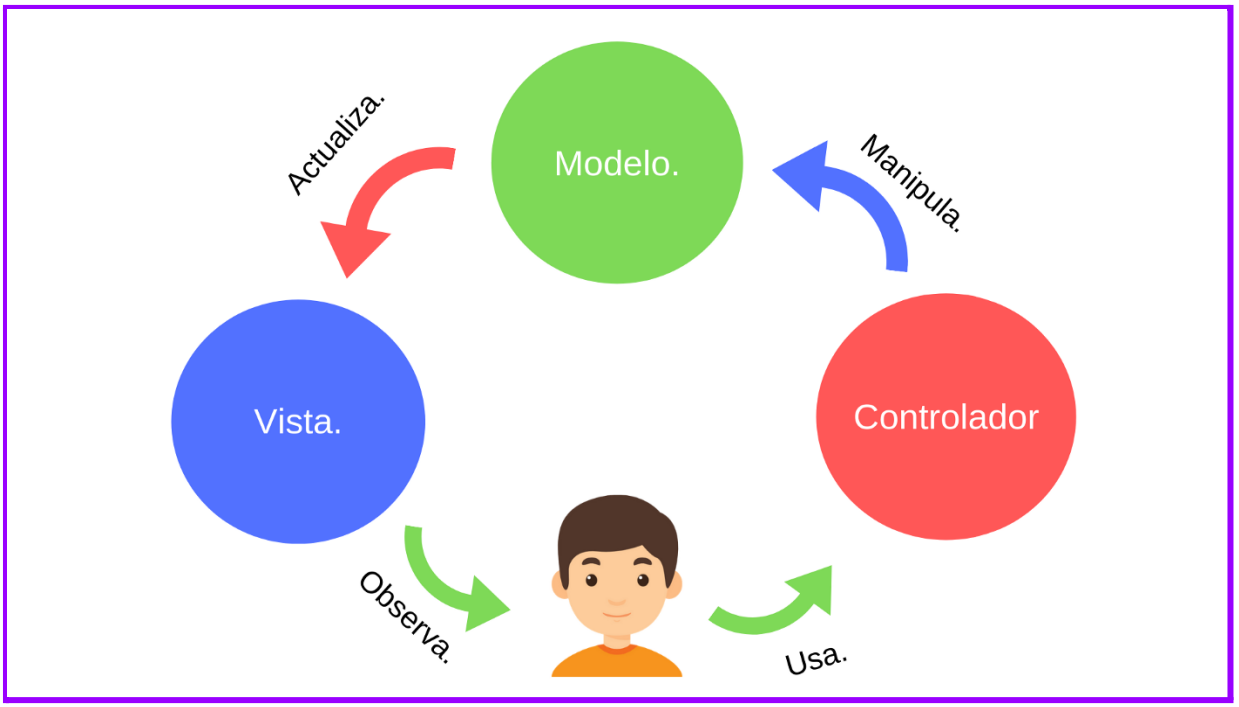
Universidad de La Laguna

Trabajo final: Patrón MVC



Sheyla Ruiz-Gómez Ferreira
Diseño Arquitectónico y Patrones

Universidad de La Laguna

Planteamiento

El objetivo de este proyecto consiste en aprender a utilizar el **patrón Modelo Vista controlador**, el cual es un patrón **de arquitectura de software** que permite separar la lógica de negocio de la interfaz de usuario. Para cumplir dicha meta, se nos solicita:

- Hacer uno del patrón MVC.
- Usar diversos patrones de diseño para facilitar la construcción del patrón MVC.
- Admitir distintos modos de representación gráfica, y al mismo tiempo, poder reusar métodos para las presentaciones similares.

Como material para el desarrollo de la práctica, se usará el proporcionado por el campus de la asignatura y la información externa a la universidad.

Propuesta de diseño

La solución propuesta ha consistido en la creación de 3 paquetes:

➔ **Controlador**: Recibe los eventos de entradas (en este caso, la acción de pulsar los botones), y en adición, dispone de las reglas para gestionar dichos eventos.

Clases que dispone:

- *Factoria*: Clase que simula el **Patrón Factory Method**. Contiene un método estático *create* que se encarga de inicializar el tipo de gráfico que el cliente ha seleccionado.
- *Controlador*: Dispone de un objeto de tipo 'vista' y otro 'modelo'. Según la acción que tomó el usuario, mostrará a través de la ventana el mensaje adecuado e inicializará la nueva vista con los datos nuevos del modelo. Asimismo, tras un tiempo concurrido, se actualizarán los datos (simulando este hecho al **Patrón Observador**).

➔ **Modelo**: Se encarga de acceder a la capa de almacenamiento de datos.

Clases que dispone:

- *Csv*: Dispone de métodos para adquirir la información descargada por la clase Api.
- *Api*: Descarga los datos de forma remota para su posterior utilización. En este caso, se trabaja con datos abiertos de Canarias sobre el **desempleo en las islas**.
- *Conversor*: Clase principal del manejo de los datos. Dispone de los siguientes métodos:
  - *getData*: Tras ser seleccionada la isla de la que se requiere la información, *getData* se encarga de adquirir solo los datos de esa isla.
  - *DesempleoSector*: Adquiere los datos para que posteriormente en la gráfica, solo se muestre el paro por sectores (independientemente de la edad o sexo).
  - *DifSexo*: Consigue los datos para mostrar la diferencia de empleo entre hombres y mujeres (independientemente del sector o edad).
  - *infEdad*: Obtiene los datos para la diferencia de edad por sectores (diferenciando en el sexo).
  - *actualizar*: Actualiza los datos que se consiguen con el acceso remoto.

➔ **Vista**: Responsable de visualizar los gráficos al usuario gracias a los datos obtenidos por el modelo.

Clases que dispone:

- *InterfazVista*: Interfaz que define métodos de visualización. Dicha interfaz, inicia el patrón **Estrategia**.
- *Grafico*: Clase abstracta que inicializa las dimensiones de las ventanas para los gráficos.

- *Ventana*: Dispone de distintos métodos para la visualización de la ventana con la que interactúa el usuario (y los métodos implicados con dicha ventana).
- *Barras*: Realiza un gráfico de barras con los datos de una isla en específico. Los datos que utiliza son el desempleo por sectores.
- *Queso*: Realiza un gráfico en forma de queso, donde compara el desempleo entre hombres y mujeres de una isla.
- *Queso3D*: Realiza un gráfico en forma de queso (en tres dimensiones), donde compara por sectores y edades el desempleo de una isla.

Finalmente, en el **programa principal**, se crea un objeto del modelo, de la vista y del controlador, para a continuación, configurar la vista inicial y arrancar la interfaz.

La estructura resultante sería la siguiente:

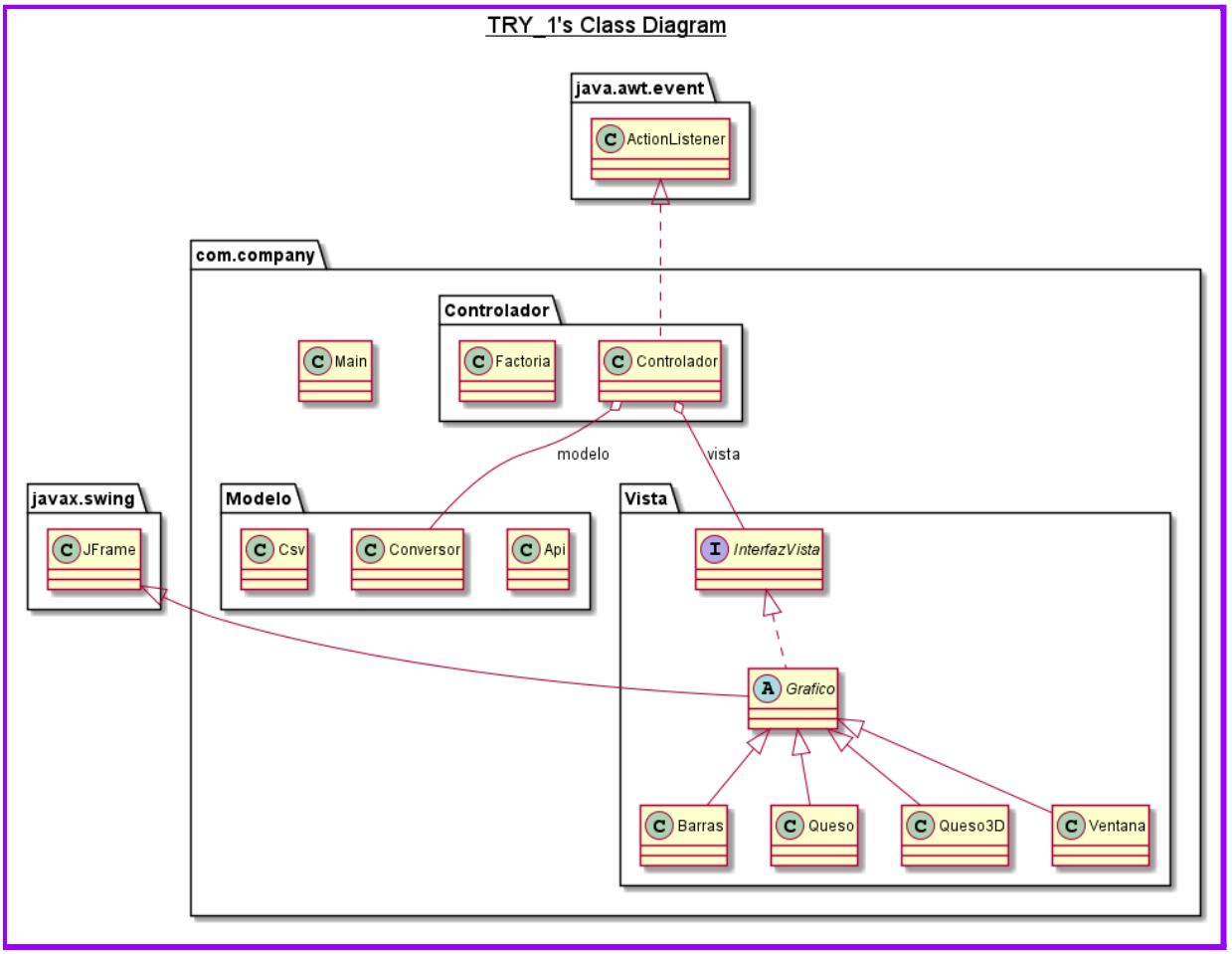


Imagen [1]: Generación de Diagrama de Clase

Patrones empleados

- **Patrón MVC**: Patrón principal de esta práctica.
- **Patrón Observador**: El objeto observado se encuentra en el modelo, y los observadores en la vista. En adición, disponemos un método para actualizar en el controlador.
- **Patrón Estrategia**: Implementada para la parte gráfica.
- **Patrón Factory Method**: Empleada para la iniciación de los gráficos.

Universidad de La Laguna

Plazos de finalización

Tabla con los plazos:

	Tiempo estimado	Tiempo real
Investigación	3h	Continua
Diseño de las clases	3h	+20h
	Implementación	
Desarrollo de las clases	8h	+18h
Mejora del programa	2h	+2h
Arreglo de errores	1h	2h

Principalmente, la mayor parte del tiempo empleado ha consistido en:

- Comprender cómo adaptar el MVC al supuesto del proyecto.

Universidad de La Laguna

Bibliografía

Patrón Observador: https://refactoring.guru/es/design-patterns/observer

Campus virtual:

https://campusingenieriaytecnologia2122.ull.es/mod/assign/view.php?id=14217

API del Gobierno de Canarias sobre el desempleo por sectores: http://www3.gobiernodecanarias.org/empleo/portal/observatorio/estadisticas/Sist_estadistico/OAT/array_dem_sectorc.php?&a=a%3A1%3A%7Bi%3A0%3Bi%3A2021%3B%7D&m=a%3A1%3A%7Bi%3A0%3Bi%3A12%3B%7D&var=2