DP1 2020-2021

Documento de Diseño del Sistema

Proyecto TEAcademy

URL repositorio: https://github.com/gii-is-DP1/dp1-2020-g1-01.git

Miembros en orden alfabético por apellidos:

- Alonso Martín, Fernando
- Álvarez García, Gonzalo
- Martínez Fernández, Javier
- Ramos Blanco, María Isabel
- Vilariño Mayo, Javier
- Yugsi Yugsi, Evelyn Gisele

Tutor: Manuel Resinas

GRUPO G1-1

Versión 1.0

08 de enero de 2021

Historial de versiones

Fecha	Versión	Descripción de los cambios	Sprint
26/12/2020	V1	• Creación del documento. • Añadido diagrama de dominio. • Añadidas decisiones de diseño.	3
08/01/2021	V2	• Añadidas decisiones de diseño. • Añadido diagrama de capas. • Terminados puntos restantes (introducción y patrones utilizados).	3

Contents

Historial de versiones 2
Introducción 5
Diagrama(s) UML: 6
Diagrama de Dominio/Diseño 6
Diagrama de Capas (incluyendo Controladores, Servicios y Repositorios) 7
Patrones de diseño y arquitectónicos aplicados 8
Patrón: MVC 8
Tipo: Arquitectónico | de Diseño 8
Contexto de Aplicación 8
Clases o paquetes creados 8
Ventajas alcanzadas al aplicar el patrón 8
Decisiones de diseño 8
Decisión 1 8
Descripción del problema: 8
Alternativas de solución evaluadas: 9
Justificación de la solución adoptada 9
Decisión 2 9
Descripción del problema: 9
Alternativas de solución evaluadas: 9
Justificación de la solución adoptada 9
Decisión 3 9
Descripción del problema: 9
Alternativas de solución evaluadas: 9
Justificación de la solución adoptada 9
Decisión 4 9
Descripción del problema: 9
Alternativas de solución evaluadas: 10
Justificación de la solución adoptada 10
Decisión 5 10
Descripción del problema: 10

Alternativas de solución evaluadas:..... 10
Justificación de la solución adoptada 10

Introducción

T.E.A es una academia de inglés sevillana, la cual se dedica tanto a la certificación de niveles oficiales de Cambridge, Trinity y Oxford, como a la realización de actividades extraescolares.

Se pretende que con la implementación del nuevo software se facilite y simplifique la gestión de alumnos, inscripciones, pagos, actividades extraescolares y otros aspectos de gestión. Por otra parte, se pretende que la disposición de material para los alumnos sea mucho más accesible y la visibilidad de la academia en cuanto a repercusión se aumente.

Tras la realización del proyecto, resaltamos la implementación de las historias relacionadas con el Wall of Fame y las del calendario ya que presentan un mayor aspecto visual y dificultad a la hora de desarrollarlo.

Diagrama(s) UML:

Diagrama de Dominio/Diseño

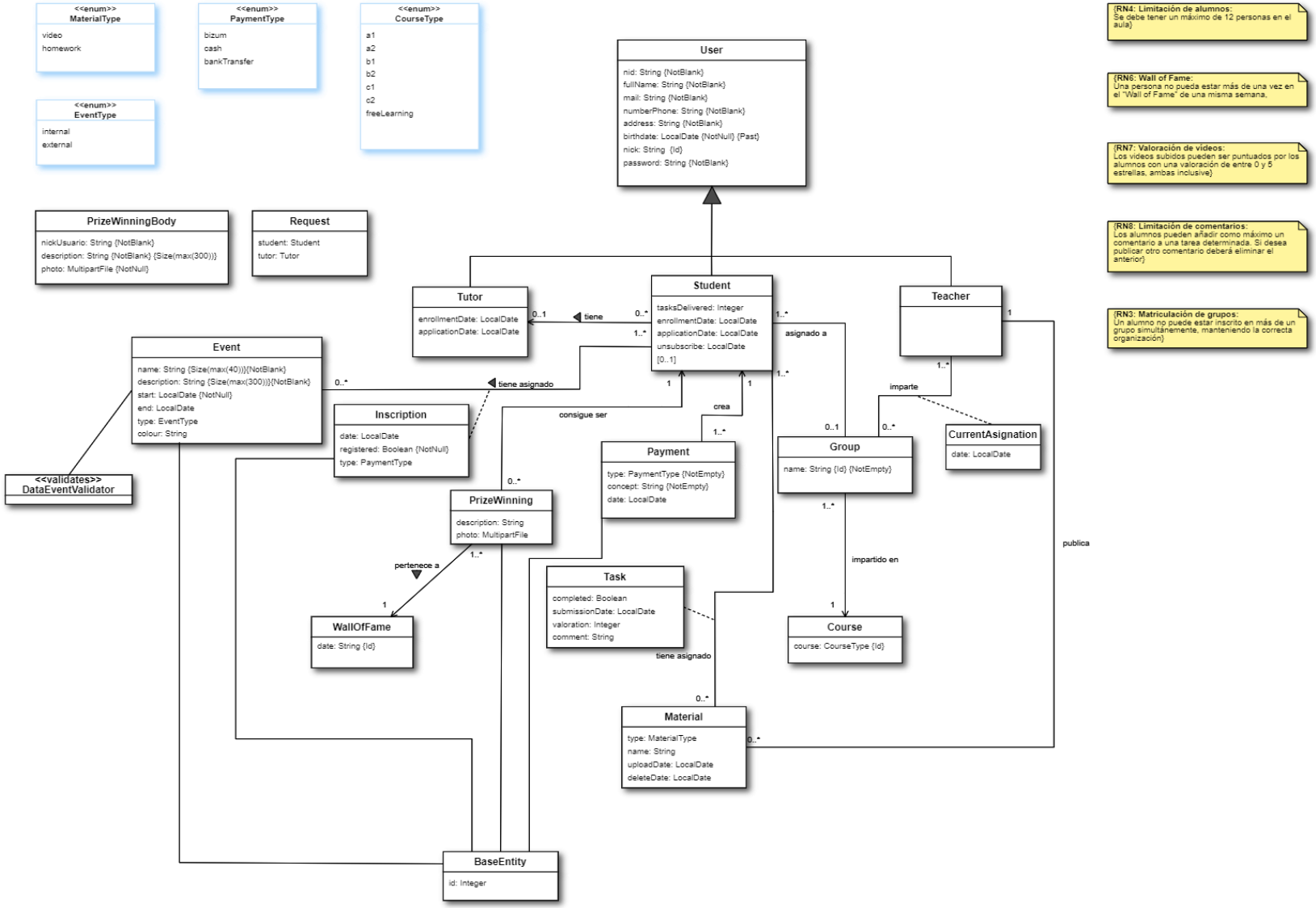
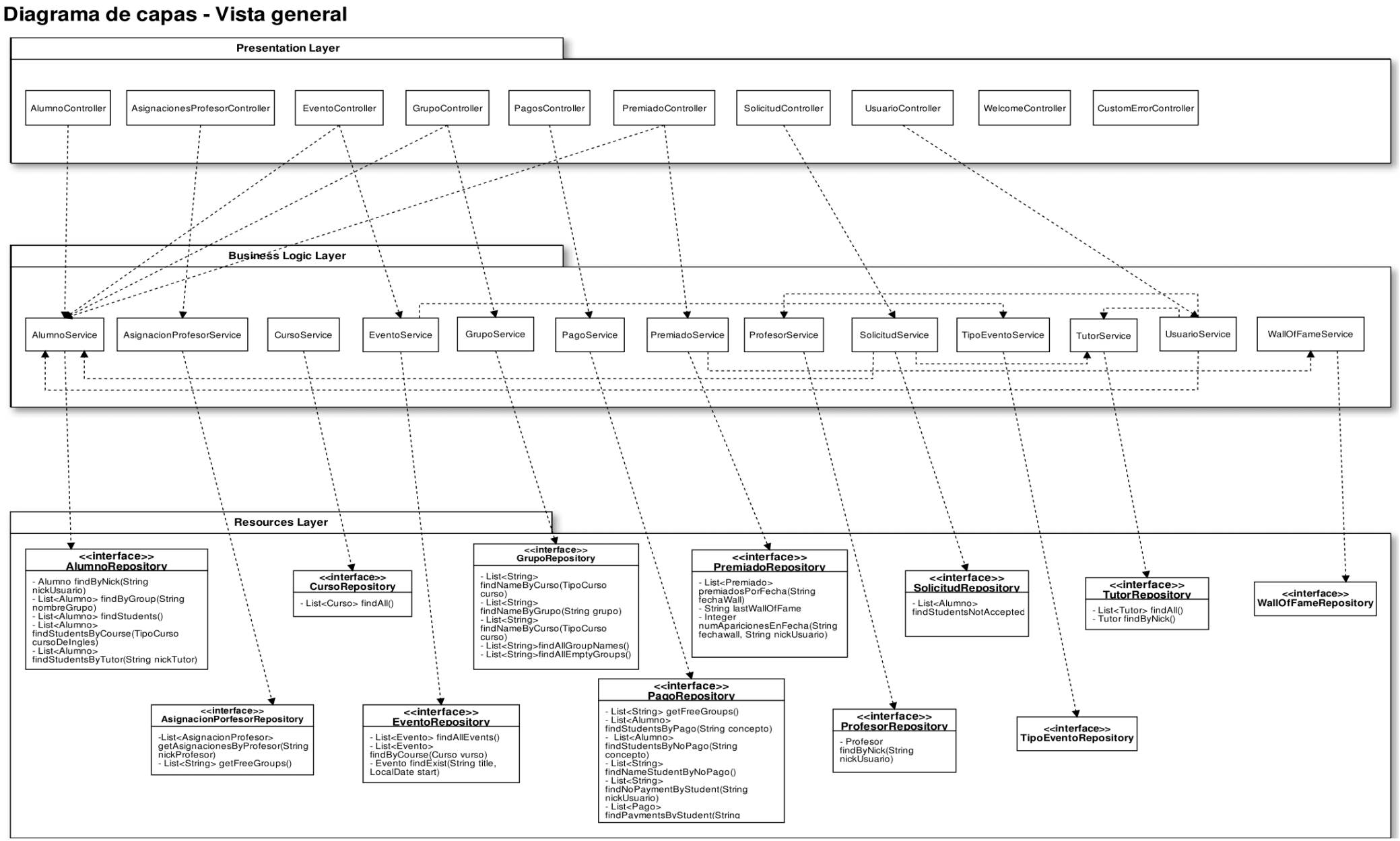


Diagrama de Capas (incluyendo Controladores, Servicios y Repositorios)



Patrones de diseño y arquitectónicos aplicados

Patrón: MVC

Tipo: Arquitectónico | de Diseño

Contexto de Aplicación

Hemos utilizado dicho patrón en toda nuestra aplicación, en concreto, las vistas y controladores se referirán a la capa de presentación del estilo arquitectónico por capas. Los servicios, las entidades de la parte del modelo del patrón se referirá a la capa de la lógica de negocio. Por último, los repositorios, también incluidos en el modelo MVC, se referirán a la capa de datos.

Para aplicar dicho patrón hemos utilizado frameworks de Spring y para las vistas hemos utilizado la tecnología de React, así como sus variantes, por ejemplo, Hooks, Redux, etc.

La interacción entre React y Spring es mediante llamadas desde las vistas en React a la API REST implementada en Spring, siendo React y Spring dos servidores distintos.

Clases o paquetes creados

Para poder interactuar con el frontend hemos decidido ubicar React concretamente en *dp1-2020-g1-01/src/main/resources/static/frontend*. Dentro de la carpeta *src/componentes* se encuentran los distintos componentes referentes a las vistas creados, necesarios para trabajar con React. Es importante destacar que el componente padre de todos se encuentra en la carpeta src y se llama App.js

En Spring los paquetes que siguen el patrón arquitectónico MVC se encuentran en la ruta *dp1-2020-g1-01/src/main/java/*org.springframework.samples.petclinic. Aquí se encuentran las distintas clases, los repositorios, los servicios y controladores asociados a dichas clases.

Ventajas alcanzadas al aplicar el patrón

Las distintas ventajas que hemos encontrado al realizar el proyecto con este patrón arquitectónico son:

- Alta cohesión
- Bajo acoplamiento
- Múltiples vistas
- Separación de responsabilidades

Decisiones de diseño

En esta sección describiremos las decisiones de diseño que se han tomado a lo largo del desarrollo de la aplicación que vayan más allá de la mera aplicación de patrones de diseño o arquitectónicos.

Decisión 1

Descripción del problema:

Como grupo se nos presentó la ocasión de poder aprender React desde cero en lugar de hacer el proyecto, como se nos proponía en la asignatura, para añadir mayor complejidad. El problema es que ninguna de nosotros tenía conocimientos previos de esta librería de Javascript.

Alternativas de solución evaluadas:

Alternativa 1.a: Hacer uso de jsp.

Alternativa 1.b: Hacer uso de Angular.

Alternativa 1.c: Hacer uso de React.

Justificación de la solución adoptada

Finalmente, decidimos hacer uso de React para implementar el frontend de la aplicación ya que de esta forma añadimos complejidad al proyecto y además, aprendemos el funcionamiento de esta librería.

Decisión 2

Descripción del problema:

Se nos presentó el problema de tener que pasar estados entre componentes independientes, no pudiendo pasar estos estados por “props” al no ser hijos.

Alternativas de solución evaluadas:

Alternativa 2.a: Hacer uso de herencias entre componentes.

Alternativa 2.b: Implementar redux

Justificación de la solución adoptada

Tras consultarlo con el profesor asignado, se vio óptimo implementar redux para hacerlo más eficiente.

Decisión 3

Descripción del problema:

Tras encontrarnos numerosos problemas con el modelado de datos a la hora de integrarlo con Spring debido a bucles creados entre las entidades, decidimos hacer modificaciones en el mismo.

Alternativas de solución evaluadas:

Alternativa 3.a: Mantener el modelo y simplificar el sistema.

Alternativa 3.b: Cambiar el modelo de datos.

Justificación de la solución adoptada

Decidimos realizar modificaciones en el modelo de datos ya que queríamos que el proyecto fuese lo más cercano posible a un proyecto real.

Decisión 4

Descripción del problema:

Al intentar ampliar la seguridad de nuestro sistema, nos encontramos con varias soluciones de autentificación de usuarios.

Alternativas de solución evaluadas:

Alternativa 4.a: Utilizar JWT & Spring Boot.

Alternativa 4.b: Utilizar sesiones.

Justificación de la solución adoptada

Decidimos hacer uso de sesiones ya que al no utilizar Spring tal y como se enseñaba en la asignatura, la solución 4.a no era viable. Además, al ser nuestro backend una API REST y al tener un proyecto que no manejaba una gran cantidad de datos, veíamos más útil usar sesiones.

Decisión 5

Descripción del problema:

Al realizar el diseño de las vistas, nos encontramos que al implementar el frontend con React teníamos que escribir todo el código desde cero haciendo uso de JSX. Para facilitar este problema y tras repetidas búsquedas, nos encontramos con la existencia de una librería externa conocida como “Prime React”

Alternativas de solución evaluadas:

Alternativa 5.a: Escribir todo el código en JSX.

Alternativa 5.b: Hacer uso de PrimeReact

Justificación de la solución adoptada

Decidimos optar por la segunda alternativa, pues esta librería nos facilita el desarrollo de los distintos componentes.