

CapSteam.....	2
Template - Project plan.....	3
Template - Decision documentation.....	5
Template - Meeting notes.....	6
Documentación.....	7
Vista de todos los juegos.....	8
Añadir videojuego.....	9
Editar Videojuego.....	10
Eliminar Videojuego.....	11
Vista detallada de un videojuego.....	12
Juegos Siglo XX.....	13
Filtro de videojuegos según su genero.....	14
Juegos Nintendo.....	15
Filtro de videojuegos según su año de publicación.....	16
Modelo de Datos.....	17
Diagrama ER.....	18
Diagrama de clases del modelo.....	19
Untitled whiteboard (1).....	20

CapSteam

Description

In a sentence or two, describe the purpose of this space.

Project Tracker

Recently updated content

i This list below will automatically update each time somebody in your space creates or updates content.

Untitled whiteboard (1)

yesterday at 9:32 AM • contributed by Sergio Salgueiro Monforte

Juegos Nintendo

yesterday at 9:03 AM • contributed by Daniel Martinez Perez

Editar Videojuego

yesterday at 8:20 AM • contributed by Daniel Martinez Perez

Filtro de videojuegos según su año de publicación

Oct 08, 2024 • contributed by Sergio Salgueiro Monforte

Añadir videojuego

Oct 08, 2024 • contributed by Sergio Salgueiro Monforte

Vista de todos los juegos

Oct 08, 2024 • contributed by Sergio Salgueiro Monforte

Vista detallada de un videojuego

Oct 08, 2024 • contributed by Daniel Martinez Perez

Filtro de videojuegos según su genero

Oct 08, 2024 • contributed by Sergio Salgueiro Monforte

Eliminar Videojuego

Oct 08, 2024 • contributed by Sergio Salgueiro Monforte

Juegos Siglo XX

Oct 08, 2024 • contributed by Jesús Alonso García

Contributors

i This list below will automatically update each time somebody in your space creates or updates content.

Sergio Salgueiro Monforte, Daniel Martinez Perez, Jesús Alonso García

Template - Project plan

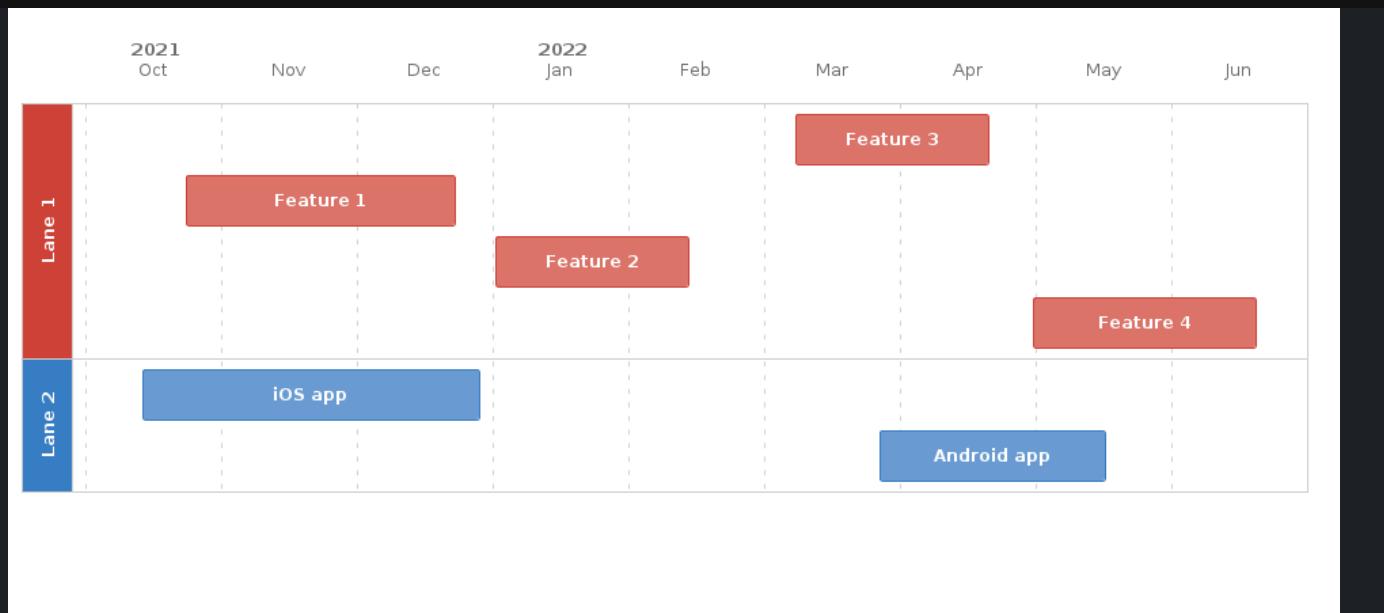
Driver	
Approver	
Contributors	
Informed	
Objective	
Due date	
Key outcomes	
Status	NOT STARTED / IN PROGRESS / COMPLETE

Problem Statement

Scope

Must have:	<ul style="list-style-type: none">••
Nice to have:	<ul style="list-style-type: none">••
Not in scope:	<ul style="list-style-type: none">••

Timeline



▶ Milestones and deadlines

Milestone	Owner	Deadline	Status

🔗 Reference materials



Template - Decision documentation

Status	NOT STARTED / IN PROGRESS / COMPLETE
Impact	HIGH / MEDIUM / LOW
Driver	
Approver	
Contributors	
Informed	
Due date	
Resources	

📘 Background

📊 Relevant data

🌈 Options considered

	Option 1	Option 2
Description		
Pros and cons	+ -	+ -
Estimated cost	LARGE	MEDIUM

✓ Action items

🌟 Outcome



Template - Meeting notes



Date



Participants

-
-



-



Discussion topics

Time	Item	Presenter	Notes
			•



Action items

-



Decisions



Vista de todos los juegos

- Crear estructura principal del proyecto.
- Configurar proyecto con GitHub.
 - Crear repositorio
 - Vincular repositorio con proyecto en eclipse
 - Crear rama main
 - Crear rama develop
 - Crear ramas por tareas
- Configurar conexión a BBDD
- Crear metodo findAll:List<Videojuego> en capa DAO
 - Dar de alta el método findAllVideojuego():List<Videojuego>
 - Hacer algoritmo del método
 - Documentar el método
 - Comprobar las pruebas de calidad
 - Subir al repo
 - Comunicar en Jira.
- Pensar PU Vista de todos los juegos 01
- Crear PU Vista de todos los juegos 01 para método findAll(capa DAO)
 - Probar que al ejecutar el findAll se extraigan elementos, es decir que la lista no esté vacía
- Crear metodo findAllVideojuegos(): List<VideojuegosDto> en capa service
- Crear metodo GET findAllVideojuegos():String con endpoint “/videojuego/findAllVideojuegos“ en capa Controller
- Realizar prototipo página HTML vistaVideojuegos
- Crear html vistaVideojuegos.html
- Asignar esta vista como página principal

Añadir videojuego

- Crear entidad Videojuego
- Crear entidad Editor
- Crear metodo add(Videojuego videojuego):Videojuego en capa DAO
 - Crear interface DAOVideojuego
 - Crear clase DAOVideojuegoImpl
 - Dar de alta el método addVideojuego(Videojuego videojuego): Videojuego
 - Hacer algoritmo del método
 - Documentar el método
 - Comprobar las pruebas de calidad
 - Subir al repo
 - Comunicar en Jira.
- Pensar PUAñadirVideojuego01
- Crear PU AñadirVideojuego01 para método add (capa DAO):
 - Probar a añadir un elemento y verificar que se ha añadido a la bd
 - Probar fallos en todos los datos con varios assert devolviendo una excepcion IllegalArgumentException
- Crear clase VideoJuegoDto
- Crear clase EditorDto
- Crear metodo addVideojuego(VideojuegoDto):VideoJuegoDto en capa service
- Incluir en la página principal un botón añadir
 - Al hacer click redireccionar al método GET addVideojuego del controlador
- Crear metodo GET addVideojuego():String con endpoint "/videojuego/addVideojuego/{todos los datos necesarios}" en capa Controller
- Crear metodo POST addVideojuego():void con endpoint "/videojuego/addVideojuego" en capa Controller
- Realizar prototipo página HTML addVideojuego
- Crear html addVideojuego.html

Editar Videojuego

- Crear método updateVideojuego(Videojuego videojuego) en capa DAO
 - Dar de alta el método updateVideojuego(Videojuego videojuego) : Videojuego
 - Hacer algoritmo del método
 - Documentar el método
 - Comprobar pruebas de calidad
 - Añadir método updateVideojuego en la interfaz DAOVideojuego
 - Implementar el método updateVideojuego en la clase DAOVideojuegoImpl
 - Subir al repositorio
 - Comunicar en Jira
- Pensar PU EditarVideojuego01
- Crear PU EditarVideojuego01 para método updateVideojuego (capa DAO):
 - Probar a editar un elemento correctamente y verificar que se ha editado en la bd
 - Probar fallos en todos los datos con varios assert devolviendo una excepcion IllegalArgumentException
- Crear metodo updateVideojuego(Videojuego videojuego) : Videojuego en capa service
- Incluir en cada item (Videojuego) un botón para actualizar los campos
 - Al hacer click redireccionar al método PUT updateVideojuego(Videojuego videojuego) : String en el Controller
- Crear metodo GET updateVideojuego(Videojuego videojuego) : String para obtener la vista en capa controller.
- Crear metodo PUT updateVideojuego(Videojuego videojuego) : void con endpoint “/videogame” en capa Controller
- Realizar prototipo página HTML edit
- Crear html editVideojuego.html
- Crear error.html

Eliminar Videojuego

- Crear metodo deleteVideojuego(int id):Void en capa DAO
 - Dar de alta el método deleteVideojuego(int id):Void
 - Hacer algoritmo del método
 - Documentar el método
 - Comprobar las pruebas de calidad
 - Subir al repo
 - Comunicar en Jira.
- Pensar PU EliminarVideojuego01
- Crear PU EliminarVideojuego01 para método delete (capa DAO):
 - Probar eliminar correctamente un elemento y verificar que se ha borrado en la bd
 - Si no existe el id que devuelva IllegalArgumentException
- Crear metodo deleteVideojuego(int id):Void en capa service
- Crear metodo DELETE deleteVideojuego(int id):void con endpoint "/videojuego" en capa Controller.
- Añadir un boton de eliminar en cada elemento al listar los videojuegos
 - Al hacer click crear diálogo para preguntar si está seguro de eliminarlo
 - Al aceptar el dialogo redireccionar al método DELETE deleteVideojuego del controlador y redireccionar a la pantalla principal, en caso contrario no hacer nada

Vista detallada de un videojuego

- La vista principal incluirá un enlace para poder ver las vista detallada del videojuego asociado a dicho enlace
- Crear método `vistaDetalleVideojuego(int idVideojuego):Videojuego` en capa DAO
 - Dar de alta el método `vistaDetalleVideojuego(int idVideojuego):Videojuego`
 - Hacer el algoritmo
 - Documentar el método
 - Comprobar las pruebas de calidad
 - Subir al repo
 - Comunicar en Jira.
- Crear PU 01 para método.
 - Caso: Si la Id que se busca no existe entonces se lanzara excepción y se mostrara error
- Crear PU 02 para método.
 - Caso: Si la Id que se busca existe entonces comprobar que los valores son los correctos
- Crear método `vistaDetalle(int idVideojuego): Videojuego` en capa service
- Crear método `vistaDetalle(int idVideojuego): String` con metodo GET “/videojuego/{id}”, siendo id la id del videojuego en capa Controller
- Realizar prototipo página HTML `vistaDetalle`
- Crear HTML `vistaDetalleVideojuego.html`

Juegos Siglo XX

- Crear método juegosSigloXX():List<Videojuego> en capa DAO
 - Dar de alta el método juegosSigloXX():List<Videojuego>
 - Hacer el algoritmo
 - Documentar el método
 - Comprobar las pruebas de calidad
 - Subir al repo
 - Comunicar en Jira.
- Pensar pruebas
- Crear PU JuegosSigloXX01
 - Caso: En un sistema con juegos publicados únicamente en el siglo XXI. No se obtendría ningún juego
- Crear PU JuegosSigloXX02
 - Caso: En un sistema con juegos publicados en el siglo XX y XXI. Solo se obtendrían los juegos de dicho siglo
- Crear método juegosSigloXX(): List<VideoJuegoDTO> en capa Service
- Crear método juegosSigloXX(): String con endpoint GET “/videojuego/juegosSigloXX” en capa Controller
- Reutilizar la plantilla de vistaVideojuegos.html

Filtro de videojuegos según su genero

- Crear Enum Genero en el paquete util
- Crear metodo findByGenero:List<Videojuego> en capa DAO
 - Dar de alta el método findByGenero(Genero genero)
 - Hacer el algoritmo del método
 - Documentar el método
 - Comprobar las pruebas de calidad
 - Subir al repo
 - Comunicar en Jira.
- Pensar PUFiltrarPorGenero 01
- Crear PU FiltrarPorGenero 01 para método findByGenero(capa DAO):
 - Probar que se obtienen videojuegos de un genero
 - Si no hay videojuegos con un genero que devuelva vacio
- Crear metodo findByGenero(Genero genero):List<VideojuegoDto> en capa service
- Crear metodo GET findByGenero(Genero genero) con endpoint "/videojuego/findVideojuegoByGenero" en capa Controller
- Añadir un combobox con todos los generos en la página principal
 - Al seleccionar un valor en el combobox redireccionar al método GET findByGenero del controller
- Reutilizar la plantilla de vistaVideojuegos.html

Juegos Nintendo

- Crear método `findNintendoGames(): List<Videojuego>`
 - Dar de alta el método `findNintendoGames(): List<Videojuego>`
 - Hacer algoritmo del método
 - Documentar el método
 - Comprobar las pruebas de calidad
 - Subir al repositorio
 - Comunicar en Jira
 - Añadir método `findNintendoGames` en la interface y registrar el método
 - Implementar el método `findNintendoGames` en la `VideojuegoServiceImpl`
- Pensar PU `NintendoGames01`
- Crear PU `NintendoGames01` para el método `findNintendoGames`
 - Si no existen juegos de Nintendo devolverá una lista vacía []
 - Simular con un mock que lanzará una `NotFoundException` en caso de error
- Crear el método `findNintendoGames():List<VideojuegoDto>` en la capa Service
- Incluir en el nav de la página principal un botón para redireccionar al método GET `findNintendoGames` que devolverá la lista de los juegos de Nintendo
- Crear el método GET `findNintendoGames():string` con el endpoint “/videojuego/nintendoGames” en la capa Controller
- Reutilizar la plantilla de vista `Videojuegos.html`

Filtro de videojuegos según su año de publicación

- Crear método findByYear(int year): List<Videojuego> en capa DAO
 - Dar de alta el método findByYear (int year) en la interfaz DAOVideojuego
 - Implementar el método findByYear(int year) en la clase DAOVideojuegoImpl
 - Hacer el algoritmo
 - Documentar el método
 - Comprobar las pruebas de calidad
 - Subir al repositorio
 - Comunicar en Jira
- Pensar PU FilterByYear01
- Crear PU FilterByYear01 para el método findByYear (capa DAO)
 - Probar que con un año de publicación se obtengan videojuegos
 - Probar que con un año de publicación se devuelva una lista vacía
 - Probar que con un año de publicacion mayor al actual salte error
- Crear método findByYear(int year):List<VideojuegoDto> en capa Service
- Incluir en la página principal un elemento de búsqueda para filtrar la lista de videojuegos con el método GET findByYear():String que devolverá la lista de los juegos filtrada por el año seleccionado/introducido
- Crear método GET findByYear(int year):String con endpoint “/videojuego/findVideojuegosByYear” en capa Controller
- Reutilizar la plantilla de vistaVideojuegos.html

Diagrama ER

Es sencillito pero tampoco da pa mas

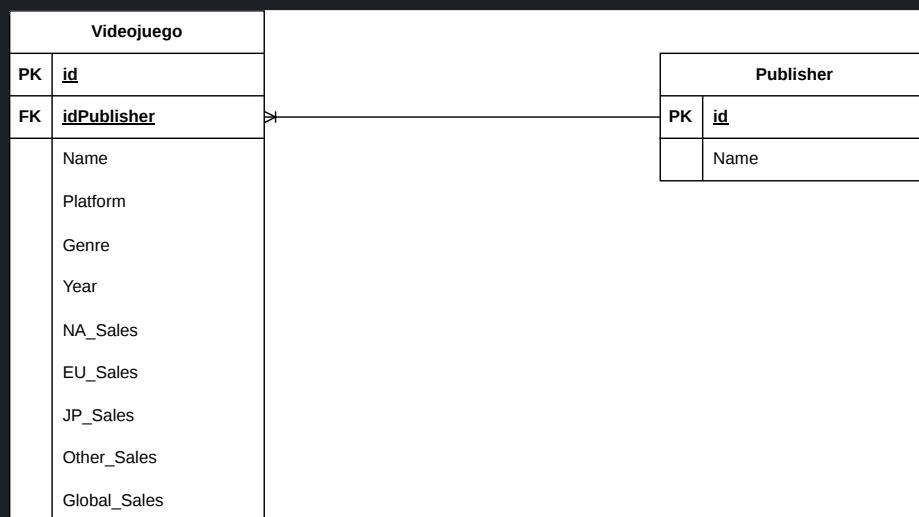


Diagrama de clases del modelo

