DP1 2020-2021

Documento de Diseño del Sistema

Proyecto Standby

https://github.com/gii-is-DP1/dp1-2020-gi-04

Miembros <en orden alfabético por apellidos>:

* Bogdan, George Laurentiu
* Diz Gil, Guillermo
* Muñoz Pérez, Carmen María
* Rodríguez Pérez, Francisco

Tutor: José María García

GRUPO GI-04

Versión 1

17/01/2021

# Historial de versiones

*Estos son ejemplo del contenido que debería tener el historial de cambios del documento a entregar a lo largo de los sprints del proyecto*

|  |  |  |  |
| --- | --- | --- | --- |
| **Fecha** | **Versión** | **Descripción de los cambios** | **Sprint** |
| 17/01/2021 | V1 | * Creación del documento | 4 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Contents

[Historial de versiones 2](#_Toc58783389)

[Introducción 4](#_Toc58783390)

[Diagrama(s) UML: 4](#_Toc58783391)

[Diagrama de Dominio/Diseño 4](#_Toc58783392)

[Diagrama de Capas (incluyendo Controladores, Servicios y Repositorios) 5](#_Toc58783393)

[Patrones de diseño y arquitectónicos aplicados 5](#_Toc58783394)

[Decisiones de diseño 5](#_Toc58783395)

[Decisión X 6](#_Toc58783396)

[Descripción del problema: 6](#_Toc58783397)

[Alternativas de solución evaluadas: 6](#_Toc58783398)

[Justificación de la solución adoptada 6](#_Toc58783399)

# Introducción

The short films are the first productions that are usually made by those who want to dedicate themselves to the world of cinema, but at the same time they are ignored in the cultural field within the film industry. The best-known short films are those that win very important awards or are created by very important companies such as Disney. But what about that majority that does not win awards? They always fall into oblivion. Despite all this, almost all the great filmmakers of our time have started making short films and these have been vital in their rise to fame. Thus, the Standby project was born out of the need of both amateur and professional filmmakers who want to make themselves known and make their way in the world of cinema through short films, as well as the need to promote and disseminate the importance of short films within the industry.

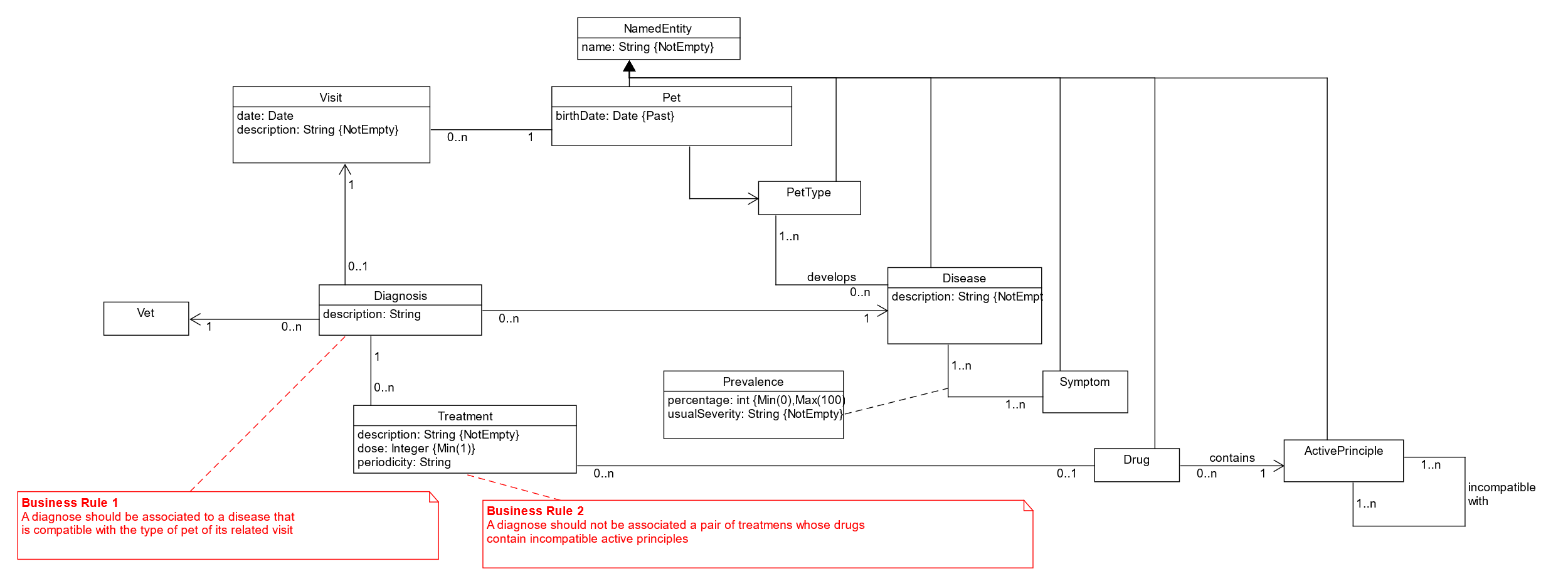
For all these reasons, this project aims to provide an application that brings together short films from around the world, as well as serving as a stimulus for those who want to join the film industry. Standby will be an application that will allow filmmakers to share their short films. Thus, users will be able to watch, comment and evaluate these short films. In addition, companies will be able to filter them and access additional information that will allow them to contact the filmmakers for possible hiring.

# Diagrama(s) UML:

## Diagrama de Dominio/Diseño

*En esta sección debe proporcionar un diagrama UML de clases que describa el modelo de dominio, recuerda que debe estar basado en el diagrama conceptual del documento de análisis de requisitos del sistema pero que debe:*

* *Especificar la direccionalidad de las relaciones (a no ser que sean bidireccionales)*
* *Especificar la cardinalidad de las relaciones*
* *Especificar el tipo de los atributos*
* *Especificar las restricciones simples aplicadas a cada atributo de cada clase de domino*
* *Incluir las clases específicas de la tecnología usada, como por ejemplo BaseEntity, NamedEntity, etc.*
* *Incluir los validadores específicos creados para las distintas clases de dominio (indicando en su caso una relación de uso con el estereotipo <<validates>>.*

*Un ejemplo de diagrama para los ejercicios planteados en los boletines de laboratorio sería (hemos omitido las generalizaciones hacia BaseEntity para simplificar el diagrama):*

## Diagrama de Capas (incluyendo Controladores, Servicios y Repositorios)

En esta sección debe proporcionar un diagrama UML de clases que describa el conjunto de controladores, servicios, y repositorios implementados, incluya la división en capas del sistema como paquetes horizontales tal y como se muestra en el siguiente ejemplo:

Diagram

Description automatically generated

El diagrama debe especificar además las relaciones de uso entre controladores y servicios, entre servicios y servicios, y entre servicios y repositorios.

Tal y como se muestra en el diagrama de ejemplo, para el caso de los repositorios se deben especificar las consultas personalizadas creadas (usando la signatura de su método asociado).

# Patrones de diseño y arquitectónicos aplicados

En esta sección de especificar el conjunto de patrones de diseño y arquitectónicos aplicados durante el proyecto. Para especificar la aplicación de cada patrón puede usar la siguiente plantilla:

## Patrón: <Nombre del patrón>

### Tipo: Arquitectónico | de Diseño

### Contexto de Aplicación

Describir las partes de la aplicación donde se ha aplicado el patrón. Si se considera oportuno especificar el paquete donde se han incluido los elementos asociados a la aplicación del patrón.

### Clases o paquetes creados

Indicar las clases o paquetes creados como resultado de la aplicación del patrón.

### Ventajas alcanzadas al aplicar el patrón

Describir porqué era interesante aplicar el patrón.

# Design decision

In this section we will describe the design decisions we have made throughout the development process of the application.

All decisions:

1. Pick the frontend framework
2. Video thumbnail generation
3. Data relation loading
4. Initial data load
5. Security layer
6. Controller creation

## Decision 1: Pick the frontend framework

### Problem description:

We need to display the view to the user and using a frontend framework would ease the workload of writing all the front from scratch.

### Alternative evaluated solutions:

Alternative 1.a: No framework, plain HTML, CSS, JavaScript

**Advantages**:

* We have full control of the application flow.
* Less overhead for the client, because of file size

**Disadvantages**:

* More work as we must write more code
* More complexity as we must write the full system

Alternative 1.b: Use Thymeleaf as server-side rendering.

**Advantages:**

* Less overhead for the client because it is rendered in the server.
* Access to the server data
* Conditional rendering

**Disadvantages:**

* We must learn Thymeleaf
* Page is reloaded almost every time we must render.

Alternative 1.c: Use React as client-side rendering.

**Advantages:**

* Dynamic page rendering
* Reactive to changes.
* Use of existing UI libraries.

**Disadvantages:**

* We must learn React.
* More complex.

### Justified solution adopted

We have chosen the alternative 1.b Thymeleaf because it was the perfect tradeoff between complexity for the developer and user experience.