



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA



Escola Tècnica
Superior d'Enginyeria
Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Desarrollo de aplicaciones en la nube (cloud computing)

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Vicente Agut Navarro

Tutor: Juan Vicente Capella Hernandez

2016-2017

Resumen

En este trabajo nos centraremos en un estudio sobre la computación en la nube, además del desarrollo en algunas de las plataformas en la nube disponibles. Para ello, estudiaremos las diferentes plataformas que podemos encontrar, para centrarnos en el desarrollo en las plataformas Windows Azure y Google App Engine, que serán las utilizadas por nuestro proyecto.

Por otro lado, analizaremos las plataformas en las nubes utilizadas, así como un estudio práctico aplicando un lenguaje de programación distinto en cada una de ellas, seguidamente se realizarán pruebas de rendimiento de cada una de ellas. Finalizaremos exponiendo sus resultados y conclusiones obtenidas, así como también las recomendaciones para desarrollar nuestra propia aplicación en la nube.

Para concluir, obtendremos las opiniones sobre nuestro trabajo y opiniones personales sobre el proyecto desarrollado.

Palabras clave: computación en la nube, plataformas, Windows Azure, Google App Engine.

Abstract

In the present work we will focus on a study of cloud computing as well as development on some of the available cloud platforms. For this, we will study the different platforms that we can find, to focus on the development in the platforms Windows Azure and Google App Engine, which will be used by our project.

On the other hand, we will analyze the platforms in the clouds used, as well as a practical study applying a different programming language in each of them, followed by performance tests of each one of them. We will conclude by presenting the results and conclusions obtained, as well as the recommendations for developing our own application in the cloud.

In conclusion, we will get opinions about our work and personal opinions about the project developed.

Keywords: cloud computing, platforms, Windows Azure, Google App Engine.

Índice

| | |
|--------------------------------------------------------------------------|----|
| 1. Introducción | 6 |
| 1.1. Justificación / Motivación | 6 |
| 1.2. Objetivos del proyecto | 7 |
| 1.3. Estructura de la memoria | 7 |
| 2. Estudio de plataformas en la nube | 8 |
| 2.1. Descripción | 8 |
| 2.1.1. Características | 9 |
| 2.1.2. Cloud computing para usuarios finales | 10 |
| 2.1.3. Cloud computing para administradores de sistemas | 10 |
| 2.1.4. Cloud computing para desarrolladores de software | 10 |
| 2.1.5. Tipos de Nubes | 11 |
| 2.1.6. Capas de nube | 12 |
| 2.1.6.1. SaaS (Software as a Service) | 12 |
| 2.1.6.2. IaaS (Infraestructura as a service) | 13 |
| 2.1.6.3. PaaS (Platform as a service) | 13 |
| 2.1.7. Servicios Web | 14 |
| 2.1.7.1. ¿Qué son los "servicios web"? | 14 |
| 2.1.7.2. Computación de alto rendimiento como un servicio (HPCaaS) | 15 |
| 2.1.7.3. Toleración a fallos en cloud computing | 15 |
| 2.1.8. Tendencias | 16 |
| 2.1.9. Ventajas y desventajas de la computación en la nube | 17 |
| 2.1.9.1. Ventajas | 17 |
| 2.1.9.2. Desventajas | 18 |
| 2.2. Diferencias entre el hosting tradicional y el cloud computing | 20 |
| 2.2.1. Ventajas de cloud computing frente a housing | 21 |
| 2.3. Google APP Engine | 22 |
| 2.3.1. Descripción | 22 |
| 2.3.2. Características | 23 |
| 2.3.3. Aplicación | 24 |
| 2.4. Microsoft Azure | 26 |
| 2.4.1. Descripción | 27 |
| 2.4.2. Copias de seguridad | 27 |



| | | |
|----------|------------------------------------------------------------------|----|
| 2.4.3. | Servicios de Windows Azure..... | 28 |
| 2.4.4. | Características de Windows Azure..... | 28 |
| 2.4.5. | Componentes | 29 |
| 2.5. | Amazon EC2 | 30 |
| 2.5.1. | Características | 30 |
| 2.5.2. | Almacenamiento persistente..... | 31 |
| 2.5.3. | Direcciones IP elásticas..... | 32 |
| 2.5.4. | Amazon CloudWatch..... | 32 |
| 2.5.5. | Escalamiento automatizado | 32 |
| 2.5.6. | Confiabilidad | 33 |
| 2.5.7. | Almacenamiento de bloque elástico | 33 |
| 2.5.8. | Tipos de instancia..... | 33 |
| 2.6. | Comparativa de plataformas..... | 34 |
| 2.6.1. | Google App Engine | 34 |
| 2.6.2. | Microsoft Azure | 34 |
| 2.6.3. | Comparación Microsoft Azure, google app Engine y Amazon EC2..... | 35 |
| 3. | Diseño e implementación de aplicaciones en la nube | 37 |
| 3.1. | Aplicación carrito compra para tienda de pintura | 37 |
| 3.1.1. | Diagrama UML de nuestra aplicación | 38 |
| 3.2. | Windows Azure | 38 |
| 3.2.1. | ASP.NET..... | 39 |
| 3.2.1.1. | Características | 39 |
| 3.2.1.2. | Modelo Code-behind..... | 39 |
| 3.2.1.3. | Controles de usuario..... | 40 |
| 3.2.1.4. | Administración del estado | 40 |
| 3.2.1.5. | Motor de plantillas | 42 |
| 3.2.1.6. | Modelos de programación en ASP.NET | 42 |
| 3.2.1.7. | Uso actual del lenguaje..... | 43 |
| 3.2.2. | Aplicación Servicolor | 43 |
| 3.2.2.1. | Pasos previos..... | 43 |
| 3.2.2.2. | Detalles de implementación | 45 |
| 3.2.2.3. | Subir aplicación en Windows Azure..... | 80 |
| 3.2.3. | Pruebas de rendimiento | 85 |
| 3.3. | Google App Engine | 86 |
| 3.3.1. | Principales características de PHP..... | 87 |
| 3.3.2. | Aplicación servicolor | 87 |

| | | |
|----------|-------------------------------------------------------------------|-----|
| 3.3.2.1. | Pasos previos | 87 |
| 3.3.2.2. | Detalles de implementación | 90 |
| 3.3.2.3. | Subir aplicación a GAE | 120 |
| 3.3.3. | Pruebas de rendimiento | 122 |
| 3.4. | Comparativa entre GAE y Windows Azure en nuestra aplicación | 123 |
| 3.4.1. | Conclusiones | 131 |
| 4. | Conclusión | 133 |
| 5. | Bibliografía | 134 |



1. Introducción

1.1. Justificación / Motivación

En la actualidad y tal vez sin darnos cuenta, utilizamos la computación en la nube, para compartir archivos con amigos o compañeros para no tener que estar cada vez en el mismo lugar todos y poder trabajar desde cualquier lugar y poder compartir cualquier archivo.

Debido a que es la computación en la nube es un campo emergente está creciendo de forma exponencial así como sus posibles aplicaciones. También se puede utilizar ésta tecnología para crear aplicaciones accesibles desde cualquier lugar y pagar solo por el uso que tenga.

Las principales ventajas que encontramos son:

- 1) Reducción de costes
- 2) Movilidad: Acceso desde cualquier dispositivo y lugar
- 3) Pagas por uso y gasto bajo control.
- 4) Tecnología siempre actualizada
- 5) Escalabilidad
- 6) Seguridad
- 7) Menos mantenimiento

1.2. Objetivos del proyecto

Los objetivos principales de nuestro Proyecto son:

- Estudiar las plataformas actuales de cloud computing, comparando las mismas y clasificarlas según las posibilidades que ofrecen.
- Diseñar y desarrollar aplicaciones hacienda uso de plataformas cloud computing.
- Evaluar las posibilidades de dos plataformas cloud computing ampliamente extendidas (Azure y GAE), atendiendo a sus capacidades, servicios disponibles, prestaciones, etc.
- Proponer recomendaciones de utilización de plataformas de cara al desarrollador en función de sus requisitos.

1.3. Estructura de la memoria

En los diferentes capítulos de la memoria encontraremos para empezar que es el cloud y sus utilidades, además también con las previsiones a medio y largo plazo de las aplicaciones cloud. Seguidamente encontraremos una explicación de las plataformas utilizadas en nuestro proyecto. Una vez hemos explicado nuestras plataformas pasaremos al diseño e implementación de nuestra aplicación en la nube, explicando paso a paso lo realizado además de realizar una comparación entre los resultados obtenidos en dichas plataformas. Para finalizar tendremos una breve conclusión de nuestro proyecto.

2. Estudio de plataformas en la nube

2.1. Descripción

La computación en la nube (del inglés cloud computing) es un paradigma que permite ofrecer servicios de computación a través de una red, que usualmente es Internet.

Cloud computing cambia todo eso al incorporar tecnología de virtualización que permite que la infraestructura física sea alquilada por poco pago por uso en comparación con las viejas maneras de diseñar software Web. Además, cambia la ecuación proporcionando herramientas de plataforma específicas para la nube para acelerar el desarrollo. El cloud computing es un modelo para permitir el acceso conveniente a la demanda a un conjunto compartido de recursos computacionales configurables (por ejemplo, redes, servidores, almacenamiento, aplicaciones y Servicios) que se pueden aprovisionar y liberar rápidamente con un mínimo esfuerzo de gestión o interacción con el proveedor de servicios.

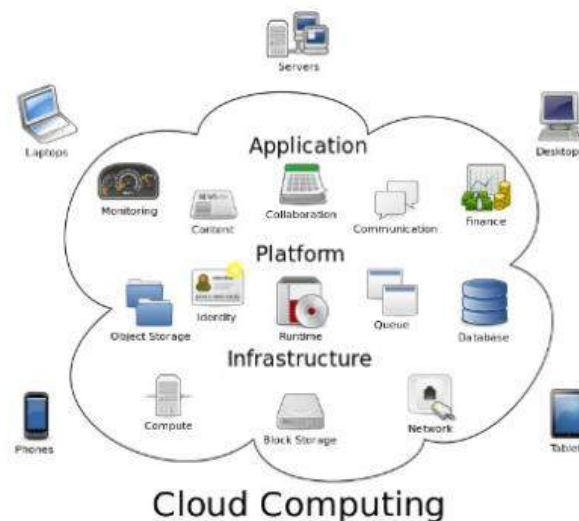


Imagen 1 Explicación gráfica cloud computing

2.1.1. Características

La computación en nube presenta las siguientes características clave:

- **Agilidad:** Capacidad de mejora para ofrecer recursos tecnológicos al usuario por parte del proveedor.
- **Coste:** Reduce barreras de entrada, ya que la infraestructura se proporciona típicamente por una tercera parte y no tiene que ser adquirida por una sola vez o tareas informáticas no frecuentes.
- **Escalabilidad y elasticidad:** aprovisionamiento de recursos sobre una base de autoservicio en casi en tiempo real, sin que los usuarios necesiten cargas de alta duración.
- **Independencia entre el dispositivo y la ubicación:** permite a los usuarios acceder a los sistemas utilizando un navegador web, independientemente de su ubicación o del dispositivo que utilice (por ejemplo, PC, teléfono móvil).
- La **tecnología de virtualización** permite compartir servidores y dispositivos de almacenamiento y una mayor utilización. Las aplicaciones pueden ser fácilmente migradas de un servidor físico a otro.
- **Rendimiento:** Los sistemas en la nube controlan y optimizan el uso de los recursos de manera automática, dicha característica permite un seguimiento, control y notificación del mismo. Esta capacidad aporta transparencia tanto para el consumidor o el proveedor de servicio.
- **Seguridad:** puede mejorar debido a la centralización de los datos. La seguridad es igual o mejor que otros sistemas tradicionales, en parte porque los proveedores son capaces de dedicar recursos a la solución de los problemas de seguridad que muchos clientes no pueden permitirse el lujo de abordar.
- **Mantenimiento:** en el caso de las aplicaciones de computación en la nube, es más sencillo, ya que no necesitan ser instalados en el ordenador de cada usuario y se puede acceder desde diferentes lugares.

2.1.2. Cloud computing para usuarios finales

La computación en la nube permite al usuario final ejecutar aplicaciones de software y acceder a datos desde cualquier lugar y hora, y desde cualquier ordenador; sin necesidad de instalar, actualizar o solucionar problemas de aplicaciones de software físicamente en un escritorio o servidor local.

Este es uno de los elementos más importantes de la computación en nube, y el por qué se ha vuelto tan popular hoy en día. La computación en la nube hace que sea más fácil hacer el trabajo en cualquier momento y desde cualquier lugar.

2.1.3. Cloud computing para administradores de sistemas.

La superioridad del modelo de nube viene cuando empezamos a darnos cuenta de que las aplicaciones de escritorio son más o menos estáticas, y las aplicaciones en la nube pueden perfeccionarse continuamente. Las aplicaciones de escritorio se deben instalar físicamente en un PC, se actualizan periódicamente, se aplican parches cuando están disponibles y se vuelven a instalar cuando el usuario se mueve a un nuevo escritorio o cuando el antiguo se bloquea. El modelo de nube elimina esos inconvenientes.

2.1.4. Cloud computing para desarrolladores de software

Cloud computing es una aplicación en la nube, gracias a esto no existe el requisito de tener nada instalado en el escritorio. Todas las actualizaciones tienen lugar en el extremo posterior, que no requieren intervención, acción, atención o paciencia del usuario final. Esto hace que sea mucho más fácil para los desarrolladores actualizar continuamente sus aplicaciones y empujar esas actualizaciones a los usuarios en tiempo real.

Al llegar a un nivel más profundo de la plataforma, el cloud computing ofrece a los desarrolladores otra ventaja crítica. Dado que la plataforma proporciona a los desarrolladores un conjunto común de servicios en la nube que ya han demostrado ser robustos, todas las aplicaciones son mucho más estables y más rápidas también.

2.1.5. Tipos de Nubes

- **La nube pública** es una nube computacional mantenida y gestionada por terceras personas no vinculadas con la organización. En este tipo de nubes tanto los datos como los procesos de varios clientes se mezclan en los servidores, sistemas de almacenamiento y otras infraestructuras de la nube. Los usuarios finales de la nube no conocen qué trabajos de otros clientes pueden estar corriendo en el mismo servidor, red, sistemas de almacenamiento, etc. Aplicaciones, almacenamiento y otros recursos están disponibles al público a través del proveedor de servicios, que es propietario de toda la infraestructura en sus centros de datos; el acceso a los servicios sólo se ofrece de manera remota, normalmente a través de internet.
- **Las nubes privadas** son una buena opción para las compañías que necesitan alta protección de datos. Las nubes privadas están en una infraestructura bajo demanda, gestionada para un solo cliente que controla qué aplicaciones debe ejecutarse y dónde. Son propietarios del servidor, red, y disco y pueden decidir qué usuarios están autorizados a utilizar la infraestructura. Al administrar internamente estos servicios, las empresas tienen la ventaja de mantener la privacidad de su información y permitir unificar el acceso a las aplicaciones corporativas de sus usuarios.
- **Las nubes híbridas** combinan los modelos de nubes públicas y privadas. Un usuario es propietario de unas partes y comparte otras, aunque de una manera controlada. Las nubes híbridas ofrecen la promesa del escalado, aprovisionada externamente, a demanda, pero añaden la complejidad de determinar cómo distribuir las aplicaciones a través de estos ambientes diferentes. Se unen mediante la tecnología, pues permiten enviar datos o aplicaciones entre ellas. Un ejemplo son los sistemas de correo electrónico empresarial.
- **Nube comunitaria.** Este modelo es aquel que se organiza con la finalidad de servir a una función o propósito común (seguridad, política...), las cuales son administradas por las organizaciones constituyentes o terceras partes.



2.1.6. Capas de nube

- Infraestructura como servicio (IAAS)
- Plataforma como servicio (PAAS)
- Software como servicio (SAAS)



Imagen 2. Diferentes servicios que podemos obtener de la nube ¹

Fuente: http://www.prospecnet.com/A_LaNube.aspx

2.1.6.1. SaaS (Software as a Service)

Consiste en poder utilizar una aplicación desde un ordenador cliente, hacia un servidor central emplazado en una empresa proveedora de sistemas y no en la compañía cliente. También puede darse el caso de compañías de mayor tamaño que alberguen sus propios servidores, y a la vez presten o vendan sus servicios de software a otras empresas del sector.

- **Características**
 - Acceso y administración a través de una red.
 - Actividades gestionadas desde ubicaciones centrales.
 - La distribución de la aplicación es más cercana al modelo uno a muchos.
 - Actualizaciones centralizadas.
 - Frecuente integración con una red mayor de software de computación.

2.1.6.2. IaaS (Infraestructura as a service)

Consiste en un modelo de aprovisionamiento, en el cual una organización coloca 'fuera de ella' el equipo usado para soportar operaciones, esto incluye el almacenamiento de la información, el hardware, servidores y componentes de redes. En ocasiones la IaaS es referida también como *Hardware as a Service* o HaaS.

- **Características**
 - El proveedor se encarga de la administración de equipos de cómputo.
 - Pago por uso.
 - Escalabilidad.
 - Disposición de un escritorio virtual.

2.1.6.3. PaaS (Platform as a service)

Este modelo consiste en incluir plataformas para crear y ejecutar aplicaciones personalizadas. Las aplicaciones PaaS también son conocidas como de sobre-demanda basadas en Web o soluciones SaaS.



- **Características**

- El proveedor además de resolver problemas en infraestructura de hardware, en este caso también se encarga del software. El cliente no necesita instalar, configurar ni dar mantenimiento a sistemas operativos, bases de datos y servidores de aplicaciones ya que todo esto lo proporciona la plataforma.
- PaaS resuelve más problemas si se compara con IaaS, ya que presenta muchas limitaciones relacionadas con el entorno de ejecución. Entre éstas se encuentra el tipo de sistema, el lenguaje de programación, el manejador de base de datos.
- Empresas como Amazon, Google, Microsoft son algunas de las empresas que emplean este modelo y hacen posible acceder a nuevas capacidades y nuevos mercados a través del navegador Web, las PaaS ofrecen un modelo más rápido y ventaja costo-beneficio para el desarrollo de aplicaciones y entrega.
- Disposición de un escritorio virtual.

2.1.7. Servicios Web

Aquí veremos otras consideraciones de computación en la nube, que no encajan muy bien en una categoría, y que son demasiado importantes como para pasar por alto.

2.1.7.1. ¿Qué son los "servicios web"?

Los servicios web suelen ser considerados como el dominio de los programadores web, no los usuarios finales. Es una técnica de programación que implica el uso de subrutinas remotas, que pueden ser llamadas a través de la nube, como hacer un cálculo o autenticar a los usuarios.

En el caso del cloud computing, los servicios web permiten a los programadores crear programas en la nube (SaaS) con maneras de administrar la infraestructura en la nube o integrarse con otros programas en la nube.

2.1.7.2. Computación de alto rendimiento como un servicio (HPCaaS)

El software especial de nube de HPC, incluyendo software de código abierto como Univa UD, hace posible encender y apagar los nodos de computación según sea necesario, mientras se orquestan cálculos intensivos en esos nodos. Con cloud HPC, el concepto es que un supercomputador nunca descansa ocioso, no se vuelve comparativamente anticuado en unos pocos años.

2.1.7.3. Toleración a fallos en cloud computing

Tolerante a fallos significa que si ocurre un fallo, la tecnología y los protocolos están en su lugar para corregir automáticamente ese fallo en tiempo real. Este es el corazón de la recuperación de desastres, y es parte del modelo de cloud computing.

Para el proveedor de la arquitectura en nube, esto significa tener centros de datos redundantes. Para el usuario de servicios en la nube, significa tener acceso constante y tiempo de actividad garantizado a aplicaciones y datos, sin tener que preocuparse por la recuperación de la pérdida de datos o la recuperación de desastres.

En GAE, Windows Azure y Amazon EC2, tienen muchos centros de datos distribuidos por todo el mundo, con lo cual si ocurre algo en alguno de ellos la información está redundada en los diferentes centros, por lo tanto no tenemos que preocuparnos por nuestros datos, los cuales siempre estarán disponibles y garantizados de que podamos acceder a ellos.



2.1.8. Tendencias

Actualmente las empresas utilizan el cloud computing por los siguientes motivos:

- Necesidad económica
- Apoyo de los principales proveedores de software
- Demanda de pequeñas empresas para las características de gama alta
- Demanda de los usuarios empresariales para soluciones más rentables
- Necesidad de herramientas colaborativas
- La tecnología Cloud ha Ya pasó la etapa de prueba de tierra

Además, los principales proveedores de software tienen muchas ofertas para la nueva demanda del mercado de cloud computing.

Las pequeñas empresas y las empresas SOHO también tienen la nube a su disposición, así, con ofertas de los principales vendedores como Microsoft, Google, Amazon e incluso Apple ofrece una amplia gama de aplicaciones basadas en la nube y servicios que promueven.

También los usuarios cada vez utilizan más Google Apps, Microsoft Live y Apple Mobile Me y al final exigirán la misma funcionalidad en el lugar de trabajo.

Otra característica será la descentralización del lugar de trabajo, la tele conmutación y soluciones de trabajo en casa exigen que las tecnologías de colaboración funcionen, y esto ahora sólo es posible a través de la tecnología de cloud computing.

Las principales previsiones para el futuro próximo que se desprenden de las principales referencias consultadas son:

- ❖ Subirá la demanda de infraestructura en la nube y los precios caerán
- ❖ El desarrollo en la nube se volverá cada vez más sencillo para los usuarios con lo cual habrá mayor competencia.

- ❖ Las oficinas físicas irán desapareciendo poco a poco, porque todos los servicios estarán disponibles en la nube.
- ❖ Las empresas se descentralizan gracias al cloud computing, que conllevará a mayor nivel de externalización.
- ❖ Los teléfonos inteligentes se podrán utilizar como máquina de trabajo real cosa que estamos viendo cómo va ocurriendo poco a poco.
- ❖ Las aplicaciones de computación en nube a nivel de empresa reemplazarán gradualmente esas enormes implementaciones locales con un enfoque más modular.

2.1.9. Ventajas y desventajas de la computación en la nube

2.1.9.1. Ventajas

- **Integración probada de servicios Red.** Por su naturaleza, la tecnología de cloud computing se puede integrar con mucha mayor facilidad y rapidez con el resto de las aplicaciones empresariales (tanto software tradicional como Cloud Computing basado en infraestructuras), ya sean desarrolladas de manera interna o externa.
- **Prestación de servicios a nivel mundial.** Las infraestructuras de cloud computing proporcionan mayor capacidad de adaptación, recuperación completa de pérdida de datos (con copias de seguridad) y reducción al mínimo de los tiempos de inactividad.
- Una **infraestructura 100% de cloud computing** permite también al proveedor de contenidos o servicios en la nube prescindir de instalar cualquier tipo de software, ya que éste es provisto por el proveedor de la infraestructura o la plataforma en la nube. Un gran beneficio del cloud computing es la simplicidad y el hecho de que requiera mucha menor inversión para empezar a trabajar.



- **Implementación más rápida y con menos riesgos**, ya que se comienza a trabajar más rápido y no es necesaria una gran inversión. Las aplicaciones del cloud computing suelen estar disponibles en cuestión de días u horas en lugar de semanas o meses, incluso con un nivel considerable de personalización o integración.
- **Actualizaciones automáticas que no afectan negativamente a los recursos de TI.** Al actualizar a la última versión de las aplicaciones, el usuario se ve obligado a dedicar tiempo y recursos para volver a personalizar e integrar la aplicación. Con el cloud computing no hay que decidir entre actualizar y conservar el trabajo, dado que esas personalizaciones e integraciones se conservan automáticamente durante la actualización.
- Otra de las ventajas es el **ahorro de tiempo**. Si bien trabajamos en un proyecto el cual los integrantes se encuentran físicamente en zonas geográficas lejanas, con las aplicaciones en la nube nos ahorramos el tener que estar todo el tiempo enviando correos ya que podemos aplicar los cambios instantáneamente desde cualquier lugar.
- **Contribuye al uso eficiente de la energía.** En este caso, a la energía requerida para el funcionamiento de la infraestructura. En los centros de datos tradicionales, los servidores consumen mucha más energía de la requerida realmente. En cambio, en las nubes, la energía consumida es sólo la necesaria, reduciendo notablemente el desperdicio.
 - La centralización de las aplicaciones y el almacenamiento de los datos origina una interdependencia de los proveedores de servicios.
 - La disponibilidad de las aplicaciones está sujeta a la disponibilidad de acceso a Internet.

2.1.9.2. Desventajas

- **Privacidad:** Los **datos "sensibles" del negocio no residen en las instalaciones de las empresas**, lo que podría generar un contexto de alta vulnerabilidad para la sustracción o robo de información.
- **La confiabilidad** de los servicios **depende** de la "salud" tecnológica y financiera **de los proveedores de servicios en nube**. Empresas emergentes o alianzas entre empresas podrían crear un ambiente propicio para el monopolio y el crecimiento exagerado en los servicios.

- La **disponibilidad de servicios** altamente especializados podría tardar meses o incluso años para que sean factibles de ser desplegados en la red, además si el sistema de redundancia falla y no logra mantener el servicio disponible para el usuario, éste no puede realizar ninguna acción correctiva para restablecer el servicio. En tal caso, deberíamos esperar a que el problema se resuelva por parte del proveedor.
- **Falta de control de recursos.** Al estar toda la infraestructura y la aplicación corriendo sobre servidores en la nube, es decir, del lado del proveedor, carecemos por completo de control sobre los recursos e incluso sobre su información, una vez subida a la nube.
- **Seguridad.** La información de la empresa debe recorrer diferentes nodos para llegar a su destino, cada uno de ellos (y sus canales) son un foco de inseguridad. Si se utilizan protocolos seguros, HTTPS por ejemplo, la velocidad total disminuye debido a la sobrecarga que éstos requieren.
- **Escalabilidad a largo plazo.** A medida que más usuarios empiecen a compartir la infraestructura de la nube, la sobrecarga en los servidores de los proveedores aumentará, si no se posee un esquema de crecimiento óptimo puede llevar a degradaciones en el servicio o altos niveles de jitter(varianza de los retardos en la red).



2.2. Diferencias entre el hosting tradicional y el cloud computing

Los **sistemas basados en la nube** hacen uso de servidores virtuales interconectados entre sí, lo que permite que en caso de caída de una de esas máquinas, se pueda levantar otra de forma inmediata sin que el servicio se vea afectado. Este conjunto de máquinas virtuales trabajan como una sola, ofreciendo mayores posibilidades al usuario.

Otra importante diferencia técnica radica en la disponibilidad de los recursos. **El hosting** siempre tienen un límite en cuanto a velocidad y capacidad de almacenamiento, mientras que los servicios Cloud, al tratarse de varias máquinas que trabajan como una sola, estos recursos son ilimitados dentro de la nube, pudiendo ampliar recursos de forma inmediata y sin necesidad de parar el servicio, cosa que no se puede hacer en los alojamientos tradicionales, donde la única solución sería pasar a un plan superior, lo que conlleva una parada del servicio mientras que hace el cambio.

Los servidores Cloud suelen utilizar tecnologías de virtualización que emulan más y mejor el comportamiento de un servidor físico, lo que se traduce en que este tipo de servicios ofrece una asignación de recursos como CPU o memoria RAM muy parecidos a las que ofrece un servidor físico.

Por último el Cloud ofrece un servicio más avanzado que el hosting, ofreciendo un servicio más adecuado para cualquier proyecto.

2.2.1. Ventajas de cloud computing frente a housing

Flexibilidad y escalabilidad: aunque los servidores tradicionales también pueden redimensionar para hacerlos crecer, no tienen la elasticidad de hacerlo en tiempo real, sin impactos de paradas en tiempos de producción ni hacer esto a unos costes más reducidos y ajustados

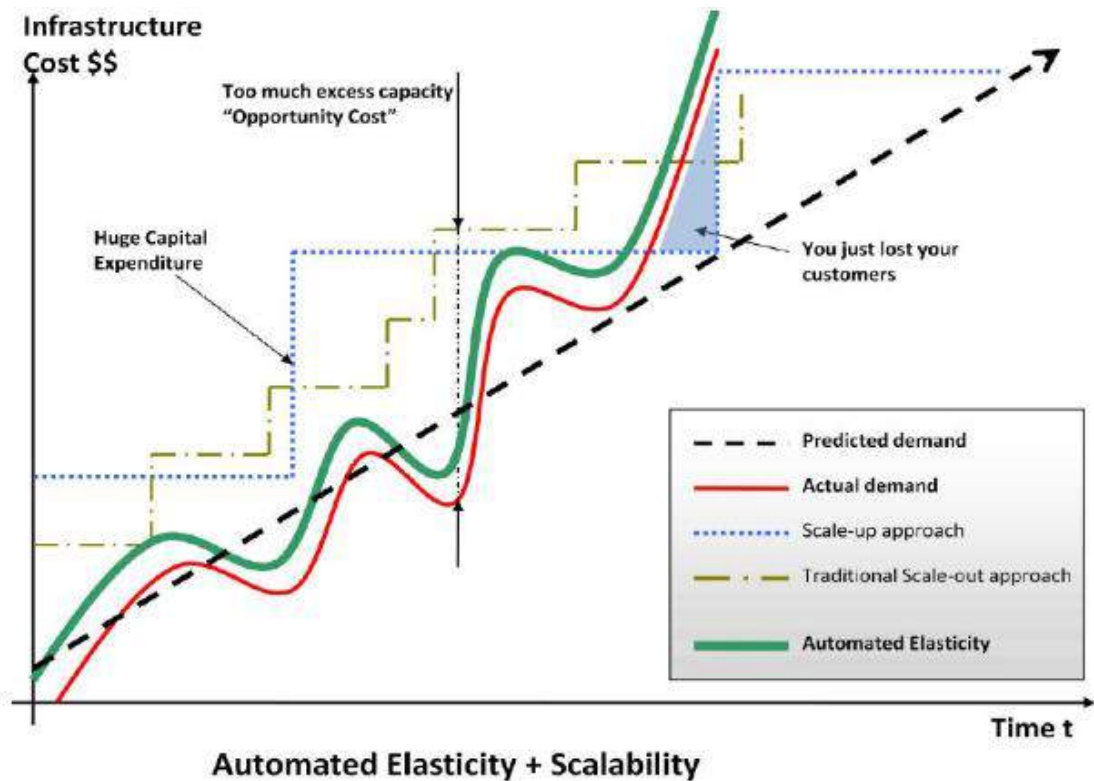


Imagen 3. Gráfica sobre la escalabilidad del cloud computing ³

Fuente: <https://www.descom.es/blog/cloud/diferencia-entre-hosting-tradicional-y-cloud-computing.html>

Se puede observar en la Imagen 3 con el cloud computing la elasticidad es automática a medida que va aumentando o bajando la demanda frente a los servidores tradicionales que les cuesta más adaptarse a cambios.

2.3. Google APP Engine

2.3.1. Descripción

Google App Engine o también conocido más comúnmente como GAE o App Engine es la infraestructura de producción de Google de forma gratuita como plataforma de desarrollo y hospedaje de aplicaciones web, además de almacenamiento de datos y la creación de redes de alta velocidad mediante la ejecución en la parte superior de infraestructura masiva de Google.

El servicio fue lanzado el 7 de abril del 2008 como un servicio de cloud pero a diferencia de otros servicios en la nube como *Amazon Web Services* o *Azure Services Platform* de Microsoft, el servicio ofrecido por Google es un servicio de Plataforma como Servicio y no de Infraestructura como Servicio.

Google App Engine te permite ejecutar tus aplicaciones web en la infraestructura de Google. Las aplicaciones App Engine son fáciles de crear, mantener y actualizar al ir aumentando el tráfico y las necesidades de almacenamiento de datos. Con App Engine, no necesitarás utilizar ningún servidor: sólo tendrás que subir tu aplicación para que tus usuarios puedan empezar a utilizarla.

Puedes publicar tu aplicación mediante un nombre de dominio gratuito en el dominio *appspot.com* o utilizar Google Apps para publicarla desde tu propio dominio. Podrás compartir tu aplicación con todo el mundo o limitar el acceso a los miembros de tu organización.

Cada cuenta gratuita permite utilizar hasta 500 MB de almacenamiento permanente y suficiente cantidad de ancho de banda y CPU para casi 5 millones de visitas mensuales.

Google App Engine permite desarrollar fácilmente aplicaciones que se ejecuten de forma fiable, incluso con pesadas cargas de trabajo y grandes cantidades de datos. El entorno incluye las siguientes funciones:

- Servidor web dinámico, totalmente compatible con las tecnologías web más comunes.
- Almacenamiento permanente con funciones de consulta, orden y transacciones.
- Escalado automático y balanceo de carga.
- API para autenticar usuarios y enviar correo electrónico a través de las cuentas de Google.
- Un completo entorno de desarrollo local que simula Google App Engine en tu equipo.

2.3.2. Características

Permite crear aplicaciones web en los mismos sistemas escalables con los que funcionan las aplicaciones de Google.

1. Alta escalabilidad.
2. Pago por uso: Google App Engine es gratuito para empezar a usar la aplicación y pagaremos solo por el uso que le demos, por lo tanto viene muy bien para inicializarnos en dicho entorno. Además el coste de la aplicación sólo será el del desarrollo en sí, no habrá otro tipo de costes adicionales (mantenimiento hardware, compra de equipos informáticos, electrónica de red para garantizar ancho de banda, etc.).
3. No serán necesarios gastos en hardware y mantenimiento de sistemas: con Google App Engine se utilizarán los equipos y sistemas de Google, por lo que no tendrás que preocuparte por el hardware, será Google quien lo haga por ti.
4. Soporta varios lenguajes de programación tales como: Python, java, PHP, go, node.js y Ruby.



5. Fiable y con soporte oficial de google.
6. Entorno seguro.

2.3.3. Aplicación

- El usuario se autentica con una cuenta de Google.
- Desde la cuenta de Google App Engine podrás ver las estadísticas de uso de CPU y demás de tus visitas.
- Zona de pruebas. Aísla la aplicación en su propio entorno seguro de confianza, totalmente independiente del hardware, del sistema operativo y de la ubicación física del servidor web. La aplicación solo podrá acceder a otros equipos de internet a través de los servicios de correo y extracción de URL proporcionados por App Engine. El resto de equipos solo podrán conectarse a la App por medio de los protocolos HTTP y HTTPS.
- Control. Puedes registrar 10 Apps por cuenta de desarrollador.
- Desarrollo local. Google App Engine permite que generes en local un entorno idéntico al de Google App Engine en la nube. Es decir, podrás realizar tu App y probarla con la máxima seguridad que una vez la subas, todas las características de tu App funcionarán a la perfección.
- Cron. Además de las lógicas solicitudes web, tu aplicación puede realizar tareas programadas según el desarrollador configure (cada día, cada hora). Las tareas programadas se conocen como 'tareas cron' ya que el servicio 'cron' (administrador regular de procesos en segundo plano) que ejecuta procesos o guiones a intervalos regulares. Los procesos que deben ejecutarse y la hora en la que deben hacerlo se especifican en el fichero crontab que es el que las gestiona.

- Frontends (Parte del software que interactúa con el o los usuarios)
 - Balanceo de carga.
 - Transfiere la solicitud a los servidores de aplicaciones o servidores de archivos estáticos.
 - Interactúa con usuarios finales haciendo optimizaciones como la compresión de los datos de respuesta.

- Servidores de aplicaciones:

Servidor de aplicaciones

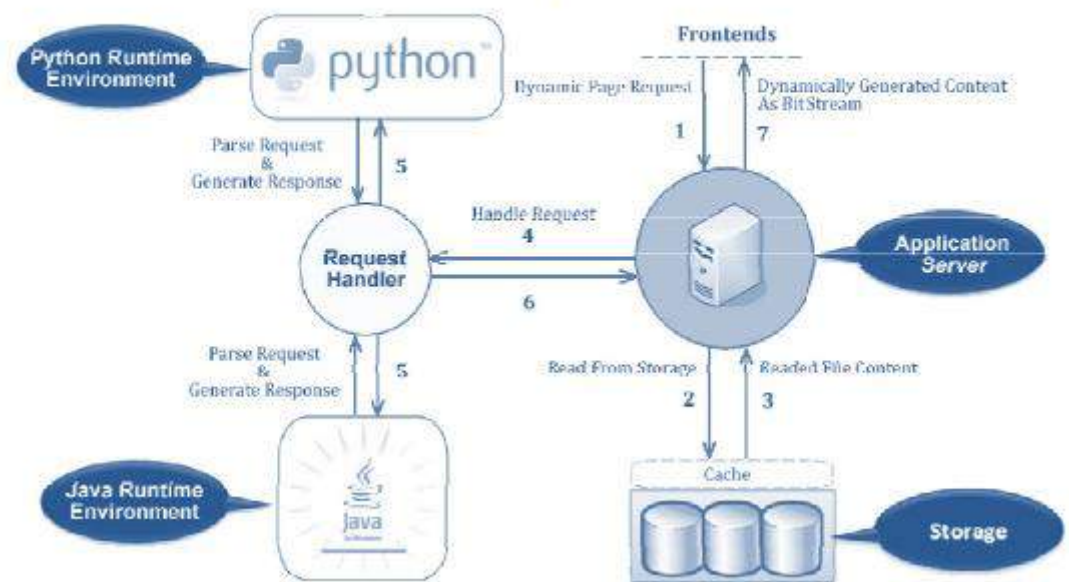


Imagen 4. Ejemplo de servidor de aplicación.

En la imagen 4 cargamos el frontend, seguidamente buscamos la aplicación en el servidor, luego lee los datos que tenemos almacenados en una base de datos. Una vez ya tenemos estos datos buscara dependiendo del lenguaje la aplicación y la ejecutará y finalmente le devolverá el resultado al usuario.

- Gestiona los recursos locales disponibles para la aplicación incluyendo ciclos de CPU, memoria y tiempo de ejecución.
- Controla los manipuladores de la conducta y el rendimiento y los mantiene separados.
- Asegura que las aplicaciones que no consumen recursos del sistema no interfiera con otras aplicaciones.

- Almacén de datos
 - Bigtable es un sistema de gestión de base de datos creado por Google con las características de ser: distribuido, de alta eficiencia y propietario.
 - Bigtable almacena la información en tablas multidimensionales cuyas celdas están, en su mayoría, sin utilizar. Para poder manejar la información, las tablas se dividen por columnas y son almacenadas como 'tabletas' de unos 200Mbytes cada una. Cada máquina almacena 100 tabletas mediante 'Google File System'. Esta disposición permite un sistema de balanceo de carga y una rápida recomposición del sistema si una máquina deja de funcionar.
- Otros servicios:
 - Extracción de URL: Recupera recursos web mediante la misma infraestructura de alta velocidad de Google.
 - Correo: La App puede enviar correos electrónicos a los usuarios por medio de las herramientas de Google.
 - Memcache: App Engine proporciona una memoria caché de valores-claves de alto rendimiento accesible desde varias instancias de tu aplicación. Resulta útil para datos que no necesitan las funciones de persistencia y transacciones del almacén de datos.
 - Este servicio es el mejor para usarlo como cache.

2.4. Microsoft Azure

Azure es la plataforma como servicio de Microsoft. Esto equivale a que Microsoft tiene cientos de centros de datos, con una gran cantidad de recursos, los cuales pueden ser consumidos por personas o empresas desde cualquier parte del mundo, pagando sólo por lo que se utilice. Además de almacenar y montar una base de datos en la nube, nos permite ofrecer servicios avanzados de plataforma cosa que es una diferencia importante con las demás plataformas de Cloud.

2.4.1. Descripción

Es una plataforma ofrecida como servicio y alojada en los Data Centers de Microsoft. Anunciada en el Professional Developers Conference de Microsoft (PDC) del 2008 en su versión beta, pasó a ser un producto comercial el 1 de enero de 2010. Windows Azure es una plataforma general que tiene diferentes servicios para aplicaciones, desde servicios que alojan aplicaciones en alguno de los centros de procesamiento de datos de Microsoft para que se ejecute sobre su infraestructura (Cloud Computing) hasta servicios de comunicación segura y federación entre aplicaciones.

2.4.2. Copias de seguridad

Windows ofrece una manera de proteger la información importante con una copia de seguridad automática dentro de un servicio de almacenamiento. Las copias de seguridad quedan cifradas antes de la transmisión y se almacenan cifradas en Windows Azure. Estas copias de seguridad están fuera de sitio, lejos de su centro de datos, lo que reduce la necesidad de asegurar y proteger los medios de copia de seguridad en el lugar.

La administración de copias de seguridad en la nube usa herramientas de copia de seguridad conocida en Windows Server, Windows Server Essentials, o el Administrador de System Center Data Protection. Estas herramientas proporcionan experiencias similares al configurar, supervisar y recuperar copias de seguridad ya sea en el disco local o el almacenamiento de Windows Azure, o puede utilizar el software propio del agente. Después de que los datos se copian a la nube, los usuarios autorizados pueden recuperar fácilmente copias de seguridad de cualquier servidor. También se pueden utilizar Copias de seguridad incremental para asegurar el uso eficiente de almacenamiento y un menor consumo de ancho de banda, al mismo tiempo que permite la recuperación de punto en el tiempo de varias versiones de los datos.



2.4.3. Servicios de Windows Azure

Dentro de la plataforma, el servicio de Windows Azure es el encargado de proporcionar el alojamiento de las aplicaciones y el almacenamiento no relacional. Se puede desarrollar en .NET, PHP, C++, Ruby, Java, PHP, etc. Además del servicio de ejecución, dispone de diferentes mecanismos de almacenamiento de datos: tablas NO SQL, blobs, blobs para streaming, colas de mensajes o 'drives' NTFS para operaciones de lectura / escritura a disco.

2.4.4. Características de Windows Azure

Proceso: el servicio de proceso de Windows Azure ejecuta aplicaciones basadas en Windows Server. Estas aplicaciones se pueden crear mediante .NET Framework en lenguajes como C# y Visual Basic, o implementar sin .NET en C++, Java y otros lenguajes.

Almacenamiento: objetos binarios grandes (blobs) proporcionan colas para la comunicación entre los componentes de las aplicaciones de Windows Azure y ofrece un tipo de tablas con un lenguaje de consulta simple.

Servicios de infraestructura: posibilidad de desplegar de una forma sencilla máquinas virtuales con Windows Server o con distribuciones de Linux.

Controlador de tejido: Windows Azure se ejecuta en un gran número de máquinas. El controlador de tejido se encarga de combinar las máquinas en un solo centro de datos de Windows Azure formando un conjunto armónico. Los servicios de proceso y almacenamiento de Windows Azure se implementan encima de toda esta eficacia de procesamiento.

Red de entrega de contenido (CDN): el almacenamiento en caché de los datos a los que se accede frecuentemente cerca de sus usuarios agiliza el acceso a esos datos.

Connect: organizaciones interactúan con aplicaciones en la nube como si estuvieran dentro del propio firewall de la organización.

Administración de identidad y acceso: Active Directory permite gestionar de forma centralizada y sencilla el control de acceso y la identidad. Esto es perfecto para la administración de cuentas y la sincronización con directorios locales.

2.4.5. Componentes

Windows Azure Compute es una plataforma para hospedar y administrar aplicaciones en los centros de datos de Microsoft. Una aplicación de Windows Azure consta de uno o varios componentes denominados 'roles.' Los roles pueden ser de tres tipos: rol web, rol de trabajo y rol de máquina virtual (VM).

Windows Azure Storage tiene servicios de básicos como parte de la cuenta de almacenamiento de Windows Azure.

Microsoft SQL Azure es un servicio de base de datos en la nube basado en las tecnologías de SQL Server. Los servicios de SQL Azure incluyen: Base de datos SQL Azure, SQL Azure Reporting y SQL Azure Data Sync Aspectos destacados de la base de datos de SQL Azure.

Azure AppFabric El servicio de Appfabric (en fase beta se llamaba .NET Services) ofrece diferentes servicios para aplicaciones. Los servicios de autenticación, autorización y mensajería permiten la comunicación segura entre aplicaciones y servicios desplegados tanto en la nube y en local.

Azure Market Place es un mercado en línea global compartir, comprar y vender aplicaciones SaaS completas y conjuntos de datos.

Azure Virtual Network es una serie de funciones de red. Windows Azure Connect es la primera característica de Azure Virtual Network que configura la conectividad de red basada en IP entre recursos locales y de Windows Azure. Windows Azure Traffic Manager equilibra la carga del tráfico en servicios hospedados.

2.5. Amazon EC2

Amazon Elastic Compute Cloud (Amazon EC2) es una parte central de la plataforma de cómputo en la nube de la empresa Amazon.com denominada Amazon Web Services (AWS). EC2 permite a los usuarios alquilar computadores virtuales en los cuales poder ejecutar sus propias aplicaciones.

EC2 fomenta el despliegue escalable de aplicaciones proporcionando un servicio web a través del cual podemos iniciar una Amazon Machine Image (AMI) para configurar una máquina virtual , lo que Amazon llama una "instancia", que contiene el software deseado. Podemos crear, iniciar y terminar servidor -instancias según sea necesario, el **pago por hora** para los servidores activos - de ahí el término "elástico". EC2 nos proporciona el control sobre la ubicación geográfica de los casos que permite la latencia optimización y altos niveles de redundancia .

2.5.1. Características

Amazon EC2 se apoya en las tecnologías de virtualización, permitiendo utilizar gran variedad de sistemas operativos a través de sus interfaces de servicios web, personalizarlos, gestionar permisos de acceso a la red y ejecutar tantos sistemas como desee.

EC2 permite el despliegue escalable de aplicaciones proveyendo un servicio Web a través del cual podemos montar una Imagen de Máquina Amazon para crear una máquina virtual, llamada por Amazon "instancia", la cual contendrá cualquier software deseado. Un usuario puede crear, lanzar y finalizar instancias de servidor tanto como necesite, pagando por hora por servidor activo, de ahí el término "elástico". EC2 provee a los usuarios control sobre localizaciones geográficas de instancias que permiten la optimización de latencia y altos niveles de redundancia.

Amazon EC2 posee una interfaz de servicios web para iniciar y configurar el servicio. Proporciona un control completo de los recursos y reduce el tiempo de arranque de sus servidores, lo que permite escalar recursos rápidamente según las necesidades del usuario. Asimismo, provee herramientas de recuperación de datos y fuerte aislamiento frente a otros procesos realizados en sus máquinas.

EC2 está disponible para los siguientes sistemas operativos: Microsoft Windows, Linux y FreeBSD. También añadir que la disponibilidad es solo en inglés y es un servidor virtual privado.

El coste suele ser de 0.0065€ / hora (4.75€/mes) para las máquinas virtuales más pequeñas.

2.5.2. Almacenamiento persistente

Una instancia EC2 puede ser lanzado de dos tipos de almacenamiento para su disco de arranque o dispositivo raíz. La primera opción es un disco local como un dispositivo raíz (originalmente la única opción). La segunda opción es utilizar un volumen de EBS (Amazon Elastic Block Store) como un dispositivo raíz. Almacén de instancias son almacenamiento temporal, que resisten al reiniciar una instancia EC2, pero cuando la instancia se detiene o finaliza (por ejemplo, por una llamada de API, o debido a un fallo), se pierde este almacenamiento.

Simple Storage Service (S3) es un sistema de almacenamiento en el que los datos son accesibles a instancias de EC2, o directamente por la red a las personas que llaman adecuadamente autenticado (toda comunicación es a través de HTTP). Amazon no cobra por el ancho de banda para las comunicaciones entre las instancias de EC2 y S3 de almacenamiento en la misma región.

Almacenamiento basado en S3 tiene un precio por gigabyte al mes. Las aplicaciones acceden a través de un API S3. Por ejemplo, Apache Hadoop es compatible con una s3 especial: sistema de archivos para apoyar la lectura y la escritura en el almacenamiento S3 durante un trabajo de MapReduce. También hay sistemas de archivos para Linux S3, permite montar un almacén de archivos S3, como si se tratara de almacenamiento local.



2.5.3. Direcciones IP elásticas

Función de dirección IP elástica de Amazon es similar a la dirección IP estática en los centros de datos tradicionales, con una diferencia clave. Podemos asignar una dirección IP mediante programación elástica a cualquier instancia de máquina virtual sin la ayuda de un administrador de red y sin tener que esperar a que el DNS propague la unión. En este sentido, una dirección IP elástica pertenece a la cuenta y no a una instancia de máquina virtual.

2.5.4. Amazon CloudWatch

Amazon CloudWatch es un servicio web que proporciona monitorización en tiempo real a los clientes de EC2 de Amazon en su utilización de recursos tales como CPU, disco y red. CloudWatch no proporciona ninguna memoria, espacio en disco o cargar las métricas promedio sin ejecutar software adicional en la instancia. Amazon ofrece scripts de ejemplo para las instancias de Linux. Los datos se agregan y proporciona a través de la consola de gestión AWS. También se puede acceder a través de las herramientas de línea de comandos y API de Web, si queremos monitorizar los recursos EC2 a través de su software de supervisión de la empresa.

Los indicadores recogidos por Amazon CloudWatch permite a la escala automática función para añadir o eliminar dinámicamente instancias de EC2. Debemos pagar por el número de casos de vigilancia.

2.5.5. Escalamiento automatizado

Función de auto escalable de Amazon EC2 de le permite adaptarse automáticamente a la capacidad de tráfico del sitio de computación. El basado en programación, basados en reglas mecanismos de escalado automático son fáciles de usar y eficiente para aplicaciones sencillas. Sin embargo, un problema potencial es que las máquinas virtuales pueden tardar hasta varios minutos para estar listo para usar, que no son adecuados para aplicaciones de tiempo crítico. El tiempo de inicio VM dependen de tamaño de la Imagen, el tipo VM, ubicaciones de los centros de datos, etc.

2.5.6. Confiabilidad

Para hacer EC2 más tolerante a fallos, Amazon ha diseñado zonas de disponibilidad que están diseñados para estar aislado de fracasos en otras zonas de disponibilidad. Zonas de disponibilidad no comparten la misma infraestructura. Aplicaciones que se ejecutan en más de una zona de disponibilidad pueden lograr una mayor disponibilidad.

EC2 proporciona a los usuarios el control sobre la ubicación geográfica de los casos que permite la optimización de latencia y un alto nivel de redundancia. Por ejemplo, para reducir al mínimo el tiempo de inactividad, un usuario puede establecer instancias de servidor en múltiples zonas que están aislados unos de otros para la mayoría de causas de fallo de tal manera que una copia de seguridad del otro.

2.5.7. Almacenamiento de bloque elástico

Amazon Elastic Block Store (EBS) proporciona dispositivos de bloque que se encuentran sin procesar, se pueden conectar a instancias de Amazon EC2. En un caso de uso típico, esto incluiría formatear el dispositivo con un sistema de archivos y montarlo. Además, EBS soporta una serie de funciones avanzadas de almacenamiento, incluyendo snapshotting y clonación.

2.5.8. Tipos de instancia

EC2 utiliza Xen de virtualización. Cada máquina virtual, llama una "instancia", funciona como un servidor privado virtual

Tipos de instancia:

- En la demanda: pagar por hora sin compromiso.
- Reservado: alquiler instancias con una sola vez descuentos de recepción de pago en el cargo por hora.
- Spot: servicio basado en la subasta: ejecuta los trabajos sólo si el precio de contado es inferior a la oferta especificada por postor. El precio spot se afirma que se basará la oferta y la demanda.



2.6. Comparativa de plataformas

2.6.1. Google App Engine

La interfaz de la nube de Google es muy fácil de utilizar y con un rápido despliegue en la plataforma.

En GAE hay montones de herramientas, incluyendo LAMP, Ruby y Hadoop. Los usuarios también pueden desplegar Hadoop especificando muchos parámetros, así como el tamaño de la instancia y el número o ubicación de almacenamiento de datos.

Google App Engine es otra útil herramienta de implementación. App Engine es una PaaS de alto rendimiento que soporta lenguajes de programación Python, Java, PHP y Go, etc. La plataforma trabaja con gran variedad de opciones de almacenamiento, incluyendo Cloud SQL, Datastore y Blobstore. Además el motor también proporciona acceso a la API de búsqueda.

La rapidez de escalabilidad y despliegue, además de herramientas de datos grandes fáciles de usar, hacen que Google Compute Engine y App Engine sean atractivas para el análisis y big data.

2.6.2. Microsoft Azure

La combinación de Microsoft Azure, Windows Server y Microsoft System Center ofrece a los administradores de nube o en las instalaciones una plataforma consolidada para la gestión de los componentes de nube híbridos.

Microsoft System Center ofrece funciones de gestión de nube, como el aprovisionamiento de infraestructura y supervisión del rendimiento. Su controlador de aplicaciones también proporciona una mirada unificada a la infraestructura a través de sistemas Azure o en las instalaciones. Visual Studio está completamente acoplado a la plataforma y es muy útil para los desarrolladores.

StorSimple es el servicio de almacenamiento híbrido de Microsoft que incluye almacenamiento principal, copia de seguridad y archivo, soporta además nubes híbridas. Los usuarios acceden a StorSimple a través de un portal basado en Azure. El servicio de almacenamiento híbrido de Microsoft utiliza una infraestructura de red de área de almacenamiento (SAN). Además, StorSimple ofrece recuperación de desastres a cualquier centro de datos y ofrece un cifrado AES-256.

Otra de las ventajas de nube de Microsoft Azure es su servicio de aprendizaje Machine Learning. Recopilar grandes volúmenes de datos sólo es útil si se aplica el análisis, pero Microsoft aplica con éxito los avances de aprendizaje automático a su propio negocio y sus análisis avanzados están disponibles para una amplia audiencia. El servicio incluye Machine Learning Studio para la creación y evaluación de modelos de aprendizaje automático. Marketplace permite crear una aplicación sin empezar desde cero.

2.6.3. Comparación Microsoft Azure, google app Engine y Amazon EC2

Para evaluar las posibilidades de las principales plataformas estudiadas se propone el diseño e implementación de una aplicación conforme se detalla a continuación.



Comparativa entre los gigantes del Cloud Computing Empresarial















| | amazon | Microsoft Azure | Google Cloud Platform |
|-----------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|  Nacimiento Servicio | 2006 | 2010 | 2011 |
|  Coste | Pago por hora o fracción. Descuentos en contrataciones de 1 o 3 años. Precio servidor pequeño: 39€ Precio servidor mediano: 134€ | Pago por minutos. No tiene opciones de ahorro. Precio servidor pequeño: 43€ Precio servidor mediano: 180€ | Pago por minutos. Descuentos proporcionales a las horas de consumo Precio servidor pequeño: 35€ Precio servidor mediano: 143€ |
|  Backups | Realiza 3 copias en misma zona geográfica. Posibilidad de replicar copias a otras zonas. | Realiza 3 copias en misma zona geográfica. Posibilidad de replicar copias a otras zonas. | Por defecto realiza las copias en todas las plataformas alrededor del mundo |
|  Disponibilidad mundial | 11 centros de datos. 37 puntos de distribución contenido. | 20 centros de datos. 32 puntos de distribución contenido | 4 centros de datos. 160 puntos de distribución contenido |
|  Marketplace | 2.400 aplicaciones | 707 aplicaciones | 160 aplicaciones |
|  Soporte | Soporte gratuito usando base de conocimientos. Contacto por email en horas de oficina: 49€/mes Contacto en 24x7 con 1 hora de tiempo de respuesta 10% de la facturación (mín. 100€/mes) | Soporte gratuito usando base de conocimientos. Contacto por web en 24x7, 8 horas de tiempo de respuesta. 24,46€/mes Contacto telefónico 24x7, 2 horas de tiempo de respuesta. 256€/mes | Soporte gratuito usando base de conocimientos. 4 horas laborables de tiempo de respuesta. 150€/mes 1 hora de tiempo de respuesta. 9% de la facturación (mín. 400€/mes) |
|  Tipos Servidores | 53 | 25 | 18 |
|  Tipos Discos | Clásicos SSD Se pueden personalizar | Clásicos SSD No se pueden personalizar | Clásicos SSD Se pueden personalizar |
|  Otros Servicios en la nube | Almacenamiento. Bases de Datos. DNS. VDI. | Almacenamiento. Bases de Datos. Suite Ofimática. Correo electrónico. | Almacenamiento. Bases de Datos. Suite Ofimática. Correo electrónico. Registro dominios y DNS. |
|  Seguridad | 20 certificaciones | 25 certificaciones | 6 certificaciones |
|  Estabilidad | 99,95% de disponibilidad mensual. Entre 99,95% y 99% penalización del 10% Por debajo del 99% penalización del 30% | 99,95% de disponibilidad mensual. Entre 99,95% y 99% penalización del 10% Por debajo del 99% penalización del 25% | 99,95% de disponibilidad mensual. Entre 99,95% y 99% penalización del 10% Entre el 99% el 95% penalización del 25% Por debajo del 95% penalización del 50% |
|  Migración servidores | Acepta servidores VMware e Hyper-V | Acepta servidores Hyper-V | Por el momento no soporta migraciones de servidores |

Imagen 5. Comparativa de los gigantes del cloud computing ³

Fuente: <http://www.apser.es/blog/2015/11/25/comparativa-amazon-web-services-vs-microsoft-azure-vs-google-cloud-platform/>

3. Diseño e implementación de aplicaciones en la nube

Después de una breve explicación de lo que significa la nube y de las aplicaciones que podemos encontrar actualmente vamos a dar paso ahora a la aplicación que hemos creado nosotros en diferentes plataformas y de las pruebas realizadas en cada una de ellas para compararlas y ver las diferencias entre Windows Azure y Google App Engine.

El motivo por el cual he escogido Windows Azure y GAE es porque son las plataformas más conocidas actualmente y además hay muchas más facilidades para el usuario crear una aplicación en dichas plataformas como primera forma de contacto con el mundo del cloud computing.

3.1. Aplicación carrito compra para tienda de pintura

Para evaluar las posibilidades de las principales plataformas estudiadas se propone el diseño e implementación de una aplicación conforme se detalla a continuación.

Para realizar nuestras pruebas, vamos a utilizar la típica aplicación de una tienda de carrito la cual un usuario seleccionara el o los productos que aparecerán en pantalla y tras finalizar la selección el usuario podrá enviarse la factura por correo o solo finalizar el pedido, además enlazaremos dicha aplicación con el acceso a la plataforma Google Calendar

Para todo ello hemos utilizado una base de datos SQL server y otra MySQL dependiendo de la plataforma utilizada.



3.1.1. Diagrama UML de nuestra aplicación

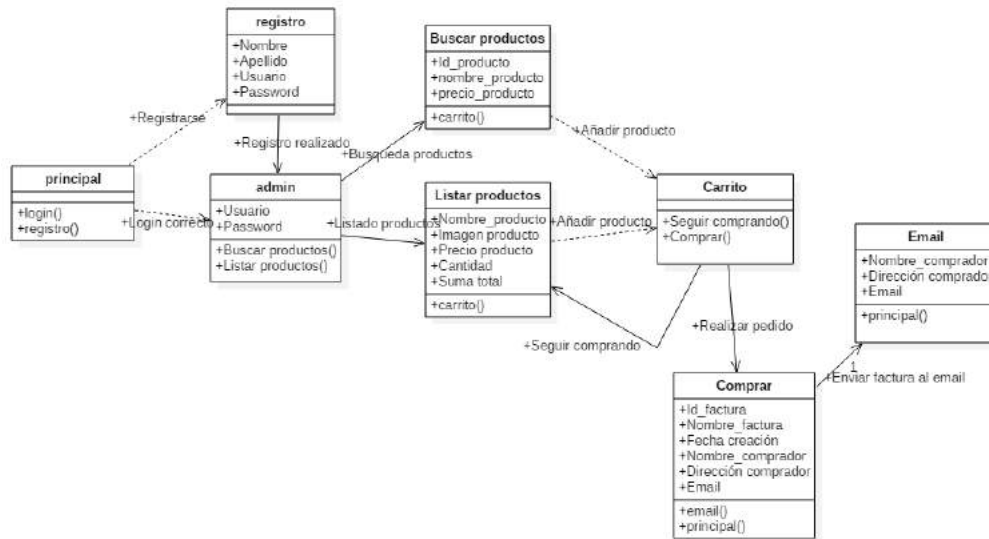


Imagen 6. Diagrama UML app servicolor

En el diagrama podemos observar las diferentes clases que tendrán nuestro proyecto y las acciones que vamos a realizar dependiendo de lo que seleccione el usuario. Añadir que el diagrama lo hemos realizado con el programa StarUML.

3.2. Windows Azure

Vamos ahora a explicar paso a paso como crear una aplicación en Windows Azure utilizando ASP.NET MVC.

El motivo por el cual he utilizado ASP.NET MVC es para empezar que es un lenguaje que no he visto en ninguna asignatura y suponía un reto para mí, además tras ver mucha documentación al respecto vi conveniente utilizar este lenguaje porque es bastante sencillo y dinámico para y separa la lógica de presentación, la lógica de negocio y la lógica de acceso a datos, fácilmente.

3.2.1. ASP.NET

Es un framework para aplicaciones web desarrollado y comercializado por Microsoft. Es usado por programadores y diseñadores para construir sitios web dinámicos, aplicaciones web y servicios web XML. Es la tecnología sucesora de la tecnología Active Server Pages (ASP). ASP.NET está construido sobre el Common Language Runtime, permitiendo a los programadores escribir código ASP.NET usando cualquier lenguaje admitido por el .NET Framework.

3.2.1.1. Características

Las páginas de ASP.NET, conocidas oficialmente como formularios web, son el principal medio de construcción para el desarrollo de aplicaciones web. Los formularios web están contenidos en archivos con una extensión ASPX, dichos archivos típicamente contienen etiquetas HTML o XHTML estático, y también etiquetas definiendo Controles Web que se procesan del lado del servidor y Controles de Usuario donde los desarrolladores colocan todo el código estático y dinámico requerido por la página web.

ASP.NET no sólo funciona sobre el servidor de Microsoft IIS, también lo hace sobre Apache.

3.2.1.2. Modelo Code-behind

Los nombres de los archivos code-behind están basados en el nombre del archivo ASPX tales como MiPagina.aspx.cs o MiPagina.aspx.vb (se suele utilizar en Microsoft Visual Studio y otros entornos de desarrollo). Cuando se usa este estilo de programación, el desarrollador escribe el código correspondiente a diferentes eventos, como la carga de la página, o el clic en un control, en vez de un recorrido lineal a través del documento.



3.2.1.3. Controles de usuario

ASP.NET permite la creación de componentes reutilizables a través de la creación de Controles de Usuario (User Controls). Un control de usuario sigue la misma estructura que un formulario web, excepto que los controles derivan de la clase `System.Web.UI.UserControl`, y son almacenados en archivos ASCX. Como los archivos ASPX, un ASCX contiene etiquetas HTML o XHTML, además de etiquetas para definir controles web y otros controles de usuario. También pueden usar el modelo code-behind.

3.2.1.4. Administración del estado

Las aplicaciones ASP.NET son alojadas en un servidor web y se tiene acceso a ellas mediante el protocolo sin estado HTTP, que no guarda ninguna información sobre conexiones anteriores. Por lo tanto, si la aplicación requiere interacción entre conexiones, tiene que implementar su propia administración del estado. ASP.NET proporciona varias maneras de administrar el estado de las aplicaciones ASP.NET.

- **Estado de aplicación**

El estado de la aplicación (Application state) es una colección de variables definidas por el usuario que son compartidas por todas las invocaciones de una aplicación ASP.NET. Estas son establecidas e inicializadas cuando el evento `Application_OnStart` se dispara en la carga de la primera instancia de las aplicaciones y están disponible hasta que la última instancia termina. Las variables de estado o variables de sesión de la aplicación son identificadas por nombres.

- **Estado de la sesión**

El estado de la sesión es una colección de variables definidas por el usuario, las cuales persisten durante la sesión de un usuario. Estas variables son únicas para diferentes instancias de una sesión de usuario, y son accedidas usando la colección `Session`.

Las variables de sesión pueden ser preparadas para ser automáticamente destruidas después de un determinado tiempo de inactividad, incluso si la sesión no ha terminado. Del lado del cliente, una sesión de usuario es identificada por una cookie o codificando el ID de la sesión en la misma URL.

ASP.NET proporciona tres modos de persistencia para variables de sesión:

- **InProc**

Las variables de sesión son mantenidas dentro del proceso. Sin embargo, en este modo, las variables son destruidas cuando el proceso ASP.NET es reciclado o terminado.

- **StateServer**

En este modo, ASP.NET ejecuta un servicio de Windows separado que mantiene las variables de estado. Como esta administración de estado ocurre fuera del proceso ASP.NET, tiene un impacto negativo en el rendimiento, pero permite a múltiples instancias de ASP.NET compartir el mismo estado del servidor, permitiendo que una aplicación ASP.NET pueda tener su carga balanceada y escalada en múltiples servidores. También, como el servicio de administración del estado se ejecuta independiente de ASP.NET, las variables pueden persistir a través de las finalizaciones del proceso ASP.NET.

- **SqlServer**

En este modo, las variables de estado son almacenadas en un servidor de base de datos, accesible usando SQL. Las variables de sesión pueden persistir a través de finalizaciones de procesos también en este modo.

- **Estado de la vista**

El estado de la vista (View state) es el mecanismo de administración de estado a nivel de página, que es utilizado por las páginas HTML generadas por las aplicaciones ASP.NET para mantener el estado de los controles de los formularios web y los widgets.



3.2.1.5. Motor de plantillas

ASP.NET 2.0 presentó el concepto de página maestra (Master Page), que permite el desarrollo de páginas basado en plantillas web. Una aplicación web puede tener una o más páginas maestras, las cuales pueden ser anidadas. Las plantillas maestras contienen controles contenedores, llamados ContentPlaceHolders para indicar dónde irá el contenido dinámico, además de HTML y JavaScript que será compartido a través de las páginas hijas.

Las páginas hijas también usan esos controles ContentPlaceholder, que deben ser relacionados con el ContentPlaceholder de la página maestra que contiene a esta página hija. El resto de la página está definido por las partes compartidas de la página maestra.

Razor es el motor de plantilla el cual facilita a las páginas ASP.NET MVC.

3.2.1.6. Modelos de programación en ASP.NET

ASP.NET soporta tres modelos de programación: ASP.NET formularios web, ASP.NET MVC y ASP.NET Páginas web. Aunque los tres modelos de programación se ejecutan sobre la misma base de ASP.NET, cada uno de ellos estructura la aplicación de maneras completamente distintas, promueve metodologías de desarrollo diferentes y se adapta a perfiles de desarrolladores distintos.

- **ASP.NET** proporciona un gran nivel de abstracción con un modelo de programación familiar basado en eventos y controles que favorece la productividad mediante la programación declarativa reduciendo la cantidad de código necesaria para implementar una determinada funcionalidad.
- **ASP.NET** proporciona un modelo de programación basado en el popular patrón de arquitectura MVC. Entre sus principales características destacan su completa integración con pruebas unitarias y su separación más clara entre la lógica de presentación, la lógica de negocio y la lógica de acceso a datos.

- **ASP.NET páginas web** fue creado como respuesta a una creciente demanda de desarrolladores web sin experiencia previa con ASP.NET, cuya iniciación en ASP.NET formularios web o MVC les suponía una inversión inicial de tiempo demasiado grande. Páginas web proporciona un modelo de programación más simple y rápida de aprender, sin renunciar a toda la funcionalidad y flexibilidad de ASP.NET.

3.2.1.7. Uso actual del lenguaje

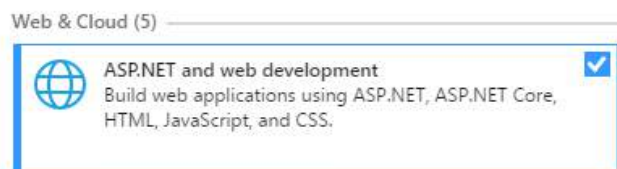
En la actualidad una aplicación .NET puede ejecutarse de dos formas distintas:

- **Aplicaciones cliente/servidor:** Estas aplicaciones están típicamente en formato de ejecutables compilados. Estos pueden integrar toda la riqueza de una interfaz de usuario, tal es el caso de las aplicaciones de desempeño y productividad, pero no se reúne la lógica de negocio como un recurso que se pueda reutilizar. Además suelen ser menos gestionables y escalables que las demás aplicaciones.
- **Aplicaciones que utilizan el navegador:** Cuentan con una interfaz de web rica y muy útil. La interfaz gráfica integra varias tecnologías, las cuales son el HTML, XHTML, scripting, etc., recordando que el navegador que se esté utilizando soporte estas tecnologías.

3.2.2. Aplicación Servicolor

3.2.2.1. Pasos previos

Para empezar necesitaremos tener instalado visual studio 2015 o superior con las siguientes características: después de descargar visual studio seleccionaremos las siguientes opciones.



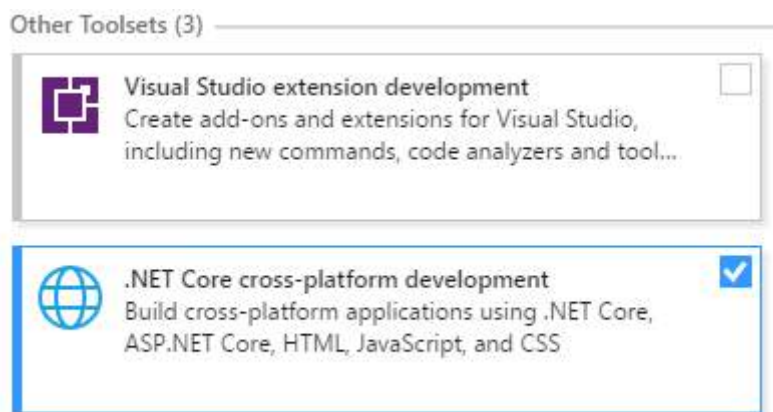


Imagen 7 y 8. Instalación Visual Studio

Seleccionaremos ASP.NET para la instalación debido a que nuestra aplicación está basada en éste lenguaje.

En otras opciones seleccionaremos el paquete .NET para que nuestra aplicación acepte archivos HTML, JavaScript y CSS.

Tras la instalación que puede demorarse unos minutos iniciaremos sesión con tu cuenta de correo. Además de tener instalado visual studio también deberemos tener instalado Windows Azure y una cuenta en Microsoft Azure, la cual nosotros hemos obtenido gracias a la universidad una cuenta de prueba para poder realizar las pruebas para el proyecto.

Seguidamente después de tener los programas principales tendremos que instalar los paquetes necesarios para desarrollar nuestra aplicación: en mi caso instales todos los paquetes para poder programar en ASP.NET (SDK de Azure para Visual Studio) y con bases de datos en el siguiente enlace (<https://msdn.microsoft.com/es-es/library/mt204009.aspx>).

3.2.2.2. Detalles de implementación

1. Abrimos Visual Studio 2017.
2. Archivo > Nuevo > Proyecto
3. Visual C# > Web > Aplicación web ASP.NET
4. Usar .NET Framework 4.6.1
5. Desactive la casilla Application Insights (Agregue Application ...)
6. Ponga el nombre de aplicación que quiera por ejemplo: Proyecto en nuestro caso. Luego ya podemos darle aceptar

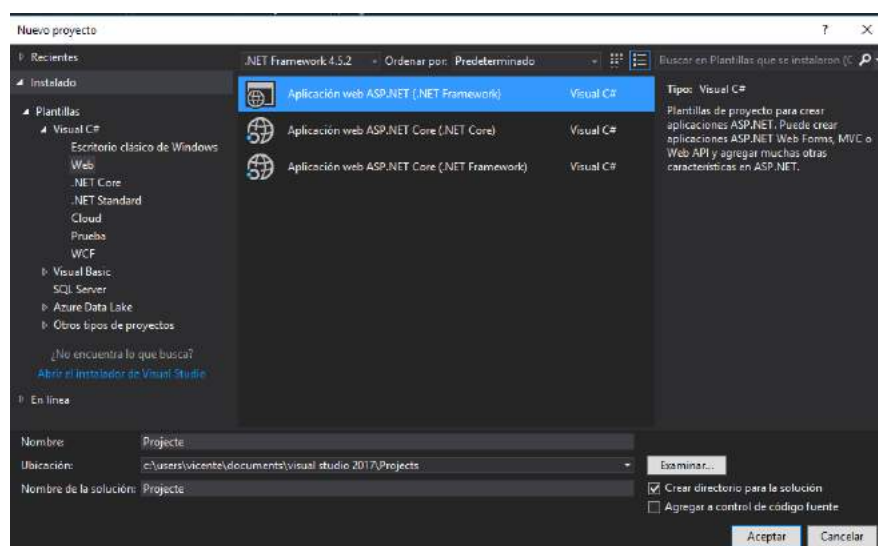


Imagen 9. Creación nuevo proyecto

7. Seleccione MVC

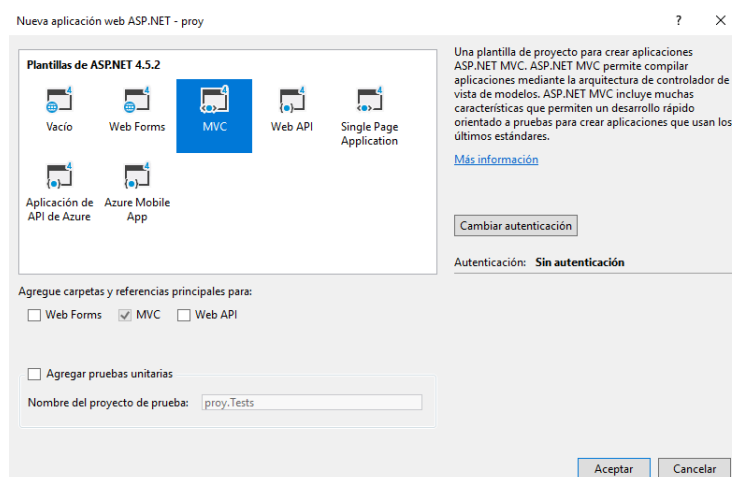


Imagen 10. Opción MVC

8. Debe aparecerle lo que vemos en la Imagen 11 y el proyecto creado en la parte izquierda



Imagen 11. Creación nuevo proyecto.

✓ **Significado de cada archivo creado:**

- Project.json -> Determina que el proyecto es en entorno .NET de ejecución (DNX). en el project.json tenemos toda la información necesaria para funcionar DNX y el paquete de su proyecto.
- global.json -> Visual studio utiliza este archivo para configurar el proyecto.
- appsettings.json -> Este archivo permite incluir información adicional del proyecto, tales como los valores de cadena de conexión.
- Startup.cs -> Es el inicio de las clases de la aplicación que proporciona el punto de entrada. Este inicio de la clase se debe definir con un Configurar método, y puede también opcionalmente definir un ConfigureServices, el cual se llama cuando se inicia la aplicación.
- Index.cshtml -> Contiene el código HTML de la página por defecto de la vista.
- _Layout.cshtml -> Vista del HTML común para varias páginas de la aplicación web.
- HomeController.cs -> Contiene las clases que manejan las solicitudes entrantes de navegación, recuperar los datos del modelo, y luego se especifican plantillas de vista que devuelven una respuesta al navegador.

✓ **Significado de MVC**

MVC significa Modelo-Vista-Controlador el cual es un patrón para el desarrollo de aplicaciones que son buena arquitectura, comprobable y fácil de mantener. Las aplicaciones basadas en MVC contienen:

- Modelos -> Las clases que representan los datos de la solicitud y que la lógica de validación uso para hacer cumplir las reglas de negocio para esos datos. Por ejemplo una base de datos.
- Vistas -> archivos de plantilla que la aplicación utiliza para generar dinámicamente las respuestas HTML.
- Controladores: Las clases que manejan las solicitudes entrantes de navegación, recuperar los datos del modelo, y luego especificar plantillas de vista que devuelven una respuesta al navegador.

✓ **Entity Framework (EF)**

Es un marco de mapeo objeto-relacional (ORM), el cual permite trabajar con datos relacionales como objetos, y esto elimina la mayor parte del código de acceso a datos que desea escribir. El uso de EF, puede emitir consultas utilizando LINQ (proporciona las pautas para consultar y actualizar datos), A continuación, recuperar y manipular datos como objetos inflexible. El uso de EF permite centrarse en crear el resto de la aplicación, en lugar de centrarse en los aspectos fundamentales de acceso a datos.

✓ **Como crear un modelo de datos y el controlador**

EF soporta el paradigma de desarrollo denominado Código primer lugar. El primer código permite definir sus modelos de datos utilizando clases. Una clase es una construcción que le permite crear sus propios tipos personalizados mediante la agrupación de las variables de otros tipos, métodos y eventos. También puede asignar clases a una base de datos existente o utilizarlos para generar una base de datos.



✓ **Como crear clases de entidad**

Las clases se crean para definir el esquema de los datos se llaman clases de entidad. Las clases de entidad son como definiciones de las tablas de una base de datos. Cada propiedad de la clase especifica una columna en la tabla de la base de datos. Estas clases proporcionan una interfaz ligera, objeto-relacional entre el código orientado a objetos y la estructura de la tabla relacional de la base de datos. Para nuestro proyecto utilizaremos una base de datos SQL ya creada anteriormente.

9. Seleccionamos models (en el menú de explorador de soluciones) -> Agregar -> Nuevo elemento.
10. A continuación Seleccionaremos en la parte izquierda Datos y de las opciones que nos aparecen ADO.NET Entity Data Model. Pondremos el nombre que queramos para nuestro proyecto (nuestro caso servicolorbd) y aceptar.
11. Una vez le hemos dado a aceptar veremos 4 tipos de opciones:
 - EF Designer desde base de datos como ya explica más abajo sería crear un modelo desde una base de datos existente.
 - La siguiente opción que encontramos Modelo vacío de EF Designer creará un modelo vacío en EF Designer como punto de partida para diseñar visualmente el modelo.
 - La tercera opción nos permitirá crear un modelo como punto de partida mediante código
 - Por último la opción Code First desde BD creará un modelo de Code First desde una base de datos existente.

En nuestro caso como ya tenemos creada la base de datos a partir de SQL 2014 elegimos la primera opción.

Solicitará que nos conectemos a una base de datos. Seleccionaremos → Nueva Conexión y nos conectaremos a nuestro servidores de MySQL en mi caso el nombre del servidor es un punto (.).

Luego si tenemos usuario y contraseña nos autenticaremos con ello y para finalizar seleccionaremos nuestra base de datos servicolorbd.

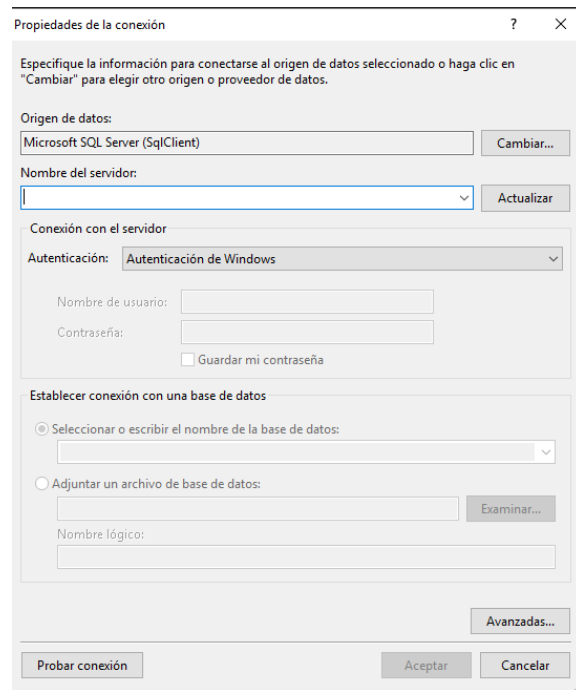


Imagen 12. Conectar a nuestra base de datos servicolorbd.

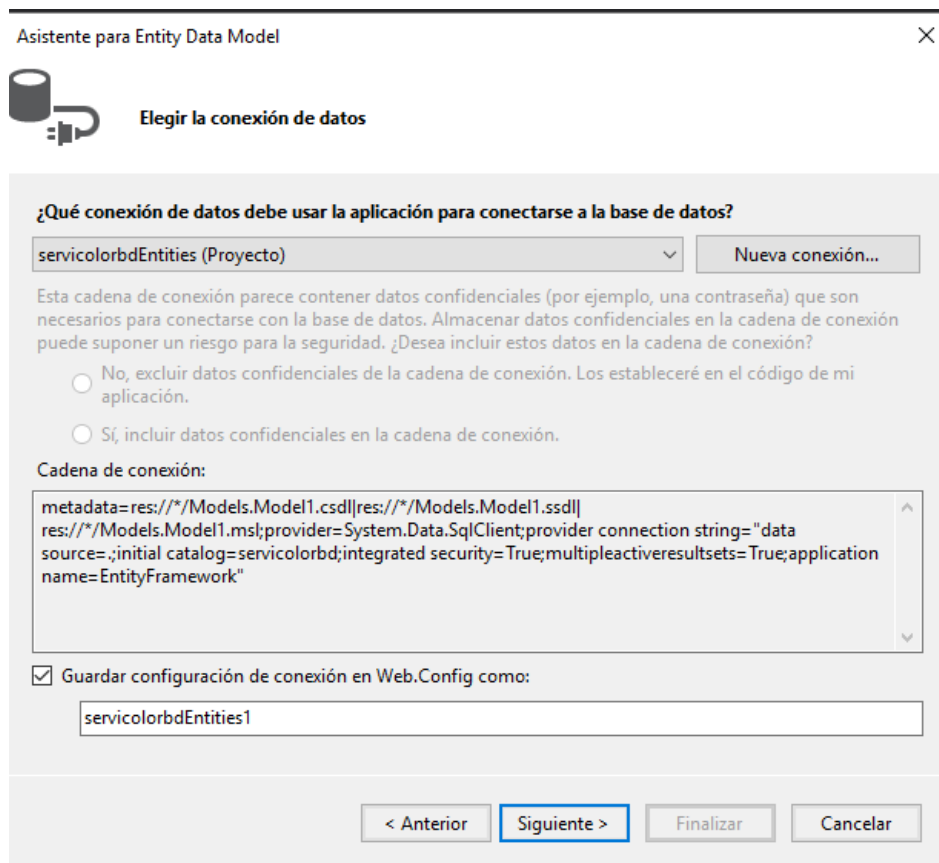


Imagen 13. Conexión correcta con servicolorbd.

Una vez le damos siguiente el proyecto se quedará como vemos en Imagen 14.

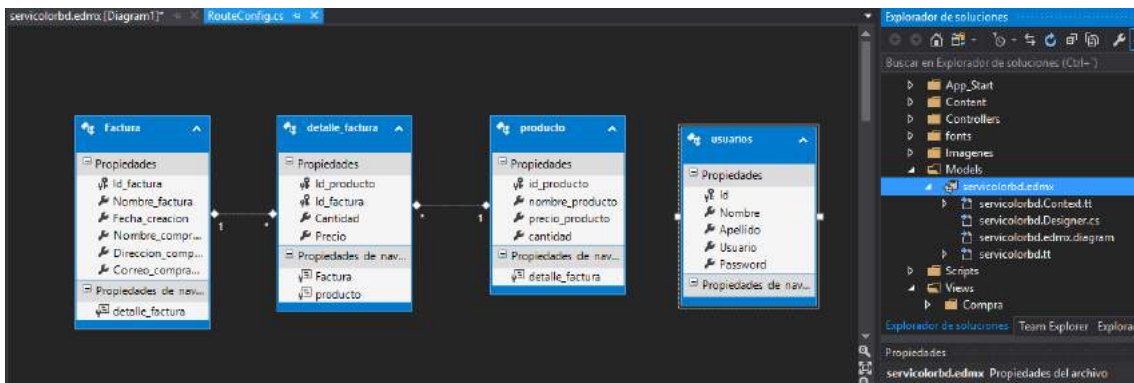


Imagen 14. Base de datos servicolorbd

Como podemos observar se ha creado el modelo a partir de una base de datos existente y cada elemento de dicha base de datos tiene sus clases propias con los consultores y constructores correspondientes.

Analizando los elementos que encontramos, vemos que tenemos 4 elementos los cuales uno es el producto donde se mostrara al usuario con sus características. Después vemos la Factura que se generara a partir de los productos seleccionados y luego los detalles de la Factura que será lo que une los productos con la Factura.

Finalmente tenemos una tabla de usuarios los cuales nos servirá para autenticarnos.

Tras analizar la BD iremos a la parte izquierda de visual Studio para ver las conexiones que tenemos a la BD.

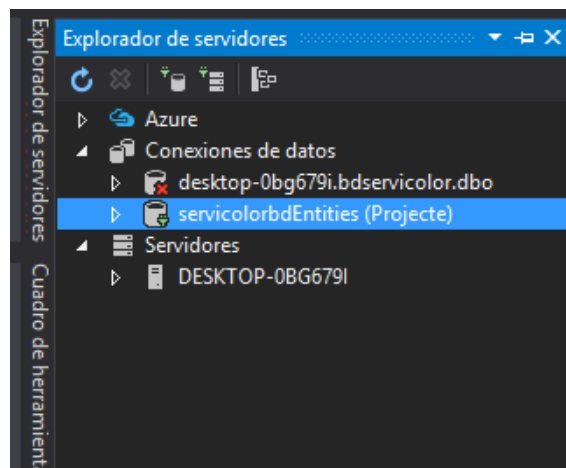


Imagen 15 Conectar con base de datos servicolorbd

En la Imagen 15 podemos ver que estamos conectados correctamente a la base de datos servicolorbd que es el nombre que le hemos puesto nosotros en SQL 2014

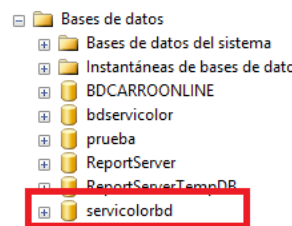


Imagen 16 servicolorbd en SQL Server 2014.

```
namespace Project.Models
{
    using System;
    using System.Collections.Generic;

    [Serializable]
    public partial class producto
    {
        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2214:DoNotCallOverridableMethodsInConstructors")]
        public producto()
        {
            this.detalle_factura = new HashSet<detalle_factura>();
        }

        [Serializable]
        public int id_producto { get; set; }
        [Serializable]
        public string nombre_producto { get; set; }
        [Serializable]
        public double precio_producto { get; set; }
        [Serializable]
        public double cantidad { get; set; }
        [Serializable]
        public string categoria { get; set; }

        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]
        public virtual ICollection<detalle_factura> detalle_factura { get; set; }
    }
}
```

Imagen 17. Clase producto creada automáticamente a partir de EF Designer desde base de datos.

✓ Añadir Controladores y vistas

Los controladores ahorran tiempo y esfuerzo de codificación mediante la generación automática del punto de partida para el crear, leer, actualizar y eliminar (CLAE) de la aplicación las operaciones.

En nuestro proyecto para añadir un controlador damos clic en la carpeta **Controladores** de la carpeta explorador de soluciones y seguidamente le damos a **Agregar -> Controlador** y seleccionaremos la opción Controlador de MVC 5 con acciones de lectura y escritura, estaría la opción de sería Controlador de MVC 5 con vistas que usa acciones de lectura y escritura esta opción crearía todas las vistas genéricas automáticamente de crear, eliminar, editar, listar pero para nuestro proyecto utilizaremos la opción primera debido a que vamos a modificar ligeramente el código que nos autogenera.



12. A continuación en la siguiente ventana pondremos el nombre del controlador en nuestro caso productoController (intentar siempre dejar esta nomenclatura xxxController para saber siempre que es el controlador y no equivocarnos).

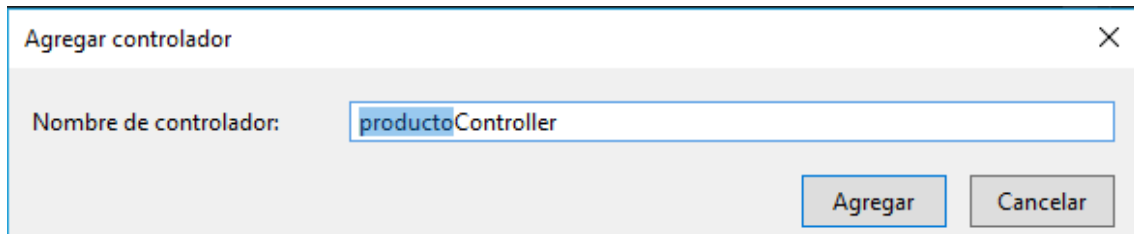


Imagen 18 Controlador productoController

Tras crear el controlador veremos muchas clases creadas automáticamente pero que ninguna tiene código definido dentro de ellas.

En nuestro caso como la página principal va a ser que el usuario vea la lista de productos, implementaremos esta función en la acción resultante índice.

13. Antes de modificar la acción resultante de índice crearemos una variable que en nuestro caso se llamará de en la cual nos conectaremos a nuestra base de datos.

El código que añadiremos a la clase productocontroller será el siguiente:

```
using Projecte.Models;
```

```
...
```

```
private servicolorbdEntities de = new servicolorbdEntities();
```

```
// Index
```

```
public ActionResult Index()
```

```
{
```

```
    ViewBag.listaProductos = de.producto.ToList();
```

```
    return View();
```

```
}
```

Con estos cambios ya tenemos todo preparado para poder ver nuestra lista de productos.

14. A continuación añadiremos la vista de Índice para poder ver estos resultados.

15. Seleccionaremos con el botón derecho en Índice()-> Agregar vista

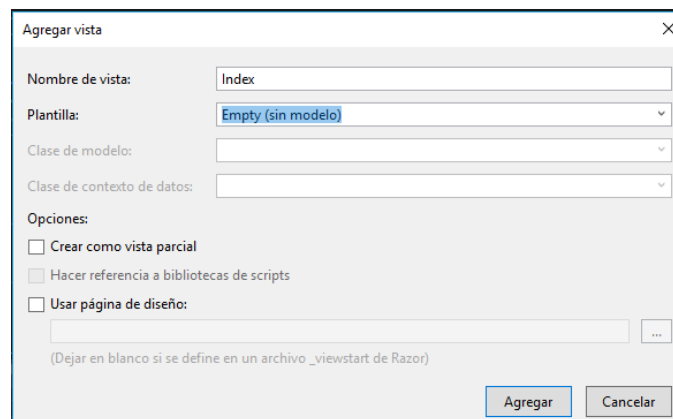


Imagen 18. Vista Índice

La crearemos sin modelo debido a que queremos hacer unas modificaciones en el código y es más sencillo hacerlo desde cero pero tenemos muchas opciones para creación de una vista.

16. A continuación veremos que el código que tenemos está bastante vacío pero nosotros lo vamos a rellenar con el siguiente código y podremos ver la lista de nuestros productos.

//utiliza la clase producto de la carpeta models

```
@model IEnumerable<Projecte.Models.producto>
```

```
@{
```

```
    Layout = null;
```

```
}
```

```
@using Projecte.Models
```

```
<!DOCTYPE html>
```

```

<html>
<head>
  <meta name="viewport" content="width=device-width" />
  <title>Index</title>
</head>
<body>
  @*<p>
    @Html.ActionLink("Create New", "Create")
  </p>*@
<table>
  <tr>
    <td>
      
    </td>
    <td>
      <label style="font-size:30px; font-family:'Comic Sans MS';
              font-weight:bold;color:blue;">SERVICOLOR</label>
    </td>
  </tr>
</table>
<table cellpadding="2" cellspacing="2" border="1">
  <tr>
    <th>
      @Html.DisplayNameFor(model => model.id_producto)
    </th>
    <th>
      @Html.DisplayNameFor(model => model.nombre_producto)
    </th>
    <th>
      @Html.DisplayNameFor(model => model.Imagen)
    </th>
    <th>
      @Html.DisplayNameFor(model => model.precio_producto)
    </th>
    <th>

```

```

        @Html.DisplayNameFor(model => model.cantidad)
    </th>
    <th>
        Añadir al carrito
    </th>

</tr>

@foreach (producto item in ViewBag.listaProductos)
{
    <tr>

        <td>
            @Html.DisplayFor(modelItem => item.id_producto)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.nombre_producto)
        </td>

```

//Aquí nos mostrara las diferentes Imagenes que tenemos de los productos en nuestra base de datos, las Imagenes las hemos guardado en una carpeta llamada Imagenes.

```

        <td align="center">
            @if (item.id_producto == 1){  }
            @if (item.id_producto == 2){  }
            @if (item.id_producto == 3){  }
            @if (item.id_producto == 4){  }
            @if (item.id_producto == 5){  }
            @if (item.id_producto == 6){  }
            @if (item.id_producto == 7){  }

```

```

        @if (item.id_producto == 8){  }
        @if (item.id_producto == 9){  }
        @if (item.id_producto == 10){  }
        @if (item.id_producto == 11){  }
        @if (item.id_producto == 12){  }
        @if (item.id_producto == 13){  }
        @if (item.id_producto == 14){  }

```

```

</td>
<td>
    @Html.DisplayFor(modelItem => item.precio_producto)
</td>
<td>
    @Html.DisplayFor(modelItem => item.cantidad)
</td>
<td align="center">

```

// Esto en principio es para más adelante pero lo utilizaremos para re direccionarlos a la hora de seleccionar un elemento.

```

    @Html.ActionLink("Añadir", "Añadir", "Compra", new { id = item.id_producto },
null) @*|
        @Html.ActionLink("Details", "Details", new { /* id=item.PrimaryKey */ }) |
        @Html.ActionLink("Delete", "Delete", new { /* id=item.PrimaryKey */ })*@
    </td>
</tr>
}
</table>
</body>
</html>

```


Tras realizar estas acciones podremos probar la aplicación y veremos cómo nos muestra una lista de los productos que tenemos en venta, el precio de cada producto y las unidades que disponemos de cada una.



| id_producto | nombre_producto | imagen | precio_producto | cantidad | Añadir al carrito |
|-------------|------------------|-------------------------------------------------------------------------------------|-----------------|----------|------------------------|
| 1 | Pintura Blanca |  | 29.95 | 500 | Añadir |
| 2 | Pintura Negra |  | 19.95 | 100 | Añadir |
| 3 | Pintura Azul |  | 19.95 | 100 | Añadir |
| 4 | Pintura Roja |  | 19.95 | 100 | Añadir |
| 5 | Pintura Amarilla |  | 19.95 | 100 | Añadir |

Imagen 19 Lista productos de nuestra base de datos servicolorbd.

Como podemos observar ya tenemos la lista de nuestros productos, ahora vamos a pasar a implementar la opción de añadir productos a nuestro carrito de compra.

Revisamos el final de la vista de Index:

```
@Html.ActionLink("Añadir", "Añadir", "Compra", new { id =
item.id_producto }, null)
```

Este código sirve para:

El primer añadir es lo que nos muestra por pantalla, el segundo añadir es a la clase que nos dirigiremos si seleccionamos la opción añadir, la opción compra es el controlador donde debe estar implementada la clase añadir y por último le pasaremos la id del producto que hemos seleccionado para seguir con nuestras pruebas.



17. Vamos a la carpeta controllers y añadiremos un nuevo controller.
18. El nombre de éste controlador será CompraController la cual la utilizaremos para guardar los datos del carrito. El tipo de controlador será el mismo que hemos creado para productoController.
19. Tras crear el controlador de compra tendremos que añadir la función añadir que es a la que hacemos referencia en la vista Index de producto.
 - El código de esta clase será el siguiente:

```
public ActionResult añadir(int id)
{
    if (Session["carrito"] == null)
    {
        List<Item> carrito = new List<Item>();
        carrito.Add(new Item(de.producto.Find(id),1));
        Session["carrito"] = carrito;
    }
    else
    {
        List<Item> carrito = (List<Item>)Session["carrito"];
        int index = existe(id);
        if (index == -1)
            carrito.Add(new Item(de.producto.Find(id), 1));
        else
            carrito[index].Cantidad++;
        Session["carrito"] = carrito;
    }
    return View("Carrito");
}
```

Hemos creado una sesión nueva la cual llamaremos carrito donde guardaremos todos los productos seleccionados por el cliente. A continuación creamos una lista de Item carrito donde vamos añadiendo dichos productos si el producto no se encuentra en la sesión del cliente. Por el contrario si dicho producto ya se encuentra en la sesión del cliente lo que haremos será añadirle 1 en la cantidad que tenemos de ese producto.

Una vez compilamos dicho código nos dirá que no podemos crear una List<Item> porque no está implementado.

20. En este paso crearemos la clase Item donde la utilizaremos para ir añadiendo productos con el mismo id a nuestra sesión.

Primero vamos a la carpeta controllers y seleccionaremos agregar -> clase la cual llamaremos Item.cs.

El código de ésta clase será el siguiente:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using Modelo;
// Para poder utilizar la clase producto
using Projecte.Models;

namespace Projecte.Controllers
{
    public class Item
    {
        private producto pr = new producto();
        private double cantidad;

        public producto Pr
        {
            get { return pr; }
            set { pr = value; }
        }
    }
}
```



```
public double Cantidad
{
    get { return cantidad; }
    set { cantidad = value; }
}

public Item() {}

public Item(producto product, double cantidad)
{

    this.pr = product;
    this.cantidad = cantidad;
}

}

}
```

Tras ésta implementación nos quedara implementar la clase existe que utilizamos en la clase añadir, la cual nos indicara si un producto ya lo tenemos en nuestra sesión o no. Tras realizar dicha comprobación nos añadirá el producto si existe sumando uno a la cantidad de dicho producto.

21. La implementación de la clase existe será la siguiente:

```
private int existe(int id)
{
    List<Item> carrito = (List<Item>)Session["carrito"];
    for (int i = 0; i < carrito.Count; i++)
        if (carrito[i].Pr.id_producto == id)
            return i;
    return -1;
}
```

Tras la implementación del controlador, nos quedara implementar las vistas.

22. En la última línea del código de la clase añadir retorna la vista "Carrito", por lo tanto seleccionaremos dicha vista y añadiremos la vista vacía.

23. En esta vista lo que veremos será la lista de los productos que hemos seleccionado para nuestro carrito y la suma del total de lo que nos va a costar la compra. El código que debemos implementar será el siguiente:

```
@{
    Layout = null;
}
@*Referencia a la clase producto y a los controladores para poder utilizar sus clases y atributos.*@
@using Projecte.Controllers
@model Modelo.producto

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Carrito</title>
</head>
<body>
    <div>
        <table cellpadding="2" cellspacing="2" border="1">
            <tr>
                <td>Opciones</td>
                <td>Id</td>
                <td>Nombre Producto</td>
                <td>Precio</td>
                <td>Cantidad</td>
                <td>Sub_total</td>
            </tr>
            @*Creamos una variable double para ir guardando en ella el total de lo que hemos
            seleccionado en el carrito*@
```



```

@{
    double s = 0;
    @*s = s + item.Pr.precio_producto * item.Cantidad;*@
}

@foreach (Item item in (List<Item>)Session["Carrito"])
{
    double sumatotal = Convert.ToDouble(item.Cantidad) *
Convert.ToDouble(item.Pr.precio_producto);
    s += sumatotal;
    <tr>

```

@*Añadimos una opción para eliminar un elemento del carrito la cual la implementaremos en la clase compracontroller* @

```

<td>@Html.ActionLink("Eliminar","Eliminar","Compra",new {id =
item.Pr.id_producto },null)</td>
<td>@item.Pr.id_producto</td>
<td>@item.Pr.nombre_producto</td>
<td>@item.Pr.precio_producto</td>
<td>@item.Cantidad</td>
<td>@(item.Pr.precio_producto * item.Cantidad)</td>

</tr>
}
<tr>
<td align="right" colspan="5">Suma Total</td>
<td>@s</td>
</tr>
</table>
<br />
<br />

```

@*Si seleccionamos esta opción iremos a la página principal de nuestro programa* @

```

@Html.ActionLink("Continuar Comprando","Index","Producto")
<br />

```

@*Esta opción la implementaremos más adelante y nos servirá para guardar los detalles de la compra en nuestra base de datos* @

```

<br />
@Html.ActionLink("Finalizar Compra", "Comprar", "Compra")
<br />
@*Con esta opción enviaremos la factura por correo al correo que nos indique el
cliente*@

<br />
@Html.ActionLink("Factura al correo", "Correo", "Compra")
<br />
</div>
</body>
</html>

```

Como nota adicional en la clase RouteConfig.cd dentro de la carpeta app_Start cambiaremos el código inicial que nos aparece por el siguiente para tener la vista de index de home como la principal:

```

public static void RegisterRoutes(RouteCollection routes)
{
    routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

    routes.MapRoute(
        name: "Default",
        url: "{controller}/{action}/{id}",
        defaults: new { controller = "Home", action = "Index", id =
UrlParameter.Optional }
    );
}

```

Llegado a este punto cuando seleccionamos añadir un producto al carrito deberá aparecer lo siguiente:

| Opciones | Id | Nombre Producto | Precio | Cantidad | Sub_total |
|--------------------------|----|-----------------|--------|----------|-----------|
| Eliminar | 2 | Pintura Negra | 19,95 | 1 | 19,95 |
| Eliminar | 1 | Pintura Blanca | 29,95 | 1 | 29,95 |
| Eliminar | 4 | Pintura Roja | 19,95 | 1 | 19,95 |
| Eliminar | 3 | Pintura Azul | 19,95 | 1 | 19,95 |
| Suma Total | | | | | 89,8 |

[Continuar Comprando](#)

[Finalizar Compra](#)

[Factura al correo](#)

Imagen 20. Vista Carrito.

24. Ahora vamos a implementar la clase eliminar compra. Esta clase se encargara de eliminar un producto de nuestra sesión del carrito sin que elimine nada más del carrito.

El código de dicha clase será el siguiente:

```
public ActionResult eliminar(int id)
{
    int index = existe(id);
    List<Item> carrito = (List<Item>)Session["carrito"];

    if (carrito[index].Cantidad > 1)
        carrito[index].Cantidad--;
    else carrito.RemoveAt(index);
    Session["carrito"] = carrito;
    return View("Carrito");
}
```

En esta clase busca si existe el id del producto seleccionado y si es el caso elimina el elemento de la sesión carrito.

25. A continuación vamos a implementar la clase finalizar compra en la cual se guardaran los datos de la compra en nuestra base de datos:

```
public ActionResult Comprar()
{
    return View("Comprar");
}
public ActionResult GuardarPedido(FormCollection fc)
{
    List<Item> carrito = (List<Item>)Session["Carrito"];

    //Guardar factura
    Factura factura = new Factura();
    factura.Fecha_creacion = DateTime.Now;
    factura.Nombre_comprador = fc["Nombre_comprador"];
    factura.Direccion_comprador = fc["Direccion_comprador"];
    factura.Correo_comprador = fc["Correo_comprador"];
    factura.Nombre_factura = "Nueva Factura";
    de.Factura.Add(factura);
    de.SaveChanges();

    //Guardar detalles factura
    foreach (Item item in carrito)
    {
        detalle_factura detalles = new detalle_factura();
        detalles.Id_factura = factura.Id_factura;
        detalles.Cantidad = item.Cantidad;
        detalles.Precio = item.Pr.precio_producto;
        detalles.Id_producto = item.Pr.id_producto;
        de.detalle_factura.Add(detalles);
        de.SaveChanges();
    }

    //Limpia todos los objetos del carrito
    Session.Remove("Carrito");
    return View("Gracias");
}
```



```
}
```

Primero retornara la vista comprar y luego cuando dicha vista nos recoge los datos que queremos retornará a la clase Guardar Pedido la cual se encarga de guardar los datos a nuestra base de datos. Y para finalizar nos retornará a la vista Gracias y eliminará la sesión Carrito.

26. En éste paso vamos a implementar la vista Comprar la cual la crearemos como anteriormente. El código que debemos implementar en dicha clase será el siguiente:

```
@{
    Layout = null;
}

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Comprar</title>
</head>
<body>
    <fieldset>
        <legend> Información comprador</legend>
        @using (Html.BeginForm("GuardarPedido", "Compra", FormMethod.Post)) {

            <table cellpadding="2" cellspacing="2">
                <tr>
                    <td>Nombre</td>
                    <td><input type="text" name="Nombre_comprador"></td>
                </tr>
                <tr>
                    <td>Dirección</td>
                    <td><input type="text" name="Direccion_comprador"></td>
                </tr>
            </table>
        }
    }
}
```

```

<tr>
  <td>Correo</td>
  <td><input type="text" name="Correo_comprador"></td>
</tr>
<tr>
  <td>&nbsp;</td>
  <td><input type="submit" value="Submit"></td>
</tr>
</table>
}
</fieldset>
</body>
</html>

```

Como podemos observar nos aparecerá un formulario el cual el cliente deberá rellenar unos campos que luego serán los que guardaremos en nuestra BD.

En el navegador nos deberá aparecer más o menos cuando seleccionamos la opción comprar lo siguiente:

The image shows a web form titled "Información comprador" enclosed in a rectangular border. The form contains three text input fields stacked vertically, each with a label to its left: "Nombre", "Dirección", and "Correo". Below these fields is a "Submit" button. The form is styled with a simple border and a light background.

Imagen 21. Vista Comprar

27. Cuando hemos finalizado la compra nos redirigirá a la vista gracias en la cual nos aparecerá un mensaje de gracias y luego nos redirigirá a la vista Index de Home que es nuestra página principal.

Para añadir esta vista lo realizaremos como anteriormente y el código de dicha vista será el siguiente:

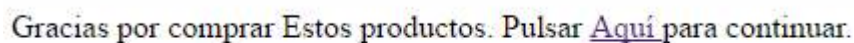
```
@{
    Layout = null;
}

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Gracias</title>
</head>
<body>
    Gracias por comprar Estos productos. Pulsar <a
href="@Url.Action("Index","Home")">Aquí </a> para continuar.
</body>
</html>
```

Como podemos observar lo único que hace esta vista es mostrar el mensaje de gracias y re direccionar a la vista principal.

Nos deberá aparecer lo que vemos en la Imagen 22:



Gracias por comprar Estos productos. Pulsar [Aquí](#) para continuar.

Imagen 22

Y en la BD veremos cómo se han insertado la nueva Factura.

Para finalizar hemos implementado la clase Email en la cual el comprador tendrá la opción de enviarse la Factura a su correo personal.

28. Lo primero que realizaremos será la implementación de la clase Email la cual el código será el siguiente:

```
using System.Net.Mail;

....

public ActionResult Correo()
{
    return View("Email");
}

[HttpPost]
public ActionResult Email(Factura fc)
{
    List<Item> carrito = (List<Item>)Session["Carrito"];

    var email = new MailMessage();

    email.Subject = fc.Nombre_factura;
    email.Body = fc.Direccion_comprador;
    email.To.Add(fc.Correo_comprador);

    email.IsBodyHtml = true;

    var smtp = new SmtpClient();
    smtp.Send(email);
    //de.SaveChanges();
    //Limpia todos los objetos del carrito
    Session.Remove("Carrito");
    return View("Gracias");
}
```

La primera acción nos llevará a la vista Email en la cual recogeremos los datos necesarios para enviarle el correo al comprador. Seguidamente crearemos la variable de tipo MailMessage en la cual rellenaremos los campos obligatorios.



Añadir que podemos aplicar lo mismo que la clase Guardar Pedido para que se guarden los cambios en nuestra BD.

29. Vamos ahora a implementar la vista Email la cual crearemos como anteriormente pero con la modificación de que ahora será de tipo crear y utilizara la clase Factura como muestra la siguiente Imagen:

Imagen creación vista Email

Agregar vista

Nombre de vista:

Plantilla:

Clase de modelo:

Clase de contexto de datos:

Opciones:

Crear como vista parcial

Hacer referencia a bibliotecas de scripts

Usar página de diseño:

(Dejar en blanco si se define en un archivo _viewstart de Razor)

Imagen 23. Vista Email.

Una vez creada la vista comentaremos la línea donde nos mostrará el id de la factura, la fecha de creación y además añadiremos un cambio al principio del código para que la vista llame a la clase Email:

```
@using (Html.BeginForm("Email", "Compra", FormMethod.Post))
```

Además comentaremos la última línea del código o la eliminaremos debido a que no la utilizaremos:

```
@* <div>  
    @Html.ActionLink("Back to List", "Index")  
</div>*
```

Una vez realizados estos cambios nos quedará modificar la clase webconfig para que nos deje enviar correos a los clientes.

30. Vamos a la clase Web.conf y al final de todo el código, antes de cerrar la configuración añadiremos el siguiente código:

```
<system.net>  
    <mailSettings>  
        //Correo desde donde enviaremos los correos al cliente  
        <smtp from="vicente_agut@hotmail.com">  
            <network enableSsl="true" defaultCredentials="false" host="smtp-  
mail.outlook.com" port="587" userName="vicente_agut@hotmail.com"  
password="Contraseña de nuestra cuenta de correo" />  
        </smtp>  
    </mailSettings>  
</system.net>
```

La configuración para enviar correos desde una cuenta de Outlook pero también se podría enviar desde Gmail cambiando la configuración ligeramente.

Ahora vamos a implementar una clase para poder acceder a nuestra app en el caso de que estemos registrados y si no es así registrarnos en la base de datos.

31. Agregaremos nuevo código en la vista Index de Home para luego dependiendo de la respuesta iremos a logearse o a registrarse.

El código de la vista será el siguiente:

```
@Html.ActionLink("Acceder", "Logearse", "Usuario")
```



```
<br /><br />  
<br /><br />  
@Html.ActionLink("Registrarse", "Registrarse", "Usuario")
```

Bienvenido a Servicolor



[Acceder](#)

[Registrarse](#)

Imagen 24. Página principal

32. Agregaremos un nuevo controlador llamado usuarioController en el cual el Código será el siguiente:

```
private servicolorbdEntities de = new servicolorbdEntities();  
  
public ActionResult Logearse()  
{  
    return View("Logearse");  
}  
  
public ActionResult Verificar(usuarios fc)  
{  
    //IEnumerable<usuarios> user = null;  
    List<usuarios> usuario = (List<usuarios>)Session["usuario"];  
  
    string user = fc.Usuario;  
    string pass = fc.Password;  
    usuarios us = de.usuarios.FirstOrDefault(d => d.Usuario == user &  
d.Password == pass);  
  
    if (us != null)  
    {  
        var Model = GetusuarioByuser(user);  
        return View("Admin", Model);  
    }  
}
```



```

    }

    else
    {
        return RedirectToAction("CerrarSesion", "Usuario");
    }
}

public ActionResult CerrarSesion()
{
    return RedirectToAction("Index", "Home");
}

// GET: Usuario
public usuarios GetusuarioByuser(string usuario)
{
    return de.usuarios.FirstOrDefault(p => p.Usuario == usuario);
}

public ActionResult Registrarse()
{
    return View("Registro");
}

public ActionResult registrar(FormCollection fc)
{
    usuarios user = new usuarios();
    user.Nombre = fc["Nombre"];
    user.Apellido = fc["Apellido"];
    user.Usuario = fc["usuario"];
    user.Password = fc["password"];
    de.usuarios.Add(user);
    de.SaveChanges();

    return RedirectToAction("Index");
}

public ActionResult Index()
{
    return View();
}
}

```

En el código siguiente vemos que lo primero que hacemos es ir a la vista Logearse si es el caso que lo hemos seleccionado o a la vista registro si por el contrario hemos pulsado éste.

33. Agregamos la vista Logearse con el siguiente código:

```

@model Proyecto.Models.usuarios

@{
    Layout = null;
}

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Logearse</title>
</head>
<body>
    @using (Html.BeginForm("Verificar", "Usuario", FormMethod.Post))
    {
        @Html.AntiForgeryToken()

        <div class="form-horizontal">
            <table>
                <tr>
                    <td>
                        
                    </td>
                    <td>
                        <label style="font-size:30px; font-family:'Comic Sans
MS'; font-
weight:bold;color:blue;">SERVICOLOR</label>
                    </td>
                </tr>
            </table>
            <hr />

            <div class="form-group">
                @Html.DisplayNameFor(model => model.Usuario)
                <div class="col-md-10">
                    @Html.TextBox("usuario")
                    @Html.ValidationMessageFor(model => model.Usuario)
                </div>
            </div>
            <br /><br />
            <div class="form-group">
                @Html.DisplayNameFor(model => model.Password)
                <div class="col-md-10">
                    @Html.Password("password")
                    @Html.ValidationMessageFor(model => model.Password)
                </div>
            </div>
            <br /><br />
            <br /><br />
            <div class="form-group">
                <div class="col-md-offset-2 col-md-10">
                    <input type="submit" value="Acceder" onclick="verif()" />
                </div>
            </div>
        </div>
    }
}

```

```

    }
</body>
    </html>

```

Ésta vista se encargara de recoger el usuario y contraseña y luego verificara si está registrado en nuestra base de datos y si no es el caso nos re direccionará a la página principal.

Imagen 25. Vista Logearse

34. Agregamos la vista Admin en la cual es el panel principal del usuario.

```

<dl class="dl-horizontal">
    <dt>
        Bienvenido de nuevo
        @Html.DisplayFor(model => model.Nombre)
        @Html.DisplayFor(model => model.Apellido)
        <br /><br /><br /><br /><br />
    </dt>
    <dd>
        <h2>¿Que le gustaría hacer?</h2>
        <br /><br />
    </dd>
    <dd>
        @Html.ActionLink("Buscar productos", "buscar", "producto")
        <br /><br /><br /><br />

```

```
<dd>  
  @Html.ActionLink("Listar productos", "Index", "producto")  
</dd>  
  
</dl>
```



Bienvenido de nuevo Vicente Agut

¿Que le gustaría hacer?

[Buscar productos](#)

[Listar productos](#)

Imagen 26. Vista Admin

Como vemos en la Imagen 26 tendremos la opción de buscar los productos que tenemos en venta o por el contrario listarlos todos.

35. Agregaremos el siguiente código en nuestra clase productoController para el caso de que el usuario seleccione buscar productos.

```
public ActionResult Buscar()
{
    return View("Buscar");
}

public ActionResult Busqueda(FormCollection fc)
{
    string var = fc["valor"];
    string opcion = fc["cerca"];

    if (opcion == "nom")
    {
        var Model = de.producto.FirstOrDefault(d => d.nombre_producto ==
var);
        return View("vista", Model);
    }

    else if (opcion == "id")
    {
        int var2 = Convert.ToInt16(var);
        var Model = de.producto.FirstOrDefault(d => d.id_producto ==
var2);
        return View("Vista", Model);
    }

    else if (opcion == "precio")
    {
        double var2 = Convert.ToDouble(var);
        var Model = de.producto.FirstOrDefault(d => d.precio_producto ==
var2);
        return View("Vista", Model);
    }

    else
    {
        return RedirectToAction("CerrarSesion", "Usuario");
    }
}
```

Se puede observar que primero accederemos a la vista buscar en la cual nos aparecerán 3 opciones y dependiendo de lo seleccionado nos buscará un producto u otro.



36. Agregaremos la vista buscar y el código será el siguiente:

```
@using (Html.BeginForm("Busqueda", "producto", /* new { id == cerca }, */
FormMethod.Post))
{
    var item = Model;
    <fieldset>
        <legend>Busqueda en los productos</legend>
        Texto a buscar: <input type="text" name="valor"><br><br>
        Selecciona una opción:<br>

        <input type="radio" name="cerca" value="nom" />Búsqueda por
nombre<br>
        <input type="radio" name="cerca" value="id" />Búsqueda por id<br>
        <input type="radio" name="cerca" value="precio" />Búsqueda por
precio<br>
    </fieldset>

    // @Html.ActionLink("Busqueda", "producto", new { item })
    <p><input type="submit" value="Buscar" /></p>
}
```

Tras realizar estas acciones podremos buscar por id, nombre o por precio.

37. Volviendo a la página principal nos quedaría implementar la clase registrarse. En la cual agregaremos la vista Registro y el código será un formulario estándar donde rellenaremos unos campos para registrarlo en nuestra base de datos.

The image shows a registration form for 'SERVICOLOR'. At the top is the logo, which consists of four overlapping circles in red, yellow, green, and blue, followed by the text 'SERVICOLOR' in blue. Below the logo is a horizontal line. The form contains four input fields: 'Nombre', 'Apellido', 'Usuario', and 'Password'. At the bottom of the form is a button labeled 'Registarse' and a link labeled 'Atras'.

Imagen 27. Vista Registro

Vamos añadir que nuestra aplicación se conecte a nuestro google calendar.

38. Nos descargaremos el código que encontraremos en el siguiente repositorio: https://github.com/ifgeny87/Test_Gapi_Calendar_V3
39. Agregaremos el proyecto descargado a el nuestro propio y además crearemos una redirección en la vista carrito para que nos aparezca la opción de acceder a google calendar. Añadir que cuando accedamos al proyecto de nuevo es posible que nos indique de descargar algun complemento para poder acceder a google calendar, lo descargaremos y luego volveremos a abrir nuestro proyecto.
40. Tras realizar dichos pasos una vez seleccionamos la opción de google calendar cuando estamos en la vista carrito, nos debería aparecer lo un recuadro donde pondremos nuestra cuenta de gmail.

Enter Account Name

Imagen 28. Acceso a Google Calendar

Una vez accedido permitiremos que la aplicación acceda a nuestra cuenta de Google.

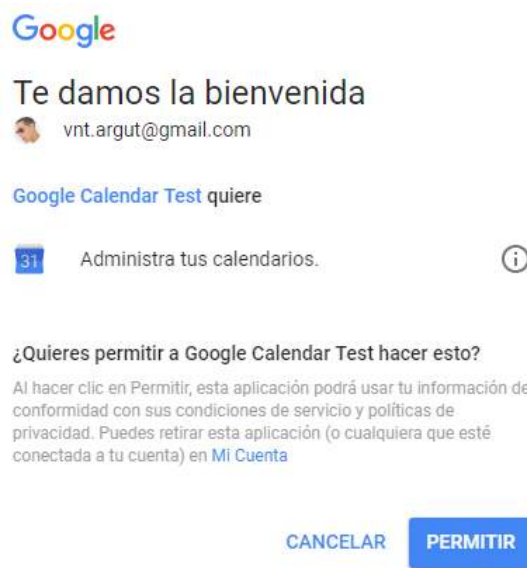


Imagen 29. Permitir Acceder a Google Calendar

41. Accederemos al calendario que deseemos y veremos los eventos que tiene dicho calendario.

Account Name: vnt.argut@gmail.com

Selected calendar: vnt.argut@gmail.com (92 events)

[Back](#)

| # | AnyoneCanAddSelf | Attendees | AttendeesOmitted | ColorId | Created | CreatedRaw | Creator | Description | End | EndTime |
|---|------------------|-----------|------------------|---------|------------------------|------------------------------|-----------------------------------------------|-------------|------------------------|---------|
| 1 | | | | | 29/01/2015 12:47:20 | 2015-01- 29T11:47:20.000Z | vicente agut navarro (vnt.argut@gmail.com) | | 09/02/2015 9:15:00 | |
| 2 | | | | | 29/01/2015 12:47:21 | 2015-01- 29T11:47:21.000Z | vicente agut navarro (vnt.argut@gmail.com) | | 09/02/2015 18:10:00 | |
| 3 | | | | | 27/02/2015 19:39:10 | 2015-02- 27T18:39:10.000Z | vicente agut navarro (vnt.argut@gmail.com) | | 19/02/2015 11:30:00 | |
| 4 | | | | | 27/02/2015 19:39:10 | 2015-02- 27T18:39:10.000Z | vicente agut navarro (vnt.argut@gmail.com) | | 19/02/2015 17:00:00 | |
| 5 | | | | | 27/02/2015 19:39:11 | 2015-02- 27T18:39:11.000Z | vicente agut navarro (vnt.argut@gmail.com) | | 03/03/2015 14:00:00 | |
| 6 | | | | | 27/02/2015 19:39:11 | 2015-02- 27T18:39:11.000Z | vicente agut navarro (vnt.argut@gmail.com) | | 26/02/2015 12:30:00 | |
| 7 | | | | | 24/04/2015 0:42:07 | 2015-04- 23T22:42:07.000Z | vicente agut navarro (vnt.argut@gmail.com) | | 04/05/2015 9:00:00 | |

Imagen 30. Ejemplo de eventos en el calendario

Con todo esto tendríamos finalizada nuestra aplicación desde ASP.NET añadir que ahora podríamos darle formato a la aplicación si queremos pero la funcionalidad básica ya la tendríamos realizada.

3.2.2.3. Subir aplicación en Windows Azure

- Una vez ya tenemos la aplicación funcionando en local procederemos a subir la aplicación a la plataforma Windows Azure para que cualquier cliente pueda acceder a ella.
- Lo primero que tenemos que hacer es tener una cuenta de Windows Azure la cual como ya comente anteriormente me descargue una licencia imagine desde la web:

https://e5.onthehub.com/WebStore/ProductsByMajorVersionList.aspx?cmi_cs=1&cmi_mnuMain=bdba23cf-e05e-e011-971f-0030487d8897&ws=1b5d1aca-826f-e011-971f-0030487d8897&vsro=8

- Una vez ya tenemos la cuenta de suscripción accederemos a la aplicación de Windows Azure mediante el siguiente link:

<https://portal.azure.com>

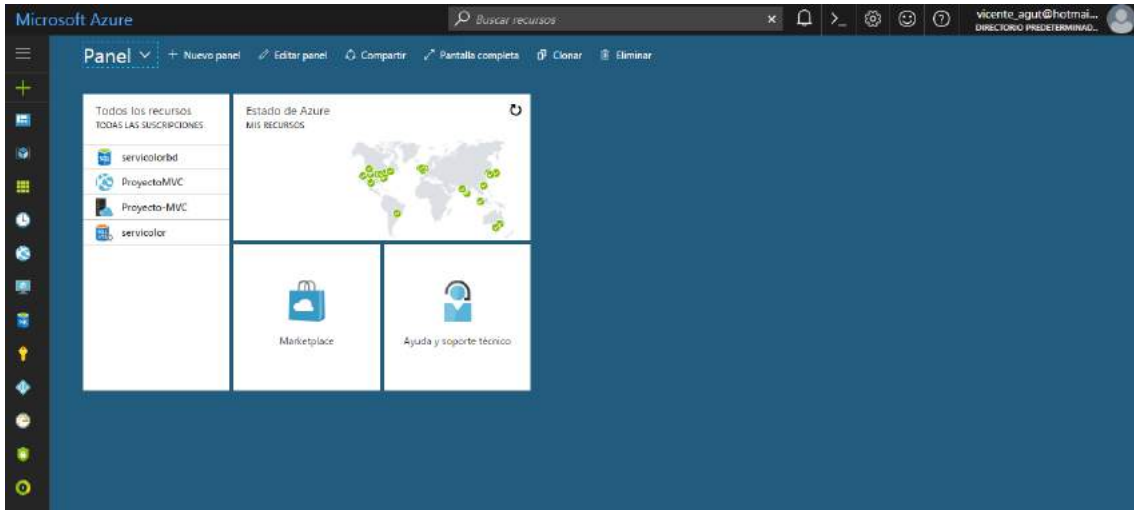


Imagen 31. Windows Azure

- Como podemos ver tenemos muchas opciones pero la opción que a nosotros nos interesa es la de SQL, pero primero deberemos publicar nuestra aplicación desde la aplicación Visual Studio.

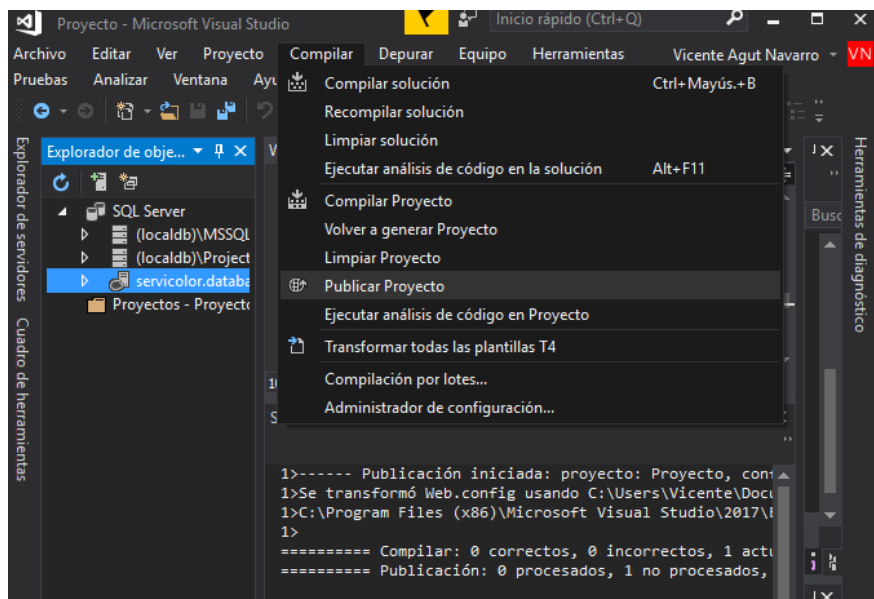


Imagen 32. Publicar Proyecto

- En la opción publicar crearemos un nuevo perfil->Servicio de aplicaciones de Microsoft Azure->Crear Nuevo y aceptar

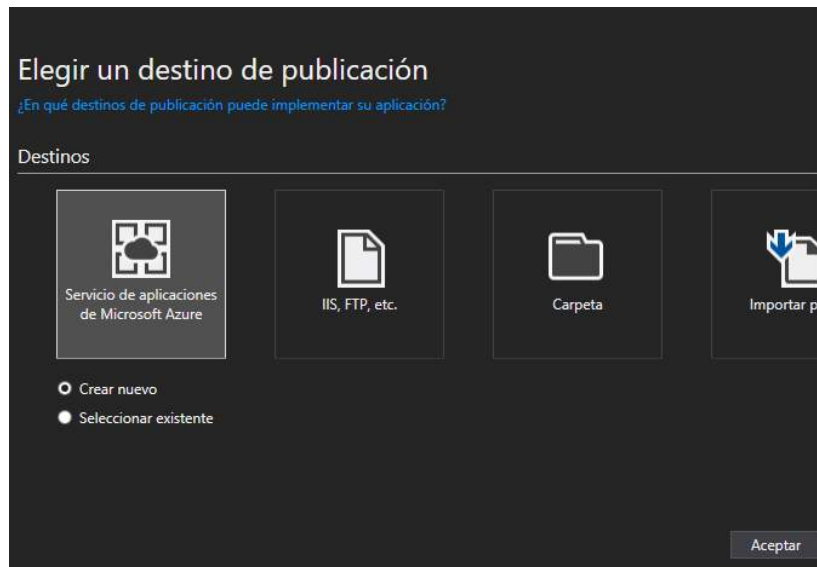


Imagen 33.Creación nuevo proyecto

- En la siguiente ventana pondremos el nombre proyecto, el tipo de subscripción el grupo de recursos y el plan de servicio.

Añadir que en el plan de servicio deberemos seleccionar nuevo y poner la ubicación donde nos encontremos y el tamaño deberá ser gratis al tener una licencia para uso de estudiante.

Una vez tenemos estos datos seleccionados le damos a crear.

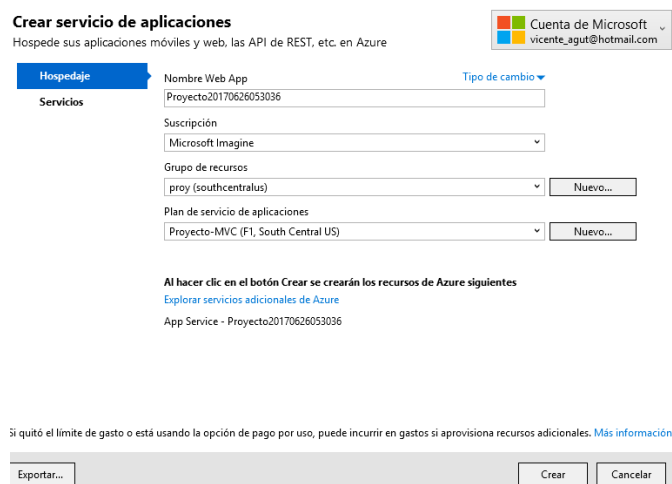


Imagen 34. Ejemplo de creación de una nueva aplicación en Windows Azure.

- Ahora que tenemos la app publicada y comprobada que no nos da ningún error, podemos ver que la aplicación no nos da ningún dato debido a que no se conecta a la base de datos.
- Vamos a la web de Azure y seleccionamos SQL-> Agregar.

Configuración para una base de datos SQL.

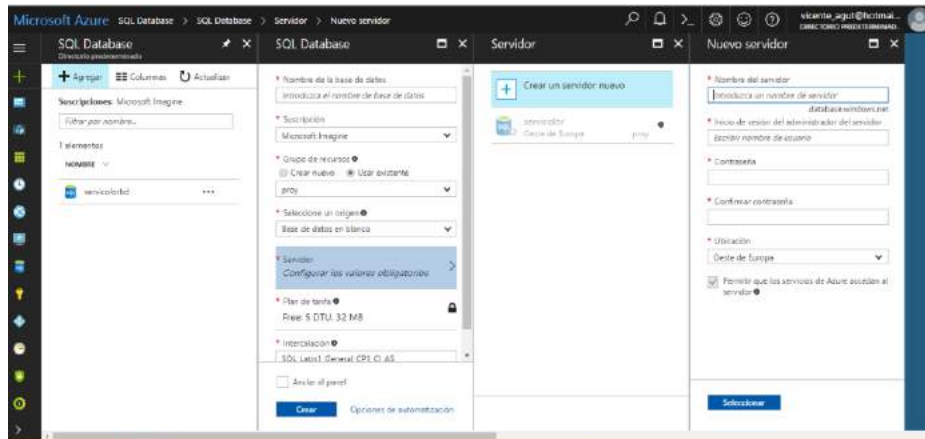


Imagen 35. Creación nueva base de datos SQL en Windows Azure.

- Rellenamos todos los campos que nos indica la web y creamos nuestra base de datos que en mi caso es servicolorbd.

Una vez creada la base de datos podemos volver a Visual Studio y agregaremos una nueva conexión la cual será a la base de datos servicolor.database.windows.net

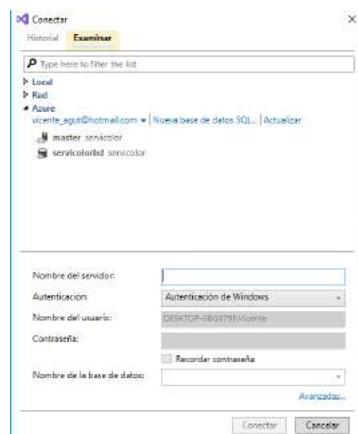


Imagen 36. Conexión a nuestra nueva base de datos creada desde Windows Azure

- Una vez conectados implementaremos nuestra base de datos que teníamos anteriormente a la nueva BD, y además también le insertamos los datos que tenemos de la lista de productos
- Tras realizar estos pasos en nuestro proyecto cambiaremos la conexión que realizábamos a la BD local y ahora la realizamos a la BD de Azure

Modificación del fichero Web.conf

```
<connectionStrings>
  <add name="servicolorbdEntities"
connectionString="metadata=res://*/Models.servicolorbd.csdl|res://*/Models.servicolorbd.ssd|res://*/Models.servicolorbd.msl;provider=System.Data.SqlClient;provider
connection string=&quot;data source=servicolor.database.windows.net;initial
catalog=servicolorbd;persist security info=True;user id=vicente;password=*Aquí su
contraseña de la BD*;multipleactiveresultsets=True;application
name=EntityFramework&quot;" providerName="System.Data.EntityClient" />
</connectionStrings>
```

- Para poder acceder localmente a nuestra base de datos en la nube deberemos añadir nuestra ip al firewall para poder acceder. Ésta opción la encontramos en: Seleccionamos nuestra base de datos -> Establecer el firewall del servidor -> Agregar IP de cliente -> Guardar.
- Actualizamos el archivo y comprobamos ahora la url de nuestra aplicación y debería funcionar. Nuestra aplicación se encuentra en la siguiente URL: <http://proyecto-servicolor.azurewebsites.net/>

| id_producto | nombre_producto | imagen | precio_producto | cantidad | Añadir al carrito |
|-------------|------------------|-----------------------------------------------------------------------------------|-----------------|----------|------------------------|
| 1 | Pintura Blanca |  | 29.95 | 500 | Añadir |
| 2 | Pintura Negra |  | 19.95 | 100 | Añadir |
| 3 | Pintura Azul |  | 19.95 | 100 | Añadir |
| 4 | Pintura Roja |  | 19.95 | 100 | Añadir |
| 5 | Pintura Amarilla |  | 19.95 | 100 | Añadir |
| 6 | Pintura Naranja |  | 29.95 | 100 | Añadir |
| 7 | Pintura Verde |  | 19.95 | 100 | Añadir |

Imagen 37. Aplicación servicolor en Windows Azure

3.2.3. Pruebas de rendimiento

Una vez la aplicación está operativa procederemos a realizar las pruebas de rendimiento.

En Visual Studio 2017 una vez ejecutamos las aplicaciones se puede observar que nos aparece lo que nos indica la Imagen 38.

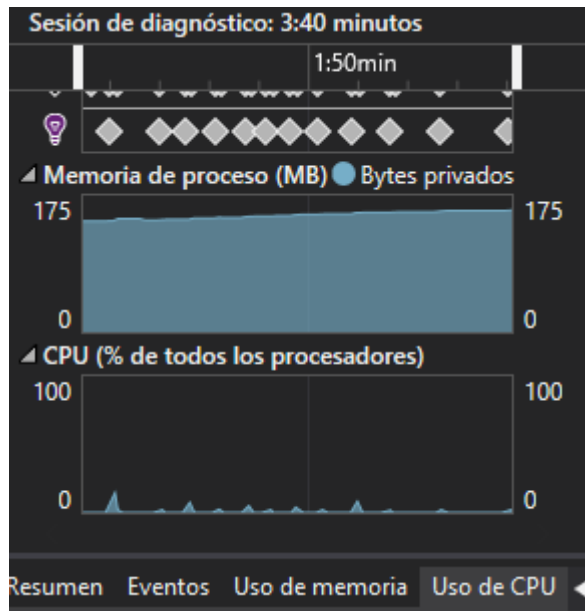


Imagen 38. Consumo memoria y CPU.

Pruebas realizadas para 3 accesos a la app desde diferentes dispositivos. Se comprueba que a medida que interactuamos por la aplicación añadiendo productos al carrito se observa que va creciendo el consumo de memoria poco a poco, y como Windows Azure utiliza el método pago por uso, dependiendo del uso de recursos nos costará un precio u otro.

3.3. Google App Engine

Ahora vamos a pasar a la implementación de la misma aplicación mediante Google App Engine (GAE) en la cual se podría implementar en diferentes lenguajes tales como: **Python, Java, .Net, PHP, etc.**

Nosotros hemos escogido implementar nuestra aplicación mediante PHP debido a que es un lenguaje bastante sencillo y hay mucha documentación de éste por internet, además como en la carrera es un lenguaje que no se toca mucho, he preferido aprenderlo un poco más.

3.3.1. Principales características de PHP

PHP es un lenguaje de programación de uso general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico. Fue uno de los primeros lenguajes de programación del lado del servidor que se podían incorporar directamente en el documento HTML en lugar de llamar a un archivo externo que procese los datos. El código es interpretado por un servidor web con un módulo de procesador de PHP que genera la página web resultante. PHP ha evolucionado por lo que ahora incluye también una interfaz de línea de comandos que puede ser usada en aplicaciones gráficas independientes. Puede ser usado en la mayoría de los servidores web al igual que en casi todos los sistemas operativos y plataformas sin ningún coste.

3.3.2. Aplicación servicolor

3.3.2.1. Pasos previos

1. Lo primero que tenemos que hacer es descargarnos el programa xampp que podemos encontrar fácilmente por internet.

Éste programa nos servirá para crear nuestra base de datos en mysql y tener abierto nuestro servidor apache.

2. Una vez hemos descargado el programa para nuestra plataforma (en mi caso ha sido en Windows) configuraremos los puertos apache debido a que por defecto escuchan en el puerto 80 y es posible que dicho puerto lo ocupe otro programa.



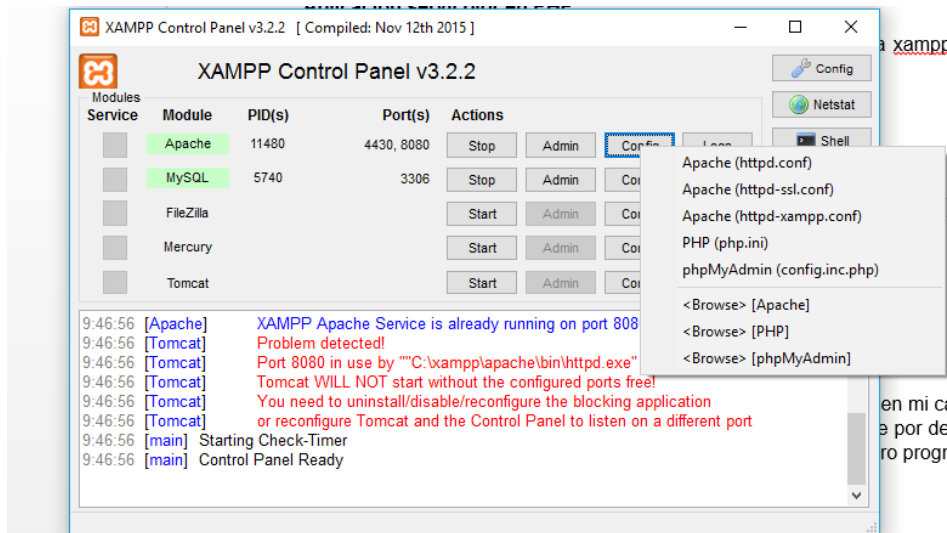


Imagen 39. Acceso a configuración apache en el XAMPP

3. Una vez llegamos donde nos muestra la Imagen, seleccionaremos la opción Apache (httpd.conf).
4. Dentro de dicho archivo buscaremos los números 80 y los sustuiremos por 8080, deberían haber dos.
5. Realizaremos los mismos pasos para el puerto 443 el cual sustuiremos por 4430.
6. Tras realizar estos cambios es conveniente que reiniciemos el apache, y luego ya iniciemos el mysql.

Ahora vamos a implementar nuestra Base de datos.

7. En el panel de control de XAMPP seleccionaremos la opción admin de MySQL y nos deberá abrir una página web como la que vemos en la siguiente Imagen:

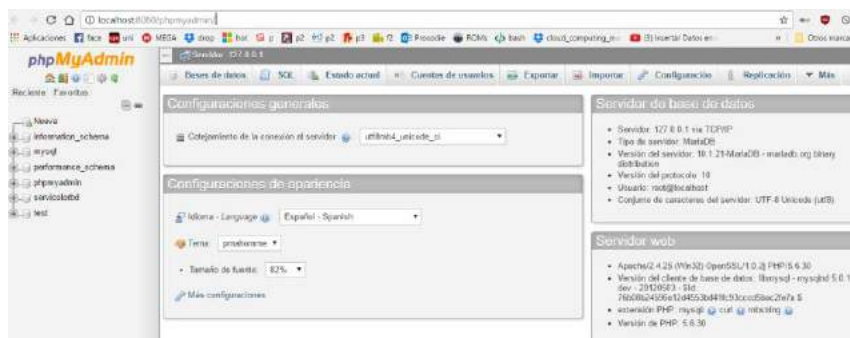


Imagen 40. PhpMyAdmin

- Seleccionaremos la Opción Nueva y crearemos nuestra base de datos con los campos que necesitemos. En mi caso mi base de datos es servicolorbd que tiene la misma estructura que la que hemos realizado en SQL server.

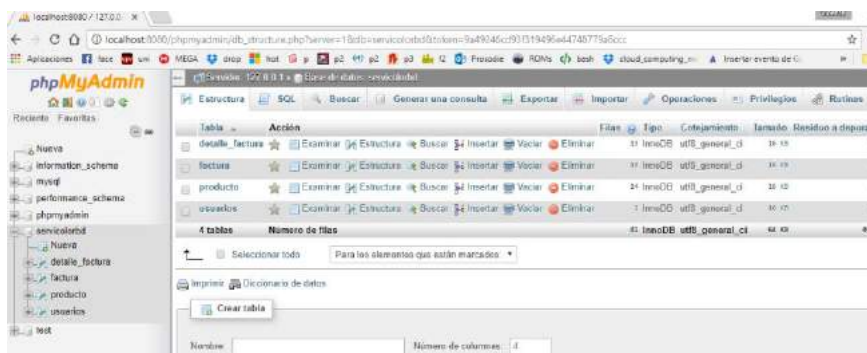


Imagen 41. Base de datos servicolorbd.

Una vez creada nuestra base de datos, ya podemos empezar a desarrollar nuestra aplicación en PHP.

En XAMPP si queremos utilizar código PHP debemos crear nuestro proyecto en la ruta siguiente: C:\xampp\htdocs. htdocs es donde crearemos nuestro proyecto debido a que las aplicaciones PHP cargan a partir de dicha ruta.

En la Imagen 39 podemos ver la ruta htdocs y donde crearemos nuestro proyecto, en mi caso el proyecto lo tengo en la carpeta proyecto.

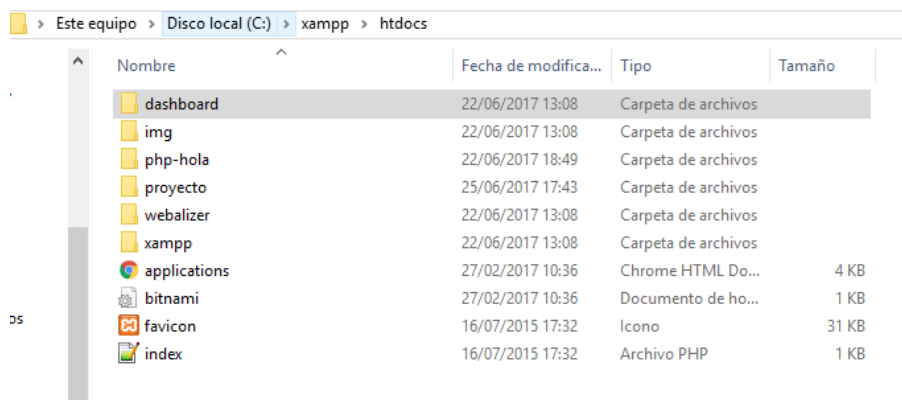


Imagen 42.Ruta donde se encuentra htdocs

Ahora ya podemos empezar a crear nuestra aplicación servicolor.



3.3.2.2. Detalles de implementación

1. Creamos un archivo nuevo llamado conexión.php y lo abrimos con un editor de texto en mi caso todo lo he editado con notepad++.

Esta clase se encargara de conectarnos con nuestra base de datos servidcolorbd y el código será el siguiente:

```
<?php
    $servidor = "localhost";
    $usuario="root";
    $clave="";
    $base="servicolorbd";
    mysql_connect($servidor,$usuario,$clave);
    mysql_select_db($base);

    function conexio(){
        $conn = mysqli_connect('localhost','root','','servicolorbd');

        return $conn;
    }
?>
```

Nos conectaremos mediante una función o solamente incluyendo ésta clase a las demás clases que crearemos en nuestro proyecto.

2. Una vez creada la conexión con nuestra BD empezamos ya a crear la primera clase donde se verán la lista de productos que tenemos en venta.

Ésta clase se llamara mostrar.php y el código es el siguiente:

```
<meta charset = "utf-8" />
<?php

    include("conexion.php");
//Consulta los productos en la base de datos.
    $consulta=mysqli_query("Select * FROM producto")or die (mysql_error());
    $registro = mysql_fetch_array($consulta);
```

```

//creamos una tabla que nos muestre los productos
echo "<table border>";
echo "<tr>";
echo "<th>Id_producto</th>";
echo "<th>Nombre_producto</th>";
echo "<th>Imagen</th>";
echo "<th>Precio Producto</th>";
echo "<th>Cantidad</th>";
echo "<th>Añadir al carrito</th>";
echo "</tr>";

do{
    $id = $registro['id_producto'];
    $nom= $registro['nombre_producto'];
    $precio= $registro['precio_producto'];
    $cantidad= $registro['cantidad'];
    echo "<tr>";
    echo "<td>$id</td>";
    echo "<td>$nom</td>";

//Dependiendo del id del producto nos muestra una Imagen o otra que //tenemos
almacenadas en la carpeta Imagenes.
    if ($id == 1)echo "<td align='center'><img src ='Imagenes/blanco.jpg'
style='height:60px;'</td>";
    if ($id == 2)echo "<td align='center'><img src ='Imagenes/negro.jpg'
style='height:60px;'</td>";
    if ($id == 3)echo "<td align='center'><img src ='Imagenes/azul.jpg'
style='height:60px;'</td>";
    if ($id == 4)echo "<td align='center'><img src ='Imagenes/roja.jpg'
style='height:60px;'</td>";
    if ($id == 5)echo "<td align='center'><img src ='Imagenes/amarilla.jpg'
style='height:60px;'</td>";
    if ($id == 6)echo "<td align='center'><img src ='Imagenes/naranja.jpg'
style='height:60px;'</td>";
    if ($id == 7)echo "<td align='center'><img src ='Imagenes/verde.jpg'
style='height:60px;'</td>";

```



```
        if ($id == 8)echo "<td align='center'><img src ='Imagenes/marron.jpg'
style='height:60px;'</td>";
        if ($id == 9)echo "<td align='center'><img src ='Imagenes/rosa.jpg'
style='height:60px;'</td>";
        if ($id == 10)echo "<td align='center'><img src ='Imagenes/violeta.jpg'
style='height:60px;'</td>";
        if ($id == 11)echo "<td align='center'><img src ='Imagenes/gris.jpg'
style='height:60px;'</td>";
        if ($id == 12)echo "<td align='center'><img src ='Imagenes/turquesa.jpg'
style='height:60px;'</td>";
        if ($id == 13)echo "<td align='center'><img src ='Imagenes/pistacho.jpg'
style='height:60px;'</td>";
        if ($id == 14)echo "<td align='center'><img src ='Imagenes/marfil.jpg'
style='height:60px;'</td>";
        echo "<td>$precio</td>";
        echo "<td>$cantidad</td>";
```

//Con esta opción tendremos la posibilidad de añadir un producto a nuestro carrito

```
        echo "<td align='center'>"?>
        <form action ="add.php" method = "post">
        <input type = "hidden" name = "id_producto" value="<?php echo $id;?>"
>
        <input type="submit" value="Añadir" />
        </form>
        <?php
        echo "</td>";
        echo "</tr>";
    }
    //mostrará todos los productos que tenemos en nuestra BD
    while ($registro = mysql_fetch_array($consulta));
    echo "</table>";
?>
```

Tras implementar éste código nos iremos a la siguiente ruta en un navegador web: <http://localhost:8080/proyecto/mostrar.php> y nos deberá aparecer lo siguiente

Imagen de la lista de productos.









| Id_producto | Nombre_producto | Imagen | Precio Producto | Cantidad | Añadir al carrito |
|-------------|------------------|-----------------------------------------------------------------------------------|-----------------|----------|---------------------------------------|
| 1 | Pintura Blanca |  | 29.95 | 500 | <input type="button" value="Añadir"/> |
| 2 | Pintura Negra |  | 19.95 | 100 | <input type="button" value="Añadir"/> |
| 3 | Pintura Azul |  | 19.95 | 100 | <input type="button" value="Añadir"/> |
| 4 | Pintura Roja |  | 19.95 | 100 | <input type="button" value="Añadir"/> |
| 5 | Pintura Amarilla |  | 19.95 | 100 | <input type="button" value="Añadir"/> |
| 6 | Pintura Naranja |  | 29.95 | 100 | <input type="button" value="Añadir"/> |
| 7 | Pintura Verde |  | 19.95 | 100 | <input type="button" value="Añadir"/> |
| 8 | Pintura Marron |  | 29.95 | 100 | <input type="button" value="Añadir"/> |

Imagen 43. Listado de productos

Como se puede observar el último campo es Añadir al carrito el cual si seleccionamos nos iremos a la clase add.php.

3. Crearemos la clase add.php en la cual veremos la sesión de carrito y será la base de las 3 opciones que tendremos: Seguir comprando, finalizar la compra o Enviar factura por correo.

El código será el siguiente:

```
<?php
//iniciamos sesión
session_start();
include("conexion.php");
if (isset($_SESSION['carrito'])){
    if(isset($_POST['id_producto'])){
        //guardamos la sesión en una variable
        $arreglo= $_SESSION['carrito'];
        $encontrar=false;
        $numero=0;
```

```

//buscamos si el producto seleccionado se encuentra en //nuestra
sesión carrito el cual si es así añade 1 a la cantidad que teníamos
for($i=0;$i<count($arreglo);$i++){
    if($arreglo[$i]['ID']==$_POST['id_producto']){

        $encontrar=true;
        $numero=$i;
    }
}
if($encontrar==true){

    $arreglo[$numero]['Cantidad']=$arreglo[$numero]['Cantidad']+1;
    $_SESSION['carrito']=$arreglo;
}
// si no encuentra ningún producto con la id seleccionada lo añadirá
//nuestra sesión carrito
else{
    $id="";
    $nom="";
    $precio=0;
    $consulta=mysql_query("Select * FROM producto where
id_producto =" .$_POST['id_producto']);
    while ($f=mysql_fetch_array($consulta)){
        $id=$f['id_producto'];
        $nom=$f['nombre_producto'];
        $precio=$f['precio_producto'];
    }
    $arreglo_nuevo=array('ID'=>$id,
                        'Nom'=>$nom,
                        'Precio'=>$precio,
                        'Cantidad'=>1);
    array_push($arreglo,$arreglo_nuevo);
    $_SESSION['carrito']=$arreglo;
}
}

```

```

}else{
    if(isset($_POST['id_producto'])){
        $id="";
        $nom="";
        $precio=0;
        $consulta=mysql_query("Select * FROM producto where
id_producto = " .$_POST['id_producto']);
        while ($f=mysql_fetch_array($consulta)){
            $id=$f['id_producto'];
            $nom=$f['nombre_producto'];
            $precio=$f['precio_producto'];
        }
        $arreglo[]=array('ID'=>$id,
                                'Nom'=>$nom,
                                'Precio'=>$precio,
                                'Cantidad'=>1);
        $_SESSION['carrito']=$arreglo;
    }
}
?>

//Para la utilización de código jquery
<script type="text/javascript" src="https://code.jquery.com/jquery-
3.2.1.min.js"></script>

//Mostramos los datos de nuestra sesión carrito
<?php
    $total= 0;
    echo "<table border>";
    echo "<tr>";
    echo "<th>Opciones</th>";
    echo "<th>Id</th>";
    echo "<th>Nombre producto</th>";
    echo "<th>Imagen</th>";
    echo "<th>Precio </th>";
    echo "<th>Cantidad</th>";
    echo "<th>Sub_total</th>";
    echo "</tr>";

```



```

if (isset($_SESSION['carrito'])){

    $datos=$_SESSION['carrito'];
    $productos= 0;
    for($i=0;$i<count($datos);$i++){
        $valor=$datos[$i]['ID'];
        $cant=$datos[$i]['Cantidad'];
        $productos = $productos + $cant;
        echo "<tr>";
        echo "<td>";
        ?>
        // Añadimos la opción de eliminar un producto de nuestro //carrito
        <form action ="eliminar.php" method = "post">
        <input type = "hidden" name = "id_producto" value="<?php echo
$valor;?>" >
        <input type = "hidden" name = "cantidad" value="<?php echo
$cant;?>" >
        <input type="submit" value="Eliminar" />
        </form>

        <?php
        echo "</td>";
        ?>

        <td><?php echo $datos[$i]['ID'];?></td>
        <td><?php echo $datos[$i]['Nom'];?></td>
        <?php
        if ($datos[$i]['ID'] == 1)echo "<td align='center'><img src
='Imagenes/blanco.jpg' style='height:60px;'</td>";
        if ($datos[$i]['ID'] == 2)echo "<td align='center'><img src
='Imagenes/negro.jpg' style='height:60px;'</td>";
        if ($datos[$i]['ID'] == 3)echo "<td align='center'><img src
='Imagenes/azul.jpg' style='height:60px;'</td>";
        if ($datos[$i]['ID'] == 4)echo "<td align='center'><img src
='Imagenes/roja.jpg' style='height:60px;'</td>";
    }
}

```



```

        if ($datos[$i]['ID'] == 5)echo "<td align='center'><img src
='Imagenes/amarilla.jpg' style='height:60px;'</td>";
        if ($datos[$i]['ID'] == 6)echo "<td align='center'><img src
='Imagenes/naranja.jpg' style='height:60px;'</td>";
        if ($datos[$i]['ID'] == 7)echo "<td align='center'><img src
='Imagenes/verde.jpg' style='height:60px;'</td>";
        if ($datos[$i]['ID'] == 8)echo "<td align='center'><img src
='Imagenes/marron.jpg' style='height:60px;'</td>";
        if ($datos[$i]['ID'] == 9)echo "<td align='center'><img src
='Imagenes/rosa.jpg' style='height:60px;'</td>";
        if ($datos[$i]['ID'] == 10)echo "<td align='center'><img src
='Imagenes/violeta.jpg' style='height:60px;'</td>";
        if ($datos[$i]['ID'] == 11)echo "<td align='center'><img src
='Imagenes/gris.jpg' style='height:60px;'</td>";
        if ($datos[$i]['ID'] == 12)echo "<td align='center'><img src
='Imagenes/turquesa.jpg' style='height:60px;'</td>";
        if ($datos[$i]['ID'] == 13)echo "<td align='center'><img src
='Imagenes/pistacho.jpg' style='height:60px;'</td>";
        if ($datos[$i]['ID'] == 14)echo "<td align='center'><img src
='Imagenes/marfil.jpg' style='height:60px;'</td>";
    ?>
    <td><?php echo $datos[$i]['Precio'];?></td>
    <td><?php echo $datos[$i]['Cantidad'];?></td>
    <td><?php echo $datos[$i]['Precio']*$datos[$i]['Cantidad'];?></td>
</tr>

<?php
    $total += ($datos[$i]['Precio']*$datos[$i]['Cantidad']);

    }
}else{
    echo "<center><h2>El carrito de compras está
vacio</h2></center>";
}
echo "<tr>";
echo "<td align='right' colspan='6'>Suma Total</td>";
echo "<td>$total</td>";

```



```

echo "</tr>";
echo "</table>";
echo " <br /><br />";
?>

```

//Esta opción nos permitirá añadir más productos a nuestro carrito

```

<form action ="mostrar.php" method = "post">
<input type="submit" value="Continuar Comprando" />
</form>
<?php
echo " <br /><br />";
?>

```

//En esta clase nos permitirá guardar la compra en nuestra BD

```

<form action ="compra.php" method = "post">
<input type = "hidden" name = "precio" value="<?php echo $total;?>" >
<input type = "hidden" name = "cant" value="<?php echo $productos;?>"
>
<input type="submit" value="Finalizar Compra" />
</form>
<?php
echo " <br /><br />";
?>

```

//Aquí nos enviará la factura por correo y además lo guardara en nuestra //BD

```

<form action ="email.php" method = "post">
<input type = "hidden" name = "precio" value="<?php echo $total;?>" >
<input type = "hidden" name = "cant" value="<?php echo $productos;?>"
>
<input type="submit" value="Factura Por Correo" />
</form>
<?php
?>

```

Como podemos observar el código es bastante extenso sobre todo para la sesión que hemos nombrado carrito, pero con esta configuración nos permitirá realizar todas las opciones descritas anteriormente.

localhost:8080/proyecto/add.php




| Opciones | Id | Nombre producto | Imagen | Precio | Cantidad | Sub_total |
|------------|----|-----------------|-----------------------------------------------------------------------------------|--------|----------|-----------|
| Eliminar | 7 | Pintura Verde |  | 19.95 | 1 | 19.95 |
| Suma Total | | | | | | 19.95 |

Continuar Comprando

Finalizar Compra

Factura Por Correo

Imagen 44. Clase add.php

| Opciones | Id | Nombre producto | Imagen | Precio | Cantidad | Sub_total |
|------------|----|------------------|-------------------------------------------------------------------------------------|--------|----------|-----------|
| Eliminar | 7 | Pintura Verde |  | 19.95 | 2 | 39.9 |
| Eliminar | 3 | Pintura Azul |  | 19.95 | 1 | 19.95 |
| Eliminar | 5 | Pintura Amarilla |  | 19.95 | 1 | 19.95 |
| Suma Total | | | | | | 79.8 |

Continuar Comprando

Finalizar Compra

Factura Por Correo

Imagen 45. Clase add.php, añadiendo más de 1 producto.

En la opción precio total nos muestra el total de nuestra compra.

4. La opción continuar comprando no necesita implementarse porque lo único que hace es volver a la página de mostrar.php en la cual podemos seguir comprando productos.
5. Ahora vamos a implementar la clase Finalizar compra en la cual guardará los datos de la compra a nuestra BD. Dicha clase la he llamado compra.php y el código es el siguiente:

```
<?php
session_start();

//Iniciamos la sesión carrito e incluimos la clase conexión.php para conectarnos.
include("conexion.php");

//Nos guardamos las variables facilitadas por el formulario anterior
$total=$_POST['precio'];
$cant=$_POST['cant'];

//Guardamos la fecha actual en una variable
$fecha = strftime("%Y-%m-%d-%H-%M-%S",time());

?>

//Creamos un formulario en el cual el comprador rellenará unos campos.
<form action ="insertar.php" method = "post">

    Nombre:

    <input type = "text" name = "ncomprador" >

    <br /><br />

    Dirección:

    <input type = "text" name = "dcomprador" >

    <br /><br />

    Correo:

    <input type = "text" name = "ccomprador" >
```

```

<br /><br />

<input type = "hidden" name = "precio" value="<?php echo $total;?>" >

<input type = "hidden" name = "cantidad" value="<?php echo $cant;?>" >

<input type = "hidden" name = "nfactura" value="Nueva Factura" >

<input type = "hidden" name = "fcreacion" value="<?php echo $fecha;?>">

<br />

<input type="submit" value="Insertar" />

</form>

```

Una vez rellenado los campos del formulario nos iremos a la clase insertar.php en la cual insertaremos los datos a nuestra BD.

The image shows a web browser window with the address bar displaying 'localhost:8080/proyecto/compra.php'. Below the address bar, there are several social media and utility icons. The main content area contains a form with three text input fields. The first field is labeled 'Nombre:', the second 'Dirrección:', and the third 'Correo:'. Below these fields is a button labeled 'Insertar'.

Imagen 46. Formulario de compra.php

6. Crearemos la clase insertar.php y el código será el siguiente:

```

<meta charset = "utf-8" />
<?php
    include("conexion.php");
    session_start();

    $conn=conexio();

```

```
$insertar1="INSERT INTO
factura(Nombre_factura,Fecha_creacion,Nombre_comprador,
        Direccion_comprador,Correo_comprador)
VALUES
('${_POST['nfactura']}', '${_POST['fcreacion']}', '${_POST['ncomprador']}',
        '${_POST['dcomprador']}', '${_POST['ccomprador']}');";
mysqli_query($conn,$insertar1);
```

```
$insertar2="INSERT INTO detalle_factura(Cantidad,Precio)
VALUES ('${_POST['cantidad']}', '${_POST['precio']}');";
mysqli_query($conn,$insertar2);
```

```
session_destroy();
```

```
/**/
```

```
?>
```

```
<script language="javascript" type="text/javascript" >
location.href="/mostrar.php";
</script>
```

Como podemos observar se conecta a nuestra base de datos y luego inserta los datos en ella.

Para finalizar destruimos la sesión carrito y volvemos a la página principal que nuestro caso es mostrar.php.

Tras realizar éstos pasos los datos insertados deberían verse en nuestra BD

Mostrando filas 0 - 4 (total de 5, La consulta tardó 0.0005 segundos.)

SELECT * FROM `factura`

Perfilando [Editar en línea] [Editar] [Explicar SQL] [Crear código PHP]

Mostrar todo | Número de filas: 25 | Filtrar filas: Buscar en esta tabla | Ordenar según la clave: Ninguna

Opciones

| | Id_factura | Nombre_factura | Fecha_creacion | Nombre_comprador | Direccion_comprador | Correo_comprador |
|----------------------|------------|----------------|----------------|------------------|---------------------|--------------------------|
| Editar Copiar Borrar | 1 | Nueva Factura | 0000-00-00 | pepito | c/pepito | v@pepito.com |
| Editar Copiar Borrar | 5 | Nueva Factura | 2017-06-25 | pepita | c/pepitaaa | v@pepita.com |
| Editar Copiar Borrar | 6 | Nueva Factura | 2017-06-25 | pepito | c/pepito | v@pepito.comaaa |
| Editar Copiar Borrar | 7 | Nueva Factura | 2017-06-25 | pepito | c/pepitaaa | vicente_agut@hotmail.com |
| Editar Copiar Borrar | 8 | Nueva Factura | 2017-06-25 | pepito | c/pepito | vicente_agut@hotmail.com |

Imagen 47. Datos insertados en nuestra BD

7. Para finalizar implementaremos la clase que nos permitirá enviar correos mediante nuestra cuenta de correo Hotmail.

Primero que nada necesitaremos las clases phpmailer y la clase class.smtp.php, que podemos encontrar fácilmente mediante internet.

El código no lo voy a poner debido a que es bastante extenso.

8. Ahora vamos a implementar la clase email.php que es idéntica a insertar.php con el cambio que estos datos los envía a insertar_email.php

9. La clase insertar_email.php se encargara de enviar la factura a la dirección seleccionada por el usuario. El código será el siguiente:

```
<meta charset = "utf-8" />
<?php
    include("conexion.php");
    session_start();

    //insertamos en nuestra Base de Datos
    $conn=conexio());
    $insertar1="INSERT INTO
factura(Nombre_factura,Fecha_creacion,Nombre_comprador,
Direccion_comprador,Correo_comprador)
```

```
VALUES
```

```
('{$_POST['nfactura']}', '{$_POST['fcreacion']}', '{$_POST['ncomprador']}',  
  '{$_POST['dcomprador']}', '{$_POST['ccomprador']}');";  
mysqli_query($conn,$insertar1);
```

```
$insertar2="INSERT INTO detalle_factura(Cantidad, Precio)  
VALUES ('{$_POST['cantidad']}', '{$_POST['precio']}');";  
mysqli_query($conn,$insertar2);
```

//referenciamos al archivo phpmailer para poder enviar correos desde nuestro correo Hotmail.

```
require_once('phpmailer.php');
```

```
$destinatario = $_POST['ccomprador'];
```

```
$asunto = $_POST['nfactura'];
```

```
$cuerpo = $_POST['precio'];
```

```
$nombre = $_POST['ncomprador'];
```

```
$dir = $_POST['dcomprador'];
```

```
$cantidad = $_POST['cantidad'];
```

```
$smtp=new PHPMailer();
```

```
//indico a la clase que use SMTP
```

```
$smtp->IsSMTP();
```

```
$smtp->CharSet="UTF-8";
```

```
//Debo de hacer autenticación SMTP
```

```
$smtp->SMTPAuth = true;
```

```
$smtp->SMTPSecure = "tls";
```

```
$smtp->Host = "smtp.live.com";
```

```
$smtp->Port = 587;
```

```
$smtp->Username = "vicente_agut@hotmail.com";
```

```
$smtp->Password = "*****";
```



```

$smtp->SetFrom("vicente_agut@hotmail.com","Codigo PHP");

$contenidoHTML="<head>";
$contenidoHTML.="<meta                http-equiv=\\"Content-Type\\"
content=\\"text/html; charset=UTF-8\>";
$contenidoHTML.="</head> <body>";
$contenidoHTML = "Nombre: $nombre<br />Direccion: $dir <br />Total
productos: $cantidad<br />Precio : $cuerpo";
$contenidoHTML.="</body>\n";

$smtp->MsgHTML($contenidoHTML);
$smtp->Subject=("Factura Servicolor");
$smtp->AddAddress($destinatario, "Nombre completo");

if(!$smtp->Send()) {
echo "Error al enviar: " . $smtp->ErrorInfo;
} else {
echo "Mensaje enviado!";
}

session_destroy();

?>
<script language="javascript" type="text/javascript" >
    location.href="/mostrar.php";
</script>

```

Insertamos en nuestra BD y luego enviamos el correo al destinatario que nos hayan indicado en el formulario anterior. Para finalizar destruye la sesión y vamos a la página principal.

Ahora vamos a proceder a añadir un poco más de funcionalidad a nuestra aplicación.

10. Crearemos una clase main.php la cual será nuestra página principal y el código será el siguiente:

```
<!DOCTYPE html>
```



```

<html lang="es">
<head>
  <meta charset="utf-8"/>
  <title>Servicolor</title>
</head>

<body>
<header>
  <h1>Bienvenido a Servicolor</h1>
</header>
<table>
<tr>
<td>
  
  </td>
<td>
  <label style="font-size:30px; font-family:'Comic Sans MS';
                                font-
weight:bold;color:blue;">SERVICOLOR</label>
  </td>
</tr>
</table>
  <form action ="login.php" method = "post">
    <br /><br />
    <br />
    <input type="submit" value="Login" />
  </form>

  <form action ="registrarse.php" method = "post">
    <br /><br />
    <br />
    <input type="submit" value="Registrarse" />
  </form>
</body>
</html>

```

Bienvenido a Servicolor



Login

Registrarse

Imagen 48. Página principal

Vemos que tendremos 2 opciones para acceder a nuestra aplicación: primero acceder si estamos registrados y si no es el caso registrarnos.

11. Crearemos la clase login.php y el código será el siguiente:

```
<form id="formulario" method="post" action="verificar.php">
    <?php
        if(isset($_GET['error'])){
            echo '<center>Datos No Validos</center>';
        }
    ?>
    <label for="usuario">Usuario</label><br>
    <input type="text" id="usuario" name="Usuario"
placeholder="usuario" ><br>
    <label for="password">Password</label><br>
    <input type="password" id="password" name="Password"
placeholder="password" ><br>
    <input type="submit" name="aceptar" value="Aceptar"
class="aceptar">
</form>
```

En dicho código iremos a la clase verificar.php para comprobar si el usuario está registrado.

12. La clase verificar.php tendrá el siguiente código:

```
<?php
```



```

ob_start();
session_start();
include "conexion.php";
$re=mysql_query("select * from usuarios where Usuario='".$_$_POST['Usuario']."'
AND
                                Password='".$_$_POST['Password']."'")
or die(mysql_error());
while ($f=mysql_fetch_array($re)) {
    $arreglo[]=array('Nombre'=>$f['Nombre'],
                    'Apellido'=>$f['Apellido']);
}
if(isset($arreglo)){
    $_SESSION['Usuario']=$arreglo;

    /*echo "<form action ='admin.php' method = 'post'>";
    echo "<br /><br />";
    echo "<br />";
    echo "<input type='submit' value='Acceder' />";
    echo "</form>";*/

    //echo          "<meta          http-equiv='Location'
content='http://localhost:10080/admin.php';
    header("Location: ./admin.php");
}else{
    header("Location: login.php?error=datos no validos");
}
ob_end_flush();
?>

```

Una vez se ha verificado si el usuario está registrado nos llevara a la clase admin.php en donde nos mostrará las opciones que podemos realizar con nuestro usuario.

13. La clase admin.php tendrá el siguiente código:

```

<?php
session_start();
ob_start();

```

```

include("barra_sup.php");
include "conexion.php";
if(isset($_SESSION['Usuario'])){

}

}else{
    header("Location: ./main.php?Error=Acceso denegado");
}
ob_end_flush();
?>
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="utf-8"/>
    <title>Panel de Administración</title>

</head>
<body>
    <section>
        <nav class="menu2">
            <menu>
                <li><a href="./mostrar.php">Listar productos</a></li>
                <li><a href="./buscar.php" class="selected">Buscar productos</a></li>
            </menu>
        </nav>

<?php

$datos = $_SESSION['Usuario'];

for($i=0;$i<count($datos);$i++){
    $valor=$datos[$i]['Nombre'];
    $cant=$datos[$i]['Apellido'];
}

?>

<center><h1>Bienvenido</h1></center>
<table border="0px" width="100%">

```



```
<tr><?php
    echo "<td>Hola de nuevo ";
    echo "$valor ";
    echo "$cant";
    echo"</td>";
?>

</tr>
</table>
</section>
<br /><br />
<form action = "main.php" method = "post">
<br /><br />
<br />
<input type="submit" value="Cerrar Sesión" />
</form>
<?php

?>

</body>
</html>
```



- [Listar productos](#)
- [Buscar productos](#)

Bienvenido

Hola de nuevo Vicente Agut

Cerrar Sesión

Imagen 49. Clase Admin.php

Nos aparecerán las opciones de listar los productos que tenemos en venta, los cuales nos llevaría a la clase mostrar.php o por el contrario iremos a buscar productos.

14. Creamos la clase buscar.php con el siguiente código:

```
<?php
    include "barra_sup.php";
?>

<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Buscar</title>
</head>
<body>
<form action="busqueda.php" method="POST">
<fieldset>
    <legend>Busqueda en los productos</legend>
    Texto a buscar: <input type="text" name="value" /><br><br>
    Selecciona una opción:<br>
    <input type="radio" name="cerca" value="nom" />Búsqueda por
nombre<br>
    <input type="radio" name="cerca" value="id" />Búsqueda por id<br>
    <input type="radio" name="cerca" value="precio" />Búsqueda por
precio<br>
</fieldset>
<input type="submit" name="buscar" value="Buscar" />
</form>
</body>
</html>

<?php
    include "barra_inferior.php";
?>
```



Busqueda en los productos

Texto a buscar:

Selecciona una opción:

Búsqueda por nombre

Búsqueda por id

Búsqueda por precio

Buscar

Atras

Imagen 50. Clase buscar.php

Añadir que hemos creado unas clases genéricas que utilizamos en muchas clases para no tener que estar escribiendo el mismo código cada vez, dichas clases son barra_sup.php y barra_inferior.php.

Dependiendo del tipo de búsqueda nos buscare por el parámetro que seleccionemos y en la clase búsqueda.php nos mostrara los campos que busquemos.

15. La clase búsqueda.php tendrá el siguiente código:

```
<?php
include "conexion.php";
include "barra_sup.php";

$opcio=$_POST['cerca'];
// echo $opcio;
$conn = conexion();
if ($opcio == "nom") {
    //echo $_POST['nom'];
    $consulta1=mysqli_query($conn,"select * from producto where
nombre_producto LIKE '%"$_POST['value']."%';");

    echo "<table border='2'>";
    while ($row = mysqli_fetch_array($consulta1)){
        echo "<tr>";
        //echo "<td>".$row['id']."</td>";
```



```

        echo "<td>".$row['id_producto']."</td>";
        echo "<td>".$row['nombre_producto']."</td>";
        echo "<td>".$row['precio_producto']."</td>";
        echo "<td>".$row['cantidad']."</td>";
        echo "</tr>";
    }
    echo "</table>";
}elseif ($opcion == "id"){
    //echo $_POST['nom'];
    $consulta2=mysqli_query($conn, "select * from producto where id_producto
LIKE '".$_POST['value']."'");

    echo "<table border='2'>";
    while ($row = mysqli_fetch_array($consulta2)){
        echo "<tr>";
        //echo "<td>".$row['id']."</td>";
        echo "<td>".$row['id_producto']."</td>";
        echo "<td>".$row['nombre_producto']."</td>";
        echo "<td>".$row['precio_producto']."</td>";
        echo "<td>".$row['cantidad']."</td>";
        echo "</tr>";
    }
    echo "</table>";

}else {
    $consulta3=mysqli_query($conn, "select * from producto where
precio_producto LIKE '".$_POST['value']."'");

    echo "<table border='2'>";
    while ($row = mysqli_fetch_array($consulta3)){
        echo "<tr>";
        //echo "<td>".$row['id']."</td>";
        echo "<td>".$row['id_producto']."</td>";
        echo "<td>".$row['nombre_producto']."</td>";
        echo "<td>".$row['precio_producto']."</td>";
        echo "<td>".$row['cantidad']."</td>";
        echo "</tr>";
    }
}

```



```

    }
    echo "</table>";

}
include "barra_inferior.php";
?>

```

| | | | |
|----|------------------------|-------|-----|
| 1 | Pintura Blanca | 29.95 | 500 |
| 2 | Pintura Negra | 19.95 | 100 |
| 3 | Pintura Azul | 19.95 | 100 |
| 4 | Pintura Roja | 19.95 | 100 |
| 5 | Pintura Amarilla | 19.95 | 100 |
| 6 | Pintura Naranja | 29.95 | 100 |
| 7 | Pintura Verde | 19.95 | 100 |
| 8 | Pintura Marron | 29.95 | 100 |
| 9 | Pintura Rosa | 29.95 | 100 |
| 10 | Pintura Violeta | 29.95 | 100 |
| 11 | Pintura Gris | 29.95 | 100 |
| 12 | Pintura Azul Turquesa | 29.95 | 100 |
| 13 | Pintura Verde Pistacho | 29.95 | 100 |

Atras

Imagen 51. Búsqueda por la palabra Pintura.

Ahora vamos a implementar la clase para registrarnos en las cuales lo que realizara el usuario será rellenar los datos de un formulario que luego insertaremos en nuestra base de datos servicolorbd.

16. La clase registrar.php tendrá el siguiente código:

```

<?php

include("barra_sup.php");

?>

```

```

<form action ="registro.php" method = "post">
    Nombre:
    <input type = "text" name = "nom" >
    <br /><br />
    Apellidos:
    <input type = "text" name = "ape" >
    <br /><br />
    Nombre usuario:
    <input type = "text" name = "user" >
    <br /><br />
    Password:
    <input type = "password" name = "pas1" >
    <br /><br />
    Repite el password:
    <input type = "password" name = "pas2" >
    <br /><br />

    <input type="submit" value="Registrarse" />
</form>

<?php
    include "barra_inferior.php";
?>

```

17. La clase registro.php tendrá este código:

```

<?php

    ob_start();
    include("conexion.php");
    session_start();

    $conn=conexio();

    $pass1= $_POST['pas1'];
    $pass2= $_POST['pas2'];

```



```
if ($pass1 == $pass2){
    $insertar1="INSERT INTO usuarios(Nombre,Apellido,Usuario,
    Password)
    VALUES ('${_POST['nom']}', '${_POST['ape']}', '${_POST['user']}',
    '$pass1');"
    //echo $insertar1;
    mysqli_query($conn,$insertar1);
}

else{
    echo "Error";

    header("Location: ./main.php?Error=no coinciden las
    contraseñas");
}
header("Location: ./admin.php");
ob_end_flush();
?>
```

En dicho código dependiendo si se ha registrado correctamente o no, nos llevara a la página principal o a la página de nuestro usuario.

Ahora vamos a añadirle una función extra a nuestra aplicación.

- Accederemos a la siguiente dirección:
<https://console.developers.google.com/flows/enableapi?apiid=calendar>

Registra tu aplicación para utilizar Google Calendar API en Consola de APIs de Google

La Consola de APIs de Google te permite administrar tu aplicación y supervisar el uso de la API.

Selecciona un proyecto en el que registrar la aplicación

Puedes utilizar un proyecto para gestionar todas tus aplicaciones o crear un proyecto diferente para cada una de ellas.

Crear proyecto

Continuar

Imagen 52. Acceso API google calendar

- Accederemos a la pantalla de autorización de OAuth y rellenaremos los campos.

The screenshot shows the 'Credenciales' page in the API Administrator. The left sidebar has 'Credenciales' selected. The main content area has tabs for 'Credenciales', 'Pantalla de autorización de OAuth', and 'Verificación de dominio'. The 'Pantalla de autorización de OAuth' tab is active. It contains several form fields: 'Dirección de correo electrónico' (vnt.argut@gmail.com), 'Nombre de producto mostrado a los usuarios' (WebDev), 'URL de página principal (Opcional)' (https:// o http://), 'URL de logotipo de producto (Opcional)' (http://www.example.com/logo.png), 'URL de la Política de Privacidad' (https:// o http://), and 'URL de las Condiciones de Servicio (Opcional)' (https:// o http://). There are 'Guardar' and 'Cancelar' buttons at the bottom. A preview of the authorization screen is shown on the right, along with explanatory text: 'La pantalla de autorización se muestra a los usuarios cuando solicitas acceso a sus datos privados mediante tu ID de cliente. Se muestra para todas las aplicaciones registradas en este proyecto.' and 'Debes proporcionar una dirección de correo electrónico y un nombre de producto para que OAuth funcione.'

Imagen 53. Pantalla de autorización

- Una vez creada la autorización accederemos a la opción credenciales con ID de cliente autorizado. Escogemos la opción otro y escribimos lo mismo que en el apartado anterior, en nuestro caso WebDev

The screenshot shows the 'Crear ID de cliente' page in the API Administrator. The left sidebar has 'Credenciales' selected. The main content area has a back arrow and the title 'Crear ID de cliente'. It contains a 'Tipo de aplicación' section with radio buttons for 'Web', 'Android Más información', 'Chrome Más información', 'iOS Más información', 'PlayStation 4', and 'Otro' (selected). There is a 'Nombre' field containing 'WebDev' and 'Crear' and 'Cancelar' buttons at the bottom.

Imagen 54. Creación ID de cliente

- Nos descargamos el archivo creado y copiamos los datos obtenidos en dicho archivo xxx.json el client_id y el client_secret.

- Ahora nos iremos al siguiente repositorio y descargaremos toda la carpeta donde se encuentre nuestro proyecto. <https://github.com/google/google-api-php-client>.
- Una vez descargado todas las carpetas, instalaremos la api de Google Calendar en nuestro proyecto con el siguiente comando :
composer require google/apiclient
- Ahora ya tenemos todos los archivos preparados entonces crearemos un archivo nuevo el cual llamaremos calendar.php con el siguiente código.

```
<html>
<body>
<?php
    session_start();
    // print_r($_SESSION);
    if(isset($_GET["logout"])){
        session_destroy();
    }
    $redirect_uri ='http://localhost/proyecyo/google-api-php-client-
    master/calendar.php';
    require_once './vendor/autoload.php';
    $client = new Google_Client();
    // Get your credentials from the console
    $client->setClientId('*Inserte su clientID*');
    $client->setClientSecret('*Inserte su clave secreta*');
    $client->setRedirectUri($redirect_uri);
    $client->addScope('profile');
    $client->addScope(Google_Service_Calendar::CALENDAR);
    print_r($client->getAccessToken());
    $authUrl = $client->createAuthUrl();
    if (isset($_GET['code'])) {
        $client->authenticate($_GET['code']);
        $_SESSION['access_token'] = $client->getAccessToken();
        header('Location: ' . filter_var($redirect_uri, FILTER_SANITIZE_URL));
    }
    if (!$client->getAccessToken() && !isset($_SESSION['access_token'])) {
        $authUrl = $client->createAuthUrl();
        print "<a class='login' href='$authUrl'>Conectar</a>";
    }

```

```

    }
    if (isset($_SESSION['access_token'])) {
        print "<a class='logout'
href='".$_._SERVER['PHP_SELF']."'?logout=1'>Salir</a><br>";
        $client->setAccessToken($_SESSION['access_token']);
        $service = new Google_Service_Calendar($client);
        $results = $service->events->listEvents('primary', array());
    if (count($results->getItems()) == 0) {
        print "<h3>No hay Eventos</h3>";
    } else {
        print "<h3>Proximos Eventos</h3>";
        echo "<table border=1>";
        foreach ($results->getItems() as $event) {
            echo "<tr>";
            $start = $event->start->dateTime;
            if (empty($start)) {
                $start = $event->start->date;
            }
            echo "<td>".$event->getSummary()."</td>";
            echo "<td>".$start."</td>";
            echo "</tr>";
        }
        echo "<table>";
    }
}
?>
</body>
</html>

```

Insertaremos los `client_id` y el `client_secret` donde corresponde y ya podremos ejecutar nuestro programa.

Ahora nos quedaría añadir un botón en la clase `add.php` para que nos redirija a la clase `calendar.php`.



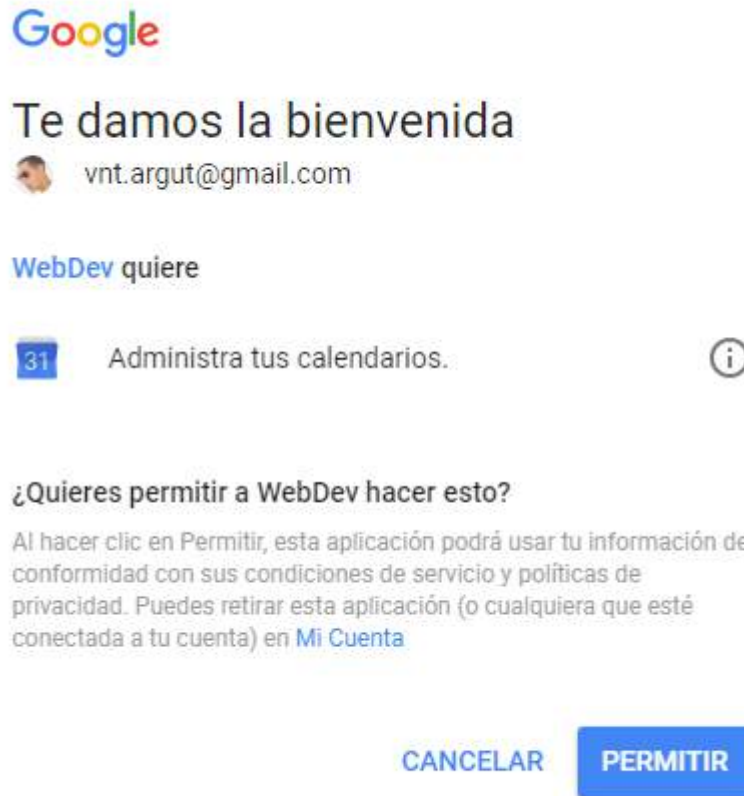


Imagen 55. Permitir acceso de nuestra aplicación.

- Obtendremos el token y podremos acceder a nuestro google calendar.

Como dato añadido en la biblioteca de google tenemos una amplica API con todas las opciones que podemos implementar en nuestra aplicación.

Una vez realizados todos éstos pasos nos quedará darle formato a nuestra aplicación por ejemplo en css.

Tras realizar todas estas acciones ya solo nos quedará publicar la aplicación en GAE.

3.3.2.3. Subir aplicación a GAE

- Primero debemos incluir una cuenta de correo den GAE en la cual tendremos unos 30 días de prueba. En mi caso he puesto mi correo personal de Gmail.

- Tras realizar dicho paso deberemos descargarnos Google Cloud SDK Shell, que la encontraremos en la página principal de google.
- También deberemos descargarnos Python, la versión deberá ser desde 2.7.5 en adelante, y seleccionaremos que nos incluya la ruta a la variable Path
- Tras haber descargado los programas ejecutaremos Google Cloud SDK Shell.
- Una vez dentro del programa comprobaremos que el programa nos funciona correctamente con el comando gcloud init.

Si no nos da ningún error seleccionaremos nuestra cuenta y la región que estamos, y el proyecto el cual queremos trabajar.

Si no hemos creado ningún proyecto en GAE por la versión web, lo podemos crear por la consola también, seguiremos los pasos y ya estaremos dentro del proyecto donde se ejecutará nuestra aplicación web.

- Ahora procederemos a cambiar a la ruta donde esté nuestro proyecto.
- Ahora crearemos un nuevo archivo donde se encuentre nuestro proyecto el cual llamaremos app.yaml el cual es necesario para subir archivos a la nube.

Dentro de dicho fichero implementaremos el siguiente código:

```
#application: php-proyecto-171815
#version : 1
runtime: php55
api_version: 1
threadsafe: yes

handlers:
- url: /Imagenes
  static_dir: Imagenes

- url: /blanco.jpg
  static_files: Imagenes/blanco.jpg
  upload: Imagenes/blanco.jpg
  expiration: 30d

- url: /\.(htm$|html$|css$|js$)
  static_files: \1
```



```
upload: (*.*(htm$|html$|css$|js$))  
application_readable: true
```

```
- url: /(.)+  
  script: \1
```

```
- url: /*  
  script: main.php
```

Con esta configuración le estamos diciendo que la aplicación es php, la versión es la 1, podemos tener diferentes versiones de nuestro código para poder ir viendo las diferencias que vamos obteniendo mientras vamos implementando código.

Y en los handlers es la clase para que nuestro código pueda ser leído por todas las clases y además poder los diferentes tipos de código e Imágenes. Por último la última línea de los handlers nos sirve para indicar dónde empieza nuestro código.

- Ejecutamos el comando `gcloud app deploy`.

Dicho comando se encargará de publicar nuestra aplicación en la nube, si no nos da ningún error la aplicación subirá a la nube, añadir que como máximo podemos subir 10000 archivos en cada versión.

- Ejecutamos `gcloud app browse` o vamos a un navegador web a la siguiente url: <https://php-proyecto-171815.appspot.com>.

Nos servirá para ver la aplicación en la nube, mediante un navegador web.

3.3.3. Pruebas de rendimiento

Igual que hemos hecho en Windows Azure ahora vamos a realizar las pruebas en GAE.

Como nuestra aplicación está creada internamente sin ayuda de ninguna aplicación externa, solo la ayuda de un editor de texto, en este caso para comprobar el rendimiento

deberemos acceder a GAE y ver la evolución de los consumos de memoria y CPU en la misma aplicación.

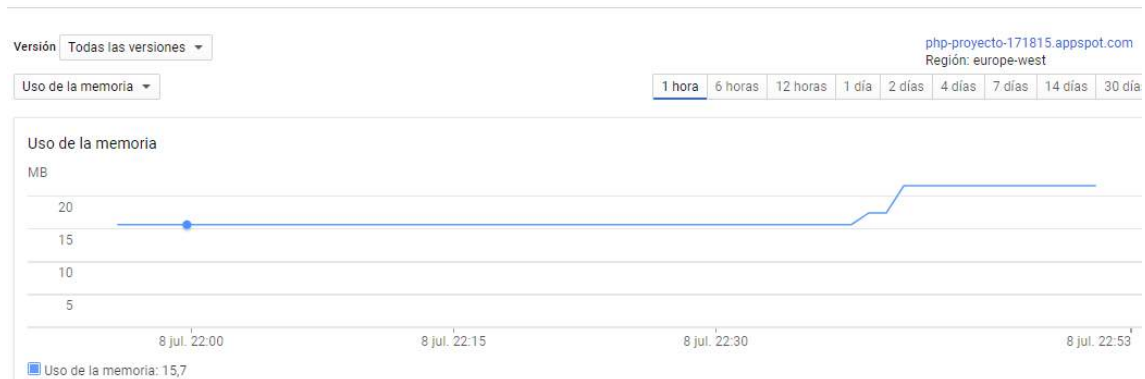


Imagen 56. Uso de memoria

Las pruebas se han realizado desde un acceso a nuestra aplicación desde 3 dispositivos y añadiendo productos a nuestro carrito de comprar, se comprueba que la primera vez le cuesta bastante pero a medida que añadimos más productos la carga es menor.

3.4. Comparativa entre GAE y Windows Azure en nuestra aplicación

Dentro de las opciones que podemos realizar en Windows Azure se encuentran las siguientes subcategorías:

- General
 - Panel: donde encontraremos nuestros diferentes proyectos.
 - Grupo de recursos: el recurso donde se encuentra nuestra app
 - Todos los recursos: muestra los diferentes recursos
 - Suscripciones: Nos muestra nuestro tipo de cuenta
 - Facturación: Lo que nos cuesta el mantenimiento de la app
 - Ayuda y soporte técnico

- Compute
 - Máquinas virtuales: se necesita actualizar a cuenta de pago
 - Conjuntos de escalas de máquina virtual
 - Servicios de contenedor
 - Cuentas de Batch
 - Clúster de Service Fabric
 - Registro de contenedor
 - Conjuntos de disponibilidad
 - Discos
 - Instantáneas
 - Imágenes
 - Aplicaciones administradas
- Redes
 - Redes virtuales
 - Equilibradores de carga
 - Puertas de enlace aplicaciones
 - Puertas de enlace de red virtual
 - Puertas de enlace de red local
 - Zona DNS
 - Tablas de rutas
 - Perfiles de CDN
 - Perfiles del Administrador de tráfico
 - Circuitos ExpressRoute
 - Network Watcher
 - Grupos de seguridad de red
 - Interfaces de red
 - Direcciones IP públicas
 - Conexiones
- Almacenamiento
 - Cuentas de almacenamiento:
 - Administradores de dispositivos
 - Almacenes de Recovery Services
 - Trabajos de importación o exportación

- Web y Móvil
 - App Services
 - Logic Apps
 - Perfiles de CDN
 - Servicios multimedia
 - Servicios de búsqueda
 - Mobile Engagement
 - Servicios API Management
 - Notificación Hubs
 - Espacio de nombre del centro de notificaciones
 - Puertas de enlace de datos locales
 - Cuentas de enlace de datos locales
 - Cuentas de integración
 - Planes de App Service
 - Conexiones de API
 - Certificados de App Service
- Base de Datos
 - SQL Database: Aquí encontramos nuestra BD
 - Almacenes de SQL Data Warehouse
 - Bases de datos de SQL Server Stretch Database
 - Azure Cosmos DB
 - Cachés en Redis
 - Base de datos de Azure para servidores MySQL
 - Base de datos de Azure para servidores PostgreSQL
 - Grupos elásticos de SQL
 - Servidores SQL Server



- Inteligencia y análisis
 - Clústeres de HDInsight
 - Áreas de trabajo de aprendizaje automático
 - Trabajos de Stream Analytics
 - Cognitive Services
 - Factorías de datos
 - Colecciones de áreas de trabajo de Power BI
 - Analysis Services
 - Catálogo de datos
 - Customer Insights
 - Log Analytics
 - Planes de servicio web Machine Learning
 - Servicios web Machine Learning
- Internet de las cosas
 - IoT Hub
 - Event Hubs
 - Tranajos de Stream Analytics
 - Áreas de trabajo de aprendizaje automático
 - Servicios web Machine Learning
- Enterprise Integration
 - Puertas Lógicas
 - Puertas de enlace de datos locales
 - Cuentas de integración
 - Servicios API Management
 - Administradores de dispositivos de StorSimple
 - Factorías de datos
 - Retransmisiones
- Seguridad e identidad
 - Securitu Center
 - Almacenes de claves
 - Azure Active Directory
 - Azure AD B2C
 - Usuarios y grupos
 - Aplicaciones empresariales
 - Registro de aplicaciones
 - Azure AD Connect Health
 - Azure AD Cloud App Discovery

- Herramientas de desarrollo
 - Team Services accounts
 - Team projects
 - DevTest Labs
 - Application Insights
 - Servicios API Management
- Supervisión y administración
 - Supervisar
 - Log Analytics
 - Cuentas de Automation
 - Almacenes de Recovery Service
 - Colecciones de trabajos del Programador
 - Perfiles del Administrador de tráfico
 - Asesor
 - Intune
 - Protección de aplicaciones de Intune
 - Conexiones de Herramientas de administrador
 - Registro de actividad
 - Métricas
 - Registros de diagnósticos
 - Alertas
 - Soluciones
- Complementos
 - Bing Maps API for Enterprise
 - Content Moderator
- Otro
 - Azure AD Domain Services
 - Conjuntos de datos de referencia para datos de serie temporal
 - Cuentas de NoSQL
 - Entornos de datos de serie temporal
 - Estado del servicio
 - Etiquetas
 - Explorador de recursos
 - Instancias de Function App
 - Marketplace
 - Novedades
 - Revientes



Desarrollo de aplicaciones en la nube (cloud computing)

Como podemos ver tenemos muchas opciones para realizar desde Windows Azure, pero desde nuestra cuenta de Imagine no podemos realizar muchas cosas. Solo podemos crear un proyecto, con una base de datos SQL y ver como monitorizamos la aplicación. Para obtener toda la funcionalidad debemos actualizar a la versión de pago.

Como añadido podemos personalizar nuestra barra de la izquierda con los submenús que más nos interesen, para poder acceder a ellos con mayor facilidad.

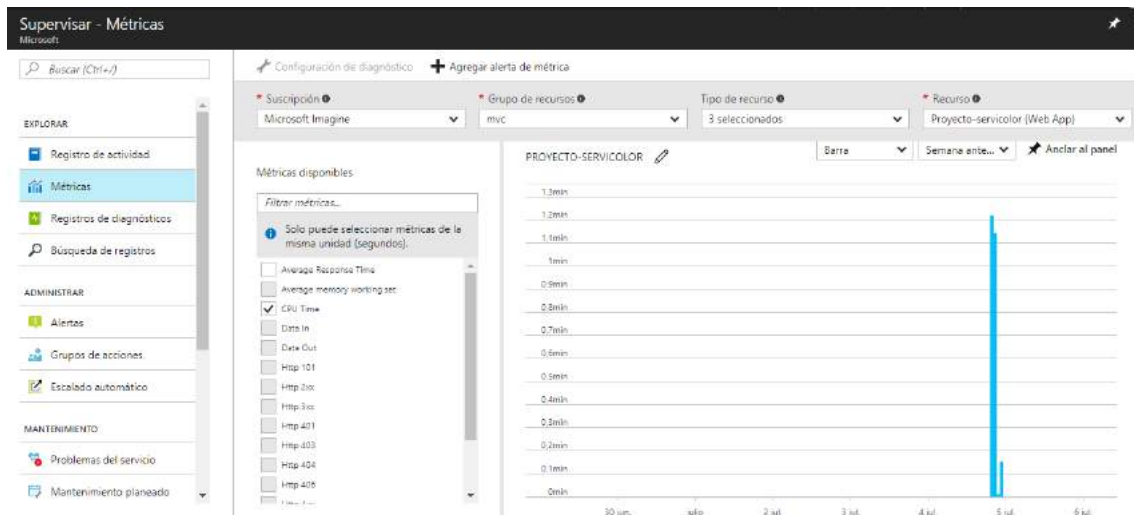


Imagen 57. Supervisión métrica de CPU.

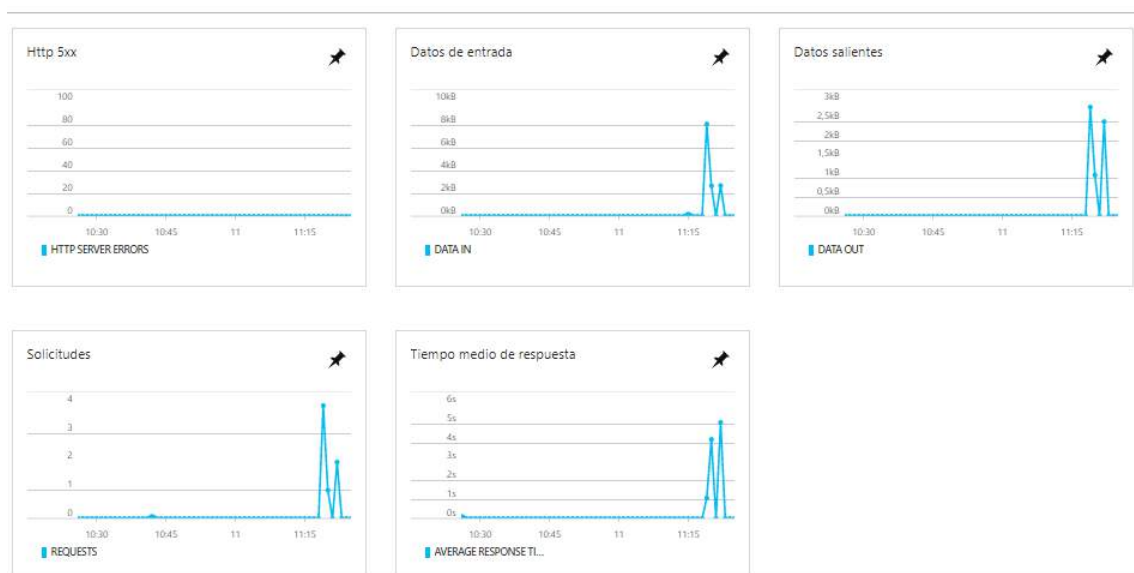


Imagen 58. Datos de entrada, salientes, solicitudes y el tiempo medio respuesta

Ahora vamos a ver todas las opciones que podemos realizar en GAE:

- General
 - Cloud Launcher.
 - Administrador de APIs
 - Facturación
 - Asistencia
 - IAM y administración
- Recursos informáticos
 - App Engine: podemos ver panel control, Memcache, Instancias, versiones de nuestra app, entre otros.
 - Compute Engine: para el uso de máquinas virtuales
 - Container Engine
 - Cloud Functions
 - Redes
- Almacenamiento
 - Bigtable
 - Datastore
 - Storage
 - SQL
 - Spanner
- Stackdriver
 - Supervisión
 - Depurar
 - Trazas
 - Logging
 - Informe de errores
- Herramientas
 - Container Registry
 - Repositorios de código fuente
 - Deployment Manager
 - End Points



- Big Data
 - Big Query
 - Pub/Sub
 - Dataproc
 - Dataflow
 - ML Engine
 - Genomics

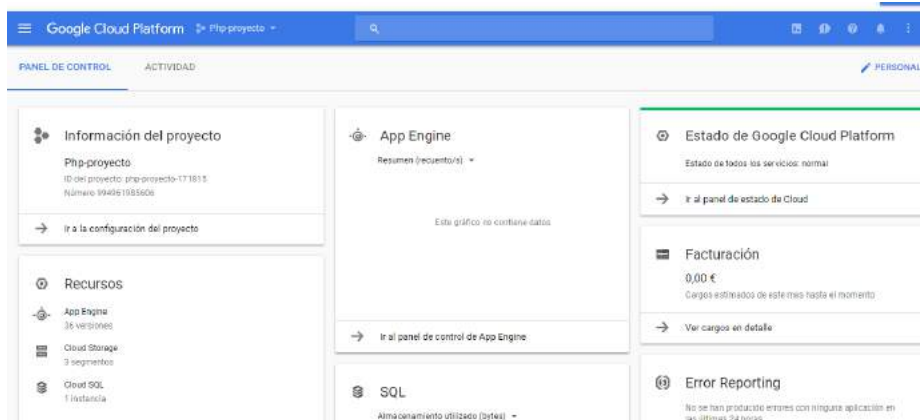


Imagen 59. Panel de control GAE

| App Engine | Versiones | | | | | | | | | |
|------------|------------------------|----------|-----------------------|------------|----------------------|------------------------------|----------|----------------------------------------------|-------------|--|
| | Actualizar | Eliminar | Detener | Iniciar | Migrar tráfico | Mostrar panel de información | | | | |
| | Filtrar las versiones: | | | | | | | | | |
| | Version | Estado | Asignación de tráfico | Instancias | Momento de ejecución | Entorno | Tamaño | Despliegue | Diagnóstico | |
| | 201707011172921 | Serving | 100 % | 0 | php55 | Standard | 127,5 KB | 1 jul. 2017 17:29:32 por vnt.argut@gmail.com | Herramienta | |
| | 201707011171134 | Serving | 0 % | 0 | php55 | Standard | 127,4 KB | 1 jul. 2017 17:11:42 por vnt.argut@gmail.com | Herramienta | |
| | 201707011170825 | Serving | 0 % | 0 | php55 | Standard | 127,4 KB | 1 jul. 2017 17:08:37 por vnt.argut@gmail.com | Herramienta | |
| | 201707011164150 | Serving | 0 % | 0 | php55 | Standard | 127,2 KB | 1 jul. 2017 16:42:00 por vnt.argut@gmail.com | Herramienta | |
| | 201706291191043 | Serving | 0 % | 0 | php55 | Standard | 127,2 KB | 29 jun. 2017 19:10:53 por | Herramienta | |

Imagen 60. Versiones de nuestro proyecto

3.4.1. Conclusiones

Como se puede observar las dos aplicaciones tienen prácticamente la misma funcionalidad, distinguiéndose pocos cambios respecto a una y otra.

- ✓ Los cambios encontrados son:
 - En GAE podemos modificar el mismo código desde el mismo cloud, sin necesidad de utilizar aplicación de terceros.
 - ASP.NET no permite el control de versiones, con lo cual tendremos que implementarlo nosotros mismos, además que en Windows Azure no tenemos control de versiones, a no sé qué lo tengamos implementado en nuestra aplicación.
 - La conexión a nuestra base de datos es mucho más sencilla en Windows Azure que en GAE.
 - En GAE tenemos la opción de guardar datos en un disco duro virtual, cosa que en Windows Azure necesitamos cuenta de suscripción.
 - La supervisión en Windows Azure no necesita cuenta de suscripción, cosa que en GAE si es necesario.

- ✓ En cuanto a rendimiento

La aplicación en Windows Azure es un poco más lenta que en GAE, sobretodo la primera vez que accedemos a nuestra app, pero una vez cargada la aplicación los tiempos de respuesta son muy parecidos.

Por lo general las dos plataformas tienen una muy buena documentación y muchos manuales para empezar a usarlas desde cero. Pero a la hora de publicarlas en su plataforma, en Windows Azure debido a que he usado una aplicación externa que es Visual Studio es mucho más fácil la manera de subir dicha aplicación a la nube, porque solo tenemos que subirla desde la misma aplicación sin necesidad de instalar otras aplicaciones.



Aunque lo más difícil de realizar es que nuestra aplicación local se conecte correctamente a nuestra base de datos que hemos creado en la plataforma correspondiente y en este tema Windows es mejor también, debido a que desde el principio no me ha propuesto ningún problema a la hora de conectarme.

Por último, después de realizar el estudio completo, aconsejaría utilizar Windows Azure, debido a que es un entorno más para principiantes y está todo un poco más claro a la hora de poder publicar nuestras aplicaciones en la nube.

4. Conclusión

El cloud computing como sucede normalmente, presenta un crecimiento gradual. Si bien es cierto que el cómputo en la nube es una tecnología que ya se utiliza desde hace algunos años, aún falta que sea completamente absorbida como una tendencia central en las organizaciones.

El nivel de aceptación entre las organizaciones variará dependiendo del tamaño de éstas. Las medianas y pequeñas empresas ya empiezan a adoptar soluciones basadas en cómputo en la nube, mientras que las grandes organizaciones lo hacen de acuerdo a necesidades particulares.

En la otra cara de la moneda, se encuentran los usuarios finales, a quienes el cómputo en la nube les ha cambiado la forma de realizar sus actividades, mejorando en la mayoría de los casos y permitiéndoles colaborar de una manera distinta con otros usuarios en diferentes lugares, tener acceso a las aplicaciones que requieren desde su navegador web y prácticamente desde cualquier equipo, incluso desde sus dispositivos móviles.

Otro dato importante, es que se han estudiado las plataformas actuales de cloud computing, se ha diseñado e implementados dos aplicaciones cada una con un lenguaje de programación distinto y además se ha evaluado de manera práctica dos plataformas cloud.

Para finalizar las plataformas actuales para el cómputo en la nube están bastante avanzadas pero en cuanto a facilidades para el usuario final es mucho más fácil utilizar Windows Azure debido a que no requiere mucha instalación, pero GAE una vez está todo instalado es más rápido a la hora de actualizar nuestras aplicaciones.



5. Bibliografía

<http://web.archive.org/web/20120915111539/http://www.itnews.ec:80/news/000396.aspx> (Computación en la nube)

<https://revista.seguridad.unam.mx/numero-08/computo-nube-ventajas-desventajas> (PasS, IaaS, SaaS)

<http://www.descom.es/faq/94-cloud-computing/345-diferencia-entre-hosting-tradicional-y-cloud-computing> (Diferencia Hosting y cloud)

<http://www.diariothc.com/diferencias-entre-el-hosting-tradicional-y-el-cloud/> (Diferencia Hosting y cloud)

<https://azure.microsoft.com/es-es/overview/choosing-a-cloud-service-provider/> (Windows Azure)

<https://platzi.com/blog/google-app-engine/> (Google App Engine)

<https://aws.amazon.com/es/documentation/ec2/> (Amazon EC2)

<http://searchdatacenter.techtarget.com/es/consejo/Azure-Vs-Google-Encuentre-al-vendedor-que-mas-se-ajusta-a-usted> (Diferencia entre Windows Azure y Google App Engine)

[https://msdn.microsoft.com/es-es/library/4w3ex9c2\(v=vs.100\).aspx](https://msdn.microsoft.com/es-es/library/4w3ex9c2(v=vs.100).aspx) (Información ASP.NET)

<https://azure.microsoft.com/es-es/documentation/articles/web-sites-dotnet-get-started/> (Tutorial Windows Azure ASP.NET)

<http://docs.asp.net/en/latest/tutorials/your-first-aspnet-application.html> (Tutorial Windows Azure ASP.NET con base de datos)

<https://docs.microsoft.com/es-es/azure/vs-azure-tools-cloud-service-publish-set-up-required-services-in-visual-studio> (Publicar en Windows Azure)

<http://es.slideshare.net/alfredovela/nube-conceptos-usos-y-aplicaciones> (Ejemplos aplicaciones nube)

<http://php.net/manual/es/intro-whatism.php> (PHP)

<https://desarrolloweb.com/articulos/phpmailer.html> (PHPMailer)

<https://developers.google.com/google-apps/calendar/quickstart/php> (Google Calendar API)

Citación de páginas web

1. http://www.prospecnet.com/A_LaNube.aspx (Computación en la nube)
2. <https://www.descom.es/blog/cloud/diferencia-entre-hosting-tradicional-y-cloud-computing.html> (diferencia entre hosting y cloud)
3. <http://www.apser.es/blog/2015/11/25/comparativa-amazon-web-services-vs-microsoft-azure-vs-google-cloud-platform/> (Comparativa de los tres proveedores de servicio más importantes)

