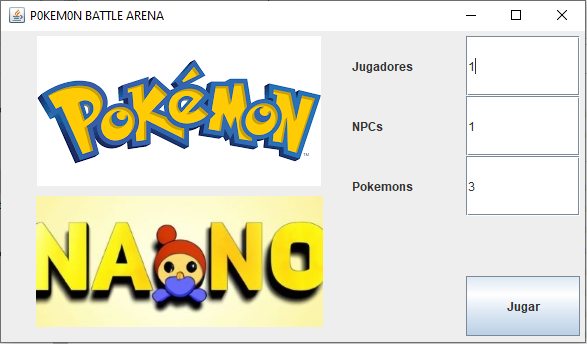


**POKEMON**

Aritz Blasco, Pablo Leclercq, Asier Cardoso, Borja Pecharromán y Janire Veganzones

Grupo “NANO”

[Ingeniería del Software](https://egela.ehu.eus/course/view.php?id=46132)



# ÍNDICE

[Introducción 2](#_a5kkhvz4wog2)

[Gestión 2](#_i4qvw9ygv4i8)

[Reparto de tareas: 2](#_4sxizw5gn2bf)

[Planificación: 3](#_6q5nz9dbbyoo)

[Recursos: 3](#_1jadwahdraqw)

[Diseño 4](#_eyntnrijo1je)

[Diagrama de clases: 4](#_p37k0z2b5jq5)

[Diagrama de secuencias: 4](#_kixs91py8kps)

[MVC y Patrones 4](#_rb9lsu6x2cbw)

[MVC: 4](#_7dt5rvviyihr)

[Patrones: 6](#_ht2qkfwndnip)

[Desarrollo 8](#_wloyuc2oy1y6)

[Sprint 1: 8](#_37v176ntxmcg)

[Sprint 2: 9](#_bodhtmhsdzdy)

[Sprint 3: 9](#_396qx11zygy5)

[Conclusiones 10](#_h7z5rvvdvn69)

# Introducción

Se ha realizado un proyecto en grupo de la manera más realista posible creando un proyecto desde cero del juego “Pokemon”.

El juego del Pokemon consiste en realizar batallas entre diferentes entrenadores, con un mínimo de 1 pokémon. En cada batalla, habrá que atacar a los diferentes pokemons hasta conseguir matar a todos, ya que el primero en matar a todos los pokemons gana.

Alguna de las funcionalidades que hemos implementado son: realizar batallas entre jugadores contra jugadores, jugadores contra bots o jugadores contra jugadores y bots, otra funcionalidad es elegir el numero de pokemons con los que quieres jugar la partida, tener en cuenta el tipo de pokemon que son y con ello los puntos que restan al atacar, y por último hemos implementado la evolución del pokémon y el estado de euforia.

Para la realización de este juego hemos implementado su diagrama de clases (usando patrones) y hemos implementado mediante el diagrama de secuencias la acción más importante del juego la de “atacar”.

Dado que era un trabajo en grupo y que cada uno tenía que implementar su funcionalidad por separado, hemos utilizado “git” para juntar todas las partes y hacerlas todas una.

Los objetivos principales de este proyecto era presentar un trabajo que cumpliese todos los puntos requeridos en el enunciado, es decir presentar una demo sin errores. Para ser más específicos estos eran los objetivos:

* Diseñar y desarrollar el juego en sí. Esto implica la implementación de mecánicas del juego y la creación de la interfaz gráfica.
* A la hora de la implementación, utilizar los patrones necesarios para facilitar su desarrollo.
* Entendernos como equipo, ya que era nuestra primera vez en el mismo grupo de trabajo.

# Gestión

## Reparto de tareas:

* Creación de código:
  + Borja Pecharroman
  + Janire Veganzones
  + Asier Cardoso
* Diseño Diagrama de clases:
  + Pablo Leclercq
  + Aritz Blasco
* Diagrama de secuencias:
  + Borja Pecharroman
  + Janire Veganzones
* Documento Informe MVC:
  + Janire Veganzones
* Documento Informe Patrones:
  + Janire Veganzones
* Documento de planificación:
  + Pablo Leclercq
  + Aritz Blasco
  + Asier Cardoso
* Documento final:
  + Janire Veganzones
  + Borja Pecharroman

## Planificación:

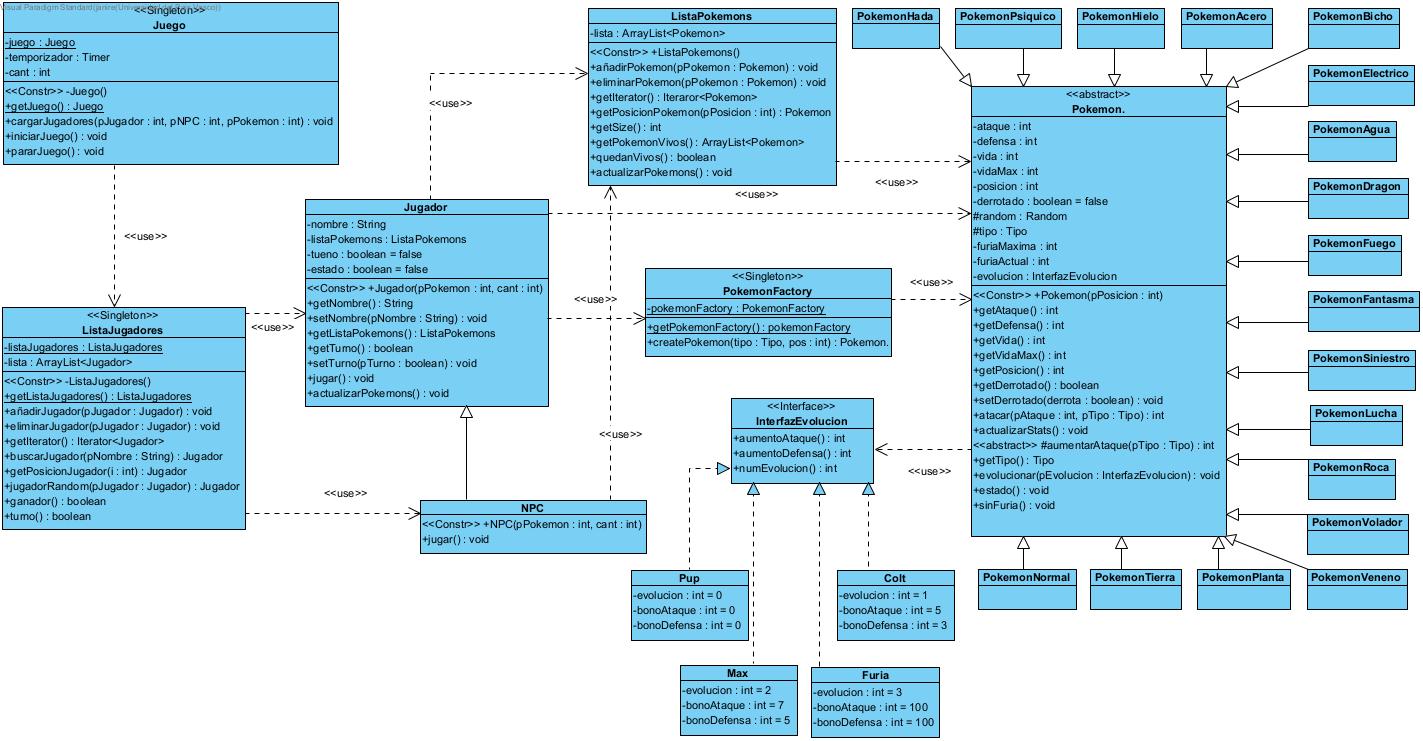
| **Tareas** | **Responsables** | **Horas estimadas** | **Horas trabajadas** |
| --- | --- | --- | --- |
| Creación de código | Borja, Janire y Asier | +25h | 44h |
| Diseño de diagrama de clases | Pablo y Aritz | +3h 30’ | 10h |
| Diagrama de secuencia | Borja y Janire | +2h | 3h |
| Informe MVC | Janire | 40’ | 30’ |
| Informe Patrones | Janire | +1h | 2h |
| Planificación | Pablo, Atriz y Asier | 50’ | 30’ |
| Documentación final | Borja y Janire | 1h | 3h 30’ |

## Recursos:

Para el desarrollo de este proyecto hemos utilizado diferentes herramientas, para la implementación del código hemos utilizado **Eclipse**, a la hora de documentar todo hemos utilizado **Drive,** para la realización de los diagramas tanto de clases como de secuencias se ha utilizado **Visual Paradigm** y por último para comunicarnos y pasarnos el código hemos utilizado **Git**.

# Diseño

## Diagrama de clases:



## Diagrama de secuencia:

El método atacar Pokemon jugador del NPC es una secuencia: primero se ejecuta el método Modelo.NPC.jugar() (imagen 1) que para cada pokemon de su lista y lo pone como atacante, elige un jugador de la lista de jugadores, elige un pokemon de la lista de pokemons del jugador y lo pone como defensor y ejecuta el método Modelo.ControladorPartida.atacar() (imagen 2). A su vez este método comprueba que los atacante y defensores están definidos y ejecuta, entre otros, el método Modelo.Pokemon.atacar(pAtaque, pTipo) (imagen 3) que calcula el daño en función del ataque y el tipo del atacante y la defensa del defensor, después resta ese daño a la vida, comprueba el estado del pokemon y actualiza sus estadísticas.

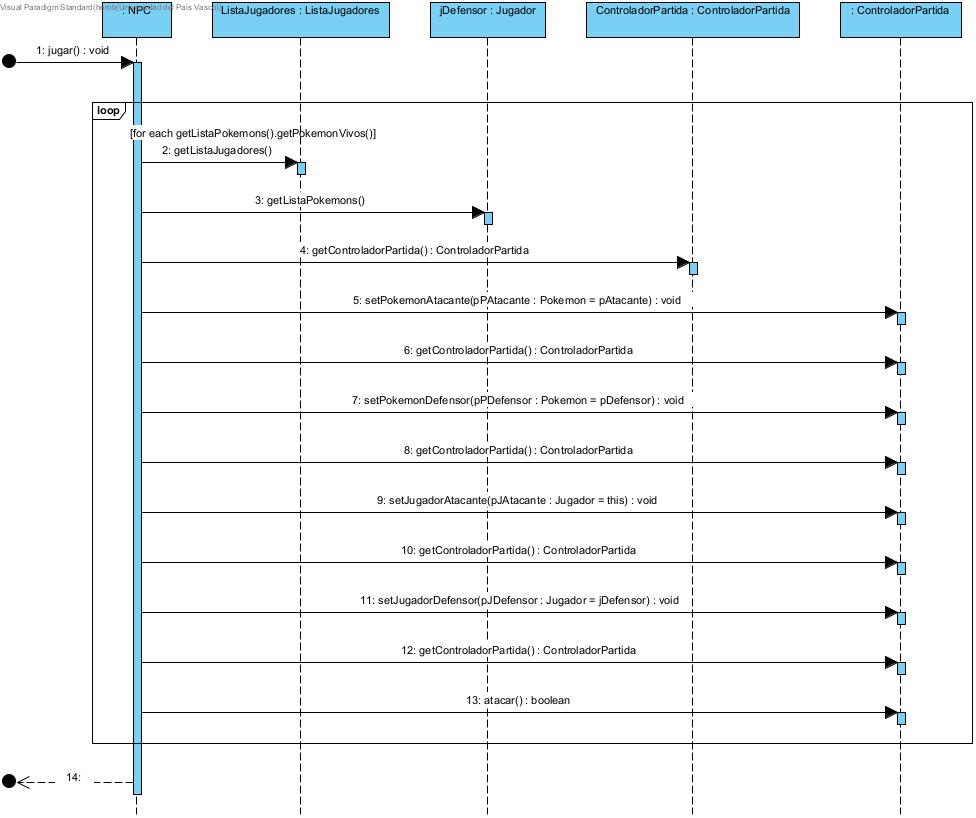


imagen 1

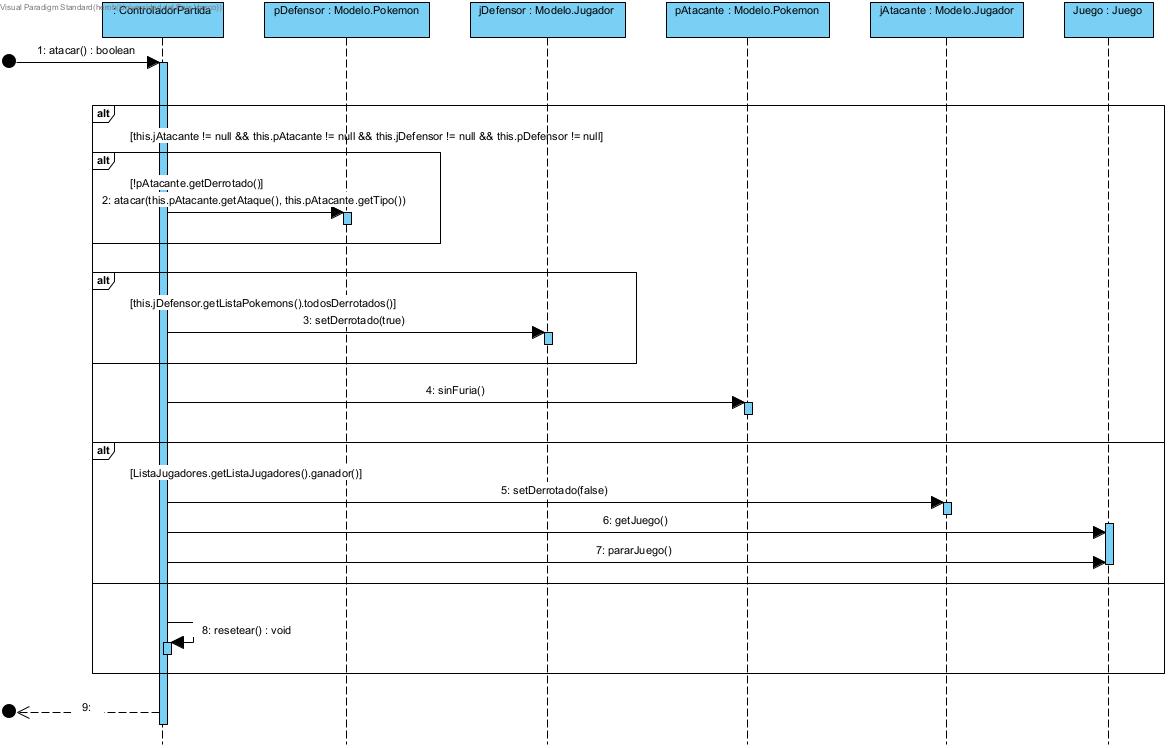


imagen 2

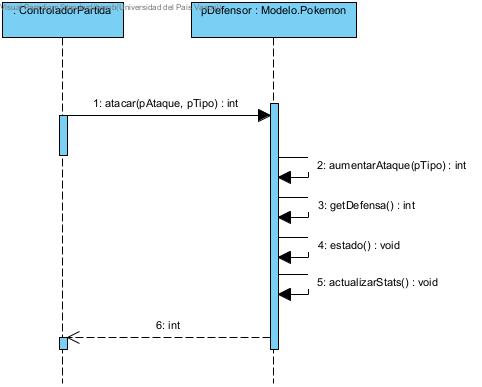


imagen 3

# 

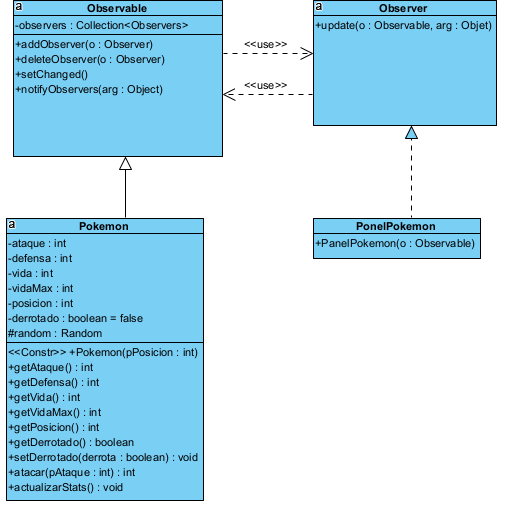
# MVC y Patrones

## MVC:

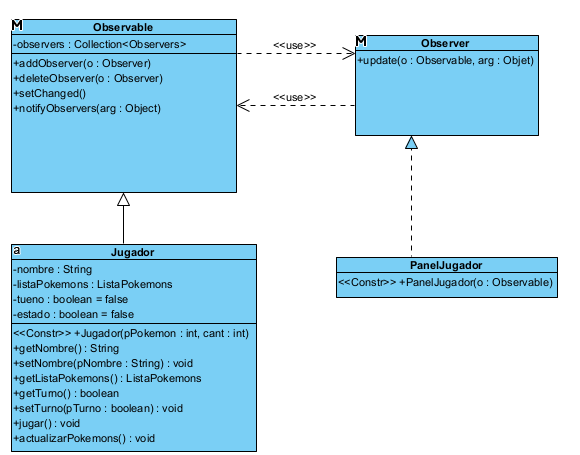
Para cumplir con el patrón Modelo Vista Controlador hemos dividido el proyecto en tres paquetes: ./Vistas, ./Controlador y ./Modelo. En el paquete **Vista** se implementa la interfaz de la aplicación. El paquete **Modelo** es el que almacena la información y trabaja con ella realizando todos los cálculos y acciones que se deben de hacer para poder jugar al Pokemon. Y por último en el paquete **Controlador** que cuenta con la clase controladorPartida que se utiliza para gestionar el juego. Ese controlador lo hemos creado en cada clase de la vista “pantallaPrincipal, PantallaJugador y PantallaPokemon” una clase llamada Controler que implementa a Actionlistener.

También hemos hecho que el modelo no pueda acceder a los elementos de la vista y alterarlos. Para ello lo que hemos hecho es implementar el patrón **Observer** que nos permite notificar a la vista cuando algo en el modelo se ha modificado para que ella misma aplique los cambios sobre sí mismo mediante los métodos setChanged() y notifyObservers(). Estos métodos mencionados anteriormente los tendrá las clases Observables.

**Patrón Observer del Pokemon:**



**Patrón Observer del Jugador:**



## Patrones:

Para la implementación del Pokémon se han utilizado tres patrones diferentes, el Observer, ya mencionado en la entrega anterior, y otros dos nuevos.

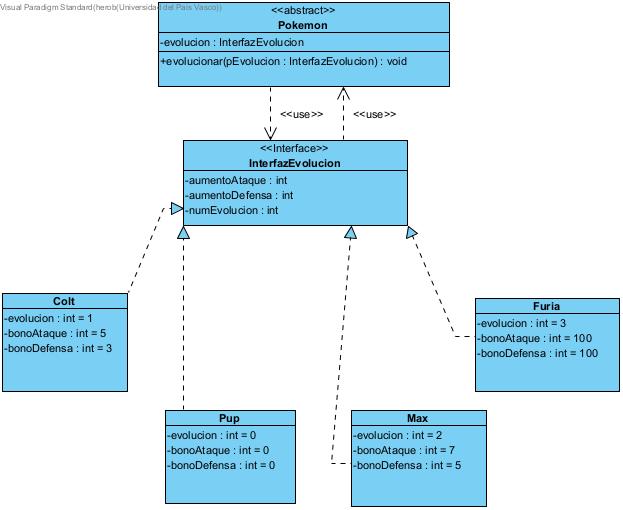
Por un lado tenemos el Factory, este patrón se utiliza para crear objetos de manera dinámica y flexible en situaciones en las que se necesitan crear muchos objetos que tienen implementaciones ligeramente diferentes o que dependen de la configuración o el contexto de la aplicación. Más adelante explicaré como lo hemos implementado en nuestro proyecto.

Y por otro lado tenemos el patrón State, este patrón State se utiliza para modelar comportamientos que cambian en función del estado interno de un objeto. Este patrón se utiliza comúnmente en situaciones en las que se necesitan implementar comportamientos que dependen del estado de un objeto, como en las máquinas de estado finito, los sistemas de control de procesos y los protocolos de comunicación.

**Patrón Factory**

## 

**Patrón State**



# 

# Desarrollo

## Sprint 1:

* **Objetivos:** Los principales objetivos del primer Sprint eran organizarnos de manera que cada uno pudiera realizar la parte que considerara más cómoda y sencilla, adaptándonos a lo solicitado en el enunciado. Por lo que en nuestros objetivos entraba realizar la interfaz gráfica del juego e iniciar la batalla simple.

| **Tarea** | **Responsables** | **Horas estimadas** | **Horas trabajadas** | **Comentario** |
| --- | --- | --- | --- | --- |
| Creación de código | Borja, Janire y Asier | +15h | 28h | Pensábamos que iba a ser más sencillo, pero el método de atacar se nos complicó un poco. |
| Diseño de diagrama de clases | Pablo y Aritz | +2h | 5h | No se ha encontrado ningún problema se realizó y se mostró al profesor y el nos corrigió algunas erratas |
| Informe MVC | Janire | 40’ | 30’ | Al principio hubo algún problema entre el observable y el observador pero mirando los apuntes se resolvió fácilmente |
| Planificación | Pablo, Atriz y Asier | 50’ | 1h | Ningún problema |

## Sprint 2:

* **Objetivos:** En este segundo Sprint al estar perfectamente organizados y ya conocer cómo trabajamos, el principal objetivo era realizar todas las tareas necesarias para cumplir la tarea y aparte adelantar trabajo para el tercer Sprint más corregir los errores que tuvimos en el primer Sprint.

| **Tarea** | **Responsables** | **Horas estimadas** | **Horas trabajadas** | **Comentario** |
| --- | --- | --- | --- | --- |
| Creación de código | Borja, Janire y Asier | +10h | 16h | Típicos problemas que siempre se encuentran a la hora de programar |
| Diseño de diagrama de clases | Pablo y Aritz | +1h | 2h | No se encontró ningún problema pero implementar el patrón Factory nos llevó más tiempo de lo pensado. |
| Informe Patrones | Janire | +1h | 2h | No se encontró ningún problema. |

## Sprint 3:

* **Objetivos:** Al haber adelantado trabajo en el Sprint 2 lo único que nos quedaba por realizar para la entrega final era la documentación final y el diagrama de secuencias del método atacar() y corregir los errores que tuvimos en el 2º Sprint.

| **Tarea** | **Responsables** | **Horas estimadas** | **Horas trabajadas** | **Comentario** |
| --- | --- | --- | --- | --- |
| Documentación Final | Janire y Borja | 1h | 3,5h | No se encontró ningún problema. |
| Diagrama de secuencia | Borja y Janire | +2h | 3h | No se encontró ningún problema. |
| Diseño de diagrama de clases | Pablo y Aritz | 15’ | 30’ | No se encontró ningún problema. |

# 

# Conclusiones

En conclusión, como equipo de desarrollo del juego de Pokémon, estamos muy satisfechos con los resultados obtenidos. Todos los miembros del equipo han participado activamente y han contribuido con su talento y conocimientos para lograr el éxito en cada entrega.

Hemos demostrado una gran capacidad de organización y colaboración, lo cual nos ha permitido cumplir con todas las entregas establecidas en el cronograma. Cada uno de nosotros ha asumido responsabilidades de manera efectiva, ajustándonos a los objetivos principales que establecimos al inicio del proyecto.

Creemos firmemente que hemos realizado un excelente trabajo. Hemos aplicado habilidades de programación avanzadas, diseñado mecánicas de juego interesantes y creado un entorno visualmente atractivo para los jugadores. Además, hemos logrado incorporar las características y elementos esenciales del Pokémon.

Nuestro esfuerzo colectivo, dedicación y compromiso han sido fundamentales para alcanzar los resultados exitosos que hemos obtenido.

En resumen, estamos orgullosos de haber realizado este proyecto de programación del juego de Pokémon en equipo. Hemos demostrado nuestra capacidad de trabajo conjunto, hemos cumplido con las entregas y objetivos establecidos, y hemos creado un juego de calidad que refleja nuestro esfuerzo y dedicación.