

PROYECTO CICLO SUPERIOR DE DESARROLLO WEB

Curso 2017/2018

Memoria Técnica del Proyecto:

Desarrollo de una aplicación web para la Gestión de un Establecimiento Hotelero, Adicionalmente, se creará el sitio de un establecimiento de estas características y pondremos en marcha el sistemas de reservas.

Presentado por: Pedro Miranda Nisa

Almendralejo, a 13 de Diciembre de 2017

AGRADECIMIENTOS.

Me gustaría dar las gracias primero a mis padres por su constante apoyo y estar siempre ahí, tanto en los momentos malos como en los buenos.

A mis hermanas por creer siempre en mi.

A Belen por soportar estos años de locura.

A mis profesores por todo lo que me han, y están, enseñando cada día.

ÍNDICE

Memoria Técnica.....	1
1 RESUMEN.....	7
2 OBJETIVOS.....	7
3 DIAGRAMA DE GANT.....	8
4 CASOS DE USO.....	8
5 ANÁLISIS DE LOS REQUISITOS.....	9
5.1. Definición del proyecto.....	9
5.2. Catálogo de requisitos.....	11
6 DISEÑO DE LA APLICACIÓN.....	15
6.1 Arquitectura web.....	15
6.2 Estructura del proyecto.....	26
6.3 Diseño de la base de datos.....	35
6.4 Diseño web adaptable.....	42
7 IMPLEMENTACIÓN.....	44
7.1. Primeros pasos con Laravel.....	44
7.2 Página Principal.....	48
7.3 Módulo Usuarios y Roles.....	57
7.4. Módulos de Gestión de Clientes, Trabajadores, Reservas y Habitaciones.....	61
8 CONCLUSIONES.....	84
9 TRABAJO FUTURO.....	85
9.1 Mejoras en la consulta al correo.....	86
9.2 Mejoras en el Sistema de Reservas.....	86
9.3 Mejoras en la web.....	86
9.4 Nuevas funcionalidades.....	86
10 PRUEBAS Y RESULTADOS.....	87

10.1 Pruebas de resolución.....	88
10.2 Pruebas con navegadores.....	96
10.3 Pruebas de los métodos.....	101
10.4 Pruebas Unitarias.....	101
11 BIBLIOGRAFÍA.....	102
11.1 Tecnologías web.....	102
11.2 Entornos de programación.....	103
11.3 Otras herramientas.....	104
Manual de Instalación.....	105
1 CLONANDO EL PROYECTO.....	106
1.1 Instalar Composer.....	106
1.2 Instalar las dependencias usando Composer.....	106
1.3 Crear el Archivo .env.....	107
1.4 Generar una clave.....	108
1.5 Levantando el servidor.....	109
2 DESCARGAR EL PROYECTO.....	109
Manual del Administrador de Sistemas.....	110
1 Descripción del Sistema.....	111
2 Configuración previa.....	111
3 Instalación de GestHotel.....	112
4 Acceso a GestHotel.....	112
5 Inicio de GestHotel.....	113
6 Página Principal.....	113
7 Panel lateral.....	114
8 Dashboard.....	125
9 Reservas.....	125
9.1 Reservadas.....	125

9.2 Disponibilidad.....	128
10 Administración.....	131
10.1 Cliente.....	132
10.2 Habitación.....	138
10.3 Trabajador.....	141
10.5 Reportes.....	148
11 Sistema.....	149
12 Logout.....	158
Manual del Administrador de Contenidos.....	159
1 Descripción del Sistema.....	160
2 Configuración previa.....	160
3 Acceso a GestHotel.....	160
4 Inicio de GestHotel.....	160
5 Página Principal.....	161
6 Panel lateral.....	162
7 Dashboard.....	164
8 Reservas.....	164
8.1 Reservadas.....	164
8.2 Disponibilidad.....	167
9 Administración.....	170
9.1 Cliente.....	170
9.2 Habitación.....	176
9.3 Trabajador.....	179
9.4 Tipo Habitación.....	183
10 Logout.....	186

Manual de Usuario.....	187
1 Descripción del Sistema.....	188
2 Acceso a GestHotel.....	188
3 Inicio de GestHotel.....	188
4 Página Principal.....	189
5 Panel lateral.....	190
6 Dashboard.....	191
7 Reservas.....	191
7.1 Reservadas.....	191
7.2 Disponibilidad.....	193
8 Administración.....	196
8.1 Cliente.....	196
8.2 Habitación.....	200
8.3 Reportes.....	201
9 Logout.....	202

1 RESUMEN

La idea principal de este Proyecto es desarrollar una herramienta para centralizar la gestión de un hotel.

En resumen, la aplicación creada permitirá gestionar el alta de clientes, reservas de habitaciones, checkin y checkout, reportes, altas de trabajadores, etc.

A lo largo de la presente memoria, se irán explicando con mas detalle las distintas funcionalidades del aplicativo.

2 OBJETIVOS.

A medida que las nuevas tecnologías han avanzado a lo largo del tiempo han ido surgiendo nuevas oportunidades que permiten mejorar la gestión de una cadena hotelera para, de esta manera, poder controlar todas las áreas de las que consta un establecimiento. En este sentido, cuando hablamos de todas las herramientas tecnológicas que pueden ayudar al director de un hotel en su día a día, sin duda la primera que viene a nuestra mente es el software de gestión hotelera. Por ello, vale la pena detenerse y analizar cómo queremos gestionar nuestro hotel y qué necesidades nos gustaría solventar.

Cabe destacar que, en el momento en el que nos encontramos es impensable, en un futuro próximo, que un hotel perviva sin la existencia de estos sistemas o con ellos ya obsoletos, fundamentalmente por las distintas exigencias del cliente, que demanda información rápida y constante de reservas, disponibilidad, tarifas...

Precisamente contar con un minucioso conocimiento del consumidor final, es una de las ventajas que nos pueden ofrecer estos tipos de sistemas de gestión hotelera.

Igualmente, en el desarrollo de estos software se ha buscado tener accesibilidad universal a cada programa desde todo tipo de dispositivos móviles.

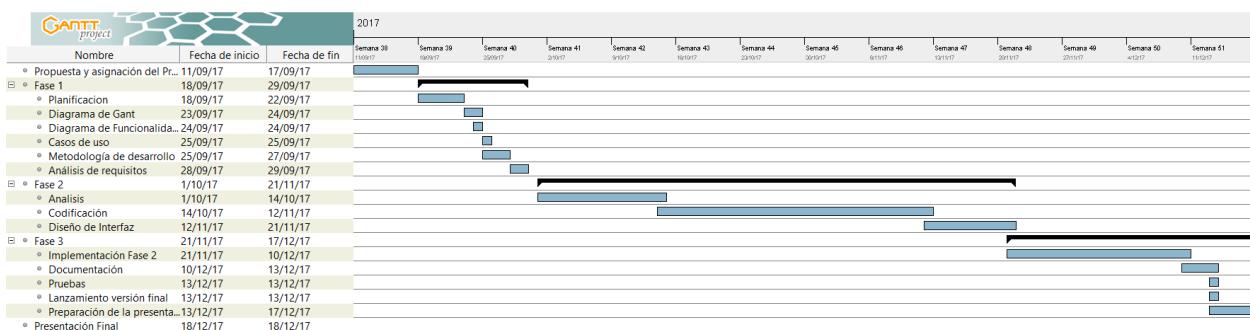
En relación a esto, cada vez existe un mayor número de soluciones diseñadas específicamente para dispositivos móviles (smartphones y tablets), mientras que otros

proveedores de software se decantan por rediseñar sus herramientas para ofrecer un servicio adaptado a este nuevo entorno.

En contrapartida, se decide implementar un aplicativo web, con el objetivo de poder utilizar el sistema desde cualquier dispositivo y en cualquier lugar con una simple conexión a internet.

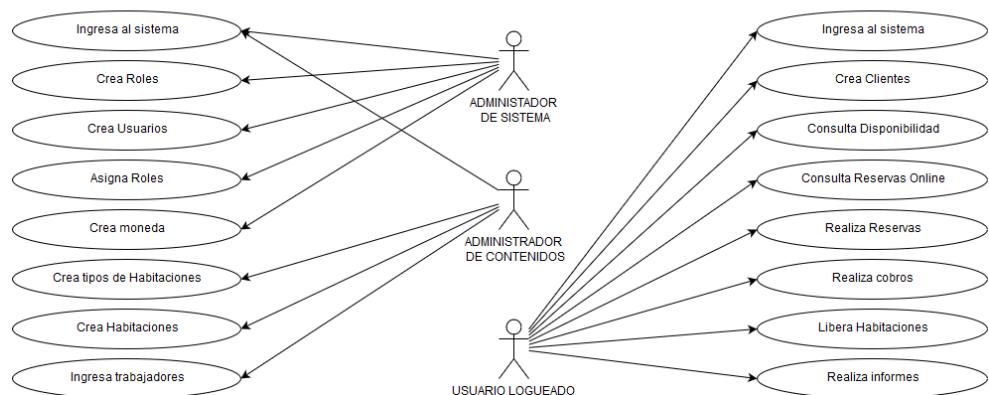
El presente trabajo propone el Análisis y Diseño de un Sistema Web para el control de información para un hotel, de manera que el sistema cumpla con los requerimientos, donde el personal del hotel que utilice el sistema puedan realizar el registro de Huéspedes, registro de Habitaciones, registro de Empleados, reportes de libros diarios, control de habitaciones y control de los servicios.

3 DIAGRAMA DE GANTT.



4 CASOS DE USO

CASOS DE USO



5 ANÁLISIS DE LOS REQUISITOS

En este capítulo se define el proyecto y los requisitos necesarios para su implementación. En los próximos apartados se profundiza en las características concretas del producto, así como en el catálogo de especificaciones y se expone el modelo de ciclo de vida a seguir.

5.1. Definición del proyecto

La idea principal de este Proyecto es desarrollar un producto software, en este caso una aplicación Web, para centralizar la gestión de un hotel.

En resumen, la aplicación creada permitirá gestionar el alta de clientes, reservas de habitaciones, checkin y checkout, reportes, altas de trabajadores, etc.

A lo largo de la presente memoria, se irán explicando con mas detalle las distintas funcionalidades del aplicativo.

Se seguirán todas las etapas del proceso de desarrollo de software (Figura 1): planificación, análisis, diseño, implementación, pruebas, instalación y mantenimiento, usando un modelo de ciclo de vida en cascada. Es un modelo clásico, ideal para proyectos estables y pequeños, donde los requisitos quedan bien documentados al inicio del ciclo.

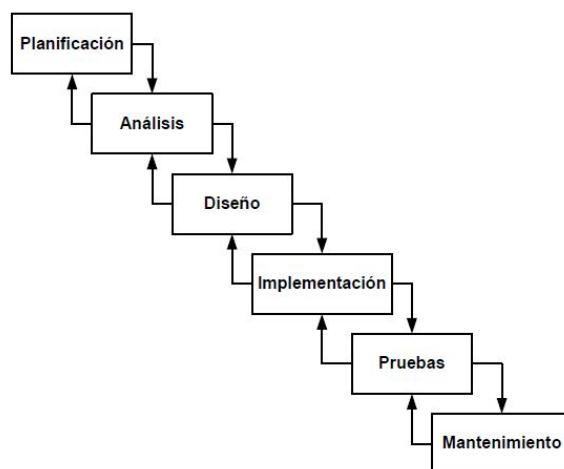


Figura 1. Ciclo de vida clásico: modelo en cascada

Cada fase se completa en orden y tras la verificación, se puede volver a la fase anterior en caso necesario.

No se realizará la entrega del producto final hasta la validación de la última etapa. Esta secuencia lineal puede tener varios inconvenientes, como los riesgos de modificar los requisitos en una fase más avanzada, por ello en proyectos reales, se opta por seguir otros modelos como el iterativo, incremental, en espiral o de prototipos.

A partir de este momento en este documento, se utilizará *GestHotel* como nombre de la aplicación real que se describe en este documento.

GestHotel surge de la necesidad de desarrollar un sistema Web para la gestión del establecimiento hotelero, con los objetivos descritos en el capítulo anterior.



Figura 2. Logo de la aplicación

5.2. Catálogo de requisitos

Una vez que se han introducido los objetivos de la aplicación y se ha estudiado las tecnologías y servicios disponibles actualmente, es hora de exponer todos los requerimientos necesarios tanto para el usuario como para el entorno del sistema.

5.2.1 Requisitos funcionales

Aplicación:

- La aplicación no permitirá mostrar el contenido si el usuario no está registrado., así como participar y modificar otras configuraciones de la página.
- La aplicación permitirá acceder a según que secciones dependiendo del rol del usuario logueado.
- El sistema dispondrá de los siguientes módulos:
 - Módulo de gestión de los usuarios y roles.
 - Módulo de gestión de los clientes.
 - Módulo de gestión de los trabajadores.
 - Módulo de gestión de las reservas.
 - Módulo de gestión de las habitaciones.
 - Módulo para la creación e impresión de reportes.
 - Modulo para la gestión de monedas y otros parámetros.
 - Modulo para la gestión del sitio web de la empresa.
- La aplicación aceptará solamente tres tipos de usuarios:
 - Administrador de sistemas.
 - Administrador de contenidos.
 - Usuarios logueados.
- La aplicación listará:
 - Los usuarios y sus roles, que están registrados.
 - Los clientes que están registrados.
 - Los trabajadores que estén registrados.

- Las habitaciones y su tipología, que estén almacenadas.
 - La moneda en curso.
 - Las reservas y el estado en el que se encuentran.
- La aplicación tendrá un servicio de correo para recibir mensajes de los usuarios de la web, tanto para recibir reservas desde la misma, como para cualquier sugerencia o queja de los mismos.
- El registro de clientes y trabajadores deberá almacenar campos necesarios como nombre y apellidos, dni, dirección, teléfono, email, etc.
- La aplicación permitirá cargar la disponibilidad de las habitaciones de forma automática, a través de un fichero que debe tener un formato específico, que cuente con los campos necesarios para realizar nuevas reservas en aquellas que estén libres.
- Las nuevas reservas deben permitir guardar valores como en numero y tipología de la habitación, el nombre del cliente, la fecha de entrada y salida, el total de la factura, etc.
- El sistema tiene que ser capaz de filtrar búsquedas de clientes, trabajadores, habitaciones, reservas, en definitiva, cualquier registro susceptible de ser listado mediante la intervención del usuario logueado.
- El sistema será capaz de crear, mostrar e imprimir, informes de las reservas entre un periodo elegido por el usuario logueado.

Administrador de sistemas:

- El sistema, en principio, solo permitirá un único administrador de sistemas, salvo que este último decida establecer más usuarios con este rol.
- El administrador de sistemas tendrá acceso a todas las secciones del panel administrativo, podrá crear, modificar y eliminar cualquier registro.
- Excepcionalmente tendrá acceso al módulo sistema desde donde se crea, edita y elimina:
 - Usuarios.
 - Roles.
 - Moneda.

- El administrador de sistemas podrá acceder a las configuraciones del sistema, como la carga o edición de cualquier contenido del mismo.
- Este será el único rol con acceso a asignar roles a los usuarios.

Administrador de contenidos:

- El rol de administrador de contenidos será asignado por el administrador de sistemas.
- Tendrá acceso a cualquier parte del panel administrativo, exceptuando el módulo sistema.

Usuarios logueados:

- Los usuarios logueados como cualquier otro rol en la aplicación, serán asignados por el administrador de sistemas.
- Tendrán acceso ilimitado a la sección Reservas.
- Podrán listar y/o editar clientes.
- Listar habitaciones.
- Consultar el correo.
- Y generar reportes.

3.2.2. Requisitos no funcionales

Interfaz y usabilidad:

- La interfaz gráfica será sencilla, atractiva y fácil de manejar, basado en usuarios con unos conocimientos mínimos en ofimática e Internet.

Documentación:

- La aplicación dispondrá de:
 - Manual de instalación.
 - Manual de Usuario logueado.

- Manual de Administrador de Contenidos.

Mantenimiento y portabilidad:

- El administrador de sistemas tendrá la opción de poder mantener el sistema con actualizaciones de versiones con el fin de corregir errores y mejorar los servicios.
- La aplicación será privada, permitiendo al administrador el acceso al servidor.
- La aplicación será escalable con el objetivo de añadir nuevas funcionalidades a la misma.

Recursos:

- La información de todos los usuarios, clientes, reservas, etc de la aplicación se almacenará en una base de datos y se dispondrá de módulos de gestión para usuarios y administradores.

Verificación y fiabilidad:

- Para registrar un usuario, habrá que realizar un cuestionario con información obligatoria como el nombre de usuario, contraseña y correo electrónico.

Rendimiento:

- La aplicación permitirá el acceso a un usuario por dispositivo de forma concurrente.
- El tiempo de respuesta de la aplicación no deberá de exceder de treinta segundos.

Requisitos tecnológicos:

- La aplicación no requiere de grandes requisitos, mas que un dispositivo con conexión a internet.
- La aplicación usará diseño responsivo y será compatible con dispositivos móviles y tabletas con conexión a Internet.

6 DISEÑO DE LA APLICACIÓN

Una vez conocidas las funcionalidades que hay que implementar, es necesario explicar cómo se va a implementar. Primero se especificará la arquitectura elegida que resuelva mejor el problema, junto con las herramientas utilizadas y posteriormente se presentarán los diagramas de diseño de la base de datos y los modelos de la aplicación.

6.1 Arquitectura web

Para explicar la arquitectura web de *GestHotel*, es necesario elegir previamente las herramientas o tecnologías que se van a usar para desarrollar la aplicación, según los requisitos necesarios de la sección anterior. Al haber múltiples alternativas, es una fase importante ya que una buena o mala elección, repercute en las siguientes fases del proyecto.

6.1.1.Herramientas de Back-End

Sistema Operativo:

El sistema operativo que se va a utilizar en el lado del servidor Web es *Windows 10*. A priori la compatibilidad, soporte y comunidad que ofrecen los productos sobre Windows, es suficiente para instalar y configurar todo lo necesario.



Figura 3. Herramientas de Back-End

Servidor:

Para el desarrollo del proyecto, resulta indispensable la utilización de un servidor Web que atienda las peticiones de los clientes y responda con los contenidos correspondientes. IIS es uno de los servidores que mayor servicio ofrecen, pese a que actualmente ha sido superado por otros como NGINX, con buenas características en el apartado de balanceo de carga y tolerancia a fallos.

Pero el servidor que se va a emplear es Apache por los siguientes motivos:

- Es un servidor Web de código abierto que implementa el protocolo HTTP/1.1 y uno de los más populares.
- Es multiplataforma y su arquitectura es muy modular y extensible, lo que permite de una manera muy sencilla ampliar sus capacidades.
- Un caso concreto del servidor web Apache es el módulo ModRewrite, que se encarga de traducir, redirigir y modificar direcciones URL para hacerlas más amigables. Hoy en día, este módulo es una herramienta indispensable sobre todo de cara al posicionamiento en Google, ya que una URL es sensible a mayúsculas y minúsculas y no es un caso favorable al duplicar contenido. Aunque generalmente se utiliza para conseguir que las direcciones visibles sean sencillas de recordar y evitar las incómodas variables.

Lenguaje de programación:

A la hora de elegir un lenguaje interpretado, existe una gran variedad: ASP.net, Java, Ruby, Python, PHP o incluso JavaScript también del lado del servidor (que puede resultar atractivo, al usar un mismo lenguaje en toda la aplicación).

Pero se ha decidido usar PHP Hypertext Preprocessor, al ser el lenguaje más conocido (cursado en el propio módulo) y con mayor comunidad de usuarios detrás, que cuenta con una extensa biblioteca de funciones.

PHP sirve para crear aplicaciones web dinámicas con acceso a información almacenada en una base de datos y que permite incorporar código en las páginas HTML de forma sencilla. Usa técnicas de programación orientada a objetos y sigue el patrón de diseño Modelo Vista Controlador. Es usado en millones de aplicaciones, entre las que destaca Wikipedia o Facebook, además es un lenguaje soportado perfectamente por el servidor web Apache.

Para entenderlo mejor, PHP se comporta como un módulo de Apache, que extrae código dentro de las páginas, lo ejecuta en el servidor y envía el resultado al cliente. Éste no puede visualizar el código del programa, solamente su resultado. La Figura 4 muestra un ejemplo sencillo de su funcionamiento.

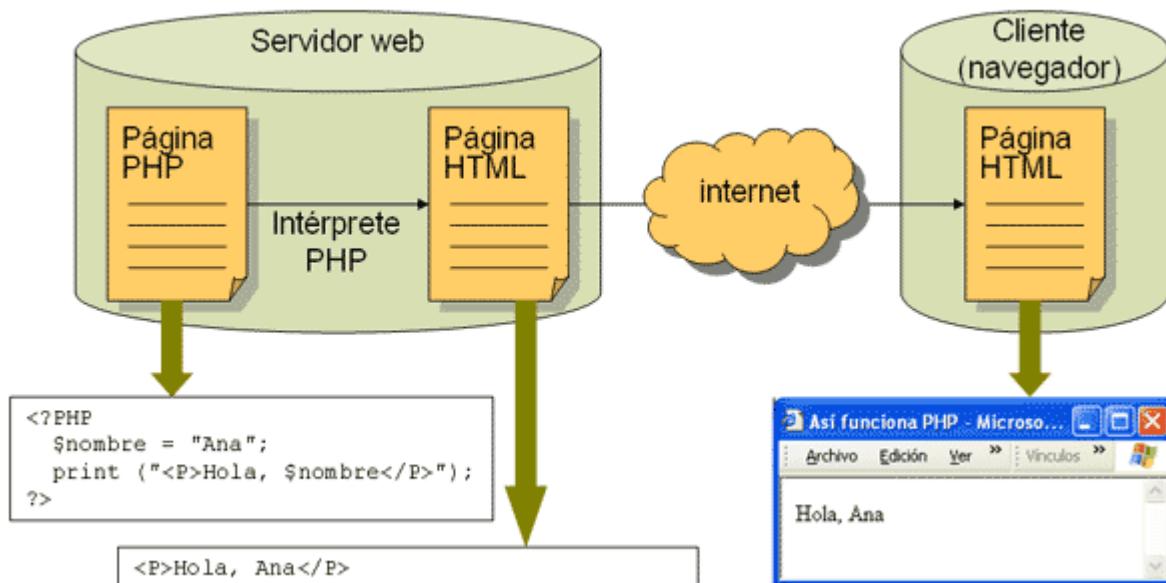


Figura 4. Funcionamiento de PHP

El servidor Web y en especial, este módulo PHP, puede conectarse a una base de datos para almacenar o extraer información a la hora de elaborar las páginas.

Base de Datos:

Aquí también existen varias alternativas: SQL Server, MySQL, PostgreSQL o MongoDB, una base de datos NoSQL en la que se puede ejecutar JavaScript para realizar consultas.

El sistema de gestión de bases de datos elegido es MySQL por dos motivos fundamentales. El primero, es un sistema que ya conozco y por tanto la curva de aprendizaje será menor. El segundo, es un sistema libre.

Además de lo ya mencionado, entre sus ventajas se encuentran: funciones en distintos lenguajes, alta concurrencia sin bloqueos al acceder a los datos, buenos

manuales y documentación, con una comunidad de desarrolladores activa bastante grande, etc..

6.1.2. Herramientas de Front-End

Los contenidos devueltos por el servidor son interpretados por los navegadores Web de los clientes, que los muestran por pantalla acorde a unas fuentes y formatos. Es decir, el servidor Web se limita a transferir el código de la página sin llevar a cabo ninguna interpretación de ella. Aquí es donde intervienen las herramientas del lado del cliente. Una de ellas se trata del lenguaje HTML, un lenguaje de marcado que se ideó con el propósito de definir la estructura de una página o documento web, que tiene como finalidad reunir una serie de información en diferentes formatos (texto, imágenes, video, audio, etc.).



Figura 5. Herramientas de Front-End

Aunque la funcionalidad y estructura de una página web es sencilla, se puede complicar en exceso si se incluye en una misma página los contenidos, el diseño y la programación. La idea es separar la estructura de un documento de su presentación y diseño.

CSS es un lenguaje utilizado para definir la presentación de un documento estructurado escrito en *HTML*.

Con el paso del tiempo, este lenguaje de hojas de estilo ha evolucionado mucho, permitiendo desde las tareas más elementales, como cambiar las dimensiones o los colores de un elemento, hasta efectos interactivos, transiciones o animaciones. Para aprovechar estas características sin requerir de un gran coste de tiempo adicional, se usa *Bootstrap 3.3.4*, un framework para CSS que ayuda a integrar al proyecto componentes prediseñados y adaptado para un diseño responsivo de la web. Es muy utilizado y es soportado por la mayoría de los navegadores web actuales, con algunas excepciones para versiones inferiores de Internet Explorer 9

	Chrome	Firefox	Internet Explorer	Opera	Safari
Android	Soportado	Soportado	N/A	No soportado	N/A
iOs	Soportado	N/A		No soportado	Soportado
Mac OS X	Soportado	Soportado		Soportado	Soportado
Windows	Soportado	Soportado	Soportado	Soportado	No soportado

Navegadores que soportan Bootstrap 3.3.4

La aparición de otras tecnologías, como el lenguaje de programación interpretado *JavaScript*, provocó que las páginas *HTML* también incluyeran el código de las aplicaciones que se utilizan para crear páginas web dinámicas, por lo que es conveniente la separación.

Se va a usar el framework *jQuery 1.11.3* de *JavaScript* entre otras librerías que hablaremos mas adelante sobre ellas, para ayudar a incorporar efectos dinámicos, animaciones, modificaciones del comportamiento en determinados eventos o ventanas con mensajes de aviso al usuario.

6.1.3. Otras Herramientas

PhpStorm 2016.1

PhpStorm es un IDE de programación desarrollado por JetBrains. Es uno de los entornos de programación más completos de la actualidad, permite editar código no sólo del lenguaje de programación php como lo indica su nombre.

Actualmente es compatible con Sistemas Operativos Windows, Linux y Mac OS X.

Algo que destaca en PhpStorm es la ejecución de nuestro código en la misma interfaz del IDE. Así como también la interpretación y visualización inmediata de código php hasta en 5 de los navegadores web más populares. Con esto nos olvidamos de tener que cargar manualmente nuestro navegador e ingresar la dirección url de la página.

Con PhpStorm podemos crear nuevos proyectos basándonos en algún framework de diseño web y Cms. Como los que se muestran a continuación:

- Twitter Bootstrap
- HTML Boilerplate
- Composer Project
- Drupal Module
- App Engine Project
- Foundation

Al seleccionar cualquiera de los anteriores, el IDE detectará la versión actual, por ejemplo Bootstrap v3.1.x y descargará los archivos de esta librería para añadirlos a la carpeta de nuestro proyecto.

LibreOffice Writer

Para el desarrollo de la presente memoria, se ha optado por LibreOffice Writer como editor de texto por ser el paquete de oficina libre y abierto más eficaz del mercado y poseer prácticamente las mismas características que cualquier otra suite de pago.

Nos permite diseñar y producir documentos de texto que pueden incluir ilustraciones, tablas y diagramas. Puede archivar los documentos en una gran variedad de formatos, incluyendo el formato estándar de "Documentos Abiertos" [(OpenDocument format (ODF)], formato de Microsoft Word (.doc) o también de HTML. De la misma manera podemos exportar nuestros documentos en documentos de Formatos Portables [Portable Document Format (PDF)].

GanttProject 2.7.2

GanttProject es una herramienta gratuita para crear una completa planificación de un proyecto de forma muy visual. Todo queda bajo control en *GanttProject*, desde los recursos necesarios en forma de personal, los días festivos, hasta dividir el proyecto en un árbol de tareas y asignar a cada uno los recursos oportunos. Un punto interesante es que permite establecer dependencias entre las tareas, de esta forma, una tarea no podrá empezar hasta que esté acabada la anterior. Como punto final, *GanttProject* permite exportar tu trabajo a una imagen (JPG, PNG), PDF y HTML.

Draw.io

Es una herramienta que nos permite elaborar diagramas en linea sin necesidad de instalar absolutamente nada en nuestro PC. Su interfaz es bastante sencilla y fácil de utilizar, ademas es tan completa que nada tiene que envidiarle a cualquier software de pago para escritorio.

Sus opciones son todas las necesarias para elaborar completos diagramas. Dispone de una gran variedad de formas y diseños predeterminadas que luego podemos moldear a nuestro gusto. Permite agregar rápidamente imágenes externas utilizando el buscador de Google y tiene múltiples opciones de texto con formato que podemos configurar como mejor nos parezca.

Para utilizar esta aplicación no es necesario ningún tipo de registro, es totalmente gratuita y los trabajos realizados pueden ser guardados en formato .XML para su posterior modificación con la herramienta, podemos imprimirlas o también permite exportar los diagramas a formatos .PNG, .GIF, .JPG, .PDF y .SVG, o si preferimos podremos insertar el diagrama realizado en cualquier sitio web utilizando un código que nos genera la aplicación.

Greenshot 1.2.9

Es una aplicación gratuita y open source, que nos permite tomar capturas de pantalla en Windows, y además incluye un editor de imágenes completo y fácil de usar.

Greenshot no solo nos deja capturar cualquier región, ventana, aplicación, escritorio, o hasta una página web completa, sino que incluye un editor incorporado para que realicemos ajustes rápidos y sencillos en un par de clics. Además de esto podemos exportar las imágenes en varios formatos, y luego compartirlas en redes sociales o almacenarla en la nube.

Justinmind

Disponible para Windows y Mac, con versión gratuita y versión pro bajo suscripción, Justinmind permite crear bocetos de aplicaciones simulando acciones (desde botones a formularios), con una gran cantidad de elementos que podemos arrastrar y soltar para crear un resultado lo más parecido al producto final posible.

6.1.4. El framework Laravel

Ponerse a desarrollar desde cero con estas herramientas puede parecer una tarea bastante complicada. Es por ello que se hace uso de frameworks generales que integren otras herramientas, como las especificadas anteriormente, para facilitar el trabajo e indicar al programador una estructura organizada de la aplicación para que luego sea posible un mejor mantenimiento.

Se puede empezar mezclando código PHP en la vista y que el controlador termine realizando todo el trabajo o seguir un patrón que resuelva estos problemas desde el punto de vista de la ingeniería del software.



Figura 6. Frameworks PHP

Laravel es un potente framework de última generación, creado por Taylor Otwelen el año 2011, que facilita a los desarrolladores la creación de aplicaciones web de forma rápida con un código limpio, elegante, ordenado y con patrones profesionales para los usuarios más avanzados. Es decir, permite crear tanto aplicaciones sencillas como más complejas, de forma fácil y divertida. Pero, ¿por qué escoger Laravel y no otros frameworks PHP?

Laravel tiene una sintaxis mucho más sencilla de escribir y de entender, manteniendo el enfoque de PHP. Frameworks como *Symfony* o *Zend* son más aburridos al intentar parecerse más a *Java*, *CakePHP* a *Ruby* y *CodeIgniter*, como opinión personal, ha sido de los pocos que se ha mantenido en los primeros puestos de referencia. Tanto *Laravel* como *CodeIgniter* son fáciles de usar, pero éste último se quedó un poco anticuado. Las diversas formas de usar PHP por millones de usuarios, ha acabado perjudicando al lenguaje y haciéndolo menos seguro y profesional.

Laravel surge para rescatarlo y volver a confiar en PHP, aprovechando al máximo las características de las últimas versiones, como el uso del paradigma *POO* (programación orientada a objetos).

Laravel se adapta a proyectos sencillos y complejos, de forma más ordenada y estructurada, haciendo que las aplicaciones sean más fáciles de mantener. Destaca su profesionalidad y velocidad de desarrollo frente al resto de opciones: routes, closures, helpers, validación de formularios, middlewares, internacionalización, objetos *ORM*, motor de plantillas, etc. Todo esto se verá más adelante en el capítulo siete.

Frameworks como *Symfony* siguen una arquitectura web orientada más bien a cliente-servidor. Sin embargo, la mayoría usa el patrón *MVC* (Model View Controller) que funciona bien en aplicaciones web y aquí es donde aparece *Laravel* para renovar el estilo de programación de PHP, usando una variante de *MVC* (Figura 7).

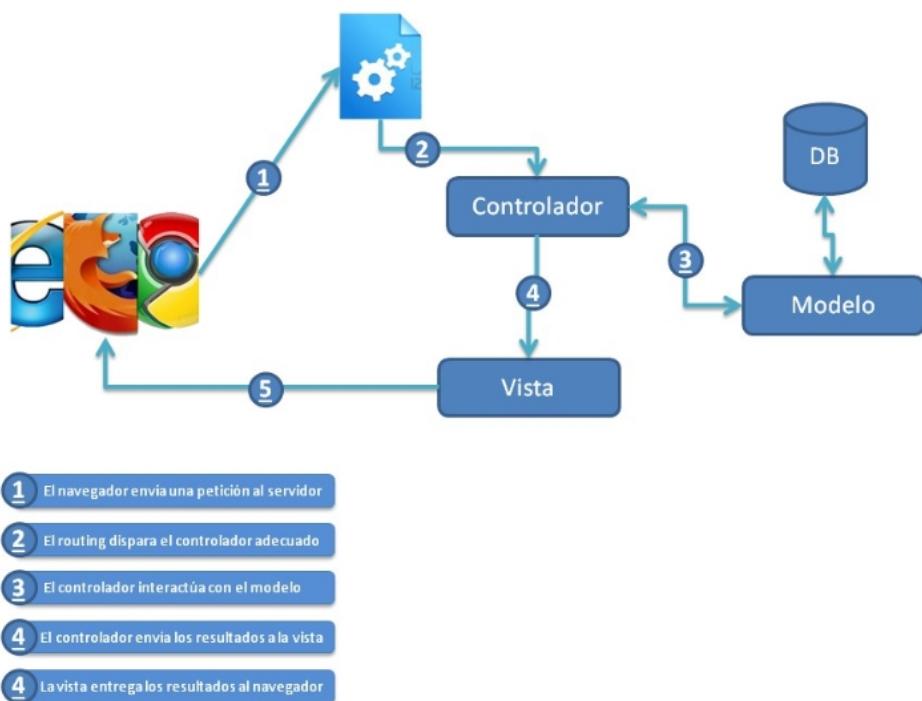


Figura 7. Patrón de diseño MVC con Laravel

En primer lugar, el navegador envía una petición al servidor. Laravel dispone de un sistema de enrutamiento que elige el controlador adecuado para procesar la solicitud. El controlador interactúa con el modelo para conectarse a la base de datos y recuperar o almacenar la información necesaria. Luego, el controlador envía los resultados a la vista. Por último, es la vista la que se encarga de entregar los resultados al navegador.

El modelo de secuencia de la aplicación sería el que se muestra en la Figura 8.

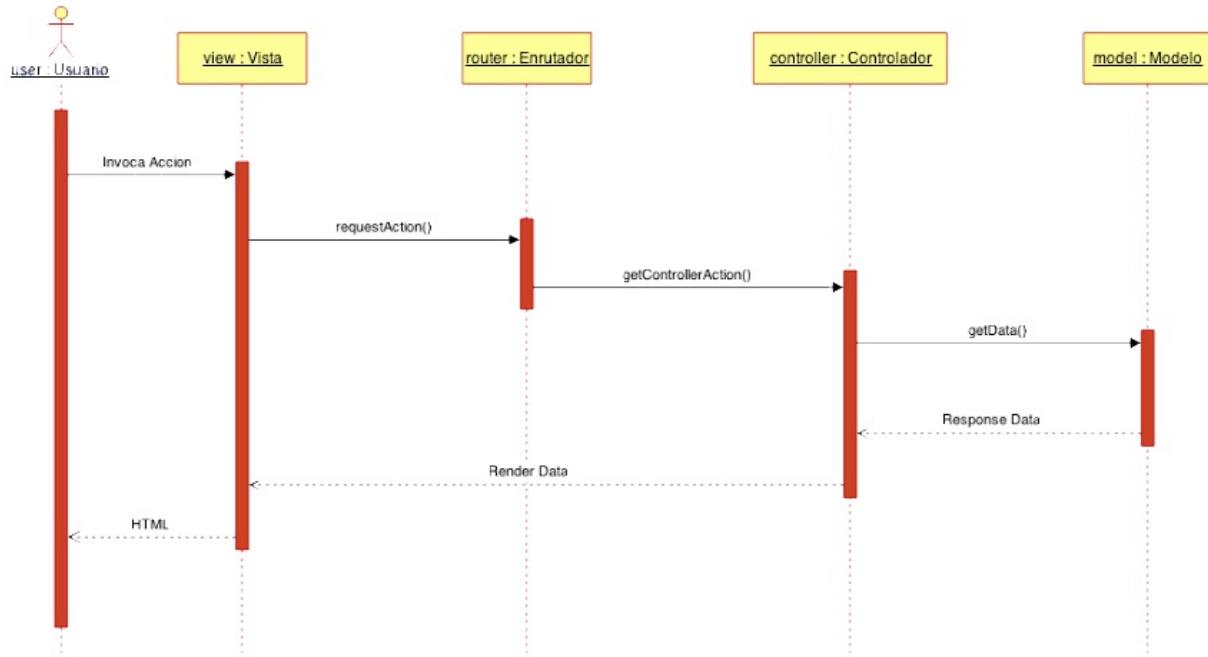


Figura 8. Diagrama de secuencia con Laravel

6.2 Estructura del proyecto

Es conveniente hacer un resumen antes de comenzar con el desarrollo de GestHotel. El proyecto utiliza *Apache*, *PHP*, *MySQL* y *Laravel*, integrado con *Bootstrap* y *jQuery*, además del *IDE* de desarrollo *PhpStorm*.

La aplicación *GestHotel* se creará dentro de la carpeta */www* que es la ruta que el servidor web *Apache* tiene configurada para poder tener acceso a los datos del sitio Web. Los clientes podrán acceder al sitio a través del navegador web *Firefox*, por ejemplo. Lo primero a tener en cuenta cuando se crea una aplicación con *Laravel* es la funcionalidad que tiene cada uno de los archivos y carpetas en los que está estructurado (Figura 9).



Figura 9. Estructura de directorios de un Proyecto Laravel.

/app

Lo primero que tenemos es el directorio app. App por defecto aloja a todo el código personal del proyecto. Eso incluye clases que puedan ofrecer funcionalidad a la aplicación, archivos de configuración y más. Es la carpeta más importante que cubriremos en detalle al final de esta sección.

/bootstrap

El directorio bootstrap contiene archivos que están relacionados con los procedimientos de inicialización del framework.

autoload.php: contiene la mayoría de esos procedimientos.

paths.php: contiene una matriz de las rutas comunes del sistema que son usadas por el framework.

start.php: contiene más procedimientos de inicialización para el framework , donde se puede establecer los entornos (environments) del framework.

/vendor

El directorio vendor contiene todos los paquetes de *Composer* que son utilizados por la aplicación. Por supuesto, esto incluye el paquete del framework de *Laravel* .

/public

Generalmente el directorio public es la única entrada de una aplicación *Laravel*, donde se encuentran los archivos CSS, JavaScript e imágenes.

packages/: será usado para contener cualquier fichero que necesite ser instalado por paquetes de terceras partes. Se mantienen en un directorio separado para que no creen conflictos con los ficheros de nuestra propia aplicación.

.htaccess: para servidores Apache. Contiene algunas directivas de configuración estándar.

favicon.ico: es una imagen de 16x16px que se muestra en las pestañas del navegador, por defecto está en blanco para ser reemplazado más tarde si se desea.

index.php: es el controlador frontal del framework de Laravel. Es el primer archivo que el servidor web ejecuta cuando llega una petición del navegador.

robots.txt: permite todos los hosts por defecto.

/artisan

El archivo artisan es un ejecutable que es usado para ejecutar la interfaz de línea de comandos Artisan para Laravel. Artisan contiene un buen número de comandos útiles para ofrecer atajos o funcionalidad adicional al framework.

/composer.json y /composer.lock

Tanto composer.json como composer.lock, contienen información sobre los paquetes de Composer usados por este proyecto.

/phpunit.xml

El archivo phpunit.xml ofrece configuración por defecto para las pruebas unitarias de PHPUnit.

Gestionará la carga de dependencias de Composer y ejecutará cualquier prueba ubicada en la carpeta app/tests. Revelaremos información sobre las pruebas en Laravel en un capítulo más adelante.

/server.php

Está destinado a ser utilizado con el servidor web interno de php.

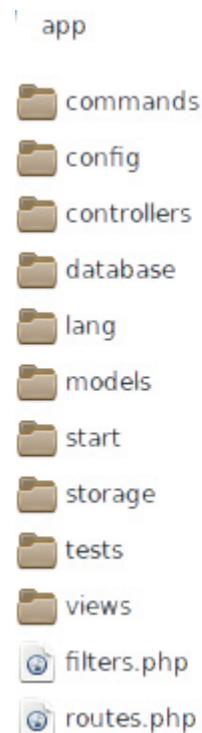
Este servidor web está diseñado solo con fines de desarrollo y no debe utilizarse en producción.

.env y /.env example

El archivo .env no existe cuando instalamos laravel, en este archivo se configurará el modo en que se ejecuta nuestra aplicación, por defecto será el modo debug, además podemos configurar la conexión a la base de datos y la conexión con el servidor de correo electrónico. El archivo .env lo creamos copiando el archivo .env.example y renombrando la copia como .env.

Por motivos de seguridad de la base de datos el archivo .env nunca se sube cuando hacemos un push en nuestro repositorio. Es por eso que aparece escrito dentro del archivo .gitignore en la raíz de nuestro proyecto.

Llegamos a la sección final, abordaremos el directorio mas importante en el cual pasaremos la mayor parte de nuestro tiempo.



Estructura del directorio de la aplicación (app)

app/commands

Este directorio contiene cualquier comando personalizado de *Artisan* que pueda necesitar la aplicación. *Artisan* no solo ofrece funcionalidad por defecto para ayudarnos a construir el proyecto, si no que también nos ofrece la oportunidad de crear nuestros propios comandos para hacer cosas.

app/config

La configuración tanto para el framework como para la aplicación se mantiene en este directorio. La configuración de *Laravel* existe como un conjunto de archivos PHP que contienen matrices clave-valor. Este directorio también contiene sub-directorios que permiten distintas configuraciones cargadas en diferentes entornos.

app/controllers

Como el nombre sugiere, este directorio contendrá los controladores. Los controladores pueden ser usados para facilitar lógica a la aplicación, y hacer de pegamento entre las partes separadas de la aplicación. Este directorio ha sido añadido al archivo `composer.json` por defecto para la auto carga de clases.

app/database

Este directorio será usado para contener los archivos que crearán el esquema de la base de datos, y los métodos para completarla con datos de ejemplo. La base de datos por defecto en *SQLite* también está ubicada en este directorio.

app/lang

El directorio lang contiene archivos PHP con matrices de cadenas que pueden ser usados para dar soporte de traducción a la aplicación. Se pueden crear sub carpetas por región para que tengamos distintos ficheros para múltiples idiomas.

app/models

Este directorio contendrá los modelos. Los modelos son usados para representar el modelo de negocio o facilitar interacción con el almacenamiento. *Laravel* trae un modelo *User* para ofrecernos autenticación en la aplicación por defecto. Al igual que el directorio *Controllers*, este ha sido añadido a la sección de carga automática del archivo `composer.json` por defecto.

app/start

Mientras que el directorio *bootstrap* contiene los procedimientos de arranque que pertenecen al *framework*, el directorio *start* contiene procedimientos de arranque que pertenecen a la aplicación. Como siempre, se ofrecen algunos por defecto.

app/storage

Cuando *Laravel* necesita escribir algo en el disco, lo hace en el directorio *storage*. Por este motivo el servidor web debe poder escribir en esta ubicación.

app/tests

El directorio *tests* contiene todas las pruebas unitarias y de aceptación para la aplicación. La configuración por defecto de *PHPUnit* que ha sido incluida por Laravel buscará pruebas en este directorio por defecto.

app/views

El directorio *views* es usado para contener las plantillas visuales de la aplicación. Se facilita una vista ‘hello’ por defecto para nuestra conveniencia.

app/filters.php

El archivo *filters.php* es usado para contener cualquier filtro de rutas de la aplicación.

app/routes.php

El archivo *routes* contiene todas las rutas de la aplicación.

Rápidamente surgen varias cuestiones. ¿Cómo es posible ordenar todos los archivos del lado del servidor en una misma carpeta manteniendo el framework sencillo?

¿Cuál es el punto de inicio y su funcionamiento? Aunque se puede ordenar libremente, la idea es separar la lógica del modelo de datos. En la carpeta App estará el archivo de rutas y todos los controladores. De hecho, su creador recomienda eliminar la clasificación en modelos y crear capas con una funcionalidad en concreto. Con esta idea, cada clase de una capa (ya sea un repositorio, un manejador, un filtro, un evento, una colección o, en general, una utilidad) debe tener una única responsabilidad. Esto favorece el seguimiento del patrón *DRY* (Don't repeat yourself) con soluciones a problemas comunes como el de repetir código que se va a utilizar en más ocasiones.

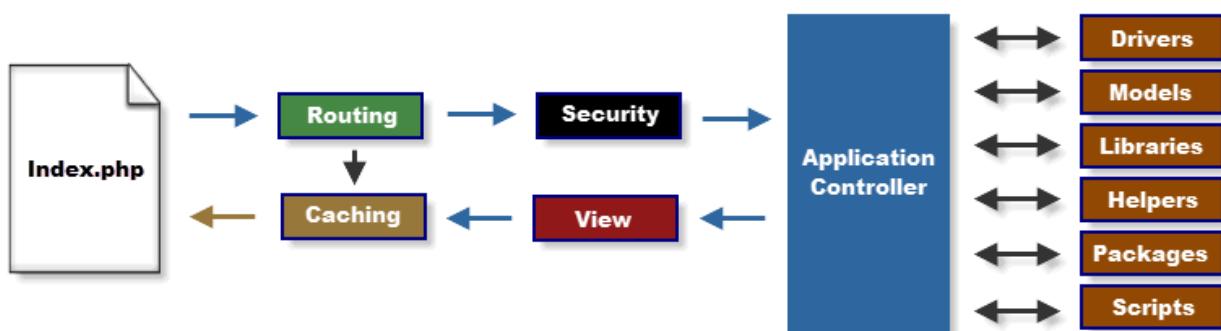


Figura 10. Patrón de diseño Front Controller

Cualquier petición del usuario se resuelve a través de las rutas definidas en *routes.php*.

En *Laravel* existe un único punto de acceso a la aplicación Web, encargado de manejar estas peticiones, incluyendo a los servicios de seguridad como la autenticación y autorización, con el objetivo de elegir una vista apropiada o devolver un error como resultado. El patrón *Front Controller* (Figura 11) sugiere la centralización del manejo de peticiones en un controlador, pero a diferencia de *Singleton*, no se limita el uso de varios controladores.

Laravel usa el estándar *PSR-4* para cargar todas las clases automáticamente. Para ello, es necesario que todas las clases de la aplicación usen al menos un espacio de nombre principal. Con esto, no es necesario usar composer dumpautoload cada vez que se cree una clase.

El patrón *Decorator* responde a la necesidad de agregar funcionalidad a un objeto por medio de la asociación de clases. En *Laravel*, se puede usar distintas combinaciones de decoraciones para generar distintas páginas web y evitar repetir código cuando se define una vista en las plantillas. De esta forma se construyen formatos específicos como la cabecera (header), el pie de página (footer), un menú, una tabla de usuarios, etc., y se pueden añadir a una vista que, por ejemplo, represente la página principal.

Laravel usa un motor de plantillas llamado *Blade*, un lenguaje muy sencillo, que antes de ser usado por la aplicación, es compilado a *PHP* plano. A la hora de crear las plantillas en la carpeta *app/views*, simplemente hay que añadir la extensión *.blade.php*.

Dentro de una plantilla se puede extender de otra vista que tenga una sección dinámica sin completar (@yield) y escribir el código que se va a sustituir (@section). Las etiquetas de *Blade* `{!! $var !!}` permiten escapar los datos automáticamente, como seguridad, para protegerse de los ataques XSS (Cross-site scripting).

Si no se desea escapar los datos, se puede usar `!! $var !!`.

En *storage/framework/views* se guardan los archivos temporales con el *HTML* final.

La inyección de dependencias de *Laravel* se usa en los métodos constructores (*__construct*), en los métodos de los controladores y en las rutas, de forma automática, por lo que no hay que preocuparse en pasar las clases al método. Se usa el patrón IoC (Inversión de Control) donde un contenedor se encarga de resolver todas las dependencias que usa el objeto a crear, de forma transparente al usuario y se devuelve una instancia de dicho objeto.

Para finalizar esta sección, se muestra la navegación entre las diferentes rutas de GestHotel.

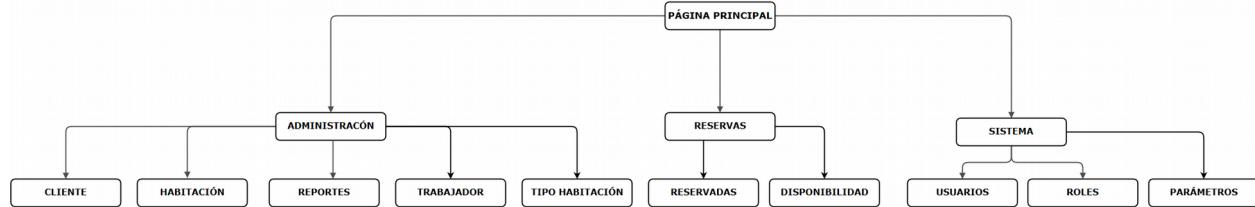


Figura 11. Diagrama de navegación de GestHotel

Desde la página principal se puede tener acceso a todas las vistas, dependiendo del rol con el que esté logueado el usuario.

Para una mejor organización, la aplicación se divide básicamente en tres grandes módulos: Reservas, Administración y Sistema.

Cada uno de ellos cuenta con prácticamente las mismas funcionalidades: una parte de administración (para crear, editar y en según que casos eliminar) y otra parte de consulta (para interactuar con los distintos listados que nos ofrece la aplicación).

6.3 Diseño de la base de datos

La aplicación GestHotel usa una base de datos relacional para almacenar la información de todos los usuarios, clientes, trabajadores y habitaciones registradas en el sistema, así como la disponibilidad, tipología y precios de las mismas (Figura 12).

Cada usuario/trabajador tendrá su perfil, donde se indicará si le corresponde el rol que ejercerá en el sistema.

De cada cliente se almacenaran sus datos y permanecerán en la base de datos para futuras reservas y/o consultas.

La base de datos acogerá el número de habitaciones con una serie de detalles entre los que estarán a que tipología pertenece y que precio le corresponde dependiendo del número de personas que ocupe la habitación.

Se almacenarán tanto las reservas por un lado, como el monto pendiente de las mismas por otro.

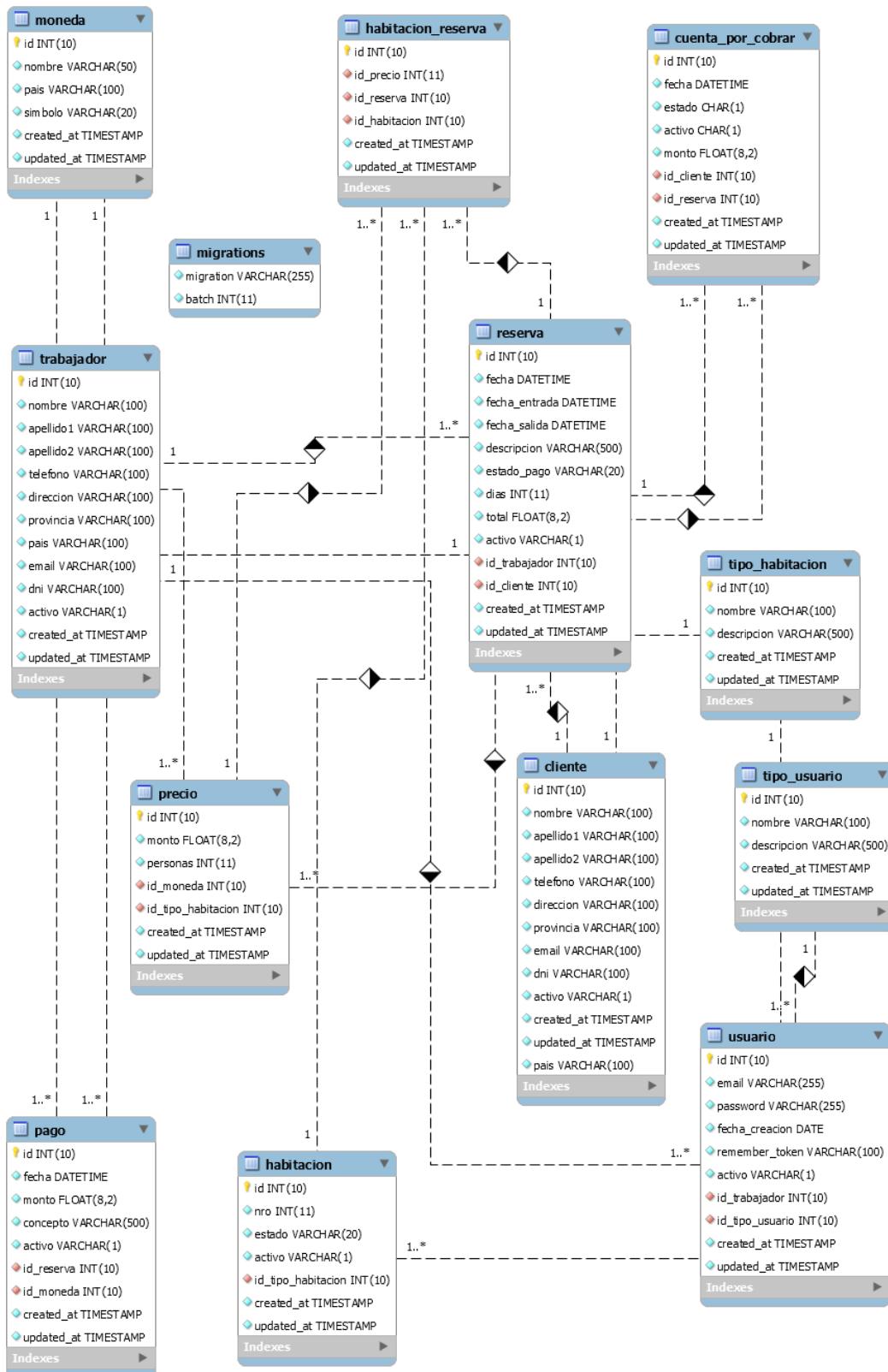


Figura 12. Modelo Entidad-Relación de GestHotel

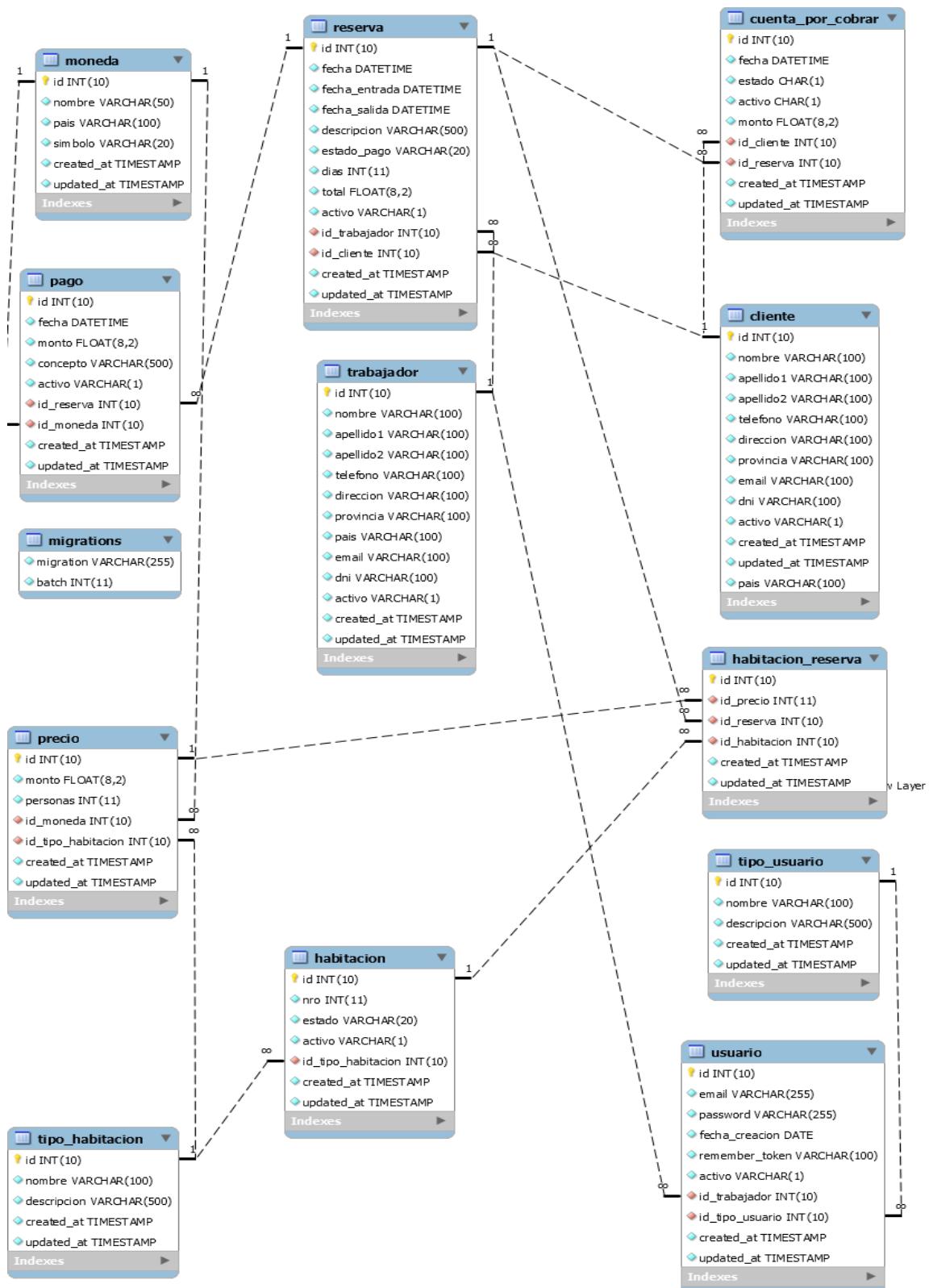


Figura 17. Esquema relacional de GestHotel

Laravel usa el patrón *Active Record* para la persistencia de los datos. Se trata de una clase que se encarga de implementar todas las operaciones de consulta y modificación de una tabla de la base de datos, que a diferencia de otros patrones como *DAO* (Data Access Object), va a aportar menor flexibilidad a cambio de un mayor aislamiento para trabajar con el sistema de gestión de la base de datos, en este caso *MySql*.

El patrón *Active Record* permite trabajar las tablas como si fueran clases y las filas como objetos. Para ello, *Laravel* cuenta con un potente *ORM* (Object-Relational mapping) llamado *Eloquent* y además tiene un constructor de consultas *SQL* (query builder) llamado *Fluent*.

También permite utilizar *SQL* puro para consultas más complicadas.

En primer lugar, *Fluent* es un constructor de consultas *SQL*, basado en *PDO*, encargado de generar cualquier consulta a la base de datos. Las consultas generadas vienen por defecto con los niveles de seguridad para evitar inyecciones *SQL* de usuarios malintencionados.

Las bases de datos relacionales son incompatibles con la forma de trabajar en la programación orientada a objetos, en la que los objetos se relacionan con otros a través de propiedades y métodos.

Eloquent permite mapear datos de la base de datos y convertirlos a objetos o, por el contrario, tomar un objeto y almacenarlo como un registro de la base de datos.

En el siguiente capítulo se explica la forma de usar estas herramientas y además se introduce el concepto de migraciones, para llevar un control de versiones de la base de datos, y seeders, para cargar información aleatoria en las tablas y poder probar la aplicación.

6.4 Diseño web adaptable

Tener un sitio especializado para cada uno de los dispositivos existentes puede llegar a ser muy complicado de mantener, ya que no se cubre la variedad de pantallas y dispositivos móviles disponibles hoy en día.

Google publicó recientemente que su algoritmo de búsqueda incluirá el factor de diseño *RWD* (Responsive Web Design) viéndose afectados todos los sitios que no sean adaptables.

Para lograr un diseño adaptable existen diferentes técnicas relacionadas con *HTML*, *CSS* y *JavaScript*.

El framework *Bootstrap* dispone de CSS *media queries* para escalar de manera fácil y eficiente los sitios web y aplicaciones, usando una base de código sencilla, válido desde teléfonos y tabletas a dispositivos de escritorio.

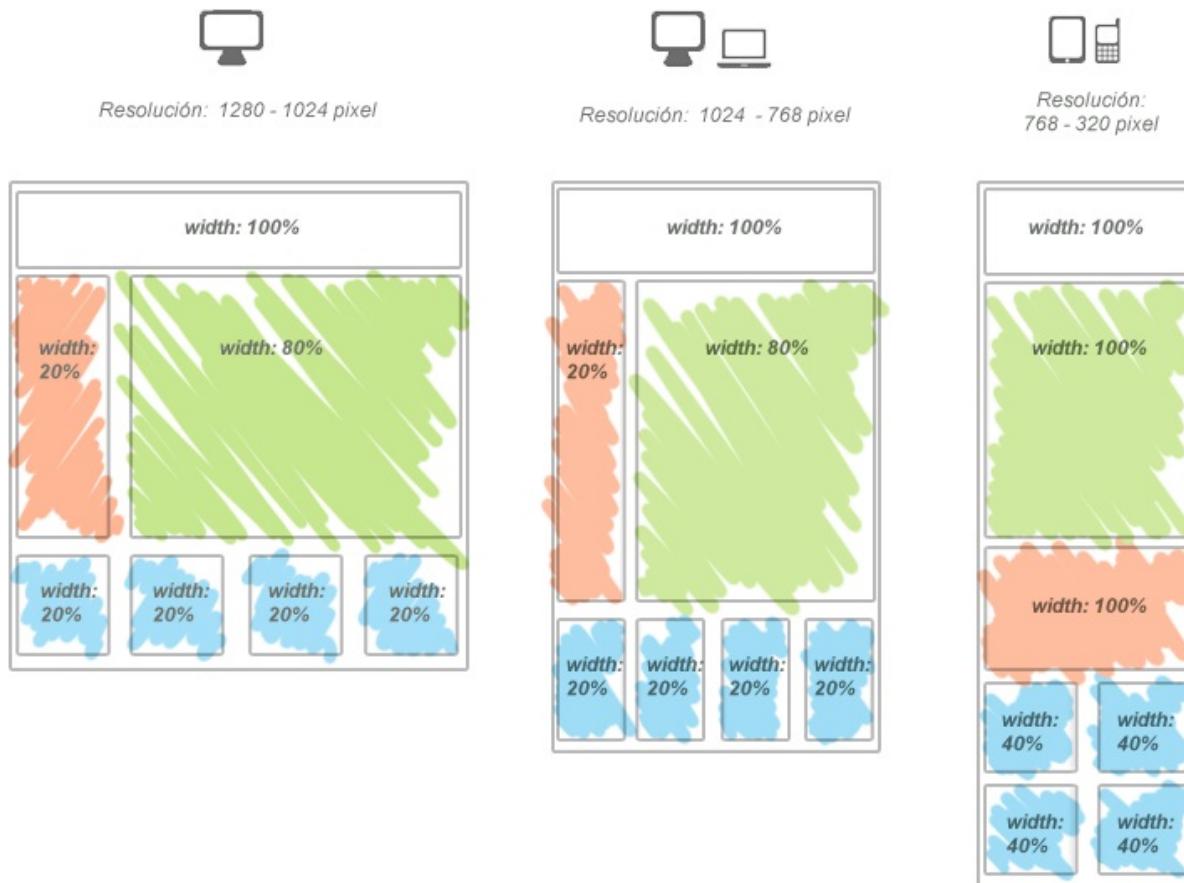


Figura 13. Diseño web adaptable

Incluyendo la regla: `<meta name="viewport" content="width=device-width, initial-scale=1">` se adapta el contenido de la página al ancho del dispositivo, permitiendo el zoom.

Es importante destacar que los píxeles que usa la etiqueta *viewport* son píxeles CSS, que pueden variar con respecto a los píxeles físicos del dispositivo. Esto es debido a la diferencia en la densidad de píxeles que tenga la pantalla. Existen herramientas para calcular la dimensión exacta en píxeles CSS.

El framework *Bootstrap* usa un sistema de rejillas para organizar todos los contenidos de la página, donde cada fila contiene doce columnas.

Las columnas (`div class="col-md-4"`) se declaran dentro de una fila (`div class="row"`) y cada fila tiene que estar dentro de un contenedor (`div class="container"`).

Los *media queries* son un método para detectar ciertas características del *viewport*, como son sus dimensiones (`width` y `height`), la resolución y la orientación del dispositivo (`portrait` o `landscape`).

Bootstrap aplica este método en sus hojas de estilo usando la regla `@media` seguido de las condiciones que debe cumplir el *viewport*, permitiendo mayor flexibilidad.

7 IMPLEMENTACIÓN

Antes de empezar a desarrollar la aplicación, es necesario explicar algunos conceptos que incluye *Laravel*.

Los principales elementos básicos de *Laravel* son *Composer*, *Artisan*, *Routing*, *Controllers*, *Requests*, *Responses* y *Views*. En relación con la base de datos hemos de explicar *Migrations*, *Seeders*, *Fluent* y *Eloquent*.

Posteriormente, se explican varios de los servicios que incluye el framework (autenticación, cache, colecciones, encriptación, eventos, internacionalización, mail, paginación, sesión, validación, motor de plantillas Blade, etc.), además de aquellos creados especialmente para cada uno de los módulos en los que se divide GestHotel.

7.1. Primeros pasos con Laravel

En los anexos de final del documento, se indican los manuales de instalación del aplicativo, necesarios para el despliegue de la aplicación GestHotel.

Composer

Durante la instalación del framework es necesario usar la herramienta *Composer*, un manejador de dependencias de *PHP*. Actualmente es una de las mejores herramientas que hay disponibles para el desarrollo con el lenguaje *PHP*. Permite manejar de una forma sencilla las dependencias del proyecto, ya sean del framework completo o de cualquier otro componente que instalemos. La herramienta se encarga de instalar todas las dependencias de manera recursiva, manteniendo los paquetes actualizados y de forma automática.



Figura 14. Manejador de dependencias Composer

Laravel usa el estándar *PSR-4* que permite a *Composer* cargar las clases automáticamente, ya que lo combina con los namespaces de *PHP* para ordenar y estructurar mejor el proyecto, por lo que, una vez creado el proyecto de *Laravel* con *Composer*, lo segundo que habrá que hacer es configurar *composer.json* para que los componentes usen esta especificación, donde la clave es el espacio de nombres y el valor es el nombre del directorio.

```
"autoload": {"psr-4": {"App\\": "app/"}}
```

Artisan

Artisan es el nombre de la linea de comandos incluida por *Laravel*. Proporciona un gran número de comandos para el programador mientras este está desarrollando su aplicación. Esta impulsado por el componente *Console* de *Symfony* y los métodos mas comunes se han dotado de una mayor fluidez para adaptarse al estilo de Laravel. Ofrece comandos muy útiles que pueden ayudar a realizar diversas tareas, tales como generar migraciones o publicar recursos de un paquete. Adicionalmente a los comandos que vienen por defecto, el desarrollador puede incluir los suyos propios.

Migraciones

Cuando se trabaja con bases de datos durante el desarrollo de una aplicación esta tiene muchas interacciones, se crean nuevas tablas, se renombran

columnas, se eliminan índices etc. En este área no podemos utilizar un sistema de control de control de versiones como haríamos en otros apartados del proyecto, aunque hay ciertas ocasiones que necesitamos maneras sofisticadas de tener controlada la trayectoria y los cambios que sufren nuestras bases de datos:

- Cuando en un equipo de desarrolladores, cada persona tiene que saber cualquier cambio en el esquema de la base de datos.
- Cuando se hace un *deploy* en un servidor de producción y se necesita una manera robusta de hacer una actualización de la base de datos.
- Al trabajar en diversas máquinas, es necesario tener todas las bases de datos sincronizadas.

Poder realizar esta sincronización de bases de datos puede llegar a ser un gran problema, ya que si por ejemplo se realizan cambios en una base de datos y el código que la acompaña tiene que volver a una versión anterior, nos aparece un gran problema al tener un sistema descompensado en cuanto a versiones de código y base de datos.

Las migraciones son la manera que tiene *Laravel* de ayudarnos a evolucionar el esquema de nuestra aplicación web sin tener que eliminar y recrear la base de datos cada vez que se realiza un cambio. Esto significa no perder los datos de la base de

datos, ya que, los únicos cambios que se realizan al ejecutar una migración son aquellos necesarios para mover el esquema de la base de datos de una versión a otra, tanto para adelante como para atrás en el tiempo. Se podría decir que Laravel consigue simular un sistema de control de versiones de las bases de datos.

Las migraciones son archivos que contienen una definición de clase con los métodos *up()* y *down()*.

El método *up()* se ejecuta cuando la migración se lanza para aplicar los cambios en la base de datos.

El método *down()* se ejecuta para revertir los cambios. Si por ejemplo necesitamos actualizar nuestra base de datos, solamente tenemos que crear una nueva migración y ya lo tenemos listo, pudiendo actualizar a los nuevos datos y volviendo para la versión anterior cuando veamos necesario.

De esta manera Laravel nos permite cambiar el esquema de nuestra base de datos utilizando PHP en vez de SQL. Esto permite, gracias al Schema Builder

de Laravel, crear tablas en la base de datos, insertar columnas o indices de una manera muy rápida.

Antes de comenzar a implementar cada uno de los módulos, hay que realizar una última configuración: la internacionalización de la aplicación.

Internacionalización

Laravel permite de una forma sencilla cambiar el idioma de los textos literales modificando una variable de localización dentro de *config/app.php*. Para cambiar su valor dinámicamente se usa: *App::setLocale('es')*. En la carpeta *app/lang* se guardará los archivos de los idiomas que contiene todas las traducciones posibles.

Tras esta breve introducción, se crean las primeras páginas de la aplicación.

Vistas y Blade.

Las vistas en Laravel son la parte pública que el usuario de nuestro sistema va a poder ver, se escriben en HTML junto con un motor de plantillas llamado *Blade*.

Las vistas se encuentran ubicadas en la carpeta *app/views/* y *Laravel* por defecto trabaja con la idea de que tenemos que escribir la menor cantidad de código repetido, modularizar nuestro código en donde mas se pueda, y si esto lo aplicamos en nuestros modelos, controladores y demás partes de nuestro proyecto.

Laravel usa unos archivos que se llaman plantillas o templates que suelen ser nuestros archivos principales, que tienen los segmentos de código que se repiten en mas de una vista, como por ejemplo la barra de navegación, el pie de página, un menú de opciones, la estructura del acomodo de nuestro proyecto, etc. y como deben de estar prácticamente presentes en todos lados, no tiene sentido estarlos repitiendo en todas las vistas.

Además de los *templates*, se cuentan con archivos que se llaman *partials*, estos archivos son pequeños segmentos de código que suelen ser usados comúnmente en partes del sistema en específico, como los formularios o secciones de mensajes, estos archivos surgen por el código que es mas pequeño que repetimos mucho pero no es lo suficientemente grande como para considerarlo un template.

Esto hace que las vistas de cada parte del proyecto, que suelen ser llamadas por una ruta o controlador sean mucho mas pequeñas que usando otro tipo de frameworks para desarrollo Web, y para poder unir todos estos archivos o piezas del rompecabezas usamos el motor de plantillas de *Laravel* llamado BLADE.

7.2 Página Principal

Lo primero ha sido crear una página maestra que contenga la cabecera, el pie de página y los scripts para integrar los frameworks *Bootstrap* y *jQuery*.

En nuestro caso se han definido dos páginas maestras que extenderán al resto de páginas de la aplicación (header.blade.php y footer.blade.php)

header.blade.php

Esta vista incluye tanto las llamadas a los estilos necesarios, como las llamadas a los script que necesitará la aplicación.

Además de implementar estas llamadas, se encarga de mostrar el menú lateral con los distintos módulos del panel de administración y de definir el enrutamiento de la parte dinámica a mostrar.

Para conseguir esto, se usan dos helpers bastante útiles en las plantillas de *Blade*: *url()* y *asset()* para conseguir los enlaces absolutos de las direcciones web y de los archivos de la carpeta pública. También es necesario instalar el componente de *Laravel Collective* para utilizar las etiquetas dinámicas de *HTML* y *FORM*, como por ejemplo *Html::style()* o *Form::text()*.

Su instalación se puede aplicar a cualquier otro componente disponible para Laravel.

En primer lugar, se añade en composer.json:

```
"require": {"laravelcollective/html": "5.1.*"}
```

Desde la terminal: *composer update*.

Se añaden en config/app.php los providers y aliases:

```
Collective\Html\HtmlServiceProvider::class,  
'Form' => 'Collective\Html\FormFacade',  
'Html' => 'Collective\Html\HtmlFacade',
```

Para cargar la página principal, por convención se crea un controlador que se encargue de su comportamiento.

```
php artisan make:controller HomeController
```

Se edita de tal forma que en la función index, la respuesta sea return view('home'), que es un helper de \View::make(plantilla_blade); y luego se añade la ruta en el archivo: app/routes.php

```
Route::get('/', ['as' => 'home', 'uses' => 'HomeController@index']);
```

De esta forma, se indica que cuando se solicite <http://miapp.com> se está accediendo con método GET cuyo alias de la ruta es 'home' y se ejecuta el método 'index' por defecto.

Pero nuestra aplicación va a diferir un poco en cuanto al acceso de la página principal, esto es debido a que el módulo de *Login* es totalmente personalizado y no se usa el que trae Laravel por defecto.

Veremos como implementar la carga de la página principal cuando hablaremos de los de los distintos módulos.

En principio nos vamos a quedar con la idea de como las rutas hacen las llamadas al controlador y desde este ese devuelve la vista.

footer.blade.php

Esta es una vista sencilla, simplemente se encarga de mostrar un pie de página estático.

index.blade.php

Esta será la página de inicio de nuestro sistema, se compone de una serie de accesos rápidos a distintas partes de los módulos de la aplicación y es en este punto donde se va explicar como extienden tanto la cabecera, como el pie a todas las páginas de la aplicación:

Anteriormente hablábamos de *templates* y *partials*, describiremos un poco de como se trabaja con estas estructuras de Blade y sus beneficios:

Templates: Estos archivos como se menciona al principio del capítulo son plantillas que nos ahorran mucho código o lenguaje HTML, y para usar un template se usa la sentencia:

```
@extends('template')
```

Claramente se tendría que sustituir la palabra template dentro de la sentencia extends por el nombre de la vista que va a funcionar como template o plantilla.

Un template es una vista como las demás, simplemente que dentro de ella se usan otras sentencias que nos va a permitir definir áreas del archivo que se van a poder sustituir mas adelante dentro de otra vista si es que lo deseamos.

Para esto se ocupa la sentencia:

```
@yield('nombre_seccion')
```

Para declarar una sección que se va a llenar en otro lugar:

```
@section('nombre_seccion')
```

que funciona de la misma forma que yield() con la diferencia que en la sección podemos definir HTML por defecto en caso de no definir la sección con un nuevo HTML.

Definiremos nuestra vista index.blade.php para que use dos templates:

```
@extends('header')
```

```
@extends('footer')
```

El template header tiene definida una @section llamada content y este a su vez un yield llamado content que significa contenido, por lo cual se añadirá a la vista el contenido dinámico de cada módulo.

Vemos a continuación el código de como se estructura la página principal y el uso de los templates, content y yield

header.blade.php

```
1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5      <meta charset="utf-8">
6      <meta http-equiv="X-UA-Compatible" content="IE=edge">
7      <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
8      <meta name="description" content="">
9      <meta name="author" content="">
10     <title>SISTEMA HOTEL</title>
11     | <!-- Estilos de Bootstrap-->
12     {{ HTML::style('css/Bootstrap-3/css/bootstrap.css') }}
13     {{ HTML::style('css/Bootstrap-4/bootstrap.css') }}
14     {{ ... }}
15
16 </head>  <!-- end head-->
17
18
19 <body class="fixed-nav sticky-footer bg-dark" id="page-top">
20 <!-- Navigation-->
21 <nav class="navbar navbar-expand-lg navbar-dark bg-dark fixed-top" id="mainNav">...
22 </nav>
23 <!-- Incluye el content-->
24 @include('content')
25 <!-- Incluye el Footer-->
26 @include('footer')
27
28     | <!-- Bootstrap 3 core JavaScript-->
29     {{ HTML::script('js/Bootstrap/bootstrap3/bootstrap.min.js') }}
30     | <!-- Plugin para crear etiquetas con JavaScript -->
31     {{ HTML::script('js/Plugins/popper/popper.min.js') }}
32     | <!-- Libreria Jquery -->
33     {{ ... }}
34
35 </body>
36
37 </html>
```

footer.blade.php

```
1  <!--footer-->
2
3 ▼<div class="container-fluid">
4 ▼  <footer class="sticky-footer">
5 ▼    <div class="container">
6      <div class="text-center">
7        <small>Copyright © Todos los Derechos Reservados 2017</small>
8      </div>
9    </div>
10   </footer>
11   @include('modal')
12   <!-- Scroll to Top Button-->
13   <a class="scroll-to-top rounded" href="#page-top">
14     <i class="fa fa-angle-up"></i>
15   </a>
16 </div>
17
18
19
```

content.blade.php

```
1
2
3 |<div class="content-wrapper" id="wrap" >
4   <div class="container-fluid">
5     @yield('content')
6   </div>
7 </div>
8
9
```

index.blade.php

```
1  @extends('header')
2
3  @section('content')
4
5      <section class="content-header">
6          <h1>
7              Acceso Rápido.
8          </h1><br>
9      </section>
10
11     <section class="content">
12
13         <!-- ===== HOME DEL DASHBOARD ===== -->
14
15         <!-- Small boxes (Stat box) -->
16         <div class="row">...
17         </div>
18         <!-- /.row -->
19
20
21         <!-- ===== ACCESOS DIRECTOS ===== -->
22
23         <div class="row">...
24         </div>
25         <!-- /.row -->
26     </section>
27
28     @stop
```

Ahora nuestra vista ya no tiene el encabezado *HTML* normal ni las etiquetas `<body>` ni `<html>`, sino que estamos diciendo que vamos a extender del *template header* y que el *yield content* lo vamos a sustituir por nuestro propio contenido, cabe mencionar que aunque en el *template* se uso la sentencia `yield('content')`, al momento de sustituirla la vamos a cambiar por `section('content')`, por lo cual en todas las vistas hijas del *template* solo se va a definir secciones y el fin de esa sección se va a declarar con la sentencia `@stop`.

7.3 Módulo Usuarios y Roles.

Antes de crear un sistema de autenticación seguro, es necesario crear las tablas en la base de datos. Esto se puede lograr fácilmente a través de migraciones, que llevan un control de versiones de la base de datos GestHotel.

Cuando se instala el sistema de migraciones de *Laravel*, desde consola: `php artisan migrate:install`, se crea una tabla *migrations* que verifica las migraciones que ya se han realizado y no hace falta volver a ejecutar.

Para crear una migración: `php artisan make:migration create_usuario_table--create="usuario"`. Este archivo contiene dos métodos, donde se puede hacer uso de Schema Builder para crear la tabla usuarios.

Cuando se trabaja en local y se hace algún cambio en la tabla (se añade un nuevo atributo por ejemplo), es recomendable usar: `php artisan migrate:rollback` para eliminar la tabla por el método DOWN y crear una nueva con el método UP. Si por el contrario, ya se ha subido el proyecto a un repositorio o al servidor compartido, se puede modificar la tabla ya creada, usando las nuevas migraciones `php artisan migrate`.

De una forma más rápida, se puede usar: `php artisan migrate:refresh --seed`, que resetea la base de datos, crea de nuevo las migraciones y ejecuta los seeders para llenar los datos.

Si no existe un tipo de datos en *Schema Builder* al cambiar de un gestor de base de datos a otro, *Laravel* lo sustituye por el tipo de datos más parecido que exista, lo que supone una gran ventaja ya que no hace falta crear nuevas instrucciones de actualización, simplemente con ejecutar el sistema de migraciones, *Laravel* crea las tablas correspondientes.

Una vez creadas las tablas de usuario y perfil de usuario por medio de las migraciones, se crea el formulario *login.blade.php* necesario para el sistema de autenticación de la aplicación.

login.blade.php

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>AUTENTICACION</title>
5      <meta charset="UTF-8">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      {{ HTML::style('css/Bootstrap-3/css/bootstrap.min.css') }}
8      {{ HTML::style('css/Bootstrap-3/css/bootstrap-theme.min.css') }}
9      {{ HTML::style('css/Custom/layout.css') }}
10     {{ HTML::script('js/Bootstrap/bootstrap3/bootstrap.min.js') }}
11     {{ HTML::script('js/main.min.js') }}
12 </head>
13 <body>
14 <div aria-hidden="true" role="dialog" tabindex="-1" class="modal show" id="loginModal">
15     <div class="modal-dialog">
16         <div class="modal-content">
17             <div class="modal-header">
18                 <h1 class="text-center">Login</h1>
19             </div>
20             <div class="modal-body">
21                 {{ Form::open(array('url' => 'login','class'=>'form col-md-12 center-block')) }}
22                 <div class="form-group">
23                     <!--{{Form::label('email', 'Email',[ 'class'=>'col-sm-3 control-label'])}}-->
24                     {{ Form::text('email','','[ 'class'=>'form-control','placeholder'=>'Email'])}}<br>
25                 </div>
26                 <div class="form-group">
27                     <!--{{Form::label('password', 'Contraseña',[ 'class'=>'col-sm-3 control-label'])}}-->
28                     {{ Form::password('password',[ 'class'=>'form-control','placeholder'=>'Password'])}}<br>
29                 </div>
30                 <div class="form-group">
31                     {{ Form::submit('Entrar',[ 'class'=>'btn btn-primary btn-block'])}}
32                 </div>
33                 {{ Form::close() }}
34             </div>
35         </div>
36     </div>
37 </div>
38 </body>
39 </html>
40
41
42
```

Como hemos comentado anteriormente en lugar de usar el sistema clásico de autenticación de Laravel, se ha diseñado un sistema personalizado, uno de los motivos es que el sistema por defecto, solo admite dos tipos de usuarios, un usuario admin y otro tipo usuario sin privilegios, como ya sabemos , nuestro sistema va a disponer en principio de tres roles y será administrador de sistemas quien podrá añadir mas roles si fuera necesario.

Para solucionar este problema, ademas de la tabla usuarios, se ha añadido una nueva: tipo_usuario que guardará los distintos roles.

Con este nuevo sistema de autenticación, en lugar de llamar directamente al *HomeController* desde el sistema de rutas para que nos devuelva la vista, en dicho sistema de rutas añadimos:

```
/*Llamadas al controlador Auth*/
Route::get('login', 'AuthController@showLogin'); // Mostrar login
Route::get('/', 'AuthController@showLogin'); // Mostrar login
Route::post('login', 'AuthController@postLogin'); // Verificar datos
Route::get('logout', 'AuthController@logOut'); // Finalizar sesión
```

Nuestro controlador queda de la siguiente forma:

```
1  <?php
2
3  class AuthController extends \BaseController {
4
5      public function showLogin()
6      {
7          // Verificamos si hay sesión activa
8          if (\Auth::check())
9          {
10              // Si tenemos sesión activa mostrará la página de inicio
11              return Redirect::to('index');
12          }
13          // Si no hay sesión activa mostramos el formulario
14          return View::make('Login.index');
15      }
16
17      public function postLogin()
18      {
19          // Obtenemos los datos del formulario
20          $data = [
21              'email' => Input::get('email'),
22              'password' => Input::get('password')
23          ];
24
25          // Verificamos los datos
26          if (\Auth::attempt($data, Input::get('remember'))) // Como segundo parámetro pasamos el checkbox para saber si queremos recordar la contraseña
27          {
28              // Si nuestros datos son correctos mostramos la página de inicio
29              return Redirect::intended('index');
30          }
31          // Si los datos no son los correctos volvemos al login y mostramos un error
32          return Redirect::back()->with('error_message', 'Invalid data')->withInput();
33      }
34
35      public function logOut()
36      {
37          // Cerramos la sesión
38          \Auth::logout();
39          // Volvemos al login y mostramos un mensaje indicando que se cerró la sesión
40          return Redirect::to('login')->with('error_message', 'Logged out correctly');
41      }
42
43 }
```

Como observamos, ya no será desde el *HomeController* desde donde nos redirigirá a la página principal, sino, será desde el *AuthController* el cuál mostrara la vista del login y en caso de validar correctamente los datos, nos mostrará la vista principal.

Cuando se crea un usuario en el sistema, el método del controlador usa *Eloquent* para almacenar los datos recibidos del formulario. Por ello, la siguiente tarea es crear los modelos de usuario, en este caso *Usuario* y *TipoUsuario*. Se sigue la notación CamelCase para todos los modelos y *snake_case* para los nombres de las tablas (por ejemplo, si se llama a la tabla: *usuario* y al modelo: *Usuario*, no hace falta indicar más datos al ORM). En el directorio *app* se irán guardando todos los modelos creados con *php artisan make:model Usuario*.

En el siguiente punto vamos a explicar con más detalles como se crean las migraciones y editarlas para llenar la base de datos, así como la creación de los modelos, rutas y controladores de cada módulo.

7.4. Módulos de Gestión de Clientes, Trabajadores, Reservas y Habitaciones.

La implementación de estos módulos es similar, lo que tienen en común es que cada uno ellos en si mismo se comportan como un sistema CRUD, donde vamos a crear, listar, editar y en su caso eliminar elementos de cada módulo.

Aunque hemos explicado el sistema de migraciones, modelos, rutas y controladores en el punto anterior, lo hicimos de una forma un poco generalizada y es conveniente detenerse un poco para detallar estas estructuras dentro de nuestra aplicación.

Básicamente, cada parte del *panel de administración*, se implementa como una aplicación independiente, aunque en sí unos dependan de otros, con esto, quiero decir que cada módulo, necesita de implementar su *migración*, *seeder* (opcional), *ruta*, *plantilla* (vista), *modelo* y *controlador*. Con estas cinco implementaciones podemos crear una simple aplicación con Laravel y veremos que sencillo es.

Usaremos como guía el *Módulo Clientes* y todos los demás módulos como decimos son similares.

7.4.1 Migraciones

Cuando creamos nuestras bases de datos solemos crear diagramas que nos facilitan la abstracción de como se va a almacenar nuestra información, pero la forma de llevarlo a la realidad en algún gestor de bases de datos, como por ejemplo: *MySQL*, *SQLite*, *PostgreSQL*, *SQL Server*, etc., lo más común es meternos al lenguaje de script encargado de implementar nuestra idea de la BD y ejecutar dicho script, o incluso ocupar programas más avanzados que nos sirven como interfaz para crearlas de una forma más gráfica y sin la necesidad de profundizar demasiado en el lenguaje, como *Workbench* o *Navicat*.

En *Laravel* se lleva a otro contexto esta situación, puesto que visto de la forma tradicional si se requieren cambios en la base de datos tenemos que meternos ya sea a otro programa para cambiar el diagrama de la base o a un archivo SQL con una sintaxis usualmente complicada o difícil de leer y ejecutar los cambios para reflejarlos en el proyecto, sin embargo, con esto no contamos con un control de los cambios (control de versiones) sobre la base de datos, si necesitamos consultar un cambio anterior o de repente la solución previa o inicial era la que se necesita al momento debemos re-escribir todo otra vez, cosa que con la migraciones se soluciona instantáneamente.

Las migraciones son archivos que se encuentran en la ruta app/database/migrations/ de nuestro proyecto Laravel, por defecto en la instalación de Laravel se encuentran dos migraciones ya creadas, create_users_table y create_password_resets_table.

Para crear nuestra migración usaremos el siguiente comando:

```
php artisan make:migration crear_cliente
```

el cual nos dará este resultado:

```
Created Migration: 2017_10_12_103417_create_cliente
```

nos agrega una plantilla de trabajo en app/database/migrations/ para empezar a trabajar.

2017_10_12_103417_create_cliente.php

```
1 <?php
2
3 use Illuminate\Database\Schema\Blueprint;
4 use Illuminate\Database\Migrations\Migration;
5
6 class CreateCliente extends Migration {
7
8     /**
9      * Run the migrations.
10     *
11     * @return void
12     */
13    public function up()
14    {
15        Schema::create('cliente', function(Blueprint $table) {
16            $table->increments('id');
17            $table->timestamps();
18        });
19    }
20
21    /**
22     * Reverse the migrations.
23     *
24     * @return void
25     */
26    public function down()
27    {
28        Schema::dropIfExists('cliente');
29    }
30
31 }
32
```

Ahora bien se puede observar que el archivo como tal no se llama simplemente `crear_cliente` sino `2017_10_12_103417_crear_cliente`, esto pasa porque Laravel al crear una migración agrega como prefijo la fecha y hora en la que fue creada la migración para poder ordenar qué migración va antes que otra.

Dentro de la estructura del archivo podemos ver dos funciones, una llamada `up()` y otra llamada `down()`, la primer función es en donde vamos a especificar la estructura de nuestra tabla, inicialmente y gracias al comando se encuentran ya algunas cosas escritas como lo son la clase `Schema` en la cual se llama al método `create`, el cual nos permite crear la tabla en nuestra base de datos, esta recibe dos parámetros, el primero es el nombre que va a recibir la tabla que es el que se le dio en el comando y por lo cual ya se encuentra en su lugar, y el segundo parámetro es una función `closure` o función anónima que lo que hace es definir las columnas de nuestra tabla, a su vez esta función anónima recibe como parámetro un objeto de tipo `Blueprint` que se agregó dentro del namespace con la palabra `use` en la cabecera del archivo, el objeto `$table` es con el que vamos a trabajar para definir los campos, como se ve en el código anterior esto se logra escribiendo `$table->tipo_dato('nombre');`, y esto puede variar dependiendo del tipo de dato que se use y para ello podemos revisar la documentación oficial de Laravel, para poder ver todos los tipos de campos con los que contamos.

En el código observamos que ya tenemos el campo 'id' de tipo `increments` que es equivalente a un campo en SQL así:

```
create table cliente (id int auto_increment);
```

Y un campo de tipo `timestamps` sin nombre, el efecto que tendrá será agregar dos columnas muy útiles que son `created_at` y `updated_at` que son campos que se usan para (como su nombre lo dice) guardar el registro de cuando fue creado y cuando fue actualizado el registro, detalles muy importantes cuando queremos obtener informes con base en el tiempo de la información de nuestra tabla, por ejemplo si quisieramos saber cuales son los clientes que se dieron de alta en el sistema en el mes de abril podríamos crear un filtro para obtener solo los clientes de ese mes usando el campo generado `created_at`.

Siguiendo la nomenclatura `$table->tipo_dato('nombre');` añadimos el resto de columnas, nombre, apellido1, apellido2, teléfono, dirección, email y dni a la tabla

Ahora bien si la función `up` crea nuestra tabla en la base de datos, la función `down` lógicamente hace lo opuesto, y eso es eliminar la tabla de la base de datos, por eso dentro de esta función podemos observar que de la misma clase `Schema` se llama al método `drop` que significa dejar caer o dar de baja.

Ahora el archivo resultante quedaría así:

```
1 <?php
2
3 use Illuminate\Database\Schema\Blueprint;
4 use Illuminate\Database\Migrations\Migration;
5
6 class CrearCliente extends Migration {
7
8     /**
9      * Run the migrations.
10     *
11     * @return void
12     */
13    public function up()
14    {
15        Schema::create('cliente', function(Blueprint $table) {
16            $table->increments('id');
17            $table->string('nombre', 100);
18            $table->string('apellido1', 100);
19            $table->string('apellido2', 100);
20            $table->string('telefono', 100);
21            $table->string('direccion', 100);
22            $table->string('email', 100);
23            $table->string('dni', 100);
24            $table->string('activo', 1);
25            $table->timestamps();
26        });
27    }
28
29    /**
30     * Reverse the migrations.
31     *
32     * @return void
33     */
34    public function down()
35    {
36        Schema::dropIfExists('cliente');
37    }
38
39 }
40
```

Si bien cada función realiza una tarea en específico, ¿Cuando es que se usan? o ¿Como se mandan a llamar?.

Para esto iremos nuevamente a nuestra línea de comandos.

Para correr o iniciar nuestras migraciones usamos el comando:

```
php artisan migrate
```

Con esto, si es la primera vez que se ejecuta este comando se creará en nuestra base de datos la tabla migrations que es la encargada de llevar el control de que migraciones que ya han sido ejecutadas, con el fin de no correr el mismo archivo más de una vez si el comando se usa nuevamente, además, se agregará la tabla cliente y en la tabla migrations se añadirá el registro de la migración recién ejecutada.

Pero, ¿si quisiera eliminar la tabla con la función down de la migración crear_cliente?

Esto se puede resolver de dos formas básicamente:

Con el comando `php artisan migrate:rollback` que lo que hará es deshacer la última migración ejecutada y registrada en la base de datos.

Con el comando `php artisan migrate:reset` que lo que hará es deshacer todas las migraciones de la base de datos.

En el dado caso que necesitáramos agregar más campos a la tabla cliente, podríamos simplemente ir a la migración crear_cliente y en la función up poner la nueva columna, pero con esto perderíamos la primer versión de la tabla, entonces para poder exemplificar como se agregan columnas con las migraciones crearemos una nueva que se llame agregar_campos_cliente con los comandos que ya hemos visto.

7.4.2 Seeders

Los Seeders por otra parte son archivos que nos van a permitir poblar nuestra base de datos para no tener que perder el tiempo escribiendo de forma manual todos los datos, un ejemplo, imaginemos llenar 15 tablas con 100 registros cada una y pensemos en que entre cada tabla deben existir registros que se relacionan entre sí, eso suena de verdad horrible y tedioso, por lo cual Laravel nos salva con estos archivos Seeders.

Un Seeder se ubica en la carpeta `app/database/seeds/` de nuestro proyecto de Laravel y para poder crear el nuevo Seeder usaremos el comando:

```
php artisan make:seeder ClienteTableSeeder
```

Esto nos creará un archivo en la carpeta `database/seeds/` que tendrá el nombre que le demos en el comando .

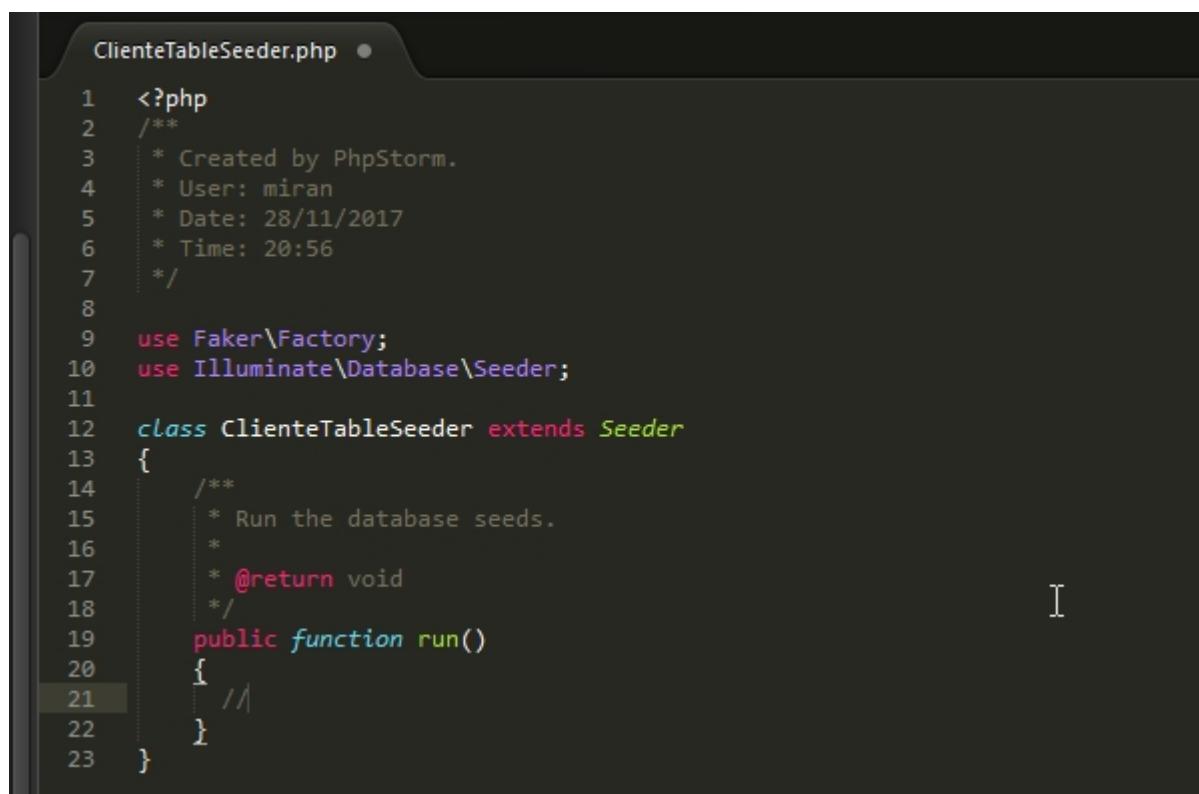
Con esto ya tenemos el archivo pero no es todo lo que necesitamos para poder trabajar con datos auto generados, para ello usaremos un componente llamado *Faker* el cual se encargará de generar estos datos, para instalar este componente, basta con ir al archivo `composer.json` y agregar en el "require-dev" las dependencias y para tener una idea más clara podemos ir a la página de [Packagist](#) donde se encuentra Faker o a su [Repositorio](#) en Github y ahí nos muestra que es lo que se debe agregar.

Al final solo se ocupa el comando `composer update` para actualizar las dependencias y descargar Faker al proyecto.

Una vez ya teniendo Faker iremos a nuestro archivo `ClienteTableSeeder` y dentro podremos observar que se encuentra una función llamada `run()` que es donde vamos a usar Faker para poblar, ahora bien antes de todo debemos agregar la clase de Faker a nuestro Seeder, para esto agregamos al inicio del archivo la linea:

```
use Faker\Factory as Faker;
```

Quedando el archivo de la siguiente forma:



```
ClienteTableSeeder.php ●

1 <?php
2 /**
3  * Created by PhpStorm.
4  * User: miran
5  * Date: 28/11/2017
6  * Time: 20:56
7 */
8
9 use Faker\Factory;
10 use Illuminate\Database\Seeder;
11
12 class ClienteTableSeeder extends Seeder
13 {
14     /**
15      * Run the database seeds.
16      *
17      * @return void
18      */
19     public function run()
20     {
21         //|
22     }
23 }
```

Después creamos una variable llamada \$faker que nos servirá para poblar la base de datos, ahora bien usando la clase DB, si bien queremos crear 40 clientes vamos a crear un for para que ejecute nuestro código de inserción 40 veces y el componente de Faker en cada pasada cambiará los valores del registro que se va a agregar, quedando de esta forma:

```
ClienteTableSeeder.php ×
1  <?php
2  /**
3   * Created by PhpStorm.
4   * User: miran
5   * Date: 28/11/2017
6   * Time: 20:56
7   */
8
9  use Faker\Factory;
10 use Illuminate\Database\Seeder;
11
12 class ClienteTableSeeder extends Seeder
13 {
14     /**
15      * Run the database seeds.
16      *
17      * @return void
18      */
19     public function run()
20     {
21         $faker = Factory::create('es_ES');
22
23         for($i=0;$i<40;$i++){
24             Cliente::create([
25                 'nombre' => $faker->firstName($gender = null|'male'|'female') ,
26                 'apellido1' => $faker->lastName,
27                 'apellido2' => $faker->lastName,
28                 'dni' => $faker->dni,
29                 'telefono' => $faker->numberBetween($min = 600000000, $max = 700000000),
30                 'email' => $faker->email,
31                 'activo' => 1,
32                 'direccion' =>$faker->address,
33                 'provincia' => $faker->state,
34                 'pais' => 'España'
35             ]);
36         }
37     }
38 }
```

Creamos nuestro objeto Faker, el cual puede generar información falsa para pruebas en nuestra base de datos y ahora usamos la clase DB el método table para llamar la tabla donde se va a insertar la información y se le concatena el método create() el cual recibe por parámetro un arreglo clave => valor con los campos de la tabla.

Faker tiene muchas variedades de datos, los cuales podemos consultar en su [Repositorio](#) de Github así como su uso básico.

Hemos usado propiedades como: firstName, lastName, dni, etc para darle nombre, apellidos, N.^º de dni... a los clientes de forma aleatoria.

Pero esto es solo nos sirve para casos de pruebas, seeders también nos provee de métodos para llenar las tablas con datos reales asignados por nosotros mismos, en tal caso, simplemente desaparecería el ciclo for y los pares clave, valor quedarían por ejemplo de la siguiente forma:

```
27 //clase para insertar usuarios
28 class UserTableSeeder extends Seeder {
29
30     public function run()
31     {
32
33         DB::table('users')->insert(array(
34             'username' => 'unodepiera',
35             'password' => Hash::make('123456'),
36             'email' => 'unodepiera@uno-de-piera.com',
37             'edad' => 32
38         ));
39
40         DB::table('users')->insert(array(
41             'username' => 'juan',
42             'password' => Hash::make('123456'),
43             'email' => 'juan@mail.com',
44             'edad' => 16
45         ));
46     }
47 }
```

Y ahora lo que sigue es abrir un archivo llamado *DatabaseSeeder.php*, en este archivo se mandan a llamar todos los seeders en el orden que los necesitemos, en este archivo se agregará la linea:

```
$this→call('ClienteTableSeeder::class');
```

que en si mandará a llamar nuestro seeder recién creado y para ejecutar este archivo se usa el comando:

```
php artisan db:seed
```

Y con esto queda poblada la tabla *Clientes* y lo podemos verificar en el gestor de base de datos.

Cuando trabajamos con *Migraciones* y *Seeder* por primera vez puede parecer un poco más complejo que a lo que estamos acostumbrados pero las ventajas que nos da superan por mucho a la forma convencional, además de ser una forma más profesional de trabajar.

7.4.3 Modelo

En Laravel podemos hacer uso de un ORM llamado Eloquent, un ORM es un *Mapeo Objeto-Relacional* por sus siglas en inglés (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.

Eloquent hace uso de los Modelos para recibir o enviar la información a la base de datos, para esto analizaremos el modelo que viene por defecto en Laravel, este es el modelo User que se ubica en la carpeta app/models, los modelos hacen uso de PSR-4 y namespaces, un modelo nos ayuda a definir que tabla, atributos se pueden llenar y que otros se deben mantener ocultos.

Los modelos usan convenciones para que a Laravel se le facilite el trabajo y nos ahorre tanto líneas de código como tiempo para relacionar más modelos, las cuales son:

- El nombre de los modelos se escribe en singular, en contraste con las tablas de la BD que se escriben en plural.
- Usan notación UpperCamelCase para sus nombres.

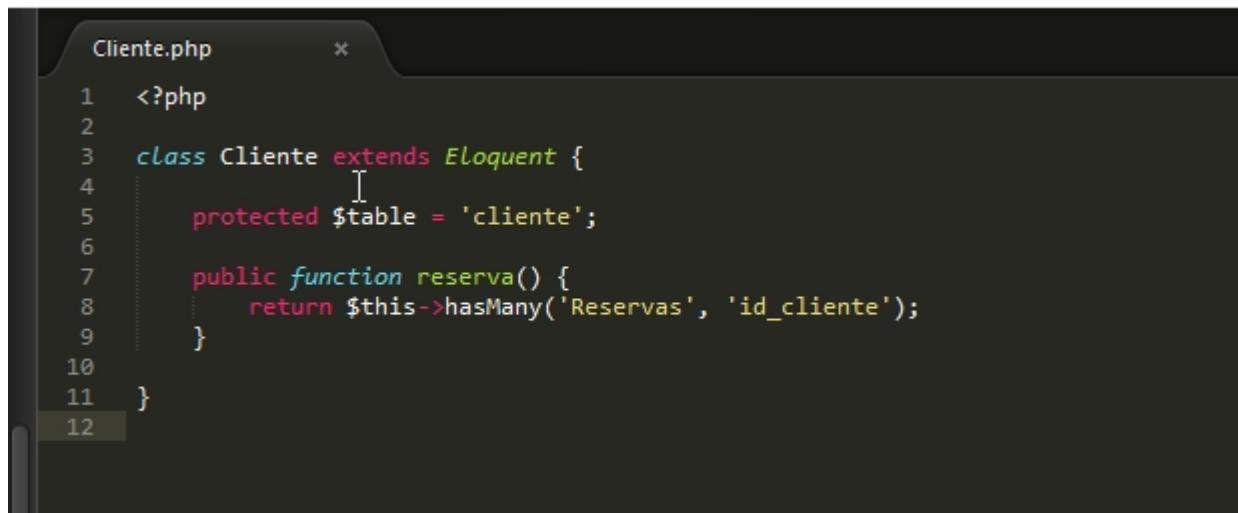
Estas convenciones nos ayudan a detectar automáticamente las tablas, por ejemplo: el modelo User se encuentra en singular y con notación UpperCamelCase y para Laravel poder definir que tabla es la que esta ligada a este modelo le es suficiente con realizar la conversión a notación underscore y plural, dando como resultado la tabla: users.

Y esto aplica para cuando queremos crear nuestros modelos, si tenemos una tabla en la base de datos con la que queremos trabajar que se llama habitación_reserva, vemos que se encuentra con las convenciones para tablas de bases de datos (plural y underscore), entonces el modelo para esta tabla cambiando las convenciones sería: HabitacionReserva (singular y UpperCamelCase).

Retomando el caso que estamos siguiendo sobre el módulo Cliente, crearemos ahora un modelo para poder trabajar con esa tabla, el cual recibirá el nombre de Cliente y el comando para poder crear nuestro modelos es:

```
php artisan make:model Cliente
```

Con esto se generara el archivo Cliente.php donde ya se encuentra el modelo **User** en la carpeta app/models y dentro de él vamos a definir tantas relaciones hubiese con el resto de tablas:



```
Cliente.php
1 <?php
2
3 class Cliente extends Eloquent {
4     [
5     protected $table = 'cliente';
6
7     public function reserva() {
8         return $this->hasMany('Reservas', 'id_cliente');
9     }
10
11 }
12
```

7.4.4 Ruta

Cuando ingresamos a una url directamente desde el navegador lo hacemos mediante una petición http de tipo GET, esta solicitud se envía al archivo *routes.php* ubicado dentro de *app/routes.php*, en caso de no existir nos dará un error, si la ruta existe, nos llevará a un controlador en el cuál se encuentra la lógica , el controlador interaccionará con un modelo para recuperar información de una base de datos. Esta información llega al controlador y desde el controlador invocamos una vista, las vistas se encuentran en el directorio *app/views*, finalmente la vista se carga y se muestra en el navegador.

Así es como funciona el modelo MVC (Model-View-Controller).

Supongamos que queremos ingresar a la siguiente URL <http://dominio.com/saludo> y desplegar una página con el mensaje “Bienvenido”. En laravel la porción /saludo pertenecería a una ruta que regresa una respuesta o una vista dependiendo lo complejo que llegue a ser lo que queramos mostrar. La parte de dominio.com pertenecería a localhost ya que lo andamos corriendo de manera local. En nuestro ejemplo lo que mostraremos es un mensaje muy simple por lo cual no es necesario hacer mostrar una vista. Para lograrlo haremos lo siguiente:

```
109 //pagina principal
110
111
112 Route::get('index', function() {
113     if (Auth::check()) {
114         return View::make('index');
115     } else {
116         return Redirect::to('login');
117     }
118 });
119
120
```

la anterior ruta nos dirige a la página principal siempre y cuando estemos autenticados, en caso contrario nos llevaría a la página del login.

Las rutas están siempre declaradas usando la clase Route . Eso es lo que tenemos al principio, antes de :: . La parte get es el método que usamos para ‘capturar’ las peticiones que son realizadas usando el verbo ‘GET’ de HTTP hacia una URL concreta.

Como vemos, todas las peticiones realizadas por un navegador web contienen un verbo. La mayoría de las veces, el verbo será GET , que es usado para solicitar una página web. Se envía una petición GET cada vez que escribimos una nueva dirección web en el navegador.

Aunque no es la única petición. También está POST , que es usada para hacer una petición y ofrecer algunos datos. Normalmente se usa para enviar un formulario en la que se necesita enviar los datos sin mostrarlo en la URL.

Hay otros verbos HTTP disponibles. He aquí algunos de los métodos que la clase de enrutado tiene disponibles:

- ➔ `Route::get();`
- ➔ `Route::post();`
- ➔ `Route::any();`
- ➔ `Route::delete();`
- ➔ `Route::put();`

Cualquier método de la clase Route recibe siempre dos argumentos, el primero es la URI con la que queremos hacer coincidir la URL y el segundo es la función a realizar que en este caso es un Clousure que no es otra cosa que una función anónima, es decir, que no tiene un nombre.

Tras una introducción general sobre el uso e implementación de las rutas, volvamos al caso que nos trae entre manos, una vez creado nuestro modelo pasaremos a crear una ruta de tipo *resource* en nuestro archivo *routes.php*, acabamos de ver el enrutamiento básico que nos ofrece Laravel, para retornar vistas desde el propio enrutamiento, pero ahí no se queda todo, Las rutas pueden ser relacionadas con métodos de un controlador, como ya vimos cuando hablamos del sistema de Autenticación:

```
13
14
15 /*Llamadas al controlador Auth*/
16 Route::get('login', 'AuthController@showLogin'); // Mostrar login
17 Route::get('/', 'AuthController@showLogin'); // Mostrar login
18 Route::post('login', 'AuthController@postLogin'); // Verificar datos
19 Route::get('logout', 'AuthController@logOut'); // Finalizar sesión
20
21
22
```

Y todavía nos quedamos sin hablar de un tipo especial de controlador llamado controlador de recurso (*resource controller*), que facilita la construcción de controladores tipo *RESTful*, que es el que vamos a usar en el caso del Cliente.

Para esto simplemente tendríamos que usar el comando de Artisan

```
php artisan make:controller <nombre-controlador> --resource
```

Para crear el controlador para la gestión de Clientes almacenados en la aplicación, en primer lugar ejecutaríamos el siguiente comando:

```
php artisan make:controller Cliente --resource
```

Esto crearía el controlador *ClienteController* (incluyendo todos los métodos necesarios) en la carpeta *app/controllers*.

Lo único que nos faltaría es registrar las rutas asociadas añadiendo al fichero *routes.php* la siguiente línea:

```
Route::resource('cliente', 'ClienteController');
```

Esta línea de ruta crea por si sola múltiples rutas para gestionar todos los tipos de peticiones *RESTful*. Además, el controlador creado mediante Artisan estará preparado con todos los métodos necesarios para responder a todas las peticiones correspondientes. En la siguiente tabla se muestra un resumen de todas las rutas generadas, el tipo de petición a la que responden y la acción que realizan en el controlador:

Verbo	Ruta	Acción	Controlador / método
GET	/cliente	index	ClienteController@index
GET	/cliente/create	create	ClienteController@create
POST	/cliente	store	ClienteController@store
GET	/cliente/{resource}	show	ClienteController@show
GET	/cliente/{resource}/edit	edit	ClienteController@edit
PUT/PATCH	/cliente/{resource}	update	ClienteController@update
DELETE	/cliente/{resource}	destroy	ClienteController@destroy

Estas son las operaciones mas usadas en una clase y para no tener que crear una ruta para cada método es que Laravel agrupa todo esto con una ruta de tipo resource que se liga a un controlador.

Estos métodos significan:

- **index**: Es el método inicial de las rutas resource, usualmente lo usamos para mostrar una vista como página principal que puede contener un catalogo o resumen de la información del modelo al cual pertenece o bien no mostrar información y solo tener la función de página de inicio.
- **create**: Este método lo podemos usar para direccionar el sistema a la vista donde se van a recolectar los datos(probablemente con un formulario) para después almacenarlos en un registro nuevo, usualmente redirige al index.
- **show**: Aquí podemos hacer una consulta de un elemento de la base de datos o de todos los elementos o registros por medio del modelo para realizar una descripción.
- **edit**: Este método es similar al de create porque lo podemos usar para mostrar una vista que recolecta los datos pero a diferencia de create es con el fin de actualizar un registro.
- **store**: Aquí es donde se actualiza un registro en específico que proviene del método create y normalmente redirige al index.
- **update**: Al igual que el store, solo que en vez de provenir de create proviene de edit y en vez de crear un nuevo registro, busca un existente y lo modifica, también suele redirigir al index.
- **destroy**: En este método usualmente se destruye o elimina un registro y la petición puede provenir de donde sea siempre y cuando sea llamado con el método DELETE, después puede redirigir al index o a otro sitio dependiendo si logra eliminar o no.

Ahora esto no quiere decir que un controlador necesariamente debe ejecutar estas peticiones obligadamente, podemos omitirlas o incluso agregar mas.

7.4.5 Controlador

Hasta el momento hemos visto solamente como devolver una cadena para una ruta y como asociar una vista a una ruta directamente en el fichero de rutas. Pero en general la forma recomendable de trabajar será asociar dichas rutas a un método de un controlador. Esto nos permitirá separar mucho mejor el código y crear clases (controladores) que agrupen toda la funcionalidad de un determinado recurso. Por ejemplo, podemos crear un controlador para gestionar toda la lógica asociada al control de usuarios o cualquier otro tipo de recurso.

Como ya vimos, los controladores son el punto de entrada de las peticiones de los usuarios y son los que deben contener toda la lógica asociada al procesamiento de una petición, encargándose de realizar las consultas necesarias a la base de datos, de preparar los datos y de llamar a la vista correspondiente con dichos datos.

Los controladores se almacenan en ficheros PHP en la carpeta app/Controllers y normalmente se les añade el sufijo Controller, por ejemplo UserController.php o ClienteController.php. A continuación se incluye un ejemplo básico de un controlador almacenado en el fichero app/controllers/UserController.php:

```
1 <?php
2 namespace App\Http\Controllers;
3
4 use App\User;
5 use App\Http\Controllers\Controller;
6
7 class UserController extends Controller
8 {
9     /**
10      * Mostrar información de un usuario.
11      * @param int $id
12      * @return Response
13      */
14     public function showProfile($id)
15     {
16         $user = User::findOrFail($id);
17         return view('user.profile', ['user' => $user]);
18     }
19 }
```

Todos los controladores tienen que extender la clase base Controller. Esta clase viene ya creada por defecto con la instalación de Laravel, la podemos encontrar en la carpeta app/controllers. Se utiliza para centralizar toda la lógica que se vaya a utilizar de forma compartida por los controladores de nuestra aplicación. Por defecto solo carga código para validación y autorización, pero podemos añadir en la misma todos los métodos que necesitemos.

Para el caso que estamos siguiendo y que de paso nos sirve como patrón de uso de implementación de cada parte del modelo-vista-controlador en Laravel, en el anterior punto vimos que hacemos uso de un caso particular de enrutamiento y mostramos como crear el controlador desde la línea de comandos:

```
php artisan make:controller Cliente –resource
```

Este comando genera el siguiente código:

```
1  <?php
2
3  class Cliente extends \BaseController {
4
5      /**
6      * Display a listing of the resource.
7      *
8      * @return Response
9      */
10     public function index()
11    {
12        //
13    }
14
15    /**
16     * Show the form for creating a new resource.
17     *
18     * @return Response
19     */
20     public function create()
21    {
22        //
23    }
24
25    /**
26     * Store a newly created resource in storage.
27     *
28     * @return Response
29     */
30     public function store()
31    {
32        //
33    }
34
35    /**
36     * Display the specified resource.
37     *
38     * @param int $id
39     * @return Response
40     */
41     public function show($id)
42    {
43        //
44    }
45
46    /**
47     * Show the form for editing the specified resource.
48     *
49     * @param int $id
50
```

Como vemos tenemos ya definidos todos los métodos, ahora es turno de rellenar esos métodos y crear otros nuevos que necesitaremos para gestionar el sistema CRUD en este caso del módulo Cliente:

Validación

En **Laravel** podemos hacer uso del método **Validator()** con el cual podremos establecer ciertas reglas que serán verificadas al momento de enviar los datos, en caso de que la validación falle la aplicación genera los mensajes de error correspondientes para cada campo que podremos mostrar en la vista para indicar al usuario las correcciones que debe realizar.

Estas reglas la definimos en un array dentro del controlador de la siguiente forma:

Constructor

Creamos un constructor donde usaremos un filtro de seguridad que nos proporciona laravel para dejar entrar solo a los usuarios registrados, estos filtro se pueden construir a medida dentro del archivo app/filters y llamarlos desde el constructor del controlador.

```
15  
16▼    public function __construct() {  
17▼        $this->beforeFilter(function() {  
18            if (!Auth::check()) {  
19                return Redirect::to('/');  
20            }  
21        });  
22    }  
23
```

Index(), create(), show(), edit()

Como ya hemos dicho anteriormente, laravel nos provee de un ORM llamado Eloquent, que nos facilita las consultas sobre la base de datos usando métodos sobre un objeto para construir nuestras consultas, una vez definimos el modelo, podemos hacer uso del mismo para las consultas que necesitemos, así nuestros métodos Index(), create(), show() y edit() quedan de la siguiente forma:

```
23 |
24     public function index() {
25         $ObjCliente = Cliente::orderBy('id','DESC')->get();
26         return View::make('Cliente.index')->with('Cliente', $ObjCliente);
27     }
28
29     public function create() {
30         return View::make('Cliente.create');
31     }
32
33     public function show($id) {
34         $ObjCliente = Cliente::find($id);
35         return View::make('Cliente.show')->with('Cliente', $ObjCliente);
36     }
37
38     public function edit($id) {
39         $ObjCliente = Cliente::find($id);
40         return View::make('Cliente.edit')->with('Cliente', $ObjCliente);
41     }
42 }
```

store() y update()

Los métodos store y update aplican las reglas de validación y permitirán en cada caso almacenar o actualizar respectivamente los datos de los clientes:

```
42
43     public function store() {
44         $input = Input::all();
45         $validation = Validator::make($input, $this->rules);
46         if (!$validation->fails()) {
47             $ObjCliente = new Cliente;
48             $ObjCliente->nombre = $input['nombre'];
49             $ObjCliente->apellido1 = $input['apellido1'];
50             $ObjCliente->apellido2 = $input['apellido2'];
51             $ObjCliente->telefono = $input['telefono'];
52             $ObjCliente->direccion = $input['direccion'];
53             $ObjCliente->provincia = $input['provincia'];
54             $ObjCliente->pais = $input['pais'];
55             $ObjCliente->dni = $input['dni'];
56             $ObjCliente->email = $input['email'];
57             $ObjCliente->activo = 1;
58             $ObjCliente->save();
59             return Redirect::to('administracion/cliente');
60         } else {
61             return Redirect::back()->withErrors($validation)->withInput();
62         }
63     }
64
65
66
67     public function update($id) {
68         $input = Input::all();
69         $validation = Validator::make($input, $this->rules);
70         if (!$validation->fails()) {
71             $ObjCliente = Cliente::find($id);
72             $ObjCliente->nombre = $input['nombre'];
73             $ObjCliente->apellido1 = $input['apellido1'];
74             $ObjCliente->apellido2 = $input['apellido2'];
75             $ObjCliente->telefono = $input['telefono'];
76             $ObjCliente->direccion = $input['direccion'];
77             $ObjCliente->provincia = $input['provincia'];
78             $ObjCliente->pais = $input['pais'];
79             $ObjCliente->dni = $input['dni'];
80             $ObjCliente->email = $input['email'];
81             $ObjCliente->activo = 1;
82             $ObjCliente->save();
83             return Redirect::to('administracion/cliente');
84         } else {
85             return Redirect::back()->withErrors($validation)->withInput();
86         }
87     }
88 }
```

destroy()

Con Laravel podemos eliminar registros de dos formas utilizando el ORM de Laravel, Eloquent dentro de nuestro resource controller.

Una de ellas es usando el método `destroy`, que acepta como parámetro una ID.

Otra es usando el método `delete` (para ello necesitamos cargar antes el objeto Cliente)

```
88
89     public function destroy($id) {
90         $ObjCliente = Cliente::find($id);
91         $ObjCliente->delete();
92         return Redirect::to('administracion/cliente');
93     }
94 
```

En este mismo controlador y en el resto de controladores de los distintos módulos de la aplicación, se han construido más métodos personalizados para distintas funciones. Están todos documentados en el código fuente, ya que sería un trabajo arduo e interminable hacer mención de todos y cada uno de ellos.

Vista.

Continuando con el módulo de los Clientes, veamos como mandar a la vista un objeto Cliente:

```
23
24     public function index() {
25         $ObjCliente = Cliente::orderBy('id','DESC')->get();
26         return View::make('Cliente.index')->with('Cliente', $ObjCliente);
27     }
28 
```

Como podemos ver, dentro del método `index()`, devolvemos la salida al navegador mediante la vista, la cual llamamos con la clase `View` y el método `make()`. Este método incluye el nombre de la vista (sin el “blade.php”), como en este caso la vista está dentro de una carpeta llamada ‘Cliente’, debemos hacerlo separando con un punto la

carpeta de la vista: “carpeta.vista”, el método with() recibe dos parámetros, la variable con el nombre que estará disponible en la vista y el valor de ésta.

Como en este caso lo que estamos mandando a la vista no es una variable sino un objeto, para acceder al mismo realizaremos dentro de la vista un simple bucle con unas connotaciones sintácticas un poco especiales, lo vemos en el siguiente pantallazo:

```
16          <tr>
17          <th>Nombre</th>
18          <th>Apellido1</th>
19          <th>Apellido2</th>
20          <th>DNI</th>
21          <th>Teléfono</th>
22          <th>Email</th>
23          <th></th>
24          <th></th>
25          <th></th>
26      </tr>
27  </thead>
28  <tbody>
29  @foreach($Cliente as $row)
30      <tr>
31          <td>{{ $row->nombre }}</td>
32          <td>{{ $row->apellido1 }}</td>
33          <td>{{ $row->apellido2 }}</td>
34          <td>{{ $row->dni }}</td>
35          <td>{{ $row->telefono }}</td>
36          <td>{{ $row->email }}</td>
```

Como vemos de ver, las vistas en laravel son simple html con la particularidad de usar una @ delante de ciertos elementos como bucles, condicionales, etc, y entre dobles llaves la parte dinámica que vamos a mostrar, así de sencillo.

Con este paso, casi por encima de la implementación del módulo Cliente, y una introducción de lo que puede hacer laravel por nosotros, espero haya servido de guía para que más o menos quede clara la construcción del proyecto.

Como ya he comentado, he tratado de comentar el código fuente lo suficiente para comprender toda la lógica de la aplicación.

8 CONCLUSIONES

Tras la realización de este proyecto se han cumplido todos los objetivos propuestos. He mejorado mucho mis conocimientos sobre tecnologías web, he ampliado lo que ya sabía sobre HTML, CSS, JavaScript y PHP y he aprendido muchas otras cosas nuevas.

He aprendido a utilizar frameworks y librerías que me han ayudado mucho a desarrollar y organizar el proyecto. He incorporado buenas prácticas a la hora de estructurar el código y utilizar herramientas para realizar un trabajo más eficiente y mantible.

Además he aprendido que dedicar un tiempo a la planificación previa, a trabajar siguiendo una arquitectura concreta como es la arquitectura MVC y aplicando una metodología como Scrum y ceñirse a ella, es una parte tan importante como la propia programación.

He descubierto que las tecnologías y herramientas que existen en este campo son muchísimas y que siguen creciendo cada día, por lo que hay que estar siempre actualizado. Pero esto no me supone un problema ya que esta experiencia ha fomentado mi interés por seguir aprendiendo y mejorando cada día en el ámbito del desarrollo y el diseño web.

9 TRABAJO FUTURO

La aplicación espera futuras actualizaciones con mejoras y con nuevas funcionalidades que aporten mas profesionalidad al conjunto del sistema.

9.1 Mejoras en la consulta al correo

Actualmente para consultar si entra una reserva desde la página web, se dispone de un acceso directo al cliente de correo, lo ideal, será poder consultar el mismo desde la propia aplicación, esto se podrá conseguir haciendo uso de la API de Google por ejemplo.

9.2 Mejoras en el Sistema de Reservas

se tratará de implementar el típico planning mensual donde de un vistazo podamos ver la ocupación por días, semanas , meses e interactuar desde ahí con el sistema de reservas, clientes, etc.

9.3 Mejoras en la web.

Se pretende aportar mas personalización a la web del cliente, que no se trate de una web tan genérica.

Con nuevas funcionalidades como: un sistema de registro y autenticación, pasarelas de pago, sección comentarios, ofertas, etc.

9.4 Nuevas funcionalidades.

Se añadirán:

- Conexión con las camareras de piso para consultar en tiempo real el estado de las habitaciones, minibar...
- Sistema de reservas para el restaurante.

- Sistema de reservas para el alquiler de salones.
- Sistema de facturación y albaranes.
- Control de mercaderías y stockage
- Control de compras, gastos, etc.
- Sistema para el control de eventos, bodas, comidas de empresa, etc..
- Control de proveedores.
- Control de contenidos de la web para poder actualizar imágenes, ofertas, comentarios de clientes,etc..

También conviene realizar un esfuerzo adicional en el aspecto del rendimiento, creando un sitio más seguro y usando componentes dinámicos junto con procesos asíncronos con AJAX, para no recargar las vistas.

En cuanto a seguridad, el framework estudiado ya cuenta con algunos mecanismos como las validaciones de los formularios, para protegerse de CSRF y las inyecciones SQL, pero se seguirá mejorando este aspecto tan importante.

Prueba de ello, es que Laravel se ha actualizado a la versión LTS 5.4 y recibirá soporte a largo plazo, por lo que se trabajará en su migración en la siguiente versión de GestHotel.

10 PRUEBAS Y RESULTADOS

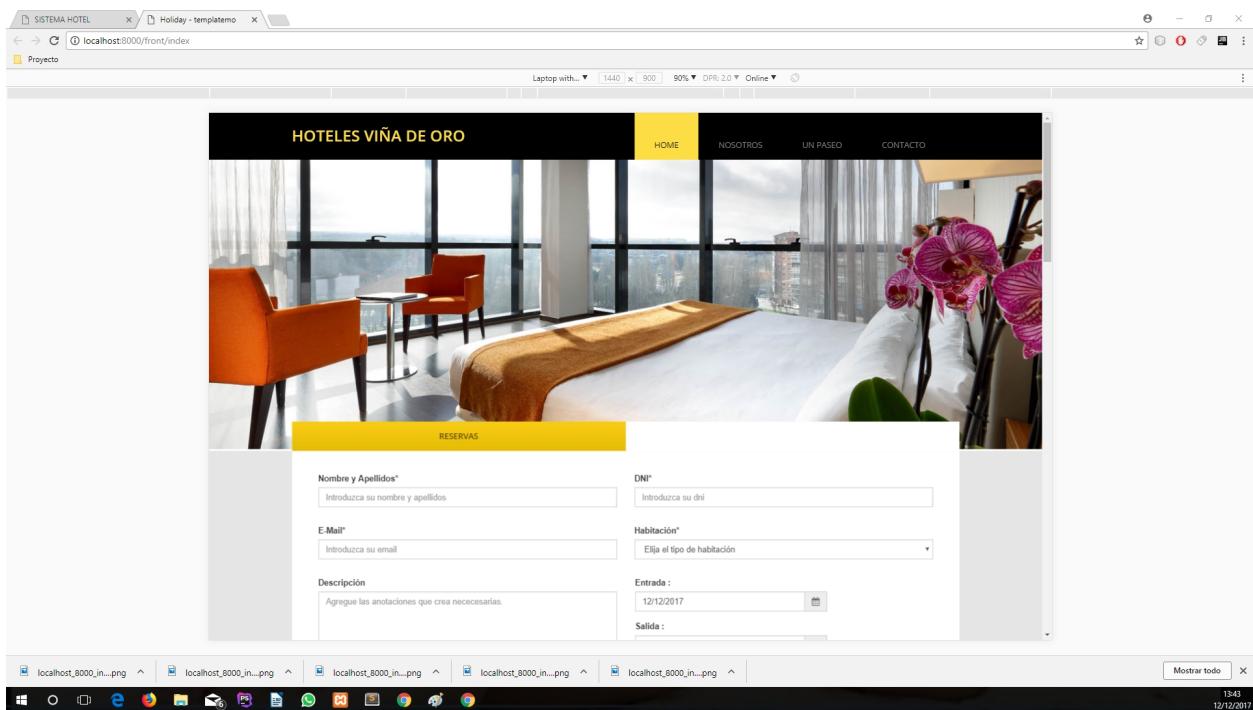
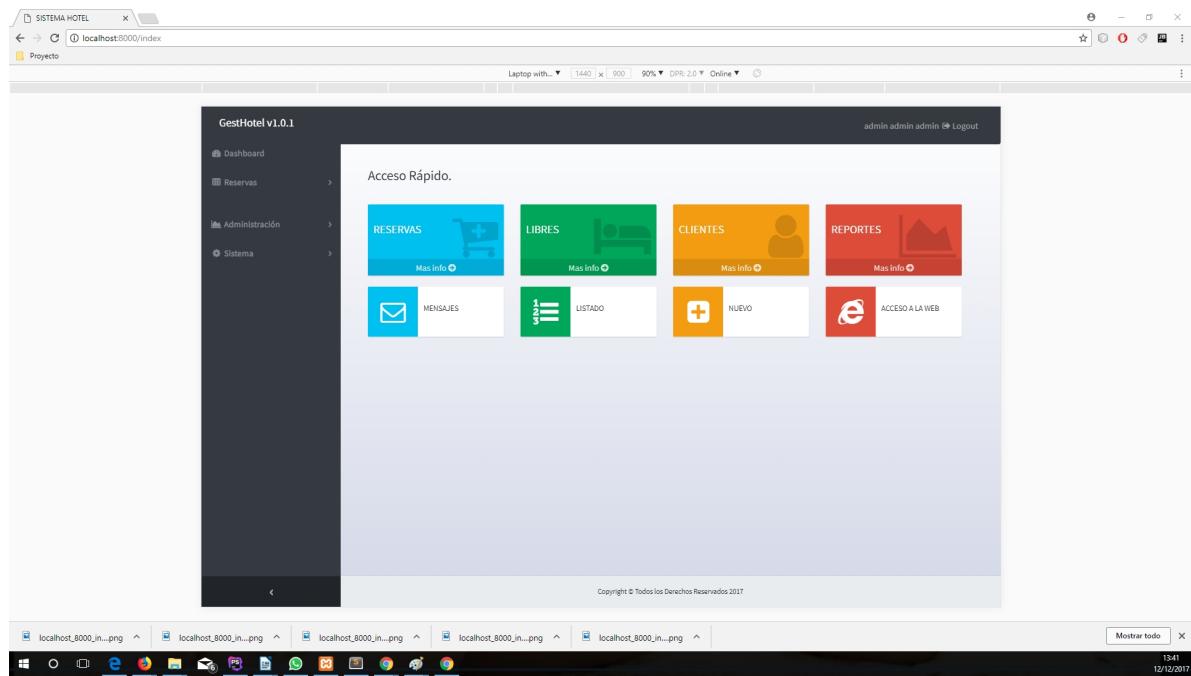
En este capítulo se comentan las pruebas hechas y los resultados de la misma, abordaremos una serie de pruebas que se detallan a continuación:

10.1 Pruebas de resolución

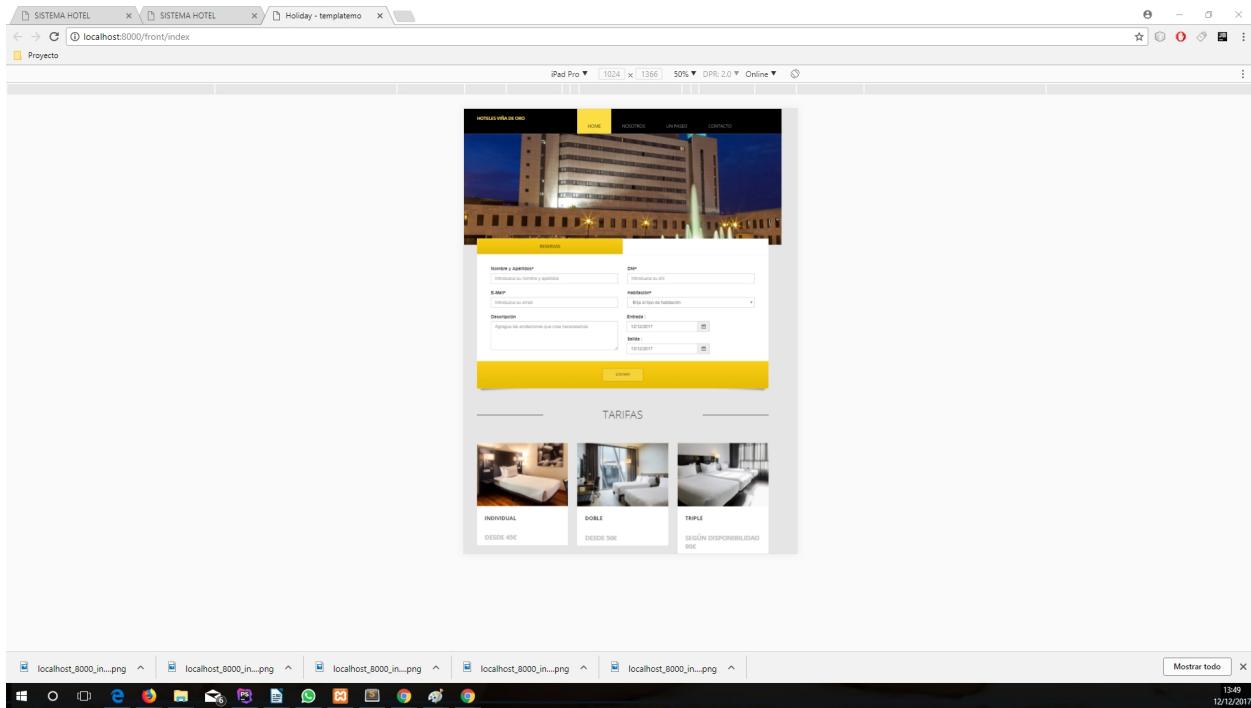
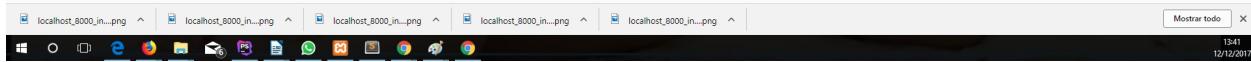
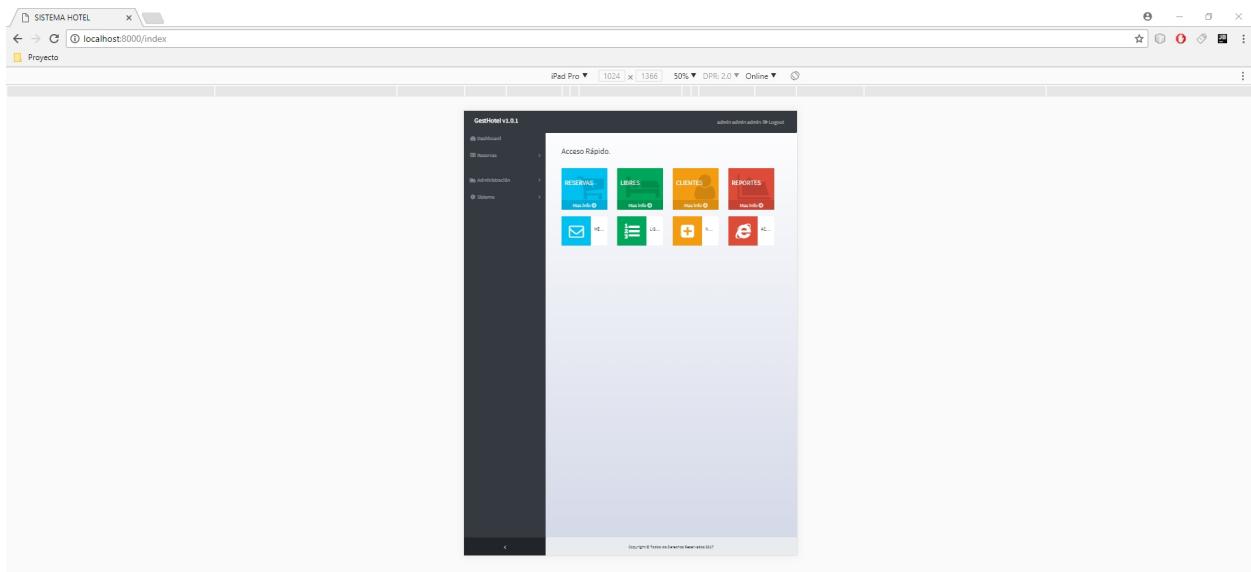
Las pruebas de resolución son necesarias para que los usuarios puedan ver correctamente tanto el panel de administración como la web que se han diseñado en este Proyecto. Hay que cuidar mucho este aspecto, por eso hemos probado distintas resoluciones usando Chrome Developer Tools, esta es la forma mas sencilla y personalizable que existe sin que sea necesario instalar una extensión, basta con abrir las Herramientas para desarrolladores de Google Chrome ([CTRL+MAYUS+I](#)), hacer clic en el piñón que aparece en la parte derecha inferior ir a “Overrides” y personalizar las versiones del navegador y la resolución del dispositivo.

Resultados:

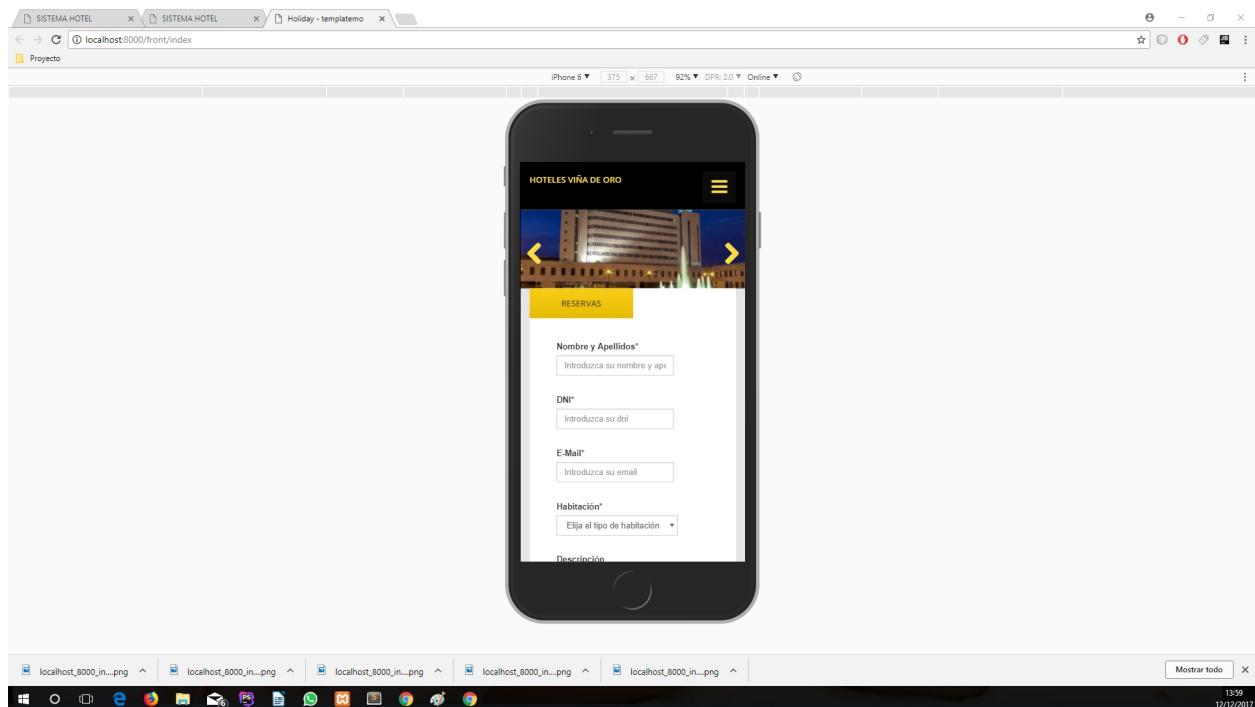
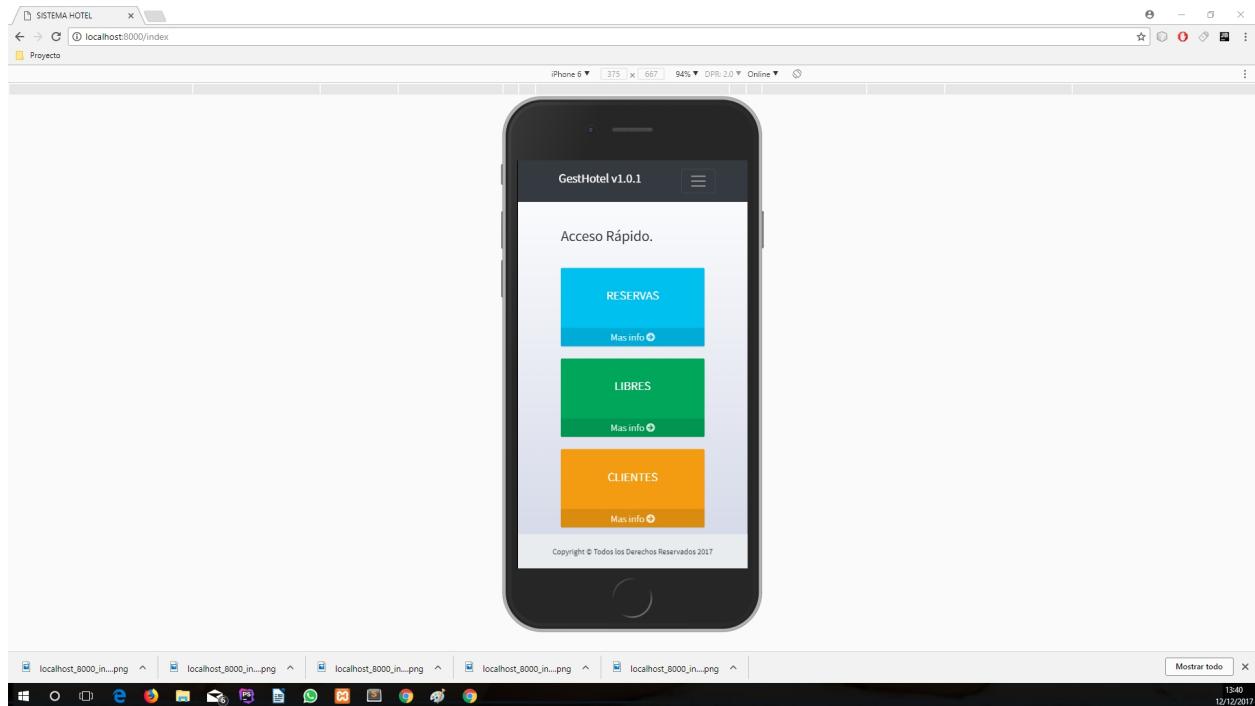
laptop 1440 x 900 px.



IpadPro 1024 x 1366 px



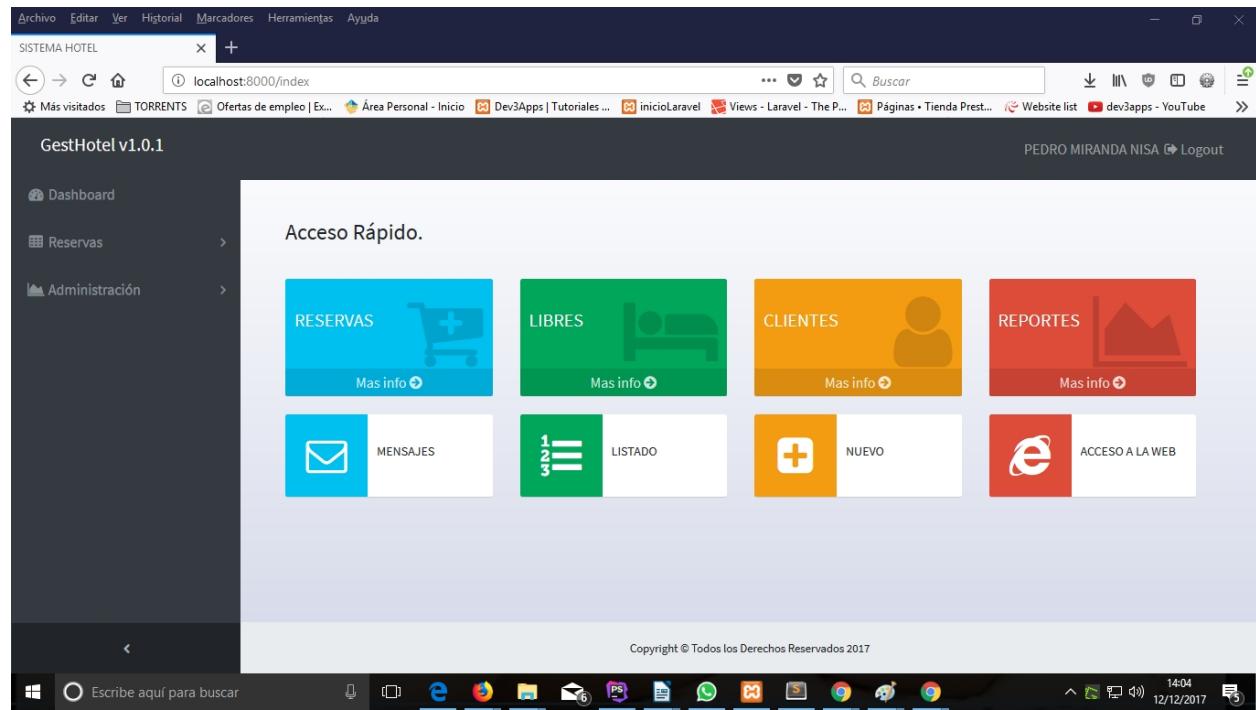
iPhone 6 375 x 667

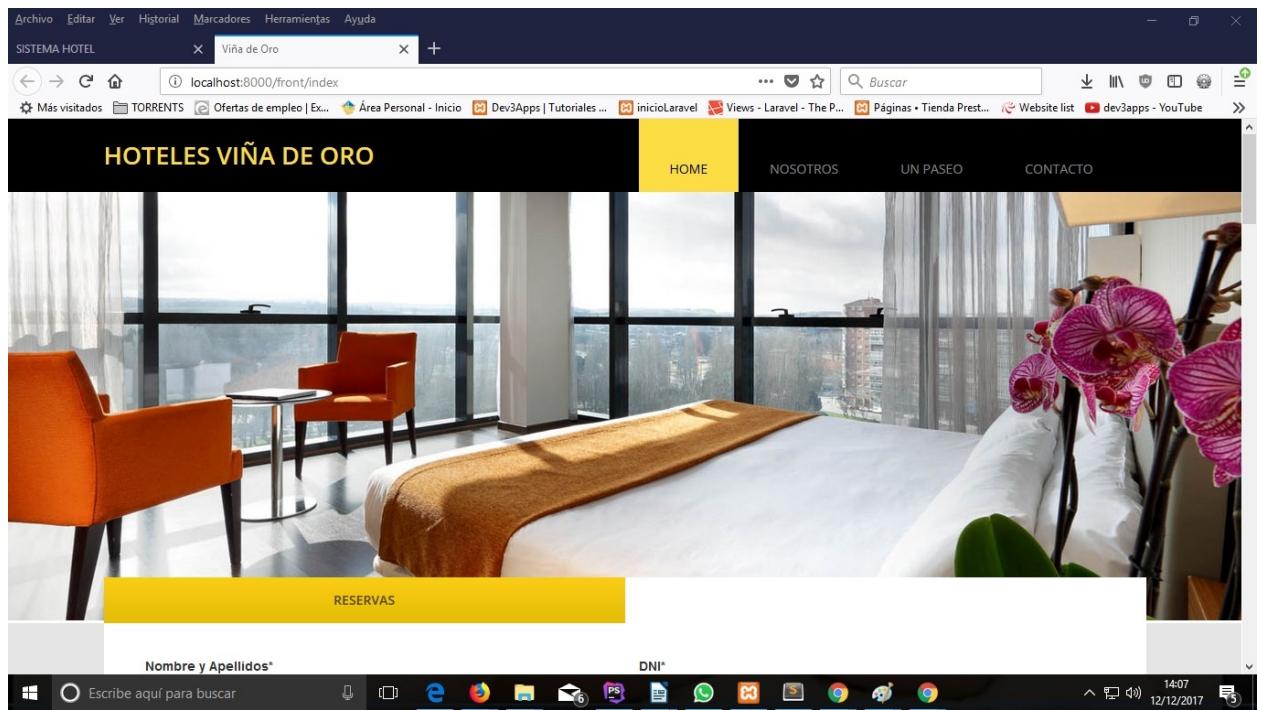


10.2 Pruebas con navegadores

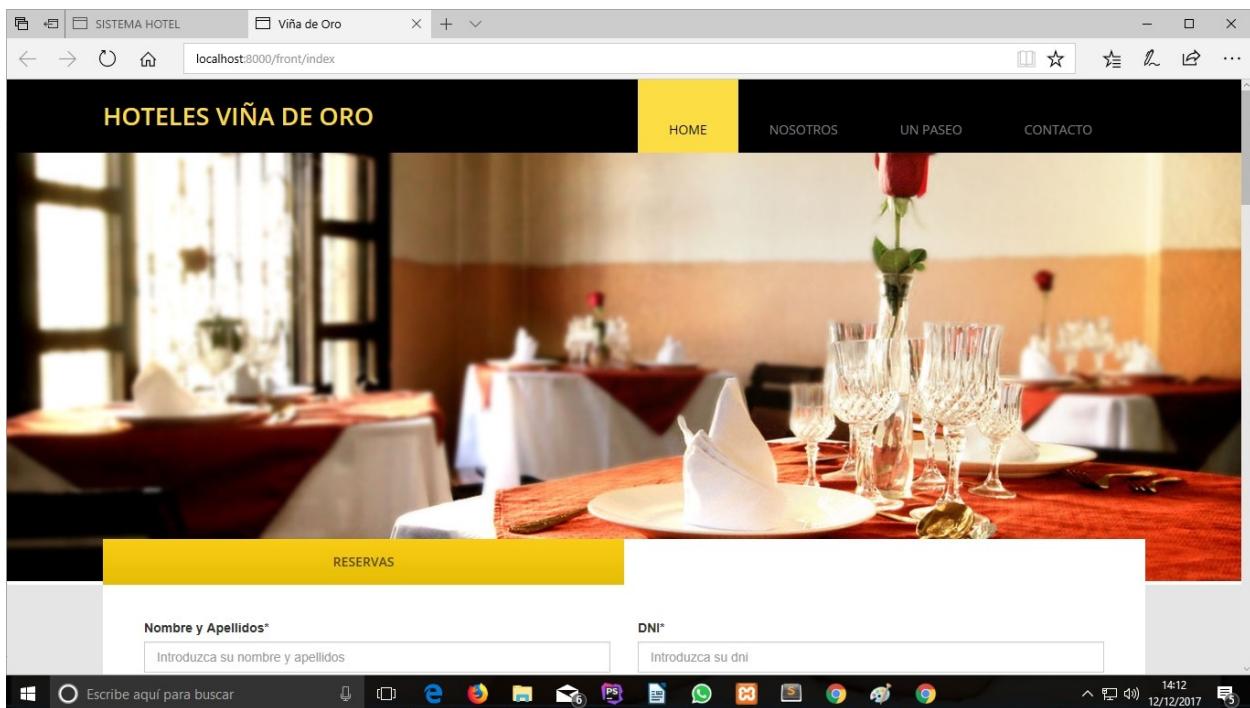
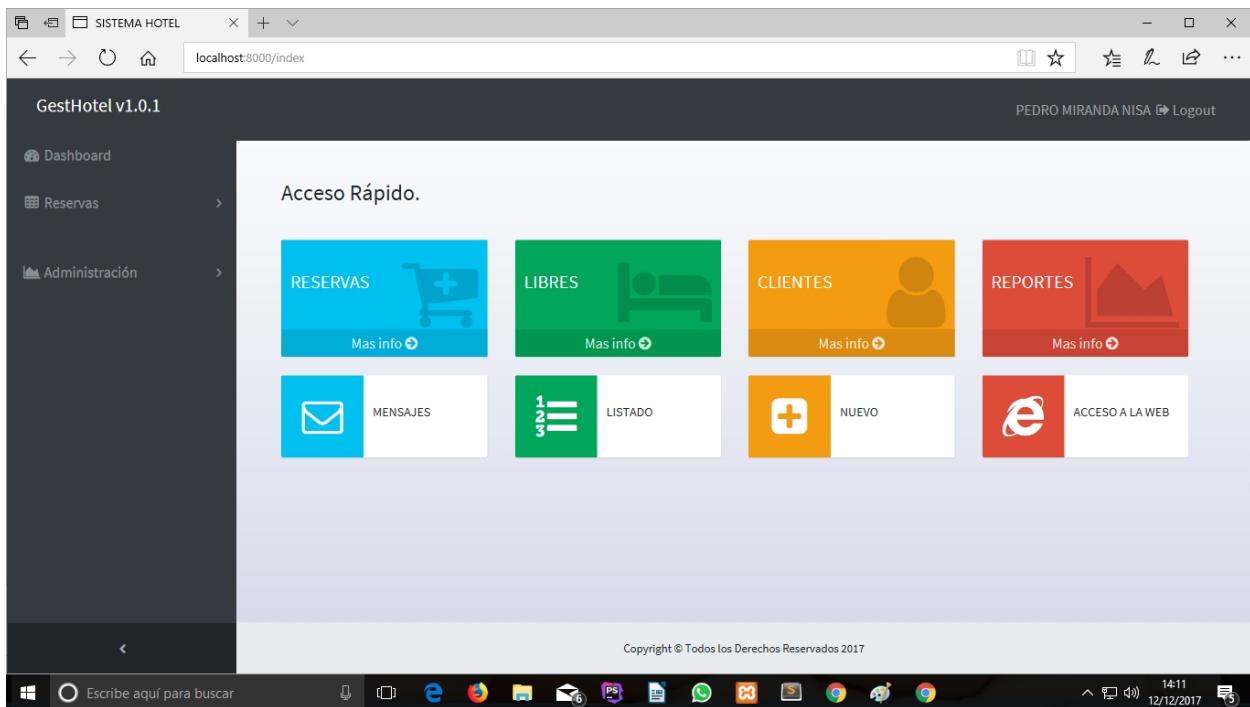
Esta prueba consiste en comprobar el correcto funcionamiento en los distintos navegadores. Hemos escogido los más representativos, Mozilla Firefox, Internet Explore, Google Chrome y Safari.

Mozilla Firefox

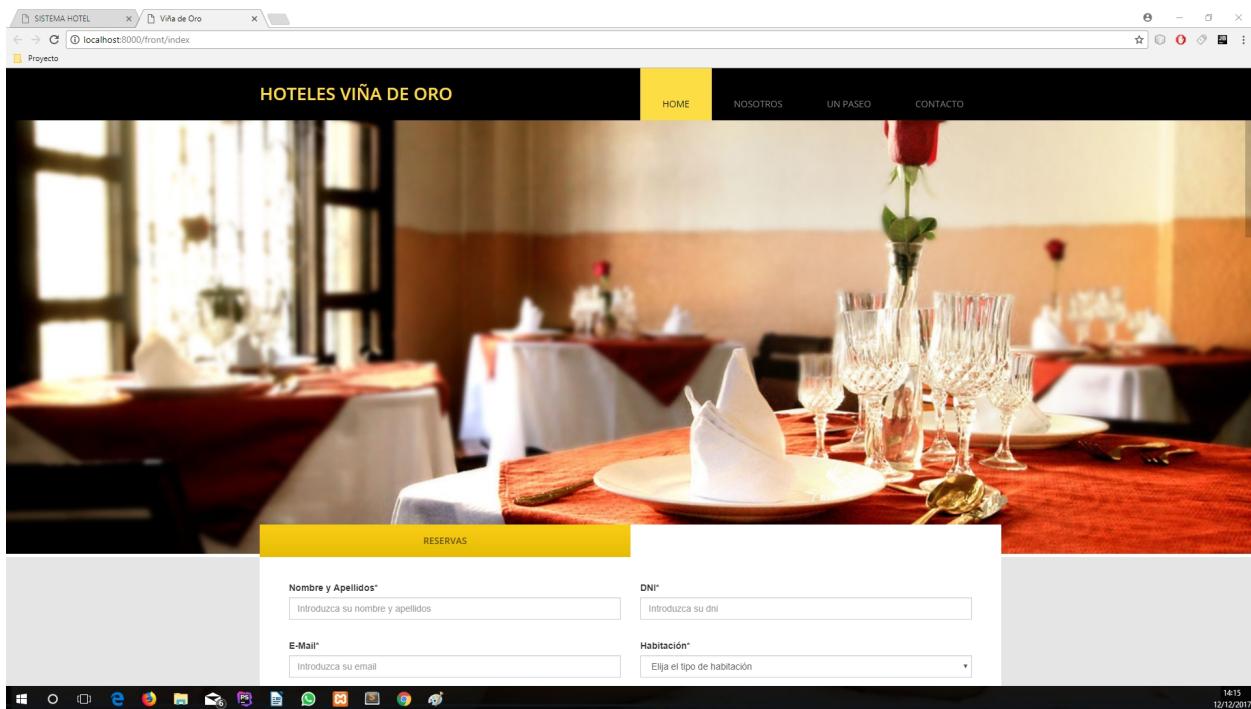
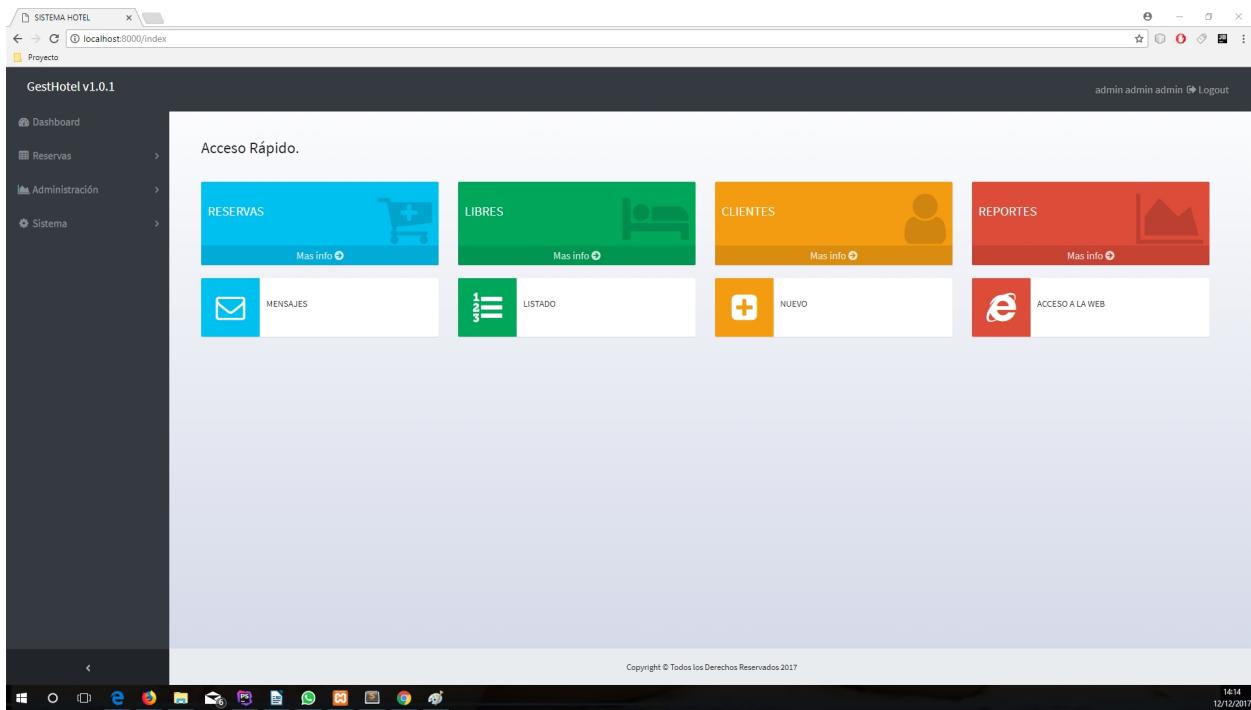




Internet Explorer



Google Chrome



Safari

The screenshot shows the GestHotel v1.0.1 application running in a Safari browser window. The title bar reads "SISTEMA HOTEL" and the address bar shows "http://localhost:8000/index". The main header displays "GestHotel v1.0.1" and the user "PEDRO MIRANDA NISA" with a "Logout" link. On the left, a sidebar menu includes "Dashboard", "Reservas", and "Administración". The main content area features a section titled "Acceso Rápido." with six colored cards: RESERVAS (blue), LIBRES (green), CLIENTES (orange), REPORTES (red), MENSAJES (white), and LISTADO (white). At the bottom, a footer bar includes "Copyright © Todos los Derechos Reservados 2017" and a system status bar showing "14:50" and "12/12/2017".

The screenshot shows the Vina de Oro website in a browser window. The title bar reads "Vina de Oro" and the address bar shows "http://localhost:8000/front/index". The header features the hotel's name "HOTELES VIÑA DE ORO" and navigation links for "HOME", "NOSOTROS", "UN PASEO", and "CONTACTO". The main content area is a large image of a hotel lobby with two women at a reception desk. Overlaid on this image is a white form for "RESERVAS" with fields for "Nombre y Apellidos*", "E-Mail*", "DNI*", and "Habitación*". The bottom of the screen shows a standard Windows taskbar with various icons.

10.3 Pruebas de los métodos

Desarrollar sobre un patrón de diseño MVC facilita las pruebas de los métodos, pero a su vez, complica bastante el seguimiento del hilo de ejecución de la aplicación, requiriendo de herramientas más complejas. Se enumeran, por orden de complejidad, los métodos utilizados.

En el archivo config/app.php se puede activar el modo debug, muy útil en la etapa de desarrollo, para localizar los errores cuando se intenta cargar una ruta inválida en el navegador web, mostrándose todo el flujo que ha seguido la ejecución y las líneas de error.

Una medida de prevención muy útil a la hora de probar un controlador, es usar el helper de Laravel dd(\$variable), parecido a la función var_dump() de PHP, que finaliza la ejecución de la aplicación e imprime el contenido de la variable que recibe como argumento.

En el transcurso de un entorno de desarrollo a uno de pruebas, es necesario llevar un registro de errores para continuar con el proceso de depuración de forma transparente. Laravel Log Viewer es un paquete que registra todos los errores ocurridos en la aplicación, de gran utilidad cuando se deshabilita el modo debug.

Se ha instalado una barra de depuración Laravel Debugbar, que permite conocer rápidamente información de todo lo que se está ejecutando al cargar una página, como por ejemplo, se pueden obtener los nombres de una ruta asociada, las consultas SQL que se han ejecutado, la memoria cache utilizada o el tiempo de respuesta. Con este paquete se mejoran los problemas que afectan al rendimiento de GestHotel, optimizando el código.

10.4 Pruebas Unitarias

En cuanto a pruebas unitarias, Laravel integra *PHPUnit* y *PHPSpec*. Estas herramientas permiten aislar cada parte de la funcionalidad de la aplicación y comprobar que funcionan individualmente.

Se ha utilizado en el módulo Clientes, a la hora de obtener resultados de las relaciones con sus contenidos. Para invocar una prueba creada con sus correspondientes asserts, que verifique el correcto funcionamiento de un método específico del controlador, que se encarga de obtener los clientes a través de un identificador, se ejecuta:

```
phpunit --bootstrap src/Cliente.php tests/ClienteTest
```

11 BIBLIOGRAFÍA

11.1 Tecnologías web

Tecnologías del lado del cliente

W3C. HTML. <<http://www.w3.org/html/>>

Mozilla Developer Network. HTML element reference.

<<https://developer.mozilla.org/en-US/docs/Web/HTML/Element>>

W3Schools. HTML Tutorial. <<http://www.w3schools.com/html/>>

W3C. CSS. <<http://www.w3.org/TR/CSS/>>

W3Schools. CSS Tutorial. <<http://www.w3schools.com/css/>>

W3Schools. JavaScript Tutorial. <<http://www.w3schools.com/js/>>

JQuery. <<https://jquery.com/>>

JQuery. JQuery Date and Time picker. <<https://plugins.jquery.com/datetimepicker/>>

Datepicker. JQuery plugin to select date and time.

<<http://xdsoft.net/jqplugins/datetimepicker/>>

JQuery User Interface. Autocomplete. <<https://jqueryui.com/autocomplete/>>

JSColor. JavaScript / HTML Color Picker. <<http://jscolor.com/>>

Semantic UI. <<http://semantic-ui.com/>>

W3Schools. AJAX Tutorial. <<http://www.w3schools.com/ajax/>>

JQuery.ajax(). <<http://api.jquery.com/jquery.ajax/>>

Tecnologías del lado del servidor

PHP. <<https://php.net/>>

PHP. ¿Qué es PHP? <<http://php.net/manual/es/intro-whatis.php>>

Laravel. Documentación en inglés <<http://laravel.com/>>

Laravel. Documentación en español <<http://laraveles.com/docs/4.2/>>

Laravel. Query Builder. Documentación en inglés. <<http://laravel.com/docs/4.2/queries>>

Laravel. Generador de consultas. Documentación en español.
<<http://laraveles.com/docs/4.2/queries>>

Laravel. Eloquent ORM. Documentación en inglés.
<<http://laravel.com/docs/4.2/eloquent>>

Laravel. Eloquent ORM. Documentación en español.
<<http://laraveles.com/docs/4.2/eloquent#introduction>>

Laravel. Artisan CLI. Documentación en inglés. <<http://laravel.com/docs/4.2/artisan>>

Laravel. Artisan CLI. Documentación en español.
<<http://laraveles.com/docs/4.2/artisan>>

Laravel wiki. Composer. <http://wiki.laravel.io/Laravel_4#Composer>

Composer. <<https://getcomposer.org/>>

XAMPP. <<https://www.apachefriends.org/es/index.html>>

PhpMyAdmin. <http://www.phpmyadmin.net/home_page/index.php>

W3Schools. SQL Tutorial. <<http://www.w3schools.com/sql/>>

MySQL. <<https://www.mysql.com/>>

W3Schools. JSON Tutorial. <<http://www.w3schools.com/json/>>

MailGun. Email Service. <<https://mailgun.com/>>

11.2 Entornos de programación

PhpStorm. <<https://www.jetbrains.com/phpstorm/>>

Sublime Text. <<http://www.sublimetext.com/>>

MySQL. MySQL Workbench. <<http://www.mysql.com/products/workbench/>>

PHPUnit. <<https://phpunit.de/>>

11.3 Otras herramientas

Drawlo. <<https://www.draw.io/>>

Git. Control de versiones. <<http://git-scm.com/>>

Hipster Logo Generator. <<http://www.hipsterlogogenerator.com/>>

HTML Color Codes.<<http://html-color-codes.info/codigos-de-colores-hexadecimales/>>

Google Developers. Google Maps API. <<https://developers.google.com/maps/?hl=es>>

Google Developers. Google Maps Geocoding API.
<<https://developers.google.com/maps/documentation/geocoding/?hl=es>>

Página oficial de Scrum. <<https://www.scrum.org/>>

Guía de Scrum. <<http://www.scrumguides.org/>>

GestHotel

Manual de Instalación

Para la instalación de GestHotel en nuestro equipo, vamos a ver dos formas muy distintas de hacerlo, pero igual de eficaces las dos:

1 CLONANDO EL PROYECTO

Como estamos trabajando en un proyecto de Laravel y usamos Git para el control de versiones, podemos clonar el proyecto, pero antes de intentar correr la aplicación, debemos saber unas cosas.

Por defecto, el archivo `.gitignore` no tomará en cuenta la carpeta “vendor” de Laravel, y tampoco el archivo `.env` (el cual es muy importante). Por ello, al clonar nuestro proyecto (usando `git clone https://github.com/PedroNisa/GestHotel`) debemos hacer unos pequeños ajustes para correr nuestra app.

1.1 Instalar Composer

Sistemas Windows

Descarga el archivo `Composer-Setup.exe` desde el sitio web del proyecto Composer (<https://getcomposer.org/installer>) y ejecútalo como cualquier otro instalador de Windows.

Demás Sistemas Operativos, consultar las distintas instalaciones en:

<http://symfony.es/documentacion/guia-de-instalacion-de-composer/>

1.2 Instalar las dependencias usando Composer

Una vez clonado nuestro proyecto, abriremos una terminal y nos situaremos en dicha carpeta. Una vez estando ahí ejecutaremos:

composer install

Automáticamente composer leerá el archivo `composer.json` y comenzará a instalar todas las dependencias.

Una vez que termine podemos seguir con el siguiente paso.

1.3 Crear el Archivo .env

Por defecto, y por razones de seguridad, el archivo .env no es tomado en cuenta en el proyecto (ya que cada contribuidor puede tener diferentes contraseñas que no deberíamos saber) así que tenemos que generar uno por nosotros mismos. Podemos hacerlo desde la terminal (si estamos en Windows) escribiendo:

```
copy NUL .env
```

También podemos hacerlo con nuestro editor favorito, o incluso con el bloc de notas; el punto es crear el archivo.

Una vez creado, procederemos a editarlo, añadiendo las siguientes variables:

```
APP_NAME=Laravel
```

```
APP_ENV=local
```

```
APP_KEY=9SiA8Jt7cR2MdSPq5ddLz3RRBAIfwTu8
```

```
APP_DEBUG=true
```

```
APP_LOG_LEVEL=debug
```

```
APP_URL=http://localhost
```

```
DB_CONNECTION=mysql
```

```
DB_HOST=127.0.0.1
```

```
DB_PORT=3306
```

```
DB_DATABASE=dbhotel
```

```
DB_USERNAME=root
```

```
DB_PASSWORD=
```

```
BROADCAST_DRIVER=log
```

```
CACHE_DRIVER=file
```

```
SESSION_DRIVER=file
```

```
QUEUE_DRIVER=sync
```

```
REDIS_HOST=127.0.0.1
```

```
REDIS_PASSWORD=null
```

```
REDIS_PORT=6379
```

Este es el que editaremos. Pondremos nuestras contraseñas, usuarios, rutas, etcétera. Por lo regular sólo se editan las credenciales de las bases de datos:

```
DB_CONNECTION=mysql
```

```
DB_HOST=127.0.0.1
```

```
DB_PORT=3306
```

```
DB_DATABASE=dbhotel
```

```
DB_USERNAME=root
```

```
DB_PASSWORD=
```

Cuando terminemos, sólo guardaremos el fichero para proceder con el paso final.

1.4 Generar una clave

Laravel necesita una clave única para nuestros proyectos. La generaremos usando el comando:

```
php artisan key:generate
```

Y, si miramos nuestro archivo .env, veremos que se ha generado una nueva clave.

1.5 Levantando el servidor

Para hacer correr la aplicación, en la terminal, bastará con situarnos en la raíz del proyecto y ejecutar

```
php artisan serve
```

De esta manera podemos ir al navegador, visitar <http://localhost:8000/> y podremos ver nuestra aplicación funcionando. Para detener el servidor, en consola ejecutamos las teclas Ctrl+C.

2 DESCARGAR EL PROYECTO

La otra forma de hacer correr la aplicación en nuestro equipo es aún más fácil, la vamos a descargar del repositorio: noFile

Una vez descargada, la descomprimimos y suponiendo que ya deberemos tener un servidor corriendo en nuestro pc, nada más copiamos la carpeta completa GestHotel dentro de la carpeta que use nuestro servidor para correr las páginas.

Si por ejemplo estamos usando Xampp que nos provee de Apache, copiaríamos la carpeta GestHotel dentro de <c://xampp/htdocs>.

Para acceder a la página principal de la aplicación, deberemos escribir en cualquier navegador <http://localhost/GestHotel/public/index>

GestHotel

Manual del Administrador de Sistemas

\Versión: 0100

Fecha: 13/12/2017

1 Descripción del Sistema

Objeto.

En este manual encontrarás toda la información necesaria para la utilización de la aplicación **GestHotel** de una forma sencilla e intuitiva.

¿Qué es GestHotel?

GestHotel es una aplicación cuya misión es centralizar la gestión de un hotel.

En resumen, la aplicación permite gestionar el alta de clientes, reservas de habitaciones, checkin y checkout, reportes, altas de trabajadores, etc.

2 Configuración previa

El dispositivo en el que se va instalar la aplicación, deberá tener acceso a la red, para que la aplicación pueda conectarse con el servicio que le proporciona los datos.

Deberá alojar la aplicación en un servidor a su libre elección, con la única salvedad que su hosting ofrezca MySql como gestor de bases de datos.

3 Instalación de GestHotel

Para la instalación de **GestHotel** se le proporcionará un fichero .zip que una vez descomprimido deberá disponer de las siguientes carpetas:

- GestHotel
- Db
- Credenciales
- Manuales

En la carpeta GestHotel dispondrá de todos los ficheros que deberá subir al servidor a través de cualquiera de los métodos existentes, recuerde que solamente debe ser visible la carpeta /public.

En la carpeta Db encontrará un fichero sql para crear la base de datos.

En la carpeta Credenciales, como su nombre indica, aparecerán sus credenciales de acceso al sistema. Recuerde que deberá proporcionarle credenciales de acceso al cliente, tanto al administrador de contenidos como al resto de usuarios logueados.

En la carpeta Manuales, encontrará un Manual para el Administrador de Contenidos y otro manual para el resto de Usuarios.

4 Acceso a GestHotel

Para acceder a la aplicación una vez esté corriendo en el servidor, simplemente use el dominio que tenga contratado con el Hosting.

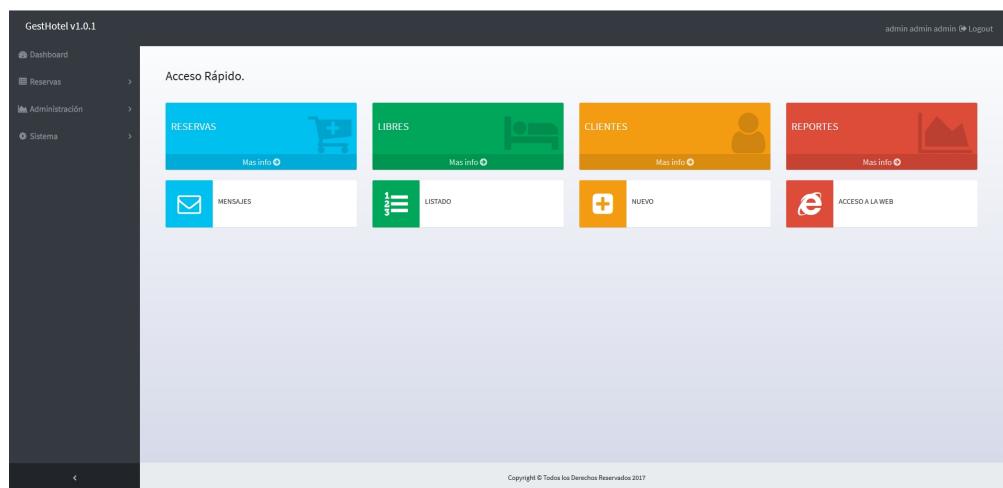
5 Inicio de GestHotel

La aplicación desplegará una pantalla de acceso a través de Login con las credenciales que se le ha proporcionado. Una vez ingresado los datos de acceso, si son correctos, dispondrá en su totalidad del panel de administración de GestHotel.

6 Página Principal

La primera pantalla que disponemos, nos proporciona, ademas del acceso a cualquier punto de la aplicación a través del menú lateral, una serie de accesos rápidos que serán los mas utilizados por el cliente:

- Acceso al listado de Reservas.
- Acceso a la disponibilidad de habitaciones.
- Acceso al listado de clientes.
- Acceso al generador de informes.
- Acceso al correo electrónico.
- Listado total de habitaciones.
- Crear un nuevo cliente.
- Acceso en una nueva pestaña a la web del Establecimiento del cliente.

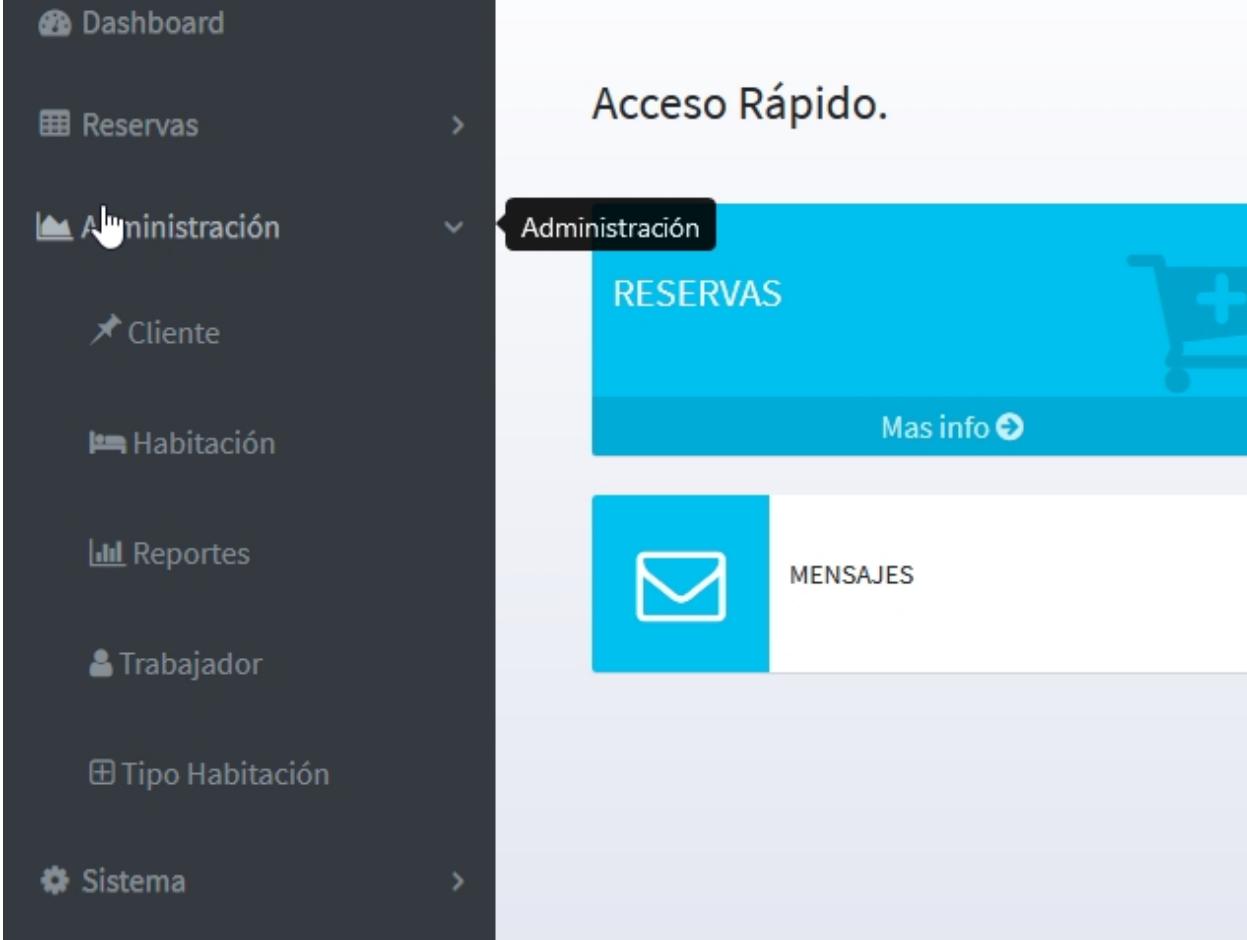


7 Panel lateral

Desde el Panel Lateral se accede a cualquier punto del sistema, si desplegamos todos sus ítems tendremos acceso a los siguientes Módulos:

- Dashboard
- Reservas
 - ➔ Reservadas
 - ➔ Disponibilidad
- Administración
 - ➔ Cliente
 - ➔ Habitación
 - ➔ Reportes
 - ➔ Trabajador
 - ➔ Tipo de Habitación
- Sistema
 - ➔ Usuarios
 - ➔ Roles
 - ➔ Parámetros

GestHotel v1.0.1



Detalle del Panel Lateral

8 Dashboard

Acceso directo a la pantalla principal de la aplicación.

9 Reservas

Despliega dos ítems:

- ➔ Reservadas
- ➔ Disponibilidad

9.1 Reservadas

Listado de habitaciones reservadas con los siguientes datos de cada una de ellas:

- Número de habitación.
- Tipología.
- Nombre y apellidos del cliente.
- Fecha y hora de ingreso.
- Fecha y hora de salida.
- Total días de la estancia .
- Días pendientes.
- Estado de pago.
 - Pendiente .
 - Cancelado.
- Total a pagar.
- Entregas a cuenta.
- Pendiente de abonar.
- Acceso al cobro total o parcial (Símbolo Dólar '\$')
- Acceso para liberar la habitación.

La tabla nos permite paginar los resultados en 10, 25, 50 o 100 registros.

También dispone de un buscador que filtra cualquier criterio.

GestHotel v1.0.1

admin admin admin Logout

Dashboard >

Reservas >

Administración >

Sistema >

LISTADO DE RESERVAS

Mostrar 10 registros Buscar:

Hab.	Tipo	Cliente	Ingreso	Salida	Total Días	Días Ptos	Estado	Total	A Cta.	Pte.	Liberar
# 106	Individual	Pedro Aponte Treviño	2017-12-12 17:49:27	2017-12-13 15:00:00	1	+0 días	CANCELADO	45	45	0	\$
# 108	Triple	Alma Arreola Castro	17-12-2017 17:51:15	24-12-2017 15:00:00	7	+11 días	PENDIENTE	630€	0	630	\$
# 110	Doble	Carmen Lebrón Delrio	12-12-2017 17:50:03	19-12-2017 15:00:00	7	+6 días	PENDIENTE	420€	0	420	\$
# 201	Doble	Gonzalo Toro Fariñas	12-12-2017 17:50:39	14-12-2017 15:00:00	2	+1 días	PENDIENTE	120€	0	120	\$
# 205	KingSize	Pedro Lugo Carvajal	13-12-2017 17:52:11	17-12-2017 15:00:00	4	+4 días	PENDIENTE	480€	0	480	\$

Mostrando registros del 1 al 5 de un total de 5 registros

Anterior Siguiente

Copyright © Todos los Derechos Reservados 2017

Vista del listado de reservas

COBRO

Personas 3 : 90 €(Euros)
Días : 7

TOTAL	Pago a cuenta	Pago pendiente	Total a Cobrar
630	0	630	630

Cobrar Cancelar

Hab.	Tipo	Cliente	Ingreso	Salida	Total Días	Días Ptos	Estado	Total	A Cta.	Pte.	Liberar
# 106	Individual	Pedro Aponte Treviño	2017-12-12 17:49:27	2017-12-13 15:00:00	1	+0 días	CANCELADO	45	45	0	\$
# 108	Triple	Alma Arreola Castro	17-12-2017 17:51:15	24-12-2017 15:00:00	7	+11 días	PENDIENTE	630€	0	630	\$
# 110	Doble	Carmen Lebrón Delrio	12-12-2017 17:50:03	19-12-2017 15:00:00	7	+6 días	PENDIENTE	420€	0	420	\$
# 201	Doble	Gonzalo Toro Fariñas	12-12-2017 17:50:39	14-12-2017 15:00:00	2	+1 días	PENDIENTE	120€	0	120	\$
# 205	KingSize	Pedro Lugo Carvajal	13-12-2017 17:52:11	17-12-2017 15:00:00	4	+4 días	PENDIENTE	480€	0	480	\$

Mostrando registros del 1 al 5 de un total de 5 registros

Anterior Siguiente

Vista del modal para realizar cobros totales o parciales

9.2 Disponibilidad

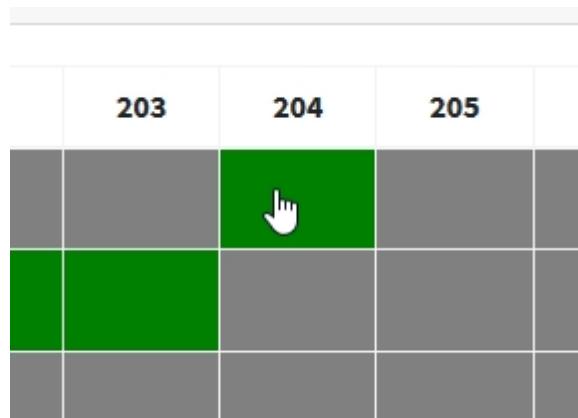
Panel que nos muestra de un vistazo las habitaciones disponibles (en verde) y las ocupadas (en rojo)

The screenshot shows the 'DISPONIBILIDAD' (Availability) section of the GestHotel v1.0.1 interface. On the left, there's a sidebar with navigation links: 'Dashboard', 'Reservas', 'Administración', and 'Sistema'. At the top right, it says 'admin admin admin' and has a 'Logout' button. The main area is titled 'DISPONIBILIDAD' and contains a grid. The columns are labeled with room numbers: 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 201, 202, 203, 204, 205, 206, 207, and 308. The rows represent room types, indicated by icons: a single person, two people, three people, a couple, and a group. Green squares indicate available rooms, while red squares indicate occupied rooms. A copyright notice at the bottom reads 'Copyright © Todos los Derechos Reservados 2017'.

Vista del panel de Disponibilidad

La primera columna nos indica la tipología de la habitación, y fila superior el numero de habitación.

Es posible acceder directamente a una Nueva Reserva cliqueando sobre el cuadro verde de la habitación libre.



En tal caso, la aplicación desplegará el siguiente formulario:

The screenshot shows the 'NUEVA RESERVA' (New Reservation) form within the GestHotel v1.0.1 application. The left sidebar shows navigation links for Dashboard, Reserves, Administration, and System. The main form area has the title 'NUEVA RESERVA'. It contains the following fields:

- Habitación N°: 104
- Tipo de Habitación: Doble
- Descripción: Uso doble o individual. Dos camas de 105 cm individuales independientes.
- Selección Precio:
 - Personas 2 : 60€
 - Personas 1 : 50€
- A cuenta: 0
- Total: 0
- Saldo: 0
- Cliente: +
- Notas
- Ingreso: Diciembre, 2017 (calendar showing dates from 3 to 31)
- Salida: Diciembre, 2017 (calendar showing dates from 3 to 31)
- Buttons: Guardar (Save) and Cancelar (Cancel)

Vista del Formulario Nueva Reserva

El formulario Nueva Reserva nos muestra:

- El número de habitación que vamos a reservar.
- La tipología de la habitación.
- Una descripción de la misma.
- Uno o varios check para seleccionar el número de personas que ocuparan la habitación.
- Un campo para ingresar si el cliente entrega alguna cantidad a cuenta.
- Un campo para ingresar el nombre y apellidos del cliente.
- Acceso directo para crear un nuevo cliente.
- Un campo para realizar anotaciones extras, como por ejemplo algún requisito que nos haga el cliente.
- Un campo de tipo datapicker para ingresar la fecha de entrada.
- Y otro campo para ingresar la fecha de salida.

A tener en cuenta:

- El check para seleccionar el numero de personas que ocuparan la habitación, es obligatorio.
- Deberá crear el precio por habitación y persona desde el menú administración/tipo habitación.
- El campo ingresar alguna cantidad a cuenta, solo permite dígitos numéricos.
- Para ingresar el nombre y apellidos del cliente, solamente debe escribir las 3 primeras letras de su nombre o de alguno de sus apellidos y automáticamente le aparecerá las coincidencias con los clientes registrados en el sistema.
- Si es un cliente nuevo, podrá darlo de alta por el procedimiento normal, desde menu/administración/cliente/nuevo, pero le será mas sencillo simplemente cliquear el símbolo 'mas' que aparece junto al campo cliente del formulario que estamos tratando y le aparecerá un formulario para dar de alta el nuevo cliente.
- Al escoger una fecha de ingreso, el sistema no le permitirá ingresar fechas inferiores al día que esté realizando la reserva.
- Al escoger una fecha de salida, el sistema le permitirá escoger como primer día, el posterior a la fecha de ingreso.
- Para ejecutar la reserva, simplemente haga click sobre el botón verde con la leyenda Guardar.
- Si desea descartar la reserva o simplemente salir del formulario, haga click sobre el botón rojo con la leyenda Cancelar.

The screenshot shows a modal dialog box titled "Nuevo Cliente" (New Client). The form contains the following fields:

- Nombre* (Name): Text input field.
- Apellido1* (Last Name 1): Text input field.
- Apellido2* (Last Name 2): Text input field.
- DNI* (DNI): Text input field.
- Teléfono* (Phone): Text input field.
- Email* (Email): Text input field.
- Dirección (Address): Text input field.
- Provincia (Province): Text input field.
- País (Country): Text input field.

At the bottom of the form is a green "Guardar" (Save) button. Below the form are two date pickers:

- Ingreso (Arrival): Set to "Diciembre, 2017".
- Salida (Departure): Set to "Diciembre, 2017".

Both date pickers show the days of the week: Lu, Ma, Mi, Ju, Vi, Sa, Do. The modal is centered over a larger background form which is partially visible on the left, showing fields for room number (Habitación N°: 104), type (Uso doble o simple), price selection (Opciones Precio), and guest count (Personas 2: 60€, Personas 1: 50€). A "Reservar" (Reserve) button is also visible on the background form.

Modal Nuevo cliente

En el caso de clicar en una habitación ocupada, nos redirigirá al listado reservadas pero filtrando el resultado a la habitación seleccionada.

10 Administración

Despliega cinco ítems:

- ➔ Cliente
- ➔ Habitación
- ➔ Reportes
- ➔ Trabajadores
- ➔ Tipo de Habitación

10.1 Cliente

Listado de clientes ingresados en la base de datos.

De cada uno de ellos se muestra:

- Nombre
- Apellido1
- Apellido2
- DNI
- Teléfono
- Email
- Acceso a Más detalles
- Acceso a Editar el cliente
- Acceso a eliminar el cliente

El listado nos proporciona un paginador para 10, 25, 50 o 100 registros.

Disponemos de un buscador para filtrar cualquier criterio.

Disponemos de un botón de acceso directo para crear un nuevo cliente.

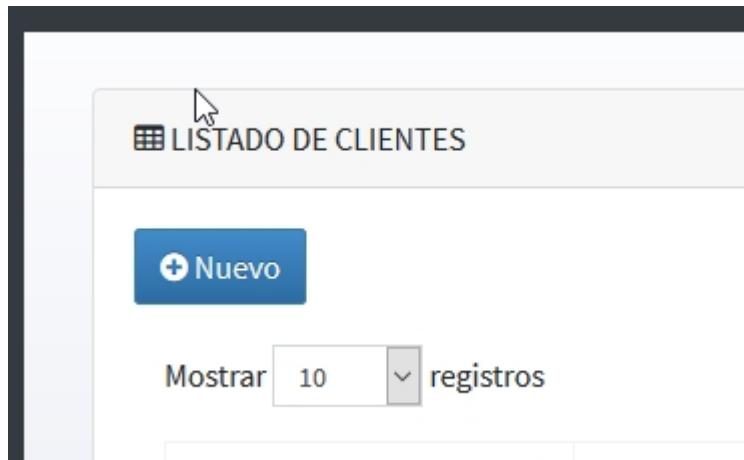
Nombre	Apellido1	Apellido2	DNI	Teléfono	Email				
Aaron	Galindo	Gil	31323948H	688047913	martina.altamirano@terra.com				
Adam	Valero	Rascón	01087415K	664581066	ismael81@zayas.com				
Adriana	Roque	Pardo	97728370J	684905340	lala.osorio@hotmail.es				
Adrián	Molina	Ramón	3924112Y	662854843	jurado.alba@hispanvista.com				
Alma	Arreola	Castro	03739289H	672231008	sandra.sisneros@latinmail.com				
Andrés	Baca	Rocha	81747589T	692928697	xdomenich@collado.org				
Andrés	Tijerina	Pozo	98453297A	621642127	mario27@live.com				
Anna	Terán	Guerra	23074374S	648015954	hugo.benitez@live.com				
Aya	Gómez	Delgado	34975446J	627520460	olimon@perea.net				
Candela	Dávila	Leiva	09917788G	610571432	moreno.helena@hotmail.com				

Mostrando registros del 1 al 10 de un total de 41 registros

Anterior 1 2 3 4 5 Siguiente

Vista listado de clientes

Detalle del buscador y de los accesos a Mas detalle, Editar y Eliminar



Detalle del paginador y acceso a crear un cliente nuevo.

Nombre	Primer Apellido	Segundo Apellido
Aaron	Galindo	Gil

DNI	Teléfono
31323948H	688047913

Dirección	Provincia
Travesía Francisco, 824, Ático 8º, 11793, As Mora de Arriba	Valencia

Vista Detalle del Cliente seleccionado

GestHotel v1.0.1

admin admin admin Logout

- Dashboard
- Reservas
- Administración
- Sistema

EDITAR CLIENTE

Nombre	Primer Apellido	Segundo Apellido
Aaron	Galindo	Gil
DNI	Teléfono	Email
31323948H	688047913	martina.altamirano@terra.com
Dirección	Provincia	País
Travesía Francisco, 824, Ático 8º, 11793, As Mora de Arriba	Valencia	España

Guardar Cancelar

Copyright © Todos los Derechos Reservados 2017

Vista Editar Cliente seleccionado

Apellido2	¿Seguro que desea eliminar este registro?		Email
Gil			3 martina.altamira
Rascón	Aceptar Cancelar		6 ismael81@zaya
Pardo	97728370J	684905340	laia.osorio@hot
Ramón	39243112Y	662854843	jurado.alba@his
Castro	03739289H	672231008	sandra.sisneros@

Modal de confirmación previo a Eliminar el cliente seleccionado

GestHotel v1.0.1

admin admin admin Logout

Dashboard Reservas Administración Sistema

NUEVO CLIENTE

Nombre	Primer Apellido	Segundo Apellido
<input type="text"/>	<input type="text"/>	<input type="text"/>
DNI	Teléfono	Email
<input type="text"/>	<input type="text"/>	<input type="text"/>
Dirección	Provincia	País
<input type="text"/>	<input type="text"/>	<input type="text"/>

Guardar Cancelar

Copyright © Todos los Derechos Reservados 2017

Formulario Nuevo cliente.

A tener en cuenta:

- No se aceptan caracteres distintos del alfabeto Español.
- Se pueden incluir tildes y diéresis.
- Los campos del formulario tendrán una extensión mínima de 3 caracteres y una máxima de 15, excepto en la dirección que no existen límites.
- La letra del dni ha de ir en mayúsculas.
- Estos patrones son aplicables al formulario Nuevo Cliente que se lanza desde una Nueva Reserva.

10.2 Habitación

Listado de las habitaciones del Establecimiento.

Nos muestra:

- Número de habitación.
- Tipología de la habitación.
- Acceso a Editar las ya existentes.
- Acceso a Eliminar la habitación seleccionada.

El listado nos ofrece paginador de 10, 25, 50 o 100 registros y buscador para filtrar por cualquier criterio.

Disponemos de un botón de acceso directo para crear una Nueva Habitación.

Número de Habitación	Tipo de Habitación	T1	T1	T1
101	Matrim			
102	Triple			
103	Matrim			
104	Doble			
105	Suites			
106	Individual			
107	Doble			
108	Triple			
109	Suites			
110	Doble			

Vista del listado de Habitaciones

GestHotel v1.0.1

admin admin admin Logout

Dashboard Reservas Administración Sistema

NUEVA HABITACIÓN

Número de Habitación

Tipo Habitación

Guardar Cancelar

Copyright © Todos los Derechos Reservados 2017

Vista para crear una nueva habitación

A tener en cuenta:

- El Número de habitación ha de tener tres cifras, o sea, una cifra comprendida entre el 100 y el 999.
- El desplegable Tipo Habitación, mostrará los tipos de habitaciones que hayamos creado anteriormente.

GestHotel v1.0.1

admin admin admin Logout

Dashboard Reservas Administración Sistema

EDITAR HABITACIÓN

Número de Habitación

Tipo Habitación

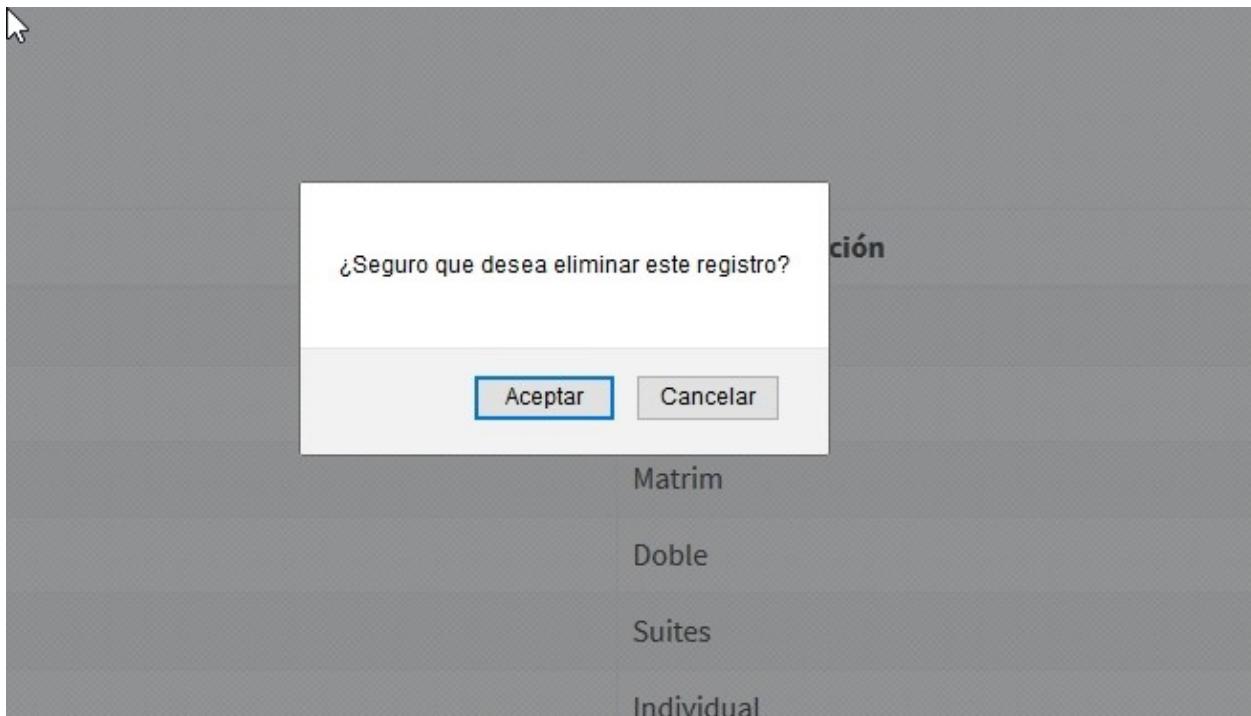
Guardar Cancelar

Copyright © Todos los Derechos Reservados 2017

Vista Editar la habitación seleccionada

A tener en cuenta:

- El número de habitación no es posible modificarlo.
- En el desplegable Tipo de Habitación, nos aparecen los tipos que hayamos dado de alta en el sistema.



Modal previo a eliminar la habitación seleccionada

10.3 Trabajador

Listado de trabajadores ingresados en la base de datos.

De cada uno de ellos se muestra:

- Nombre
- Apellido1
- Apellido2
- DNI
- Teléfono
- Email
- Acceso a Más detalles
- Acceso a Editar el trabajador
- Acceso a Eliminar el trabajador

El listado nos proporciona un paginador para 10, 25, 50 o 100 registros.

Disponemos de un buscador para filtrar cualquier criterio.

Disponemos de un botón de acceso directo para crear un nuevo cliente.

Nombre	Apellido1	Apellido2	DNI	Teléfono	Email	Acción	Acción	Acción	Acción
ANGELA	PEREZ	PEREZ	44455566F	666000111	angela@hotel.com				
Aya	Esteve	Barrios	48422030S	653583737	miguel.zayas@yahoo.es				
Claudia	Mejía	Espino	22984418N	629804939	gael61@yahoo.com				
Diego	Soler	Linares	97374975J	604710360	bruno98@benito.com				
Gael	Miramontes	Rey	56366197M	680356232	odavia@live.com				
Guillem	Quintero	Arias	72612607C	636925145	pmateo@medoza.com				
Iker	Pedraza	Valero	00204965X	67958855	jon70@gimeno.es				
Isabel	Laureano	Téllez	24418453C	649469994	zepeda.nayara@mas.com.es				
Izan	Delarosa	Carrión	74413786T	676020132	jbermudez@lopez.es				
Lucía	Duran	Bustos	18220062Z	631460939	apalacios@terra.com				

Vista listado de trabajadores

GestHotel v1.0.1

admin admin admin Logout

Dashboard Reservas Administración Sistema

DETALLE TRABAJADOR

Nombre ANGELA	Primer Apellido PEREZ	Segundo Apellido PEREZ
DNI 44455566F	Teléfono 666000111	Email angela@hotel.com
Dirección calle Estafeta nº5 Almendralejo	Provincia Badajoz	País España

Volver al listado

Copyright © Todos los Derechos Reservados 2017

Vista detalle del trabajador seleccionado

GestHotel v1.0.1

admin admin admin Logout

Dashboard Reservas Administración Sistema

EDITAR CLIENTE

Nombre ANGELA	Primer Apellido PEREZ	Segundo Apellido PEREZ
DNI 44455566F	Teléfono 666000111	Email angela@hotel.com
Dirección calle Estafeta nº5 Almendralejo	Provincia Badajoz	País España

Guardar Cancelar

Copyright © Todos los Derechos Reservados 2017

Vista Editar trabajador seleccionado

Apellido2	Apellido1	DNI	Email
PEREZ	Barrios	111	angela@hotel.c
Espino	Linares	737	miguel.zayas@
Rey		629804939	gael61@yahoo.
		604710360	bruno98@beni
		56366197M	odavila@live.co

Modal de confirmación previo a eliminar el cliente seleccionado

GestHotel v1.0.1

admin admin admin Logout

ALTA TRABAJADOR

Nombre	Primer Apellido	Segundo Apellido
DNI	Teléfono	Email
Dirección	Provincia	País

Guardar Cancelar

Copyright © Todos los Derechos Reservados 2017

Formulario Nuevo trabajador

A tener en cuenta:

- No se aceptan caracteres distintos del alfabeto Español.
- Se pueden incluir tildes y diéresis.
- Los campos del formulario tendrán una extensión mínima de 3 caracteres y una máxima de 15, excepto en la dirección que no existe límites.
- La letra del dni ha de ir en mayúsculas.

10.4 Tipo Habitación

Listado que muestra las tipologías de habitaciones del establecimiento.

El listado nos muestra:

- Nombre (Denominación común de la tipología).
- Descripción.
- Acceso a crear nuevas tipologías.
- Acceso a Editar los tipos guardados.
- Acceso a Eliminar los tipo guardados.

El listado nos ofrece paginador de 10, 25, 50 o 100 registros y buscador para filtrar por cualquier criterio.

The screenshot shows the GestHotel v1.0.1 application. The left sidebar has a dark theme with white text and includes links for Dashboard, Reservas, Administración (which is currently selected), and Sistema. The main content area has a light gray background and displays a table titled 'TIPO DE HABITACIONES'. At the top of this table is a blue button labeled 'Nuevo'. Below it, there are search and filter options: 'Mostrar 10 registros' and a 'Buscar:' input field. The table itself has columns for 'Nombre' and 'Descripción'. It lists six types: Doble, Individual, KingSize, Matrim, Suites, and Triple. Each entry includes a detailed description and two small icons for edit and delete. At the bottom of the table, it says 'Mostrando registros del 1 al 6 de un total de 6 registros'. A navigation bar at the bottom right shows 'Anterior 1 Siguiente'.

Vista del listado de Tipologías

GestHotel v1.0.1

admin admin admin Logout

Dashboard Reservas Administración Sistema

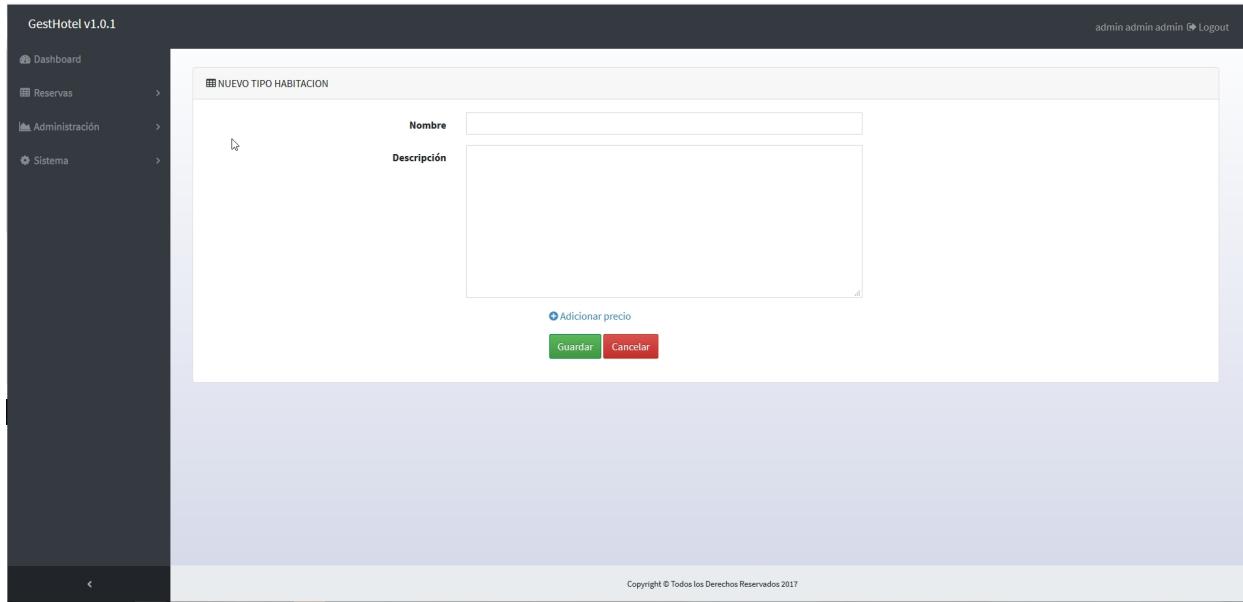
NUEVO TIPO HABITACION

Nombre
Descripción

Adicionar precio

Guardar Cancelar

Copyright © Todos los Derechos Reservados 2017



Vista para crear un nuevo tipo de habitación

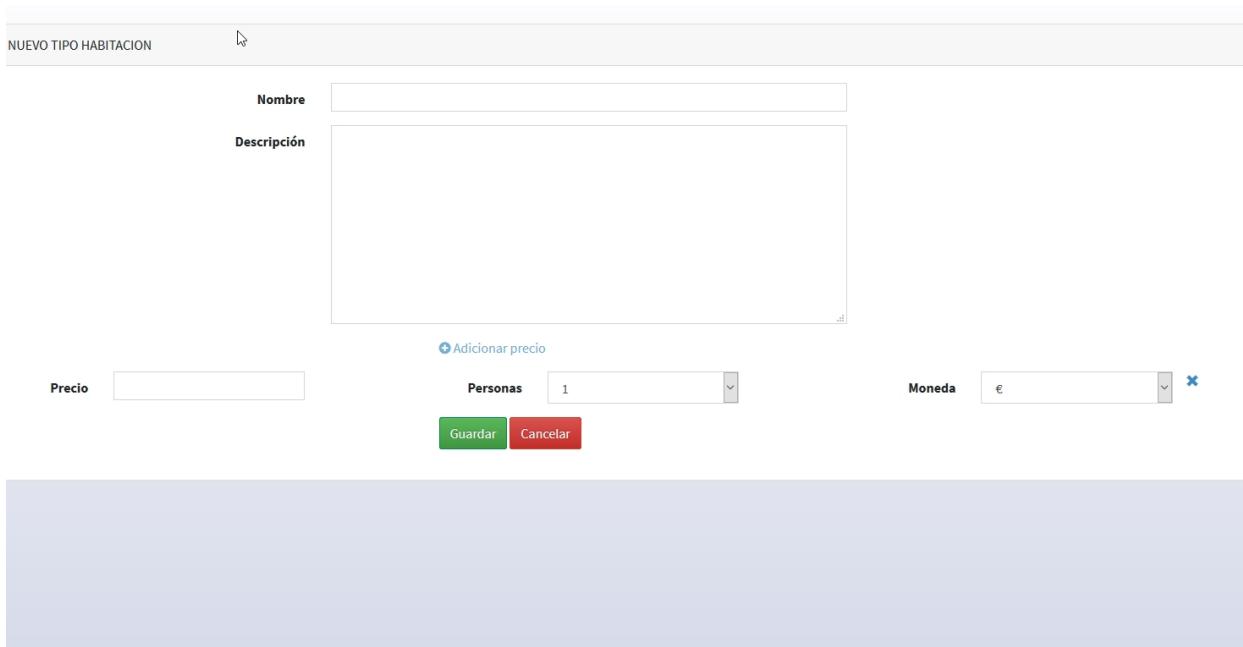
NUEVO TIPO HABITACION

Nombre
Descripción

Adicionar precio

Precio Personas Moneda €

Guardar Cancelar



Detalle para adicionar un precio al tipo de habitación

A tener en cuenta:

- Podemos fijar tantos precios distintos para la misma habitación como queramos.
- Los distintos precios aparecerán en el formulario para generar una reserva.
- Se puede eliminar cualquier precio al listado, pulsando el aspa que aparece al final del precio que queremos eliminar.

GestHotel v1.0.1

admin admin admin Logout

Dashboard Reservas Administración Sistema

EDITAR TIPO HABITACIÓN

Nombre: Doble

Descripción: Uso doble o individual.
Dos camas de 105 cm individuales independientes.

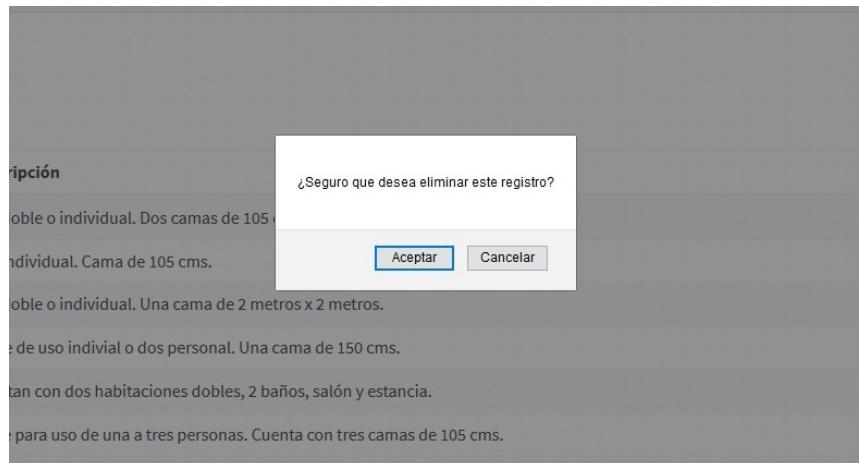
Precio: 60 Personas: 2 Moneda: €

Precio: 50 Personas: 1 Moneda: €

Guardar Cancelar

Copyright © Todos los Derechos Reservados 2017

Vista Editar tipo de habitación seleccionada



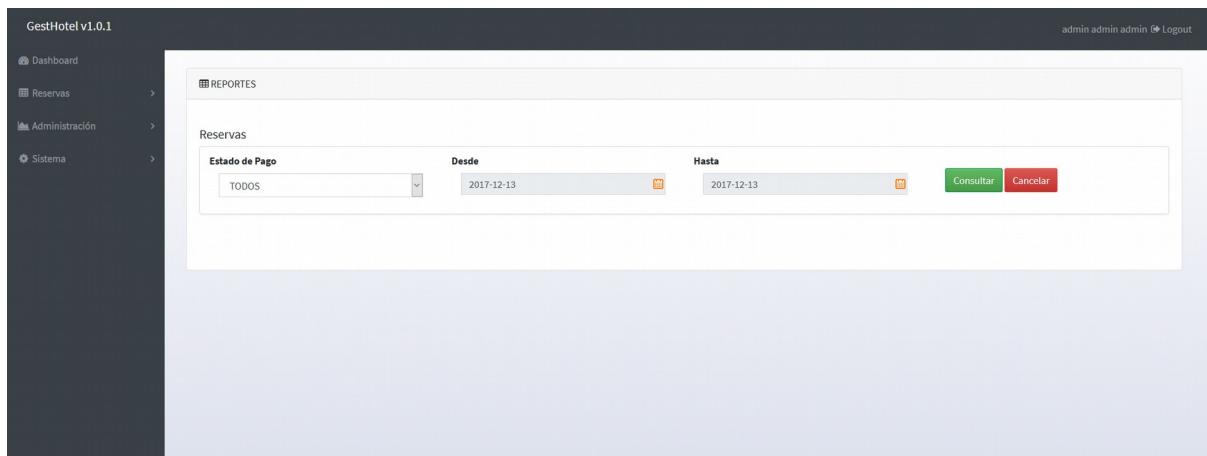
Modal previo a eliminar el tipo de habitación seleccionada

10.5 Reportes

El sistema dispone de un generador de informes de las reservas del establecimiento.

Para ejecutar un informe, previamente deberemos seleccionar:

- Consultar todas las reservas.
- Consultar las reservas pendientes de pago.
- Consultar la reservas que hayan abonado el pago.
- Escoger un rango de fechas.



A continuación, pulsaremos en Consultar y se generará un informe en formato .pdf de las Reservadas con los filtros aplicados anteriormente.

#	Hab	Cliente	Ingresa	Salida	Empleado	Estado	Precio	Días	Pagado	Total
27	106	Adán Valero Rascón	2017-12-02 19:06:24	2017-12-04 15:00:00	Aya Esteve Barrios	CANCELADO	45	2	90€	90
28	106	Pedro Aponte Treviño	2017-12-12 17:49:27	2017-12-13 15:00:00	Aya Esteve Barrios	CANCELADO	45	1	45€	45
29	110	Carmen Lebrón Delrio	2017-12-12 17:50:03	2017-12-19 15:00:00	Aya Esteve Barrios	PENDIENTE	60	7	0€	420
30	201	Gonzalo Tor Farias	2017-12-12 17:50:39	2017-12-14 15:00:00	Aya Esteve Barrios	PENDIENTE	60	2	0€	120
31	108	Alma Arreola Castro	2017-12-17 17:51:15	2017-12-24 15:00:00	Aya Esteve Barrios	PENDIENTE	90	7	0€	630
32	205	Pedro Lugo Carvajal	2017-12-13 17:52:11	2017-12-17 15:00:00	Aya Esteve Barrios	PENDIENTE	120	4	0€	480

11 Sistema

Despliega 3 ítems:

- ➔ Usuarios
- ➔ Roles
- ➔ Parámetros

11.1 Usuarios

Listado de Usuarios registrados en el sistema.

De cada uno de ellos se muestra:

- Email con el que se ha registrado.
- Fecha de ingreso en el sistema.
- Nombre y apellidos.
- Rol que ocupa en el sistema.
- Acceso a Editar el usuario.
- Acceso a Eliminar el usuario.

El listado nos proporciona un paginador para 10, 25, 50 o 100 registros.

Disponemos de un buscador para filtrar cualquier criterio.

Disponemos de un botón de acceso directo para crear un nuevo Usuario.

Email	Fecha Creación	Trabajador	Tipo Usuario
angela@hotel.com	2017-11-14	ANGELA PEREZ PEREZ	USUARIO
aya@hotel.com	2017-12-02	Aya Esteve Barrios	USUARIO
maria@hotel.com	2017-11-27	MARIA MONTAÑO PEREZ	USUARIO
peminisa@hotmail.com	2017-11-28	PEDRO MIRANDA NISA	ADMINISTRADOR DE CONTENIDOS

Vista del listado de Usuarios

Email	<input type="text"/>
Contraseña	<input type="password"/>
Trabajador	ELEGIR
Tipo	ELEGIR

Formulario para ingresar un Nuevo Usuario

A tener en cuenta:

- El desplegable Trabajador mostrará aquellos que estén ingresado en el sistema.
- El desplegable Tipo, mostrará los roles dados de alta en el sistema.

GestHotel v1.0.1

admin admin admin Logout

Dashboard Reservas Administración Sistema

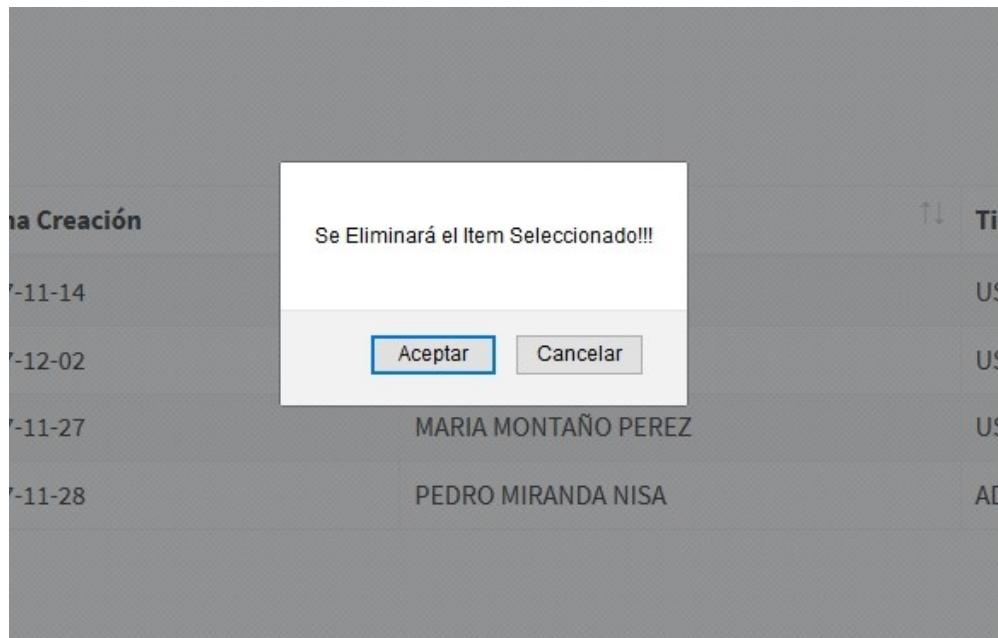
EDITAR USUARIO

Email	angela@hotel.com
Contraseña	<input type="password"/>
Tipo	USUARIO
Trabajador	ANGELA PEREZ PEREZ

Guardar Cancelar

Copyright © Todos los Derechos Reservados 2017

Formulario Editar Usuario



Modal de confirmación previo a eliminar el Usuario seleccionado

11.2 Roles

Listado de Roles dados de alta en el sistema

De cada uno de ellos se muestra:

- Nombre o nomenclatura con el que se distinguirá.
- Descripción del alcance de cada de ellos.
- Acceso a Editar el rol
- Acceso a eliminar el rol

El listado nos proporciona un paginador para 10, 25, 50 o 100 registros.

Disponemos de un buscador para filtrar cualquier criterio.

Disponemos de un botón de acceso directo para crear un nuevo rol.

Nombre	Descripción
ADMINISTRADOR DE CONTENIDOS	Tiene un acceso total a la administración del panel. Acceso restringido a la configuración del sistema.
USUARIO	Puede acceder al sistemas de reservas. Crear y editar clientes. Consultar datos de trabajadores. Realizar reportes.

Listado de Roles

GestHotel v1.0.1

admin admin admin Logout

Dashboard Reservas Administración Sistema

NUEVO TIPO USUARIO

Nombre

Descripción

Guardar Cancelar

Copyright © Todos los Derechos Reservados 2017

This screenshot shows the 'Nuevo TIPO USUARIO' (New User Type) form. It features two input fields: 'Nombre' (Name) and 'Descripción' (Description). Below the fields are 'Guardar' (Save) and 'Cancelar' (Cancel) buttons. The page includes a header with the application name and user information, and a footer with copyright details.

Formulario Nuevo Rol

GestHotel v1.0.1

admin admin admin Logout

Dashboard Reservas Administración Sistema

NUEVO USUARIO

Nombre

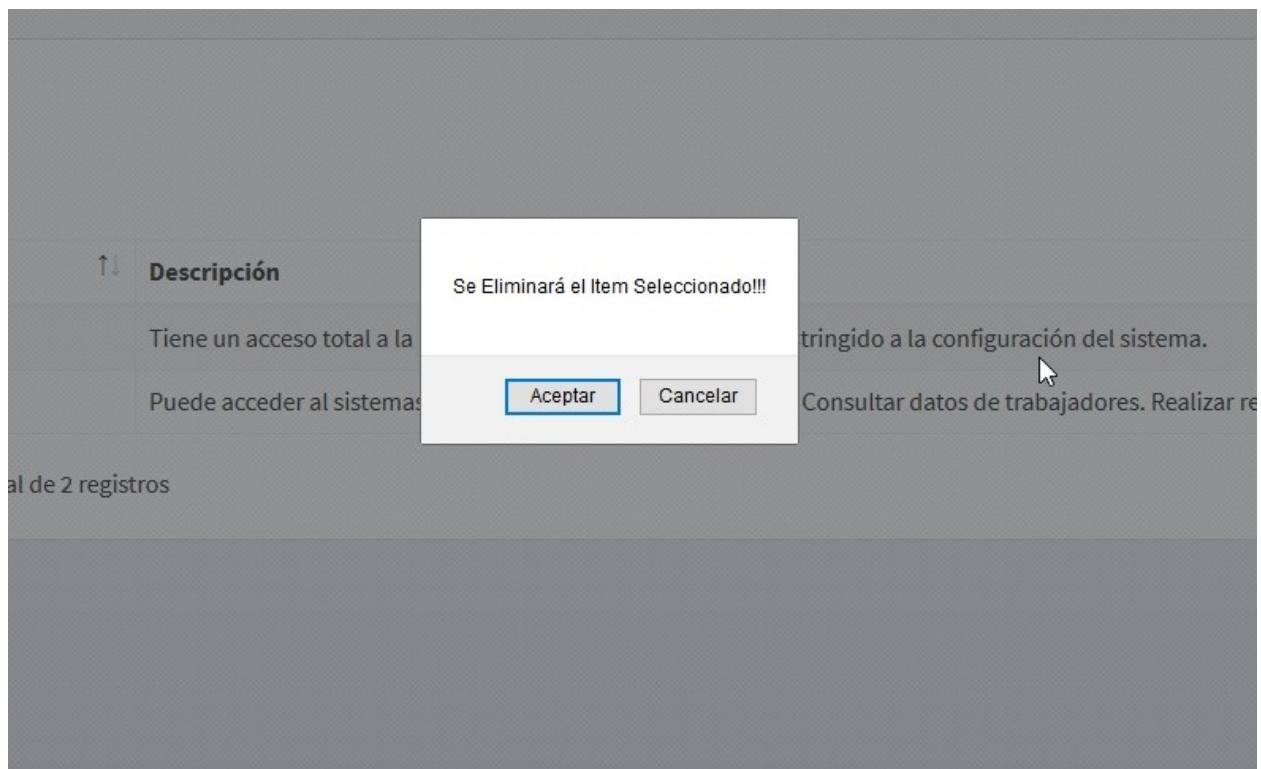
Descripción

Guardar Cancelar

Copyright © Todos los Derechos Reservados 2017

This screenshot shows the 'Nuevo USUARIO' (New User) form. It has fields for 'Nombre' (Name) and 'Descripción' (Description). The 'Nombre' field is filled with 'ADMINISTRADOR DE CONTENIDOS'. The 'Descripción' field contains a detailed description of the user's access rights. The form includes standard save and cancel buttons.

Formulario Editar el rol seleccionado



Modal de confirmación previo a eliminar el rol seleccionado

11.3 Parámetros

Moneda

Listado con las monedas dadas de alta en el sistema
de cada una de ellas se muestra:

- Nombre o denominación de la misma
- País en que se utilización
- Símbolo que se le asocia.
- Acceso a Editar la moneda
- Acceso a eliminar la moneda

El listado nos proporciona un paginador para 10, 25, 50 o 100 registros.

Disponemos de un buscador para filtrar cualquier criterio.

Disponemos de un botón de acceso directo para crear un nuevo cliente.

The screenshot shows the 'Moneda' (Currency) list page of the GestHotel v1.0.1 software. The page has a dark header with the title 'GestHotel v1.0.1' and a user menu 'admin admin admin Logout'. On the left is a sidebar with navigation links: 'Dashboard', 'Reservas', 'Administración', and 'Sistema'. The main content area has tabs 'Parametros de negocio' and 'Moneda' (which is selected). Below is a table titled 'LISTADO DE MONEDAS' with one row. The table columns are 'Nombre', 'País', and 'Símbolo'. The row contains 'Euros', 'España', and '€'. At the bottom of the table are edit and delete icons. A 'Nuevo' (New) button is located at the top left of the table area. The footer includes the URL 'localhost/GestHotel/public/sistema/parametros/moneda' and the copyright notice 'Copyright © Todos los Derechos Reservados 2017'.

Listado de monedas dadas de alta en el sistema

GestHotel v1.0.1

admin admin admin Logout

Dashboard Reservas Administración Sistema

Parametros de negocio Moneda

NUEVA MONEDA

Volver

Nombre:

País:

Símbolo:

Guardar Cancelar

Copyright © Todos los Derechos Reservados 2017

This screenshot shows the 'Nuevo Moneda' (New Currency) creation form. It includes fields for the currency's name ('Nombre'), country ('País'), and symbol ('Símbolo'). The 'Nombre' field contains 'NUEVA MONEDA'. The 'País' field contains 'España'. The 'Símbolo' field contains '€'. There are 'Guardar' (Save) and 'Cancelar' (Cancel) buttons at the bottom.

Formulario Nueva Moneda

GestHotel v1.0.1

admin admin admin Logout

Dashboard Reservas Administración Sistema

Parametros de negocio Moneda

NUEVA MONEDA

Volver a Listado

Nombre: Euros

País: España

Símbolo: €

Guardar Cancelar

localhost/GestHotel/public/sistema/parametros/moneda/5/edit

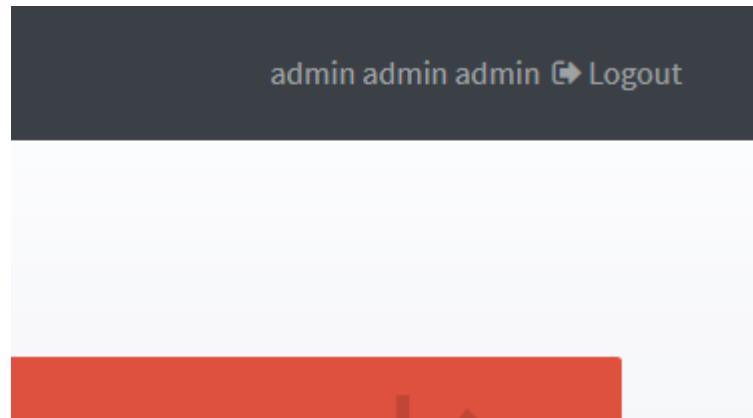
Copyright © Todos los Derechos Reservados 2017

This screenshot shows the 'Nuevo Moneda' (New Currency) creation form. The 'Nombre' field is populated with 'Euros', the 'País' field with 'España', and the 'Símbolo' field with '€'. The 'Guardar' (Save) and 'Cancelar' (Cancel) buttons are visible at the bottom.

Formulario Editar la moneda seleccionada

12 Logout

Para salir del sistema, simplemente pulse sobre su nombre de Usuario, que aparece en la parte superior derecha del panel de administración.



GestHotel

Manual del Administrador de Contenidos

Versión: 0100

Fecha: 13/12/2017

1 Descripción del Sistema

Objeto.

En este manual encontrarás toda la información necesaria para la utilización de la aplicación **GestHotel** de una forma sencilla e intuitiva.

¿Qué es GestHotel?

GestHotel es una aplicación cuya misión es centralizar la gestión de un hotel.

En resumen, la aplicación permite gestionar el alta de clientes, reservas de habitaciones, checkin y checkout, reportes, etc.

2 Configuración previa

El dispositivo en el que se va instalar la aplicación, deberá tener acceso a la red, para que la aplicación pueda conectarse con el servicio que le proporciona los datos.

3 Acceso a GestHotel

Simplemente acceda desde cualquier navegador a la url que su Administrador le haya facilitado.

4 Inicio de GestHotel

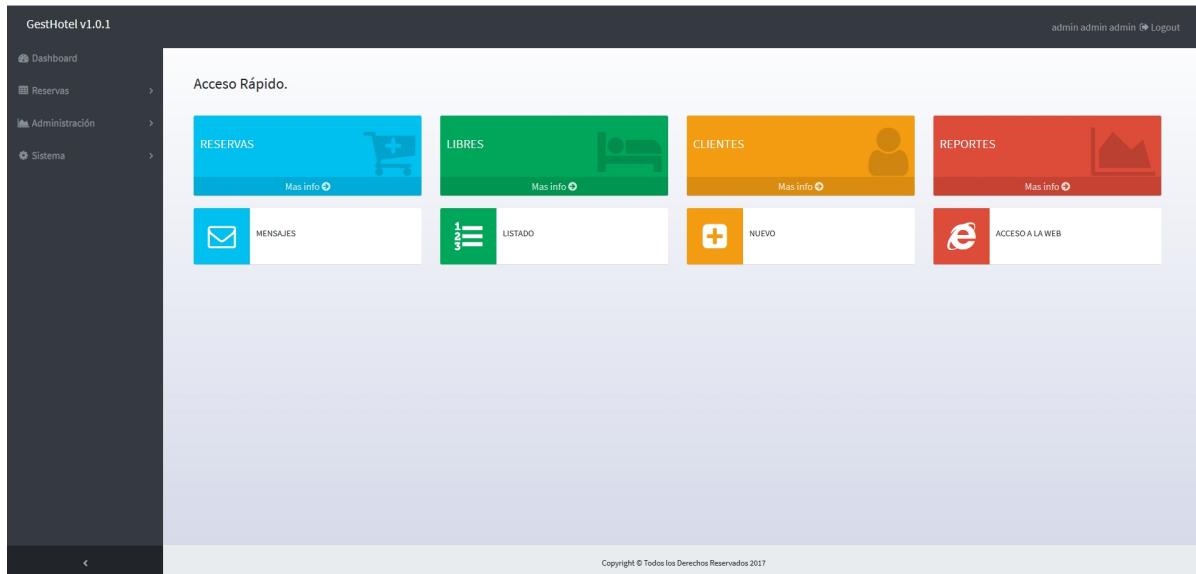
La aplicación desplegará una pantalla de acceso a través de Login con las credenciales que le proporcionará su Administrador.

Una vez ingresado los datos de acceso, si son correctos, dispondrá del panel de administración de GestHotel.

5 Página Principal

La primera pantalla que disponemos, nos proporciona, ademas del acceso a cualquier punto de la aplicación a través del menú lateral, una serie de accesos rápidos que serán los mas utilizados por el cliente:

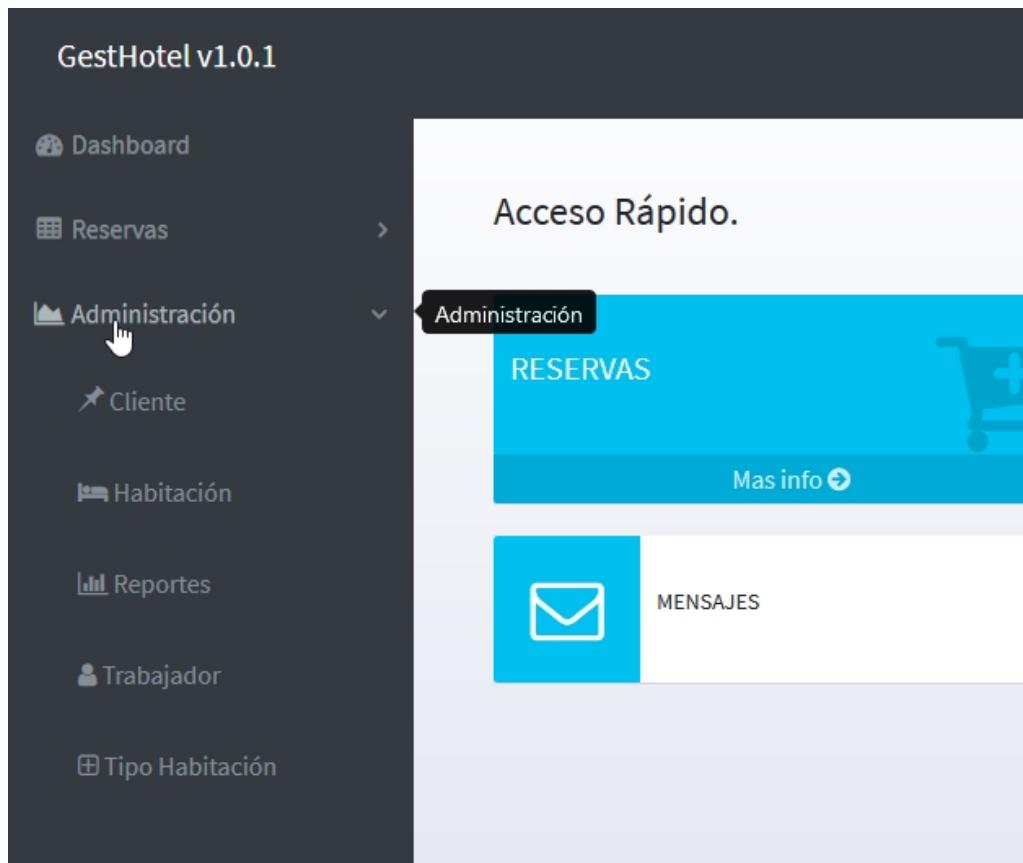
- Acceso al listado de Reservas.
- Acceso a la disponibilidad de Habitaciones.
- Acceso al listado de Clientes.
- Acceso al generador de Informes.
- Acceso al correo electrónico.
- Listado total de habitaciones.
- Crear un nuevo cliente.
- Acceso en una nueva pestaña a la web del Establecimiento del cliente.



6 Panel lateral

Desde el Panel Lateral se accede a cualquier punto del sistema, si desplegamos todos sus ítems tendremos acceso a los siguientes Módulos:

- Dashboard
- Reservas
 - ➔ Reservadas
 - ➔ Disponibilidad
- Administración
 - ➔ Cliente
 - ➔ Habitación
 - ➔ Reportes
 - ➔ Trabajador
 - ➔ Tipo de Habitación



Detalle del Panel Lateral

7 Dashboard

Acceso directo a la pantalla principal de la aplicación.

8 Reservas

Despliega dos ítems:

- ➔ Reservadas
- ➔ Disponibilidad

8.1 Reservadas

Listado de habitaciones reservadas con los siguientes datos de cada una de ellas:

- Número de habitación.
- Tipología.
- Nombre y apellidos del cliente.
- Fecha y hora de ingreso.
- Fecha y hora de salida.
- Total días de la estancia .
- Días pendientes.
- Estado de pago.
 - Pendiente .
 - Cancelado.
- Total a pagar.
- Entregas a cuenta.
- Pendiente de abonar.
- Acceso al cobro total o parcial (Símbolo Dólar '\$')
- Acceso para liberar la habitación.

La tabla nos permite paginar los resultados en 10, 25, 50 o 100 registros.

También dispone de un buscador que filtra cualquier criterio.

GestHotel v1.0.1

admin admin admin Logout

Dashboard >

Reservas >

Administración >

Sistema >

LISTADO DE RESERVAS

Mostrar 10 registros Buscar:

Hab.	Tipo	Cliente	Ingreso	Salida	Total Días	Días Ptos	Estado	Total	A Cta.	Pte.	Liberar
# 106	Individual	Pedro Aponte Treviño	2017-12-12 17:49:27	2017-12-13 15:00:00	1	+0 días	CANCELADO	45	45	0	\$
# 108	Triple	Alma Arreola Castro	17-12-2017 17:51:15	24-12-2017 15:00:00	7	+11 días	PENDIENTE	630€	0	630	\$
# 110	Doble	Carmen Lebrón Delrío	12-12-2017 17:50:03	19-12-2017 15:00:00	7	+6 días	PENDIENTE	420€	0	420	\$
# 201	Doble	Gonzalo Toro Farías	12-12-2017 17:50:39	14-12-2017 15:00:00	2	+1 días	PENDIENTE	120€	0	120	\$
# 205	KingSize	Pedro Lugo Carvajal	13-12-2017 17:52:11	17-12-2017 15:00:00	4	+4 días	PENDIENTE	480€	0	480	\$

Mostrando registros del 1 al 5 de un total de 5 registros

Anterior 1 Siguiente

Copyright © Todos los Derechos Reservados 2017

Vista del listado de reservas

Personas 3 : 90 €(Euros)

Dias : 7

COBRO

TOTAL	Pago a cuenta	Pago pendiente	Total a Cobrar
630	0	630	630

Cobrar Cancelar

Hab.	Tipo	Cliente	Ingreso	Salida	Total Días	Días Ptos	Estado	Total	A Cta.	Pte.	Liberar
# 106	Individual	Pedro Aponte Treviño	2017-12-12 17:49:27	2017-12-13 15:00:00	1	+0 días	CANCELADO	45	45	0	\$
# 108	Triple	Alma Arreola Castro	17-12-2017 17:51:15	24-12-2017 15:00:00	7	+11 días	PENDIENTE	630€	0	630	\$
# 110	Doble	Carmen Lebrón Delrío	12-12-2017 17:50:03	19-12-2017 15:00:00	7	+6 días	PENDIENTE	420€	0	420	\$
# 201	Doble	Gonzalo Toro Farías	12-12-2017 17:50:39	14-12-2017 15:00:00	2	+1 días	PENDIENTE	120€	0	120	\$
# 205	KingSize	Pedro Lugo Carvajal	13-12-2017 17:52:11	17-12-2017 15:00:00	4	+4 días	PENDIENTE	480€	0	480	\$

Mostrando registros del 1 al 5 de un total de 5 registros

Anterior 1 Siguiente

Vista del modal para realizar cobros totales o parciales

8.2 Disponibilidad

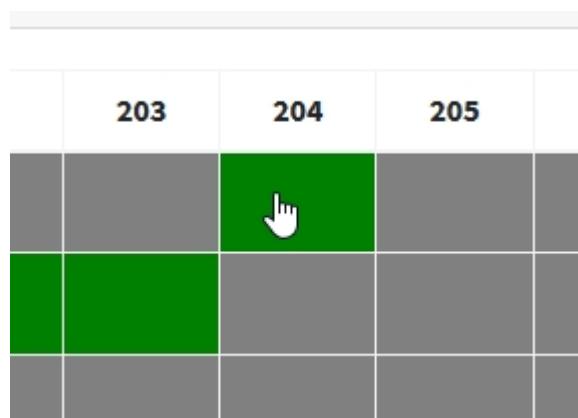
Panel que nos muestra de un vistazo las habitaciones disponibles (en verde) y las ocupadas (en rojo)

The screenshot shows the 'DISPONIBILIDAD' (Availability) section of the GestHotel v1.0.1 interface. On the left, there's a sidebar with navigation links: 'Dashboard', 'Reservas', 'Administración', and 'Sistema'. At the top right, it says 'admin admin admin' and has a 'Logout' button. The main area is titled 'DISPONIBILIDAD' and contains a grid. The columns are labeled with room numbers: 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 201, 202, 203, 204, 205, 206, 207, and 308. The rows represent room types, indicated by icons: a single person (101), two people (102), three people (103), one person and one dog (104), two people and one dog (105), three people and one dog (106), four people (107), five people (108), six people (109), seven people (110), and so on. Green squares indicate available rooms, while red squares indicate occupied rooms. A copyright notice at the bottom reads 'Copyright © Todos los Derechos Reservados 2017'.

Vista del panel de Disponibilidad

La primera columna nos indica la tipología de la habitación, y fila superior el numero de habitación.

Es posible acceder directamente a una nueva reserva cliqueando sobre el cuadro verde de la habitación libre.



En tal caso, la aplicación desplegará el siguiente formulario:

The screenshot shows the 'NUEVA RESERVA' (New Reservation) form. At the top, it displays 'Habitación N°: 104' and 'Tipo de Habitación: Doble'. Below this, there's a description: 'Descripción: Uso doble o individual. Dos camas de 105 cm individuales independientes.' A section for 'Selección Precio:' shows two options: 'Personas 2 : 60€' (selected) and 'Personas 1 : 50€'. The 'A cuenta' field contains '0'. The 'Total' field also contains '0'. The 'Saldo' field contains '0'. There are two date pickers: one for 'Ingreso' (Arrival) set to December 13, 2017, and one for 'Salida' (Departure) set to December 31, 2017. At the bottom, there are 'Guardar' (Save) and 'Cancelar' (Cancel) buttons.

Vista del Formulario Nueva Reserva

El formulario Nueva Reserva nos muestra:

- El número de habitación que vamos a reservar.
- La tipología de la habitación.
- Una descripción de la misma.
- Uno o varios check para seleccionar el número de personas que ocuparan la habitación.
- Un campo para ingresar si el cliente entrega alguna cantidad a cuenta.
- Un campo para ingresar el nombre y apellidos del cliente.
- Acceso directo para crear un nuevo cliente.
- Un campo para realizar anotaciones extras, como por ejemplo algún requisito que nos haga el cliente.
- Un campo de tipo datapicker para ingresar la fecha de entrada.
- Y otro campo para ingresar la fecha de salida.

A tener en cuenta:

- El check para seleccionar el numero de personas que ocuparan la habitación, es obligatorio.
- El campo ingresar alguna cantidad a cuenta, solo permite dígitos numéricos.
- Para ingresar el nombre y apellidos del cliente, solamente debe escribir las 3 primeras letras de su nombre o de alguno de sus apellidos y automáticamente le aparecerá las coincidencias con los clientes registrados en el sistema.
- Si es un cliente nuevo, podrá darlo de alta por el procedimiento normal, desde menu/administración/cliente/nuevo, pero le será mas sencillo simplemente cliquear el símbolo 'mas' que aparece junto al campo cliente del formulario que estamos tratando y le aparecerá un formulario para dar de alta el nuevo cliente.
- Al escoger una fecha de ingreso, el sistema no le permitirá ingresar fechas inferiores al día que esté realizando la reserva.
- Al escoger una fecha de salida, el sistema le permitirá escoger como primer día, el posterior a la fecha de ingreso.
- Para ejecutar la reserva, simplemente haga click sobre el botón verde con la leyenda Guardar.
- Si desea descartar la reserva o simplemente salir del formulario, haga click sobre el botón rojo con la leyenda Cancelar.

Modal Nuevo cliente

En el caso de clicar en una habitación ocupada, nos redirigirá al listado reservadas pero filtrando el resultado a la habitación seleccionada.

9 Administración

Despliega cinco ítems:

- ➔ Cliente
- ➔ Habitación
- ➔ Reportes
- ➔ Trabajadores
- ➔ Tipo de Habitación

9.1 Cliente

Listado de clientes ingresados en la base de datos.

De cada uno de ellos se muestra:

- Nombre
- Apellido1
- Apellido2
- DNI
- Teléfono
- Email
- Acceso a Más detalles
- Acceso a Editar el cliente
- Acceso a eliminar el cliente

El listado nos proporciona un paginador para 10, 25, 50 o 100 registros.

Disponemos de un buscador para filtrar cualquier criterio.

Disponemos de un botón de acceso directo para crear un nuevo cliente.

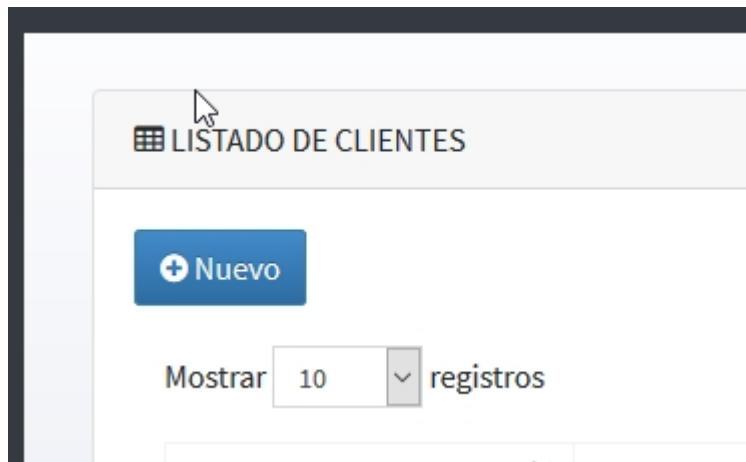
The screenshot shows the 'GestHotel v1.0.1' application. The left sidebar has a dark theme with navigation items: Dashboard, Reservas, Administración, and Sistema. The main content area is titled 'LISTADO DE CLIENTES'. It features a table with columns: Nombre, Apellido1, Apellido2, DNI, Teléfono, Email, and four small icons (edit, delete, etc.). A search bar labeled 'Buscar:' is at the top right. Below the table, it says 'Mostrando registros del 1 al 10 de un total de 41 registros'. At the bottom right is a pagination control with buttons for Anterior, 1, 2, 3, 4, 5, and Siguiente. The footer contains the copyright notice 'Copyright © Todos los Derechos Reservados 2017'.

Nombre	Apellido1	Apellido2	DNI	Teléfono	Email				
Aaron	Galindo	Gil	31323948H	688047913	martina.altamirano@terra.com				
Adam	Valero	Rascón	01087415K	664581066	ismael81@zayas.com				
Adriana	Roque	Pardo	97728370J	684905340	lala.osorio@hotmail.es				
Adrián	Molina	Ramón	39249112Y	662854843	jurado.alba@hispanvista.com				
Alma	Arreola	Castro	03739289H	672231008	sandra.sisneros@latinmail.com				
Andrés	Baca	Rocha	81747589T	692928697	xdomenech@collado.org				
Andrés	Tijerina	Pozo	98453297A	621642127	mario27@live.com				
Anna	Terán	Guerra	23074374S	648015954	hugo.benitez@live.com				
Aya	Gómez	Delgado	34975446J	627520460	olimon@perea.net				
Candela	Dávila	Leiva	099117788G	610571432	moreno.helena@hotmail.com				

Vista listado de clientes

This image shows a close-up of the client list table from the previous screenshot. It highlights the search bar labeled 'Buscar:', the sorting icons (up and down arrows), and the row of action icons (+, edit, delete) for each row.

Detalle del buscador y de los accesos a Mas detalle, Editar y Eliminar



Detalle del paginador y acceso a crear un cliente nuevo.

GestHotel v1.0.1

admin admin admin Logout

DETALLE CLIENTE

Nombre	Primer Apellido	Segundo Apellido
Aaron	Galindo	Gil
DNI	Teléfono	Email
31323948H	688047913	martina.altamirano@terra.com
Dirección	Provincia	País
Travesía Francisco, 824, Ático 8º, 11793, As Mora de Arriba	Valencia	España

Volver al listado

Copyright © Todos los Derechos Reservados 2017

Vista Detalle del Cliente seleccionado

GestHotel v1.0.1

admin admin admin Logout

- Dashboard
- Reservas
- Administración
- Sistema

EDITAR CLIENTE

Nombre	Primer Apellido	Segundo Apellido
Aaron	Galindo	Gil
DNI	Teléfono	Email
31323948H	688047913	martina.altamirano@terra.com
Dirección	Provincia	País
Travesía Francisco, 824, Ático 8º, 11793, As Mora de Arriba	Valencia	España

Guardar **Cancelar**

Copyright © Todos los Derechos Reservados 2017

Vista Editar Cliente seleccionado

Apellido2	¿Seguro que desea eliminar este registro?		Email
Gil			3 martina.altamira
Rascón	Aceptar	Cancelar	6 ismael81@zayas
Pardo	97728370J	684905340	laia.osorio@hot
Ramón	39243112Y	662854843	jurado.alba@his
Castro	03739289H	672231008	sandra.sisneros@

Modal de confirmación previo a Eliminar el Cliente seleccionado

GestHotel v1.0.1

admin admin admin Logout

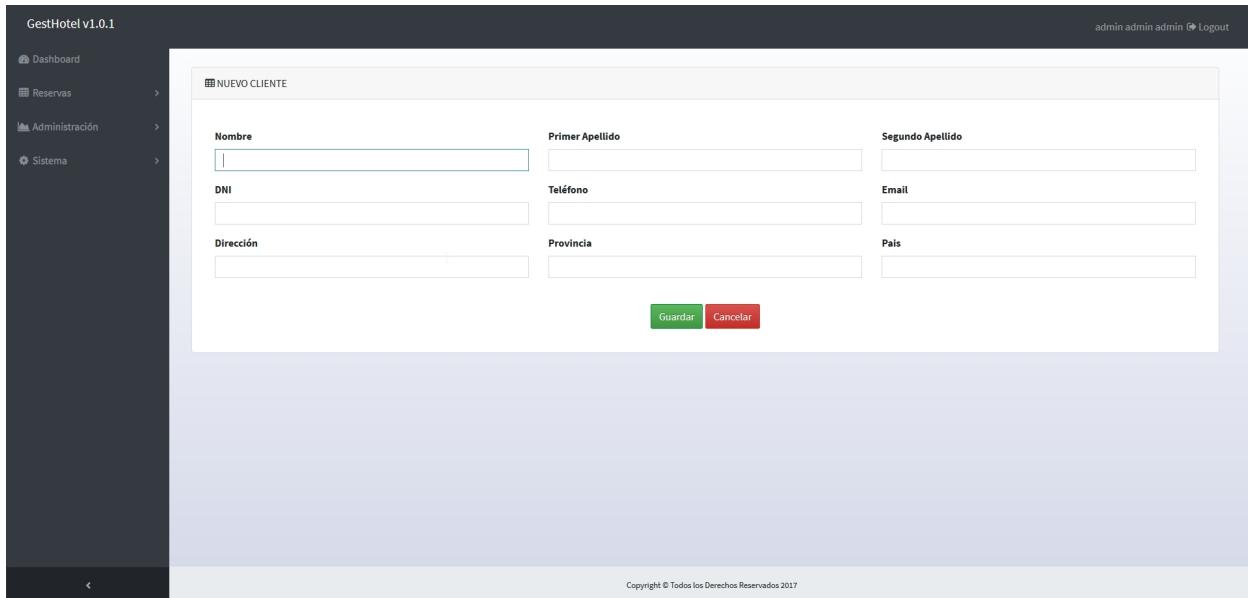
Dashboard Reservas Administración Sistema

NUEVO CLIENTE

Nombre	Primer Apellido	Segundo Apellido
<input type="text"/>	<input type="text"/>	<input type="text"/>
DNI	Teléfono	Email
<input type="text"/>	<input type="text"/>	<input type="text"/>
Dirección	Provincia	País
<input type="text"/>	<input type="text"/>	<input type="text"/>

Guardar Cancelar

Copyright © Todos los Derechos Reservados 2017



Formulario Nuevo cliente.

A tener en cuenta:

- No se aceptan caracteres distintos del alfabeto Español.
- Se pueden incluir tildes y diéresis.
- Los campos del formulario tendrán una extensión mínima de 3 caracteres y una máxima de 15, excepto en la dirección que no existen límites.
- La letra del dni ha de ir en mayúsculas.
- Estos patrones son aplicables al formulario Nuevo Cliente que se lanza desde una Nueva Reserva.

9.2 Habitación

Listado de las habitaciones del Establecimiento.

Nos muestra:

- Número de habitación.
- Tipología de la habitación.
- Acceso a Editar las ya existentes.
- Acceso a Eliminar la habitación seleccionada.

El listado nos ofrece paginador de 10, 25, 50 o 100 registros y buscador para filtrar por cualquier criterio.

Disponemos de un botón de acceso directo para crear una Nueva Habitación.

The screenshot shows the GestHotel v1.0.1 application interface. The left sidebar has a dark theme with white text and icons. It includes sections for Dashboard, Reservas, Administración (with Clientes and Habitación), Reportes, Trabajador, Tipo Habitación, and Sistema. The 'Habitación' section is currently selected. The main content area is titled 'LISTADO DE HABITACIONES'. It features a 'Nuevo' (New) button, a dropdown for 'Mostrar' (Show) with options for 10, 25, 50, or 100 registros, and a search bar labeled 'Buscar:' with a placeholder for text input. Below these are two tables. The first table lists 'Número de Habitación' (Room Number) from 101 to 110 and 'Tipo de Habitación' (Room Type). The second table lists 'Número de Habitación' from 101 to 110 and 'Estado' (Status). At the bottom, there is a footer with copyright information: 'Copyright © Todos los Derechos Reservados 2017'.

Número de Habitación	Tipo de Habitación	T1	T1	T1
101	Matrím			
102	Triple			
103	Matrím			
104	Doble			
105	Suites			
106	Individual			
107	Doble			
108	Triple			
109	Suites			
110	Doble			

Número de Habitación	Estado
101	Activo
102	Activo
103	Activo
104	Activo
105	Activo
106	Activo
107	Activo
108	Activo
109	Activo
110	Activo

Vista del listado de Habitaciones

GestHotel v1.0.1

admin admin admin Logout

Dashboard Reservas Administración Sistema

NUEVA HABITACIÓN

Número de Habitación

Tipo Habitación

Guardar Cancelar

Copyright © Todos los Derechos Reservados 2017

Vista para crear una nueva habitación

A tener en cuenta:

- El Número de habitación ha de tener tres cifras, o sea, una cifra comprendida entre el 100 y el 999.
- El desplegable Tipo Habitación, mostrará los tipos de habitaciones que hayamos creado anteriormente.

GestHotel v1.0.1

admin admin admin Logout

Dashboard Reservas Administración Sistema

EDITAR HABITACIÓN

Número de Habitación

Tipo Habitación

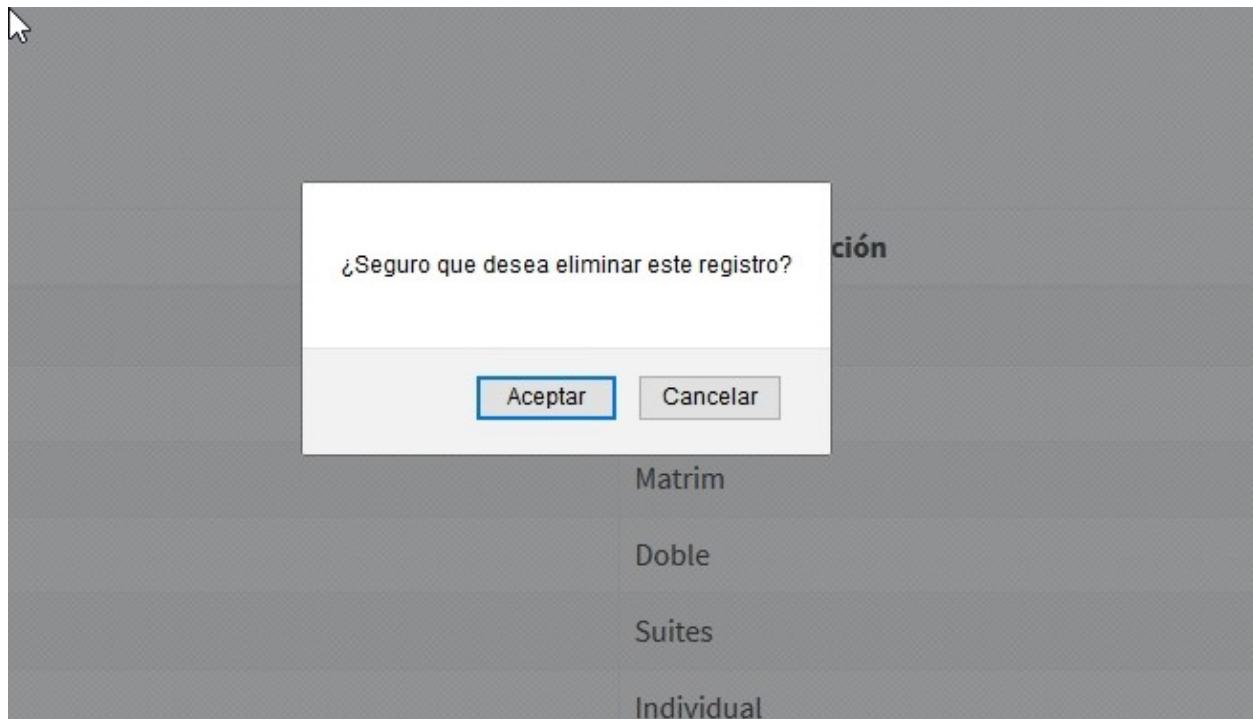
Guardar Cancelar

Copyright © Todos los Derechos Reservados 2017

Vista Editar la habitación seleccionada

A tener en cuenta:

- El número de habitación no es posible modificarlo.
- En el desplegable Tipo de Habitación, nos aparecen los tipos que hayamos dado de alta en el sistema.



Modal previo a eliminar la habitación seleccionada

9.3 Trabajador

Listado de trabajadores ingresados en la base de datos.

De cada uno de ellos se muestra:

- Nombre
- Apellido1
- Apellido2
- DNI
- Teléfono
- Email
- Acceso a Más detalles
- Acceso a Editar el trabajador
- Acceso a eliminar el trabajador

El listado nos proporciona un paginador para 10, 25, 50 o 100 registros.

Disponemos de un buscador para filtrar cualquier criterio.

Disponemos de un botón de acceso directo para crear un nuevo cliente.

Nombre	Apellido1	Apellido2	DNI	Teléfono	Email	Acción	Acción	Acción	Acción
ANGELA	PEREZ	PEREZ	44455566F	666000111	angela@hotel.com				
Aya	Esteve	Barrios	48422030S	653583737	miguel.zayas@yahoo.es				
Claudia	Mejía	Espino	22984418N	629804939	gael61@yahoo.com				
Diego	Soler	Linares	97374975J	604710360	bruno98@benito.com				
Gael	Miramontes	Rey	56366197M	680356232	odavia@live.com				
Guillem	Quintero	Arias	72612607C	636925145	pmateo@mendoza.com				
Iker	Pedraza	Valero	00204365X	679585855	jon70@gimeno.es				
Isabel	Laureano	Téllez	24418453C	649469994	zepeda.nayara@mas.com.es				
Izan	Delarosa	Carrión	74413786T	676020132	jbermudez@lopez.es				
Lucía	Duran	Bustos	18220062Z	631460939	apalacios@terra.com				

Vista listado de trabajadores

GestHotel v1.0.1

admin admin admin Logout

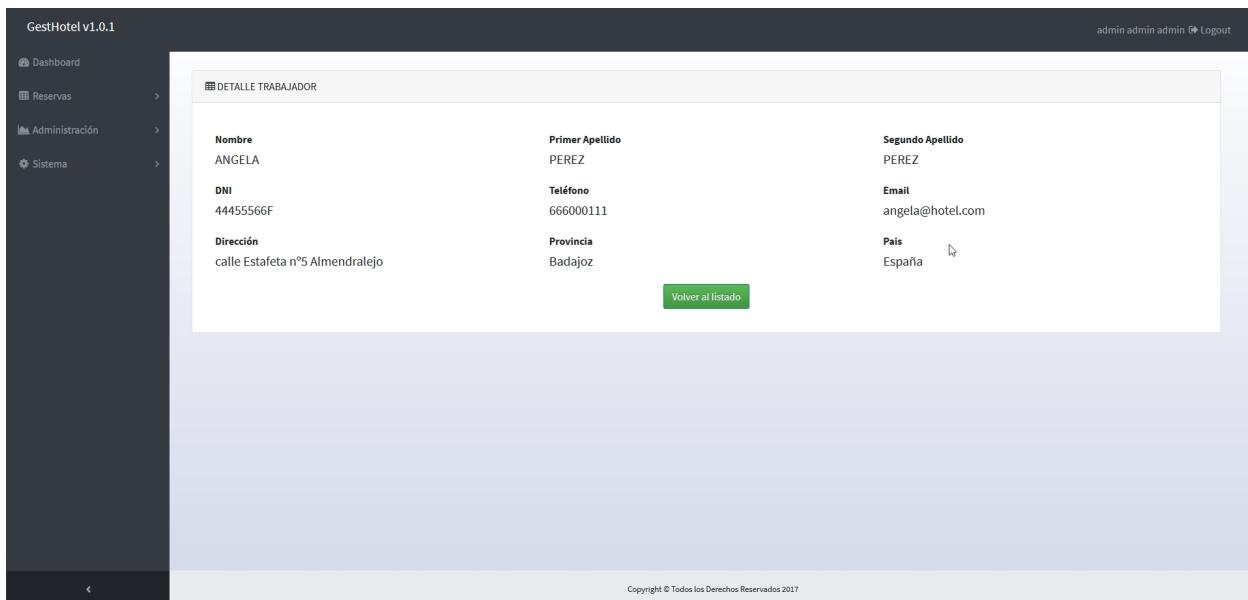
Dashboard Reservas Administración Sistema

DETALLE TRABAJADOR

Nombre ANGELA	Primer Apellido PEREZ	Segundo Apellido PEREZ
DNI 44455566F	Teléfono 666000111	Email angela@hotel.com
Dirección calle Estafeta nº5 Almendralejo	Provincia Badajoz	País España

Volver al listado

Copyright © Todos los Derechos Reservados 2017



Vista detalle del trabajador seleccionado

GestHotel v1.0.1

admin admin admin Logout

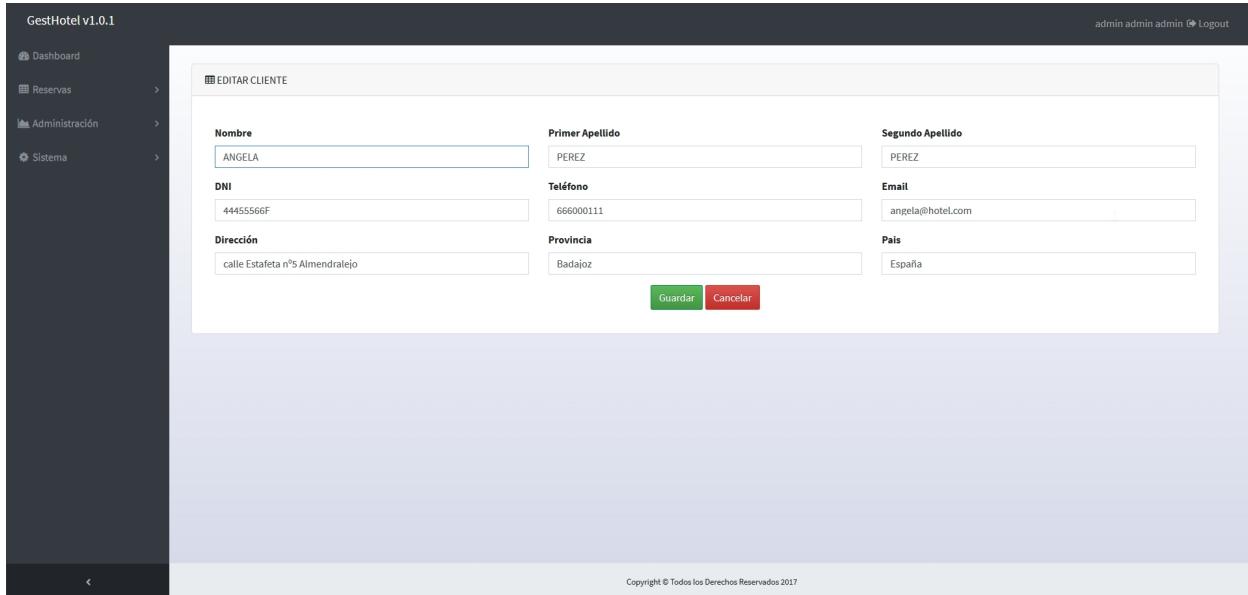
Dashboard Reservas Administración Sistema

EDITAR CLIENTE

Nombre ANGELA	Primer Apellido PEREZ	Segundo Apellido PEREZ
DNI 44455566F	Teléfono 666000111	Email angela@hotel.com
Dirección calle Estafeta nº5 Almendralejo	Provincia Badajoz	País España

Guardar Cancelar

Copyright © Todos los Derechos Reservados 2017



Vista Editar trabajador seleccionado

Apellido2	Email		
PEREZ	111	angela@hotel.c	
Barrios	737	miguel.zayas@j	
Espino	22984418N	629804939	gael61@yahoo.
Linares	97374975J	604710360	bruno98@beni
Rey	56366197M	680356232	odavila@live.co

Modal de confirmación previo a eliminar el cliente seleccionado

GestHotel v1.0.1

admin admin admin Logout

Dashboard Reservas Administración Sistema

ALTA TRABAJADOR

Nombre	Primer Apellido	Segundo Apellido
DNI	Teléfono	Email
Dirección	Provincia	País

Guardar Cancelar

Copyright © Todos los Derechos Reservados 2017

Formulario Nuevo trabajador

A tener en cuenta:

- No se aceptan caracteres distintos del alfabeto Español.
- Se pueden incluir tildes y diéresis.
- Los campos del formulario tendrán una extensión mínima de 3 caracteres y una máxima de 15, excepto en la dirección que no existe límites.
- La letra del dni ha de ir en mayúsculas.

9.4 Tipo Habitación

Listado que muestra las tipologías de habitaciones del establecimiento.

El listado nos muestra:

- Nombre (Denominación común de la tipología).
- Descripción.
- Acceso a crear nuevas tipologías.
- Acceso a Editar los tipos guardados.
- Acceso a Eliminar los tipo guardados.

El listado nos ofrece paginador de 10, 25, 50 o 100 registros y buscador para filtrar por cualquier criterio.

The screenshot shows the GestHotel v1.0.1 application interface. On the left is a dark sidebar with navigation links: Dashboard, Reservas, Administración (selected), and Sistema. The main content area has a title 'TIPO DE HABITACIONES' with a 'Nuevo' button. It includes a search bar and a dropdown for 'Mostrar' (10, 25, 50, 100) registros. A 'Buscar:' input field is also present. The table lists six room types with their descriptions and edit/delete icons. At the bottom, it says 'Mostrando registros del 1 al 6 de un total de 6 registros' and includes 'Anterior' and 'Siguiente' buttons.

Nombre	Descripción			
Doble	Uso doble o individual. Dos camas de 105 cm individuales independientes.			
Individual	Uso individual. Cama de 105 cms.			
KingSize	Uso doble o individual. Una cama de 2 metros x 2 metros.			
Matrim	Doble de uso individual o dos personal. Una cama de 150 cms.			
Suites	Cuentan con dos habitaciones dobles, 2 baños, salón y estancia.			
Triple	Triple para uso de una a tres personas. Cuenta con tres camas de 105 cms.			

Vista del listado de Tipologías

GestHotel v1.0.1

admin admin admin Logout

Dashboard Reservas Administración Sistema

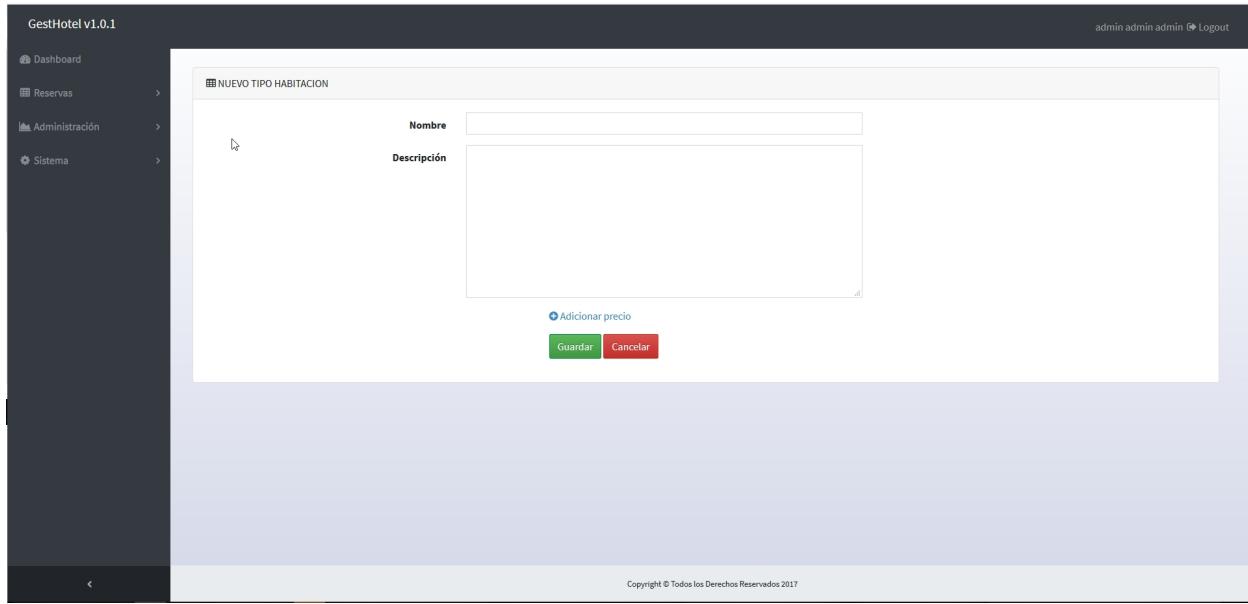
NUEVO TIPO HABITACION

Nombre
Descripción

⊕ Adicionar precio

Guardar Cancelar

Copyright © Todos los Derechos Reservados 2017



Vista para crear un nuevo tipo de habitación

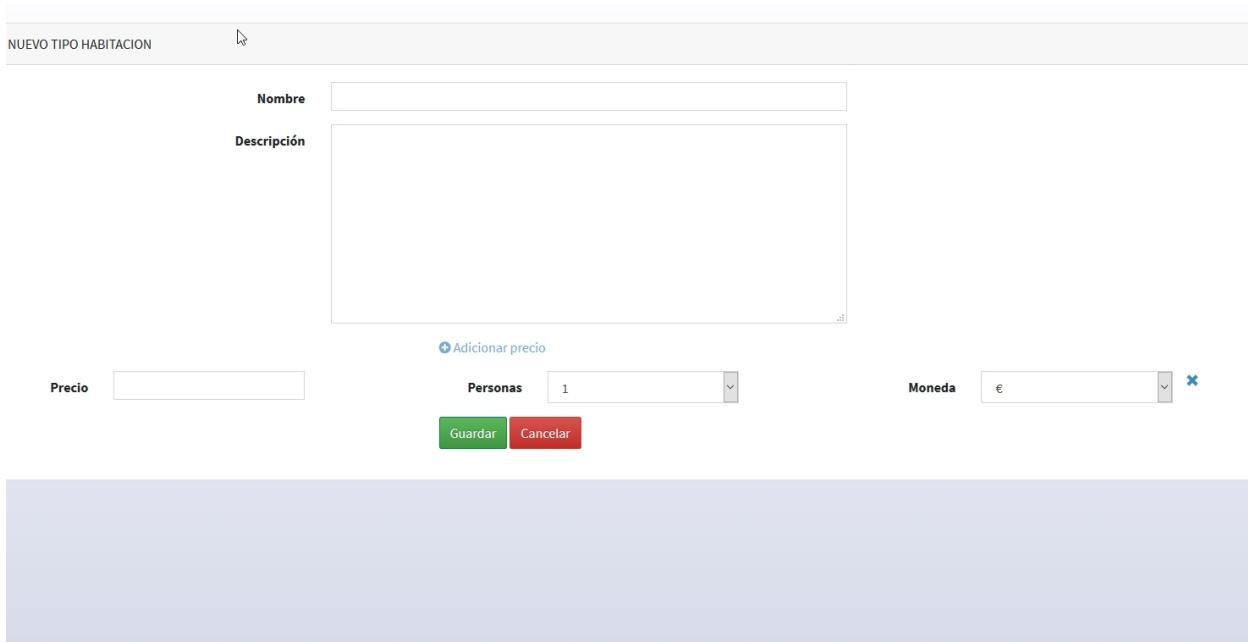
NUEVO TIPO HABITACION

Nombre
Descripción

⊕ Adicionar precio

Precio Personas Moneda €

Guardar Cancelar



Detalle para adicionar un precio al tipo de habitación

A tener en cuenta:

- Podemos fijar tantos precios distintos para la misma habitación como queramos.
- Los distintos precios aparecerán en el formulario para generar una reserva.
- Se puede eliminar cualquier precio al listado, pulsando el aspa que aparece al final del precio que queremos eliminar.

GestHotel v1.0.1

admin admin admin Logout

Dashboard Reservas Administración Sistema

EDITAR TIPO HABITACION

Nombre: Doble

Descripción: Uso doble o individual.
Dos camas de 105 cm individuales independientes.

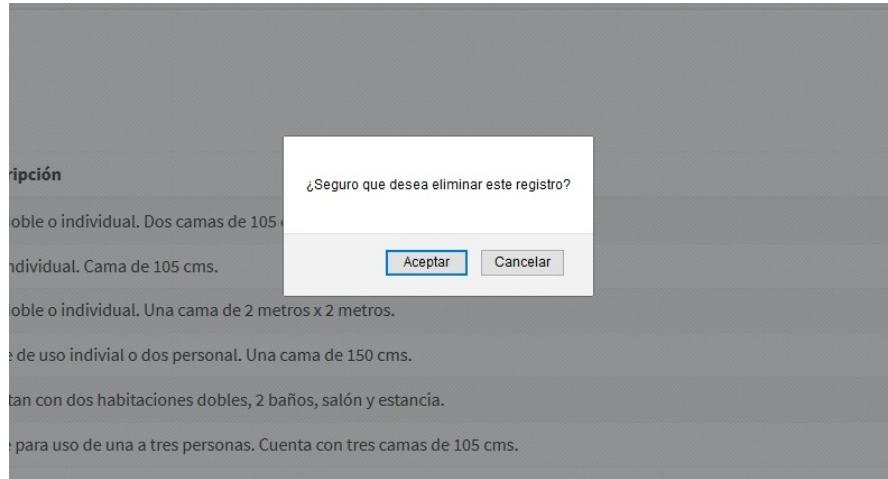
Precio: 60 Personas: 2 Moneda: €

Precio: 50 Personas: 1 Moneda: €

Guardar Cancelar

Copyright © Todos los Derechos Reservados 2017

Vista Editar el tipo de habitación seleccionada



Modal previo a eliminar el tipo de habitación seleccionada

10 Logout

Para salir del sistema, simplemente pulse sobre su nombre de Usuario, que aparece en la parte superior derecha del panel de administración.

The screenshot shows a dark header bar with the text "PEDRO MIRANDA NISA" and a "Logout" button. Below this is a search bar labeled "Buscar:" with an empty input field. Underneath is a table with columns: "D", "Total", "A Cta.", and "Pte.". The first row contains the value "ENTE" under "D", "360€" under "Total", "0" under "A Cta.", and "\$" under "Pte.". The table has small up/down arrows in its header. At the bottom center of the page is the text "Detalle del Logout".

D	Total	A Cta.	Pte.
ENTE	360€	0	\$

GestHotel
Manual de Usuario

Versión: 0100

Fecha: 12/12/2017

1 Descripción del Sistema

Objeto.

En este manual encontrarás toda la información necesaria para la utilización de la aplicación **GestHotel** de una forma sencilla e intuitiva.

¿Qué es GestHotel?

GestHotel es una aplicación cuya misión es centralizar la gestión de un hotel.

En resumen, la aplicación permite gestionar el alta de clientes, reservas de habitaciones, checkin y checkout, reportes, etc.

2 Acceso a GestHotel

Simplemente acceda desde cualquier navegador a la url que su Administrador le haya facilitado.

3 Inicio de GestHotel

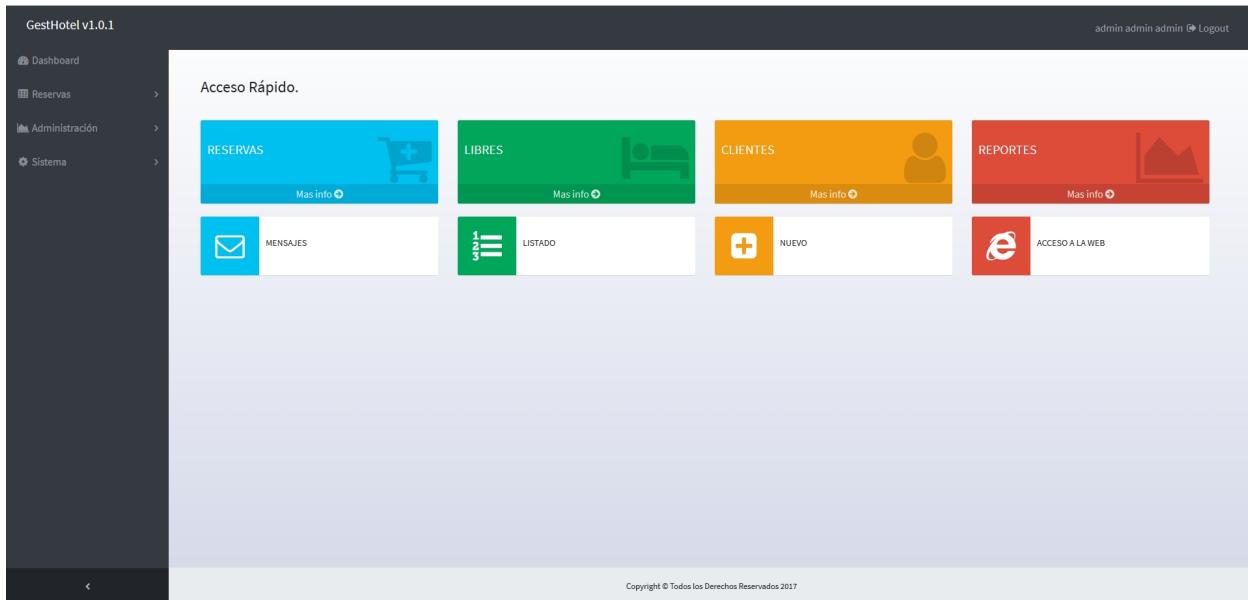
La aplicación desplegará una pantalla de acceso a través de Login con las credenciales que le proporcionará su Administrador.

Una vez ingresado los datos de acceso, si son correctos, dispondrá del panel de administración de GestHotel.

4 Página Principal

La primera pantalla que disponemos, nos proporciona, ademas del acceso a cualquier punto de la aplicación a través del menú lateral, una serie de accesos rápidos que serán los mas utilizados:

- Acceso al listado de Reservas.
- Acceso a la disponibilidad de habitaciones.
- Acceso al listado de clientes.
- Acceso al generador de informes.
- Acceso al correo electrónico.
- Listado total de habitaciones.
- Crear un nuevo cliente.
- Acceso en una nueva pestaña a la web del Establecimiento del cliente.

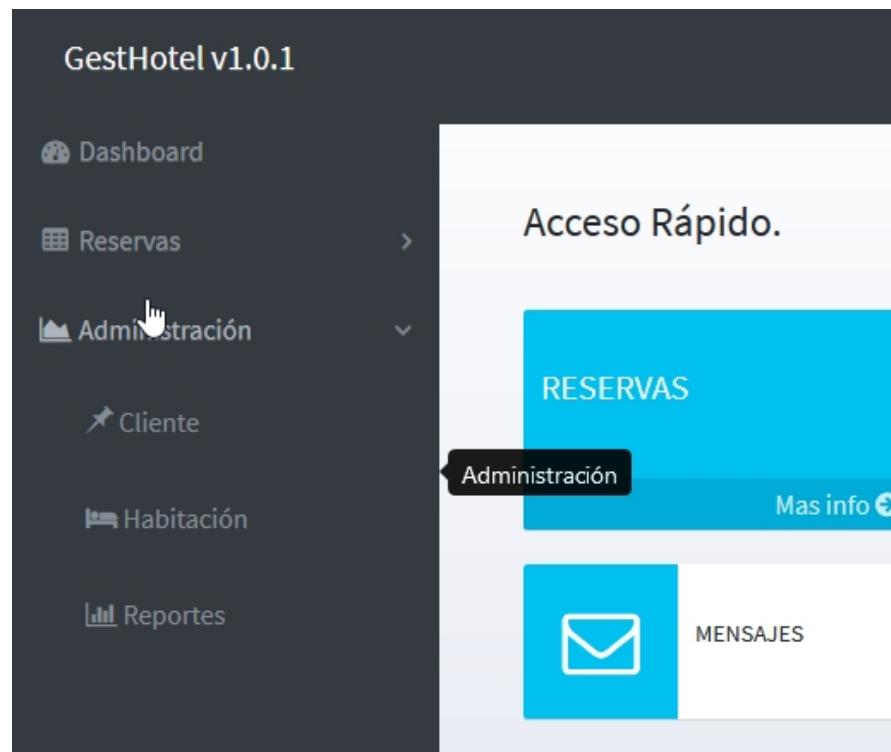


Página Principal

5 Panel lateral

Desde el Panel Lateral se accede a cualquier punto del sistema, si desplegamos todos sus ítems tendremos acceso a los siguientes Módulos:

- Dashboard
- Reservas
 - ➔ Reservadas
 - ➔ Disponibilidad
- Administración
 - ➔ Cliente
 - ➔ Habitación
 - ➔ Reportes



Detalle del Panel Lateral

6 Dashboard

Acceso directo a la pantalla principal de la aplicación.

7 Reservas

Despliega dos ítems:

- ➔ Reservadas
- ➔ Disponibilidad

7.1 Reservadas

Listado de habitaciones reservadas con los siguientes datos de cada una de ellas:

- Número de habitación.
- Tipología.
- Nombre y apellidos del cliente.
- Fecha y hora de ingreso.
- Fecha y hora de salida.
- Total días de la estancia .
- Días pendientes.
- Estado de pago.
 - Pendiente .
 - Cancelado.
- Total a pagar.
- Entregas a cuenta.
- Pendiente de abonar.
- Acceso al cobro total o parcial (Símbolo Dólar '\$')
- Acceso para liberar la habitación.

La tabla nos permite paginar los resultados en 10, 25, 50 o 100 registros.

También dispone de un buscador que filtra cualquier criterio.

GestHotel v1.0.1

admin admin admin Logout

Dashboard >

Reservas >

Administración >

Sistema >

LISTADO DE RESERVAS

Mostrar 10 registros Buscar:

Hab.	Tipo	Cliente	Ingreso	Salida	Total Días	Dias Ptos	Estado	Total	A Cta.	Pte.	Tareas
# 106	Individual	Pedro Aponte Treviño	2017-12-12 17:49:27	2017-12-13 15:00:00	1	+0 días	CANCELADO	45	45	0	Liberar
# 108	Triple	Alma Arreola Castro	17-12-2017 17:51:15	24-12-2017 15:00:00	7	+11 días	PENDIENTE	630€	0	630	\$
# 110	Doble	Carmen Lebrón Delrío	12-12-2017 17:50:03	19-12-2017 15:00:00	7	+6 días	PENDIENTE	420€	0	420	\$
# 201	Doble	Gonzalo Toro Farías	12-12-2017 17:50:39	14-12-2017 15:00:00	2	+1 días	PENDIENTE	120€	0	120	\$
# 205	KingSize	Pedro Lugo Carvajal	13-12-2017 17:52:11	17-12-2017 15:00:00	4	+4 días	PENDIENTE	480€	0	480	\$

Mostrando registros del 1 al 5 de un total de 5 registros

Anterior 1 Siguiente

Copyright © Todos los Derechos Reservados 2017

Listado de reservas

Personas 3 : 90 €(Euros)
Días : 7

COBRO

TOTAL	Pago a cuenta	Pago pendiente	Total a Cobrar
630	0	630	630

Cobrar Cancelar

#	Habitación	Cliente	Ingreso	Salida	Total Días	Dias Ptos	Estado	Total	A Cta.	Pte.	Tareas
# 106	Individual	Pedro Aponte Treviño	2017-12-12 17:49:27	2017-12-13 15:00:00	1	+0 días	CANCELADO	45	45	0	
# 108	Triple	Alma Arreola Castro	17-12-2017 17:51:15	24-12-2017 15:00:00	7	+11 días	PENDIENTE	630€	0	630	\$
# 110	Doble	Carmen Lebrón Delrío	12-12-2017 17:50:03	19-12-2017 15:00:00	7	+6 días	PENDIENTE	420€	0	420	\$
# 201	Doble	Gonzalo Toro Farías	12-12-2017 17:50:39	14-12-2017 15:00:00	2	+1 días	PENDIENTE	120€	0	120	\$
# 205	KingSize	Pedro Lugo Carvajal	13-12-2017 17:52:11	17-12-2017 15:00:00	4	+4 días	PENDIENTE	480€	0	480	\$

Mostrando registros del 1 al 5 de un total de 5 registros

Anterior 1 Siguiente

Modal para realizar cobros totales o parciales

7.2 Disponibilidad

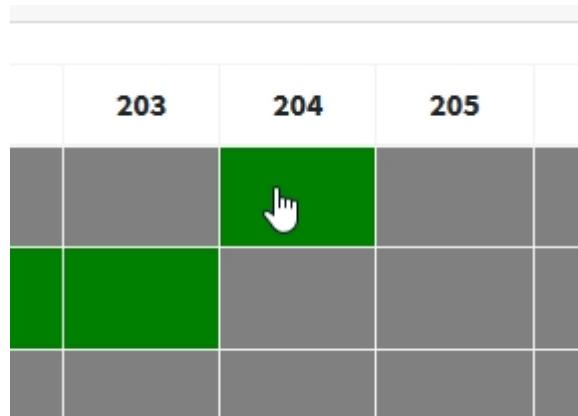
Panel que nos muestra de un vistazo las habitaciones disponibles (en verde) y las ocupadas (en rojo)

The screenshot shows the 'DISPONIBILIDAD' (Availability) section of the GestHotel v1.0.1 dashboard. The interface includes a sidebar with navigation links: Dashboard, Reservas, Administración, and Sistema. The main area features a grid titled 'DISPONIBILIDAD' with columns labeled from 101 to 308. The first column, 'Tipo', contains icons representing room types: a single person, two people, three people, four people, a couple, and a group. The grid cells are colored green for availability and red for occupancy. A legend at the bottom right identifies the colors: green for availability and red for occupancy.

Panel de Disponibilidad

La primera columna nos indica la tipología de la habitación, y fila superior el numero de habitación.

Es posible acceder directamente a una nueva reserva cliqueando sobre el cuadro verde de la habitación libre.



En tal caso, la aplicación desplegará el siguiente formulario:

The screenshot shows the 'NUEVA RESERVA' (New Reservation) form. At the top, it displays 'Habitación N°: 103' and 'Tipo de Habitación: Matrim'. Below this, there's a note: 'Descripción: Doble de uso individual o dos personal. Una cama de 150 cms.' A section for 'Selección Precio:' shows two options: 'Personas 2 : 90€' and 'Personas 1 : 80€'. There are fields for 'A cuenta' (Account), 'Total' (Total), and 'Saldo' (Balance). A 'Cliente +' button is present. The 'Notas' (Notes) field contains a calendar for December 2017. At the bottom, there are 'Guardar' (Save) and 'Cancelar' (Cancel) buttons.

El formulario Nueva Reserva nos muerta:

- El número de habitación que vamos a reservar.
- La tipología de la habitación.
- Una descripción de la misma.
- Uno o varios check para seleccionar el número de personas que ocuparan la habitación.
- Un campo para ingresar si el cliente entrega alguna cantidad a cuenta.
- Un campo para ingresar el nombre y apellidos del cliente.
- Acceso directo para crear un nuevo cliente
- Un campo para realizar anotaciones extras, como por ejemplo algún requisito que nos haga el cliente.
- Un campo de tipo calendario para ingresar la fecha de entrada.
- Y otro campo para ingresar la fecha de salida.

A tener en cuenta:

- El check para seleccionar el numero de personas que ocuparan la habitación, es obligatorio.
- El campo ingresar alguna cantidad a cuenta, solo permite dígitos numéricos.
- Para ingresar el nombre y apellidos del cliente, solamente debe escribir las 3 primeras letras de su nombre o de alguno de sus apellidos y automáticamente le aparecerá las coincidencias con los clientes registrados en el sistema.
- Si es un cliente nuevo, podrá darlo de alta por el procedimiento normal, desde menu/administración/cliente/nuevo, pero le será mas sencillo simplemente cliquear el símbolo ‘mas’ que aparece junto al campo cliente del formulario que estamos tratando y le aparecerá un formulario para dar de alta el nuevo cliente.
- Al escoger una fecha de ingreso, el sistema no le permitirá ingresar fechas inferiores al día que esté realizando la reserva.
- Al escoger una fecha de salida, el sistema le permitirá escoger como primer día, el posterior a la fecha de ingreso.
- Para ejecutar la reserva, simplemente haga click sobre el botón verde con la leyenda Guardar.
- Si desea descartar la reserva o simplemente salir del formulario, haga click sobre el botón rojo con la leyenda Cancelar.

The screenshot displays the 'Nuevo Cliente' (New Client) form within the GestHotel v1.0.1 application. The form contains the following fields:

- Nombre*
- Apellido1*
- Apellido2*
- DNI*
- Teléfono*
- Email*
- Dirección
- Provincia
- País

Below the form are two date pickers:

- Check-in:** Diciembre, 2017 (Lu Ma Mi Ju Vi Sa Do)
- Check-out:** Diciembre, 2017 (Lu Ma Mi Ju Vi Sa Do)

The sidebar on the left provides navigation through various sections such as Dashboard, Reservas, Administración, Habitaciones, and Cliente. The date pickers show the month of December 2017 with specific dates highlighted.

Formulario Nuevo Cliente

En el caso de cliquear en una habitación ocupada, nos redirigirá al listado de habitaciones reservadas pero filtrando el resultado a la habitación seleccionada.

8 Administración

Despliega tres opciones:

- ➔ Cliente
- ➔ Habitación
- ➔ Reportes

8.1 Cliente

Listado de clientes ingresados en la base de datos.

De cada uno de ellos se muestra:

- Nombre
- Apellido1
- Apellido2
- DNI
- Teléfono
- Email
- Acceso a Más detalles
- Acceso a Editar el cliente

El listado nos proporciona un paginador para 10, 25, 50 o 100 registros.

Disponemos de un buscador para filtrar cualquier criterio.

Disponemos de un botón de acceso directo para crear un nuevo cliente.

GestHotel v1.0.1

Dashboard

Reservas

Administración

Sistema

Nuevo

Mostrar 10 registros

Buscar:

Nombre	Apellido1	Apellido2	DNI	Teléfono	Email	Acción	Acción	Acción	Acción
Aaron	Galindo	Gil	31323948H	688047913	martina.altamirano@terra.com				
Adam	Valero	Rascón	01087415K	664581066	ismael81zayas.com				
Adriana	Roque	Pardo	97728370J	684905340	lala.osorio@hotmail.es				
Adrián	Molina	Ramón	39249112Y	662854843	jurado.alba@hispanavista.com				
Alma	Arreola	Castro	03739289H	672231008	sandra.sisneros@latinmail.com				
Andrés	Baca	Rocha	81747589T	692928697	xdomenech@collado.org				
Andrés	Tijerina	Pozo	98453297A	621642127	mario27@live.com				
Anna	Terán	Guerra	23074374S	648015954	hugo.benitez@live.com				
Aya	Gómez	Delgado	34975446J	627520460	olimon@perea.net				
Candela	Dávila	Leiva	09911778G	610571432	moreno.helena@hotmail.com				

Mostrando registros del 1 al 10 de un total de 41 registros

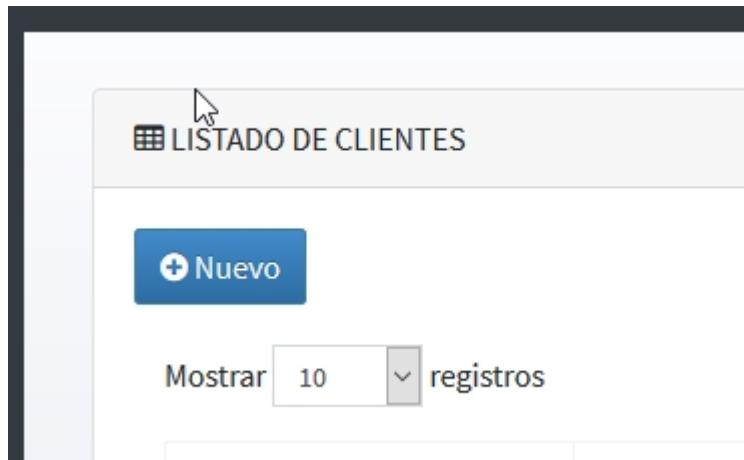
Anterior 1 2 3 4 5 Siguiente

Copyright © Todos los Derechos Reservados 2017

Listado de clientes

@hotel.com		
ndariz@dominguez.com		
cova@yahoo.com		
jua.hector@najera.es		
alba@hispanavista.com		

Detalle del buscador y de los accesos a Mas detalles y Editar



Detalle del paginador y acceso a crear un cliente nuevo.

Nombre	Primer Apellido	Segundo Apellido
Aaron	Galindo	Gil
DNI	Teléfono	Email
31323948H	688047913	martina.altamirano@terra.com
Dirección	Provincia	País
Travesía Francisco, 824, Ático 8º, 11793, As Mora de Arriba	Valencia	España

Volver al listado

Detalle del cliente seleccionado

GestHotel v1.0.1

admin admin admin Logout

Dashboard Reservas Administración Sistema

EDITAR CLIENTE

Nombre	Primer Apellido	Segundo Apellido
Aaron	Galindo	Gil
DNI	Teléfono	Email
31323948H	688047913	martina.altamirano@terra.com
Dirección	Provincia	País
Travesía Francisco, 824, Ático 8º, 11793, As Mora de Arriba	Valencia	España

Guardar Cancelar

Copyright © Todos los Derechos Reservados 2017

Formulario Editar cliente seleccionado

GestHotel v1.0.1

admin admin admin Logout

Dashboard Reservas Administración Sistema

NUEVO CLIENTE

Nombre	Primer Apellido	Segundo Apellido
<input type="text"/>		
DNI	Teléfono	Email
Dirección	Provincia	País

Guardar Cancelar

Copyright © Todos los Derechos Reservados 2017

Formulario Nuevo cliente.

A tener en cuenta:

- No se aceptarán caracteres distintos del alfabeto Español.
- Se pueden incluir tildes y diéresis.
- Los campos del formulario tendrán una extensión mínima de 3 caracteres y una máxima de 15, excepto en la dirección que no existe límites.
- La letra del dni ha de ir en mayúsculas.
- Estas condiciones son válidas para el formulario desplegado desde una nueva reserva.

8.2 Habitación

Listado de las habitaciones del Establecimiento.

Nos muestra:

- Número de habitación
- Tipología de la habitación

El listado nos ofrece paginador de 10, 25, 50 o 100 registros y buscador para filtrar por cualquier criterio.

Número de Habitación	Tipo de Habitación
101	Matrim
102	Triple
103	Matrim
104	Doble
105	Suites
106	Individual
107	Doble
108	Triple
109	Suites
110	Doble

Lista de Habitaciones

8.3 Reportes

Panel para filtrar la consulta de informes de las reservas del establecimiento.

Para ejecutar un informe, previamente deberemos seleccionar:

- Consultar todas las reservas.
- Consultar las reservas pendientes de pago.
- Consultar la reservas que hayan abonado el pago.
- Escoger un rango de fechas.

GestHotel v1.0.1

ANGELA PEREZ PEREZ Logout

Dashboard >

Reservas >

Administración >

REPORTES

Reservas

Estado de Pago

TODOS

Desde

2017-12-13

Hasta

2017-12-13

Consultar Cancelar

Copyright © Todos los Derechos Reservados 2017

Panel Reportes

A continuación, pulsaremos en Consultar y se generará un informe en formato .pdf de las Reservadas con los filtros aplicados anteriormente.

#	Hab	Cliente	Ingreso	Salida	Empleado	Estado	Precio	Días	Pagado	Total
27	106	Juan Valero Bascón	2017-12-02 19:06:24	2017-12-04 15:00:00	Aya Esteve Barrios	CANCELADO	45	2	90€	90
28	106	Pedro J Aponte Trujillo	2017-12-12 17:49:27	2017-12-13 15:00:00	Aya Esteve Barrios	CANCELADO	45	1	45€	45
29	110	Carmen Lebrón Delrio	2017-12-12 17:50:03	2017-12-19 15:00:00	Aya Esteve Barrios	PENDIENTE	60	7	0€	420
30	201	Gonzalo Farias	2017-12-12 17:50:39	2017-12-14 15:00:00	Aya Esteve Barrios	PENDIENTE	60	2	0€	120
31	108	Alma Arreola Castro	2017-12-17 17:51:15	2017-12-24 15:00:00	Aya Esteve Barrios	PENDIENTE	90	7	0€	630
32	205	Paulo Lugo Carvajal	2017-12-13 17:52:11	2017-12-17 15:00:00	Aya Esteve Barrios	PENDIENTE	120	4	0€	480

9 Logout

Para salir del sistema, simplemente pulse sobre su nombre de Usuario, que aparece en la parte superior derecha del panel de administración.

