UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

NutriMenu

Presentado por Álvaro Manjón Vara
en Universidad de Burgos — 4 de enero de 2024
Tutores: Raúl Marticorena Sánchez y Antonio Jesús Canepa Oneto

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

D. nombre tutor, profesor del departamento de nombre departamento, área de nombre área.

Expone:

Que el alumno D. Álvaro Manjón Vara, con DNI dni, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado título de TFG.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 4 de enero de 2024

Vº. Bº. del Tutor:	Vº. Bº. del co-tutor:
D. nombre tutor	D. nombre co-tutor

Resumen

En este primer apartado se hace una **breve** presentación del tema que se aborda en el proyecto.

Descriptores

Palabras separadas por comas que identifiquen el contenido del proyecto Ej: servidor web, buscador de vuelos, android ...

Abstract

A **brief** presentation of the topic addressed in the project.

Keywords

keywords separated by commas.

Índice general

Índice de figuras

Índice de tablas

Introducción

Descripción del contenido del trabajo y del estrucutra de la memoria y del resto de materiales entregados.

Objetivos del proyecto

Este apartado explica de forma precisa y concisa cuales son los objetivos que se persiguen con la realización del proyecto. Se puede distinguir entre los objetivos marcados por los requisitos del software a construir y los objetivos de carácter técnico que plantea a la hora de llevar a la práctica el proyecto.

Conceptos teóricos

En aquellos proyectos que necesiten para su comprensión y desarrollo de unos conceptos teóricos de una determinada materia o de un determinado dominio de conocimiento, debe existir un apartado que sintetice dichos conceptos.

Algunos conceptos teóricos de LaTeX[1].

3.1. Secciones

Las secciones se incluyen con el comando section.

Subsecciones

Además de secciones tenemos subsecciones.

Subsubsecciones

Y subsecciones.

3.2. Referencias

Las referencias se incluyen en el texto usando cite [**?**]. Para citar webs, artículos o libros [**?**].

[1] Créditos a los proyectos de Álvaro López Cantero: Configurador de Presupuestos y Roberto Izquierdo Amo: PLQuiz

3.3. Imágenes

Se pueden incluir imágenes con los comandos standard de LaTeX, pero esta plantilla dispone de comandos propios como por ejemplo el siguiente:

Figura 3.1: Autómata para una expresión vacía

3.4. Listas de items

Existen tres posibilidades:

- primer item.
- segundo item.

1. primer item.
2. segundo item.

Primer item más información sobre el primer item.

Segundo item más información sobre el segundo item.

-

Herramientas	App AngularJS	API REST	BD	Memoria
HTML5	X			
CSS3	X			
BOOTSTRAP	X			
JavaScript	X			
AngularJS	X			
Bower	X			
PHP		X		
Karma + Jasmine	X			
Slim framework		X		
Idiorm		X		
Composer		X		
JSON	X	X		
PhpStorm	X	X		
MySQL			X	
PhpMyAdmin			X	
Git + BitBucket	X	X	X	X
MikTEX				X
TEXMaker				X
Astah				X
Balsamiq Mockups	X			
VersionOne	X	X	X	X

Tabla 3.1: Herramientas y tecnologías utilizadas en cada parte del proyecto

3.5. Tablas

Igualmente se pueden usar los comandos específicos de LATEXo bien usar alguno de los comandos de la plantilla.

Técnicas y herramientas

4.1. Patrón de diseño Modelo-Vista-Controlador (MVC)

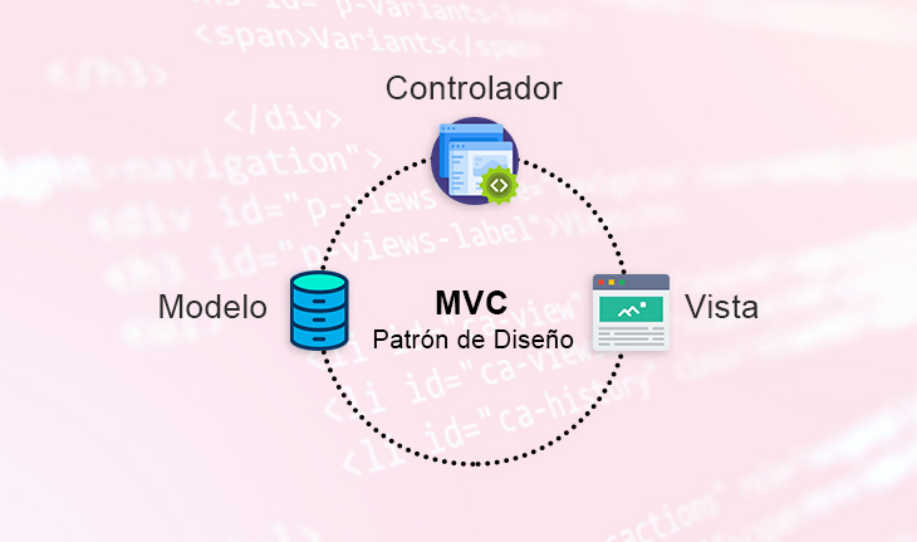

Figura 4.1: Diagrama del patrón Modelo-Vista-Controlador

La función del patrón de diseño conocido como **Modelo-Vista-Controlador** es la de organizar de forma estructurada los componentes fundamentales de un determinado *sistema de software*, estableciendo relaciones entre ellos.

Como metodología de trabajo, fue propuesta por Trygve Reenskaug en 1979, basándose su utilidad en tres ideas fundamentales [1]:

- La organización y clasificación de la información que se aporta al sistema.
- La unificación y gestión de la lógica del sistema.
- La presentación de datos al usuario mediante una interfaz comprensible y asequible.

Los tres componentes fundamentales de este patrón tienen misiones diferenciadas sobre la forma en cómo tratan la información [3]:

- **Modelo**: Es el componente cuya misión es únicamente la de gestionar el contenido de una base de datos, incluida su manipulación y utilización. Es la parte que se va a encargar de la **gestión de los datos**.
- **Vista**: Es el componente que sirve para acceder al contenido de la base de datos y presentar los datos al usuario. Va a ser **la parte con la que va a interactuar el usuario**.
- **Controlador**: Este componente es el que pone en conexión el modelo y la vista, gestionando y procesando las instrucciones que se reciben para obtener un resultado. Su forma básica de trabajar consiste en acceder a la base de datos, manipularlos con una determinada finalidad y presentar los resultados de esa manipulación de los datos. Se va a encargar de gestionar **la lógica y la gestión de los datos**.

El motivo por el cual he decidido utilizar este patrón es porque permite trabajar de una forma precisa y eficaz, haciendo que la estructura del sistema sea clara, ordenada y separada, lo que facilita su comprensión y posterior mantenimiento.

4.2. Contenerización

Se conoce como contenerización a la tecnología de virtualización que permite ejecutar **contenedores**, entornos completamente aislados del resto de la máquina que los está ejecutando, y que contienen todo lo necesario (bibliotecas, código, archivos de configuración, dependencias...) para que se pueda ejecutar una aplicación en cualquier entorno.[2]

A diferencia de entornos clásicos de virtualización como las **máquinas virtuales**, en los que se emula el hardware por completo y se debe disponer

de una instalación completa del sistema operativo para funcionar, en los contenedores se permite que múltiples instancias compartan un mismo sistema operativo, a pesar de estar usando espacios de ejecución distintos. Esto se consigue usando características de Linux como los *espacios de nombres del kernel* o los *cgroups* [7], puesto que los contenedores están basados en Linux, pero esto no impide su ejecución en distintas plataformas y sistemas operativos, incluido Windows.

Docker

Docker es una plataforma de código abierto que permite el desarrollo, ejecución y distribución de aplicaciones en contenedores.

Arquitectura de Docker

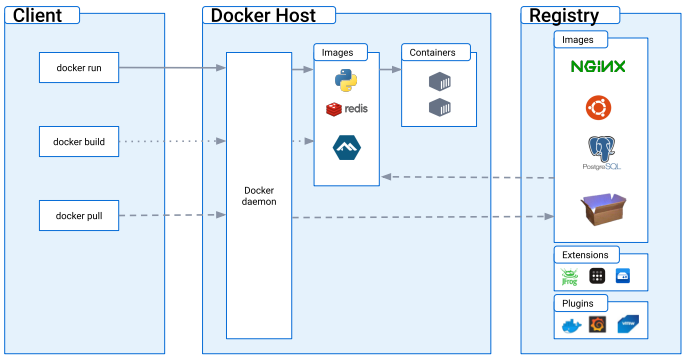

Figura 4.2: Arquitectura de Docker

La arquitectura de Docker es una arquitectura **cliente-servidor** [5]. El cliente de Docker, que es con lo que interactúa el usuario, habla con el *daemon* haciendo llamadas API REST, y este se encarga de gestionar las peticiones y los objetos de Docker, como son las imágenes, los contenedores, los volúmenes y las redes. Finalmente, para conseguir las imágenes, el *daemon* habla con *Docker Registry*, el repositorio que contiene todas las imágenes necesarias para el despliegue. Por defecto se usa **Docker Hub**, un repositorio público en el que cualquiera puede subir y descargar imágenes, pero se puede usar también un repositorio privado.

Imágenes

Las imágenes son plantillas que contienen instrucciones y parámetros para crear contenedores de Docker. Se puede crear un contenedor usando directamente una imagen del *registry*, reutilizando imágenes ya existentes como base para crear nuestras propias imágenes, o incluso creando nuestras propias imágenes desde cero. Las imágenes se crean usando archivos *Dockerfile*, añadiendo en él las distintas dependencias, instrucciones y parámetros, y cada uno de estos elementos es una capa dentro de la imagen. De esta manera, cada vez que se modifica el *Dockerfile* y se vuelve a construir la imagen, realmente sólo se vuelven a construir las capas en las que han habido cambios, acelerando de esta forma el proceso de despliegue. [6]

Docker Compose

Docker Compose es una herramienta que nos permite definir, orquestrar, desplegar y escalar **aplicaciones Docker multi-contenedor**. [4] Para ello, se define un archivo *YAML* en el que se van a definir las configuraciones necesarias para los distintos servicios pertenecientes a la aplicación. Esto nos va a permitir establecer relaciones y dependencias entre distintos contenedores que forman parte de un mismo aplicativo, permitiendo así que varios contenedores formen parte de una misma red o tengan la capacidad de compartir volúmenes de almacenamiento, por poner unos ejemplos.

He decidido usar esta herramienta ya que facilita bastante la conexión e integración entre los distintos servicios que componen la aplicación, como son la base de datos, el backend, y el frontend. Además de esto, a la hora de poner la aplicación en producción, obtenemos varias ventajas respecto a un entorno tradicional:

- **Automatización de despliegues**, ya que una vez hayamos definido en el archivo de configuración de Docker Compose todas las dependencias, configuraciones y relaciones necesarias, con un sólo comando Docker Compose se va a encargar de crear y desplegar los contenedores, configurar los elementos necesarios, e iniciar los servicios.

- **Creación de entornos aislados y consistentes**, lo que nos permite desplegar entornos de desarrollo locales, así como entornos CI/CD de validación y pruebas, y que estos repliquen la infraestructura de producción, que también puede ser desplegada usando Docker Compose.

- **Escalabilidad de infraestructura**, puesto que con un sólo comando podemos aumentar o disminuir el número de instancias de las que dispone un servicio.

Aspectos relevantes del desarrollo del proyecto

Este apartado pretende recoger los aspectos más interesantes del desarrollo del proyecto, comentados por los autores del mismo. Debe incluir desde la exposición del ciclo de vida utilizado, hasta los detalles de mayor relevancia de las fases de análisis, diseño e implementación. Se busca que no sea una mera operación de copiar y pegar diagramas y extractos del código fuente, sino que realmente se justifiquen los caminos de solución que se han tomado, especialmente aquellos que no sean triviales. Puede ser el lugar más adecuado para documentar los aspectos más interesantes del diseño y de la implementación, con un mayor hincapié en aspectos tales como el tipo de arquitectura elegido, los índices de las tablas de la base de datos, normalización y desnormalización, distribución en ficheros3, reglas de negocio dentro de las bases de datos (EDVHV GH GDWRV DFWLYDV), aspectos de desarrollo relacionados con el WWW... Este apartado, debe convertirse en el resumen de la experiencia práctica del proyecto, y por sí mismo justifica que la memoria se convierta en un documento útil, fuente de referencia para los autores, los tutores y futuros alumnos.

Trabajos relacionados

Este apartado sería parecido a un estado del arte de una tesis o tesina. En un trabajo final grado no parece obligada su presencia, aunque se puede dejar a juicio del tutor el incluir un pequeño resumen comentado de los trabajos y proyectos ya realizados en el campo del proyecto en curso.

Conclusiones y Líneas de trabajo futuras

Todo proyecto debe incluir las conclusiones que se derivan de su desarrollo. Éstas pueden ser de diferente índole, dependiendo de la tipología del proyecto, pero normalmente van a estar presentes un conjunto de conclusiones relacionadas con los resultados del proyecto y un conjunto de conclusiones técnicas. Además, resulta muy útil realizar un informe crítico indicando cómo se puede mejorar el proyecto, o cómo se puede continuar trabajando en la línea del proyecto realizado.

Bibliografía

[1] José María Aguilar. ¿Qué es el patrón MVC en programación y por qué es útil? https://www.campusmvp.es/recursos/post/que-es-el-patron-mvc-en-programacion-y-por-que-es-util.aspx, Oct 2019. [Internet; descargado 20-septiembre-2023].

[2] Microsoft Azure. ¿Qué es un contenedor? https://azure.microsoft.com/es-es/resources/cloud-computing-dictionary/what-is-a-container/, 2023. [Internet; descargado 11-junio-2023].

[3] Easy App Code. Patrón de diseño MVC. ¿Qué es y cómo puedo utilizarlo? https://www.easyappcode.com/patron-de-diseno-mvc-que-es-y-como-puedo-utilizarlo, Sept 2020. [Internet; descargado 20-septiembre-2023].

[4] Docker Docs. Docker Compose overview. https://docs.docker.com/compose/, 2023. [Internet; descargado 12-junio-2023].

[5] Docker Docs. Docker overview - Docker architecture. https://docs.docker.com/get-started/overview/#docker-architecture, 2023. [Internet; descargado 12-junio-2023].

[6] Docker Docs. Docker overview - Docker objects. https://docs.docker.com/get-started/overview/#docker-objects, 2023. [Internet; descargado 12-junio-2023].

[7] Sascha Grunert. Demystifying Containers - Part I: Kernel Space. https://medium.com/@saschagrunert/demystifying-containers-part-i-kernel-space-2c53d6979504, Mar 2019. [Internet; descargado 11-junio-2023].