de cargar archivos del servidor, y sobre todo por su adaptabilidad (es de Google, claro) a todos los servicios Google que prácticamente acaparan hoy en día cualquier servicio online que usemos, ya sea Gmail, YouTube, Drive, etc. Es compatible con muchos dispositivos, incluidos smartphones y tablets y con todos los sistemas operativos. Asimismo es el origen de Node.js y posee herramientas de desarrollo impecables. Además cuenta con innumerables extensiones que permiten personalizar el navegador y hacerlo aún más llamativo desde el punto de vista de la usabilidad. En definitiva, es prácticamente la mejor opción hoy en día. El único pero que puede atribuirsele es la gran cantidad de memoria RAM que consume, lo que puede hacer que en algunos dispositivos, pasado el tiempo, fluya más lento.

## PalomitasTime

- Mozilla Firefox: Cada año mejoran un poco más. Actualmente trabajan en la optimización de su código haciendo que sea cada vez más veloz y fiable. Posee al igual que Chrome un conjunto muy grande de extensiones de todo tipo y es muy compatible con varios dispositivos y sistemas operativos. Posee una interfaz muy personalizable y un gestor independiente de descargas. Usan la filosofía de software libre lo que lo hace muy atractivo para desarrolladores. Los contras que tiene pueden ser su innegable comparación con Chrome (en la que suele salir perdiendo) o su interfaz que en algunos casos se hace demasiado ostentosa. Aun así, sigue siendo uno de los preferidos por los usuarios.
- Microsoft Edge: Es el sucesor del tosco e impopular Internet Explorer. Surge con el nuevo Windows 10, lo que hace que se integre perfectamente con este SO y mucha gente haya optado por seguir sus pasos. Rinde muy bien con JavaScript y es compatible con Chrome a la hora de renderizar las páginas. Algo muy llamativo es que viene preinstalado de serie en los nuevos SO de Windows, por lo que la gente lo utiliza de primeras y en muchos casos acaba convenciendo. Algunos contras que tiene son que al ser utilizado muy intensamente en algunos casos llega a cerrarse puesto que acapara mucha RAM. Es compatible únicamente con SO Windows y Windows Phone y tiene pocas extensiones
- Internet Explorer: Todo navegador de internet conoce a IE. Es posiblemente el navegador más popular (para mal) por las innumerables pegas que recibe de los usuarios, entre las que destacan su lentitud a la hora de cargar páginas, sus excesiva precaución (debido a los numerosos ataques que han recibido a lo largo de los años, pregunta hasta cuándo bostezas), mal entorno de trabajo para desarrolladores. Únicamente es compatible con Windows y tiene bajo o nulo respeto a los estándares.  
Aun así suele ser una opción respetada para gente que tiene baja experiencia y pocos conocimientos en la materia de navegadores. Viene instalado de serie en los SO Windows y tiene soporte oficial.

## PalomitasTime

- Opera: No tan conocido y utilizado como Chrome o Firefox pero siempre presente. Opera cuenta con un modo turbo que ayuda mucho a gente con lenta conexión a internet. Además sus versiones para móviles son geniales. Los contras que tiene son que su código es propietario (algo lógico puesto que la compañía propietaria vive de vender software), no suele ser actualizado lo que hace que se quede atrás de sus competidores (además las actualizaciones para Linux tardan mucho en llegar) y no ofrece nada nuevo que no ofrezcan Chrome o Firefox, lo que hace que los usuarios opten por estas dos opciones antes que por Opera.
- Safari: Es el navegador por excelencia en sistemas Mac OS e iOS puesto que su integración es sublime. Tiene una interfaz limpia, clara y sencilla lo que lo hace muy atractivo. Utiliza de forma muy inteligente el touchpad y para usuarios de Mac OS es, sin duda, el navegador más rápido y eficaz en este SO. Los contras que tiene es que es código propietario (como todo el software Apple), suele estar desactualizado lo que lo hace algo más inseguro, extensiones básicas, pero sobre todo y lo más perjudicial para él es su única compatibilidad con sistemas Apple (y probablemente siempre sea así) lo que hace que sea una opción sólo válida para dispositivos de esta marca. Aun así, ha de ser mencionado.

## 5.3.. Herramientas y tecnologías utilizadas.

NODE.JS - Lo hemos usado principalmente para su ecosistema de paquetes npm, que es el más grande de librerías de código abierto en el mundo.

YARN - Lo hemos utilizado principalmente por estas dos ventajas:

1<sup>a</sup>- Soporte offline: Es una de las más geniales, si ya has instalado anteriormente una dependencia no necesaria conexión a internet para instalarla otra vez.

2<sup>a</sup>- Instalación flexible: Si hay conflicto con una dependencia las demás no se ven afectadas y son auditadas de nuevo.

BULMA – Hemos optado por Bulma en lugar de Bootstrap (que es un equivalente) por estas razones:

## PalomitasTime

1. CSS sin JavaScript: Proporciona un solo archivo CSS sin ningún archivo JS, las interacciones con el código CSS se tienen que hacer manualmente permitiendo así tener todo el control a la hora de crear dichas interacciones y el entorno JS, en nuestro caso el framework Vue.js.
2. Modular: 21kB comprimido, pudiendo ser incluso menos si sólo importamos los módulos que necesitamos.
3. Es 100% Responsivo: Diseñado usando el concepto de 'Mobile First', práctica que hemos respetado al desarrollar nuestro proyecto.
4. Personalización: Es simplemente fantástica. Gracias a que el código está dividido en archivos separados a través de componentes nos ha permitido personalizar casi todas las partes de la aplicación y lograr el resultado que deseamos.
5. Tipografía responsive: Cambiar el tamaño del texto dependiendo del tamaño de la ventana ha tenido mucho peso a la hora de elegir este framework en lugar de Bootstrap que no lo tiene.
6. Moderno: Construido en Flexbox.
7. Compatibilidad: Tiene compatibilidad con múltiples navegadores, entre ellos:

- \* Chrome 45+
- \* Edge 13+
- \* Firefox 43+
- \* Internet Explorer 10+
- \* Safari 9+
- \* Opera 37+

FLEXBOX - En PalomitasTime lo hemos utilizado al crear nuestros componentes propios respetando así también a Bulma.

Los componentes que hemos creado usando Flexbox son:

1. El card de la película que aparece en el home de nuestra web.
2. El hero image con la sinopsis y la cartelera a la izquierda es otro componente propio que ha sido creado de cero en la información de cada película.
3. Las reseñas tanto al crear una como al verlas.

## PalomitasTime

4. Comprar entrada, la gestión de seleccionar el día, hora y butacas están hechas enteramente con Flexbox.
5. Restaurante el card de cada producto.

VUE.JS - A diferencia de otros frameworks monolíticos, Vue está diseñado desde el inicio para ser adoptado incrementalmente. La biblioteca principal se enfoca solo en la capa visual, y es muy simple de utilizar e integrar con otros proyectos o bibliotecas existentes. Además, Vue es capaz de soportar aplicaciones tipo SPA. Una característica de Vue es su diseño progresivo, además se su escalabilidad, lo que la hace muy versátil, es muy optimizada (su núcleo ocupa unos 74KB) y tiene detrás una comunidad cada vez más creciente. Para un funcionamiento mejorado usa plugins como Vue-Router, es posible realizar pruebas unitarias sobre él y da posibilidad de conectarnos a servicios externos.

PUSHER - y la funcionalidad de la aplicación que desarrollemos sin la necesidad de tener que ejecutar un servidor de websockets propio, pudiendo escalar automáticamente según el número de conexiones simultáneas y el número de mensajes enviados.

WEBSOCKETS – En nuestro caso hemos usado websockets para poder hacer la pila de reseñas en tiempo real de las películas. Cuando un usuario comenta una película, y otro usuario está visualizando las reseñas de dicha película, al publicarse la nueva reseña, el usuario lo verá inmediatamente después de haber sido posteada por el primer usuario. Además también fueron utilizados para la reserva de butacas en tiempo real de las sesiones de las películas.

JWT – Para nuestro proyecto JWT juega un papel importante ya que es la herramienta que utilizamos para manejar la autenticación en el caso de los usuarios en la aplicación o de los administradores en el panel de administrador. JWT genera un token que guarda las credenciales necesarias para validar la autenticación del usuario, este token se registra en una cookie que se mantiene durante un tiempo determinado, o bien se borra en el caso de que el usuario haga logout.

## PalomitasTime

JQUERY – JQuery juega un papel fundamental en nuestro proyecto sobre todo en el panel de administración. Toda la interfaz de usuario en el panel admin está basada en JQuery, lo que nos permite darle un toque alegre a la página a través de animaciones por ejemplo al crear o modificar elementos en la base de datos, al utilizar el menú lateral se despliegan sus diferentes funcionalidades, e incluso dentro de las vistas de creación, en algunos casos, hay desplegables que han sido hechos también con JQuery. También nos resultó muy cómoda su versatilidad a la hora de crear tablas o formularios cuyos campos aparecen o desaparecen en función de lo que el usuario introduce o borra. La versión que hemos usado es la 3.3.1.

TMDB – Es una base de datos de películas y series de TV que nos ha sido muy útil. Su API nos ha otorgado la posibilidad de incluir películas que aparezcan en su base de datos directamente a nuestra base de datos, a través de una conexión a la base de datos que nos proporciona una serie de datos como el título, el título original, los actores, un rango de edad, una descripción, una imagen, un vídeo del tráiler, y más datos. Nosotros hemos seleccionado aquellos que nos son necesarios para incluir las películas. Para incluir estos datos, en el panel administrador, optamos por diseñar un “buscador” en el que introduciendo una o varias palabras se realiza una búsqueda en la base de datos de TMDB la cual nos devuelve los títulos de aquellas películas que coincidan con la búsqueda. Y el administrador elige la que desea añadir a la base de datos.

MAILTRAP – También es necesario hablar de mailtrap. Es un servidor SMTP que nos otorga la posibilidad de envíos de correos de prueba cuando estamos desarrollando para no llenar una bandeja de entrada de un correo real de mails de prueba de nuestra aplicación. Mailtrap pone a disposición de los desarrolladores Mailtrap.io para realizar los envíos necesarios. Para poder usarlo es necesario únicamente registrarse en Mailtrap.io. La versión gratuita nos otorga una bandeja de entrada de una capacidad de 50 mensajes, suficiente para realizar aquellas pruebas que el desarrollador considere necesarias.

En nuestro caso el Mailtrap es configurado en el archivo .env de la configuración de Laravel, indicando el host, el puerto, el username que utilizamos y la contraseña. Ha resultado muy útil para probar que el envío de correos después de un registro fuese satisfactorio, o el envío de un correo de confirmación después de realizar la compra de alguna película o producto del restaurante.

## **PalomitasTime**

ELOQUENT – Se trata de un ORM (Object-Relational mapping), que es una forma de mapear los datos que se encuentran en la base de datos almacenados en un lenguaje de script SQL a objetos de PHP y viceversa, esto surge con la idea de tener un código portable con el que no tengamos la necesidad de usar lenguaje SQL dentro de nuestras clases de PHP.

## 6.. IMPLEMENTACIÓN

Partiendo del diseño, en esta fase se construye el sistema.

### 6.1.. Implementación del modelo de datos

Gracias a Eloquent se definen dentro del proyecto las tablas de la base de datos y sus relaciones mediante migraciones. De esta forma podemos recrear la base de datos en cualquier Sistema Gestor de Bases de Datos adaptando sólo el fichero de configuración.

### 6.2.. Carga de datos

Para generar datos en la base de datos se han utilizado los *seeders* de Eloquent. Los *seeders* son ficheros que se ejecutan tras la migración de la base de datos que contienen la información que van a contener las tablas.

Se ha creado un *seeder* por cada objeto de negocio que se considera imprescindible para el buen funcionamiento inicial de la página:

- Administradores.
- Salas.
- Butacas.
- Películulas.
- Sesiones.
- Ingredientes.
- Categorías.
- Menús.
- Productos.

### 6.3.. Configuraciones realizadas en el sistema

Para implementar el proyecto y poder verlo en una maquina local, hace falta seguir esta serie de pasos:

1. Necesitamos tener git instalado, para instalar git iremos a su pagina de descargas y seguiremos los pasos de aquí: <https://git-scm.com/downloads>

## PalomitasTime

2. Cuando la instalación de git termine vamos a instalar node.js desde <https://nodejs.org/es/>, nodejs no viene solo también nos trae su gestor de paquetes npm
3. Como hemos comentado mas arriba, nosotros utilizamos yarn así que vamos a instalarlo desde su pagina oficial <https://yarnpkg.com/lang/en/docs/install/#windows-stable>, al igual que las anteriores instalación solo seguimos los pasos.
4. También necesitamos tener instalado composer para instalar las dependencias de Laravel, lo podemos descargar desde aquí: <https://getcomposer.org/download/>
5. Una vez que tenemos todo instalado, tenemos que colocar el proyecto desde github:
  1. Abrimos la consola de comandos y escribimos:
    1. git clone <https://github.com/Tranity06/Proyecto.git>
  5. Iremos a la raíz de la carpeta del proyecto usando la consola
  6. Escribimos en la consola: composer install y acto seguido se van a instalar todas las dependencias del proyecto.
  7. Cuando termine la instalación, es importante que en la misma consola escribamos "copy .env.example .env", .env es el archivo de configuración del proyecto, dentro podemos configurar la conexión con la base de datos, y los Api keys para Algolia, Pusher, Mailtrap, etc.
  8. Cuando terminemos la configuración del fichero env. En la misma consola ejecutamos "php artisan key:generate", este comando va a generar una clave maestra que va a usar para encriptar las contraseñas en la base de datos.
  9. También ejecutamos "php artisan jwt:secret" parecido al anterior punto pero para generar la clave maestra de JWT.
  10. También ejecutamos "php artisan migrate —seed" para que Laravel automáticamente cree las tablas en la base de datos y con el comando "—seed", insertamos datos en los campos de las tablas que queremos, de esta forma Laravel nos simplifica mucho la vida.
  11. Bien, hemos terminado con la parte back ahora solo nos queda configurar Vue.js para hacerlo tenemos que instalar todas las dependencias que necesita para compilar, usando yarn y en la misma consola ejecutamos "yarn install". Este comando nos va generar una carpeta "node\_modules" en la carpeta raíz del proyecto.

## PalomitasTime

12. Cada vez que modifiquemos algo en un archivo ".vue" es necesario que ejecutemos "yarn run dev", para que Laravel mix compile de nuevo Vue y podamos ver los cambios.

13. Y por ultimo pero no menos importante para desplegar la página en local, ejecutamos "php artisan serve".



```
1 APP_NAME=Laravel
2 APP_ENV=local
3 APP_KEY=base64:es2pexSREW/E0XePxnn7/LfB22oyz4JGZXHzB8d1Ix8=
4 APP_DEBUG=true
5 APP_URL=http://localhost:8000
6
7 LOG_CHANNEL=stack
8
9 DB_CONNECTION=mysql
10 DB_HOST=127.0.0.1
11 DB_PORT=8889
12 DB_DATABASE=cinema
13 DB_USERNAME=root
14 DB_PASSWORD=root
15
16 BROADCAST_DRIVER=pusher
17 CACHE_DRIVER=file
18 SESSION_DRIVER=file
19 SESSION_LIFETIME=120
20 QUEUE_DRIVER=sync
21
22 REDIS_HOST=127.0.0.1
23 REDIS_PASSWORD=null
24 REDIS_PORT=6379
25
26 MAIL_DRIVER=smtp
27 MAIL_HOST=smtp.mailtrap.io
28 MAIL_PORT=2525
29 MAIL_USERNAME=8c60f328aa073f
30 MAIL_PASSWORD=0e4d309d549f31
31 MAIL_ENCRYPTION=tls
32 MAIL_CORREO=hello@example.com
33
34 PUSHER_APP_ID="515825"
35 PUSHER_APP_KEY="718c5f0fe7343b0ea5c4"
36 PUSHER_APP_SECRET="e19fe5cc8b981575fa9b"
37 PUSHER_APP_CLUSTER="eu"
38
39 MIX_PUSHER_APP_KEY="${PUSHER_APP_KEY}"
40 MIX_PUSHER_APP_CLUSTER="${PUSHER_APP_CLUSTER}"
41
42 JWT_SECRET=oZfqmjULLeuZbSbcIwSVa95iSsW0s8M7
43
44 ALGOLIA_APP_ID=Z6VIFIH#1C
45 ALGOLIA_SECRET=48e3d951455bb00f67cbe7a409376887
46 ALGOLIA_SEARCH=7bab21d0c2cdf3c7724d5fc9498e8c0c
```

Figura 5: Fichero .env.

Y esta es la configuración final del fichero ".env", donde podemos ver las claves para conectarnos a las distintas Apis que hemos usado:

## PalomitasTime

También vamos a ver todas las dependencias que hemos usado con Laravel:



```
composer.json
This configuration file contains list of Composer dependencies
1 {
2     "name": "laravel/laravel",
3     "description": "The Laravel Framework.",
4     "keywords": ["framework", "laravel"],
5     "license": "MIT",
6     "type": "project",
7     "require": {
8         "php": ">=7.1.3",
9         "algolia/algoliasearch-client-php": "^1.25",
10        "barryvdh/laravel-cors": "~0.11.0",
11        "fideloper/proxy": "~4.0",
12        "intervention/image": "^2.4",
13        "jeroennoten/laravel-adminlte": "^1.23",
14        "laravel/framework": "5.6.*",
15        "laravel/scout": "~4.0",
16        "laravel/tinker": "~1.0",
17        "milo/vendor-versions": "^1.1",
18        "pusher/pusher-php-server": "~3.0",
19        "spatie/laravel-cookie-consent": "~2.2",
20        "tymon/jwt-auth": "1.0.0-rc.2"
21    },
22    "require-dev": {
23        "filp/whoops": "~2.0",
24        "fzaninotto/faker": "~1.4",
25        "mockery/mockery": "~1.0",
26        "nunomaduro/collision": "~2.0",
27        "phpunit/phpunit": "~7.0",
28        "symfony/thanks": "~1.0",
29        "codacy/coverage": "dev-master"
30    }
},
```

Figura 6: Fichero composer.json.

Vamos a comentar algunas de las dependencias que se pueden ver:

**Algolia:** necesaria para conectar el proyecto con la api a través de las api keys que hemos especificado en el archivo .env

**laravel/scout:** Es la dependencia que se encarga de que Algolia en la nube tenga una copia exactamente igual de la base de datos local.

**laravel/cors:** Debido a que Vue.js se conecta a través de Api endpoints usando Ajax por debajo a dominios diferentes al local(localhost:8000), tuvimos que configurar Cross Origin Resource Sharing para permitir las conexiones entre los dominios a los que nos conectamos, los navegadores por defecto bloquean toda conexión saliente o entrante desde un dominio diferente, gracias a haber configurado el CORS pudimos permitir dichas conexiones.

**laravel-adminlte:** Es la dependencia que se ha utilizado como plantilla para agilizar el desarrollo de la parte de la administración.

**tymon/jwt-auth:** Para que Laravel utilice la autenticación mediante tokens json(JWT), en lugar de su autenticación nativa. Tuvimos que instalar esta dependencia y configurar el tiempo de expiración del token, la forma de generarlo y sus mensajes de error.

## PalomitasTime

Y por último las dependencias que se usan en la parte Front:



```
1  "private": true,
2  "scripts": {
3    "dev": "npm run development",
4    "development": "cross-env NODE_ENV=development node_modules/webpack/bin/webpack.js --progress --hide-modules --config=node_modules/laravel-mix/setup/webpack.config.js",
5    "watch": "cross-env NODE_ENV=development node_modules/webpack/bin/webpack.js --watch --progress --hide-modules --config=node_modules/laravel-mix/setup/webpack.config.js",
6    "hot": "cross-env NODE_ENV=development node_modules/webpack-dev-server/bin/webpack-dev-server.js --inline --hot --config=node_modules/laravel-mix/setup/webpack.config.js",
7    "prod": "npm run production",
8    "production": "cross-env NODE_ENV=production node_modules/webpack/bin/webpack.js --no-progress --hide-modules --config=node_modules/laravel-mix/setup/webpack.config.js"
9  },
10  "devDependencies": {
11    "axios": "^0.18",
12    "bootstrap": "^4.0.0",
13    "cross-env": "5.1",
14    "jquery": "3.2",
15    "laravel-mix": "2.0",
16    "lodash": "4.17.4",
17    "popper.js": "1.12",
18    "vue": "2.5.7"
19  },
20  "dependencies": {
21    "gdmaksimovic/vue-countdown": "1.0.0",
22    "laravel-echo": "1.3.5",
23    "moment": "2.22.1",
24    "nprogress": "0.2.8",
25    "pusher-js": "4.2.2",
26    "vee-validate": "2.0.9",
27    "vue-click-outside": "1.0.7",
28    "vue-draggable": "2.14.0",
29    "vue-google-signin-button": "1.0.2",
30    "vue-instantsearch": "1.0.0",
31    "vue-notification": "1.3.7",
32    "vue-router": "3.0.1",
33    "vue-server-renderer": "2.5.16",
34    "vuex": "3.0.1"
35  }
36}
```

Figura 7: Fichero package.json.

Como en el caso anterior voy a comentar algunas dependencias:

**axios:** Para que las peticiones ajax fuesen más fáciles e intuitivas hemos usado axios, que funciona a base de promesas usando ES6.

**laravel-echo:** Laravel Echo es una librería JavaScript que facilita la suscripción a canales y la escucha de eventos emitidos por Laravel usando Pusher. Funciona mediante el patrón Observer y usa Websockets proporcionando una alternativa más robusta y eficiente para sondear continuamente nuestra aplicación en busca de cambios.

**Vuex:** Es un patrón de gestión de estado + librería para aplicaciones Vue.js. Sirve como almacén centralizado para todos los componentes de una aplicación, con reglas que garantizan que el estado sólo puede mutar de forma predecible. Lo hemos usado en PalomitasTime para almacenar estados de forma global, como por ejemplo si un usuario está logueado o no.

**Vue-server-renderer:** Vue.js es un framework para construir aplicaciones del lado del cliente. Por defecto, los componentes Vue.js producen y manipulan DOM en el navegador. Sin embargo, también es posible renderizar los mismos componentes en cadenas HTML en el servidor, enviarlos directamente al navegador y finalmente "hidratar" el marcado estático en una aplicación completamente interactiva en el cliente. Este renderizado en el servidor por supuesto lleva más tiempo, pero tiene numerosos beneficios como por ejemplo mejor SEO gracias a que la página viene ya renderizada desde el servidor.

## PalomitasTime

**Vue-router:** Nos permite atar o enlazar una o más URLs a uno o más componentes específicos, indicarle a Vue.js dónde queremos mostrar las secciones y por supuesto también nos ayuda a generar los enlaces de las diferentes secciones.

## 6.4. Implementaciones de código realizadas

Como ejemplo de una implementación de código, he elegido la parte de la reseña, concretamente escribir una reseña y publicar una reseña donde participan Vue, Pusher, Axios, Vuex, JWT, Flexbox Y Laravel.

### EscribirResenia.vue

En la primera parte encontramos una sección `<template></template>`, es la sección que contiene la plantilla html, para que el código sea fácil de entender lo voy a comentar.

```
<template>
  <!-- ... -->
  <!-- Este div es el que tiene toda la chicha, podemos ver que para el estilo hemos utilizado las
  clases de Bulma -->
  <div class="media-content">
    <div class="field">
      <p class="control">
        <!-- En Vue.js para activar una clase dinámicamente no existe como en jQuery addClass(), sino
        que se usa la sintaxis v-bind:class o :class para abbreviar, en este caso la clase 'is-danger' solo se
        interpreta si los caracteres sobrepasan el número 140. -->
        <textarea class="textarea" :class="{ 'is-danger': caracteres > 140 }"
          placeholder="Escribe un comentario...">
          <!-- Parecido al anterior comentario existe v-on:click o @click para eventos, aquí se puede ver
          que usando el evento 'keyup' llamamos al método contarCaracteres que estará implementado en la
          segunda parte de este archivo-->
          @keyup="contarCaracteres"
          :value="this.comento"
        <!-- En Vue.js una buena práctica es usar componentes para modularizar nuestro código, y con
        ello siguen algunos patrones de diseño, ya que un componente padre puede tener hijos y estos hermanos
        o más hijos en su interior.
        En este caso "EscribirResena.vue" es hijo de un componente padre y para comunicarse con el padre
        necesita enviarle un evento que llama al método 'update:comento' del padre y le pasa en el segundo
        parámetro el texto del textarea -->
          @input="$emit('update:comento',
            $event.target.value)">
        </textarea>
      </p>
    </div>
  </div>
  <nav class="level">
    <div class="level-left">
      <div class="level-item">
        <!-- Aquí podemos ver otra directiva 'v-if' que se encarga de renderizar o no esa parte html
        basada en una condición, en este caso "ocultarOpciones" tiene que ser false para que aparezca el botón
        "publicar". -->
        <a class="button is-warning" :disabled="caracteres > 140"
          @click="publicarComentario" v-if="!this.ocultarOpciones">Publicar</a>
      </div>
    </div>
  </nav>
```

## PalomitasTime

```
<!-- Y cuando la condición anterior no se cumple se renderiza la etiqueta <p> usando la directiva "v-else" -->
<p class="buttons" v-else>
  <a class="button" :disabled="caracteres > 140" @click="actualizarComentario">
    <span>
      Actualizar
    </span>
  </a>
  <a class="button is-danger" @click="eliminarComentario">
    <span>
      Eliminar
    </span>
  </a>
</p>
</div>
</div>
<div class="level-right">
  <div class="level-item">
    <!-- Para que Vue.js interprete propiedades javascript, se debe usar la sintaxis {{ propiedadJs }}, en este caso cada vez que caracteres cambie el DOM será actualizado con el nuevo valor.-->
    <span>{{ caracteres }}<b>140</b></span>
  </div>
</div>
</nav>
</div>
</article>
<!-- la clase critica utiliza Flexbox para darle estilo, en la siguiente sección se puede ver su código -->
<div v-else class="critica"><span>Identifícate para escribir un comentario.</span></div>
</template>
```

En la segunda parte encontramos la sección `<script></script>` que es la que contiene la lógica del componente, para que el código sea fácil de entender voy a acortarlo y comentarlo.

```
<script>
  /* Primer contacto con ES6, la sentencia import se usa para importar funciones de otro archivo, antes se usaba require. Store es VUEX, dentro almacenó valores de forma global y de fácil acceso entre componentes. */
  import store from '../store';

  export default {
    /**
     * Props define mediante un array las propiedades que este componente interpretará, es muy importante entender esto. Un componente es una etiqueta html igual que la etiqueta <a> que admite propiedades como href="valor", "EscribirResenia" admite las propiedades comento,idResena,etc... que son recibidas desde el componente padre.
     */
    props: ['comento','idResena','ocultarOpciones'],
    /* El nombre de la etiqueta html que vue interpretará al renderizar en el navegador y usará la plantilla de arriba */
    name: "escribir-resenia",
    /* Data es un objeto que contiene las "variables", que hemos usado en la <template> */
    data(){
      return {
        caracteres: 0,
        idPelícula: this.$route.params.id,
```

## PalomitasTime

```
        }
    },
/* computed: Actua como un método pero es muy diferente a este ya que es un método que actúa como si
fuese una variable reactiva, es decir siempre se actualiza su valor cuando la afecta algún cambio.
Tenemos dos computed:
isLogged que nos devuelve si el usuario esta logueado y getAvatar, no olvidemos que store es el archivo
importado al principio y que usa Vuex para las variables y funciones globales entre componentes.*/
computed: {
    isLogged() {
        return store.state.isLoggedIn;
    },
    getAvatar(){
        return store.getters.avatar;
    }
},
/* Y methods qué es donde se declaran las funciones, tenemos “publicarComentario” que usa Axios(Ajax
con vitaminas y promesas) para publicar la reseña al servidor.*/
methods: {
    contarCaracteres(){
        this.caracteres = this.comento.length;
    },
    publicarComentario(){
        /* Destacar el “?token” de la URL, el token JWT se obtiene al hacer login contra el Servidor y
este al comprobar las credenciales devuelve un token valido, este token lo vuelvo a mandar para que JWT
y Laravel puedan extraer el usuario que esta logueado, de esta forma no tengo que pasarle ningún UserId
o parecido al servidor.*/
        axios.post('/api/resena?token=${store.getters.token}', {
            valoracion: 0,
            comentario: this.comento,
            pelicula_id: this.idPelicula
        })
        .then(response => {
            this.resena = response.data;
            this.$emit('publicar', response.data);
            this.$notify({
                group: 'auth',
                type: 'success',
                title: 'Comentario Publicado',
                text: 'Tu comentario se ha publicado',
                duration: 5000,
            });
        })
        .catch(error => {
            console.log(error);
        })
    },
},
```

## PalomitasTime

En la tercera parte encontramos la sección `<style></style>` que es la que contiene el estilo del componente, para que el código sea fácil de entender voy a comentar mi código.

### `<style scoped>`

`/* Se utiliza scoped para que este estilo no sea global y solo pertenezca a este componente, es una buena práctica ya que así se evitan conflictos con otras clases.*/`

```
.critica{  
    /* Al usar flex en la propiedad se entiende que se va a usar flexbox */  
    display: flex;  
    /* Como el display es flex podemos usar align-items para centrar el div verticalmente */  
    align-items: center;  
    /* Y justify-content para centrar el div horizontalmente, así de fácil sin necesidad de usar float */  
    justify-content: center;  
    width: 100%;  
    padding: 12px 20px;  
    background: #fff;  
    border: 1px solid #e3e3e3;  
    border-radius: 5px;  
    box-shadow: 0 2px 8px rgba(0,0,0,0.1);  
    margin-bottom: 1rem;  
}  
  
.image img{  
    border-radius: 50%;  
}  
/>
```

Acabamos de ver una parte de la implementación de “EscribirResenia.vue” para escribir una reseña, veamos ahora la parte back.

Primero tenemos que entender que es un Api-endpoint ya que mediante axios se envía por post al siguiente Api-endpoint: /api/resena?token=valorToken y en el fichero Api.php se puede ver que controlador llama y también el método que ejecuta de ese controlador.

### Api.php:

```
/* ... */  
  
Route::post('/resena', 'ResenaController@crearResenia')-  
>name('resena.crearResenia');  
  
/* ... */
```

Vamos a ver el código del método crearResenia en ResenaController.

### ResenaController.php:

```
/* Otros métodos */  
/**  
 * Si el usuario está logueado registra la reseña.  
 */
```

## PalomitasTime

```
* El usuario no puede escribir dos reseñas sobre la misma pelicula.  
*/  
public function crearResenia(Request $request){  
    //Comprobar autenticación contra JWT  
    try{  
        $user = JWTAuth::toUser(JWTAuth::getToken());  
    } catch (JWTException $e){  
        return response()->json('Debes autenticarte para poder comentar.', 403);  
    }  
  
    //Comprobar si el usuario ya ha comentado la película consultando la Base de datos.  
    $user_resenas = Resena::where([  
        ['user_id', $user->id],  
        ['pelicula_id', $request->pelicula_id]  
    ])->get();  
    if ( sizeof($user_resenas) > 0 ){  
        return response('Ya has comentado sobre esta película! Puedes editar tu reseña desde tu perfil.', 403);  
    }  
  
    //Validar que los datos en el lado Servidor.  
    $credentials = $request->only('valoracion', 'comentario', 'pelicula_id');  
    $rules = [  
        'valoracion' => 'required|min:1',  
        'comentario' => 'required|string|min:1',  
        'pelicula_id' => 'required|min:1'  
    ];  
    $validator = Validator::make($credentials, $rules);  
    if ($validator->fails()) {  
        return response('Debes llenar todos los campos.', 403);  
    }  
  
    //Crear la reseña en la base de datos  
    $res = Resena::create([  
        'valoracion' => $request['valoracion'],  
        'comentario' => $request['comentario'],  
        'user_id' => $user->id,  
        'pelicula_id' => $request['pelicula_id']  
    ]);  
    $resenaJson = [  
        'tipo' => 'write',  
        'id' => $res->id,  
        'tiempo' => $this->time_elapsed_string($res->created_at),  
        'comentario' => $res->comentario,  
        'imagen_usuario' => $user->avatar,  
        'nombre_usuario' => $user->name,  
    ];  
  
    //Y ahora con pusher lanzar un evento a todos los navegadores menos el que esta enviando el evento para que actualizan en tiempo real su lista de reseñas.  
    broadcast(new ResenaEvent($resenaJson))->toOthers();  
  
    //Devolver un json con un código de estado 201 con los datos de $resenaJson si todo ha ido bien.  
    return response()->json($resenaJson, 201);  
}  
  
/* Otros métodos */
```

## 7. PRUEBAS

Son muchas las pruebas que pueden realizarse en un proyecto para eliminar los posibles errores y garantizar su correcto funcionamiento. Los casos de prueba establecen las condiciones/variables que permitirán determinar si los requisitos establecidos se cumplen o no.

A continuación se detallan algunos de los casos de prueba que se ejecutarán para comprobar la correcta construcción de este proyecto.

### 7.1. Casos de pruebas

#### Casos de pruebas de integración sobre el controlador AdministradoresController.

##### Pruebas sobre las peticiones a la ruta '/admin'

Prueba 001:

- Autor: Lorena
- Nombre: ruta\_admin
- Condiciones de ejecución: ninguna.
- Descripción: Comprueba que al entrar al panel de administrador redirecciona a la página de loggin.
- Resultado esperado: status 302, redirección a '/admin/login'.
- Resultado obtenido: status 302, redirección a '/admin/login'.

##### Pruebas sobre las peticiones a la ruta '/admin/login'

Prueba 002:

- Autor: Lorena
- Nombre: ruta\_login\_sin\_sesion
- Condiciones de ejecución: el administrador no está autenticado.
- Descripción: Comprueba que se muestra el formulario de registro si el administrador no está autenticado.
- Resultado esperado: status 200, se muestra el formulario de registro.
- Resultado obtenido: status 200, se muestra el formulario de registro.

## PalomitasTime

### Prueba 003:

- Autor: Lorena
- Nombre: ruta\_login\_con\_sesion
- Condiciones de ejecución: el administrador debe estar autenticado.
- Descripción: Comprueba que se accede al panel de administrador si el administrador ya está autenticado.
- Resultado esperado: status 200, se muestra el panel de administración.
- Resultado obtenido: status 200, se muestra el panel de administración.

### Pruebas sobre las peticiones a la ruta '/admin/settings'

#### Prueba 004:

- Autor: Lorena
- Nombre: ruta\_settings\_sin\_sesion
- Condiciones de ejecución: el administrador no debe estar autenticado.
- Descripción: Comprueba que se redirige al login si el administrador no está autenticado.
- Resultado esperado: status 302, redirección a '/admin'.
- Resultado obtenido: status 302, redirección a '/admin'.

#### Prueba 005:

- Autor: Lorena
- Nombre: ruta\_settings\_con\_sesion
- Condiciones de ejecución: el administrador debe estar autenticado.
- Descripción: Comprueba que se muestran los datos de perfil de administrador logueado.
- Resultado esperado: status 200, se muestra 'Datos de la cuenta'.
- Resultado obtenido: status 200, se muestra 'Datos de la cuenta'.

#### Prueba 006:

- Autor: Lorena
- Nombre: ruta\_settings\_con\_sesion\_user
- Condiciones de ejecución: el usuario está autenticado con una cuenta de usuario.

## PalomitasTime

- Descripción: Comprueba que un usuario no puede acceder al panel de administración.
- Resultado esperado: status 302, redirección a '/admin'.
- Resultado obtenido: status 302, redirección a '/admin'.

## Pruebas sobre las peticiones a la ruta '/admin/comprobar'

### Prueba 007:

- Autor: Lorena
- Nombre: ruta\_comprobar\_get\_sin\_sesion
- Condiciones de ejecución: el administrador no debe estar autenticado.
- Descripción: Comprueba que se redirige al login si el administrador no está autenticado.
- Resultado esperado: status 302, redirección a '/admin'.
- Resultado obtenido: status 302, redirección a '/admin'.

### Prueba 008:

- Autor: Lorena
- Nombre: ruta\_comprobar\_nombre\_post\_sin\_sesion
- Condiciones de ejecución: el administrador no debe estar autenticado.
- Descripción: Comprueba que se redirige al login si el administrador no está autenticado.
- Resultado esperado: status 302, redirección a '/admin'.
- Resultado obtenido: status 302, redirección a '/admin'.

### Prueba 009:

- Autor: Lorena
- Nombre: ruta\_comprobar\_get\_con\_sesion
- Condiciones de ejecución: el administrador debe estar autenticado.
- Descripción: Comprueba que se redirige al '/admin/settings' si el administrador hace una petición GET.
- Resultado esperado: status 302, redirección a '/admin/settings'.
- Resultado obtenido: status 302, redirección a '/admin/settings'.

## PalomitasTime

### Prueba 010:

- Autor: Lorena
- Nombre: ruta\_comprobar\_nombre\_post\_con\_sesion
- Condiciones de ejecución: el administrador debe estar autenticado.
- Descripción: Comprueba que el nombre no está en la base de datos el nombre consultado no está en la base de datos.
- Resultado esperado: status 201.
- Resultado obtenido: status 201.

### Prueba 011:

- Autor: Lorena
- Nombre: ruta\_comprobar\_nombre\_repetido\_post\_con\_sesion
- Condiciones de ejecución: el administrador debe estar autenticado, el nombre introducido existe en la base de datos.
- Descripción: Comprueba que el nombre existe en la base de datos.
- Resultado esperado: status 204.
- Resultado obtenido: status 204.

### Prueba 012:

- Autor: Lorena
- Nombre: ruta\_comprobar\_email\_post\_con\_sesion
- Condiciones de ejecución: el administrador debe estar autenticado, el email introducido no existe en la base de datos.
- Descripción: Comprueba que el email no existe en la base de datos.
- Resultado esperado: status 201.
- Resultado obtenido: status 201.

### Prueba 013:

- Autor: Lorena
- Nombre: ruta\_comprobar\_email\_repetido\_post\_con\_sesion

### **PalomitasTime**

- Condiciones de ejecución: el administrador debe estar autenticado, el nombre introducido existe en la base de datos.
- Descripción: Comprueba que el nombre existe en la base de datos.
- Resultado esperado: status 204.
- Resultado obtenido: status 204.

### Pruebas sobre las peticiones a la ruta '/admin/comprobar'

#### Prueba 014:

- Autor: Lorena
- Nombre: ruta\_modificaradmin\_get\_sin\_sesion
- Condiciones de ejecución: el administrador no debe estar autenticado.
- Descripción: Comprueba que se redirige al login si el administrador no está autenticado.
- Resultado esperado: status 302, redirección a '/admin'.
- Resultado obtenido: status 302, redirección a '/admin'.

#### Prueba 015:

- Autor: Lorena
- Nombre: ruta\_modificaradmin\_post\_sin\_sesion
- Condiciones de ejecución: el administrador no debe estar autenticado.
- Descripción: Comprueba que se redirige al login si el administrador no está autenticado.
- Resultado esperado: status 302, redirección a '/admin'.
- Resultado obtenido: status 302, redirección a '/admin'.

#### Prueba 016:

- Autor: Lorena
- Nombre: ruta\_modificaradmin\_get\_con\_sesion
- Condiciones de ejecución: el administrador debe estar autenticado.
- Descripción: Comprueba que se redirige al '/admin/settings' si el administrador hace una petición GET.

### PalomitasTime

- Resultado esperado: status 302, redirección a '/admin/settings'.
- Resultado obtenido: status 302, redirección a '/admin/settings'.

### Prueba 017:

- Autor: Lorena
- Nombre: ruta\_modificaradmin\_post\_perfil\_con\_sesion
- Condiciones de ejecución: el administrador debe estar autenticado, el nombre introducido no debe coincidir con ninguno registrado.
- Descripción: Comprueba que se guardan los cambios y muestra los nuevos datos de la cuenta.
- Resultado esperado: status 200, se muestra 'Datos de la cuenta'.
- Resultado obtenido: status 200, se muestra 'Datos de la cuenta'.

### Prueba 018:

- Autor: Lorena
- Nombre: ruta\_modificarperfil\_post\_con\_sesion\_clave\_correcto
- Condiciones de ejecución: el administrador debe estar autenticado, la clave introducida debe tener un formato correcto.
- Descripción: Comprueba que se guardan los cambios y muestra los nuevos datos de la cuenta.
- Resultado esperado: status 200, se muestra 'Datos de la cuenta'.
- Resultado obtenido: status 200, se muestra 'Datos de la cuenta'.

### Prueba 019:

- Autor: Lorena
- Nombre: ruta\_modificarperfil\_post\_con\_sesion\_clave\_nocorrecto
- Condiciones de ejecución: el administrador debe estar autenticado, la clave introducida no debe tener un formato correcto.
- Descripción: No realiza ningún cambio redirige a '/admin/settings'.
- Resultado esperado: status 302, regirige a '/admin/settings'.
- Resultado obtenido: status 302, regirige a '/admin/settings'.

## PalomitasTime

### Prueba 020:

- Autor: Lorena
- Nombre: ruta\_modificarperfil\_post\_con\_sesion\_nombre\_repe
- Condiciones de ejecución: el administrador debe estar autenticado, el nombre introducido debe existir en la base de datos.
- Descripción: No realiza ningún cambio redirige a '/admin/settings'.
- Resultado esperado: status 302, regirige a '/admin/settings'.
- Resultado obtenido: status 302, regirige a '/admin/settings'.

### Prueba 021:

- Autor: Lorena
- Nombre: ruta\_modificaradmin\_post\_con\_sesion\_no\_superuser\_otroadmin
- Condiciones de ejecución: el administrador debe estar autenticado con una cuenta de administrador normal.
- Descripción: No realiza ningún cambio Muestra mensaje de autorización denegada.
- Resultado esperado: status 200, muestra los mensajes 'Permiso denegado.' Y 'No tienes permisos para realizar esta acción.'
- Resultado esperado: status 200, muestra los mensajes 'Permiso denegado.' Y 'No tienes permisos para realizar esta acción.'

### Prueba 022:

- Autor: Lorena
- Nombre:  
 ruta\_modificaradmin\_post\_con\_sesion\_superuser\_otroadmin\_nombre\_correcto
- Condiciones de ejecución: el administrador debe estar autenticado, el nombre introducido no existe en la base de datos.
- Descripción: Registra los cambios en la base de datos y muestra los administradores registrados con la información actualizada.
- Resultado esperado: status 200, muestra el mensaje 'Administradores registrados'.
- Resultado esperado: status 200, muestra el mensaje 'Administradores registrados'.

## PalomitasTime

### Prueba 023:

- Autor: Lorena
- Nombre:  
 ruta\_modificaradmin\_post\_con\_sesion\_superuser\_otroadmin\_nombre\_incorrecto
- Condiciones de ejecución: el administrador debe estar autenticado, el nombre introducido existe en la base de datos.
- Descripción: No registra ningún cambio y redirige a la lista de administradores.
- Resultado esperado: status 302.
- Resultado esperado: status 302.

### Prueba 024:

- Autor: Lorena
- Nombre:  
 ruta\_modificaradmin\_post\_con\_sesion\_superuser\_otroadmin\_email\_correcto
- Condiciones de ejecución: el administrador debe estar autenticado, el email introducido no existe en la base de datos.
- Descripción: Registra los cambios en la base de datos y muestra los administradores registrados con la información actualizada.
- Resultado esperado: status 200, muestra el mensaje 'Administradores registrados'.
- Resultado esperado: status 200, muestra el mensaje 'Administradores registrados'.

### Prueba 025:

- Autor: Lorena
- Nombre:  
 ruta\_modificaradmin\_post\_con\_sesion\_superuser\_otroadmin\_email\_incorrecto
- Condiciones de ejecución: el administrador debe estar autenticado, el email introducido existe en la base de datos.
- Descripción: No registra ningún cambio y redirige a la lista de administradores.
- Resultado esperado: status 302.
- Resultado esperado: status 302.

### Pruebas sobre las peticiones a la ruta '/admin/crear'

## PalomitasTime

### Prueba 026:

- Autor: Lorena
- Nombre: ruta\_crear\_get\_sin\_sesion
- Condiciones de ejecución: el administrador no debe estar autenticado.
- Descripción: Comprueba que se redirige al login si el administrador no está autenticado.
- Resultado esperado: status 302, redirección a '/admin'.
- Resultado obtenido: status 302, redirección a '/admin'.

### Prueba 027:

- Autor: Lorena
- Nombre: ruta\_crear\_get\_con\_sesion\_no\_superadm
- Condiciones de ejecución: el administrador debe estar autenticado con una cuenta de administrador normal.
- Descripción: No realiza ningún cambio Muestra mensaje de autorización denegada.
- Resultado esperado: status 200, muestra los mensajes 'Permiso denegado.'
- Resultado esperado: status 200, muestra los mensajes 'Permiso denegado.'

### Prueba 028:

- Autor: Lorena
- Nombre: ruta\_crear\_get\_con\_sesion\_superadm
- Condiciones de ejecución: el administrador debe estar autenticado.
- Descripción: Muestra el formulario de creación de nuevos administradores.
- Resultado esperado: status 200, muestra el mensaje 'Crear nuevo administrador'.
- Resultado esperado: status 200, muestra el mensaje 'Crear nuevo administrador'.

### Prueba 029:

- Autor: Lorena
- Nombre: ruta\_crear\_post\_con\_sesion\_superadm
- Condiciones de ejecución: el administrador debe estar autenticado, todos los datos introducidos son correctos.

### **PalomitasTime**

- Descripción: Registra los datos del nuevo administrador.
- Resultado esperado: status 200, muestra el mensaje 'Usuario creado'.
- Resultado esperado: status 200, muestra el mensaje 'Usuario creado'.

### **Prueba 030:**

- Autor: Lorena
- Nombre: ruta\_crear\_post\_con\_sesion\_superadm\_nombre\_repetido
- Condiciones de ejecución: el administrador debe estar autenticado, el nombre introducido existe en la base de datos.
- Descripción: Redirige a un mensaje de error.
- Resultado esperado: status 302.
- Resultado esperado: status 302.

### **Prueba 031:**

- Autor: Lorena
- Nombre: ruta\_crear\_post\_con\_sesion\_superadm\_email\_incorrecto
- Condiciones de ejecución: el administrador debe estar autenticado, el email introducido tiene un formato incorrecto.
- Descripción: Redirige a un mensaje de error.
- Resultado esperado: status 302.
- Resultado esperado: status 302.

### **Prueba 032:**

- Autor: Lorena
- Nombre: ruta\_crear\_post\_con\_sesion\_no\_superadm
- Condiciones de ejecución: el administrador debe estar autenticado con una cuenta de administrador normal.
- Descripción: No realiza ningún cambio Muestra mensaje de autorización denegada.
- Resultado esperado: status 200, muestra los mensajes 'Permiso denegado.'
- Resultado esperado: status 200, muestra los mensajes 'Permiso denegado.'

## PalomitasTime

### Pruebas sobre las peticiones a la ruta '/admin/administradores'

Prueba 033:

- Autor: Lorena
- Nombre: ruta\_administradores\_get\_sin\_sesion
- Condiciones de ejecución: el administrador no debe estar autenticado.
- Descripción: Comprueba que se redirige al login si el administrador no está autenticado.
- Resultado esperado: status 302, redirección a '/admin'.
- Resultado obtenido: status 302, redirección a '/admin'.

Prueba 034:

- Autor: Lorena
- Nombre: ruta\_administradores\_get\_con\_sesion\_superuser
- Condiciones de ejecución: el administrador debe estar autenticado con una cuenta de administrador normal.
- Descripción: Muestra la lista de administradores registrados y permite modificarlos.
- Resultado esperado: status 200, muestra los mensajes 'Administradores registrados' y 'Modificar'.
- Resultado esperado: status 200, muestra los mensajes 'Administradores registrados' y 'Modificar'.

Prueba 035:

- Autor: Lorena
- Nombre: ruta\_administradores\_get\_con\_sesion\_no\_superuser
- Condiciones de ejecución: el administrador debe estar autenticado con una cuenta de administrador normal.
- Descripción: Muestra la lista de administradores registrados y permite modificarlos.
- Resultado esperado: status 200, muestra los mensajes 'Administradores registrados' y no muestra 'Modificar'.
- Resultado esperado: status 200, muestra los mensajes 'Administradores registrados' y no muestra 'Modificar'.

## PalomitasTime

### Pruebas sobre las peticiones a la ruta '/admin/borrar'

#### Prueba 036:

- Autor: Lorena
- Nombre: ruta\_borrar\_get\_sin\_sesion
- Condiciones de ejecución: el administrador no debe estar autenticado.
- Descripción: Comprueba que se redirige al login si el administrador no está autenticado.
- Resultado esperado: status 302, redirección a '/admin'.
- Resultado obtenido: status 302, redirección a '/admin'.

#### Prueba 037:

- Autor: Lorena
- Nombre: ruta\_borrar\_get\_con\_sesion
- Condiciones de ejecución: el administrador debe estar autenticado.
- Descripción: Comprueba que se redirige al '/admin' si el administrador hace una petición GET.
- Resultado esperado: status 302, redirección a '/admin'.
- Resultado obtenido: status 302, redirección a '/admin'.

#### Prueba 038:

- Autor: Lorena
- Nombre: ruta\_borrar\_post\_con\_sesion\_superadm
- Condiciones de ejecución: el administrador debe estar autenticado, el administrador que se quiere borrar debe existir en la base de datos.
- Descripción: Comprueba que elimina el administrador indicado.
- Resultado esperado: status 204.
- Resultado obtenido: status 204.

#### Prueba 039:

- Autor: Lorena
- Nombre: ruta\_borrar\_post\_con\_sesion\_no\_superadm

### **PalomitasTime**

- Condiciones de ejecución: el administrador debe estar autenticado con una cuenta de administrador normal.
- Descripción: Prohibe la acción al administrador.
- Resultado esperado: status 403.
- Resultado obtenido: status 403.

### **Casos de pruebas de integración sobre el controlador UserController.**

#### **Prueba 040:**

- Autor: Andrei
- Nombre: it\_can\_register
- Condiciones de ejecución: el usuario ha llenado el formulario de registro con datos correctos.
- Descripción: Comprueba que se registran los datos, muestra mensaje de confirmación y envía un email al usuario para la validación.
- Resultado esperado: status 201, envía JSON con los datos 'true' y 'thanks for singing up!', envía un email a la dirección de correo del usuario.
- Resultado obtenido: status 201, envía JSON con los datos 'true' y 'thanks for singing up!', envía un email a la dirección de correo del usuario.

#### **Prueba 041:**

- Autor: Andrei
- Nombre: it\_sends\_wrong\_data\_to\_register
- Condiciones de ejecución: el usuario ha llenado el formulario de registro con datos incorrectos.
- Descripción: No registra los datos y envía un mensaje de error.
- Resultado esperado: status 400, envía JSON con los datos 'false' y 'Los datos introducidos no son correctos'.
- Resultado obtenido: status 400, envía JSON con los datos 'false' y 'Los datos introducidos no son correctos'.

#### **Prueba 042:**

### PalomitasTime

- Autor: Andrei
- Nombre: it\_can\_verify
- Condiciones de ejecución: el usuario se ha registrado, ha recibido el email de verificación y lo ha confirmado.
- Descripción: Cambia el estado del usuario a validado y muestra un mensaje de confirmación.
- Resultado esperado: status 200, envía JSON con los datos 'true' y 'Has verificado correctamente tu cuenta'.
- Resultado obtenido: status 200, envía JSON con los datos 'true' y 'Has verificado correctamente tu cuenta'.

### Prueba 043:

- Autor: Andrei
- Nombre: it\_is\_already\_verified
- Condiciones de ejecución: el usuario se ha registrado, ha recibido el email de verificación y ya había confirmado antes su validación.
- Descripción: Muestra un mensaje informando de que su cuenta ya está verificada.
- Resultado esperado: status 200, envía JSON con los datos 'true' y 'Cuenta ya verificada...'.
- Resultado obtenido: status 200, envía JSON con los datos 'true' y 'Cuenta ya verificada...'.

### Prueba 044:

- Autor: Andrei
- Nombre: verification\_code\_is\_invalid
- Condiciones de ejecución: el código de verificación es incorrecto.
- Descripción: Muestra un mensaje de que el código no es correcto.
- Resultado esperado: status 400, envía JSON con los datos 'false' y 'El código de verificación no es válido'.
- Resultado obtenido: status 400, envía JSON con los datos 'false' y 'El código de verificación no es válido'.

## PalomitasTime

### Prueba 045:

- Autor: Andrei
- Nombre: it\_can\_login
- Condiciones de ejecución: el usuario ha activado su cuenta y no está autenticado.
- Descripción: Crea los datos de sesión al autenticarse.
- Resultado esperado: crea datos de sesión.
- Resultado obtenido: crea datos de sesión.

### Prueba 046:

- Autor: Andrei
- Nombre: it\_sends\_wrong\_data\_to\_login
- Condiciones de ejecución: los datos de autenticación no son correctos.
- Descripción: Muestra un mensaje de error.
- Resultado esperado: status 400.
- Resultado obtenido: status 400.

### Prueba 047:

- Autor: Andrei
- Nombre: it\_is\_unauthorized
- Condiciones de ejecución: los datos no coinciden con los registrados.
- Descripción: Muestra un mensaje de error.
- Resultado esperado: status 401, envía JSON con los datos 'false' y 'We cant find an account with this credentials.'
- Resultado obtenido: status 400, envía JSON con los datos 'false' y 'We cant find an account with this credentials.'

### Prueba 048:

- Autor: Andrei
- Nombre: it\_can\_logout
- Condiciones de ejecución: el usuario está autenticado.

### PalomitasTime

- Descripción: Cuando el usuario desloguea se borran los datos de sesión y muestra un mensaje.
- Resultado esperado: envía JSON con los datos 'true' y 'You have successfully logged out'.
- Resultado obtenido: envía JSON con los datos 'true' y 'You have successfully logged out'.

### Prueba 049:

- Autor: Andrei
- Nombre: it\_can\_recovery\_password
- Condiciones de ejecución: el usuario no está autenticado.
- Descripción: El usuario pide una recuperación de contraseña de una cuenta registrada.
- Resultado esperado: status 200, envía JSON con los datos 'true' y 'A reset email has been sent!'.
- Resultado obtenido: status 200, envía JSON con los datos 'true' y 'A reset email has been sent!'.

### Prueba 050:

- Autor: Andrei
- Nombre: it\_cant\_find\_email
- Condiciones de ejecución: el usuario no está autenticado.
- Descripción: El usuario intenta autenticarse con un email no registrado.
- Resultado esperado: status 401, envía JSON con los datos 'false' y 'Your email address was not found'.
- Resultado obtenido: status 401, envía JSON con los datos 'false' y 'Your email address was not found'.

Pruebas sobre las peticiones POST a la ruta '/resena'

Prueba 051:

- Autor: Lorena
- Nombre: crear\_resena\_con\_usuario\_logueado
- Condiciones de ejecución: el usuario está autenticado.
- Descripción: El usuario autenticado crear una reseña.
- Resultado esperado: status 201.
- Resultado obtenido: status 201.

Prueba 052:

- Autor: Lorena
- Nombre: crear\_resena\_sin\_usuario\_logueado
- Condiciones de ejecución: el usuario no está autenticado.
- Descripción: El usuario no autenticado crear una reseña.
- Resultado esperado: status 403, envía el mensaje 'Debes autenticarte para poder comentar.'.
- Resultado obtenido: status 403, envía el mensaje 'Debes autenticarte para poder comentar.'.

Prueba 053:

- Autor: Lorena
- Nombre: crear\_resena\_con\_usuario\_logueado\_pelicula\_repetida
- Condiciones de ejecución: el usuario está autenticado.
- Descripción: El usuario autenticado crear una reseña sobre una película repetida.
- Resultado esperado: status 403, envía el mensaje '¡Ya has comentado sobre esta película!'.
- Resultado obtenido: status 403, envía el mensaje '¡Ya has comentado sobre esta película!'.

## PalomitasTime

### Prueba 054:

- Autor: Lorena
- Nombre: crear\_resena\_con\_usuario\_logueado\_sin\_valoración
- Condiciones de ejecución: el usuario está autenticado.
- Descripción: El usuario autenticado crea una reseña sin valorar la película.
- Resultado esperado: status 403, envía el mensaje 'Debes llenar todos los campos'.
- Resultado obtenido: status 403, envía el mensaje 'Debes llenar todos los campos'.

### Prueba 055:

- Autor: Lorena
- Nombre: crear\_resena\_con\_usuario\_logueado\_sin\_comentario
- Condiciones de ejecución: el usuario está autenticado.
- Descripción: El usuario autenticado crea una reseña sin comentar la película.
- Resultado esperado: status 403, envía el mensaje 'Debes llenar todos los campos'.
- Resultado obtenido: status 403, envía el mensaje 'Debes llenar todos los campos'.

### Prueba 056:

- Autor: Lorena
- Nombre: crear\_resena\_con\_usuario\_logueado\_sin\_pelicula
- Condiciones de ejecución: el usuario está autenticado.
- Descripción: El usuario autenticado crea una reseña sin indicar la película.
- Resultado esperado: status 403, envía el mensaje 'Debes llenar todos los campos'.
- Resultado obtenido: status 403, envía el mensaje 'Debes llenar todos los campos'.

## Pruebas sobre las peticiones PUT a la ruta '/resena'

### Prueba 057:

- Autor: Lorena
- Nombre: update\_resena\_con\_usuario\_logueado
- Condiciones de ejecución: el usuario está autenticado.
- Descripción: El usuario autenticado modifica una reseña propia.

### PalomitasTime

- Resultado esperado: status 201, envía un mensaje con el texto introducido.
- Resultado obtenido: status 201, envía un mensaje con el texto introducido.

### Prueba 058:

- Autor: Lorena
- Nombre: update\_resena\_sin\_usuario\_logueado
- Condiciones de ejecución: el usuario no está autenticado.
- Descripción: El usuario no autenticado modifica una reseña.
- Resultado esperado: status 403, envía un mensaje con el 'Debes autenticarte para poder comentar.'
- Resultado obtenido: status 403, envía un mensaje con el 'Debes autenticarte para poder comentar.'

### Prueba 059:

- Autor: Lorena
- Nombre: update\_resena\_con\_usuario\_logueado\_resena\_no\_existe
- Condiciones de ejecución: el usuario está autenticado.
- Descripción: El usuario autenticado modifica una reseña que no está registrada.
- Resultado esperado: status 400, envía un mensaje con el 'no existe.'
- Resultado obtenido: status 400, envía un mensaje con el 'no existe.'

### Prueba 060:

- Autor: Lorena
- Nombre: update\_resena\_con\_usuario\_logueado\_sin\_valoración
- Condiciones de ejecución: el usuario está autenticado.
- Descripción: El usuario autenticado modifica una reseña sin introducir la valoración.
- Resultado esperado: status 403, envía un mensaje con el 'Debes llenar todos los campos.'
- Resultado obtenido: status 403, envía un mensaje con el 'Debes llenar todos los campos.'

## PalomitasTime

### Prueba 061:

- Autor: Lorena
- Nombre: update\_resena\_con\_usuario\_logueado\_sin\_comentario
- Condiciones de ejecución: el usuario está autenticado.
- Descripción: El usuario autenticado modifica una reseña sin introducir el comentario.
- Resultado esperado: status 403, envía un mensaje con el 'Debes llenar todos los campos.'
- Resultado obtenido: status 403, envía un mensaje con el 'Debes llenar todos los campos.'

### Prueba 062:

- Autor: Lorena
- Nombre: update\_resena\_otro\_usuario
- Condiciones de ejecución: el usuario está autenticado.
- Descripción: El usuario autenticado modifica una reseña de otro usuario.
- Resultado esperado: status 403.
- Resultado obtenido: status 403.

### Pruebas sobre las peticiones DELETE a la ruta '/resena'

### Prueba 063:

- Autor: Lorena
- Nombre: delete\_resena\_con\_usuario\_logueado
- Condiciones de ejecución: el usuario está autenticado.
- Descripción: El usuario autenticado bora una reseña propia.
- Resultado esperado: status 204.
- Resultado obtenido: status 204.

### Prueba 064:

- Autor: Lorena
- Nombre: delete\_resena\_sin\_usuario\_logueado

### PalomitasTime

- Condiciones de ejecución: el usuario no está autenticado.
- Descripción: El usuario no autenticado borra una reseña propia.
- Resultado esperado: status 403, envía un mensaje con el 'Debes autenticarte para poder comentar.'
- Resultado obtenido: status 403, envía un mensaje con el 'Debes autenticarte para poder comentar.'

### Prueba 065:

- Autor: Lorena
- Nombre: delete\_resena\_con\_usuario\_logueado\_resena\_no\_existe
- Condiciones de ejecución: el usuario está autenticado.
- Descripción: El usuario autenticado borra una reseña que no está registrada.
- Resultado esperado: status 400, envía un mensaje con el 'no existe.'
- Resultado obtenido: status 400, envía un mensaje con el 'no existe.'

### Prueba 066:

- Autor: Lorena
- Nombre: delete\_resena\_otro\_usuario
- Condiciones de ejecución: el usuario está autenticado.
- Descripción: El usuario autenticado borra una reseña que de otro usuario.
- Resultado esperado: status 403.
- Resultado obtenido: status 403.

### Pruebas sobre las peticiones GET a la ruta '/resena'

### Prueba 067:

- Autor: Lorena
- Nombre: ver\_resenas\_usuario\_logueado
- Condiciones de ejecución: el usuario está autenticado.
- Descripción: El usuario autenticado quiere ver todas sus reseñas.
- Resultado esperado: status 201, envía en formato JSON todas las reseñas del usuario.

## PalomitasTime

- Resultado obtenido: status 201, envía en formato JSON todas las reseñas del usuario.

## Prueba 068:

- Autor: Lorena
- Nombre: ver\_resenas\_usuario\_no\_logueado
- Condiciones de ejecución: el usuario no está autenticado.
- Descripción: El usuario no autenticado quiere ver todas sus reseñas.
- Resultado esperado: status 403.
- Resultado obtenido: status 403.

## Casos de pruebas de integración sobre las peticiones a PeliculaController.

### Pruebas sobre las peticiones GET a la ruta '/película/crear'

#### Prueba 069:

- Autor: Lorena
- Nombre: crear\_pelicula\_sin\_admin
- Condiciones de ejecución: el administrador no debe estar autenticado.
- Descripción: Se realiza una petición get para crear una película sin un administrador autenticado.
- Resultado esperado: status 302, redirección a '/admin'.
- Resultado obtenido: status 302, redirección a '/admin'.

#### Prueba 070:

- Autor: Lorena
- Nombre: crear\_pelicula\_con\_admin
- Condiciones de ejecución: el administrador debe estar autenticado.
- Descripción: el administrador pide crear una película.
- Resultado esperado: status 200, Se muestran el mensaje 'Registrar nueva película' y el nombre del administrador autenticado.

## PalomitasTime

- Resultado obtenido: status 200, Se muestran el mensaje 'Registrar nueva película' y el nombre del administrador autenticado.

### Pruebas sobre las peticiones POST a la ruta '/película/crear'

#### Prueba 071:

- Autor: Lorena
- Nombre: post\_crear\_pelicula\_sin\_admin
- Condiciones de ejecución: el administrador no debe estar autenticado.
- Descripción: Se realiza una petición post para crear una película sin un administrador autenticado.
- Resultado esperado: status 302, redirección a '/admin'.
- Resultado obtenido: status 302, redirección a '/admin'.

#### Prueba 072:

- Autor: Lorena
- Nombre: post\_crear\_pelicula\_con\_admin
- Condiciones de ejecución: el administrador debe estar autenticado.
- Descripción: el administrador registra los datos de una nueva película.
- Resultado esperado: status 200, Se muestran los mensajes 'Película registrada' y 'Registrar nueva película' y el nombre del administrador autenticado.
- Resultado obtenido: status 200, Se muestran los mensajes 'Película registrada' y 'Registrar nueva película' y el nombre del administrador autenticado.

#### Prueba 073:

- Autor: Lorena
- Nombre: crear\_pelicula\_repetida
- Condiciones de ejecución: el administrador debe estar autenticado.
- Descripción: el administrador registra los datos de una nueva película que ya está registrada.

### PalomitasTime

- Resultado esperado: status 200, Se muestran los mensajes 'La película ya estaba registrada en la base de datos' y 'Registrar nueva película' y el nombre del administrador autenticado.
- Resultado obtenido: status 200, Se muestran los mensajes 'La película ya estaba registrada en la base de datos' y 'Registrar nueva película' y el nombre del administrador autenticado.

### Prueba 074:

- Autor: Lorena
- Nombre: crear\_pelicula\_campo\_inválido
- Condiciones de ejecución: el administrador debe estar autenticado.
- Descripción: el administrador registra los datos de una nueva película con algún campo con formato incorrecto.
- Resultado esperado: status 403, Se muestran el mensaje 'Debes llenar todos los campos'.
- Resultado obtenido: status 403, Se muestran el mensaje 'Debes llenar todos los campos'.

### Pruebas sobre las peticiones GET a la ruta '/películas/mostrar'

#### Prueba 075:

- Autor: Lorena
- Nombre: mostrar\_peliculas\_sin\_admin
- Condiciones de ejecución: el administrador no debe estar autenticado.
- Descripción: Se realiza una petición para mostrar todas las películas registradas.
- Resultado esperado: status 302, redirección a '/admin'.
- Resultado obtenido: status 302, redirección a '/admin'.

#### Prueba 076:

- Autor: Lorena
- Nombre: mostrar\_peliculas\_con\_admin
- Condiciones de ejecución: el administrador debe estar autenticado.

## PalomitasTime

- Descripción: el administrador quiere ver todas las películas registradas.
- Resultado esperado: status 200, Se muestran el nombre del administrador autenticado y todas las películas.
- Resultado obtenido: status 200, Se muestran el nombre del administrador autenticado y todas las películas.

## Pruebas sobre las peticiones POST a la ruta '/películas/borrar'

### Prueba 077:

- Autor: Lorena
- Nombre: borrar\_pelicula\_sin\_admin
- Condiciones de ejecución: el administrador no debe estar autenticado.
- Descripción: Se realiza una petición para borrar una película sin un administrador autenticado.
- Resultado esperado: status 302, redirección a '/admin'.
- Resultado obtenido: status 302, redirección a '/admin'.

### Prueba 078:

- Autor: Lorena
- Nombre: borrar\_pelicula\_con\_admin
- Condiciones de ejecución: el administrador debe estar autenticado.
- Descripción: el administrador quiere borrar una película registrada.
- Resultado esperado: status 204.
- Resultado obtenido: status 204.

### Prueba 079:

- Autor: Lorena
- Nombre: borrar\_pelicula\_no\_registrada
- Condiciones de ejecución: el administrador debe estar autenticado.
- Descripción: el administrador quiere borrar una película no registrada.
- Resultado esperado: status 400.
- Resultado obtenido: status 400.

Prueba 080:

- Autor: Lorena
- Nombre: ver\_peliculas\_y\_sesiones\_programadas
- Condiciones de ejecución: ninguna.
- Descripción: el usuario quiere ver todas las películas y sus secciones activas.
- Resultado esperado: status 200, el atributo 'estado' de las películas tiene valor 0.
- Resultado obtenido: status 200, el atributo 'estado' de las películas tiene valor 0.

Prueba 081:

- Autor: Lorena
- Nombre: ver\_info\_y\_resenas\_pelicula
- Condiciones de ejecución: ninguna.
- Descripción: el usuario quiere ver la información y reseñas de una película.
- Resultado esperado: status 200, se muestra toda la información de la película indicada y las reseñas pertenecen a la película indicada.
- Resultado obtenido: status 200, se muestra toda la información de la película indicada y las reseñas pertenecen a la película indicada.

Prueba 082:

- Autor: Lorena
- Nombre: ver\_resenas\_pelicula
- Condiciones de ejecución: ninguna.
- Descripción: el usuario quiere ver todas reseñas de una película.
- Resultado esperado: status 200, las reseñas pertenecen a la película indicada.
- Resultado obtenido: status 200, las reseñas pertenecen a la película indicada.

Pruebas sobre las peticiones GET a la ruta '/salas'

Prueba 083:

- Autor: Lorena
- Nombre: mostrar\_salas\_no\_admin
- Condiciones de ejecución: el administrador no debe estar autenticado.
- Descripción: Se realiza una petición para mostrar todas las salas registradas.
- Resultado esperado: status 302, redirección a '/admin'.
- Resultado obtenido: status 302, redirección a '/admin'.

Prueba 084:

- Autor: Lorena
- Nombre: mostrar\_salas\_admin
- Condiciones de ejecución: el administrador debe estar autenticado.
- Descripción: el administrador quiere ver todas las salas registradas.
- Resultado esperado: status 200, Se muestran el nombre del administrador autenticado y todas las salas.
- Resultado obtenido: status 200, Se muestran el nombre del administrador autenticado y todas las salas.

Pruebas sobre las peticiones GET a la ruta '/salas/{idSala}'

Prueba 085:

- Autor: Lorena
- Nombre: mostrar\_info\_sala\_no\_admin
- Condiciones de ejecución: el administrador no debe estar autenticado.
- Descripción: Se realiza una petición para mostrar los detalles de la sala indicada sin un administrador autenticado.
- Resultado esperado: status 302, redirección a '/admin'.
- Resultado obtenido: status 302, redirección a '/admin'.

## PalomitasTime

### Prueba 086:

- Autor: Lorena
- Nombre: mostrar\_info\_sala\_admin
- Condiciones de ejecución: el administrador debe estar autenticado.
- Descripción: el administrador quiere los detalles de la sala indicada.
- Resultado esperado: status 200, Se muestran el nombre del administrador autenticado y los detalles de la sala indicada.
- Resultado obtenido: status 200, Se muestran el nombre del administrador autenticado y los detalles de la sala indicada.

### Pruebas sobre las peticiones GET a la ruta '/sala'

### Prueba 087:

- Autor: Lorena
- Nombre: ver\_formulario\_crear\_no\_admin
- Condiciones de ejecución: el administrador no debe estar autenticado.
- Descripción: Se realiza una petición para recibir el formulario de creación de nueva sala sin un administrador autenticado.
- Resultado esperado: status 302, redirección a '/admin'.
- Resultado obtenido: status 302, redirección a '/admin'.

### Prueba 088:

- Autor: Lorena
- Nombre: ver\_formulario\_crear\_sala\_admin
- Condiciones de ejecución: el administrador debe estar autenticado.
- Descripción: el administrador quiere el formulario para registrar una nueva sala.
- Resultado esperado: status 200, Se muestran el nombre del administrador autenticado y el mensaje 'Crear nueva sala.'
- Resultado obtenido: status 200, Se muestran el nombre del administrador autenticado y el mensaje 'Crear nueva sala.'

Prueba 089:

- Autor: Lorena
- Nombre: crear\_sala\_no\_admin
- Condiciones de ejecución: el administrador no debe estar autenticado.
- Descripción: Se crea una sala sin un administrador autenticado.
- Resultado esperado: status 302, redirección a '/admin'.
- Resultado obtenido: status 302, redirección a '/admin'.

Prueba 090:

- Autor: Lorena
- Nombre: crear\_sala\_admin
- Condiciones de ejecución: el administrador debe estar autenticado.
- Descripción: el administrador quiere registrar una nueva sala.
- Resultado esperado: status 200, Se muestran el nombre del administrador autenticado y el mensaje 'Sala creada.'
- Resultado obtenido: status 200, Se muestran el nombre del administrador autenticado y el mensaje 'Sala creada.'

Prueba 091:

- Autor: Lorena
- Nombre: crear\_sala\_repetida
- Condiciones de ejecución: el administrador debe estar autenticado.
- Descripción: el administrador quiere registrar una sala ya registrada.
- Resultado esperado: status 403, Se muestran el mensaje 'La sala ya existe.'
- Resultado obtenido: status 403, Se muestran el mensaje 'La sala ya existe.'

Prueba 092:

- Autor: Lorena
- Nombre: crear\_sala\_datos\_incorrectos
- Condiciones de ejecución: el administrador debe estar autenticado.

## PalomitasTime

- Descripción: el administrador quiere registrar una sala con datos incorrectos.
- Resultado esperado: status 403, Se muestran el mensaje 'Los datos introducidos no son correctos.'
- Resultado obtenido: status 403, Se muestran el mensaje 'Los datos introducidos no son correctos.'

## Pruebas sobre las peticiones POST a la ruta '/sala/borrar'

Prueba 093:

- Autor: Lorena
- Nombre: borrar\_sala\_no\_admin
- Condiciones de ejecución: el administrador no debe estar autenticado.
- Descripción: Se borra una sala sin un administrador autenticado.
- Resultado esperado: status 302, redirección a '/admin'.
- Resultado obtenido: status 302, redirección a '/admin'.

Prueba 094:

- Autor: Lorena
- Nombre: borrar\_sala\_admin
- Condiciones de ejecución: el administrador debe estar autenticado.
- Descripción: el administrador quiere borrar una sala.
- Resultado esperado: status 204.
- Resultado obtenido: status 204.

Prueba 095:

- Autor: Lorena
- Nombre: borrar\_sala\_no\_registrada
- Condiciones de ejecución: el administrador debe estar autenticado.
- Descripción: el administrador quiere borrar una sala que no está registrada.
- Resultado esperado: status 403, se muestra el mensaje 'La sala indicada no existe'.
- Resultado obtenido: status 403, se muestra el mensaje 'La sala indicada no existe'.

## PalomitasTime

### Prueba 096:

- Autor: Lorena
- Nombre: borrar\_sala\_con\_sesiones
- Condiciones de ejecución: el administrador debe estar autenticado.
- Descripción: el administrador quiere borrar una sala que tiene sesiones activas registradas.
- Resultado esperado: status 403, se muestra el mensaje 'La sala tiene sesiones programadas'.
- Resultado obtenido: status 403, se muestra el mensaje 'La sala tiene sesiones programadas'.

## Casos de pruebas de integración sobre las peticiones a SalaController

### Pruebas sobre las peticiones POST a la ruta '/butaca/bloquear'

#### Prueba 097:

- Autor: Lorena
- Nombre: bloquear\_butaca\_no\_admin
- Condiciones de ejecución: el administrador no debe estar autenticado.
- Descripción: Se bloquea una butaca sin un administrador autenticado.
- Resultado esperado: status 302, redirección a '/admin'.
- Resultado obtenido: status 302, redirección a '/admin'.

#### Prueba 098:

- Autor: Lorena
- Nombre: bloquear\_butacas
- Condiciones de ejecución: el administrador debe estar autenticado.
- Descripción: el administrador quiere bloquear varias butacas de una fila.
- Resultado esperado: status 201, en atributo estado de las butacas enviadas en el JSON es 1 y nunca es 0.
- Resultado obtenido: status 201, en atributo estado de las butacas enviadas en el JSON es 1 y nunca es 0.

## PalomitasTime

### Prueba 099:

- Autor: Lorena
- Nombre: bloquear\_fila
- Condiciones de ejecución: el administrador debe estar autenticado.
- Descripción: el administrador quiere bloquear todas las butacas de una fila.
- Resultado esperado: status 201, en atributo estado de las butacas enviadas en el JSON es 1 y nunca es 0.
- Resultado obtenido: status 201, en atributo estado de las butacas enviadas en el JSON es 1 y nunca es 0.

### Prueba 100:

- Autor: Lorena
- Nombre: bloquear\_butaca\_fila\_no\_registrada
- Condiciones de ejecución: el administrador debe estar autenticado.
- Descripción: el administrador quiere bloquear todas las butacas de una fila que no está registrada.
- Resultado esperado: status 403, se muestra el mensaje 'La fila o butaca no existe'.
- Resultado obtenido: status 403, se muestra el mensaje 'La fila o butaca no existe'.

### Prueba 101:

- Autor: Lorena
- Nombre: bloquear\_butaca\_no\_registrada
- Condiciones de ejecución: el administrador debe estar autenticado.
- Descripción: el administrador quiere bloquear una butaca que no existe.
- Resultado esperado: status 403, se muestra el mensaje 'La fila o butaca no existe'.
- Resultado obtenido: status 403, se muestra el mensaje 'La fila o butaca no existe'.

### Pruebas sobre las peticiones POST a la ruta '/butaca/desbloquear'

### Prueba 102:

- Autor: Lorena
- Nombre: desbloquear\_butaca\_no\_admin

### **PalomitasTime**

- Condiciones de ejecución: el administrador no debe estar autenticado.
- Descripción: Se desbloquea una butaca sin un administrador autenticado.
- Resultado esperado: status 302, redirección a '/admin'.
- Resultado obtenido: status 302, redirección a '/admin'.

### Prueba 103:

- Autor: Lorena
- Nombre: desbloquear\_butaca
- Condiciones de ejecución: el administrador debe estar autenticado.
- Descripción: el administrador quiere desbloquear varias butacas de una fila.
- Resultado esperado: status 201, se muestra el mensaje 'Butaca actualizada'.
- Resultado obtenido: status 201, se muestra el mensaje 'Butaca actualizada'.

### Prueba 104:

- Autor: Lorena
- Nombre: desbloquear\_butaca\_no\_registrada
- Condiciones de ejecución: el administrador debe estar autenticado.
- Descripción: el administrador quiere desbloquear una butaca no registrada.
- Resultado esperado: status 403, se muestra el mensaje 'La fila o butaca no existe'.
- Resultado obtenido: status 403, se muestra el mensaje 'La fila o butaca no existe'.

## **Casos de pruebas de integración sobre las peticiones a PlantillaSesionController**

### Pruebas sobre las peticiones GET a la ruta '/plantillas'

#### Prueba 105:

- Autor: Lorena
- Nombre: mostrar\_plantillas\_no\_admin
- Condiciones de ejecución: el administrador no debe estar autenticado.
- Descripción: Se solicita la lista de plantillas sin un administrador autenticado.
- Resultado esperado: status 302, redirección a '/admin'.
- Resultado obtenido: status 302, redirección a '/admin'.

Prueba 106:

- Autor: Lorena
- Nombre: mostrar\_plantillas\_admin
- Condiciones de ejecución: el administrador debe estar autenticado.
- Descripción: el administrador quiere ver las plantillas registradas.
- Resultado esperado: status 200, se muestra el nombre del administrador y el mensaje 'plantillas de sesiones'.
- Resultado obtenido: status 200, se muestra el nombre del administrador y el mensaje 'plantillas de sesiones'.

Pruebas sobre las peticiones GET a la ruta '/plantilla'

Prueba 107:

- Autor: Lorena
- Nombre: ver\_formulario\_crear\_plantilla\_no\_admin
- Condiciones de ejecución: el administrador no debe estar autenticado.
- Descripción: Se solicita el formulario de creación de plantillas sin un administrador autenticado.
- Resultado esperado: status 302, redirección a '/admin'.
- Resultado obtenido: status 302, redirección a '/admin'.

Prueba 108:

- Autor: Lorena
- Nombre: ver\_formulario\_crear\_plantilla\_admin
- Condiciones de ejecución: el administrador debe estar autenticado.
- Descripción: el administrador crear una nueva plantilla.
- Resultado esperado: status 200, se muestra el nombre del administrador y el mensaje 'Crear nueva plantilla'.
- Resultado obtenido: status 200, se muestra el nombre del administrador y el mensaje 'Crear nueva plantilla'.

Prueba 109:

- Autor: Lorena
- Nombre: crear\_plantilla\_no\_admin
- Condiciones de ejecución: el administrador no debe estar autenticado.
- Descripción: Se crea una plantilla sin un administrador autenticado.
- Resultado esperado: status 302, redirección a '/admin'.
- Resultado obtenido: status 302, redirección a '/admin'.

Prueba 110:

- Autor: Lorena
- Nombre: crear\_plantilla\_admin
- Condiciones de ejecución: el administrador debe estar autenticado.
- Descripción: el administrador crea una nueva plantilla.
- Resultado esperado: status 200, se muestra el nombre del administrador y el mensaje 'Plantilla nueva'.
- Resultado obtenido: status 200, se muestra el nombre del administrador y el mensaje 'Plantilla nueva'.

Pruebas sobre las peticiones POST a la ruta '/plantilla/modificar'

Prueba 111:

- Autor: Lorena
- Nombre: modificar\_plantilla\_no\_admin
- Condiciones de ejecución: el administrador no debe estar autenticado.
- Descripción: Se modifica una plantilla sin un administrador autenticado.
- Resultado esperado: status 302, redirección a '/admin'.
- Resultado obtenido: status 302, redirección a '/admin'.

Prueba 112:

- Autor: Lorena
- Nombre: modificar\_nombre\_plantilla
- Condiciones de ejecución: el administrador debe estar autenticado.

### **PalomitasTime**

- Descripción: el administrador modifica el nombre de una plantilla registrada.
- Resultado esperado: status 204.
- Resultado obtenido: status 204.

### **Prueba 113:**

- Autor: Lorena
- Nombre: modificar\_descripcion\_plantilla
- Condiciones de ejecución: el administrador debe estar autenticado.
- Descripción: el administrador modifica la descripción de una plantilla registrada.
- Resultado esperado: status 204.
- Resultado obtenido: status 204.

### **Prueba 114:**

- Autor: Lorena
- Nombre: modificar\_plantilla\_no\_registrada
- Condiciones de ejecución: el administrador debe estar autenticado.
- Descripción: el administrador modifica una plantilla no registrada.
- Resultado esperado: status 403, se muestra el mensaje 'La plantilla no existe'.
- Resultado obtenido: Resultado esperado: status 403, se muestra el mensaje 'La plantilla no existe'.

### **Pruebas sobre las peticiones POST a la ruta '/plantilla/borrar'**

### **Prueba 115:**

- Autor: Lorena
- Nombre: borrar\_plantilla\_no\_admin
- Condiciones de ejecución: el administrador no debe estar autenticado.
- Descripción: Se borra una plantilla sin un administrador autenticado.
- Resultado esperado: status 302, redirección a '/admin'.
- Resultado obtenido: status 302, redirección a '/admin'.

### **Prueba 116:**

### PalomitasTime

- Autor: Lorena
- Nombre: borrar\_plantilla\_admin
- Condiciones de ejecución: el administrador debe estar autenticado.
- Descripción: el administrador borra una plantilla registrada.
- Resultado esperado: status 204.
- Resultado obtenido: status 204.

### Prueba 117:

- Autor: Lorena
- Nombre: borrar\_plantilla\_no\_registrada
- Condiciones de ejecución: el administrador debe estar autenticado.
- Descripción: el administrador borra una plantilla no registrada.
- Resultado esperado: status 403, se muestra el mensaje 'La plantilla no existe'.
- Resultado obtenido: Resultado esperado: status 403, se muestra el mensaje 'La plantilla no existe'.

## Casos de pruebas de integración sobre las peticiones a SesionVaciaController

### Pruebas sobre las peticiones POST a la ruta '/sesionvacia/borrar'

#### Prueba 118:

- Autor: Lorena
- Nombre: borrar\_sesionvacia\_no\_admin
- Condiciones de ejecución: el administrador no debe estar autenticado.
- Descripción: Se solicita la lista de plantillas sin un administrador autenticado.
- Resultado esperado: status 302, redirección a '/admin'.
- Resultado obtenido: status 302, redirección a '/admin'.

#### Prueba 119:

- Autor: Lorena
- Nombre: borrar\_sesionvacia\_admin
- Condiciones de ejecución: el administrador debe estar autenticado.

### **PalomitasTime**

- Descripción: el administrador quiere borrar la sesión registrada de una plantilla.
- Resultado esperado: status 204.
- Resultado obtenido: status 204.

### **Prueba 120:**

- Autor: Lorena
- Nombre: borrar\_sesionvacia\_no\_registrada
- Condiciones de ejecución: el administrador debe estar autenticado.
- Descripción: el administrador quiere borrar la sesión no registrada de una plantilla.
- Resultado esperado: status 403, se muestra el mensaje 'no existe.'
- Resultado obtenido: status 403, se muestra el mensaje 'no existe.'

### Pruebas sobre las peticiones POST a la ruta '/sesionvacia/crear'

### **Prueba 121:**

- Autor: Lorena
- Nombre: crear\_sesionvacia\_no\_admin
- Condiciones de ejecución: el administrador no debe estar autenticado.
- Descripción: Se sesiones para una plantilla sin un administrador autenticado.
- Resultado esperado: status 302, redirección a '/admin'.
- Resultado obtenido: status 302, redirección a '/admin'.

### **Casos de pruebas de integración sobre las peticiones a CategoriaController**

### **Prueba 122:**

- Autor: Jorge
- Nombre: crear\_categoria\_sin\_admin
- Condiciones de ejecución: el administrador no está autenticado.
- Descripción: no se podrá acceder a crear una nueva categoría.
- Resultado esperado: status 302.
- Resultado obtenido: status 302.

## PalomitasTime

### Prueba 123:

- Autor: Jorge
- Nombre: crear\_categoria\_con\_admin
- Condiciones de ejecución: el administrador está autenticado.
- Descripción: el administrador crea una nueva categoría.
- Resultado esperado: status 201.
- Resultado obtenido: status 201.

### Prueba 124:

- Autor: Jorge
- Nombre: mostrar\_categorias\_sin\_admin
- Condiciones de ejecución: el administrador no está autenticado.
- Descripción: no se podrá acceder a mostrar las categorías.
- Resultado esperado: status 302.
- Resultado obtenido: status 302.

### Prueba 125:

- Autor: Jorge
- Nombre: mostrar\_categorias\_con\_admin
- Condiciones de ejecución: el administrador está autenticado.
- Descripción: el administrador visualiza la lista de categorías.
- Resultado esperado: status 200.
- Resultado obtenido: status 200.

### Prueba 126:

- Autor: Jorge
- Nombre: crear\_categoria\_usuario\_logueado
- Condiciones de ejecución: el administrador está autenticado.

### **PalomitasTime**

- Descripción: el administrador crea una nueva categoría.
- Resultado esperado: status 201.
- Resultado obtenido: status 201.

### Prueba 127:

- Autor: Jorge
- Nombre: modificar\_categoria\_sin\_admin
- Condiciones de ejecución: el administrador no está autenticado.
- Descripción: no se puede acceder a modificar categoría.
- Resultado esperado: status 302.
- Resultado obtenido: status 302.

### Prueba 128:

- Autor: Jorge
- Nombre: modificar\_categoria\_con\_admin
- Condiciones de ejecución: el administrador está autenticado.
- Descripción: el administrador modifica una categoría.
- Resultado esperado: status 200.
- Resultado obtenido: status 200.

### Prueba 129:

- Autor: Jorge
- Nombre: modificar\_categoria\_no\_registrada
- Condiciones de ejecución: el administrador está autenticado.
- Descripción: no se puede modificar una categoría no registrada.
- Resultado esperado: status 400.
- Resultado obtenido: status 400.

### Prueba 130:

### PalomitasTime

- Autor: Jorge
- Nombre: borrar\_categoria\_sin\_admin
- Condiciones de ejecución: el administrador no está autenticado.
- Descripción: no se puede acceder a borrar categoría.
- Resultado esperado: status 302.
- Resultado obtenido: status 302.

### Prueba 131:

- Autor: Jorge
- Nombre: borrar\_categoria\_con\_admin
- Condiciones de ejecución: el administrador está autenticado.
- Descripción: el administrador borra una categoría.
- Resultado esperado: status 204.
- Resultado obtenido: status 204.

### Prueba 132:

- Autor: Jorge
- Nombre: borrar\_categoria\_no\_registrada
- Condiciones de ejecución: el administrador está autenticado.
- Descripción: no se puede borrar una categoría no registrada.
- Resultado esperado: status 400.
- Resultado obtenido: status 400.

## Casos de pruebas de integración sobre las peticiones a MenuController.

### Prueba 133:

- Autor: Jorge
- Nombre: crear\_menu\_sin\_admin
- Condiciones de ejecución: el administrador no está autenticado.
- Descripción: no se podrá acceder a crear un nuevo menú.

### **PalomitasTime**

- Resultado esperado: status 302.
- Resultado obtenido: status 302.

### Prueba 134:

- Autor: Jorge
- Nombre: crear\_menu\_con\_admin
- Condiciones de ejecución: el administrador está autenticado.
- Descripción: el administrador crea un nuevo menú.
- Resultado esperado: status 201.
- Resultado obtenido: status 201.

### Prueba 135:

- Autor: Jorge
- Nombre: mostrar\_menus\_sin\_admin
- Condiciones de ejecución: el administrador no está autenticado.
- Descripción: no se podrá acceder a mostrar los menús.
- Resultado esperado: status 302.
- Resultado obtenido: status 302.

### Prueba 136:

- Autor: Jorge
- Nombre: mostrar\_menus\_con\_admin
- Condiciones de ejecución: el administrador está autenticado.
- Descripción: el administrador visualiza la lista de menús.
- Resultado esperado: status 200.
- Resultado obtenido: status 200.

### Prueba 137:

### PalomitasTime

- Autor: Jorge
- Nombre: crear\_menu\_usuario\_logueado
- Condiciones de ejecución: el administrador está autenticado.
- Descripción: el administrador crea un nuevo menú.
- Resultado esperado: status 201.
- Resultado obtenido: status 201.

### Prueba 138:

- Autor: Jorge
- Nombre: modificar\_menu\_sin\_admin
- Condiciones de ejecución: el administrador no está autenticado.
- Descripción: no se puede acceder a modificar menú.
- Resultado esperado: status 302.
- Resultado obtenido: status 302.

### Prueba 139:

- Autor: Jorge
- Nombre: modificar\_menu\_con\_admin
- Condiciones de ejecución: el administrador está autenticado.
- Descripción: el administrador modifica un menú.
- Resultado esperado: status 200.
- Resultado obtenido: status 200.

### Prueba 140:

- Autor: Jorge
- Nombre: modificar\_menu\_no\_registrada
- Condiciones de ejecución: el administrador está autenticado.

### **PalomitasTime**

- Descripción: no se puede modificar un menú no registrado.
- Resultado esperado: status 400.
- Resultado obtenido: status 400.

### **Prueba 141:**

- Autor: Jorge
- Nombre: borrar\_menu\_sin\_admin
- Condiciones de ejecución: el administrador no está autenticado.
- Descripción: no se puede acceder a borrar menú.
- Resultado esperado: status 302.
- Resultado obtenido: status 302.

### **Prueba 142:**

- Autor: Jorge
- Nombre: borrar\_menu\_con\_admin
- Condiciones de ejecución: el administrador está autenticado.
- Descripción: el administrador borra un menú.
- Resultado esperado: status 204.
- Resultado obtenido: status 204.

### **Prueba 143:**

- Autor: Jorge
- Nombre: borrar\_menu\_no\_registrada
- Condiciones de ejecución: el administrador está autenticado.
- Descripción: no se puede borrar un menú no registrado.
- Resultado esperado: status 400.
- Resultado obtenido: status 400.

## PalomitasTime

Para que los **tests de integración y unitarios se ejecuten más rápido** en lugar de usar Mysql como se hace en producción, **se ha decidido utilizar una base de datos en memoria**, en concreto **sql lite**. Es importante destacar que después de ejecutar cada método se hace un rollback de los cambios en la base de datos para que no persistan y no altere el resto del métodos.

Un breve resumen más visual con la cobertura de cada clase que hemos creado, dicha cobertura se puede consultar en codacy:

Clase	Cobertura
SalaController	100 %
SpaController	100 %
ButacaController	100 %
ResenaEvent	100 %
ResenaController	99 %
AdministradoresController	94 %
APIAuthController	91 %
VerificarEmail	80 %
PeliculaController	74 %
PlantillaSesionController	71 %
CategoríaController	80 %
MenuController	80 %
PerfilController	70 %

Tabla 26: Cobertura de pruebas.

## 8.. EXPLOTACIÓN

La implantación es la fase más crítica del proyecto ya que el sistema entra en producción, es decir opera en un entorno real, con usuarios reales.

### 8.1.. Planificación

Una vez alcanzados unos objetivos mínimos en cuanto a funcionalidad de la aplicación, esta se subirá al servidor elegido.

Este servidor se elegirá en función del tiempo disponible y compatibilidad con nuestro sistema.

Los objetivos mínimos son los siguientes:

1. Debe realizar de forma automática la migración de la base de datos.
2. Debe generar de forma automática una cuenta de super administrador, las salas y butacas de cada sala, un número concreto de películas, sesiones lógicas de las películas en cada sala, las butacas asociadas a cada sala, una lista de productos y menús.
3. Debe ser posible acceder al panel de administrador y realizar algunas tareas.
4. Debe ser posible poder moverse por el restaurante y el cine.

### 8.2.. Preparación para el cambio

Se configurará el repositorio con las siguientes ramas:

- Rama máster: esta es la rama que contiene la versión que se sube al servidor.
- Rama desarrollo: esta rama contiene la última versión del proyecto, cuando se compruebe que todo funcione correctamente en esta rama se actualizará la rama máster.
- Ramas por desarrollador o tarea: en estas ramas se actualizan pequeñas partes del proyecto. Cuando su funcionamiento sea correcto se realizará un merge con la rama de desarrollo y se tratarán los posibles conflictos.

De esta forma el despliegue del proyecto y su desarrollo se pueden llevar de formas separadas e independiente, permitiendo a uno de los desarrolladores encargarse de esta parte mientras los demás continúan actualizando la aplicación.

### 8.3.. Manual de usuario

Se ha desarrollado una maqueta de un manual para los usuarios que entran por primera vez en la aplicación explicando los puntos más importantes de esta.

#### Página de inicio

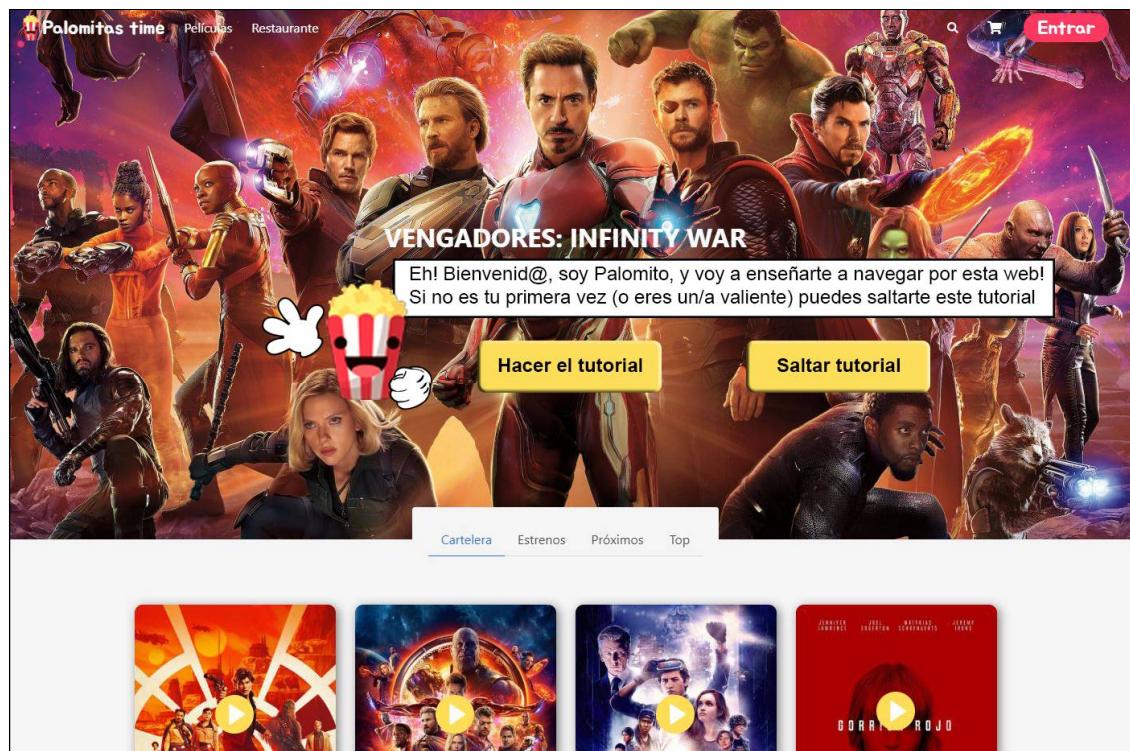


Figura 8: Manual de usuario (1/12).

## PalomitasTime

### Página 1

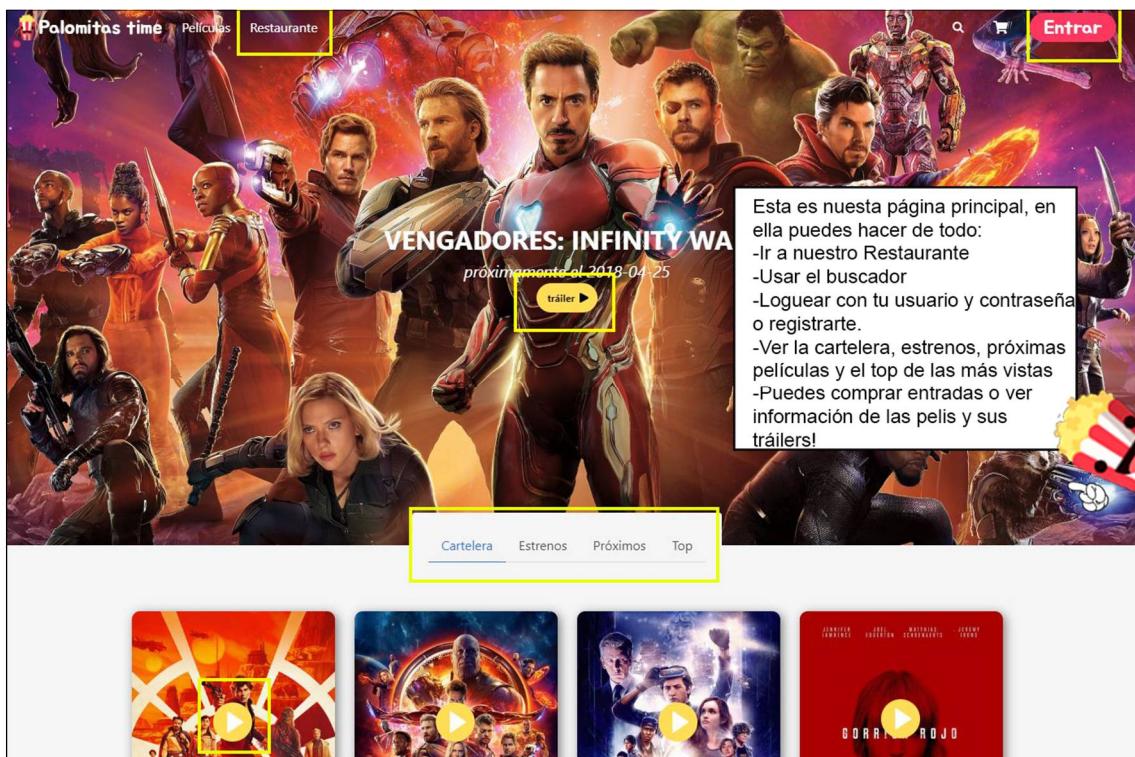


Figura 9: Manual de usuario (2/12).

### Página 2

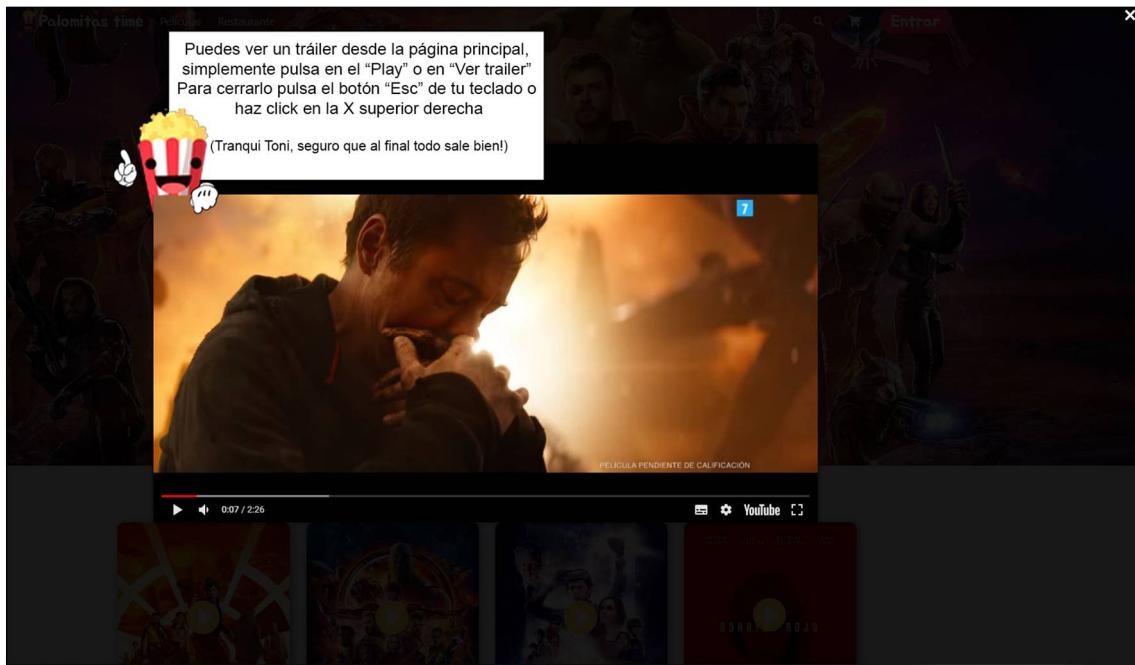


Figura 10: Manual de usuario (3/12).

### Página 3

## PalomitasTime

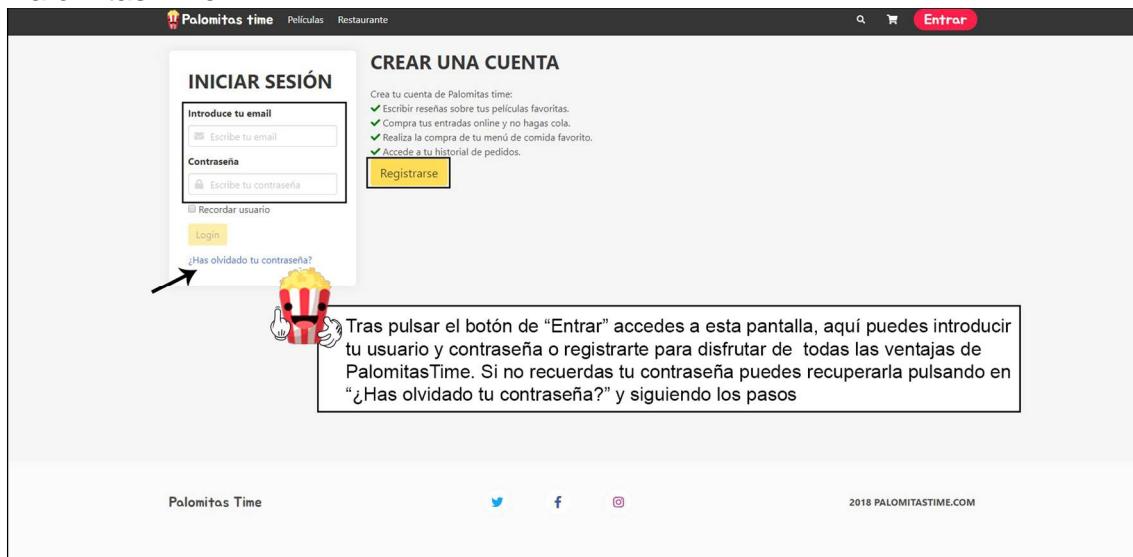


Figura 11: Manual de usuario (4/12).

## Página 4

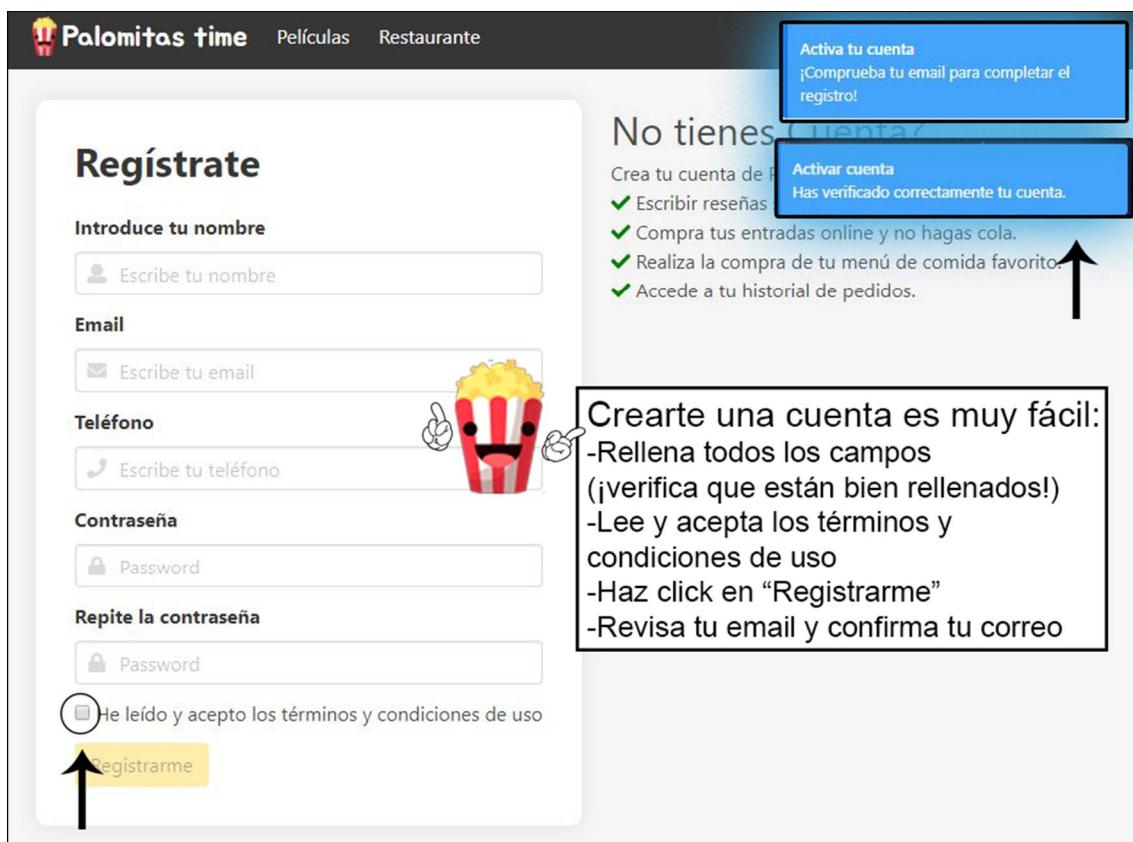


Figura 12: Manual de usuario (5/12).

## PalomitasTime

### Página 5

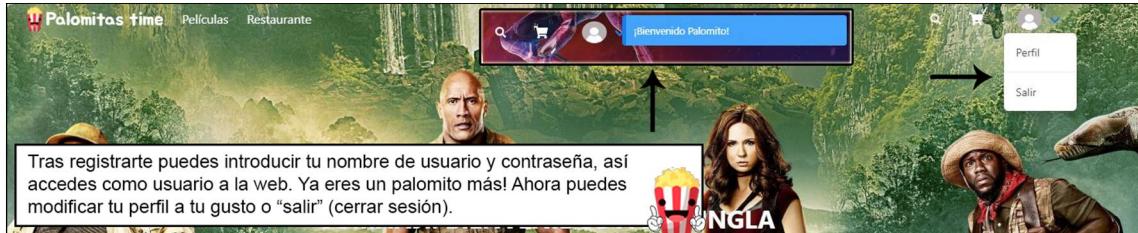


Figura 13: Manual de usuario (6/12).

### Página 6

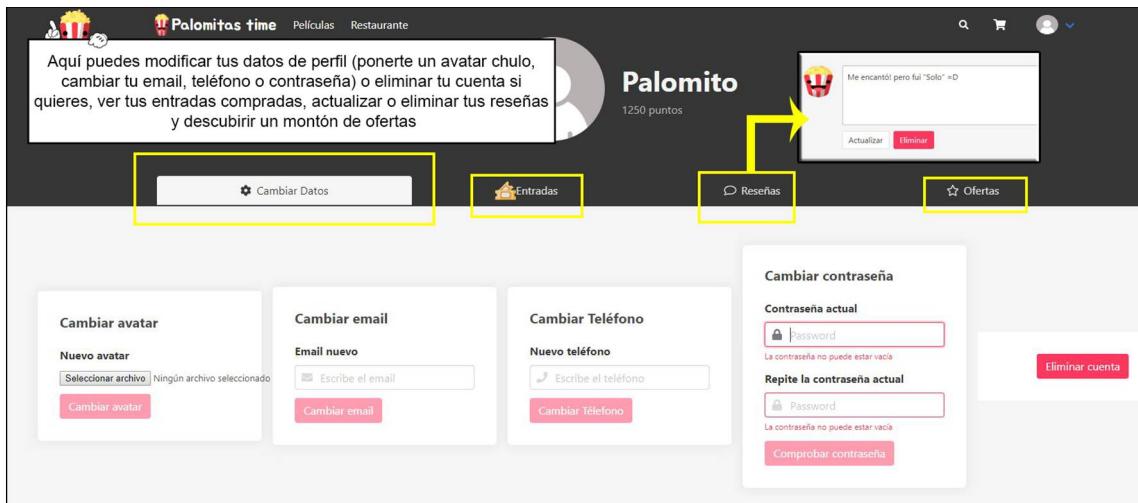


Figura 14: Manual de usuario (7/12).

## PalomitasTime

### Página 6

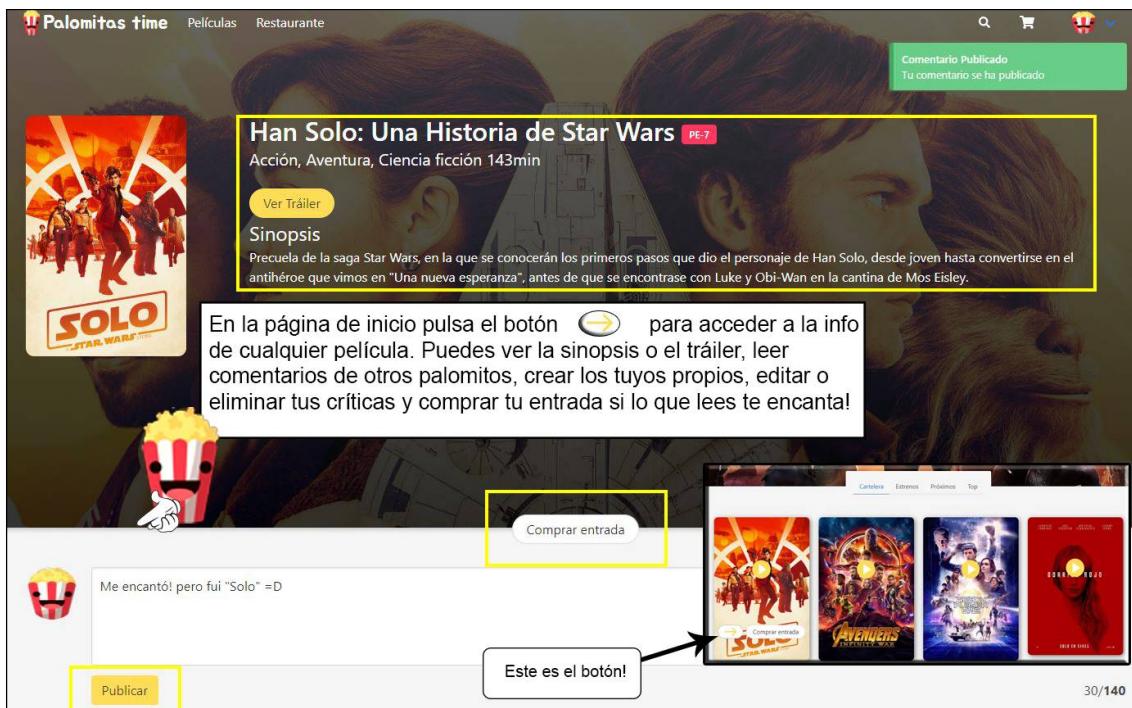


Figura 15: Manual de usuario (8/12).

### Página 7

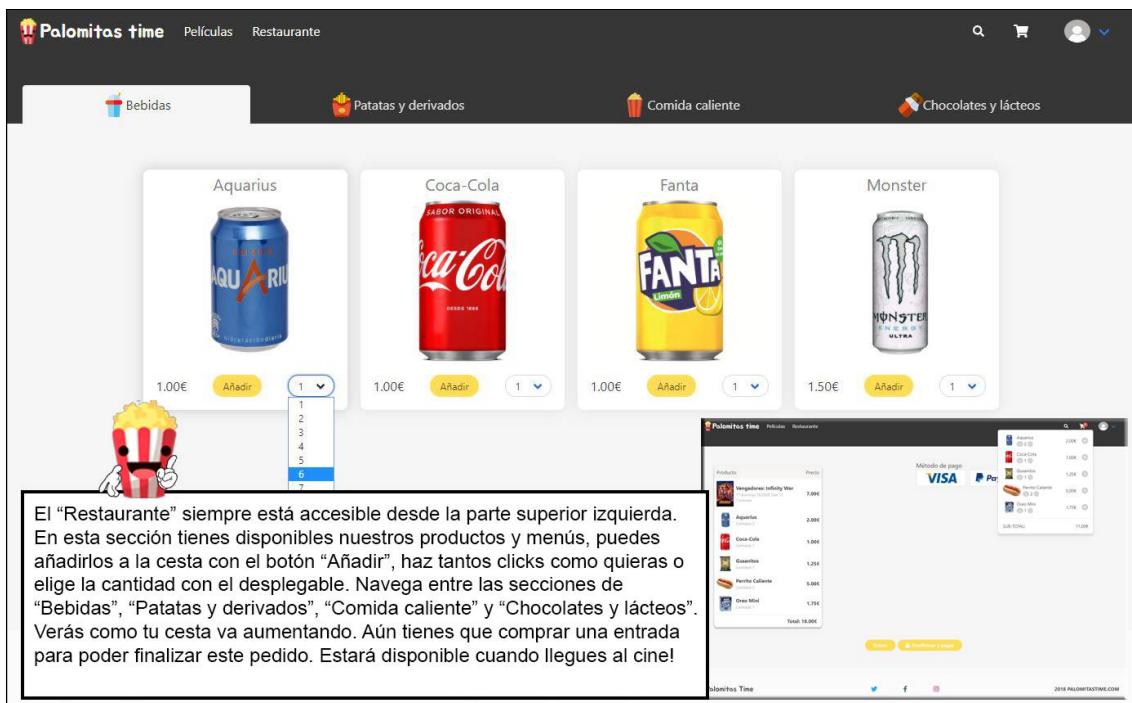


Figura 16: Manual de usuario (9/12).

## PalomitasTime Página 8



Figura 75: Manual de usuario (10/12).

## Página 9

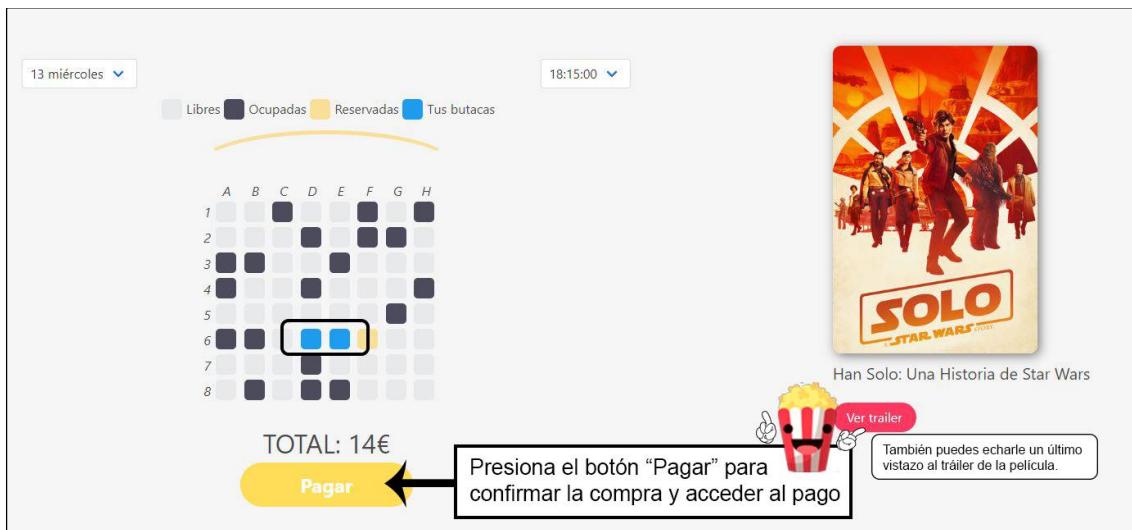


Figura 18: Manual de usuario (11/12).



Figura 19: Manual de usuario (12/12).

#### 8.4.. Manual de administrador.

En este Manual de Administrador proporcionado por los desarrolladores de PalomitasTime, aprenderás a:

1. Gestionar tu perfil como Administrador o Súper Administrador.
- 1.1 Como Súper Administrador, gestionar el resto de administradores.
2. Realizar todas las gestiones relacionadas con las películas de la plataforma.
3. Gestionar el Slider de la web.
4. Gestionar las salas del cine: creación de nuevas salas o edición de las existentes, bloqueo de butacas o filas específicas
5. Gestión de las sesiones del cine y la web: creación y uso de las plantillas de las sesiones de cada película, modificación de los mismos.
6. Gestión de la tienda: creación o modificación de las categorías de la tienda online, creación o modificación de los productos y creación o modificación de los menús (o packs).

## 1. Administradores: el Administrador y el Súper Administrador.

The screenshot shows the PalomitasTime administrator interface. On the left, there's a sidebar with a dark background containing several menu items under categories: 'ADMINISTRADORES' (Perfil, Gestionar administradores), 'CINE' (Películas, Slider, Salas, Sesiones), and 'TIENDA' (Productos, Menús). The main content area has a light gray background. At the top, it says 'Conectado como Lorena' and 'Home'. Below that, it says 'Panel de administración.' and 'Desde aquí puedes gestionar todos los productos y servicios que se pueden ver en la aplicación web.'

Figura 20: Manual de administrador. Administradores (1/7).

Al conectarnos veremos nuestro perfil como Administrador o como Súper administrador. Las diferencias son notables. Mientras que el Súper Administrador puede crear y modificar administradores o eliminar los existentes, el Administrador no.

Tanto el Administrador como el Súper Administrador pueden realizar modificaciones en su propio perfil. El Súper Administrador además, puede modificar los perfiles completos de cada administrador. Para modificar tu perfil entra en "Perfil" y verás los "Datos de la cuenta", pulsa sobre el lápiz para modificar el dato que precises. Pulsa en "Guardar cambios" para salvar tus nuevos datos.

This screenshot shows the 'Perfil' (Profile) edit page. The sidebar is identical to Figure 20. The main content area shows a form titled 'Datos de la cuenta' (Account Data). It contains three fields: 'Nombre: Lorena' with an edit icon, 'Email: tranity06@gmail.com' with an edit icon, and 'Contraseña: \*\*\*\*\*' with an edit icon. Below the form is a blue button labeled 'Guardar cambios' (Save changes).

Figura 21: Manual de administrador. Administradores (2/7).

## PalomitasTime

Al modificar un valor, hay validarla pulsando el botón “Aceptar”, si existe algún error, se mostrará y deberemos introducir los datos de manera correcta. Si no existen errores, vemos el dato modificado, para salvar los nuevos datos pulsamos en “Guardar cambios”.



Figura 22: Manual de administrador. Administradores (3/7).

Para crear un nuevo administrador con tu cuenta de Súper Administrador, accede desde el apartado “Gestionar administradores” y pulsa en “Crear”, rellena cada apartado correspondiente. Las contraseñas deben de tener:

- Al menos una mayúscula
- Al menos una minúscula
- Al menos un número
- Al menos un carácter especial
- Una longitud de al menos seis caracteres

## PalomitasTime

The screenshot shows the 'PalomitasTime' administrator interface. On the left, there's a sidebar with sections for 'ADMINISTRADORES' (Perfil, Gestionar administradores, Crear, Ver todos), 'CINE' (Películas, Slider, Salas, Sesiones), and 'TIENDA'. The main area is titled 'Conectado como Lorena' and shows the 'Home > Crear nuevo usuario' path. It has fields for 'Nombre' (with an empty input), 'Email' (with an empty input), 'Contraseña' (with an empty input), and 'Repetir contraseña' (with an empty input). A blue 'Guardar' button is at the bottom.

Figura 23: Manual de administrador. Administradores (4/7).

Cuando hayamos introducido los cambios pulsamos “Guardar” para finalizar el proceso y crear el nuevo Administrador. Si algo no ha salido bien, veremos un mensaje de error en rojo, introducimos de nuevo los datos y verificamos que la contraseña cumple los requisitos especificados arriba y que esta coincida las dos veces que la debemos introducir.

The screenshot shows the 'PalomitasTime' administrator interface. The sidebar is identical to Figure 23. The main area is titled 'Conectado como Lorena' and shows the 'Home > Lista de usuarios' path. It displays a table titled 'Administradores registrados' with two rows of data:

ID	Nombre	Email	Modificar	Borrar
1	Lorena	tranity06@gmail.com		
2	Andrei	andrei@gmail.com		

Figura 24: Manual de administrador. Administradores (5/7).

Una vez creado el nuevo Administrador, lo veremos en el apartado “Ver todos” dentro del menú “Gestionar administradores”. Como hemos comentado, podemos modificar los datos del Administrador en el ícono del lápiz en la columna “Modificar” o borrarlo directamente con el ícono de la papelera en la columna “Borrar”. Cuando modifiques un Administrador, rellenamos todos los campos correctamente como cuando lo creamos por primera vez, validamos los datos con el botón “Aceptar” para comprobar que no existen errores o duplicidades de nombres o emails. Introducimos de nuevo dos veces la contraseña con las características que nombramos arriba la primera vez y pulsamos “Guardar los cambios” para salvar los nuevos datos.

## PalomitasTime

ID	Nombre	Email	Modificar	Borrar
1	Lorena	tranity06@gmail.com		
2	Andrei	andrei@gmail.com		

Nombre:

Email:

Contraseña:



**Aceptar**

**Guardar cambios**

Figura 25: Manual de administrador. Administradores (6/7).

Si decidimos eliminar un Administrador, veremos un mensaje de confirmación que deberemos aceptar para suprimirlo definitivamente.

palomitastime.online dice

¿Seguro que quieres eliminar al administrador Andrei ?

**Aceptar** **Cancelar**

Home > Lista de usuarios

Administradores registrados

ID	Nombre	Email	Modificar	Borrar
1	Lorena	tranity06@gmail.com		
2	Andrei	andrei@gmail.com		

Figura 26: Manual de administrador. Administradores (7/7).

## 2. Cine: películas, slider, salas y sesiones.

En el apartado “Cine” tenemos todas las opciones relacionadas con las películas y las salas.

### 2.1 “Películas”:

-Registrar nueva: para añadir nuevas películas introducimos el nombre del título que queramos registrar en la web en el apartado “Título de la película”, veremos varios títulos que coinciden con las búsquedas (antiguas, nuevas y futuras). La base de datos recoge automáticamente la información de todas ellas, no hace falta consultar nada de forma externa a la aplicación.

Título	Fecha de estreno	Mostrar
Jurassic World	2015-06-06	
Jurassic World 2: El reino caído	2018-06-06	
El mundo perdido (Jurassic Park)	1997-05-23	
LEGO Jurassic World: The Indominus Escape	2016-10-03	
Jurassic World 3	2021-06-11	

Figura 27: Manual de administrador. Cine (1/28).

Para añadir la película a la web, tan solo debemos pulsar sobre el ícono del lápiz, veremos, muy detallada, la información recogida sobre esa película de forma automática (no tenemos por qué llenar ningún dato, pero podemos modificarlos si se requiere).

# PalomitasTime

PalomitasTime

ADMINISTRADORES

- Perfil
- Gestionar administradores

CINE

- Películas
- Registrar nueva
- Ver todas
- Slider
- Salas
- Sesiones

TIENDA

- Productos
- Menús

Conectado como Lorena

Home > Registrar nueva película

### Registrar nueva película

Título de la película:

Buscar

Título:

Título original:

Fecha de estreno:

Géneros:

Director:

Actores:

Sinopsis:

Tras cuatro años de abandono del complejo turístico Jurassic World, Isla Nublar sólo está habitada por los dinosaurios supervivientes. Cuando el volcán de la isla entra en erupción, Owen (Chris Pratt) y Claire (Bryce Dallas Howard) vuelven allí para rescatar a los dinosaurios de la extinción. Owen va en busca de Blue, el raptor al que creó mientras que Claire, que ha empezado a valorar a estas criaturas, se centra más en salvar al resto. Cuando llegan a la isla descubren una conspiración que pretende llevar al planeta de nuevo a la era prehistórica.

Duración:

Cartel:



## PalomitasTime

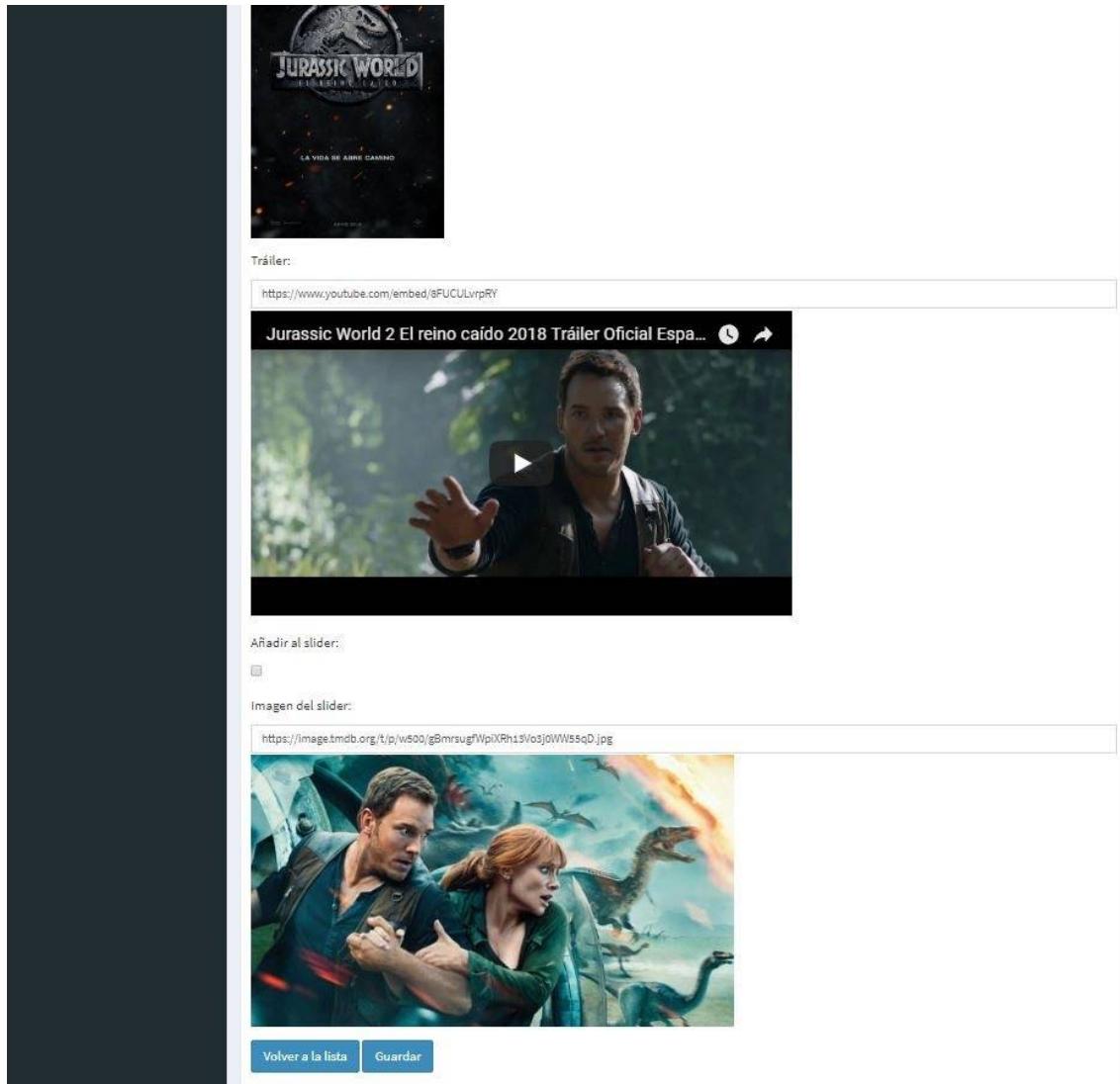


Figura 28: Manual de administrador. Cine (2/28).

Si nos hemos equivocado de película podemos volver al menú anterior pulsando en “Volver a la lista”. Si es correcta, pulsaremos en “Guardar” y ya aparecerá en la lista de películas registradas en el cine (a partir de ahí podrás crear sesiones y añadir la imagen al slider de dicha película, veremos cómo hacerlo más adelante).

## PalomitasTime

The screenshot shows the PalomitasTime administrator dashboard. On the left, there's a sidebar with sections for 'ADMINISTRADORES' (Perfil, Gestionar administradores) and 'CINE' (Películas, Registrar nueva, Ver todas, Slider, Salas, Sesiones). The main content area is titled 'Registrar nueva película'. It displays a green success message: 'Película nueva registrada.' Below this, there's a search form with a placeholder 'Título de la película:' and a 'Buscar' button.

Figura 29: Manual de administrador. Cine (3/28).

Este mensaje de confirmación da por finalizado nuestro registro. Si la película ya estaba registrada veremos un mensaje de error.

The screenshot shows the same 'Registrar nueva película' page. However, the message is red and contains the text 'La película ya estaba registrada en la base de datos.' (The movie was already registered in the database).

Figura 30: Manual de administrador. Cine (4/28).

Para ver esa y todas las películas registradas en el cine pulsamos en el submenú "Ver todas" dentro del menú "Películas" del apartado "Cine".

## PalomitasTime

Nombre	Estreno	Duración	Sesiones activas	Borrar
Vengadores: Infinity War	2018-04-25	149	TODO	
Thor: Ragnarok	2017-10-25	130	TODO	
Jumanji: Bienvenidos a la jungla	2017-12-09	119	TODO	
Un monstruo viene a verme	2016-10-07	108	TODO	
12 valientes	2018-01-18	130	TODO	
Ready Player One	2018-03-28	140	TODO	
Gorrión rojo	2018-03-01	139	TODO	
Han Solo: Una Historia de Star Wars	2018-05-23	143	TODO	
Jurassic World 2: El reino caído	2018-06-06	154	TODO	

Figura 31: Manual de administrador. Cine (5/28).

- “Ver todas”: como vemos arriba, los títulos de las películas registradas en la web aparecen en esta sección. Podemos borrarla si ya no vamos a dar más pases de esa película o si nos hemos equivocado al añadirla. Este apartado no permite modificación, para modificar un dato concreto de la película, bórrala y añádela de nuevo siguiendo los pasos anteriores de “Registrar nueva”.

## PalomitasTime

The screenshot shows the PalomitasTime online administrator interface. At the top, there is a confirmation dialog box with the text "palomitastime.online dice" and "¿Seguro que quieres eliminar esta película?". Below the dialog, there are two buttons: "Aceptar" (Accept) and "Cancelar" (Cancel). In the background, the main page displays a list of registered movies with columns for Nombre (Name), Estreno (Release Date), Duración (Duration), Sesiones activas (Active Sessions), and Borrar (Delete). The movies listed are: Vengadores: Infinity War (2018-04-25, 149, TODO, delete icon), Thor: Ragnarok (2017-10-25, 130, TODO, delete icon), Jumanji: Bienvenidos a la jungla (2017-12-09, 119, TODO, delete icon), Un monstruo viene a verme (2016-10-07, 108, TODO, delete icon), 12 valientes (2018-01-18, 130, TODO, delete icon), Ready Player One (2018-03-28, 140, TODO, delete icon), Gorrión rojo (2018-03-01, 139, TODO, delete icon), Han Solo: Una Historia de Star Wars (2018-05-23, 143, TODO, delete icon), and Jurassic World 2: El reino caído (2018-06-06, 154, TODO, delete icon). The last movie in the list is highlighted with a gray background.

Nombre	Estreno	Duración	Sesiones activas	Borrar
Vengadores: Infinity War	2018-04-25	149	TODO	
Thor: Ragnarok	2017-10-25	130	TODO	
Jumanji: Bienvenidos a la jungla	2017-12-09	119	TODO	
Un monstruo viene a verme	2016-10-07	108	TODO	
12 valientes	2018-01-18	130	TODO	
Ready Player One	2018-03-28	140	TODO	
Gorrión rojo	2018-03-01	139	TODO	
Han Solo: Una Historia de Star Wars	2018-05-23	143	TODO	
Jurassic World 2: El reino caído	2018-06-06	154	TODO	

Figura 32: Manual de administrador. Cine (6/28).

Para borrar una película de la sección “Ver todas” debemos aceptar un mensaje de confirmación. Este también te avisa si dicha película tiene pases programados.

## 2.2 El Slider:

Nos permite que los carteles de tres de las películas de la semana aparezcan destacados en la web. Se observa una imagen en buena calidad que pasa automáticamente cada pocos segundos en la parte superior de PalomitasTime.online. Estas llaman rápidamente la atención del usuario por lo que es importante reflejar las películas más importantes (estrenos, eventos...) en dicha sección. El máximo de películas en el Slide son tres. Si ya tienes tres verás bloqueada en gris la barra de búsqueda encima de “Añadir”.

## PalomitasTime

Conectado como Lorena

Películas activas en el slider

Añadir película (máximo 3):

Un monstruo viene a verme

Añadir

Nombre	Borrar
Vengadores: Infinity War	
Thor: Ragnarok	
Jumanji: Bienvenidos a la jungla	

Figura 33: Manual de administrador. Cine (7/28).

Podemos borrar uno de los títulos con el botón de la papelera en la columna “Borrar”. Justo encima de “Añadir” podemos escribir un nuevo título y al pulsar “Añadir” lo veremos reflejado en la lista.

Ready Player One

Añadir

Nombre	Borrar
Vengadores: Infinity War	
Thor: Ragnarok	

Figura 34: Manual de administrador. Cine (8/28).

Nombre	Borrar
Vengadores: Infinity War	
Thor: Ragnarok	
Ready Player One	

Figura 35: Manual de administrador. Cine (9/28).

## 2.3 Las "Salas":

**-“Mostrar”:** En este apartado podemos ver todas las salas creadas en nuestro cine. Cada una contiene un “Número de sala” (que debería coincidir con las salas físicas del cine), un “Aforo” (que debería coincidir con el aforo físico limitado por las butacas de cada sala) y un número de “Sesiones” (que refleja el número total de sesiones de varios días).

The screenshot shows the 'PalomitasTime' administrator dashboard. On the left, there's a sidebar with categories like 'ADMINISTRADORES', 'CINE' (selected), and 'TIENDA'. Under 'CINE', 'Salas' is selected. The main area displays a table titled 'Lista de salas registradas.' with columns: N° de sala, Aforo, Sesiones, Detalles, and Borrar. The table contains five rows with data: (1, 64, 27, edit icon, delete icon), (2, 64, 27, edit icon, delete icon), (3, 64, 27, edit icon, delete icon), (4, 64, 27, edit icon, delete icon), and (5, 64, 26, edit icon, delete icon). At the top right of the main area, it says 'Conectado como Lorena'.

Figura 36: Manual de administrador. Cine (10/28).

Podemos entrar en los “Detalles” de cada sala pulsando en el ícono del lápiz, o borrar toda la sala pulsando en el ícono de la papelera en la columna “Borrar”.

This screenshot shows a confirmation dialog box over the administrator interface. The dialog asks, “¿Seguro que quieres eliminar la sala 1?”. It has two buttons: “Aceptar” (Accept) and “Cancelar” (Cancel). The background shows the same table from Figure 36, with row 1 highlighted in grey. The top navigation bar includes “palomitastime.online dice”, “al - Yoast SEO”, “Otros marcadores”, and a “Cerrar” button.

Figura 37: Manual de administrador. Cine (11/28).

Si entramos en los “Detalles” de una sala, podremos bloquear butacas para que el sistema no deje que un cliente las reserve.

## PalomitasTime

The screenshot shows the PalomitasTime administrator interface. On the left, there is a sidebar with the following menu items under 'ADMINISTRADORES': 'Perfil' (selected), 'Gestionar administradores', 'CINE' (selected), 'Películas', 'Slider', 'Salas' (selected), 'Mostrar', 'Crear', and 'Sesiones'. Under 'CINE', there are sub-options: 'Nº Sala: 1', 'Aforo: 64', 'Butacas bloqueadas: 35' (with a pencil icon), 'Mostrar' (button), 'Sesiones: 27', and another 'Mostrar' button. At the top right, there is a 'Cerrar' (Close) button.

Figura 38: Manual de administrador. Cine (12/28).

Para hacerlo pulsaremos en el ícono del lápiz junto a “Bloquear Butacas” y seleccionaremos la fila y la butaca que queremos bloquear, podemos seleccionar varias butacas de la misma fila pulsando sobre los distintos números que muestra el desplegable. También podemos bloquear una fila de butacas entera marcando en el recuadro de “Toda la fila”. Veremos en azul las butacas recién bloqueadas y en gris/negro las que ya estaban.

## PalomitasTime

The screenshot shows the PalomitasTime administrator interface. The left sidebar has a dark theme with white text. It includes sections for ADMINISTRADORES (Perfil, Gestionar administradores), CINE (Películas, Slider, Salas, Mostrar, Crear, Sesiones), and TIENDA (Productos, Menús). The 'Salas' section is currently selected. The main content area is titled 'Detalle sala 1' and shows information for 'Nº Sala: 1' (Aforo: 64, Butacas bloqueadas: 41). It includes a dropdown for 'Fila' (set to 1) and a checkbox for 'Toda la fila'. Buttons for 'x 3', 'x 5', and 'Bloquear' are present. Below this is a 'Ocultar' button. A table lists 'Butaca' numbers (3, 6, 4, 3, 2, 1, 5, 7) with a trash icon in the 'Desbloquear' column for each row.

Fila	Butaca	Desbloquear
1	3	
1	6	
1	4	
1	3	
1	2	
1	1	
1	5	
1	7	

Figura 39: Manual de administrador. Cine (13/28).

Para modificar las “Butacas bloqueadas” pulsamos en el botón azul “Mostrar” debajo de “Butacas bloqueadas”. De ese modo vemos las columnas “Fila” y “Butaca” que representan los asientos bloqueados del cine. Para desbloquearlos, pulsaremos en el ícono de la papelera de la columna “Desbloquear”, debemos aceptar un mensaje de confirmación por cada butaca que queramos liberar. La información sobre las sesiones es solo representativa y no podemos modificarlas desde ese apartado.

## PalomitasTime

The screenshot shows a software interface titled "PalomitasTime". On the left, there's a sidebar with sections for "TIENDA", "ADMINISTRADORES", and "CINE". Under "CINE", "Salas" is selected, and under it, "Crear" is highlighted. The main content area displays a table titled "Sesiones: 28". The table has columns for "Fecha" (Date) and "Hora" (Time). The data shows sessions on June 17 and 18 at various times. There are delete icons in the top right corner of each row.

Figura 40: Manual de administrador. Cine (14/28).

-“**Crear**”: permite crear nuevas salas si es necesario, para ello, introduce el número de la nueva sala y pulsa el botón azul “**Crear**”. De momento no se pueden modificar el número de filas o butacas.

This screenshot shows the "Crear" (Create) page for creating a new hall. The sidebar on the left shows "CINE" selected. The main form has fields for "Nº de sala" (Hall number) set to 6, "Nº de filas" (Number of rows) set to 8, and "Nº de butacas por fila" (Number of seats per row) set to 8. A blue "Crear" button is at the bottom of the form.

Figura 41: Manual de administrador. Cine (15/28).

Veremos una confirmación de que nuestra nueva sala está creada o un error si el número de sala ya existía.

## PalomitasTime

The screenshot shows the PalomitasTime administrator interface. On the left, there's a sidebar with sections for ADMINISTRADORES (Perfil, Gestionar administradores), CINE (Películas, Slider, Salas, Mostrar, Crear), and other options like Home, Crear sala, and Salas. The main area is titled 'Crear nueva sala.' and shows a green success message box containing 'Sala creada.' Below it, there are input fields for 'Nº de sala:' (with value '6') and 'Nº de filas:' (with value '8'). At the top right, there's a 'Cerrar' button.

Figura 42: Manual de administrador. Cine (16/28).

This screenshot shows the same interface as Figure 42, but with an orange error message box at the top stating 'La sala ya existe.' (The hall already exists). The input field for 'Nº de sala:' contains the value '8'. The rest of the interface is identical to Figure 42.

Figura 43: Manual de administrador. Cine (17/28).

Si hemos obtenido el mensaje de confirmación ya veremos la nueva sala creada en el submenú “Mostrar” dentro del apartado “Cine” (que vimos antes).

This screenshot shows the 'Lista de salas registradas.' (List of registered halls) page. It features a table with columns: Nº de sala, Aforo, Sesiones, Detalles, and Borrar. The table contains seven rows, each representing a hall with the following data:

Nº de sala	Aforo	Sesiones	Detalles	Borrar
1	64	28		
2	64	28		
3	64	28		
4	64	28		
5	64	28		
6	64	0		
8	64	0		

Figura 44: Manual de administrador. Cine (18/28).

## 2.4 Las “Sesiones”:

En el apartado sesiones se gestionan los pases de cada película y la hora de los pases. Esto se basa en unas plantillas que también son modificables. Dentro del menú sesiones está el submenú “Plantillas” y el “Calendario”.

### -Plantillas:

Dentro de “Plantillas” encontramos dos herramientas.

- El submenú “Crear”. Aquí se generan las plantillas que facilitarán la planificación de sesiones al Administrador. Para crear una nueva plantilla, se introduce un nombre, una descripción y se pulsa en el botón “Crear”.

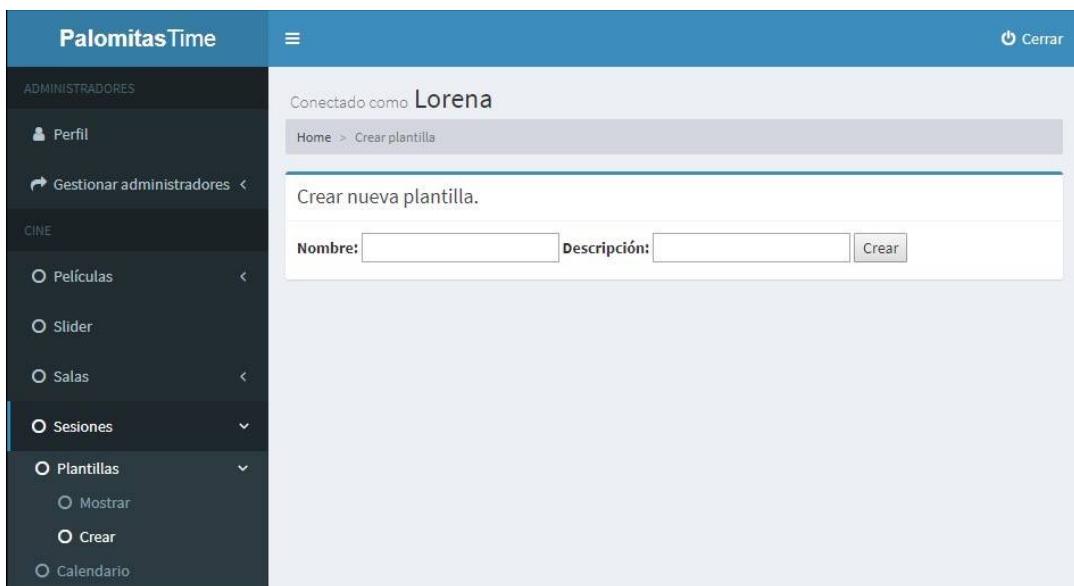


Figura 45: Manual de administrador. Cine (19/28).

Automáticamente se genera una cuadrícula con cuatro pases y el número de salas que existan en el menú “Salas” del apartado “Cine” que se mostró anteriormente (se han eliminado las salas 6 y 8 para dejar las 5 originales).

## PalomitasTime

Figura 46: Manual de administrador. Cine (20/28).

Se tipografián a mano cada una de las sesiones de la plantilla y se guardan los datos con el botón “guardar cambios”. En este paso aún se puede modificar sin problema el nombre y la descripción de la plantilla. Se pueden generar un número ilimitado de plantillas.

Figura 47: Manual de administrador. Cine (21/28).

- El submenú “Mostrar”. Lista todas las plantillas creadas. Para modificar una plantilla se pulsa en el ícono del lápiz de la columna “Sesiones”, para borrar una plantilla pulsamos en el ícono de la papelera de la columna “Borrar”.

## PalomitasTime

The screenshot shows the PalomitasTime administrator interface. The left sidebar has a dark theme with white text. It includes sections for ADMINISTRADORES (Perfil, Gestionar administradores), CINE (Películas, Slider, Salas, Sesiones, Plantillas, Mostrar, Crear, Calendario). The 'Plantillas' section is currently selected. The main content area is titled 'Plantillas de sesiones.' and shows a table with one row: 'Ejemplo Plantilla' (Descripción: 'Descripción'). There are edit and delete icons next to the row.

Figura 48: Manual de administrador. Cine (22/28).

Si se edita una plantilla entraremos en “Detalle plantilla”. Aquí se puede modificar el nombre, la descripción y las horas de cada pase y sala de una plantilla guardada. Se guardan los cambios con el botón “Guardar cambios”.

The screenshot shows the 'Detalle plantilla Ejemplo Plantilla' screen. The left sidebar is identical to Figure 48. The main content area shows the template details: Nombre: 'Ejemplo Plantilla', Descripción: 'Descripción', and a 'Modificar' button. Below is a table for scheduling showtimes across five screens (Sala 1 to Sala 5) for four sessions (Pase 1 to Pase 4). The table shows the following data:

Sala	Pase 1	Pase 2	Pase 3	Pase 4
1	--:--	22:00	--:--	--:--
2	--:--	--:--	--:--	--:--
3	--:--	--:--	--:--	--:--
4	--:--	--:--	--:--	23:00
5	--:--	--:--	--:--	--:--

A 'Guardar cambios' button is at the bottom of the table.

Figura 49: Manual de administrador. Cine (23/28).

## -Calendario

En el submenú “Calendario” se programan los pases de cada día del mes de forma rápida gracias a las plantillas creadas anteriormente.

## PalomitasTime

The screenshot shows the PalomitasTime software interface. On the left, there is a sidebar with the following navigation options under 'CINE': Películas, Slider, Salas, Sesiones (which is currently selected), Plantillas, and Calendario. The main content area is titled 'Sesiones.' and displays a form with fields for 'Fecha' (dd/mm/aaaa), 'Plantillas' (Ejemplo Plantilla), and four dropdown menus labeled 'Pase 1', 'Pase 2', 'Pase 3', and 'Pase 4'. Below the form is a 'Guardar cambios' button.

Figura 50: Manual de administrador. Cine (24/28).

Se escribe la fecha manualmente a la derecha de “Fecha” o se despliega un calendario donde se selecciona el día para programar las sesiones.

This screenshot is similar to Figure 50, but the 'Fecha' field is now a date picker showing 'junio de 2018'. The calendar grid for June 2018 is displayed, with the 17th highlighted. The other interface elements remain the same, including the 'Plantillas' dropdown and the four pase dropdowns.

Figura 51: Manual de administrador. Cine (25/28).

Una vez elegido el día, la plantilla seleccionada en el apartado “Plantillas”, a la derecha de la fecha, lanza automáticamente todas las horas recogidas en la plantilla elegida.

## PalomitasTime

Figura 52: Manual de administrador. Cine (26/28).

Los desplegables “-Seleccionar película-” muestran los títulos de las películas que se registran en el menú “Películas” del apartado “Cine”, se selecciona el título requerido y se modifican las horas si es necesario. Para validar los datos, se pulsa en el botón “Guardar cambios”.

Figura 53: Manual de administrador. Cine (27/28).

Para modificar las horas de las sesiones o las películas, se busca de nuevo en “Fecha” el día, se mostrarán todas las sesiones planificadas ese día y sus horas, tras los cambios, se validan los datos con el botón “Guardar cambios”.

## PalomitasTime

The screenshot shows the 'Cine' section of the PalomitasTime software. On the left, there's a sidebar with 'ADMINISTRADORES' (Perfil, Gestión de administradores), 'CINE' (Películas, Slider, Salas, Sesiones, Plantillas, Calendario), and 'TIENDA' (Categorías, Productos, Menús). The main area is titled 'Sesiones' and shows a grid for scheduling movies. The grid has 'Sala' (Row) and 'Pase 1', 'Pase 2', 'Pase 3', 'Pase 4' (Column) headers. Rows represent movie showtimes, and columns represent different screening sessions. Each cell contains a dropdown menu. A search bar at the top says 'Fecha: 18/06/2018' and 'Plantilla: Ejemplo Plantilla'. A 'Guardar cambios' button is at the bottom.

Figura 54: Manual de administrador. Cine (28/28).

## 3. Tienda

El apartado tienda se divide en los menús “Categorías”, “Productos” y “Menús”

### 3.1 Categorías.

Gestiona la clasificación de los productos según su naturaleza. Existen dos submenús:

- “Crear”. Permite introducir nuevas categorías que engloben nuestros productos.

## PalomitasTime

The screenshot shows the PalomitasTime administrator interface. On the left, there is a sidebar with the following navigation options:

- ADMINISTRADORES**
  - Perfil
  - Gestionar administradores
- CINE**
  - Películas
  - Slider
  - Salas
  - Sesiones
- TIENDA**
  - Categorías
    - Crear
    - Ver todas
  - Productos
  - Menús

The main content area is titled "Crear nueva categoría" (Create new category). It has a form with the label "Nombre de la categoría:" (Category name:) and a text input field containing "Prueba". Below the input field is a blue "Crear" (Create) button.

Figura 55: Manual de administrador. Cine (1/23).

Para crear una categoría se introduce el “Nombre de la categoría” en la barra de escritura y se pulsa en el botón azul “Crear”. Un aviso nos confirma que la nueva categoría ha sido creada.

The screenshot shows the "Crear nueva categoría" (Create new category) screen again. A green success message box at the top states "Categoría creada." (Category created). Below it is the same form as in Figure 55, with the "Nombre de la categoría:" (Category name:) field containing "Prueba" and the "Crear" (Create) button.

Figura 56: Manual de administrador. Cine (2/23).

- “Ver todas”. Muestra las categorías que hemos creado anteriormente. Con el ícono del lápiz de la columna “Editar” podemos modificar el nombre de la categoría.

## PalomitasTime

The screenshot shows the administrator interface for 'PalomitasTime'. The left sidebar has a dark theme with white text. It includes sections for 'ADMINISTRADORES' (Perfil, Gestionar administradores), 'CINE' (Películas, Slider, Salas, Sesiones), and 'TIENDA' (Categorías, Crear, Ver todas, Productos, Menús). The 'Categorías' section is currently selected. The main content area is titled 'Categorías existentes' and displays a table of existing categories:

Nombre	Editar	Borrar
Bebidas		
Patatas y derivados		
Comida caliente		
Chocolates y lácteos		
Todos		
Prueba		
sad		

Figura 57: Manual de administrador. Cine (3/23).

The screenshot shows the administrator interface for 'PalomitasTime'. The left sidebar has a dark theme with white text. It includes sections for 'ADMINISTRADORES' (Perfil, Gestionar administradores), 'CINE' (Películas, Slider, Salas, Sesiones), and 'TIENDA' (Categorías, Crear, Ver todas, Productos, Menús). The 'Categorías' section is currently selected. The main content area is titled 'Editando categoría Prueba' and displays a form for editing the category 'Prueba':

Nombre de la categoría:  
Pruebaaa|

**Editar**

Figura 58: Manual de administrador. Cine (4/23).

Tras tipografiar el nuevo nombre se pulsa en el botón azul "Editar". Un cartel de confirmación muestra que la categoría se ha editado correctamente.

## PalomitasTime

Editando categoría Prueba

Categoría editada correctamente.

Nombre de la categoría:

Pruébaa

Editar

Figura 59: Manual de administrador. Cine (5/23).

Para borrar una categoría, se usa el ícono de la papelera de la columna “Borrar”.

The screenshot shows the PalomitasTime administrator interface. On the left, there is a sidebar with categories: ADMINISTRADORES (Perfil, Gestión de administradores), CINE (Películas, Slider, Salas, Sesiones), and TIENDA (Categorías, Crear, Ver todas, Productos, Menús). The 'Categorías' item under TIENDA is selected and expanded. The main content area is titled 'Categorías existentes'. It displays a table with the following data:

Nombre	Editar	Borrar
Bebidas		
Patatas y derivados		
Comida caliente		
Chocolates y lácteos		
Todos		
Pruebaa		
sad		

Figura 60: Manual de administrador. Cine (6/23).

Una notificación nos advierte de que una categoría va a ser eliminada. Tras aceptar se ve como la categoría desaparece de la lista.

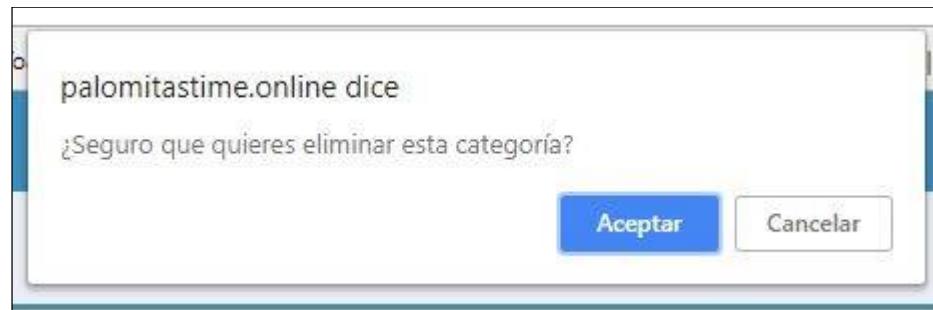


Figura 61: Manual de administrador. Cine (7/23).

### 3.2 Productos.

En este menú se crean nuevos productos o se editan los existentes, estos son los que se agruparán en las distintas categorías. Este menú contiene los submenús “Crear” y “Ver todos”.

-“Crear”: En este submenú se introduce el nombre del nuevo producto en el apartado “Nombre del producto”, el precio en el apartado “Precio del producto”, hay que seleccionar una imagen que representativa del producto pulsando en “Seleccionar archivo”, la imagen debe de estar en tu dispositivo, se navega por las carpetas del dispositivo y se selecciona la imagen del producto, luego se pulsa en aceptar.

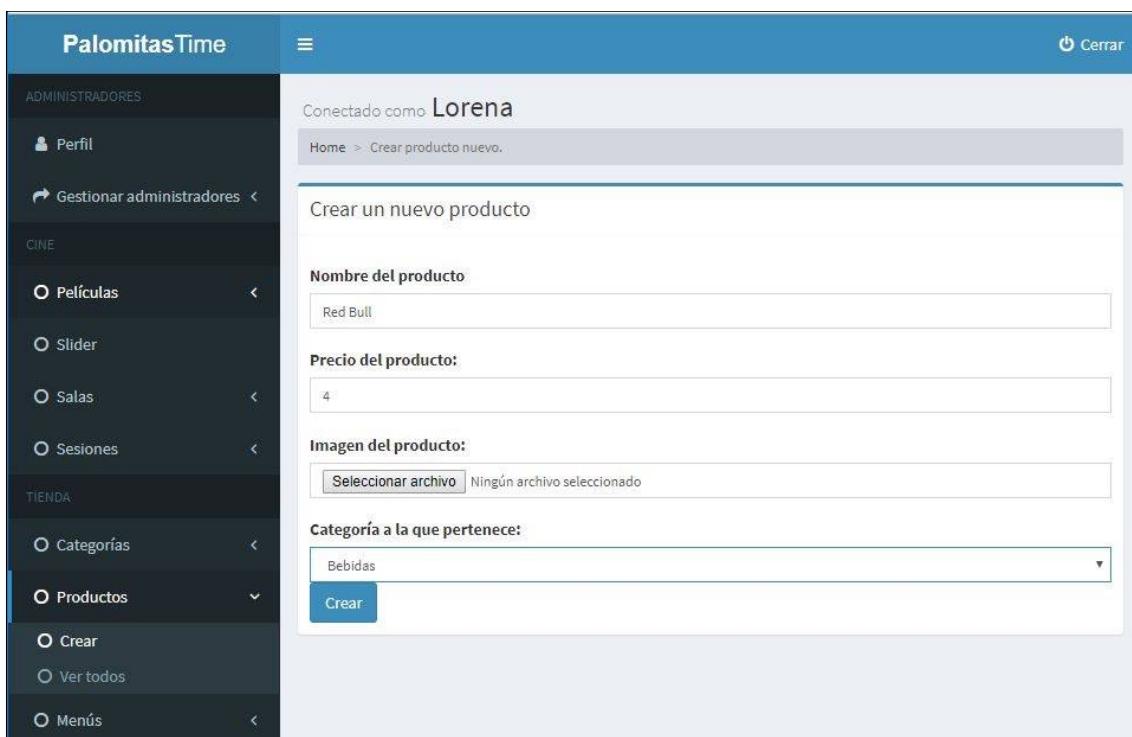


Figura 62: Manual de administrador. Cine (8/23).

También hay que desplegar el apartado “Categoría a la que pertenece” y seleccionarla de la lista (aparecen las creadas en el menú “Categoría” explicado anteriormente).

## PalomitasTime



Figura 63: Manual de administrador. Cine (9/23).

Tras introducir todos los datos de forma correcta, se pulsa en el botón azul “Crear”. Un aviso indica que el producto se ha creado correctamente.



Figura 64: Manual de administrador. Cine (10/23).

Si falta por introducir algún dato o existe algún error, un mensaje avisa de que el producto no ha sido creado.



Figura 65: Manual de administrador. Cine (11/23).

-“Ver todos”: Este submenú muestra los productos almacenados en la base de datos. Se pueden modificar los datos con el icono del lápiz de la columna “Editar”.

## PalomitasTime

The screenshot shows the 'Productos' section of the administrator interface. On the left, there's a sidebar with categories like Películas, Slider, Salas, Sesiones, Categorías, and Menús. The 'Productos' category is selected. The main area displays a table titled 'Productos existentes' with columns: Imagen, Nombre, Precio, Editar, and Borrar. Each row represents a product with its name, price, and edit/delete icons.

Imagen	Nombre	Precio	Editar	Borrar
/uploads/productos/aquarius.png	Aquarius	1.00		
/uploads/productos/coca-cola.png	Coca-Cola	1.00		
/uploads/productos/fanta.png	Fanta	1.00		
/uploads/productos/monster.png	Monster	1.50		
/uploads/productos/boca-bits.png	Boca Bits	1.50		
/uploads/productos/gusanitos.png	Gusanitos	1.25		
/uploads/productos/patatas.png	Patatas Fritas	1.50		
/uploads/productos/bollicao-mini.png	Bollicao Mini	1.75		
/uploads/productos/donuts-choco.png	Donuts Choco	1.85		

Figura 66: Manual de administrador. Cine (12/23).

Al pulsar sobre el lápiz vamos al apartado “Editando producto”. Se puede modificar el nombre, el precio y la categoría a la que pertenece el producto.

The screenshot shows the 'Editando producto' form for the product 'RED BULL'. The sidebar is identical to Figure 66. The main area has fields for 'Nombre del producto:' (RED BULL), 'Precio del producto:' (4,00), and 'Categoría a la que pertenece:' (Bebidas). A blue 'Editar' button is at the bottom of the form.

Figura 67: Manual de administrador. Cine (13/23).

Una vez editados se pulsa en el botón azul “Editar”. Un mensaje de confirmación nos indica que los datos han sido validados.

## PalomitasTime

The screenshot shows the PalomitasTime administrator dashboard. On the left, there's a sidebar with categories: ADMINISTRADORES (Perfil, Gestionar administradores), CINE (Películas, Slider, Salas, Sesiones), and TIENDA (Categorías, Productos, Menús). The main area is titled "Editando producto RED BULL". It shows a success message "Producto editado correctamente." and fields for Nombre del producto (RED BULL), Precio del producto (5), and Categoría a la que pertenece (Bebidas). An "Editar" button is visible. At the top right, there's a "Cerrar" (Close) button.

Figura 68: Manual de administrador. Cine (14/23).

De nuevo en “Ver todos”, para eliminar un producto se pulsa en el ícono de la papelera de la columna “Borrar”. Hay que aceptar un mensaje de confirmación para eliminar definitivamente el producto. Una vez aceptado, se ve como el producto desaparece de la lista.

The screenshot shows a confirmation dialog box asking "¿Seguro que quieres eliminar este producto?". Below it, a table lists three products: Aquarius (1.00), Coca-Cola (1.00), and Fanta (1.00). Each row has edit and delete icons. The delete icon for Aquarius is highlighted. The table has columns for URL, Name, Price, Edit, and Delete.

		Precio	Editar	Borrar
/uploads/productos/aquarius.png	Aquarius	1.00		
/uploads/productos/coca-cola.png	Coca-Cola	1.00		
/uploads/productos/fanta.png	Fanta	1.00		

Figura 69: Manual de administrador. Cine (15/23).

### 3.3 Menús

Aquí se pueden crear, ver y editar los menús existentes. Estos incluyen varios productos en packs y suelen incluir algún tipo de descuento por comprar los productos agrupados.

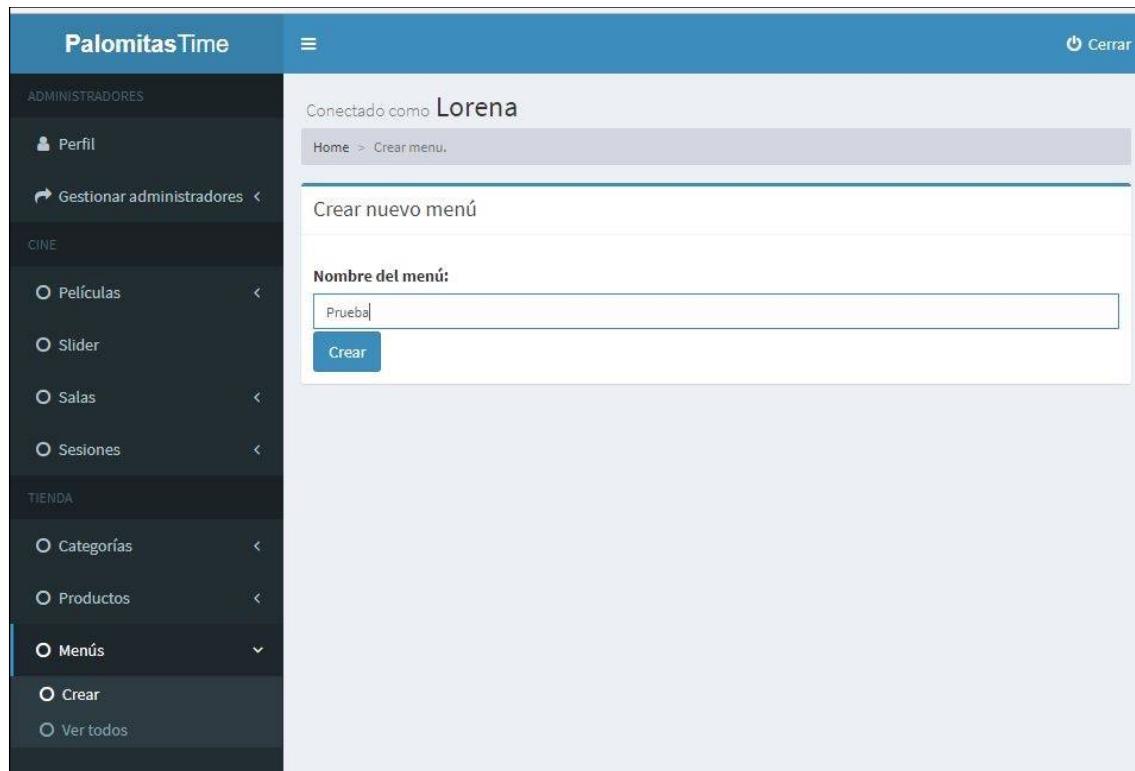


Figura 70: Manual de administrador. Cine (16/23).

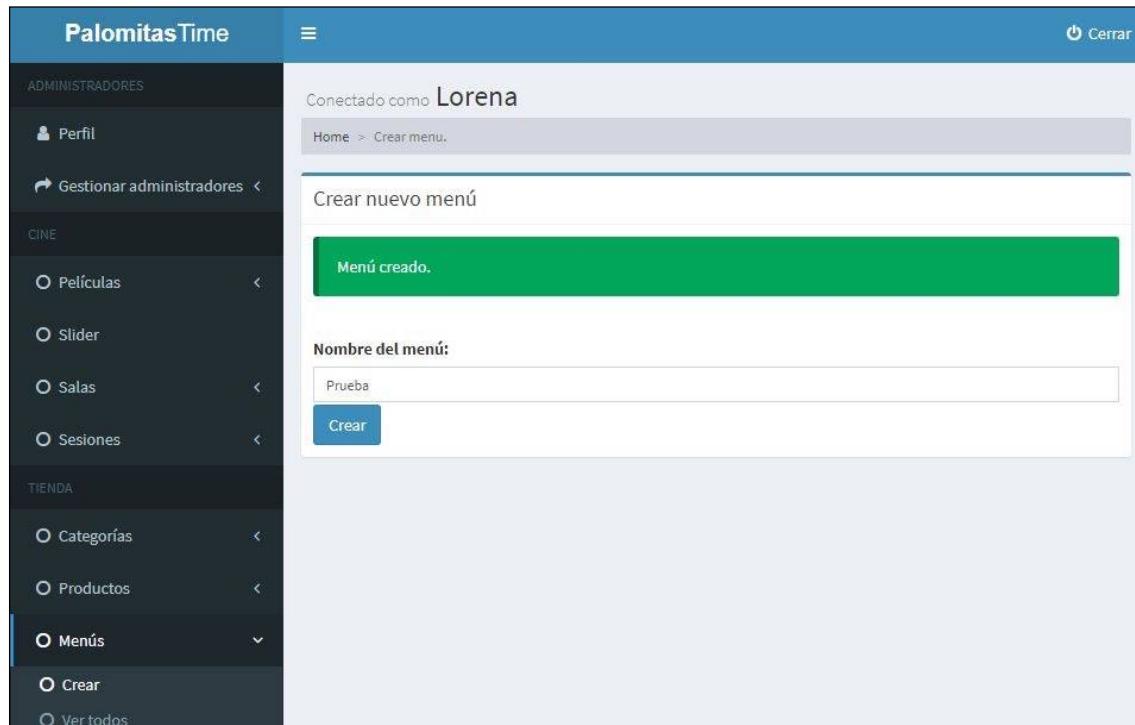


Figura 71: Manual de administrador. Cine (17/23).

## PalomitasTime

The screenshot shows the administrator interface for PalomitasTime. On the left, there is a sidebar with the following navigation:

- ADMINISTRADORES**
  - Perfil
  - Gestionar administradores
- CINE**
  - Películas
  - Slider
  - Salas
  - Sesiones
- TIENDA**
  - Categorías
  - Productos
  - Menús
  - Crear
  - Ver todos

The main content area is titled "Menús existentes" (Existing Menus). It displays a table with the following data:

Nombre	Añadir Productos	Borrar Productos	Editar	Borrar
Menú 1				
Menú 2				
Menú 3				
Menú 4				
Prueba				

Figura 72: Manual de administrador. Cine (18/23).

The screenshot shows the administrator interface for PalomitasTime. On the left, there is a sidebar with the following navigation:

- ADMINISTRADORES**
  - Perfil
  - Gestionar administradores
- CINE**
  - Películas
  - Slider
  - Salas
  - Sesiones
- TIENDA**
  - Categorías
  - Productos
  - Menús
  - Crear
  - Ver todos

The main content area is titled "Añadir nuevos productos" (Add new products). It displays a green success message: "Productos añadidos." Below this, there is a list of products:

- Bollicaos Mini
- Donuts Choco
- Kinder Bueno
- Kinder Maxi
- Kit Kat
- Mikado
- Oreo Mini

At the bottom, there is a blue "Añadir" (Add) button.

Figura 73: Manual de administrador. Cine (19/23).

## PalomitasTime

The screenshot shows the PalomitasTime administrator interface. The left sidebar has a dark theme with white text. It includes sections for 'ADMINISTRADORES' (Perfil, Gestión de administradores), 'CINE' (Películas, Slider, Salas, Sesiones), and 'TIENDA' (Categorías, Productos, Menús). The main content area has a light blue header bar with the title 'PalomitasTime' and a 'Cerrar' (Close) button. Below the header, it says 'Conectado como Lorena' and shows the breadcrumb path 'Home > Menus > Borrar productos.' The main content is titled 'Borrar productos' and contains a section for 'Productos:' with a list: Aquarius, Coca-Cola, Fanta. At the bottom is a blue 'Borrar' (Delete) button.

Figura 74: Manual de administrador. Cine (20/23).

This screenshot is from the same interface as Figure 74, but it shows the result of a deletion. The main content area now features a green banner at the top stating 'Productos borrados.' (Products deleted.) The list of products below is identical to Figure 74: Aquarius, Coca-Cola, and Fanta. The 'Borrar' button is still present at the bottom.

Figura 75: Manual de administrador. Cine (21/23).

## PalomitasTime

The screenshot shows the PalomitasTime administrator interface. On the left, there is a sidebar with a dark background containing navigation links: 'ADMINISTRADORES' (Perfil, Gestionar administradores), 'CINE' (Películas, Slider, Salas, Sesiones), and 'TIENDA' (Categorías, Productos, Menús). The main content area is titled 'Borrar productos'. It displays a list of products: 'Aquarius' and 'Coca-Cola'. Below the list is a blue 'Borrar' (Delete) button. At the top of the main area, it says 'Conectado como Lorena' and shows a breadcrumb trail: Home > Menus > Borrar productos.

Figura 76: Manual de administrador. Cine (22/23).

The screenshot shows the PalomitasTime administrator interface. The main content area is titled 'Menús existentes'. It displays a table of existing menus:

Nombre	Añadir Productos	Borrar Productos	Editar	Borrar
Menú 1				
Menú 2				
Menú 3				
Menú 4				
Prueba				

A modal dialog box is overlaid on the screen, asking '¿Seguro que quieres eliminar este menú?' (Are you sure you want to delete this menu?). It has two buttons: 'Aceptar' (Accept) and 'Cancelar' (Cancel). The background shows a breadcrumb trail: Home > Lista de menús.

Figura 77: Manual de administrador. Cine (23/23).

## 8.5. Implantación propiamente dicha

Para desplegar la página web se han necesitado 4 servicios:

1. **Laravel Forge**: Es una herramienta de aprovisionamiento de aplicaciones web que permite crear una página web en “cuestión de segundos”.
2. **DigitalOcean**: Es un proveedor estadounidense de servidores virtuales privados.
3. **GoDaddy**: Es una empresa registradora de dominios de Internet y de alojamiento web.
4. **Let's encrypt**: Es un gestor gratuito y automático para obtener certificados SSL.

El registro es muy sencillo en todos, voy a explicar de forma breve como se realizó el despliegue web.

The screenshot shows the Laravel Forge control panel. At the top, there are two tabs: 'Resources' (which is selected) and 'Activity'. Below this, there are two main sections: 'DROPLETS (1)' and 'DOMAINS (1)'. The 'DROPLETS (1)' section contains one item: 'MainServer' (status: green dot), IP: 139.59.184.158, with a 'Copy' button and 'Add tags' link. There is also a small icon and three dots. The 'DOMAINS (1)' section contains one item: 'palomitastime.online' (status: green dot), with '2 A / 3 NS / 1 SOA' information and three dots.

Figura 78: Implantación.

Para empezar hace falta un servidor privado que aloje el sitio web, escogimos **DigitalOcean** donde se creó un Ubuntu 18.04 x64 1GB de RAM y 25GB SSD. Y conectado al dominio **palomitastime.online**, dominio proporcionado por GoDaddy.

Para automatizar el proceso del despliegue usamos LaravelForge, para que LaravelForge cree automáticamente permisos para acceder a DigitalOcean, en la pestaña Server Providers hay que introducir el Api Token que es proporcionada por DigitalOcean al alquilar un servidor.

## PalomitasTime

The screenshot shows the "New Provider" section of the DigitalOcean API settings. It includes a provider selection dropdown with options: DigitalOcean (2.0) (selected), Linode, Amazon, and Vultr. A note below says: "You can create a new DigitalOcean API access token for yourself or your team from the [DigitalOcean API settings panel](#)". Below this are fields for "Profile Name" (Personal) and "API Token", with a "Add Credential" button. The "Active Providers" section lists "Personal" under "Name" and "DigitalOcean (2.0)" under "Provider". There are edit and delete icons next to the entry.

Figura 79: Implantación.

The screenshot shows the "Site Details" page for "PALOMITASTIME.ONLINE". It displays site information: MAINSERVER, PALOMITASTIME.ONLINE, 139.59.184.158, (10.131.35.167), ACTIVE, and a plus icon. On the left is a sidebar with links: Apps, Notifications, Environment, Queue, SSL, Redirects, and Meta. The main area has a "Deployment" section with a "Deploy Now" button and a note about quick deploy. It also has a "Deploy Script" section containing a script:

```
1 cd /home/forge/palomitastime.online
2 git pull origin master
3 composer install --no-interaction --prefer-dist --optimize-autoloader
4 echo "" | sudo -S service php7.2-fpm reload
5
6 if [ -f artisan ]
7 then
8     php artisan migrate --force
9 fi
```

With a "Save Script" button at the bottom.

Figura 80: Implantación.

En la parte de abajo se puede ver que actualmente tengo activo un Proveedor con el nombre Personal y cada Proveedor está conectado a un site.

En “Deploy Script” están declarados los comandos que se utilizan cada vez que el botón “Deploy Now” es pulsado. El flujo es el siguiente:

1. LaravelForge clona la rama master del repositorio de Github.

## PalomitasTime

2. La configuración del proyecto hay que hacerla en la pestaña “Envinroment” y es un archivo “.env”, que es igual que el archivo “.env” local.

3. Para ejecutar los seeders y llenar con datos la base de datos hay que conectarse mediante ssh al servidor. Para eso tenemos que proporcionar una clave ssh valida al servidor. Usaremos la clave ssh de nuestro ordenador, para generarla abrimos la consola de comandos y escribimos “ssh-keygen” y seguimos los pasos. Copiamos la clave generada y en la pestaña Servers/SSH Keys añadimos nuestra clave con el nombre que queremos.

4. Yo ya he añadido mi clave llamada Macbook como se puede ver en la siguiente imagen:

5. Ya solo queda conectarnos a la base de datos y poder hacer cambios, abrimos la consola y escribimos ssh forge@ip\_servidor como se puede ver en la siguiente imagen:

Y está ha sido toda la configuración y la prueba de que esta desplegado online:

<https://palomitastime.online/>

## 9.DEFINICIÓN DE PROCEDIMIENTOS DE CONTROL Y EVALUACIÓN

### Integración continua con Travis y Codacy:

Para que nuestro código cumpla la calidad que esperábamos decidimos integrarlo en Travis que permite conectar el repositorio de Github y pasar automáticamente las pruebas después de cada push que se haga, y si las pruebas fallan se envía una notificación via email.

Pero la integración con Travis no era suficiente, necesitábamos un análisis automatizado de código también y bajo la premisa "Review less, merge faster" los creadores de Codacy proporcionan una plataforma capaz de integrarse con múltiples repositorios y servicios como GitHub, Bitbucket, Slack o Jira, de forma que no solo nos permite ver la calidad de nuestros proyectos sino gestionar desde un único lugar las tareas a realizar para mejorar o solucionar incidencias detectadas por la herramienta.

**Avisar de que si se va a nuestro repositorio en Github se puede ver mediante etiquetas la la cobertura de Codacy, la calidad del proyecto proporcionada por Codacy y si la compilación pasa proporcionada por Travis. Si se hace click en alguna etiqueta se va directamente a la página y se puede ver sin necesidad de tener cuenta.**

A continuación explicaré la configuración de Travis y Codacy:

Para usar Travis y Codacy debemos registrarnos en su página web mediante la cuenta de github, ambos son gratuitos para proyectos open-source.

En ambos tras registrarse en la siguiente pantalla podremos elegir qué repositorio queremos activar, nosotros activaremos el nuestro. Una vez activado el proyecto, al realizar el siguiente push se lanzará la compilación de Travis y nos enviará un email con el resultado de la compilación y lo mismo con Codacy.

Pero la compilación de Travis fallara ya que no encontrara el archivo ".travis.yml" en la raíz del repositorio que contiene la configuración del lenguaje usado y los scripts para compilar con éxito. El nuestro es el siguiente:

## PalomitasTime

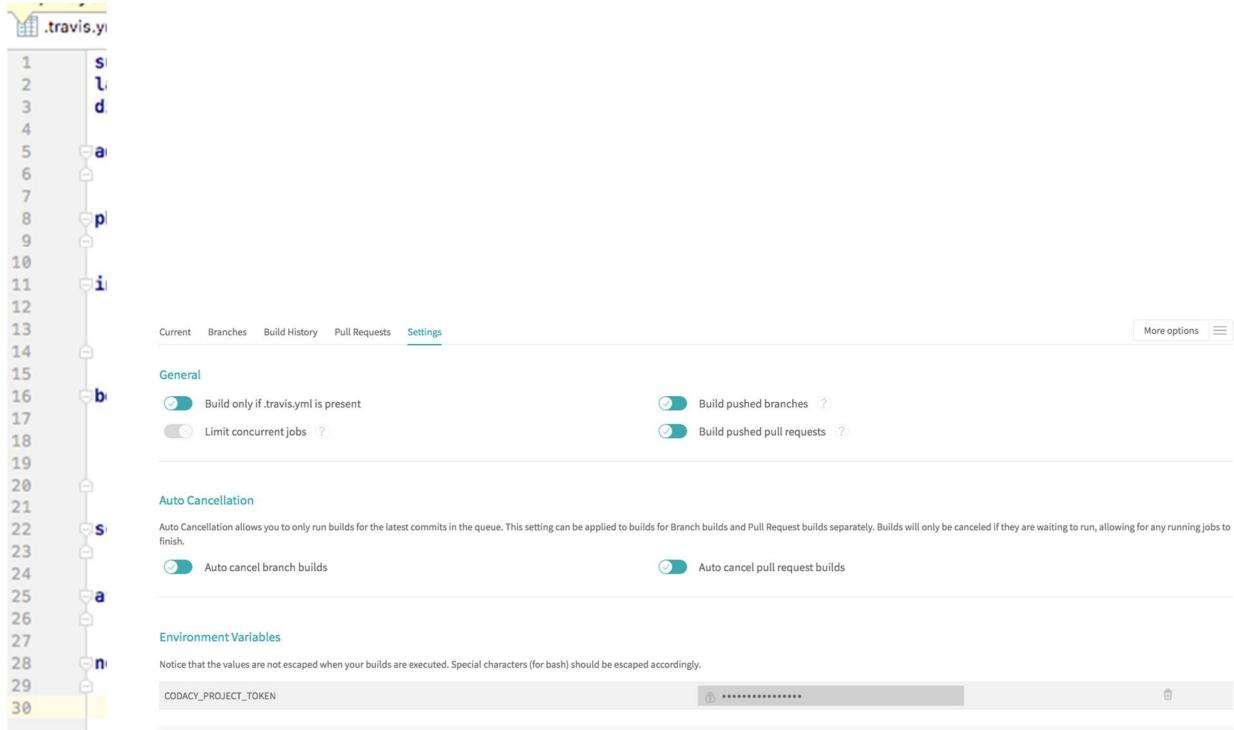


Figura 81: Control de evaluación.

Pero desgraciadamente Codacy y Travis son independientes y para que sepan el uno del otro y se puedan comunicar debemos seguir una serie de pasos:

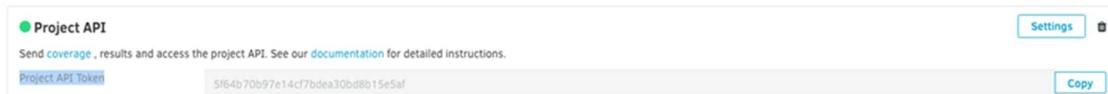


Figura 82: Control de evaluación.

1. Obtener el Api Token de Codacy. Iremos a la página de Codacy, haremos click en la pestaña settings, después en la pestaña integrations y por ultimo copiaremos el token:
2. Para conectar Travis con codacy vamos a la página web de este y en la pestaña settings en el apartado de Environment Variables introducimos una nueva variable llamada "CODACY\_PROJECT\_TOKEN" y pegamos el token copiado antes. Esta es nuestra configuración en Travis:

Y ya está, a partir de ahora con cada push Travis compilara el proyecto y enviara la cobertura a Codacy donde podemos ver el porcentaje global y por fichero.

Ejemplos:

## PalomitasTime



Figura 83: Control de evaluación.

GRADE	FILENAME	ISSUES	DUPLICATION	COMPLEXITY	COVERAGE
A	app/Models/Sala.php	0	0	1	100%
A	app/Http/Controllers/SalaController.php	0	0	6	100%
A	app/Http/Controllers/Spacontroller.php	0	0	1	100%
A	app/Http/Controllers/ButacaController.php	0	0	5	100%
A	app/Events/ResenaEvent.php	0	0	1	100%
A	app/Http/Controllers/ResenaController.php	1	1	6	99%
A	app/Http/Controllers/Admin/AdministradoresController.php	2	0	12	94%
A	app/Http/Controllers/APIAuthController.php	0	0	4	91%
A	app/Mail/VerificarEmail.php	0	0	1	80%
A	app/Exceptions/Handler.php	0	0	3	77%
A	app/Models/User.php	0	0	1	75%
B	app/Http/Controllers/PeliculaController.php	1	1	6	74%
A	app/Http/Controllers/PlantillaSesionController.php	0	0	5	71%
B	app/Models/Resena.php	0	0	1	50%
B	app/Http/Controllers/SesionVaciaController.php	0	0	4	29%
B	app/Http/Controllers/Admin/Home.php	0	0	1	29%

Figura 84: Control de evaluación.

## PalomitasTime

```
66      */
67  public function verifyUser(Request $request)
68  {
3x  69      $check = DB::table('user_verifications')->where('token', $request->verification)->first();
3x  70      if (!is_null($check)) {
2x  71          $user = User::find($check->user_id);
2x  72          if ($user->is_verified == 1) {
1x  73              return response()->json([
1x  74                  'success' => true,
1x  75                  'message' => 'Cuenta ya verificada...'
1x  76              ],200);
1x  77      }
1x  78      $user->update(['is_verified' => 1]);
1x  79      DB::table('user_verifications')->where('token', $request->verification)->delete();
1x  80      return response()->json([
1x  81          'success' => true,
1x  82          'message' => 'Has verificado correctamente tu cuenta.'
1x  83      ],200);
1x  84  }
1x  85  return response()->json(['success' => false, 'message' => "El código de verificación no es válido."],400);
1x  86  }
1x  87 }
```

Figura 85: Control de evaluación.

## Control de Versiones:

**Github** ha sido la herramienta de control de versiones que se ha utilizado. El workflow que hemos seguido es que **master ha sido la rama de producción** y solo hacíamos merge cuando comprobábamos que todo funciona como queríamos, **desarrollo es la rama principal del proyecto** donde hemos picado más código y a partir de desarrollo se han ido creado **nuevas ramas para cada funcionalidad** grande que queríamos implementar.

Algunas ramas, teníamos más pero las hemos ido borrando:

## PalomitasTime

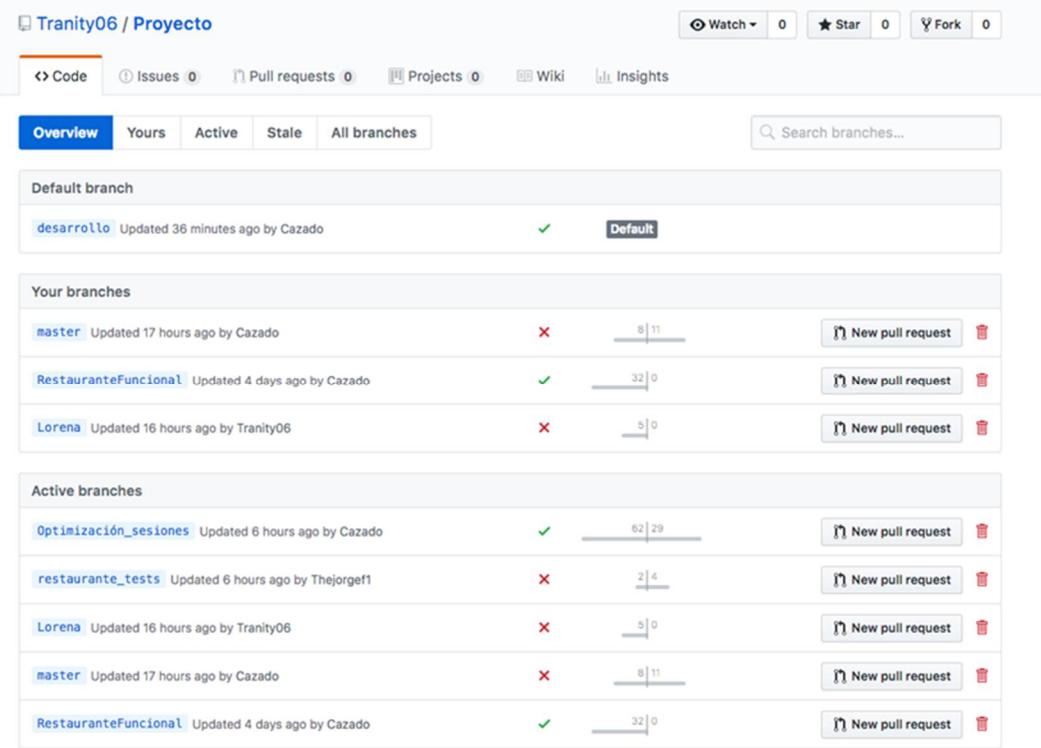


Figura 86: Control de versiones.

Algunas bifurcaciones:

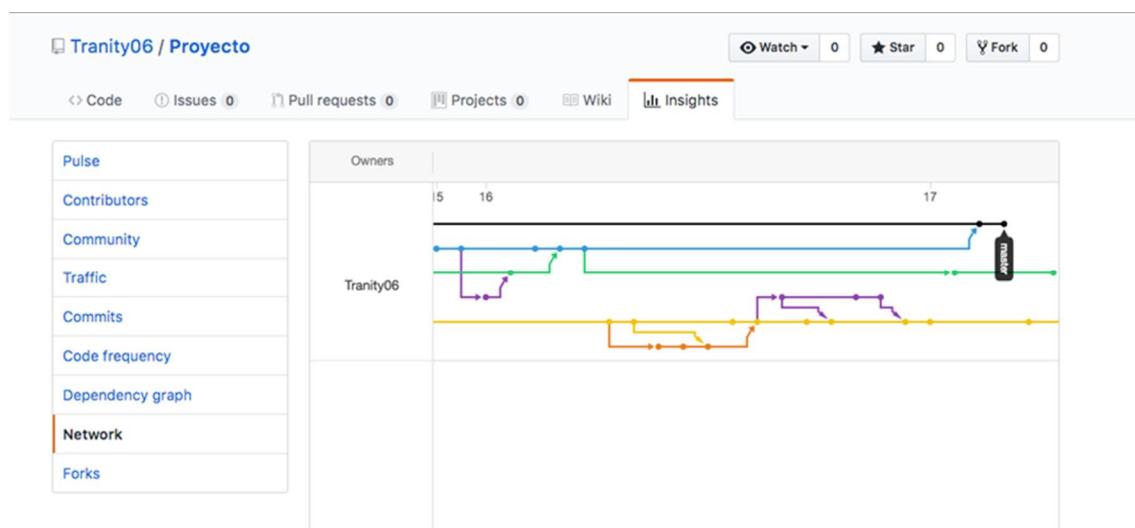


Figura 87: Control de versiones.

## PalomitasTime

### Ejemplos de Merge:

Commits on Jun 16, 2018
FIX: pequeños errores Tranity06 committed a day ago ✓
Merge branch 'desarrollo' of https://github.com/Tranity06/Proyecto in... Cazado committed a day ago ✓
Fix: peliculas proximo Cazado committed a day ago
Merge branch 'desarrollo' into Lorena Tranity06 committed a day ago ✓
arreglado crear y borrar productos Thejorgef1 committed a day ago ✓

Figura 89: Control de versiones.

## **10. CONCLUSIONES**

Nos ha faltado tiempo para poder completar todo lo que nos habría gustado del proyecto. Con las prácticas, el tiempo que se puede dedicar al final para el proyecto no es ni la mitad de lo estimado.

A pesar de esto estamos satisfechos con el resultado de nuestro trabajo. Hemos aprovechado para aprender nuevas tecnologías que no hemos tenido ocasión de estudiar en clase. Esto es una de las principales causas de que no esté terminado, pues para cada paso teníamos algo nuevo que aprender antes de empezar a trabajar. Por otro lado, la continua sensación de aprendizaje durante la ejecución de este proyecto es lo que ha hecho que nos resulte tan interesante y divertido de realizar.

Esperamos esté a la altura de las expectativas ya que lo presentamos con mucha ilusión.

PalomitasTime  
**11.. FUENTES**

***1.Legislación***

ASIR

Enseñanzas mínimas: Real Decreto 1629/2009 (B.O.E. 18/11/2009)

[http://www.madrid.org/fp/ense\\_fp/catalogo\\_LOE/pdf/IFCS01/titulo/RD20091629\\_TS\\_Admon\\_Sistemas\\_Informaticos\\_en\\_Red.pdf](http://www.madrid.org/fp/ense_fp/catalogo_LOE/pdf/IFCS01/titulo/RD20091629_TS_Admon_Sistemas_Informaticos_en_Red.pdf)

Curriculum: [Decreto 12/2010 \(B.O.C.M. 15/04/2010\)](#)

[http://www.madrid.org/fp/ense\\_fp/catalogo\\_LOE/pdf/IFCS01/curriculo/D20100012\\_Administracion\\_SistemasInformaticos.pdf](http://www.madrid.org/fp/ense_fp/catalogo_LOE/pdf/IFCS01/curriculo/D20100012_Administracion_SistemasInformaticos.pdf)

DAM

Enseñanzas mínimas: Real Decreto 450/2010, de 16 de abril (BOE 20/05/2010)

[http://pdf/IFCS02/titulo/RD20100450\\_TS\\_Desarrollo\\_Aplicaciones\\_Multiplataforma.pdf](http://pdf/IFCS02/titulo/RD20100450_TS_Desarrollo_Aplicaciones_Multiplataforma.pdf)

Curriculum: [D. 3/2011, de 13 de enero \(BOCM 31/01/2011\)](#)

[http://pdf/IFCS02/curriculo/D20110003\\_TS\\_Desarrollo\\_Aplicaciones\\_Multiplataforma.pdf](http://pdf/IFCS02/curriculo/D20110003_TS_Desarrollo_Aplicaciones_Multiplataforma.pdf)

DAW

Enseñanzas mínimas: Real Decreto 686/2010, de 20 de mayo (BOE 12/06/2010)

[http://pdf/IFCS03/titulo/RD20100686\\_TS\\_Desarrollo\\_Aplicaciones\\_Web.pdf](http://pdf/IFCS03/titulo/RD20100686_TS_Desarrollo_Aplicaciones_Web.pdf)

Curriculum: [Decreto 1/2011, de 13 de enero \(BOCM 31/01/2011\)](#)

[http://pdf/IFCS03/curriculo/D20110001\\_TS\\_Desarrollo\\_Aplicaciones\\_Web.pdf](http://pdf/IFCS03/curriculo/D20110001_TS_Desarrollo_Aplicaciones_Web.pdf)

Definición de procedimientos de control y evaluación:

- <http://www.xperta.es/es/descripcion.asp>
- <http://www.xperta.es/es/aquienvadirigido.asp>
- <http://churriwifi.wordpress.com/2010/04/10/gestion-de-incidencias/>
- [http://es.wikipedia.org/wiki/Control\\_de\\_versiones](http://es.wikipedia.org/wiki/Control_de_versiones)
-

## 11.1. **Bibliografía**

1. <https://styde.net/installacion-y-uso-de-vue-router-en-vue-js-2/>
2. <https://ssr.vuejs.org/#what-is-server-side-rendering-ssr>
3. <https://laravel.com/docs/5.6/broadcasting>
4. <https://vuex.vuejs.org/>
5. <https://yarnpkg.com/lang/en/docs/install/#windows-stable>
6. <https://es-vuejs.github.io/vuejs.org/v2/guide/class-and-style.html>
7. <https://aws.amazon.com/es/devops/continuous-integration/>
8. <http://enmilocalfunciona.io/analisis-de-codigo-con-codacy/>
9. <https://www.emezeta.com/articulos/digitalocean-guia-para-alojar-tu-web-en-un-servidor-vps>
10. <https://laravel.com/>
11. <https://laravel-news.com>
12. <https://pusher.com/tutorials/web-notifications-laravel-pusher-channels>
13. <https://www.algolia.com/>
14. <https://www.algolia.com/doc/api-client/laravel/algolia-and-scout/>
15. <https://community.algolia.com/vue-instantsearch/getting-started/getting-started.html>
16. <https://code.tutsplus.com/es/tutorials/deploy-php-web-application-using-laravel-forge--cms-30329>
17. <https://semaphoreci.com/community/tutorials/getting-started-with-phpunit-in-laravel>
18. <https://bulma.io/>
19. <https://css-tricks.com/>
20. <https://stackoverflow.com/>
21. <https://apiumhub.com/es/tech-blog-barcelona/integracion-continua/>
22. <https://www.adictosaltrabajo.com/tutoriales/integracion-de-un-proyecto-de-github-con-travis-y-codacy/>
23. <https://es-vuejs.github.io/vuejs.org/>

PalomitasTime

- 24.<https://medium.com/@mosesesan/tutorial-5-how-to-build-a-laravel-5-4-jwt-authentication-api-with-e-mail-verification-61d3f356f823>
- 25.<https://codecourse.com/subjects/laravel>
- 26.<https://github.com/axios/axios>
- 27.<https://styde.net/laravel-5/>
- 28.<https://styde.net/porque-laravel-no-es-mvc-y-tu-deberias-olvidarte-de-mvc/>
- 29.<https://www.arsys.es/blog/programacion/diseno-web/spa-unico-pagina/>
- 30.<https://medium.com/@NeotericEU/single-page-application-vs-multiple-page-application-2591588efe58>
- 31.<https://jquery.com/>
- 32.<https://www.google.es/chrome/index.html>
- 33.<https://www.mozilla.org/es-ES/firefox/new/>
- 34.<https://docs.microsoft.com/es-es/microsoft-edge/deploy/enterprise-guidance-using-microsoft-edge-and-ie11>
- 35.<https://caniuse.com/>