IW 2019-2010

[Dirección de correo electrónico]



Descripción breve

[Dibujar su lector con un resumen de la participación. Normalmente es un breve resumen del documento.   
Cuando esté listo para agregar contenido, haga clic aquí y empiece a escribir.]

Tienda deportiva

**ÍNDICE**

[**1.** **Introducción** 2](#_Toc30119128)

[**Tecnologías usadas** 2](#_Toc30119129)

[**Arquitectura** 3](#_Toc30119130)

[**Servicios** 3](#_Toc30119131)

[**Servidores** 3](#_Toc30119132)

[**2.** **Wireframes y mockups** 4](#_Toc30119133)

[**3.** **Patrones de diseño utilizados** 8](#_Toc30119134)

[**Patrones de identificación de usuarios** 8](#_Toc30119135)

[**Patrones de página principal** 8](#_Toc30119136)

[**Patrones de formulario** 10](#_Toc30119137)

[**Patrones de navegación** 12](#_Toc30119138)

[**Patrones de listados** 12](#_Toc30119139)

[**Patrones de búsqueda en el sitio** 13](#_Toc30119140)

[**4.** **Diagramas** 14](#_Toc30119141)

[**Diagrama de casos de uso** 14](#_Toc30119142)

[**Diagrama de clases** 15](#_Toc30119143)

[**5.** **Metodologías y planificación** 15](#_Toc30119144)

[**6.** **Descripción de la implementación** 16](#_Toc30119145)

[**Vue**: 16](#_Toc30119146)

[**Axios** 18](#_Toc30119147)

[**Vue-Router:** 18](#_Toc30119148)

[**Vuex** 19](#_Toc30119149)

[**Master Page y Componentes:** 19](#_Toc30119150)

[**Seeders** 19](#_Toc30119151)

[**Migraciones** 20](#_Toc30119152)

[**Controllers** 20](#_Toc30119153)

[**Routing/API Rest:** 21](#_Toc30119154)

[**Rutas con protección de autorización** 21](#_Toc30119155)

[**Resources** 22](#_Toc30119156)

[**Test** 22](#_Toc30119157)

[**Modelos** 23](#_Toc30119158)

[**7.** **Problemas y solución** 23](#_Toc30119159)

[**8.** **Mejoras y ampliaciones** 24](#_Toc30119160)

[**9.** **Referencias** 24](#_Toc30119161)

# **Introducción**

## **Tecnologías usadas**

**Laravel**: Laravel es un framework PHP de código abierto que intenta aprovechar las ventajas de otros Frameworks y que posee como puntos fuertes una extensa documentación, integración y escalabilidad, está basado en un modelo MVC (Modelo-Vista-Controlador) y está en constante mantenimiento y expansión por parte de sus desarrolladores lo que asegura la continuidad y seguridad del framework con actualizaciones regulares.

Laravel nos aporta un sistema de plantillas para la creación de vistas llamado Blade, sin embargo, hemos decidido hacer uso del sistema de vistas que nos ofrece Vue para la creación de las mismas por su gran cantidad de componentes. Hemos hecho uso de muchas otras características de laravel como:

* **Eloquent**: que es el ORM que incluye Laravel para manejar de una forma fácil y sencilla los procesos correspondientes al manejo de bases de datos en nuestro proyecto.
* **Routing**: sistema de organización y gestión de rutas de laravel.

**Vuejs**: es un framework progresivo para elaborar interfaces de usuario utilizando HTML y JavaScript, es muy simple de utilizar e integrar con otros proyectos o bibliotecas existentes.

Trabaja con componentes que son elementos que pueden ser usados para la reutilizacion de codigo. Algunas de sus características más importantes son su renderizado declarativo y su reactividad.

**Vuetifyjs**: es un framework que combina Vuejs con la estética de Material Design. Permite acelerar el desarrollo de aplicaciones web, incorporando una gran cantidad de componentes listos para usar. Su uso es similar al de bootstrap

Dispone de una gran librería de componentes que incluyen todo tipo de elemento que además les aportan un amplio rango de configuración, como son botones, inputs, sliders, etc.

**Vuex**: Vuex es una librería para la gestión del estado de aplicaciones Vue. Vuex simplifica la forma en la que nuestros componentes se comunican, centralizando el proceso. Por tanto, nos permite evitar tener que compartir demasiado estados comunes entre componentes, tener métodos o acciones muy parecidas en muchos componentes.

**Vue-Router:** Es una librería para manejar las rutas que hacen referencia a componentes.

**Faker**: Es una librería que permite generar datos aleatorios de prueba de una manera sencilla y extensible. Hemos hecho uso de la misma para poblar, en un inicio, los campos de nuestra base de datos con datos generados en los seeders.

## **Arquitectura**

**Arquitectura cliente-servidor:** Es una arquitectura consistente en dos partes, por una parte, el cliente, que realiza peticiones las cuales son respondidas por una segunda parte llamada servidor encargados de proveer los recursos o servicios.

**Patrón de diseño de arquitectura Modelo-Vista-Controlador:** Se basa en la separación de la parte lógica de la aplicación de la parte de las vistas, utilizando para ello tres componentes:

* **Modelo**: Encargado generalmente de las consultas a base de datos. Aunque en nuestro caso también usamos Eloquent que facilita esta tarea.
* **Vista**: Se encarga de la presentación visual de los datos.
* **Controlador**: Se encarga de la comunicación con el modelo para hacer consultas a la base de datos, y con las vistas para devolver una respuesta al cliente.

## **Servicios**

**API de servicios REST como backend:**

## **Servidores**

**Laragon apache server y Mysql:** Laragon es una suite de desarrollo (conjunto de programas para desarrollar aplicaciones) PHP como WAMP (Windows, Apache, Mysql/MariaDB, PHP) orientado a laravel. Está servido por apache server y trabaja con Mysql/MariaDB, aunque también da soporte a Postgresql y MongoDB. Sus principales ventajas son:

* Su facilidad de uso e instalación.
* Su portabilidad.

# **Wireframes y mockups**

Antes de comenzar el desarrollo de nuestra aplicación, el grupo ha realizado una serie de mockups los cuales han ayudado en la posterior creación de las vistas finales realizadas en la aplicación, para algunos de ellos se han usado patrones de diseño en su creación:

**Login**



**Detalles**



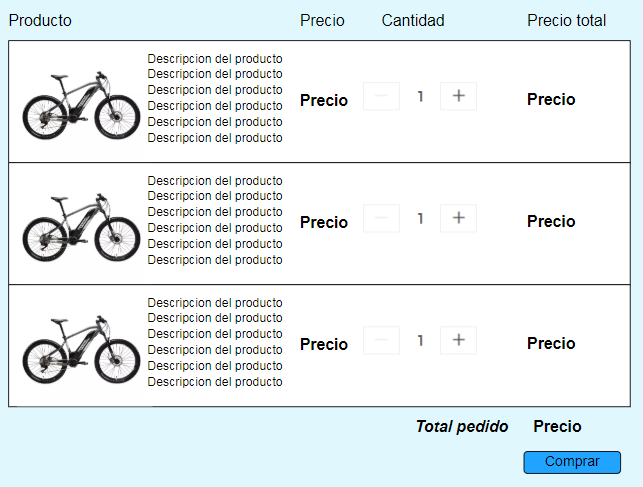
**Registro**



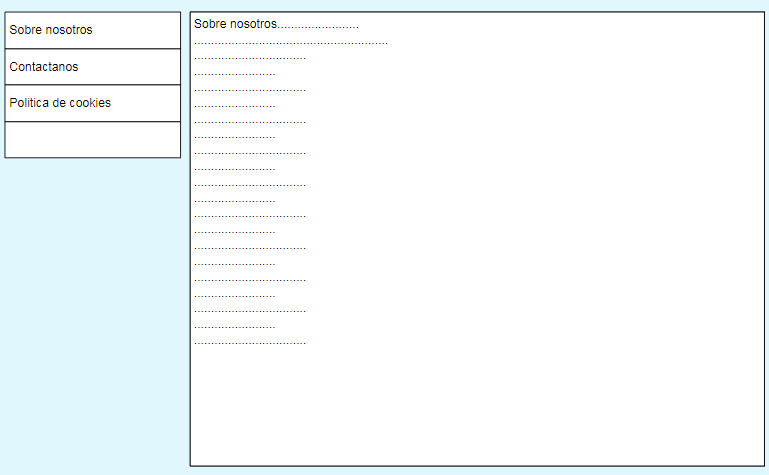
**Home**



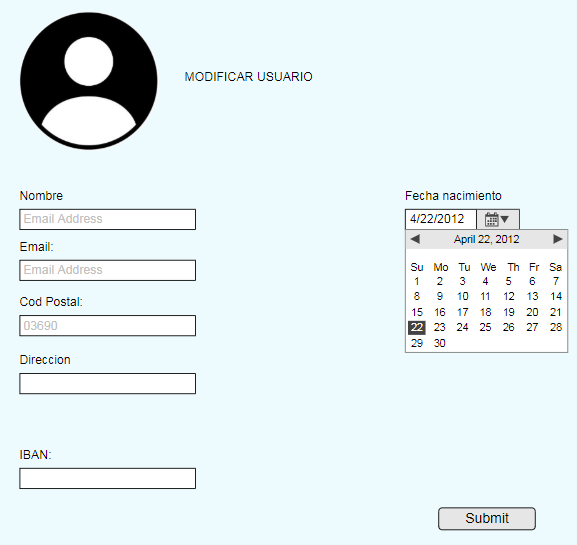
**Cesta**



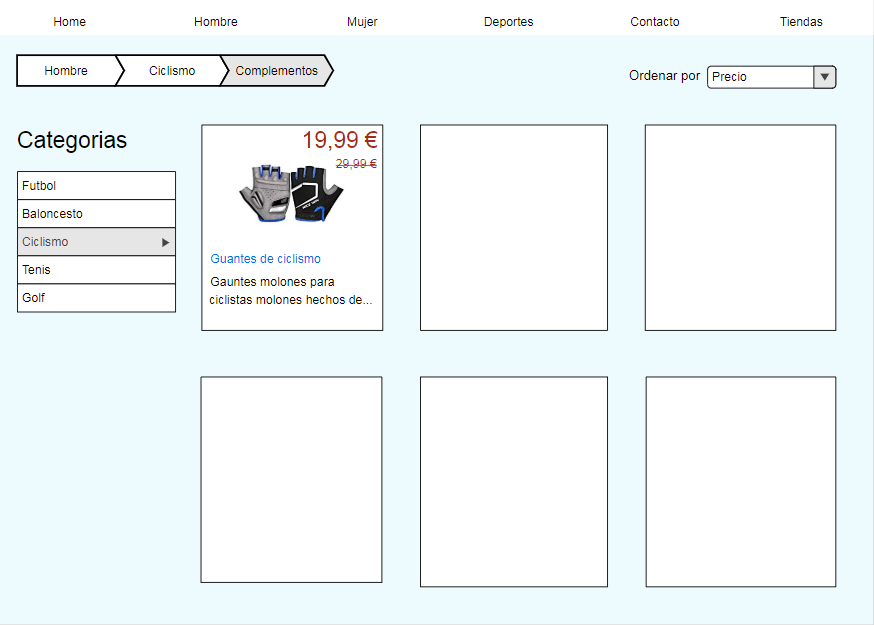
**Contacto**



**CRUD Usuario**



**Búsqueda**



# **Patrones de diseño utilizados**

## **Patrones de identificación de usuarios**

**Registro de usuarios:** Se ha intentado que la página sea lo más intuitiva para favorecer una rápida navegación por la misma.

Los usuarios pueden navegar y hacer uso de la página casi en su totalidad, solo siendo necesario para la realización de una compra y proceso de pedido y para realizar comentarios, evitando de esta manera incomodar al usuario y el llamado **lazy registration**.

Por otra parte, durante el registro, se ha intentado simplificar el formulario añadiendo solo los campos esenciales y siendo solo requisito algunos de ellos, se realizan las correspondientes validaciones de campos (forma y contenido), así como la confirmación del campo contraseña, también debemos señalar que se hace uso de una casilla para poder marcar y aceptar rápidamente las políticas de uso.

Además de lo anterior se ha añadido un aviso para la aceptación de la política de cookies.

**Log in:** Consistente en un campo nombre y un campo contraseña, haciendo uso de una casilla para que las credenciales del usuario queden recordadas y sea más cómodo su posterior log in y visitas a la página. Un log in sin éxito da como resultado un mensaje de credenciales inválidas, en un tono rojo para entendimiento de su negatividad.

**Log out:** El cierre de sesión se realiza mediante un botón señalizado en un tono rojo en el navegador, tiene un icono con el cual queda claro que su propósito es el de cerrar la sesión del usuario. De esta forma se evitan problemas de seguridad como el acceso de terceros a tu cuenta.

## **Patrones de página principal**

**Panel del control:** El usuario dispone de una lista de productos deseados por el mismo, donde puede guardar cualquier artículo para poder realizar más cómodamente su compra posteriormente en la cesta, que se encuentra en un icono separado para que su acceso, que normalmente es más frecuente, se realice de manera más fluida. También tiene acceso en este mismo menú a sus pedidos realizados y ver su estado, informando al cliente.

Otra de las opciones es la cesta donde el usuario puede ver sus pedidos por tramitar (realizar la compra) con sus artículos y su cantidad la cual puede ser alterada en esta misma sección para hacerlo más cómodo.

**Dashboard:** El usuario tiene un icono de perfil en la barra de navegación donde puede ver en todo momento los datos de su cuenta (datos personales, compras, pedidos realizados...). Puede realizar la edición de sus datos personales en la opción del mismo menú de editar perfil. La separación entre la vista con los datos de la cuenta de usuario y su edición se ha realizado para no saturar la vista de elementos, intentado aportar claridad al usuario.

La sección Home de la web ofrece una lista de artículos destacados o en “tendencia”, intentando informar de los productos más solicitados. Además, proporciona un carrusel con algunos artículos de nuestras secciones intentando guiar la navegación hasta los artículos y la compra al usuario.

**Configuración o 'Customización**'**:** La aplicación web realizada se trata de una tienda de deportes con la capacidad de poner a la venta muchísima variedad de artículos. Por este motivo se han implementado subcategorías para nuestras categorías. Esto quiere decir que el usuario a la hora de realizar una búsqueda, puede filtrar no solo por la categoría deportiva sino por una serie de subcategorías de la misma que a su vez también pueden ser filtradas por su subcategoría, aportando una precisión en la búsqueda de los artículos de la tienda muy buena. Antes de realizar el filtrado por categoría, hemos decidido filtrar en el menú los artículos por género, mostrando estas opciones en el menú como pestañas, nos pareció bien hacerlo así ya que el género es propio de todos los artículos. También hemos decidido que otro parámetro de filtrado para los artículos sea el precio, ya que suele ser un factor que el cliente suele tener muy en cuenta, para ello se dispone de una barra que puede ser desplazada hasta conseguir el precio querido. Otro campo de filtrado son las marcas de los productos y su valoración, este último nos parece bastante útil ya que así el usuario tiene más posibilidades de quedar más contento con el producto y se pueden evitar grandes volúmenes de quejas y devoluciones.

Otra configuración a tratar es la posibilidad de añadir varias direcciones de envío, lo cual puede resultar bastante útil si se dispone de varios domicilios. Este dato puede ser agregado posteriormente, en la compra, para no abrumar al cliente a la hora de registrarse.

Otras opciones de configuración son las de añadir o retirar artículos de la lista de deseados o la cesta, lugar donde se permite esta modificación para evitarle al usuario tener que volver y hacer otro pedido.

Para finalizar debemos destacar la existencia de ordenación por campos y de una barra de búsqueda para el listado de artículos, marcas y categorías del administrador.

**Tabla en blanco:** Las páginas que muestran contenido, como una cesta vacía, muestran un mensaje informando de porque se encuentran en ese estado, dejando claro, y nunca un espacio vacío, el contenido que debería mostrar y porque esto no ocurre. Informamos al usuario de esta manera de porque no se muestra el contenido que busca y le damos una indicación de cómo solucionarlo.

## **Patrones de formulario**

**Formularios breves:** Hemos intentado que los formularios utilizados contuvieran sólo la información imprescindible y hemos dejado, y marcado, solo algunos de sus campos como obligatorios, siendo posible completarlos más adelante. Además, hemos agregado validación a los campos, mostrando al futuro cliente que debe contener el mismo, si es requerido o si existe algún error, informar de cual es en un tono rojizo para señalar de qué se trata de un error. De esta forma pretendemos evitar que el cliente se vea abrumado o desista a la hora de registrarse en la web.

También hemos dividido algunos de los formularios por pestañas, para dar la sensación de que no son excesivos los datos que se solicitan, este es el caso de la creacion de articulos.

Se ha optado por el uso de autocompletados en algunos campos como la categoría o la marca, para que el administrador pueda filtrar o buscar en la lista desplegada el campo que desea con mucha facilidad.

**Alineado de las etiquetas de los campos:** En nuestro caso, la etiqueta está incrustada en el mismo campo y se eleva al intentar insertar algún texto en el mismo, de esta manera conseguimos que siempre quede claro el contenido del campo.

**Señalizar campos obligatorios:** Hemos señalado los campos obligatorios con un asterisco y además estos cambian a un color rojo cuando el cliente no inserta nada en ellos, intentando dar a entender que el campo es obligatorio y aportando un mensaje en el mismo campo que haga evidente que debe ser completado.

**Valores por defecto 'inteligentes':** Este podría ser el caso de los autocompletados realizados para los campos de categorías, género y marcas, donde nos aporta una lista de los posibles valores del campo y la posibilidad de búsqueda de algún valor específico, de esta forma evitamos la introducción de datos erróneos y en el caso de tener una gran cantidad de valores poder encontrar con facilidad el deseado.

Otro caso es el de campos como la fecha de nacimiento, donde facilitamos un calendario para que el usuario pueda de manera interactiva y clara seleccionar la fecha que desee.

**Formato de los datos:** Los campos que solicitamos en nuestros formularios tiene validación y para no desorientar al cliente o administrador hemos intentado negar la introducción de algunos valores, como letras en el campo de precio y, además, mostramos un mensaje en rojo bajo el campo señalando que existe un error en el valor que se intenta introducir y una “pista” para que este pueda solucionarlo.

**Pistas o sugerencias:** Como hemos señalado en formato de datos, intentamos no frustrar al usuario a la hora de rellenar un formulario, remarcando el campo en rojo y con una pista del mismo color bajo el mismo, informando al usuario de cómo tiene que solventar el error.

**Navegación con el teclado:** La ordenación de los campos permite una navegación entre los mismos con tab, sin embargo, no se ha realizado ninguna configuración de este tipo en la vista que configure esta acción. Si que hemos tenido en cuenta que no debemos enfocar un campo automáticamente al entrar al formulario para no ocultar el resto.

**Botones de comando:** Solo hemos incluido un único botón de submit para nuestros formularios en un color verde que deje claro que debe usarse, el botón solo está disponible al rellenar los campos obligatorios correctamente. No hacemos uso de botón de reset en los formularios excepto para el campo de las imágenes. Señalamos si se ha producido algún error o un éxito en algunos de los formularios, dando al usuario la seguridad de que no ha habido ningún problema.

**Mensajes de error:** Los datos erróneos en nuestros formularios no se borran y se puede saber con al instante si se ha introducido algún valor erróneo, por ejemplo, números en vez de letras o longitud del campo. El campo erróneo se marca en rojo y da una pista o mensaje del tipo de error que se está dando, intentando que sea fácil para el usuario localizarlo y solventarlo.

## **Patrones de navegación**

**Menú de primer nivel:** La barra de navegación cuenta con varios apartados de la web que hemos creído son los más interesantes para su uso por parte de los usuarios, facilitando de este modo su navegación. También cuenta con las opciones de login y una barra de navegación que busca por nombre de artículo. Además, una vez se inicia sesión aparecen en el menú algunos iconos que permiten la gestión de acciones por parte del usuario. Las secciones de la página se dividen en Home, para que el usuario vuelva a la página inicial, un filtro de artículos por género (mujer y hombre), un listado de artículos (artículos) y un contáctanos y una sección de tiendas disponibles. Justificamos el uso de esos campos en el menú al uso de algunos patrones en su realización y a que el campo común de todos los artículos es el género siendo este campo el mejor para filtrarlos.

**Menús de segundo nivel:** Usados para acceder a las opciones del menú de usuario, donde puede ver en un desplegable las acciones que puede realizar en la página, se ha decantado por esta opción para no saturar demasiado el menú.

**Herramientas de navegación:** Usado para el icono de login y logout en el menú, facilitando el inicio o cierre de sesión en cualquier momento.

**Navegación suplementaria:** Aquí podemos comentar el uso de la sección artículos recomendado o el carrusel que son usados para fomentar el interés por la página de los usuarios, ya sean secciones o artículos.

## **Patrones de listados**

**Lista simple:** Hacemos uso de las mismas para mostrar el listado de acciones que puede realizar el cliente, donde es mejor no saturar el menú principal con las mismas.

**Lista tabular o tabla y lista jerárquica:** Podemos considerar su uso a la hora de usar el filtrado de categorías, donde cada categoría se despliega en varias subcategorías y las mismas pueden desplegar sus propias subcategorías, de esta forma damos información al usuario de cómo ha llegado a esa categoría y en qué categoría principal se encuentra.

**Lista tabular:** Usado en las tablas de artículos, marcas y categorías y también en algunos campos donde se despliega una lista con opción de filtrado, caso de categorías o marca. Creemos que es una buena forma de organizarlo debido a la gran cantidad de filas que nos podemos encontrar.

**Acciones en un listado:** Las distintas tablas que usamos en la parte del administrador, tienen botones de acción de editar y eliminar que sólo actúan sobre un elemento. Los botones están debidamente marcados con un icono que representa la acción a llevar a cabo. De esta forma es muy fácil para el administrador la eliminación y edición de artículos gracias a las opciones de búsqueda y filtrado que poseen nuestras tablas.

## **Patrones de búsqueda en el sitio**

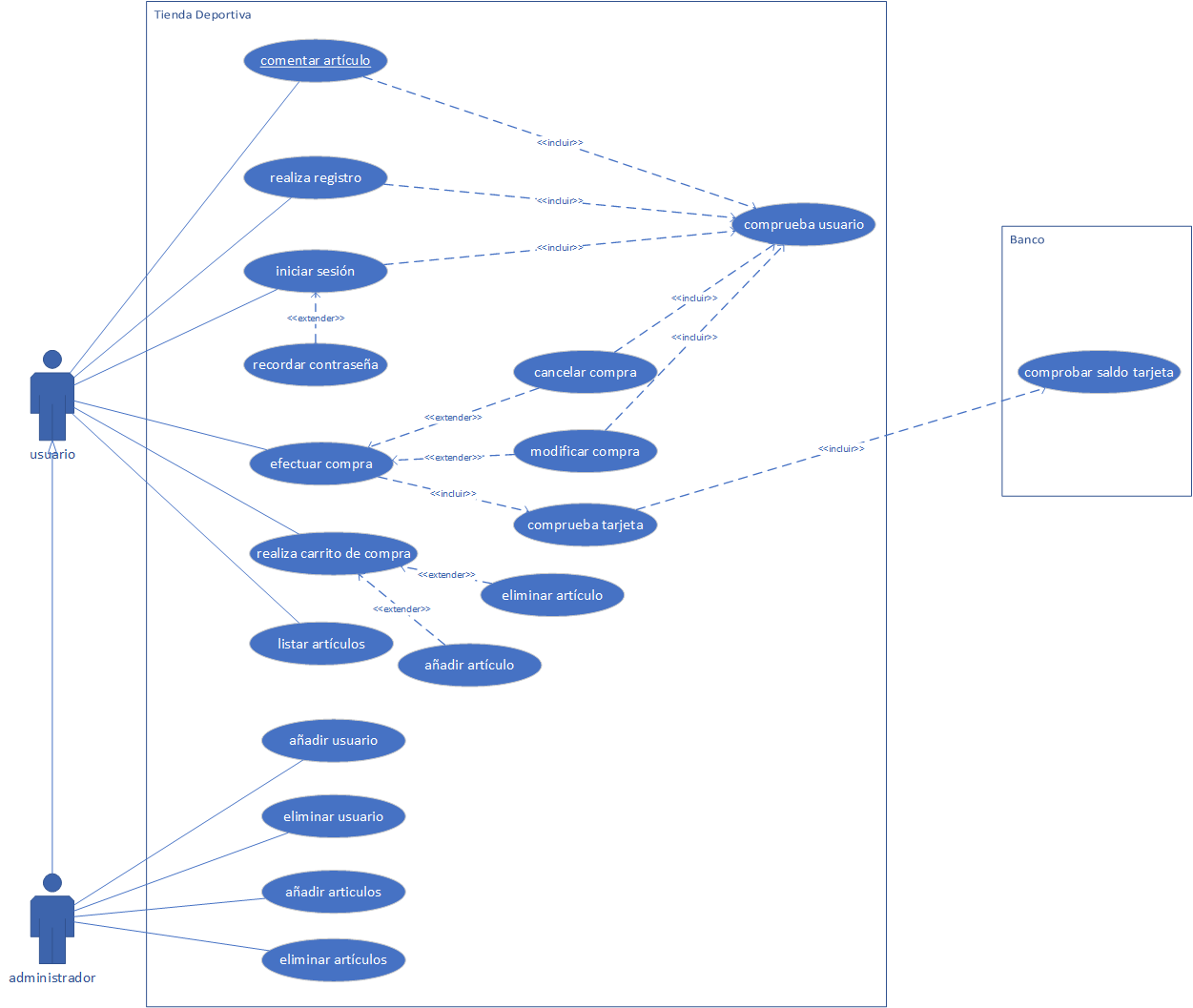
**Búsqueda simple:** Se hace uso de un buscador global, disponible en cualquier sección de la web, realiza búsquedas sobre artículos sin embargo no le hemos dado opciones de ayuda para el autocompletado.

**Paginación:** Debido a la gran cantidad de datos si hemos hecho uso de paginación en tablas donde se muestran datos de artículos marcas o categorías, ya que tal cantidad de datos ocupan demasiado espacio y pueden incomodar al usuario a la hora de buscar algo en específico.

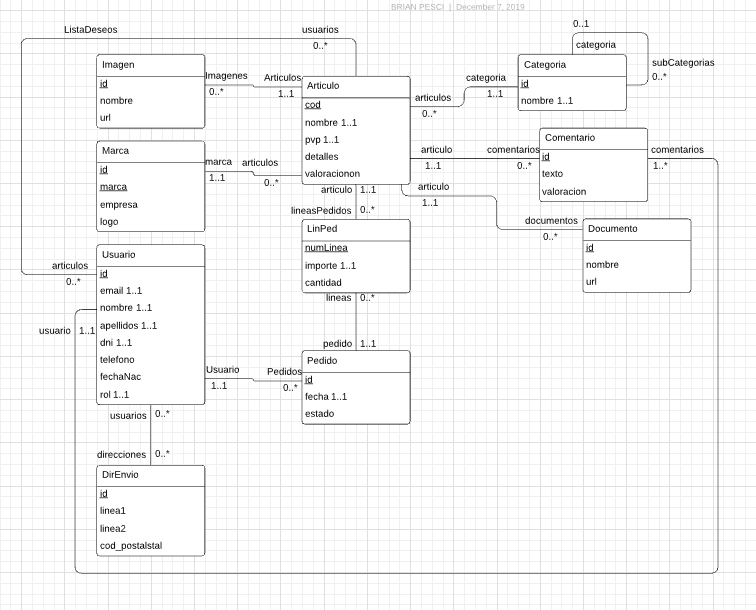
**Filtrado:** Se permite la ordenación por los campos mostrados en las tablas del administrador, además se pueden filtrar los artículos por nombre mediante el buscador o por género en el menú de navegación, una vez realizado este primer filtro, damos la opción de filtrar los datos por precio, categoría e incluso valoración, lo cual es bastante útil ya que las búsquedas pueden acotar bastante poco la cantidad de artículos y se observaba necesario el uso de otros filtros para reducir coincidencias.

# **Diagramas**

## **Diagrama de casos de uso**

****

## **Diagrama de clases**

****

# **Metodologías y planificación**

Hemos usado una metodología **Scrum** o lo más fiel posible a esta. También hay que añadir que el grupo ha estado en contacto via **whatsapp** en todo momento y se han realizado reuniones o ayudas entre los miembros por aplicaciones como **hangouts** o **uber**.

La metodología Scrum consiste en el trabajo en equipo a partir de iteraciones o Sprints.

Al final de cada Sprint, se va revisando el trabajo de la anterior semana. En función de esto, se priorizan y planifican las actividades del siguiente Sprint.

Se ha dividido el trabajo en diferentes **sprints** de duración aproximada de una semana, cada uno con sus respectivas tareas libres para su asignación por un miembro del grupo.

La finalización de cada sprint se cerraba con una reunión del grupo en la que se discuten las dificultades tenidas durante el desarrollo de las tareas, se aportan soluciones y se finalizaban con la discusión de qué nuevas tareas asignar para el nuevo sprint y su duración.

Para la planificación, creación y asignación de las tareas se ha usado el tablero de issues de **github**, que también se ha usado como controlador de versiones del proyecto. En su uso se han creado las ramas pertinentes por parte de cada usuario para su respectiva tarea.

# **Descripción de la implementación**

## **Vue**:

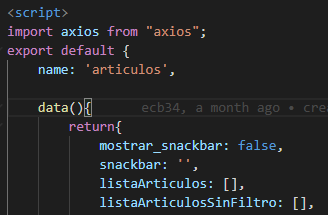
**Instancia**

Lo primero es crear la instancia de Vue y declarar que vamos a usar librerias como vue-router, vuetify o vuex.

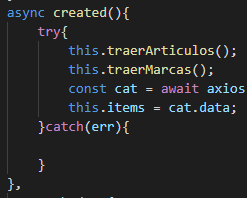
****

***Variables***

Utilizamos como framework JS vue, dentro de data tenemos las variables que vamos a utilizar, el código html se actualizará automáticamente al modificar las variables por ser reactivo.



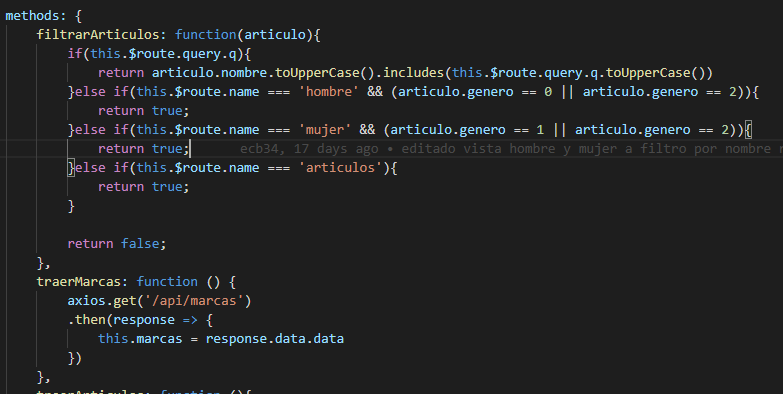
***Funciones de Vue***



Estos métodos se llaman como callback a diferentes estados de la aplicación.

**Funciones**

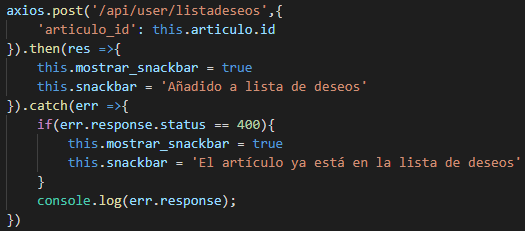
Se declaran dentro de la variable **methods**



Variables y métodos deben ir dentro de la etiqueta 

## **Axios**

Utilizamos esta librería para hacer llamadas asíncronas a una API REST y se invocará a un método cuando llegue la respuesta.



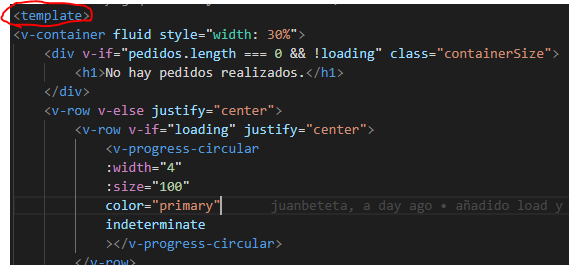
## **Vue-Router:**

Se encarga de redirigir al componente correspondiente, también se pueden colocar navigation wards para ejecutar funciones antes o después de entrar a una ruta. Se puede activar aquí la propiedad props, para pasar parámetros de ruta.

## **Vuex**

## **Master Page y Componentes:**

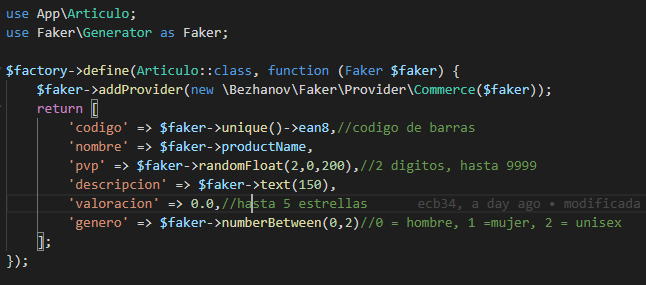
Utilizamos el archivo App.vue como master page y dentro de este es donde se cargan los componentes que son las vistas, de esto se encarga vue-router.



## **Seeders**

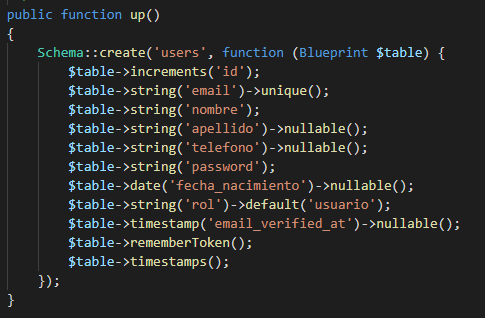
Utilizamos estas clases para poblar la base de datos de forma que en cualquier máquina se tienen los datos.

En nuestro caso también hemos utilizado factorías con la librería Faker que nos genera datos aleatorios.



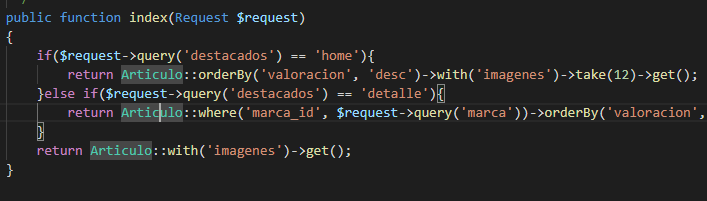
## **Migraciones**

Utilizamos las migraciones para crear la base de datos, debemos crear también los campos para las relaciones necesarias. Como ejemplo vemos la función up(), definimos el tipo, y las restricciones (string, unique)



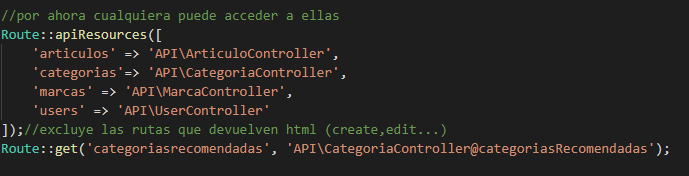
## **Controllers**

Pertenecen al backend, son las funciones que se encargan de responder a las peticiones HTTP.

Estas funciones están conectadas con las correspondientes rutas en el fichero 

## **Routing/API Rest:**

Rutas de resources que incluyen los métodos básicos del CRUD:

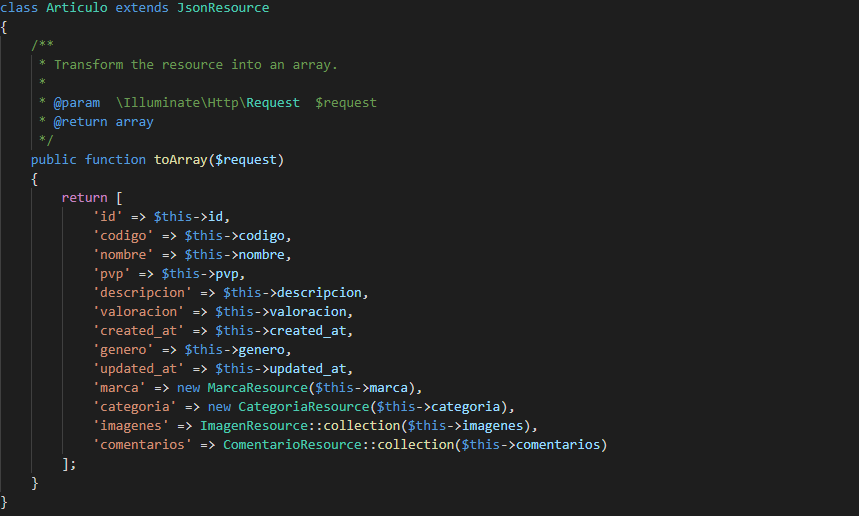


## **Rutas con protección de autorización**

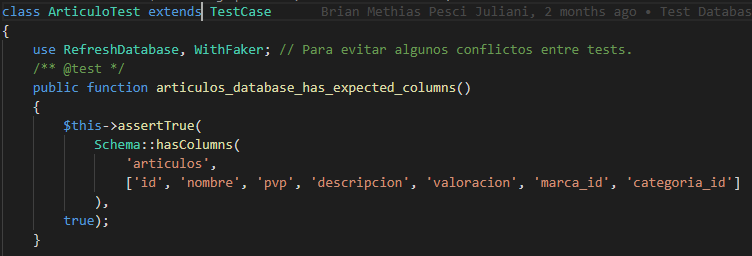


## **Resources**

Estas clases nos sirven para dar formato JSON a cualquier objeto.

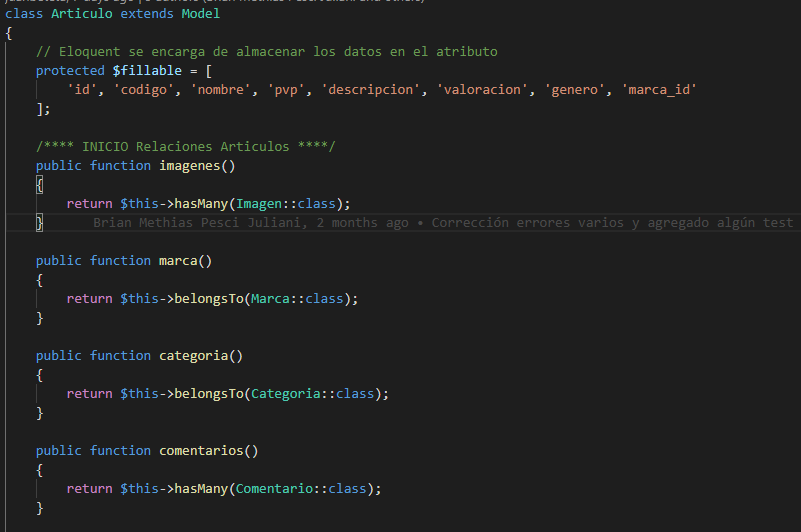
****

## **Test**

Utilizamos test en la capa de migraciones para asegurarnos que la BD está bien formada:

## **Modelos**

Estas clases se usan para representar y acceder a la BD con un objeto, también definimos las relaciones.



# **Problemas y solución**

* Encontramos un problema para reaccionar a los cambios de una variable que se pasa por props, lo arreglamos utilizando watchers sobre la variable y acabamos usando route-guards.
* Investigamos sobre añadir mapas de google para la localización de las tiendas y acordamos que para este sprint no lo tendríamos a tiempo y lo dejamos como posible mejora de la web.
* Encontramos problemas para la iteración de las imágenes a previsualizar en base64, se solucionó fijando el número de imágenes por usuario a un máximo de tres y su introducción por separado. Además, también hemos tenido problemas al tratar con las imágenes, teniendo problemas al guardarlas y recuperarlas en la base de datos.

# **Mejoras y ampliaciones**

* Como mejora se podría plantear la recuperación de contraseña de usuario
* Añadir mapas al listado de tiendas para ver la localización
* Añadir más categorías y filtros de búsqueda
* Llevar un control del stock de los artículos
* Tener gráficas de los artículos menos vendidos y poder añadir ofertas de artículos que lleven mucho tiempo en stock

# **Referencias**

<https://vuetifyjs.com/en/components/>

<https://laravel.com/>

<https://app.moqups.com/>

<https://laragon.org/>

<https://vuex.vuejs.org/>

<https://www.youtube.com/watch?v=J2Wp4_XRsWc>