**Proyecto de Rector Simulator**

**Documento de Arquitectura de Software**

**DAS**

Versión 1.0

**Lima, 28 de septiembre del 2024**

**Control de Versiones**

|  |  |  |  |
| --- | --- | --- | --- |
| **Fecha** | **Versión** | **Descripción** | **Autor(es)** |
| 28/09/2024 | 1.0 | Creación del documento | * Karlo Toro * Diego Gamarra * Andrew Serna |
| 13/11/2024 | 1.1 | Corrección de errores | * Karlo Toro * Diego Gamarra * Andrew Serna |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**ÍNDICE**

[**1. Introducción**](#_q9tkgcjm9ld5) **4**

[1.1. Propósito](#_8w3kmu11ea30) 4

[1.2. Alcance](#_jvwhpntsjzn7) 4

[1.3. Objetivos](#_5308mtx0z5pu) 4

[**2. Consideraciones generales**](#_t50m3uvw1mh2) **4**

[2.1. Tecnologías utilizadas:](#_avelsf3rbxqp) 4

[2.2. Arquitectura basada en el modelo MTV](#_ltpitpqw2ez3) 5

[2.3. Metas y restricciones arquitectónicas](#_qxg6fjeulxev) 5

[**3. Diseño de software**](#_iyb1sasxsqzs) **7**

[3.1. Definición de módulos](#_y94bz7odwqaf) 7

[3.2. Relaciones, dependencias y prioridad](#_xs89nj4b3abe) 7

# Introducción

Este documento presenta la arquitectura del sistema, definiendo las tecnologías y procesos involucrados en su desarrollo. Se detallan las herramientas utilizadas para gestionar el backend, el frontend, la base de datos, y el control de versiones. Además, se explican las capas arquitectónicas que forman el sistema, así como los módulos específicos que permiten su funcionamiento óptimo. Finalmente, se identifican los riesgos y restricciones que podrían surgir durante la implementación del sistema y el manejo de decisiones.

## Propósito

El propósito de este documento es mostrar la arquitectura del software del sistema propuesto. Se identifican todas las tecnologías que serán empleadas en el desarrollo, cubriendo tanto el backend como el frontend, así como los sistemas de almacenamiento y control de versiones. También se describe la estructura modular del software, dividiendo las responsabilidades y asegurando la flexibilidad necesaria para futuras modificaciones.

## Alcance

* Proveer una visión general de la arquitectura de software del proyecto.
* Especificar las tecnologías y herramientas que se emplearán en el desarrollo.
* Describir el diseño modular del sistema, detallando las relaciones y dependencias entre los módulos.
* Identificar los riesgos y restricciones arquitectónicas que podrían afectar el proyecto.

## Objetivos

* Establecer y especificar las tecnologías utilizadas para el backend, frontend, y la gestión de datos.
* Proporcionar una base para que los desarrolladores puedan implementar el sistema de manera eficiente.
* Facilitar un desarrollo colaborativo mediante el uso de plataformas de control de versiones.
* Asegurar que la arquitectura sea flexible para adaptarse a futuros cambios o requerimientos adicionales.

# Consideraciones generales

## Tecnologías utilizadas:

* **HTML 5:** Especifica la estructura de las páginas web y sus contenidos.
* **CSS 3:** Maneja el diseño y presentación de las páginas web.
* **React:** Biblioteca de JavaScript para construir interfaces de usuario.
* **MySQL:** Sistema de gestión de bases de datos relacional.
* **Django:** Framework de desarrollo web para Python que facilita el trabajo con el Modelo MTV y una rápida creación de una página de administración para que los usuarios no técnicos con altos privilegios puedan modificar la página de forma sencilla.
* **Git:** Software de control de versiones.
* **GitHub:** Plataforma para alojar repositorios de Git.
* **JavaScript:** Lenguaje interpretado que añade interactividad en aplicaciones web.

## Arquitectura basada en el modelo MTV

A diferencia del modelo arquitectónico MVC (Model View Cotroller) en Django se maneja otro concepto, el cual es MTV (Model View Template) y manifiesta sus diferencias en cuanto a la forma en que son concebidas estas capas y también como se manejan internamente en Django.

Model: Representa la capa de persistencia de la aplicación, es decir, aquí se generan los modelos (clases y sus relaciones) las cuales luego son persistidas en la base de datos y se realizan distintas operaciones, nos permite una conexión con la BD de forma bidireccional.

View: Es el análogo al controlador en el modelo MVC es decir que gestiona la comunicación entre la interfaz de usuario y la lógica de negocio, por ejemplo cuando un usuario realiza un clic en el botón comprar el controlador notifica al modelo que elimine esos productos del stock y los actualice según sea necesario. A su vez, cuando esto sucede, el modelo envía al controlador la información para que actualice el Template para así representar de forma correcta la oferta actual de productos. En esta capa se representa la lógica de negocio de la aplicación, se procesan las solicitudes HTPP y se determinan los datos que seran mostrados por la Template.

Template: En el patrón MTV de Django, el Template cumple una función similar a la vista en el modelo MVC, pero se limita a la presentación de la información, sin incluir lógica de compleja de formateo o presentación de datos y mucho menos de negocio de manera que se separen mejor las responsabilidades con View. En esta capa se utilizan herramientas como HTML, CSS y el Django Template Language (DTL) para mostrar al usuario la información estática de la aplicación y su estructura. La interacción y el dinamismo se gestionan mediante archivos JavaScript y otros recursos estáticos que el Template referencia a través de la carpeta de archivos estáticos.

Diagrama

Descripción generada automáticamente

Figura 1 Diagrama de físico-Rector Simuleitor

## Metas y restricciones arquitectónicas

|  |  |  |  |
| --- | --- | --- | --- |
| **Atributos de calidad “Observables”** | | | |
| Atributo | Descripción | Tácticas /Patrón de  Arquitectura | Aplicación |
| Seguridad | Garantizar que el acceso a los recursos y funcionalidades esté restringido según los roles asignados a los usuarios autorizados. | Control de Acceso Basado en Roles: Asignación de permisos según los roles de usuario (estudiante, bibliotecaria). | Aplicación al momento de autenticación de usuarios y accesos a funcionalidades específicas. |
| Usabilidad | Facilidad de uso del sistema para los usuarios finales. | Realizacion de pruebas exhaustivas de aceptacion de usuario centradas en la GUI e interactividad. | Desarrollo de interfaces intuitivas, minimalistas y claras que permitan una rapida adaptación al sistema. |

# 

|  |  |  |  |
| --- | --- | --- | --- |
| **Atributos de calidad “No observables”** | | | |
| Atributo | Descripción | Tácticas /Patrón de  Arquitectura | Aplicación |
| Modificabilidad | Facilidad para realizar cambios en el sistema, como agregar nuevas funcionalidades o adaptarse a requisitos cambiantes. | Arquitectura de Módulos: Descomposición del sistema en módulos específicos e independientes para cada funcionalidad del sistema. | División de la lógica de gestión de recursos, toma de decisiones, generación del ranking y de eventos, así como el acceso a datos se encontraran en módulos separados e independientes. |
| Mantenibilidad | Capacidad de realizar mantenimiento y mejoras al sistema de manera eficiente y económica. | Separación de capas: División clara de las capas de presentación, lógica de negocios y acceso a datos para facilitar las actualizaciones. | Implementación del enfoque orientado a objetos que facilita la división de responsabilidades entre cada clase e instancia del sistema. |
| Portabilidad | Adaptabilidad del sistema para funcionar en diferentes entornos. | Diseño Responsivo: Desarrollo de interfaces de usuario que se ajustan automáticamente. | Implementación de técnicas de diseño web responsivo para una experiencia de usuario consistente y accesible. |

# Diseño de software

En esta parte se definirán los módulos del software a desarrollar, es decir, componentes del software que mantienen funcionalidades específicas del mismo.

## Definición de módulos

* Módulo de Decisiones: Presenta las opciones de decisión al jugador en eventos.
* Módulo de Consecuencias: Calcula los efectos de cada decisión sobre los recursos y aprobación del alumnado.
* Módulo de Recursos y reputación: Se encarga de actualizar los recursos y aprobación del alumnado.
* Módulo de Reportes: Presenta al jugador el estado actual de los recursos y aprobación del alumnado.
* Módulo de Puntuación: Ajusta la puntuación general según las acciones del jugador.
* Módulo de Generación: Produce eventos de manera no predecible, alterando el flujo del juego.
* Módulo de Almacenamiento: Guarda los puntajes obtenidos en una base de datos.
* Módulo de Visualización: Presenta los 5 mejores puntajes al jugador en una tabla.
* Módulo de Evaluación: Analiza las decisiones y el desempeño del jugador para determinar el final adecuado.
* Módulo de Narrativa: Despliega el final y cierra la historia de acuerdo con el rendimiento del jugador.

## Relaciones, dependencias y prioridad

|  |  |
| --- | --- |
| Módulo de Decisiones | |
| Relaciones | * Módulo de Consecuencias * Módulo de Evaluación |
| Dependencias | Módulo de Generación |
| Prioridad | Alta |

|  |  |
| --- | --- |
| Módulo de Consecuencias | |
| Relaciones | * Módulo de Recursos y reputación * Módulo de Puntuación |
| Dependencias | Módulo de Decisiones |
| Prioridad | Alta |

|  |  |
| --- | --- |
| Módulo de Recursos y reputación | |
| Relaciones | Módulo de reportes |
| Dependencias | Módulo de Consecuencias |
| Prioridad | Alta |

|  |  |
| --- | --- |
| Módulo de Reportes | |
| Relaciones | - |
| Dependencias | Módulo de Recursos y reputación |
| Prioridad | Media |

|  |  |
| --- | --- |
| Módulo de Puntuación | |
| Relaciones | * Módulo de Almacenamiento * Módulo de Visualización |
| Dependencias | Módulo de Consecuencias |
| Prioridad | Media |

|  |  |
| --- | --- |
| Módulo de Generación | |
| Relaciones | Módulo de Decisiones |
| Dependencias | - |
| Prioridad | Alta |

|  |  |
| --- | --- |
| Módulo de Almacenamiento | |
| Relaciones | Módulo de Visualización |
| Dependencias | Módulo de Puntuaciones |
| Prioridad | Media |

|  |  |
| --- | --- |
| Módulo de Visualización | |
| Relaciones | - |
| Dependencias | Módulo de Almacenamiento |
| Prioridad | Baja |

|  |  |
| --- | --- |
| Módulo de Evaluación | |
| Relaciones | - |
| Dependencias | * Módulo de Puntuación * Módulo de Decisiones |
| Prioridad | Media |

|  |  |
| --- | --- |
| Módulo de Narrativa | |
| Relaciones | - |
| Dependencias | Módulo de Evaluación |
| Prioridad | Media |

## Identificación de riesgos

* Persistencia de puntajes fallida: El sistema de almacenamiento de puntajes podría perder información si no se maneja adecuadamente la persistencia de datos.
* Visualización de información confusa o incompleta: Si el módulo de reportes o visualización no presenta de forma clara los recursos, la reputación o los puntajes, el jugador podría no comprender el impacto de sus decisiones.
* Poca variabilidad en finales o decisiones: Si los eventos aleatorios, decisiones o finales no son suficientemente variados, el jugador podría sentir que el juego es repetitivo y perder interés rápidamente.
* Dificultad para añadir nuevos contenidos: Si la arquitectura no es lo suficientemente modular, agregar nuevos eventos, decisiones o finales puede volverse complicado.