MVC Pattern

V 0.01 “MVP” | Rafael Venegas | Assembler School

BRIEFING

Introducción

MVC (Modelo-Vista-Controlador) es un patrón de arquitectura de software, que separa los datos y la lógica de negocio de una aplicación de su representación y el módulo encargado de gestionar los eventos y las comunicaciones.

Algunos frameworks de desarrollo web como AngularJS, Spring, Symfony o Laravel hace uso de este patrón para organizar los recursos de la aplicación. Esto se debe a que es un patrón muy popular y fácil de implementar. Gracias a este patrón conseguimos obtener una arquitectura que permite organizar de forma escalable los recursos de nuestro proyecto.

¿Cuáles son los principales objetivos en este proyecto?

- Entender que es un patrón de arquitectura, para que se usa, y qué ventajas puede aportar al proyecto
- Aprender a implementar el patrón MVC
- Conocer mejor el uso de este patrón de cara a futuras implementaciones en Frameworks de trabajo populares

1. Análisis general

Esta píldora se divide en dos fases principales:

1.1. Investigación

En esta primera fase centrarás los recursos en investigar e interiorizar las siguientes cuestiones:

- ¿Qué es un patrón de arquitectura?
- ¿En qué consiste el patrón MVC?
- Dibuja un esquema donde se entienda que es el patrón MVC de forma clara
- Explica en qué casos usarías este patrón
- Describe paso a paso que ocurre en este patrón desde que sea crea una Request hasta que se devuelve una Response (todo el camino)
- ¿Qué ventajas crees que tiene usar este patrón?

1.2. Desarrollo

Una vez realizada la fase de investigación, deberás implementar los conocimientos aprendidos en una parte práctica en la que se deberán cumplir los siguientes requisitos:

- Implementar el patrón MVC mediante PHP sin ninguna librería
- Debes de tener una estructura de directorios clara para poder implementar el patrón

- Debes de tener un mínimo de:
  - 2 controladores
  - 2 modelos de datos que obtengan información de una base de datos (puedes usar **MySQL** o un fichero **local**)
  - Deberás adjuntar en el repositorio de tu proyecto una copia de la base de datos con la información necesaria para realizar las pruebas.
  - 2 vistas
  - 2 URL's que demuestren que se implementa el patrón de forma correcta y sean capaces de mostrar los datos obtenidos de la fuente en una vista.
    - Las URL's se podrán controlar por:
    - **Parámetros en la URL**
    - **mod_rewrite** (has de tener en cuenta que esta opción requiere el uso del fichero ".htaccess"

IMPORTANTE:

- No se valora el diseño
- No se tiene en cuenta la temática
- Se valora el orden y la organización de los ficheros y directorios
- Se requiere el uso de la programación orientada objetos
- Se recomienda ver ejemplos de MVC en php para asimilar el concepto
- Céntrate en la implementación del patrón y no tanto en cómo obtener o pintar los datos
- Si usas MySQL (u otro motor de base de datos) no dediques demasiado tiempo al diseño de la base de datos, por ahora NO es importante, lo importante es entender el patrón y el recorrido de una request desde que se crea hasta que obtiene una respuesta.

2. Organización del proyecto

A continuación, deberás crear un documento donde se exponga detalladamente de qué forma se organiza la píldora actual. Es importante que se actualice durante toda la vida de la píldora. El documento debe de contemplar como mínimo:

- Documentación de requisitos.
- Lista de tareas a realizar.
- Prioridad de cada tarea
- Título y descripción de cada una de ellas
- Nivel de dificultad
- Estimación de tiempo para cada tarea.
- Registro de incidencias que se han detectado durante la ejecución de la píldora.
- Documentación acerca del WORKFLOW de git que vas a usar
- Documentación acerca de las herramientas usadas en el proyecto
- Registro de lecciones aprendidas.

3. Requisitos

- Crear una estructura de directorios clara y ordenada
- La página principal deberá mostrar los links de las URL's con las que podrás navegar al resto de controladores
- Deberás crear un controlador principal que se encargue de recibir la petición de la página principal
- Tanto el código como los comentarios tienen que estar escritos en inglés
- Usa el estilo de código camelCase para la definición de variables y funciones
- En el caso de estar usando HTML nunca uses estilos en línea
- Recuerda que es importante dividir las tareas en varias sub-tareas para que de esta forma puedas asociar cada paso en particular de la construcción con un commit específico
- Para la documentación del proyecto es requerida una versión en PDF dentro del repositorio
- Debes intentar en la medida de lo posible, cada commit esté asociado a una tarea
- Borra los ficheros que no se usen o no sean necesarios para evaluar el proyecto
- Deberás crear un fichero README correctamente documentado en el directorio raíz del proyecto (ver pautas en Recursos)

4. Desarrollo

A continuación, deberás ponerte manos a la obra y desarrollar la píldora, **es importante que revises los requisitos y cumplas con todos ellos**.

5. Entregables

Para evaluar el proyecto serán necesarios los siguientes entregables:

- Repositorio con el código
- Documento PDF incluyendo además la parte teórica:
- Esquema gráfico que explique el patrón MVC
- Respuesta a las dudas
- Información que consideres importante
- ¿Te ha resultado útil conocer este patrón?

Investigación

¿Qué es un patrón de arquitectura?

> Un patrón arquitectónico es una solución general y reutilizable a un problema común en la arquitectura de software dentro de un contexto dado. Los patrones arquitectónicos son similares al patrón de diseño de software, pero tienen un alcance más amplio.

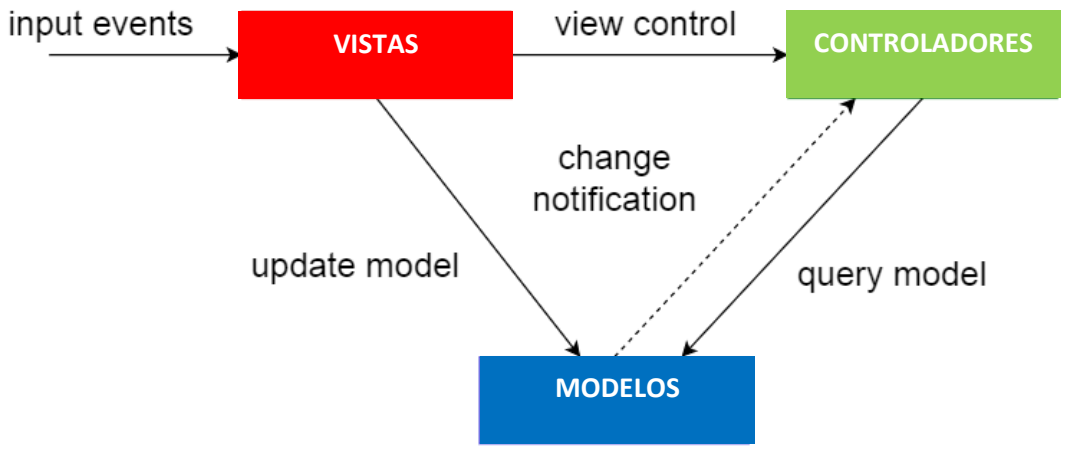
¿En qué consiste el patrón MVC?

El patrón modelo-vista-controlador divide una aplicación interactiva en 3 partes.

1. **El modelo:** contiene la funcionalidad y los datos básicos
2. **La vista**: muestra la información al usuario (se puede definir más de una vista)
3. **El controlador**: maneja la entrada del usuario

Esto se hace para separar las representaciones internas de información de las formas en que se presenta y acepta la información del usuario. Desacopla los componentes y permite la reutilización eficiente del código.

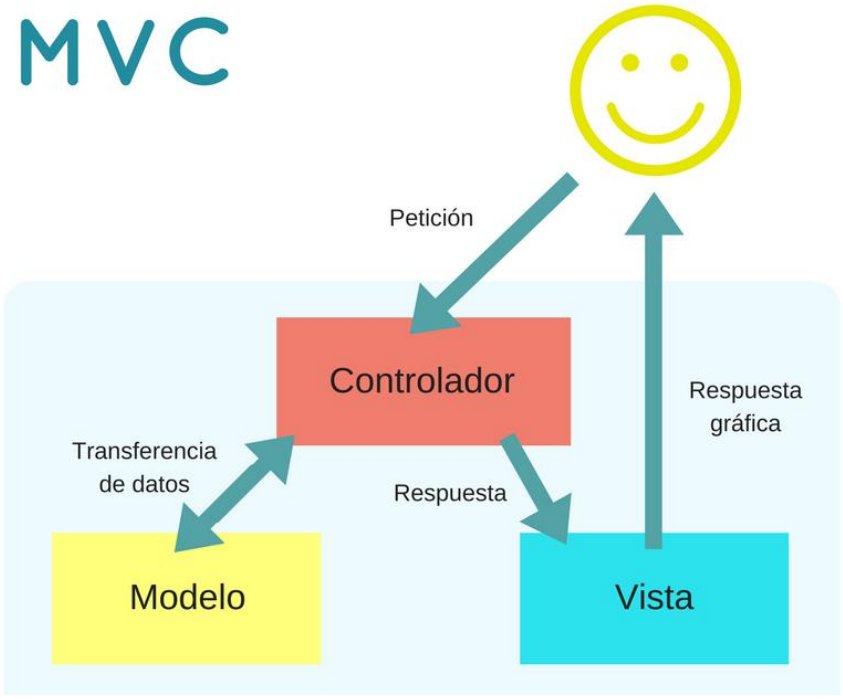
Dibuja un esquema donde se entienda que es el patrón MVC de forma clara



Explica en qué casos usarías este patrón

- ✓ Arquitectura para aplicaciones World Wide Web en los principales lenguajes de programación.
- ✓ Marcos web como Django y Rails.
- ✓ Si la aplicación tiene que mantenerse a largo plazo, es preferible usar MVC. Aunque la primera versión tarde un poco más en desarrollarse, luego será más fácil de "testear" y de comprender ordenádamente la lógica de la aplicación.

Describe paso a paso que ocurre en este patrón desde que sea crea una Request hasta que se devuelve una Response (todo el camino)



¿Qué ventajas crees que tiene usar este patrón?

- ✓ Dividir la lógica de negocio del diseño, haciendo el proyecto más escalable.
- ✓ Facilidad para el uso de URL amigables, importantes para el SEO (Posicionamiento web).
- ✓ Puedes utilizar abstracción de datos, como lo hace Ruby on Rails o con frameworks como Hibernate para Java o NHibernate para ASP .NET MVC, facilitando la realización de consultas a la base de datos.
- ✓ Mayor velocidad de desarrollo en equipo
- ✓ Múltiples vistas a partir del mismo modelo, pudiendo reaprovechar mucho mejor los desarrollos y asegurando consistencia entre ellas.
- ✓ Facilidad para realización de pruebas unitarias.

Tabla de organización del proyecto

	Descripción	*Prioridad*	*Dificultad*	*Tiempo estimado*
Análisis de los requisitos del cliente	Lectura y discusión del briefing del proyecto	Media	Baja	1 hora
Búsqueda previa de información	Aprendizaje a través de tutoriales y ejemplos	Media	Alta	1 hora
Organización inicial	Discusión del proceso a seguir para llevar a cabo el proyecto	Alta	Media	30 min
Documentación	Creación del archivo de la documentación	Media	Media	2 horas
Creación del repositorio	Creación del repo en Github	Baja	Baja	10 min
Creación de los folders y archivos generales	Creación e inicialización de los archivos principales	Baja	Baja	15 min
Configuración del enrutamiento	Creación del archivo de configuración .htaccess	Baja	Baja	10 min
Creación de una versión inicial de los archivos principales (libs)	Desarrollo de un archivo principal para controlador, vista y modelo.	Media	Media	1 hora
Creación de una versión inicial de archivos secundarios	Desarrollo de modelos, vistas y controladores a utilizar en el proyecto	Alta	Alta	2 horas
Prueba de interactividad entre los libs y los archivos secundarios	Pruebas con los archivos principales M-V-C con los secundarios	Media	Media	30 min
Prueba de la interactividad de	Prueba de los controladores con las vistas y los models (aún	Media	Media	1 hora

archivos secundarios	sin conexión a la base de datos)			
Creación de los métodos	Creación, configuración y prueba de la interactividad de todos los métodos	Media	Alta	30 min
Configuración de la base de datos	Creación de la Base de datos y querys que se utilizarán en los models.	Baja	Media	30 min
Configuración de la conectividad con la BD	Configuración de los models con la base de datos	Media	Alta	1 hora
Prueba de interactividad M-V-C secundarios	Prueba de los controladores con las vistas y models (con acceso a la BD)	Media	Media	2 horas
Testing	Testing en varios exploradores y correcciones	Baja	Baja	1 hora

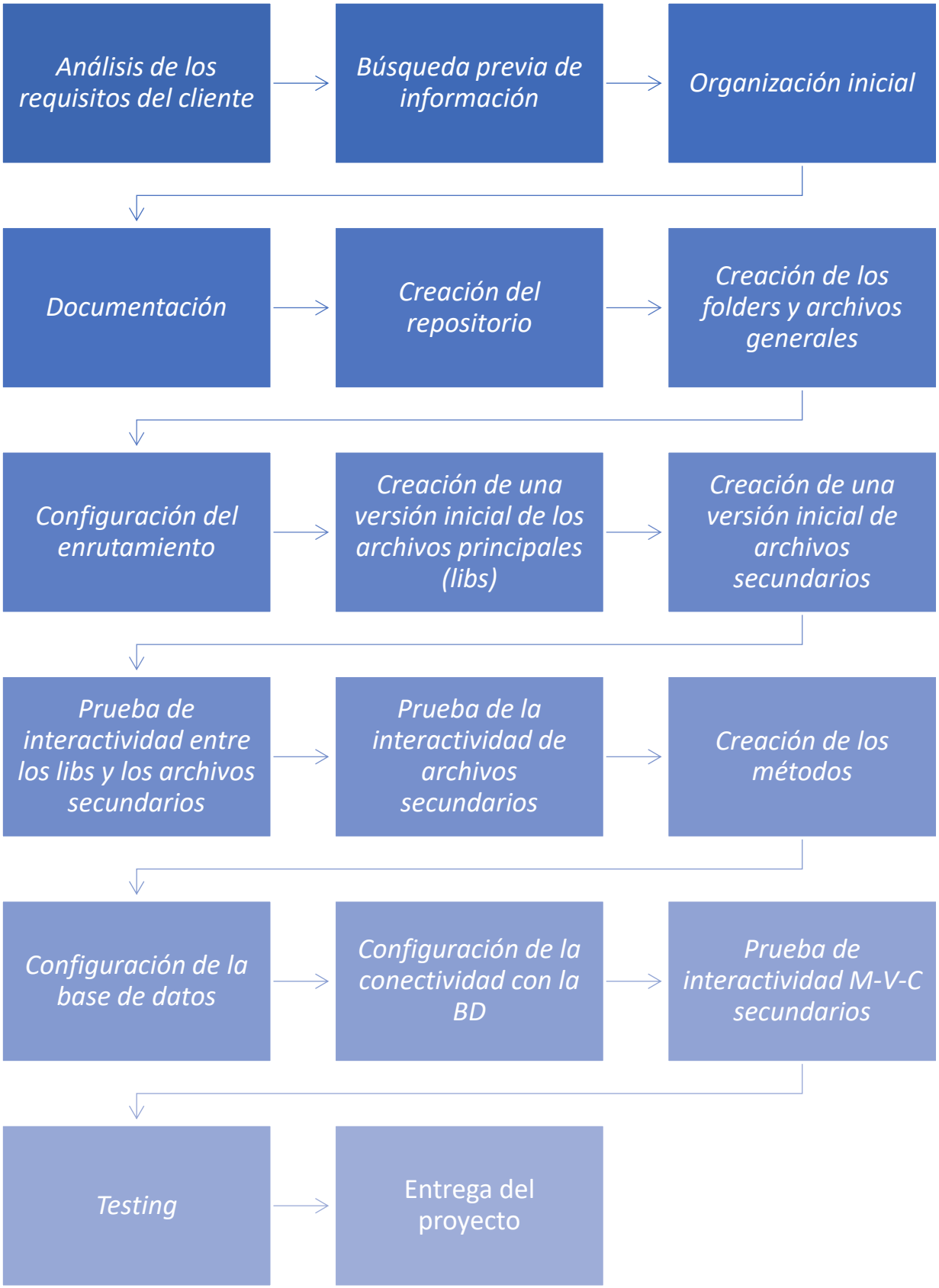
Registro de incidencias que se han detectado durante la ejecución del proyecto.

- Al ya haber realizado un proyecto con el patrón MVC fue relativamente fácil realizar esta pill por lo que no se presentaron incidencias.

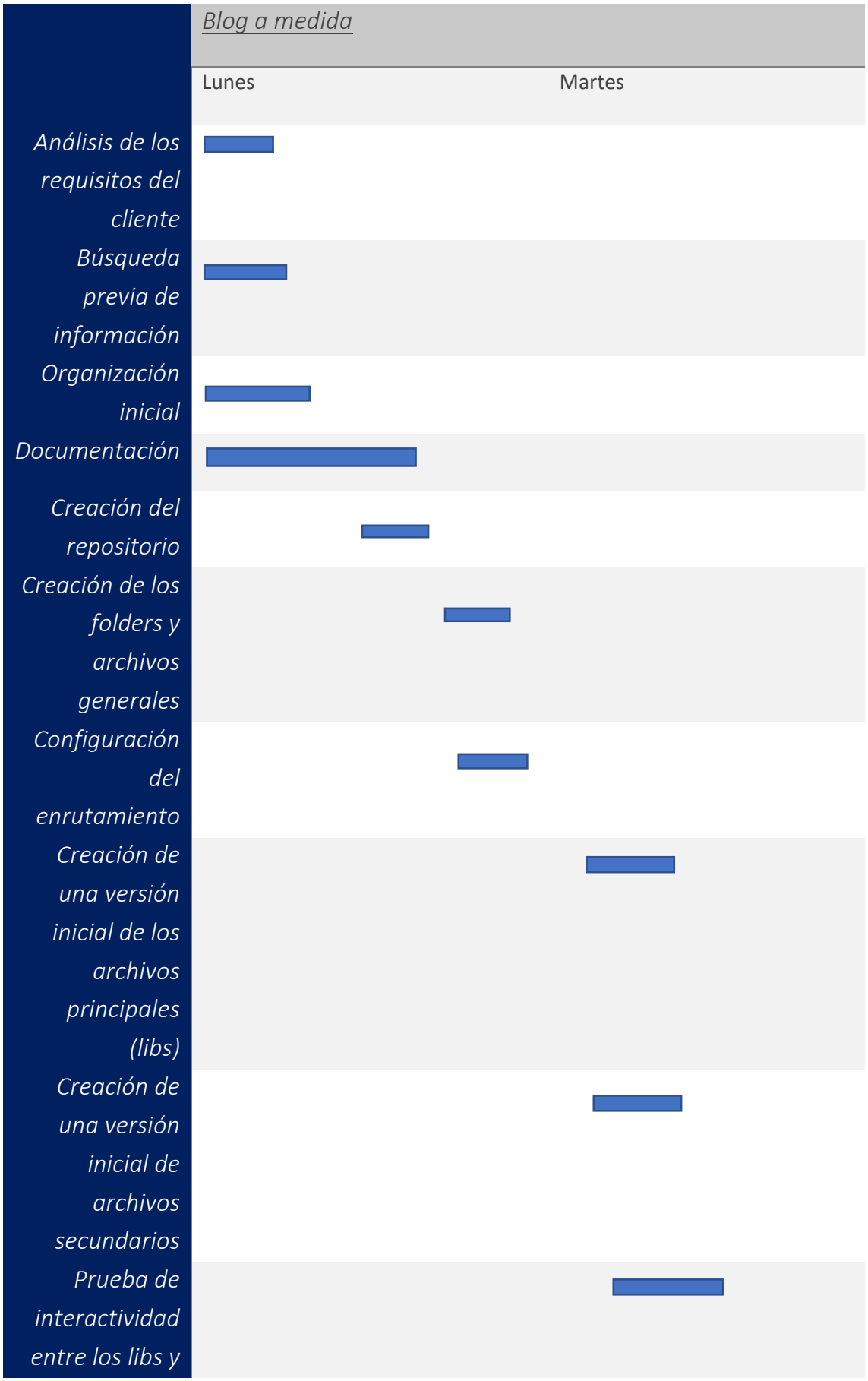
Registro de lecciones aprendidas.

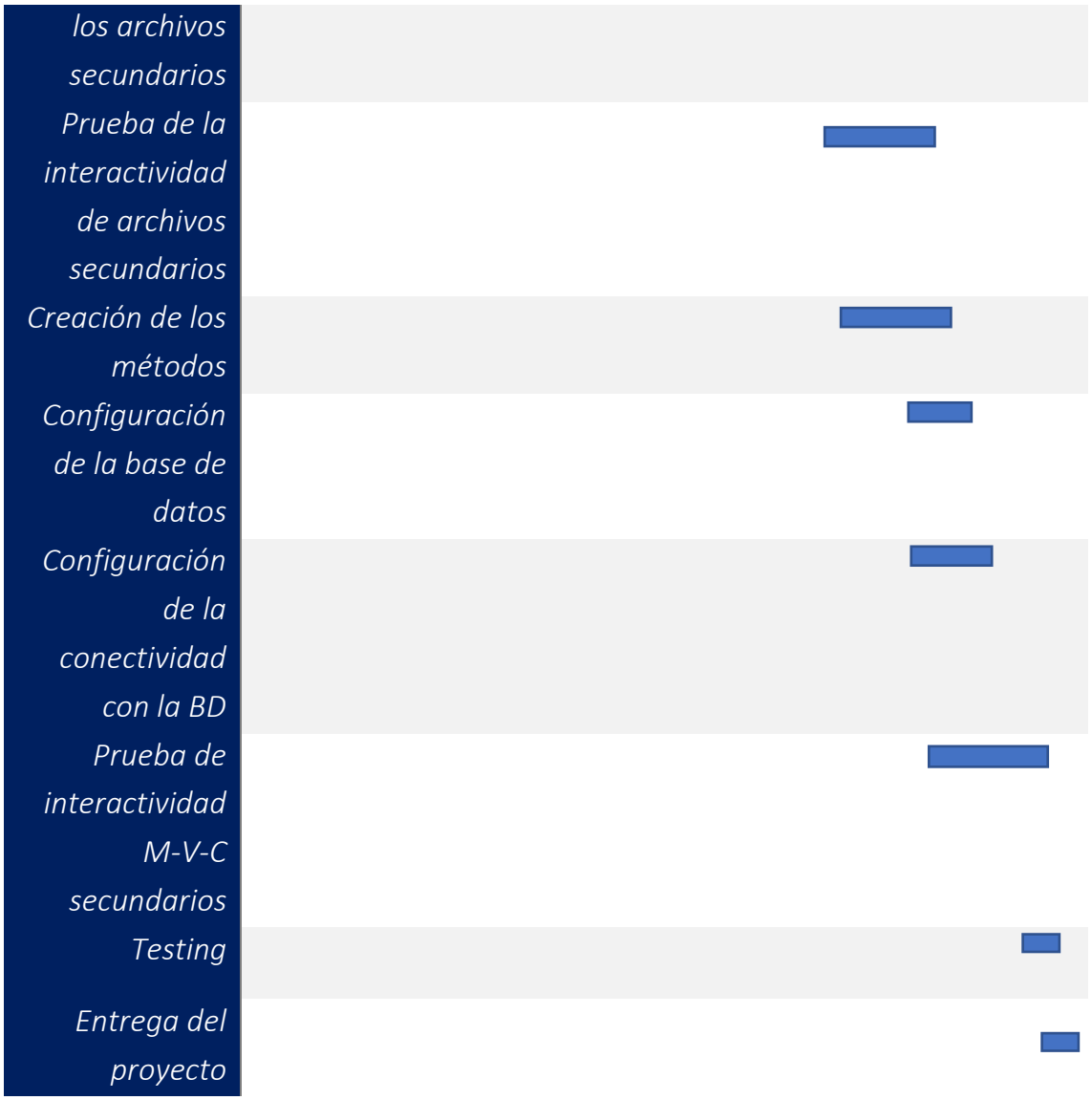
- Trabajar adecuadamente con el modelo-vista-controlador
- Conexiones a la base de datos a través de PHP
- Trabajo con entidades
- Configuración de archivo .htaccess
- Manejo de querys en SQL.

Calendario del proyecto.



Cronograma del proyecto.


Blog a medida
Lunes
Martes
Análisis de los requisitos del cliente
Búsqueda previa de información
Organización inicial
Documentación
Creación del repositorio
Creación de los folders y archivos generales
Configuración del enrutamiento
Creación de una versión inicial de los archivos principales (libs)
Creación de una versión inicial de archivos secundarios
Prueba de interactividad entre los libs y



Estructura de los archivos

Las carpetas se organizaron de la siguiente forma:

blog-a-medida/

- assets/
  - images/ Carpeta que contiene todas las imágenes usadas.
  - css/ Carpeta que contiene todos los estilos usados.
  - documentation/ Carpeta que contiene la documentación
  - databases Carpeta con archivo .sql de la base de datos para pruebas
- config Carpeta con el archivo de configuración de la DB
- controllers Carpeta con todos los controladores
- entities Carpeta con las entidades utilizadas
- libs Carpeta con los archivos primarios
- models Carpeta con todos los modelos
- view Carpeta con todas las vistas

- .htaccess Archivo de configuración general
- index.php Archivo principal que carga todos los controladores

Mediciones de control de calidad.

A través de un checklist, se realizó una verificación de cada uno de los requisitos del cliente y si se cumplían.

Métricas de calidad.

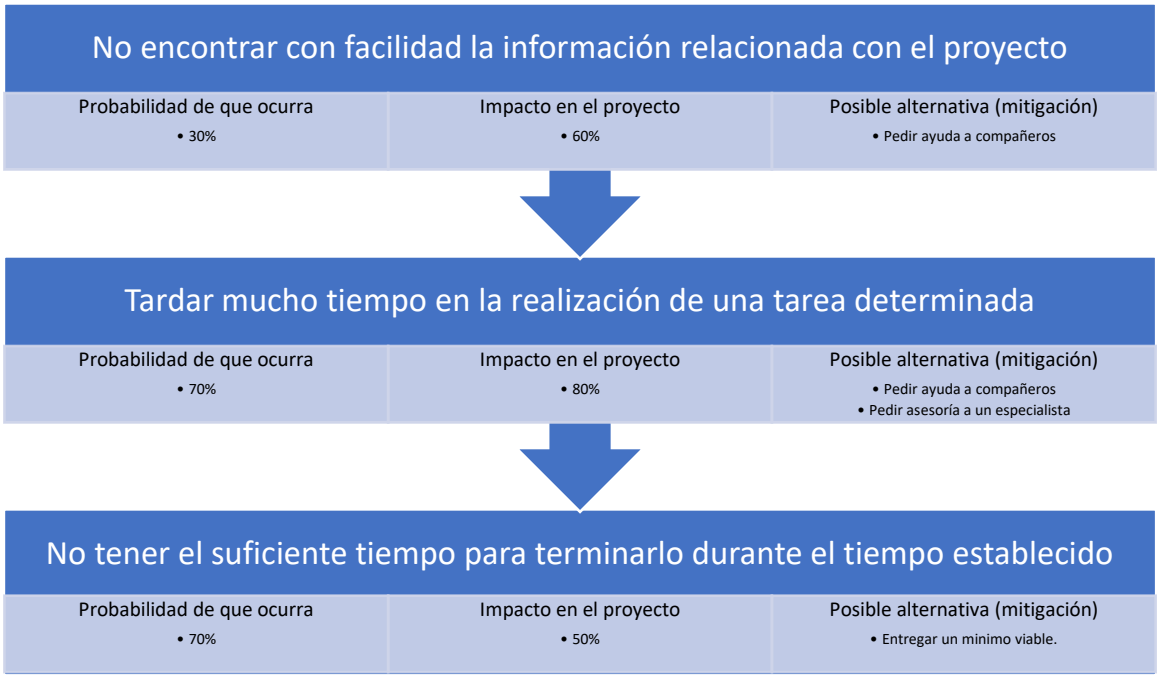
Se verificó individualmente la calidad a través de las siguientes métricas:
- Compatibilidad con los diferentes exploradores
- Obtención de los datos correctamente desde la BD
- Uso adecuado de las carpetas
- Orientación a objetos

Documentación de requisitos.

- Instalación de XAMPP
- Sistema operativo PC: Windows, Mac o Linux. Movil: MAC, IOS.
- En navegadores compatibles (versiones más recientes): Opera, Google Chrome, Firefox, Internet Explorer.

Documentación de riesgos en el caso de que estos existan

En una etapa inicial se plantearon los siguientes riesgos posibles:



Documentación acerca del WORKFLOW de git que vas a usar

- En una etapa inicial, se realizaron pruebas en una rama “pruebas”, en la cual se hicieron commits de la estructura de la página principal.
- A partir de la realización de la estructura se continuó trabajando solamente en la rama “master”, a través del workflow “Gitflow”.
Mas info --> https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow

Documentación acerca de las herramientas usadas en el proyecto

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