

Diseño de Arquitectura de Software

Patrón de Arquitectura monolithic

**M.C. Ángel Santiago Jaime Zavala**

**Iván Carreón Silva**

**Cuauhtémoc Martínez Requenes**

**Georgina Valenzuela Mireles**

**Diciembre 2017**

Contenido

[Introducción 2](#_Toc500428187)

[Síntesis del patrón 3](#_Toc500428188)

[Historia de la arquitectura de monolítica 4](#_Toc500428189)

[Objetivos 4](#_Toc500428190)

[Diagrama de la Arquitectura 5](#_Toc500428191)

[Ventajas y Desventajas 5](#_Toc500428192)

[Similitudes con otras arquitecturas 6](#_Toc500428193)

[Frameworks 7](#_Toc500428194)

[Ejemplos de Frameworks 7](#_Toc500428195)

[Diagrama de la página web para el Sistema de reservaciones de un hotel 9](#_Toc500428196)

[Estructura de la Web App 10](#_Toc500428197)

[Bibliografía 11](#_Toc500428198)

[Anexo 12](#_Toc500428199)

[Screenshots de la Web App 12](#_Toc500428200)

# Introducción

“… más allá de los algoritmos y estructuras de datos de la computación, diseñar y especificar

la estructura del sistema completo emerge como un nuevo tipo de problema. Los

aspectos estructurales incluyen la organización gruesa y la estructura de control global; protocolos

para comunicación, sincronización, y acceso a datos; asignación de funcionalidad a

elementos de diseño; distribución física; composición de elementos de diseño; escalabilidad

y performance; y selección de alternativas de diseño.” - (Garlan y Shaw, 1993)

“La organización fundamental de un sistema, encarnada en sus componentes, sus relaciones

entre sí y con el entorno, y los principios que gobiernan su diseño y evolución.”

* (ANSI/IEEE Std 1471-2000)

Una Arquitectura de Software, también denominada Arquitectura lógica, es el diseño de más alto nivel de la estructura de un sistema, consiste en un conjunto de patrones y abstracciones coherentes.

Una arquitectura de software se selecciona y diseña con base en objetivos y restricciones. Los objetivos son aquellos prefijados para el sistema de información, pero no solamente los de tipo funcional, también otros objetivos como la mantenibilidad, auditabilidad, flexibilidad e interacción con otros sistemas de información. Las restricciones son aquellas limitaciones derivadas de las tecnologías disponibles para implementar sistemas de información.

La ***arquitectura de software*** define, de manera abstracta, los componentes que llevan a cabo alguna tarea de computación, sus interfaces y la comunicación entre ellos. Toda arquitectura debe ser implementada en una arquitectura física, que consiste simplemente en determinar qué computadora tendrá asignada cada tarea.

Los ***patrones arquitectónicos*** se utilizan para expresar una estructura de organización base o esquema que ofrecen soluciones a problemas de arquitectura de software. Proporcionando un conjunto de sub-sistemas predefinidos, especificando sus responsabilidades, reglas, directrices que determinan la organización, comunicación, interacción y relaciones entre ellos.

# Síntesis del patrón

Una arquitectura monolítica es el modelo tradicional unificado para el diseño de un programa de software, quiere decir que son implementadas dentro de un solo componente de software; Las aplicaciones monolíticas son aquellas en las que el software *se estructura en grupos funcionales muy acoplados*, involucrando los aspectos referidos a la presentación, procesamiento y almacenamiento de la información. A estas las conocemos como aplicaciones de estación en otras palabras interfaces graficas de usuario GUI’S son servicios de presentación, negocios y pertinencia de datos, en la misma maquina no hay concurrencia de usuarios.

Las características fundamentales de este tipo de aplicaciones son:

* Construcción del programa final a base de componentes compilados al mismo tiempo.
* Carecen de protecciones yprivilegios al entrar a rutinas que manejan diferentes aspectos de los recursos de la computadora, como memoria, disco, etc.
* Generalmente están hechos a medida, por lo que son eficientes y rápidos en su ejecución y gestión, pero como consecuencia carecen de flexibilidad para soportar diferentes ambientes de trabajo o tipos de aplicaciones.
* No hay distribución, tanto a nivel físico como a nivel lógico.
* Estructura interna indefinida. Los niveles de funcionalidad no están bien separados. Una aplicacion monolítica puede componerse de uno o más niveles

Una aplicacion monolítica se elabora bajo el supuesto de que será ejecutada en un único equipo de cómputo. Pero este tipo de aplicacion también puede ser implementada en un computador central o mainframe. Desde una terminal de servicio se puede monitorear y recibir los resultados que la aplicacion monolítica devuelva.

# Historia de la arquitectura de monolítica

En el inicio de la era de la informática, las computadoras eran grandes y caras. Debido a su escasez y coste, funcionaban de forma independiente. Por ejemplo, la configuración más habitual en los sistemas bancarios de los años 60 y 70 consistía en disponer de un mainframe con una base de datos de tipo jerárquico y en el que se ejecutaba una aplicación COBOL.

Históricamente, representan la estructura de los primeros softwares, constituidos fundamentalmente por un solo programa compuesto por un conjunto de funciones entrelazadas de tal forma que cada una puede llamar a cualquier otra. Por lo general, los clientes eran máquinas sin procesador que se limitaban a enviar peticiones al mainframe y mostrar las respuestas al usuario.

Las aplicaciones ejecutadas en entornos de mainframes y las maquinas sin procesador eran lo que hoy conocemos como aplicaciones monolíticas.

Tras la aparición de los primeros PC's, ya se podía repartir la carga de los mainframes. Esta redistribución de funciones tuvo grandes repercusiones económicas ya que los PC's reducían el trabajo de los mainframes y podían ser sustituidos por máquinas más baratas, conocidas como servidores.

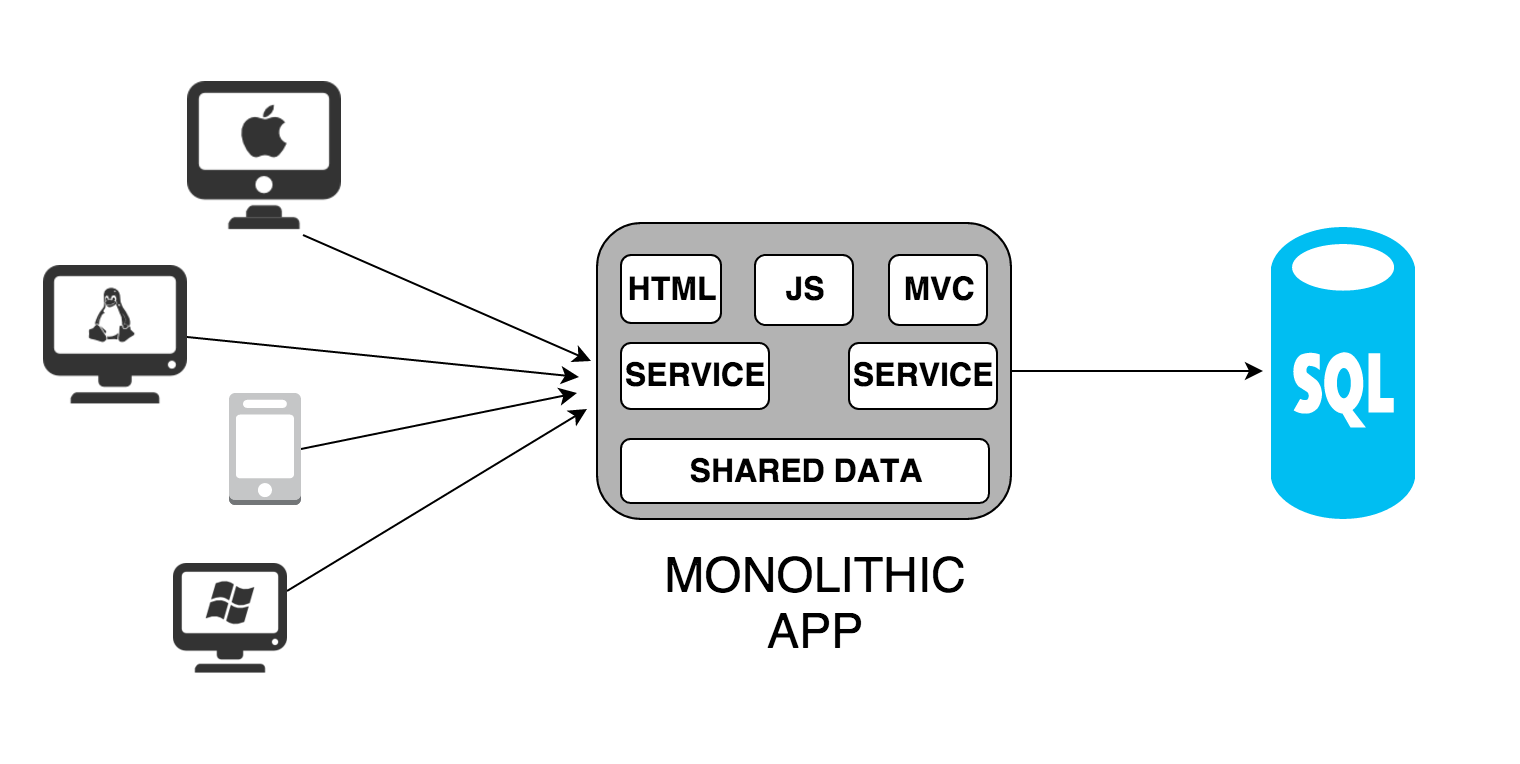
A estas aplicaciones en las que las funciones se repartían entre un servidor y los PC's se las llamó aplicaciones cliente-servidor.

# Objetivos

Los patrones de diseño pretenden:

* Proporcionar catálogos de elementos reusables en el diseño de sistemas software.
* Evitar la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente.
* Formalizar un vocabulario común entre diseñadores.
* Estandarizar el modo en que se realiza el diseño.
* Facilitar el aprendizaje de las nuevas generaciones de diseñadores condensando conocimiento ya existente.

# Diagrama de la Arquitectura



# Ventajas y Desventajas

**Ventajas**

* Eficiencia, ya que se producen pocos cambios en el contexto.

**Desventajas**

* Difícil de depurar
* Difícil de ampliar
* Difícil de distribuir
* Difícil de implantar.

**Tipos de software al cual aplica este Estilo Arquitectónico:**

* Firmware
* Software base de Sistema Operativo (Servicios, deamons, Kernell)
* Pequeñas utilidades especificas (Desde virus, keyloggers, sniffers)

# Similitudes con otras arquitecturas

* Arquitectura Cliente-Servidor.

Esta arquitectura consiste básicamente en un cliente que realiza peticiones a otro programa (el servidor) que le da respuesta. Aunque esta idea se puede aplicar a programas que se ejecutan sobre una sola computadora es más ventajosa en un sistema operativo multiusuario distribuido a través de una red de computadoras.

En esta arquitectura la capacidad de proceso está repartida entre los clientes y los servidores, aunque son más importantes las ventajas de tipo organizativo debidas a la centralización de la gestión de la información y la separación de responsabilidades, lo que facilita y clarifica el diseño del sistema.

La separación entre cliente y servidor es una separación de tipo lógico, donde el servidor no se ejecuta necesariamente sobre una sola máquina ni es necesariamente un sólo programa. Los tipos específicos de servidores incluyen los servidores web, los servidores de archivo, los servidores del correo, etc. Mientras que sus propósitos varían de unos servicios a otros, la arquitectura básica seguirá siendo la misma.

Una disposición muy común son los sistemas multicapa en los que el servidor se descompone en diferentes programas que pueden ser ejecutados por diferentes computadoras aumentando así el grado de distribución del sistema.

La arquitectura cliente-servidor sustituye a la arquitectura monolítica en la que no hay distribución, tanto a nivel físico como a nivel lógico.

*Características*

En la arquitectura C/S el remitente de una solicitud es conocido como cliente. Sus características son:

* Es quien inicia solicitudes o peticiones, tienen por tanto un papel activo en la comunicación

(dispositivo maestro o amo).

* Espera y recibe las respuestas del servidor.
* Por lo general, puede conectarse a varios servidores a la vez.
* Normalmente interactúa directamente con los usuarios finales mediante una interfaz gráfica de usuario.

Al receptor de la solicitud enviada por cliente se conoce como servidor. Sus características son:

* Al iniciarse esperan a que lleguen las solicitudes de los clientes, desempeñan entonces un papel pasivo en la comunicación (dispositivo esclavo).
* Tras la recepción de una solicitud, la procesan y luego envían la respuesta al cliente.
* Por lo general, aceptan conexiones desde un gran número de clientes (en ciertos casos el número máximo de peticiones puede estar limitado).
* No es frecuente que interactúen directamente con los usuarios finales

# Frameworks

*¿Qué es un Framework?*

Es un entorno de trabajo que cuenta con una estructura conceptual o un esquema (un esqueleto o un patrón) para el desarrollo y/o la implementación de una aplicación.

## Ejemplos de Frameworks

* **Angular**, creado por Google, está específicamente pensado para proyectos complejos, para los que aporta herramientas para la gestión de dependencias, modelos, controladores y enrutado. Gracias a ideas novedosas en forma de directivas que se asemejan a etiquetas HTML y que son gestionadas por Angular se ha convertido en unos de los Frameworks de cliente más populares
* **Polymer**, publicada en 2013 también por Google, es una librería orientada a componentes. La idea es crear webs a partir de elementos reutilizables, de modo que podamos componer aplicaciones a partir de bloques sin necesidad de conocer el funcionamiento interno de cada uno, de modo que se extiende HTML de forma natural.

Mientras que Angular está más pensado para un desarrollo integral, Polymer está más centrado en la capa de presentación y es integrable con otros frameworks como Angular2.

* **React,** a diferencia de la solución de Google está más centrada en la interfaz de usuario y en representar de forma eficiente los cambios de estado de la aplicación. Está pensada para ser combinada con otras librerías, como routers o Flex para gestión de los modelos.

Uno de los puntos fuertes de React es el proyecto React Native, que permite obtener rendimiento nativo en plataformas iOS y Android, transpilando el código JavaScript a componentes nativos.

* **Symfony** es un completo framework diseñado para optimizar el desarrollo de las aplicaciones web basado en el patrón Modelo Vista Controlador. Para empezar, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. El resultado de todas estas ventajas es que no se debe reinventar la rueda cada vez que se crea una nueva aplicación web. Symfony está desarrollado completamente en PHP 5.3.
* **Ruby on Rails**, también conocido como RoR o Rails, es un framework de aplicaciones web de código abierto escrito en el lenguaje de programación Ruby, siguiendo el paradigma del patrón Modelo Vista Controlador (MVC). Trata de combinar la simplicidad con la posibilidad de desarrollar aplicaciones del mundo real escribiendo menos código que con otros frameworks y con un mínimo de configuración. El lenguaje de programación Ruby permite la metaprogramación, de la cual Rails hace uso, lo que resulta en una sintaxis que muchos de sus usuarios encuentran muy legible. Rails se distribuye a través de RubyGems, que es el formato oficial de paquete y canal de distribución de bibliotecas y aplicaciones Ruby.

Para nuestra Web app utilizamos Django

*¿Qué es Django?*

**Django** es un framework para aplicaciones web gratuito y open source, escrito en Python. Es un WEB framework - un conjunto de componentes que ayudan a desarrollar sitios web más fácil y rápidamente.

Se necesita un conjunto de componentes similares: una manera de manejar la autenticación de usuarios (registrarse, iniciar sesión, cerrar sesión), un panel de administración para tu sitio web, formularios, una forma de subir archivos, etc.

Lo primero es que el servidor necesita saber qué quieres que se sirva de una página web. Tenemos un puerto el cual es monitoreado por peticiones. Esto es realizado por un servidor web. El servidor web lee la petición, y envía una respuesta con una página web, cuando se requiere enviar algo, se tiene que tener algún contenido y Django es ese algo que te ayuda a crear el contenido.

## Diagrama de la página web para el Sistema de reservaciones de un hotel

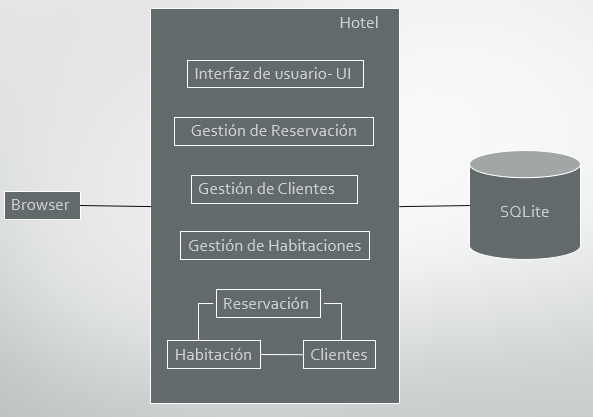


Diagrama de Clases

Imagen que contiene captura de pantalla

Descripción generada con confianza muy alta

## Estructura de la Web App

Imagen que contiene captura de pantalla

Descripción generada con confianza alta

# Bibliografía

<http://historiqnx.blogcindario.com/2007/09/00010-arquitectura.html>

<https://developer.mozilla.org/es/docs/Learn/Server-side/Django/Introducci%C3%B3n>

<http://qualityandprogramming.blogspot.mx/2012/04/estilo-arquitectonico-software.html>

<https://sg.com.mx/revista/38/conceptos-dise%C3%B1o-patrones-t%C3%A1cticas-y-frameworks#.Wimt6UqWbIU>

<https://tallerbd.wikispaces.com/Arquitectura+Monol%C3%ADtica>

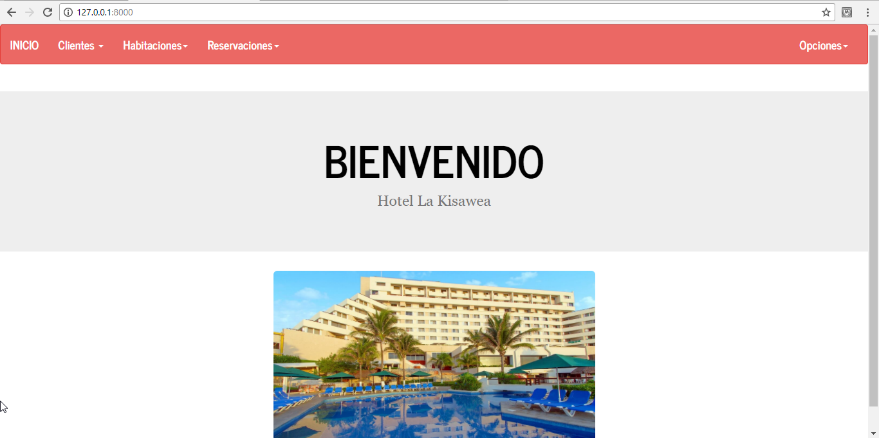
<http://es.sandramarramirez.wikia.com/wiki/Arquitectura_Monol%C3%ADtica_y_Arquitectura_Cliente_-_Servidor>

<http://magdalyithunid3.blogspot.mx/>

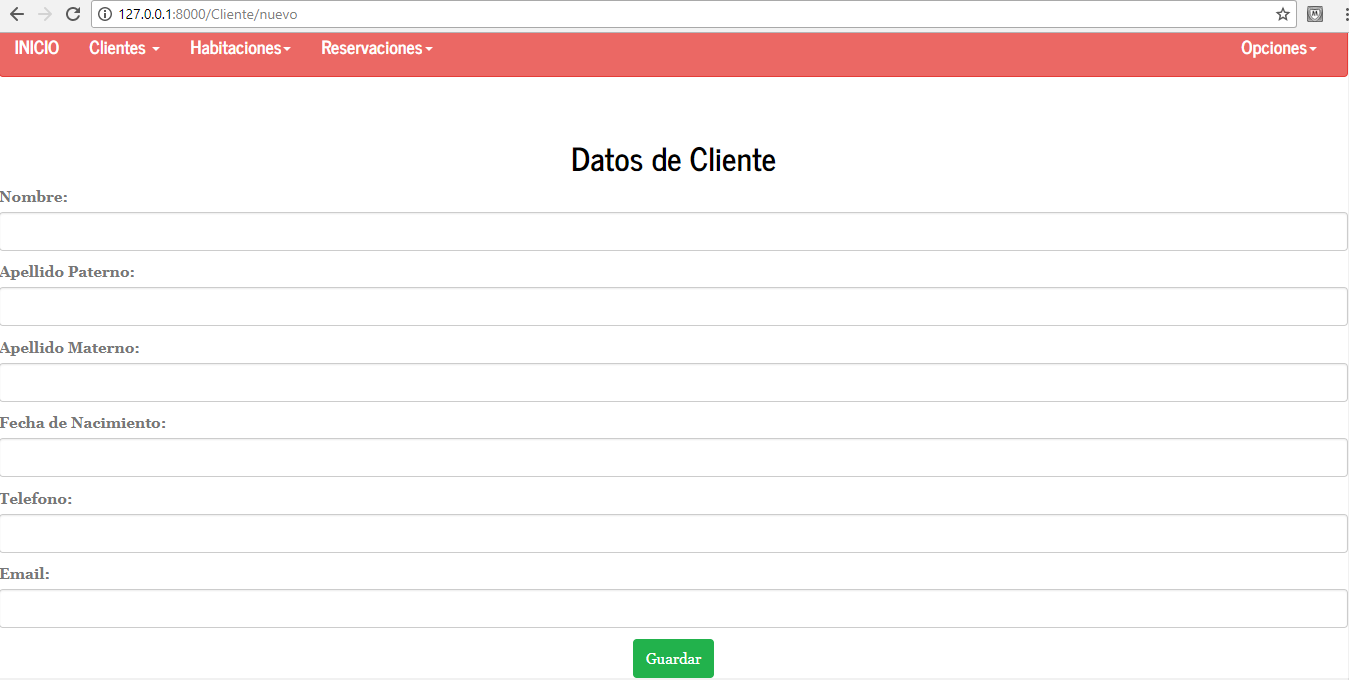
# Anexo

## Screenshots de la Web App

Página de inicio



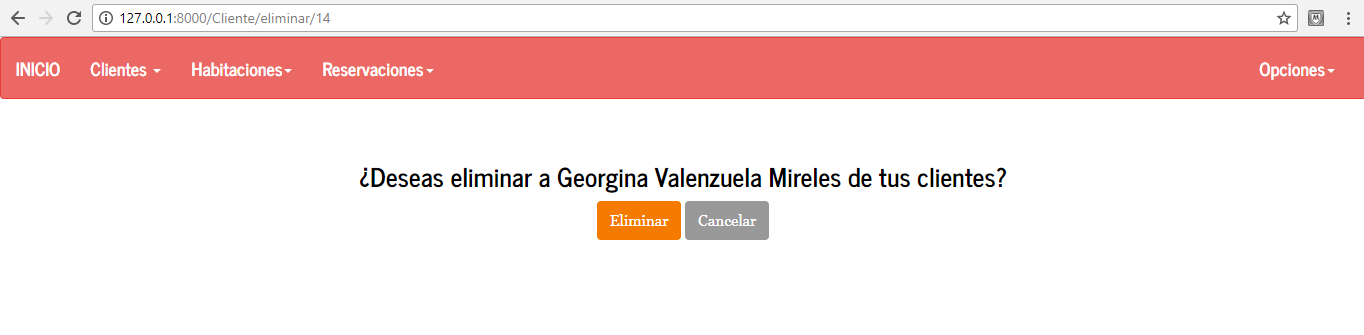
Capturar los datos del cliente



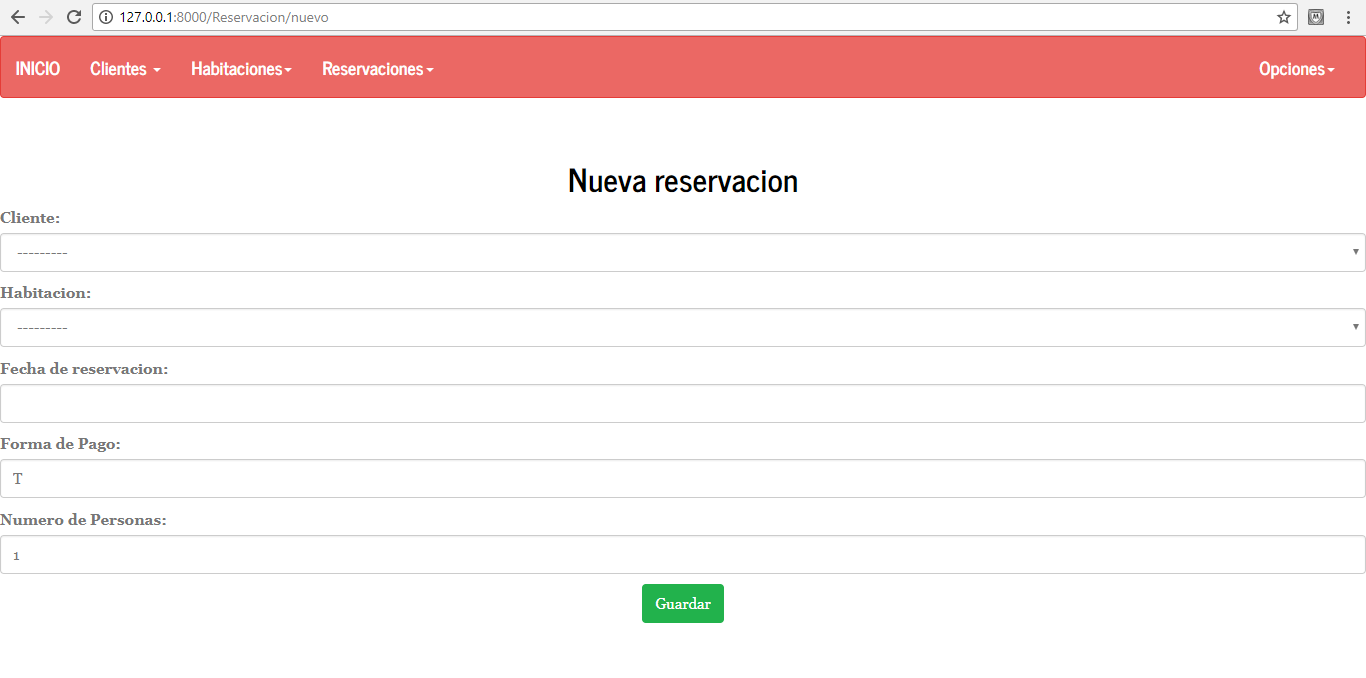
Ver Clientes Registrados



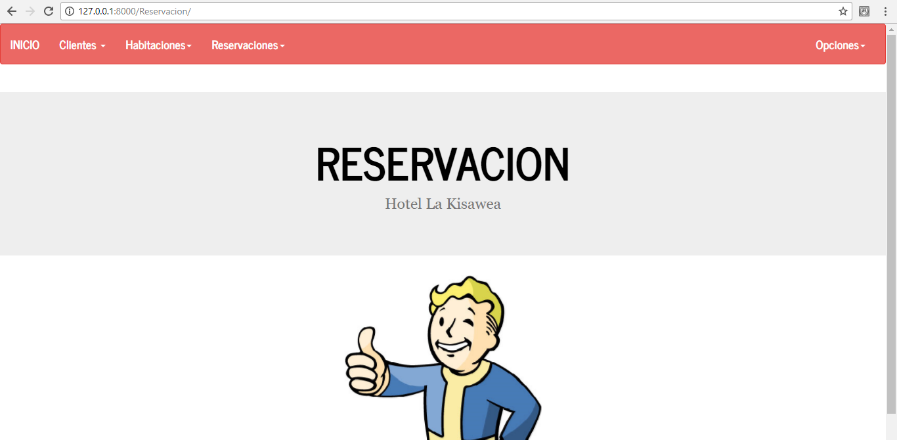
Eliminar un Cliente



Realizar una reservación



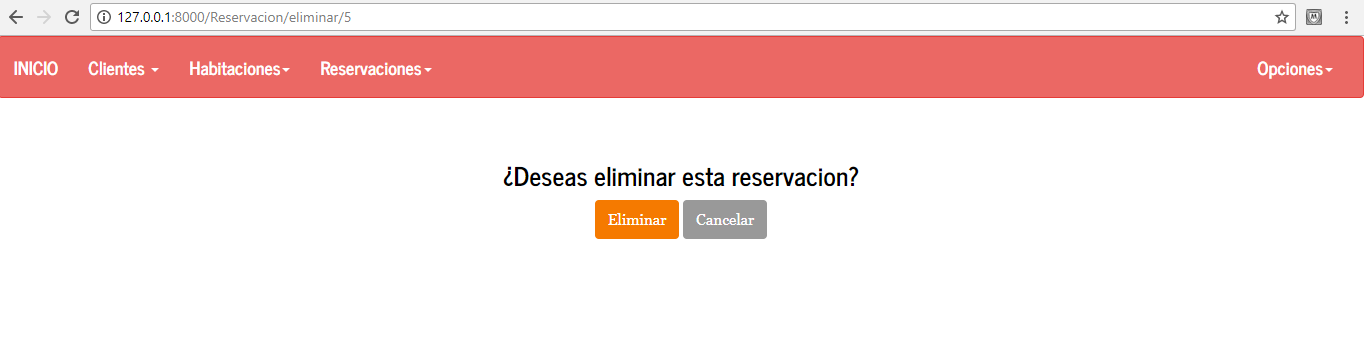
Pagina donde nos muestra que hemos realizado una reservación



Ver Reservaciones



Eliminar una reservación



Ver habitaciones



Eliminar una habitación

