UNMDP - Facultad de Ingeniería - Departamento de Informática

Proyecto final para optar al grado de Ingeniero en Informática

Juegos serios como refuerzo en el proceso de aprendizaje en alumnos de primaria

Autor: Luis Mariano Fantini

Email: marianof.06@gmail.com

Director: Adolfo Tomás Spinelli

Codirector: Franco Kühn

Referente Funcional: Gladys Alina Ludueña

Mar del Plata, julio de 2021

UNMDP - Facultad de Ingeniería - Departamento de Informática

Proyecto final para optar al grado de Ingeniero en Informática

Juegos serios como refuerzo en el proceso de aprendizaje en alumnos de primaria

Autor: Luis Mariano Fantini

Email: marianof.06@gmail.com

Director: Adolfo Tomás Spinelli

Codirector: Franco Kühn

Referente Funcional: Gladys Alina Ludueña

Mar del Plata, julio de 2021

Agradecimientos,

A mi familia, por apoyarme a lo largo de la carrera y por su amor incondicional.

A mis amigos, por incluirme en sus vidas y por estar siempre a mi lado.

A mis compañeros de cursada, por ayudarme e incentivarme a mejorar como estudiante.

A mis profesores de Ingeniería, por contribuir con mi aprendizaje y con mi desarrollo académico.

A mis directores y a mi referente funcional, por acompañarme a lo largo del desarrollo del proyecto y evacuar todas mis dudas.

Resumen

En este trabajo se llevó a cabo la elicitación y especificación de requerimientos, el diseño, la implementación y la puesta en funcionamiento de un sistema de software, cuyo objetivo principal, es contribuir en el proceso de aprendizaje de lectura y escritura en los alumnos que arrancan en sus primeros años de primaria.

Para ello, se desarrolló un videojuego para los dispositivos Android que permite a los estudiantes entretenerse mientras aprenden de forma indirecta, recopilando datos sobre su avance, para que luego los docentes puedan analizarlo.

Dicha información se presenta en distintos gráficos que son mostrados en una página web, que además permite la gestión de los distintos docentes que están habilitados para acceder al sitio, administrar que alumnos tienen permiso para acceder al videojuego y, finalmente, la creación y edición de niveles dentro del juego.

Índice General

Capítulo 1: Introducción

1.1 Preámbulo

Muchos estudiantes se esfuerzan a diario y aun así no alcanzan lo esperado socialmente: concluir primer grado leyendo y escribiendo. Ante esta situación, la decisión que se tomaba era la repitencia, convirtiendo a primero en el grado con mayor índice de permanencia de toda la escolaridad.

Los alumnos ingresan con diferentes niveles de conceptualización y pretender que concluyan su primer año leyendo de forma convencional, puede ser para algunos algo sencillo y para otros un verdadero reto.

El punto de partida de un niño que asistió al jardín de infantes e ingresa con una escritura silábica no es el mismo que el de otro que no reconoce la escritura de su nombre, realiza grafismos primitivos y no diferencia números de letras. Lógicamente, el primero se encuentra con una ventaja importante.

Es por este motivo que hoy en día primer y segundo año forman parte de un único "bloque", un bloque alfabetizador, conocido como Unidad Pedagógica, en el que primer grado no se puede repetir.

Como Claudia [1] menciona en su blog, si un alumno no logra las expectativas de logro es imprescindible analizar las causas y proponer nuevas instancias de aprendizaje empleando otras estrategias. No trabajar con las causas hará que muchos, que no repitieron primero, repitan segundo. Es decir que, el problema no se solucionaría, simplemente se trasladaría.

Además, cabe resaltar que la pandemia ocasionada por el coronavirus ha provocado una crisis sin precedentes en todos los ámbitos. En el entorno de la educación, se ha decidido recurrir a la enseñanza de manera virtual en todos los niveles y esta permanecerá como una herramienta más de la docencia. Debajo de este contexto, es importante que surjan alternativas que permitan reforzar el aprendizaje.

1.2 Objetivo

El objetivo general del proyecto es desarrollar y poner a disposición de los docentes, una herramienta que contribuya en el proceso de aprendizaje de lectura y escritura de los alumnos que cursan la Unidad Pedagógica.

Para alcanzar el objetivo general del proyecto, es necesario satisfacer los siguientes objetivos específicos:

- Especificar los requerimientos a partir de entrevistas con los docentes.
- Diseñar un videojuego que les resulte entretenido a los alumnos y que a su vez les permita fortalecer y comprobar los conocimientos adquiridos en clases.
- Desarrollar el videojuego en cuestión.
- Desarrollar una interfaz que el docente utilizará para obtener analíticas a partir del progreso de los alumnos que se entretienen y aprenden con el videojuego.

1.3 Análisis FODA

Fortalezas:

- El producto estará adaptado a los más recientes métodos de aprendizaje que se utilizan en la Unidad Pedagógica.
- Probar que aprender no necesariamente es una tarea aburrida, sino que puede ser todo lo contrario bajo las correctas condiciones.
- La posibilidad de obtener estadísticas detalladas a partir del progreso de los alumnos.

Oportunidades:

- Debido a la pandemia global, resulta interesante experimentar con métodos de aprendizaje alternativos.
- Generar interés en los alumnos hacia el dispositivo que utilizarán para jugar al videojuego.

- El mercado objetivo, que son los niños de 5 y 6 años de edad que ingresan a la primaria, es un nicho desatendido en el mundo de los videojuegos.

Debilidades:

- No va a estar definido para todas las plataformas.
- El arte del videojuego no va a ser creado, por lo que el videojuego deberá ser adaptado.

Amenazas:

- Los alumnos que cursan la Unidad Pedagógica puede que no dispongan de un dispositivo que les permita jugar el videojuego.
- Culturalmente los videojuegos no son bien aceptados en la educación.

Capítulo 2: Marco Teórico

2.1 Videojuegos

2.1.1 Definición

Según el diccionario Merriam-Webster [2], un videojuego es un juego electrónico en el que las personas controlan imágenes en una pantalla de video. Todo videojuego es un juego, pero no todo juego es un videojuego.

Zimmerman [3], define a un juego como una actividad interactiva voluntaria, en la que uno o más participantes siguen reglas que restringen su comportamiento, promulgando un conflicto artificial que termina en un resultado cuantificable.

2.1.2 Modelo de Diseño e Investigación MDA

El marco MDA definido por Hunicke et al. [4], es un enfoque formal para la comprensión de juegos que intenta relacionar la brecha entre el diseño, desarrollo, criticismo e investigación técnica en los videojuegos. El modelo dice que los juegos son creados por diseñadores/equipos de desarrolladores y son consumidos por los jugadores. Se compran, usan y finalmente se desechan como la mayoría de los demás bienes de consumo.

Figura 1: La producción y el consumo de videojuegos. Adaptado de Hunicke et al. [4].

El marco MDA formaliza el consumo de juegos dividiéndolos en sus distintos componentes:

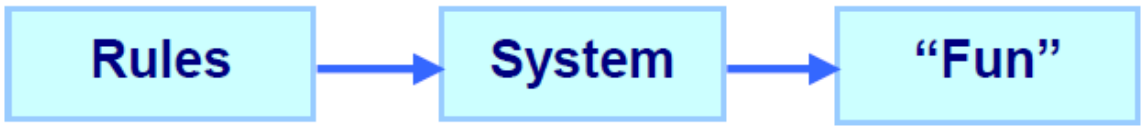


Figura 2: Partes de un videojuego. Adaptado de Hunicke et al. [4].

Y establece sus contrapartes de diseño:

Figura 3: Contrapartes de diseño del videojuego. Adaptado de Hunicke et al. [4].

Las mecánicas describen los componentes particulares del juego, a nivel de representación de datos y algoritmos.

Las dinámicas describen el comportamiento en tiempo de ejecución de las mecánicas que actúan sobre las entradas de los jugadores y las salidas a lo largo del tiempo.

Las estéticas describen las respuestas emocionales deseables evocadas en el jugador, cuando este interactúa con el juego. Al describir las estéticas, hablamos de las siguientes áreas en las que el jugador se siente identificado:

- **Sensación:** El juego como placer sensorial
- **Fantasía:** El juego como fantasía
- **Narrativa:** El juego como drama
- **Desafío:** El juego como carrera de obstáculos
- **Compañerismo:** El juego como marco social
- **Descubrimiento:** El juego como territorio desconocido
- **Expresión:** El juego como autodescubrimiento
- **Sumisión:** El juego como pasatiempo

Cada elemento del marco de MDA puede pensarse como una "lente" o una “perspectiva” del juego. Desde la perspectiva del diseñador, las mecánicas generan las dinámicas y éstas dan lugar a las experiencias estéticas. El jugador percibe o experimenta las estéticas y va generando las dinámicas mediante el uso de las mecánicas que creó el diseñador.

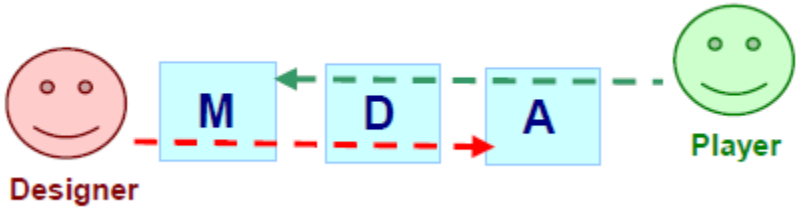


Figura 4: Tanto el diseñador como el jugador tienen una perspectiva distinta del videojuego. Adaptado de Hunicke et al. [4].

2.1.3 Componentes de los videojuegos

Además de las mecánicas, las dinámicas y las estéticas, podemos encontrar como componentes esenciales de los videojuegos la narrativa, la historia, los personajes y los efectos del mismo.

Scolari [5] define a la narrativa como el conjunto de herramientas que señalan los cambios de estado de un hecho ficticio. Es un recuento ordenado de cómo un suceso evoluciona .

Hiebaum [6] dice que por historia comprendemos a la secuencia de eventos que se desenvuelve dentro del juego.

A estas dos definiciones Scolari [5] añade que, si bien ambas son parecidas, la narrativa no es un sinónimo de historia. La historia es lo que pasa a lo largo del juego, es decir, el marco argumental, mientras que la narrativa relata los sucesos descritos en la historia a lo largo del videojuego.

Wikijuegos [7] define a los personajes como cada una de las personas o seres (humanos, animales o de cualquier otra naturaleza) reales o imaginarios que aparece en un videojuego.

Por último, el diccionario Significados [8] establece que se denomina efecto a una sensación, un impacto o una impresión producida en el ánimo o en los sentimientos de una persona. En los videojuegos estos efectos se dan a través del uso de alegorías y símbolos, imágenes, situaciones, sonidos y música, entre otros.

2.2 Mecánicas de Juego

Como indican Hunicke et al. [4], las Mecánicas de Juego son las diversas acciones, comportamientos y mecanismos de control proporcionados al jugador dentro del contexto de un juego.

Järvinen [9] agrega que las Mecánicas de Juego son un medio para guiar al jugador hacia comportamientos particulares al restringir el espacio de posibles planes para alcanzar objetivos.

Plass et al. [10] añaden que, a través de las representaciones y reglas del sistema de juego, los jugadores pueden formar modelos mentales o entendimientos de sistemas, conceptos o fórmulas similares del mundo real. Los jugadores/alumnos pueden obtener conocimientos más profundos no solo aprendiendo las variables involucradas en un sistema, sino también al comprender cómo estas variables interactúan entre sí.

2.3 Mecánicas de Aprendizaje

Zimmerman y Salen [11] definen las Mecánicas de Aprendizaje como los patrones de comportamiento o componentes básicos de la interactividad del alumno, que puede ser una sola acción o un conjunto de acciones interrelacionadas que forman la actividad de aprendizaje esencial que se repite a lo largo de un juego.

Plass et al. [10] mencionan que las Mecánicas de Aprendizaje adaptan la actividad momento a momento de una Mecánica de Juego en una actividad de aprendizaje significativa. Los aspectos de aprendizaje se integran de manera que se conviertan en una parte integral del juego y no simplemente en una adición a la Mecánica de Juego.

Además, Plass et al. añaden que, un ejemplo de mala integración del aprendizaje en un juego es cuando se utiliza una Mecánica de Juego establecida, como una mecánica de carreras o una mecánica de disparos, y la Mecánica de Aprendizaje consiste en una pregunta emergente que se les pide a los jugadores que respondan antes de que puedan continuar. las actividades de carrera o tirador.

2.4 Juegos Serios

Michael y Chen [12] definen a los Juegos Serios como juegos que no tienen como objetivo principal el entretenimiento, el disfrute o la diversión y que, por lo tanto, son juegos que usan su medio artístico para brindar un mensaje, enseñar una lección o proporcionar una experiencia.

Marcano [13] destaca que en la actualidad se le asigna este nombre a un grupo de videojuegos y simuladores cuyo objetivo principal es la formación antes que el entretenimiento y que, además, esta área de desarrollo y creación de videojuegos ha surgido como una manera inteligente de combinar los beneficios de los videojuegos, su poder de penetración en la población y las necesidades de educación y formación efectiva tanto a nivel político-institucional como empresarial y comercial.

Marcano también menciona que las siguientes características son distintivas en los Juegos Serios:

- Están destinados para la educación, el entrenamiento en habilidades determinadas, la comprensión de procesos complejos, sean sociales, políticos, económicos o religiosos; también para publicitar productos y servicios.
- Están vinculados en forma evidente con algún aspecto de la realidad. Esto favorece la identificación del jugador con el área de la realidad que se está representando en el ambiente virtual.
- Constituyen un ambiente tridimensional virtual en el que se le permite una práctica "segura" a los aprendices en algunas áreas. En los casos de entrenamiento, por ejemplo, en el campo militar, se entrena a los soldados a manipular las armas.
- Hay intereses manifiestos en sus contenidos (políticos, económicos, psicológicos, religiosos, etc.).

Además, es importante agregar que todo Juegos Serio es un videojuego, por lo que también se puede definir en ellos las mecánicas, dinámicas, estéticas, una narrativa, una historia, personajes y efectos.

2.4.1 Mecánicas de los Juegos Serios

Como Spinelli [14] indica en su revisión sistemática, la unión entre las Mecánicas de Aprendizaje, las Mecánicas de Juego y el vínculo bidireccional entre ellas (un mapeo) constituyen las Mecánicas de los Juegos Serios.

Además, Spinelli también agrega que no existe un mecanismo útil para especificar las Mecánicas de los Juegos Serios y que un punto de partida razonable para tal fin, es observar cómo se especifican las Mecánicas de Juego en el ámbito de los videojuegos, para luego ver en qué forma se puede hacer lo propio con las Mecánicas de Aprendizaje y el mapeo.

Arnab et al. [15] señalan que las Mecánicas de los Juegos Serios deben reflejar las complejas relaciones entre la pedagogía, el aprendizaje y el entretenimiento/diversión y que, en última instancia, necesitan fusionar las agendas educativas y de juego. Por lo tanto, las Mecánicas de los Juegos Serios en este contexto son los elementos/aspectos del juego que vinculan una práctica pedagógica con una Mecánica de Juego concreta directamente relacionada con las acciones de un jugador. La descripción centrada en el usuario de la Mecánica de Juego representa una alternativa atractiva a la representación centrada en el juego de los patrones de diseño.

2.5 Analíticas de Aprendizaje

Fournier et al. [16] definen las Analíticas de Aprendizaje como la medición, recopilación, análisis y reporte de datos sobre los estudiantes y sus contextos, con el fin de comprender y optimizar el aprendizaje y los entornos en los que se produce.

Pérez [17] indica que las analíticas tienen como aporte sustancial la identificación de patrones de interacción en los diferentes órdenes del acto didáctico; analizando las actividades donde los estudiantes logran los mejores resultados y los recursos utilizados para tal fin. Esto permite escalar la información y realizar el análisis de los contenidos más

eficientes, visualizando las mejores prácticas que aportan al avance del curso y que disminuyen el riesgo académico y/o desvinculación del estudiante.

Además, Pérez menciona que las analíticas aportan a los docentes información sobre el aprendizaje de los alumnos, realizando una correlación cruzada con los datos, mediante la representación de la información en reportes y gráficos.

Sin embargo, Baalsrud Hauge et al. [18] aclaran que un alto rendimiento en un juego no implica necesariamente un aprendizaje efectivo. En general, el juego está intrínsecamente relacionado con el rendimiento, que va con una actitud de lograr hitos y puntajes altos. Por el contrario, el aprendizaje a menudo requiere oportunidades para la reflexión, la repetición informada, pausas e incluso la preparación para cometer errores y aprender de ellos.

Como señalan Freire et al. [19], las Analíticas de Aprendizaje también pueden ayudar a los docentes a identificar a los alumnos con dificultades e incluso ofrecer acciones de remediación específicas para esos estudiantes.

2.5.1 Analíticas de Aprendizaje en Juegos Serios

Massa et al. [20] señalan que la integración de Analíticas de Aprendizaje en el diseño de Juegos Serios ofrece oportunidades para rastrear y analizar datos del comportamiento de los estudiantes sobre la base de su interacción individual o grupal e interpretar el proceso de aprendizaje.

Como indican Alonso-Fernandez et al. [21], sin las Analíticas de Aprendizaje los Juegos Serios en educación son similares a cajas negras: simplemente brindan un estado final que muestra los resultados del juego de los estudiantes, generalmente en forma de métricas simples como el puntaje final del jugador; pero eso no proporciona información sobre el proceso de aprendizaje.

Además, Alonso-Fernandez et al. agregan que abrir la caja puede proporcionar mucha más información sobre el uso de los Juegos Serios y cómo sus jugadores interactúan con ellos. Eventualmente, si el juego está bien diseñado y se capturan los datos de

interacción relevantes, debería ser posible rastrear la evolución del conocimiento de cada jugador en cada parte del juego e identificar las áreas en las que luchan o brillan.

De la revisión sistemática realizada por Kühn y Massa [22], se deduce que no existe una metodología específica para diseñar las Analíticas de Aprendizaje para los Juegos Serios. Sin embargo, Spinelli [14] añade que los autores dejan en claro que las analíticas deben estar fundadas a partir del diseño del Juego Serio y la participación de los stakeholders. Además, aclara que es preciso incorporar en las Mecánicas de Aprendizaje, las analíticas para que, a través del mapeo, se puedan registrar las decisiones del jugador y de esa manera evaluar su desempeño tanto en el juego como en el aprendizaje.

Capítulo 3: Metodología

El modelo de desarrollo utilizado en este proyecto es el modelo de desarrollo incremental. El blog comparasoftware [23] señala que este proceso descompone un proyecto en una sucesión de agregados denominados incrementos que conforman un fragmento de la funcionalidad total del producto. Es un modelo prescriptivo que entrega un componente de trabajo con cada incremento. Cada etapa debe desarrollarse debidamente. Es decir, con sus requisitos, diseño, codificación y, por último, su testeo.

Ortiz [24] agrega que cuando se utiliza un modelo incremental, el primer incremento es a menudo un producto esencial, sólo con los requisitos básicos. Los primeros incrementos son versiones incompletas del producto final, pero proporcionan al usuario la funcionalidad que precisa y también una plataforma para la evaluación. La característica desarrollada en cada etapa se agregará a la funcionalidad llevada a cabo anteriormente. Esto último se repite hasta que el software esté completamente desarrollado.

Además, la selección de la funcionalidad a llevar a cabo es dirigida por el riesgo, es decir, que al momento de elegir que requerimiento se implementa, se prioriza aquellos en los que era es más difícil estimar el tiempo de desarrollo.

Ventajas:

- Mediante este modelo se genera software operativo de forma rápida y en etapas tempranas del ciclo de vida del software
- Es un modelo más flexible, por lo que se reduce el coste en el cambio de alcance y requisitos
- Es más fácil probar y depurar en una iteración más pequeña
- Es más fácil gestionar riesgos
- Cada iteración es un hito gestionado fácilmente

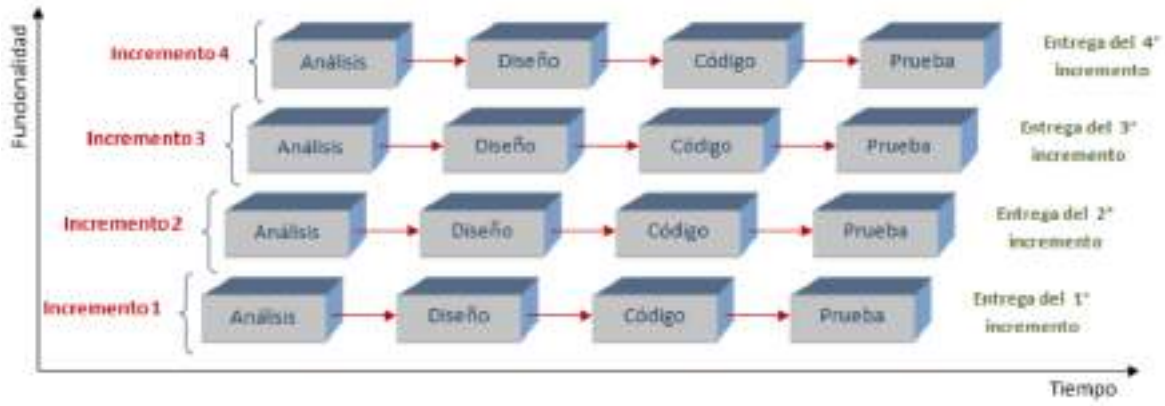


Figura 5: Esquema gráfico que representa el modelo de desarrollo incremental. Adaptado de Intelisoft [25].

El método utilizado en este proyecto es el de comenzar realizando una encuesta a docentes para comprender, a nivel general, lo que se quiere desarrollar. Luego se continúa con entrevistas a la referente funcional, que permiten definir un subconjunto de requerimientos y se comienza por el desarrollo del que tenga un mayor riesgo.

Dicho riesgo es determinado considerando algunos factores como, la posibilidad de llevar a cabo el requerimiento completamente o parcialmente, la facilidad de estimar el tiempo que llevaría terminarlo y además el impacto que tiene el agregar la funcionalidad en el sistema.

Una vez agregadas esas funcionalidades dentro del sistema y validadas por Alina, se comienza con el siguiente incremento, inicialmente definiendo una nueva reunión con la referente funcional para determinar el nuevo subconjunto de requerimientos a desarrollar, y así hasta alcanzar el producto final.

Capítulo 4: Proyecto PTF

4.1 Elicitación y Especificación de Requerimientos

Siguiendo la metodología descripta, se realizó una encuesta a las maestras de la escuela en donde trabaja Alina, la Escuela Primaria N° 63 Constancio Carlos Vigil, que ayudó a comprender el entorno en el cual se desarrolló el proyecto y colaboró al momento de diseñar el juego que se terminó implementando.

Una vez formuladas las preguntas a realizar en la encuesta sobre el Juego Serio a desarrollar, se creó el formulario utilizando Google Forms y luego se envió el link del mismo a Alina para que se lo comparta a los distintos docentes que quieran participar. En total, participaron 7 maestros.

Posteriormente a la encuesta, la elicitación y especificación de requerimientos del proyecto fue abordada mayormente mediante entrevistas realizadas a la referente funcional, Alina Ludueña. Después de cada reunión con la referente funcional, se definían un conjunto de funcionalidades a llevar a cabo y se comenzaba a desarrollar la que se consideraba que tenía el mayor riesgo. Después de agregar las características al proyecto, Alina se encargaba de validarlas, lo que daba comienzo al siguiente incremento del sistema.

Finalmente, se llegó a los siguientes requerimientos, los cuales fueron agrupados formalmente en requerimientos del videojuego, del ingreso al videojuego, de la página, y de las Analíticas de Aprendizaje con el fin de facilitar su lectura.

Requerimientos del videojuego:

- Deberá poder ser usado en dispositivos móviles con sistema operativo Android.
- Tendrá que ocupar poco espacio en memoria y además consumir pocos recursos.
- Será 2D.
- El progreso estará dado en niveles, los cuales permitirán a los alumnos trabajar con letras de palabras como también con palabras completas dependiendo del modo de juego.

- Al iniciar un nivel, el juego mostrará en texto lo que el estudiante deberá realizar para completarlo.
- Todas las letras del videojuego deben ser legibles y estar en imprenta mayúscula.
- En cada nivel habrá letras, palabras o corazones que se moverán de derecha a izquierda y viceversa, dependiendo del modo de juego. Dichas letras y palabras pueden ser correctas o incorrectas en función del nivel y del progreso del estudiante.
- El jugador deberá poder moverse hacia la izquierda o derecha y además deberá poder disparar un proyectil que podrá colisionar con los objetos del nivel.
- Cada nivel empezará con tres (3) vidas representadas con corazones. Hacer colisionar un proyectil con una letra o palabra equivocada hará que se pierda un corazón. Caso contrario, se obtendrá progreso en el nivel. En caso de que se tengan menos de tres (3) vidas, colisionar el proyectil con un corazón hará que el jugador obtenga una vida adicional. Llegar a cero (0) vidas tendrá como resultado la finalización del nivel con un puntaje igual a cero (0).
- Cada nivel contará con un cronómetro a contra reloj con un valor variable en función del modo de juego y la dificultad seleccionada. Si el reloj llega a cero (0), se tendrá como resultado la finalización del nivel con un puntaje igual a cero (0).
- Los niveles tendrán tres (3) dificultades: Fácil, en la que habrá pocos objetos incorrectos posibles y más tiempo, Normal, en la que habrá una mayor cantidad de objetos incorrectos posibles y menos tiempo, y Difícil, en la que habrá aun una mayor cantidad de objetos incorrectos posibles y aún menos tiempo.
- Al terminar un nivel se deberá mostrar una pantalla que muestre el puntaje del jugador. Este puntaje estará dado por la cantidad de corazones

restantes, el tiempo que tardó en terminarlo y en cuantos objetos correctos colisionó con el proyectil.

- Los niveles podrán ser pausados o reiniciados.
- El videojuego tendrá cinco (5) modos de juego, en dos (2) de ellos el jugador deberá atrapar letras (Letras del Nombre, Letras de Palabra) y en los otros tres (3) (Nombre Completo, Palabra Completa, Múltiples Palabras) deberá atrapar palabras.
- En los modos de juego en los que habrá que atrapar palabras, los docentes podrán agregar palabras incorrectas y, además, podrá seleccionar si entre las palabras incorrectas aparecen los nombres de los alumnos que pertenecen a la misma aula que el estudiante que juega.
- Los modos de juego en los que habrá que atrapar palabras se darán finalizados una vez que el estudiante atrape cinco (5) palabras correctas.
- Los distintos modos de juego contarán con distintos parámetros para finalizarlos:
    - **Letras del Nombre:** El jugador deberá armar su nombre en orden disparando proyectiles a las letras que aparecen en pantalla. Por ejemplo, si el nombre del participante es “Mariano”, primero debe arrancar agarrando la “M”, luego la “A” y así hasta llegar a la “O”. En el caso de que el jugador deba agarrar determinada letra y atrape otra, entonces perderá un corazón. Utilizando el ejemplo anterior, si el estudiante tiene que agarrar la letra “R” y agarra la “N” entonces perderá un corazón, a pesar de que la letra aparezca en su nombre.
    - **Nombre Completo:** El jugador deberá hacer colisionar el proyectil con su propio nombre. El nivel terminará cuando se agarre su nombre unas cinco (5) veces. Solo una palabra es correcta.
    - **Letras de Palabra:** Se utiliza el mismo concepto que el modo Letras del Nombre, con la excepción de que la palabra que se deberá completar será una seleccionada por un docente.

    - **Palabra Completa:** Se utiliza el mismo concepto que el modo Nombre Completo, con la excepción de que la palabra a atrapar será una seleccionada por un docente.
    - **Múltiples Palabras:** Se utiliza el mismo concepto que el modo Nombre Completo, con la excepción de que habrá más de una palabra que es correcta.
- Al terminar un nivel, se deberá mandar la información necesaria para luego mostrar las analíticas a los docentes.

Requerimientos del ingreso al videojuego:

- Deberá contar con conexión a internet para permitir el ingreso de un alumno.
- A las aulas de la Unidad Pedagógica se les asignará un código de identificación único que los alumnos luego usarán en su primer ingreso al videojuego para indicar el aula a la que pertenecen.
- El primer ingreso al videojuego solicitará al estudiante su nombre, apellido y el código del aula a la que pertenece.
- Si un alumno realiza un ingreso con un código de aula válido, un docente asignado a dicha aula deberá admitir su ingreso para que el mismo pueda acceder al videojuego.
- Si el ingreso de un alumno fue aprobado por un docente, el mismo podrá jugar a todos los ejercicios/niveles que el aula ofrece.

Requerimientos de la página:

- La página deberá ser completamente gratuita.
- La página deberá tener un acceso y creación de usuarios que los docentes que permitirá a los docentes utilizar el sistema.
- Luego de que un usuario se haya registrado, un administrador deberá aprobar su ingreso al sitio y además asignar las aulas en las cuales es docente.

- En caso de que un usuario haya ingresado mal su nombre o apellido, un administrador podrá modificarlo.
- Un usuario con su ingreso aprobado, es decir, un docente, podrán ver, crear, editar y eliminar niveles del juego de las aulas en las que esté asignado. Además, un docente podrá ver la lista de alumnos del aula, las analíticas de cada alumno y además podrá editar o eliminar a los estudiantes.
- Los docentes también podrán ver el código único de las aulas en las que está asignado y podrán modificarlo.

Una vez definidos, desarrollados y aprobados los requerimientos del sistema general, se llevó a cabo una última reunión con la referente funcional en la cual se definieron los requerimientos de las Analíticas de Aprendizaje, y posteriormente, desarrollados y validados.

Requerimientos para mostrar las Analíticas de Aprendizaje:

- Un gráfico de barras mediante el cual en el eje de ordenadas se mostrará el número de proyectiles que colisionaron con letras o palabras correctas e incorrectas y en el eje de abscisas el número de intento.
- Un gráfico de barras mediante el cual en el eje de ordenadas se mostrará la puntuación final de cada intento y en el eje de abscisas el número de intento.
- Un gráfico de tortas que represente la tasa de victorias del alumno en el nivel.

4.2 Diseño

Luego de determinar las tecnologías que se utilizaron como se puede ver en el Anexo 1, se diseñó la arquitectura general del proyecto, como se ve en la Figura 6.

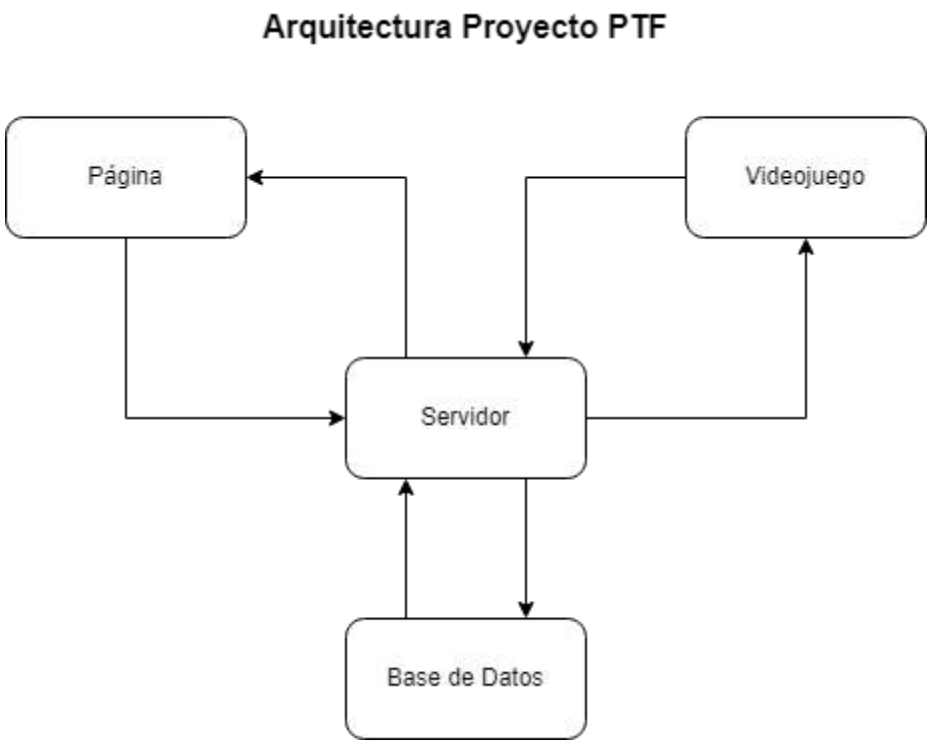


Figura 6: Arquitectura del proyecto.

Tanto la página como el videojuego funcionan como clientes y se comunican con el servidor realizando peticiones HTTP para acceder y utilizar información de la base de datos. Los datos enviados en estas solicitudes se dan mayoritariamente en formato JSON.

La página web cuenta con distintos roles de acceso, utilizados para determinar las vistas a las que tienen autorización las personas que inicien sesión. Un usuario con el rol de invitado es una persona que creó una cuenta en la página pero que todavía no puede acceder a las aulas ya que no tiene permisos de docente debido a que aún no ha sido aceptado por un administrador.

En cambio, un docente, es un usuario de la página que ha sido aceptado por un administrador y al cual se le han asignado aulas. Las personas con dicho rol solo podrán administrar los ejercicios y alumnos de las aulas en la que está asignado. Esto no es así para los usuarios con rol de administrador, que pueden ver todas las aulas en todo momento y, además, tienen acceso a la lista de usuarios del sistema ya que son los encargados de gestionar quién es un docente y quién no.

Con respecto al videojuego, existen únicamente dos roles de acceso, los estudiantes y no estudiantes. Una persona con el rol de videojuego es alguien que se ha registrado como estudiante en el mismo, pero que todavía no ha sido aceptado como tal por un docente y, por lo tanto, no podrá acceder al videojuego. Una vez la persona con este rol sea asignada como estudiante por un docente, podrá acceder al videojuego y aventurarse en los niveles del mismo.

4.2.1 Diseño del servidor

El servidor fue implementado en Spring Boot y posee una arquitectura REST, la cual es definida en Wikipedia [26] como una arquitectura de desarrollo web que utiliza directamente HTTP para obtener datos o indicar la ejecución de operaciones sobre los datos, en cualquier formato (XML, JSON, etc.) sin las abstracciones adicionales de los protocolos basados en patrones de intercambio de mensajes. Inicialmente, el término REST se refería al siguiente conjunto de principios:

- **Un protocolo cliente/servidor sin estado:** cada mensaje HTTP contiene toda la información necesaria para comprender la petición. Como resultado, ni el cliente ni el servidor necesitan recordar ningún estado de las comunicaciones entre mensajes.
- **Un conjunto de operaciones bien definidas que se aplican a todos los recursos de información:** HTTP en sí define un conjunto pequeño de operaciones, las más importantes son POST, GET, PUT y DELETE. Con frecuencia estas operaciones se equiparan a las operaciones CRUD en bases de datos (CLAB en castellano: crear, leer, actualizar, borrar) que se requieren para la persistencia de datos, aunque POST no encaja exactamente en este esquema.
- **Una sintaxis universal para identificar los recursos:** en un sistema REST, cada recurso es direccionable únicamente a través de su URI.
- **El uso de hipermedios, tanto para la información de la aplicación como para las transiciones de estado de la aplicación:** la representación de este estado en un sistema REST son típicamente HTML, XML o JSON. Como

resultado de esto, es posible navegar de un recurso REST a muchos otros, simplemente siguiendo enlaces sin requerir el uso de registros u otra infraestructura adicional.

4.2.2 Diseño del videojuego

Para poder desarrollar el videojuego, previamente se diseñaron y detallaron sus componentes como se indica a continuación:

Componentes del videojuego:

- **Historia**: El juego consiste en un arquero que dispara sopapas con un arco agarrando letras o palabras, según el modo de juego, para completar el nivel antes de que el tiempo se acabe. En caso de que el modo de juego sea el de completar una palabra, el jugador debe dispararles a las letras en orden hasta completar la palabra. En el otro modo de juego, el jugador debe dispararles a las palabras hasta acertar cierta cantidad de veces.
- **Narrativa**: A medida que el arquero va disparando sopapas, se van recogiendo letras o palabras. Si el objeto al que se le disparó es el correcto, se consigue progresar en el nivel y además sumando puntos. Pero si el objeto al que se le disparó es incorrecto, se pierde un corazón. Perder los tres corazones o quedarse sin tiempo tiene como resultado la finalización del nivel con un puntaje igual a cero.
- Mecánicas:
- **Disparar**: se lanza una sopapa que puede colisionar con una letra, una palabra o un corazón. Para ello se debe apretar por encima de la valla.
- **Caminar**: el personaje se mueve hasta la posición apretada. Para ello se debe apretar por debajo de la valla.
- **Estética**: El videojuego cuenta con una estética de fantasía medieval y la sensación principal que genera es la del desafío debido a que los niveles deben ser completados a contrarreloj.
- **Efectos**: Tanto en el menú principal como dentro de un nivel al jugador se le presenta una música acorde a la sensación que se quiere generar

(relajación en el menú principal, tensión al jugar un nivel). Además, cuenta con varios efectos de sonido como ayudas auditivas. Por ejemplo, al recoger una letra o palabra incorrecta se reproduce un sonido feo para la audición mientras que si es correcta se reproduce un sonido placentero.

- **Personajes**: Solo existe un personaje protagonista y es el que el jugador deberá controlar moviendo y disparando sopapas para completar los niveles.

4.2.3 Diseño de analíticas

Para el diseño de las analíticas que son mostradas en la página web, inicialmente se elaboró el diagrama de las mecánicas de juego como se ve en la Figura 7.

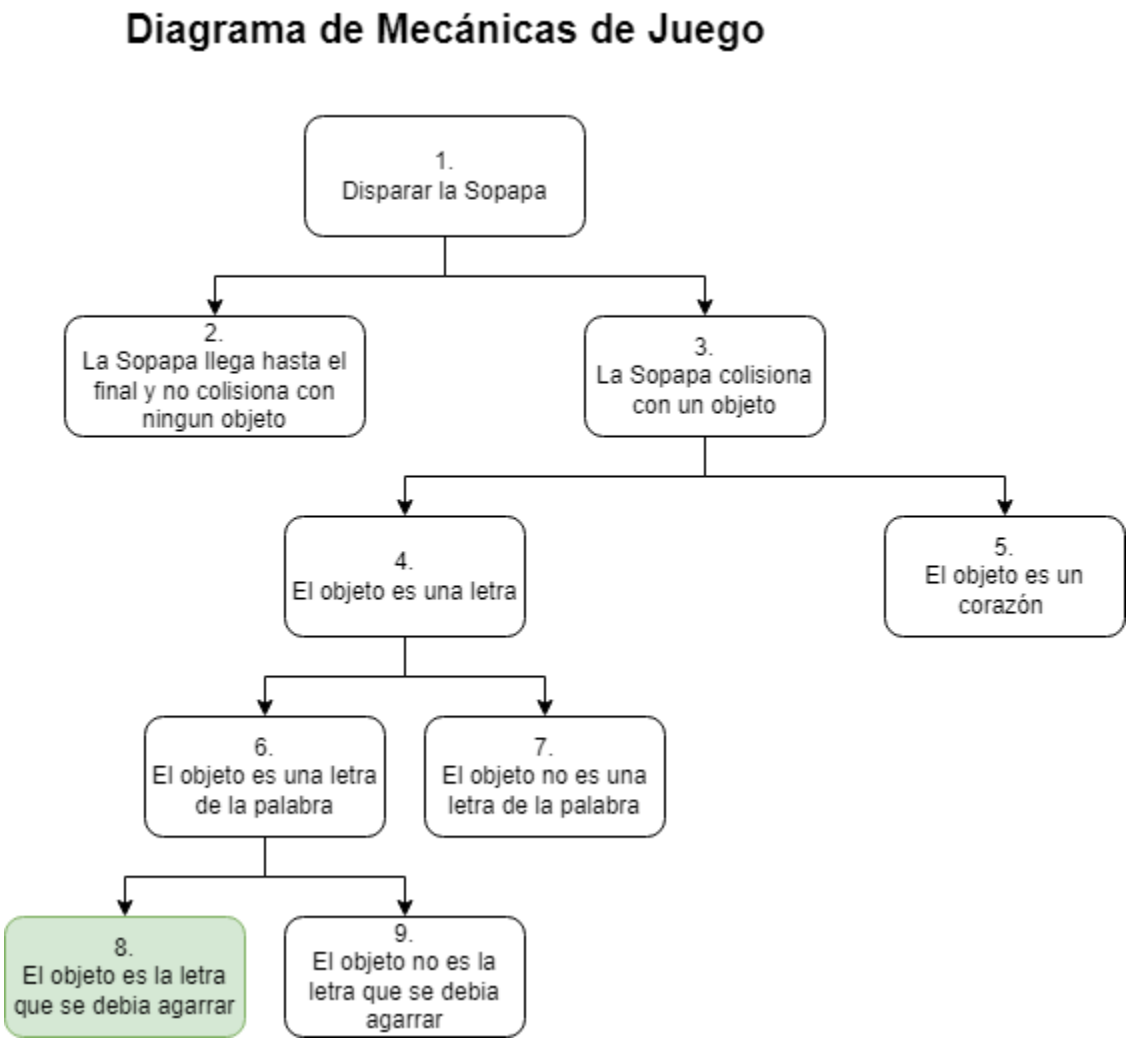


Figura 7: Diagrama de mecánicas de juego.

Luego, se armó una tabla que muestra el progreso de las Mecánicas de Juego como se puede ver en la Figura 8.

Progreso de las Mecánicas de Juego	
Número de Mecánica de Juego	**Progreso de la Mecánica de Juego**
2	La sopapa abandonará la pantalla y no habrá ninguna consecuencia.
5	El jugador obtendrá un corazón en caso tener dos o menos.
7	El jugador perderá un corazón*.
8	El jugador completará su nombre con una letra. Al completar todo su nombre, el jugador completará el nivel.
9	El jugador perderá un corazón*.

Figura 8: Tabla que muestra el progreso de las Mecánicas de Juego.

*El jugador al perder todos los corazones perderá el nivel y deberá comenzar de vuelta desde la primera letra.

También se armaron los diagramas de las Mecánicas de Aprendizaje. En la Figura 9 se puede ver la Mecánica de Aprendizaje que se denominó como “Reconocer las Letras de la Palabra”.

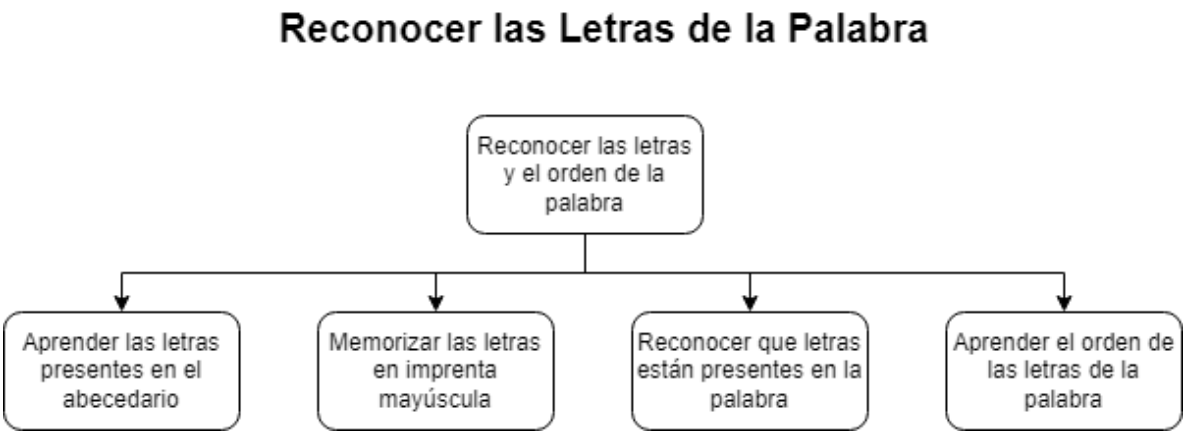


Figura 9: Mecánica de aprendizaje “Reconocer las Letras de la Palabra”.

Luego se definió el progreso en las Mecánicas de Aprendizaje y se las relacionó en una tabla con las Mecánicas de Juego para obtener las Mecánicas de los Juegos Serios como se puede observar en la Figura 10.

Mapeo entre las Mecánicas de Juego y las Mecánicas de Aprendizaje			
Mecánica de Aprendizaje general	**Mecánica de Aprendizaje**	**Número de Mecánica de Juego**	**Progreso de la Mecánica de Aprendizaje**
Reconocer las letras y el orden de la palabra	Aprender las letras presentes en el abecedario	4	El estudiante aprenderá a reconocer las distintas letras del abecedario.
	Memorizar las letras en imprenta mayúscula	4	El estudiante aprenderá a reconocer las distintas letras del abecedario que están escritas en imprenta mayúscula.
	Reconocer que letras están presentes en el nombre	6, 7	El estudiante aprenderá a escribir, deletrear su nombre y distinguir de las letras que no están presentes en el mismo.
	Aprender el orden de las letras del nombre	8, 9	El estudiante aprenderá a escribir y deletrear su nombre.

Figura 10: Tabla que muestra el mapeo entre las Mecánicas de Juego y las Mecánicas de Aprendizaje.

La analítica desarrollada que indica cuantas sopapas golpearon letras correctas e incorrectas permite distinguir si un alumno reconoce las letras presentes en una palabra y el orden de las mismas. Por ejemplo, si un estudiante que no cumple con este requisito y juega varias veces el mismo nivel, se prevé que en los primeros intentos la cantidad de sopapas que golpean letras incorrectas sean varias y lentamente este valor vaya disminuyendo a medida que va a aprendiendo.

Como se esperaría que un estudiante que está aprendiendo cada vez logre mejores puntajes (ya que este valor depende de la cantidad de corazones restantes, el tiempo que tardó en terminarlo y contra cuantos objetos correctos colisionó el proyectil) se desarrolló otra analítica que indica el puntaje final de cada intento en un nivel determinado que además ayuda a complementar la primera analítica mencionada.

Por último, también se determinó que un gráfico de tortas mediante el cual se indique la tasa de victorias sería otra analítica muy útil debido a que permitiría identificar si un alumno está aprendiendo o no. Por ejemplo, de un estudiante con una tasa de victorias cercana al 100% y muchos intentos, se podría deducir que ya reconoce del ejercicio y no tiene problemas para resolverlo. En cambio, alguien que tiene una menor tasa de victorias,

lo más probable es que en los primeros intentos haya perdido y al aprender haya empezado a ganar.

De esta forma se determinó que, analizando los gráficos de dichas analíticas previamente mencionadas, se puede llegar a determinar que, si un alumno que juega varias veces a un nivel determinado del videojuego, está realmente aprendiendo, ya es un tópico que tiene estudiado, o tiene problemas en el aprendizaje.

4.3 Implementación

4.3.1 Página PTF

Para la página se decidió utilizar una de las librerías de Vue con más componentes, una gran comunidad y que además permite crear interfaces responsives, llamada Element UI (https://element.eleme.io/).

En la página principal se codificó una ventana que permite a los docentes identificarse como se muestra en la Figura 11.

Figura 11: Vista de inicio de sesión de la página web.

En caso de que un docente que quiere ingresar al sistema no tenga usuario, el mismo puede apretar en el botón “Crear Usuario” y esto les muestra otra vista en la cual pueden registrarse, como se observa en la Figura 12, y que luego les permitirá iniciar sesión.

Figura 12: Vista de registro de la página web.

Cuando un usuario inicia sesión en la página se guarda en el almacenamiento local del navegador, un token que recibe del servidor que luego utiliza para poder realizar las peticiones con el nombre de “access_token”, como se puede observar en la Figura 13.

Figura 13: Variable “access_token” almacenada en el “Local Storage” del navegador.

Luego de que un usuario inicia sesión, es redirigido a una de las vistas en función de su rol. En caso de que el usuario tenga el rol de invitado, el mismo es direccionado hacia la página que indica que el usuario no es un docente, como se puede ver en la Figura 14.

Figura 14: Vista de un usuario con el rol “Guest” luego de iniciar sesión en la página web.

En cambio, si el usuario tiene el rol de docente, es redirigido a la página que muestra las aulas a las que está asociado el maestro, y, además, en la barra de navegación solo se

puede dirigir a la página de aulas y, dentro de ésta, ver aquellas a las que está asignado. En la Figura 15 se puede ver esta vista.

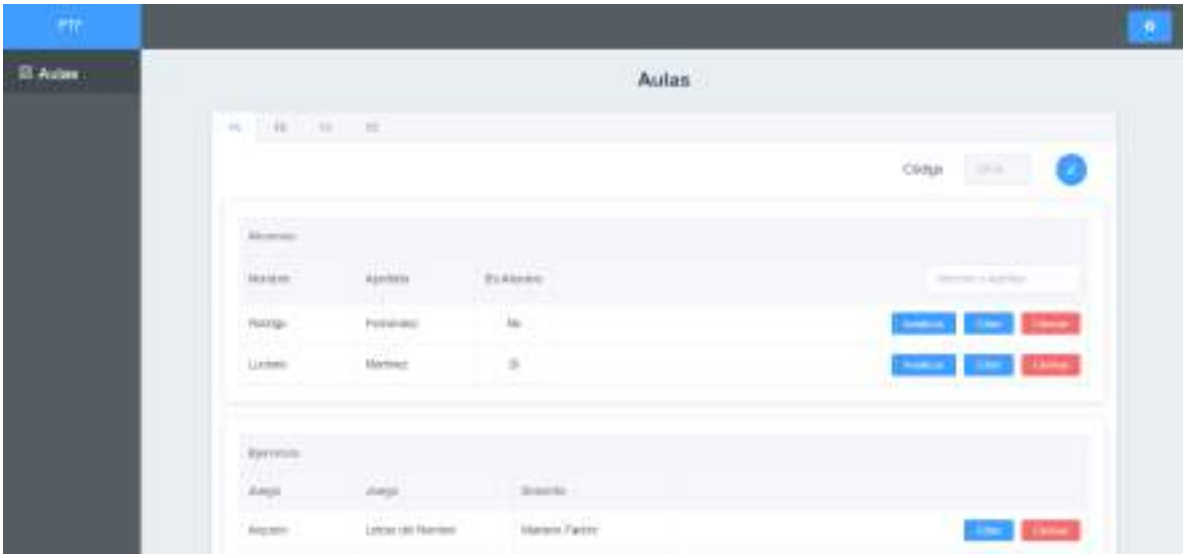


Figura 15: Vista de administración de aulas de la página web.

Por otra parte, si el usuario tiene rol de administrador, es redirigido a la página de administración de usuarios, que se puede observar en la Figura 16, lo que le permite editar y eliminar usuarios registrados. Desde la barra de navegación se puede acceder tanto a la página de docentes como a la de aulas. En la página de aulas se pueden ver todas las que hay en el sistema. La vista de edición de docentes está mostrada en la Figura 17.

Figura 16: Vista de administración de usuarios de la página web.

Figura 17: Vista de edición de usuarios de la página web.

Desde la vista de aulas, los usuarios con los permisos de docente o administrador pueden navegar entre las distintas aulas, cambiar sus códigos, editar, eliminar o ver las analíticas de los usuarios registrados desde el videojuego y además son capaces de agregar, editar y eliminar niveles/ejercicios.

Para la edición del código del aula, se debe tocar el botón a la derecha del texto "Código", lo que permite modificar el valor que existe dentro del "input" y luego aceptar los cambios o cancelarlos. En la Figura 18 se puede apreciar este comportamiento.

Figura 18: Sección en donde se puede modificar el código de un aula que se encuentra en la vista de administración de aulas.

La edición de los alumnos permite seleccionar el aula a la que pertenecen, el nombre y apellido, y además aceptar o no al estudiante como un usuario del videojuego. Esta vista se puede ver en la Figura 19.

Figura 19: Vista de edición de estudiantes de la página web.

Las analíticas se muestran para cada ejercicio que hay en el aula. Los gráficos fueron hechos mediante la librería vue-chartjs y se pueden apreciar en la Figura 20.

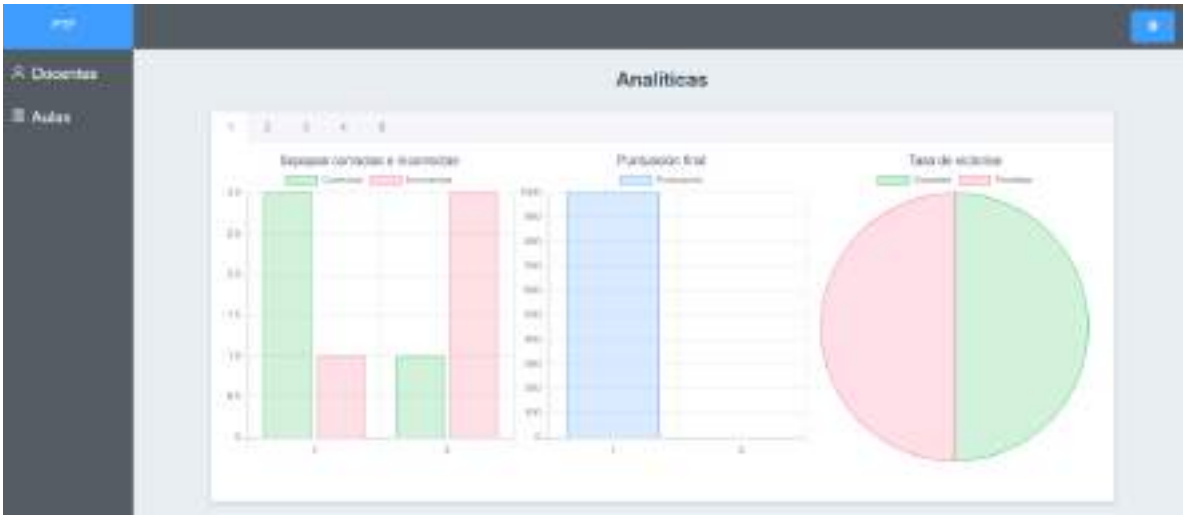


Figura 20: Vista de analíticas de un estudiante de la página web.

En cuanto la adición y edición de los ejercicios, las opciones son visibles y dinámicas en función del modo de juego. En la Figura 21 y la Figura 22 se observa este comportamiento.

Figura 21: Vista de agregar un ejercicio de la página web.

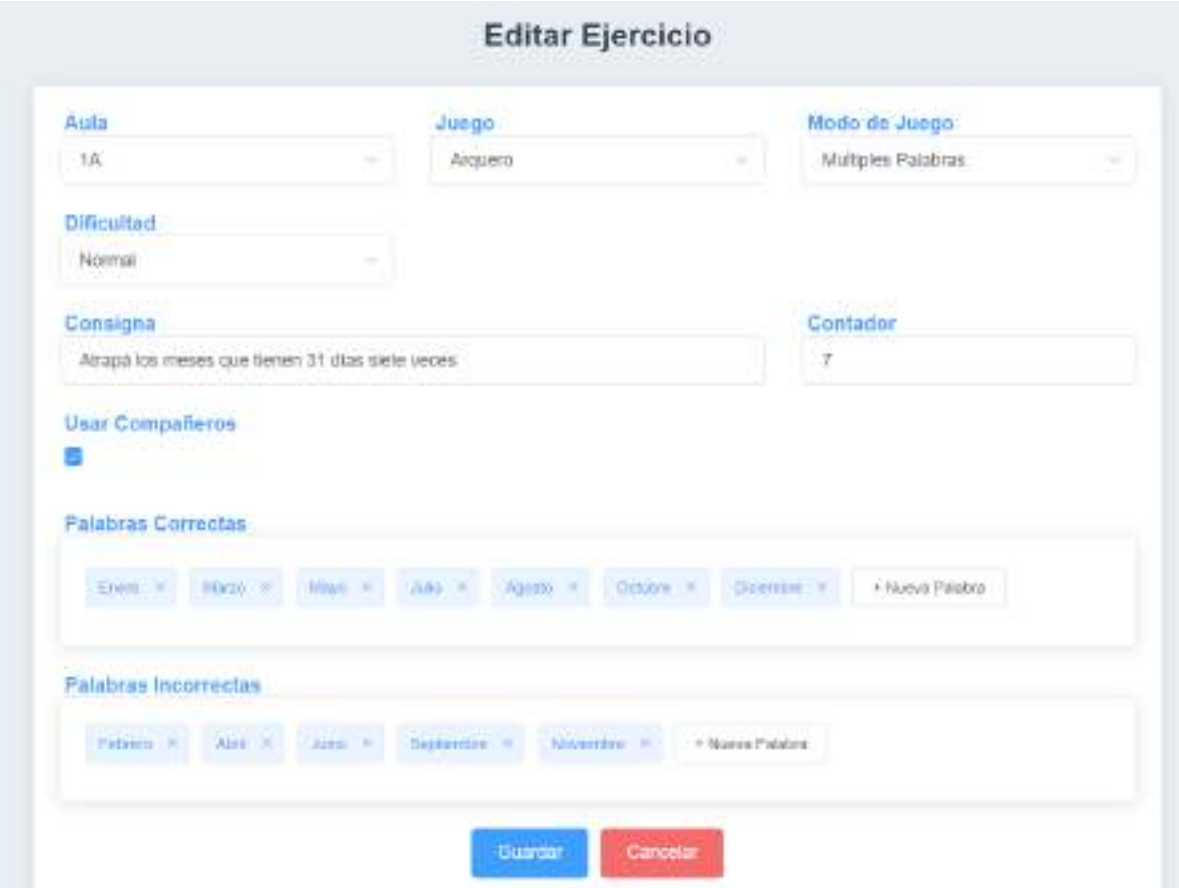


Figura 22: Vista de edición de ejercicios de la página web.

4.3.2 Juego PTF

4.3.2.1 Arquitectura del videojuego

En cuanto al videojuego codificado en Godot, se creó un nodo raíz “SystemManager” del cual heredan todas las interfaces, al cual se le agregó el script “SystemManager.gd”, que es utilizado principalmente para manejar que interfaces son visibles y cuáles no. Esta interfaz es mostrada en la Figura 23.

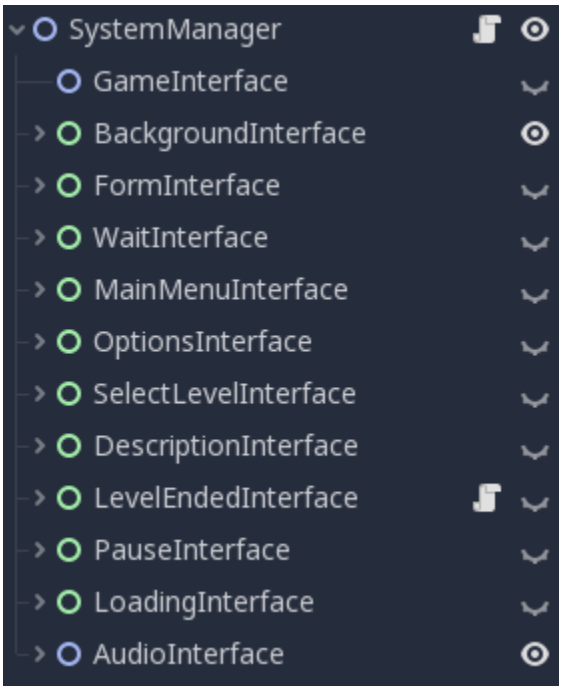


Figura 23: Nodos interfaz utilizados en el videojuego.

Al iniciar el videojuego por primera vez, la interfaz que se observa es "FormInterface". En esta interfaz los estudiantes ingresan el código del aula a la que pertenecen y su nombre y apellido, como se puede ver en la Figura 24.

Figura 24: Interfaz de formulario del videojuego.

Una vez la información es ingresada, se muestra la interfaz “WaitInterface”, como se ve en la Figura 25, que indica que el usuario tiene que esperar hasta que un docente habilite su ingreso al videojuego y, por ende, pueda acceder a la lista de niveles/ejercicios.

Figura 25: Interfaz de espera del videojuego.

Una vez el estudiante sea aceptado en el sistema, se muestra la interfaz “MainMenuInterface”, es decir, el menú principal del videojuego, como se observa en la Figura 26.

Figura 26: Interfaz del menú principal del videojuego.

Al apretar en el botón “Opciones”, se muestra el aula a la que está asignado el estudiante, además de su apellido y nombre, como se ve en la Figura 27.

Figura 27: Interfaz de opciones del videojuego.

Si en cambio se selecciona la opción "Jugar" se muestra la interfaz "SelectLevelInterface", que se observa en la Figura 28, desde la cual se puede elegir un ejercicio/nivel para aprender.

Figura 28: Interfaz de selección de nivel del videojuego.

Apretar en uno de los botones con números hace visible el nodo "DescriptionInterface" que indica la consigna del nivel como se ve en la Figura 29.

Figura 29: Interfaz de consigna que indica que hay que hacer para completar el nivel.

Luego de que se aprete en el botón continuar, el alumno puede jugar el ejercicio seleccionado. En la Figura 30 se aprecia el modo de juego "Letras de Palabra".

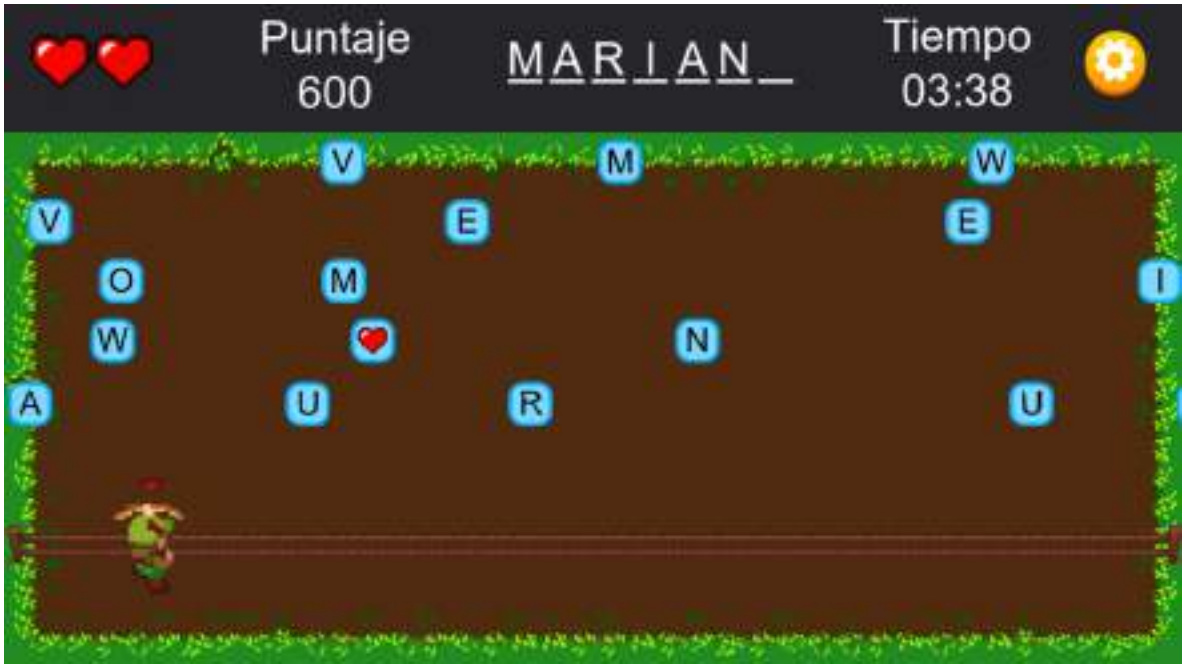


Figura 30: Interfaz de juego configurada para el modo de juego de "Letras de Palabra".

Una vez en el nivel, el estudiante puede apretar en el botón que está arriba a la derecha para hacer visible la interfaz “PauseInterface”, desde la cual el alumno es capaz continuar con el nivel, reiniciarlo o volver al menú principal. Esta interfaz se observa en la Figura 31.

Figura 31: Interfaz de pausa del videojuego.

Una vez el estudiante termina de jugar un nivel, ya sea habiéndolo ganado o perdido, se hace visible la interfaz “LevelEndedInterface” en la cual se muestra el puntaje final del nivel, como se muestra en la Figura 32. Al apretar en el botón “Continuar”, se hace visible el menú de los niveles como se ve en la Figura 28.

Figura 32: Interfaz de nivel terminado del videojuego.

Además de los nodos previamente mencionados, hay otros que siempre son visibles, ya que son utilizados en varias vistas. Dentro del nodo “BackgroundInterface” se encuentra el fondo que se puede ver en los menús.

La interfaz “AudioInterface” se usa para reproducir la música y los distintos sonidos que existen en los menús y dentro de los niveles del videojuego. Esta interfaz se muestra en la Figura 33.

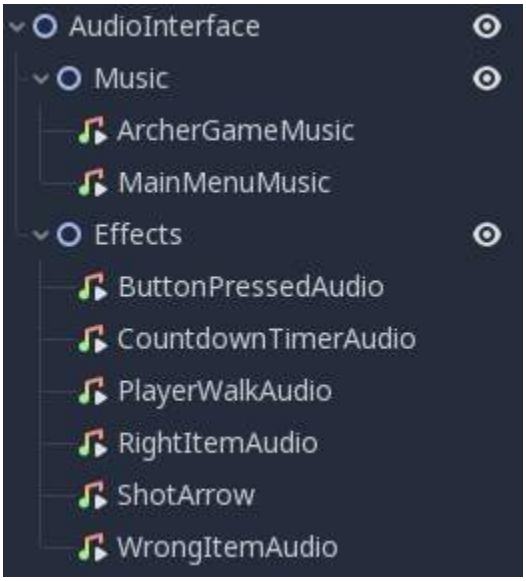


Figura 33: Nodos utilizados para reproducir sonidos dentro de la interfaz de audio.

Por último, el nodo “LoadingInterface” se muestra cuando se le pide información al servidor y el estudiante debe esperar una respuesta, como se observa en la Figura 34. Estas peticiones ocurren cuando el estudiante se registra en “FormInterface”, cuando se inicia el videojuego y se quiere comprobar si el usuario existe en la base de datos, y además cuando se accede a la interfaz “SelectLevelInterface” por primera vez para actualizar los niveles que existen del aula.

Figura 34: Interfaz utilizada cuando el juego debe esperar una respuesta del servidor para continuar.

Al iniciar el videojuego, lo primero que se hace es ver si existe el archivo en donde se guarda la información del alumno. En caso de que exista, se realiza una petición al servidor con el identificador del estudiante para traer los posibles cambios que haya hecho un docente sobre el mismo.

Si el servidor responde con la información del alumno (o en caso de que no haya respuesta y la información del alumno no ha sido vulnerada), se selecciona la interfaz “MainMenuInterface” (o la interfaz “WaitInterface” en caso de que el usuario todavía no haya sido aceptado como estudiante por un docente) y si hubo respuesta, se almacena la información actualizada en un archivo para su uso posterior y, además, se le enviará al servidor los eventos que el estudiante haya realizado jugando sin conexión a internet (en caso de que exista el archivo), para luego poder mostrar las analíticas a los docentes.

Si en cambio el servidor responde indicando que el identificador del alumno no existe en la base de datos, debido a que el archivo de configuración ha sido vulnerado o

porque un docente eliminó al usuario desde la página, la interfaz seleccionada es “FormInterface” para que el estudiante vuelva a registrarse.

Cuando un usuario desde la vista “MainMenuInterface” aprieta en el botón “Jugar”, se le solicita al servidor la lista de ejercicios del aula correspondiente para que el alumno pueda acceder a los niveles. Si el servidor no responde con esta petición, se intenta cargar el archivo en el cual se almacenan la información de los ejercicios. Si en cambio, si hay una respuesta, se actualiza el archivo en el cual se persisten los niveles.

En caso de que un alumno seleccione un nivel en la interfaz “SelectLevelInterface”, se carga el nivel de forma dinámica en función de los parámetros del nivel. Hasta el momento se desarrollaron 5 modos de juego que pueden ser agrupados en 2 tipos, en función de si el alumno tiene que juntar letras (modos “Letras del Nombre” y “Letras de Palabra”, mostrados en la Figura 30) o juntar palabras (modos “Nombre Completo”, “Palabra Completa” y “Multiples Palabras”, como se muestra en la Figura 35).

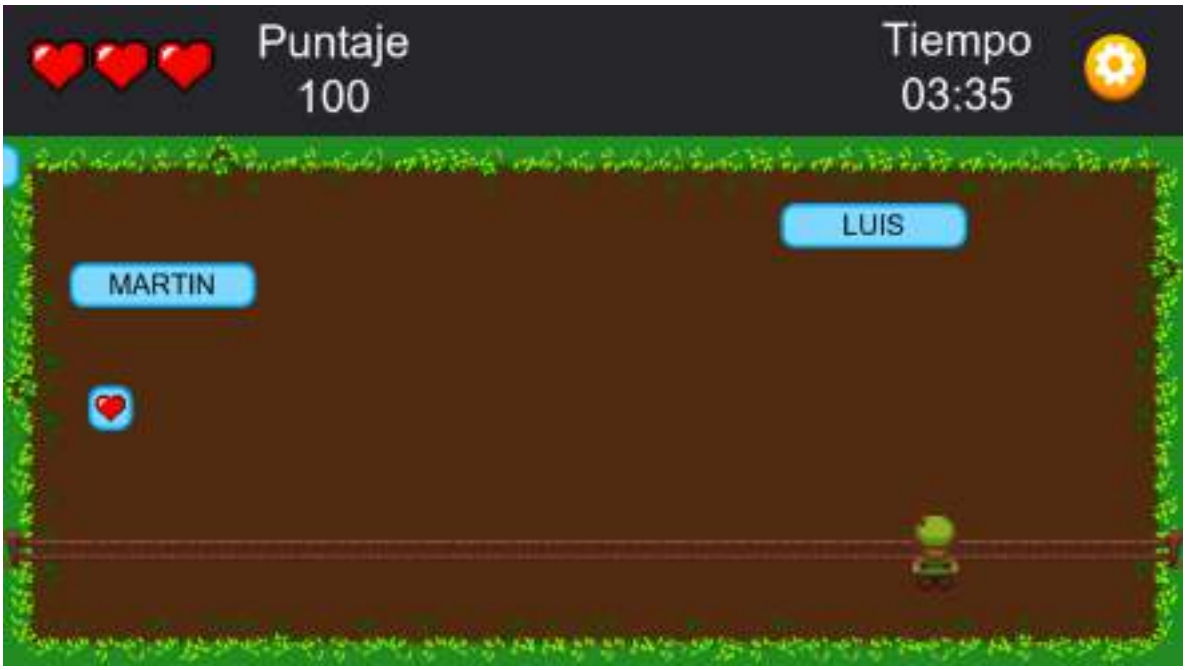


Figura 35: Interfaz de juego configurada para el modo de juego de “Nombre Completo”.

Una vez el usuario inicia un nivel, el mismo puede mover al arquero apretando debajo de la valla y puede disparar sopapas, con las que junta letras o palabras, apretando por encima de la cerca.

El estudiante puede terminar un nivel de varias formas. Una manera es ganándolo, es decir, agarrando todas las letras o juntando cierta cantidad de palabras, en función del modo de juego. Otra manera es perdiéndolo, y es algo que ocurre cuando, el nivel se queda sin tiempo o, cuando se pierden todos los corazones.

La puntuación final del nivel está dada en función de los corazones restantes al momento de terminar el nivel, la cantidad de letras o palabras correctas agarradas, y el tiempo restante. En caso de que el nivel se pierda, la puntuación final es igual a cero.

Una vez el nivel se termine, se hace una solicitud al servidor enviando los datos del evento para guardarlo en la base de datos. En caso de que no haya respuesta, la información es añadida a un archivo y, la próxima vez que el estudiante abra el juego, se envían todos los datos del fichero al servidor.

4.4 Puesta en Funcionamiento

Una vez terminada la codificación de la versión final del proyecto, se subió la página y el servidor al servicio de hosting web Heroku. El motivo por el que se escogió subir la página en ese sitio fue principalmente debido a experiencias previas utilizándolo, además de que es gratuito y fácil de usar.

En cuanto al videojuego, se generó un archivo con extensión ".apk", el cual fue entregado a la referente funcional Alina Ludueña para que luego ella los distribuya entre los docentes que quieran participar del sistema y los distintos alumnos que quieran entretenerse y aprender con el videojuego.

Como es normal que en este tipo de proyectos se encuentren errores luego de su entrega a la referente funcional, se proporcionó un mes previo al inicio de clases para que Alina y los docentes utilicen el sistema con el fin de encontrar problemas en la programación y así tener un sistema más robusto y a prueba de fallos.

Finalmente, una vez acabada la fase de testeo y arreglados los errores, se reinició la base de datos de Heroku, dejándola sola con los datos iniciales (roles del sistema, aulas,

etc.) y se le entregó una versión final del videojuego a Alina para que distribuya entre los alumnos que quieran participar en el videojuego.

4.5 Gestión del Proyecto

4.5.1 Plan de Trabajo estimado

La planificación del proyecto realizada en mayo del 2021, establecía las siguientes tareas:

1. Escritura de las entrevistas a utilizar para determinar requerimientos específicos del proyecto.
2. Entrevistas para definir los requerimientos específicos del proyecto.
3. Análisis de los requerimientos específicos del proyecto a partir de las entrevistas.
4. Diseño de un videojuego que permita satisfacer los requerimientos.
5. Reuniones sobre el diseño del videojuego.
6. Posibles cambios en el diseño del videojuego.
7. Desarrollo de una versión prototipo del videojuego.
8. Lanzado del prototipo.
9. Reuniones para mostrar el prototipo.
10. Posibles cambios en el diseño del videojuego final.
11. Desarrollo del videojuego final.
12. Desarrollo de la interfaz para ver analíticas.
13. Lanzado de la versión final del videojuego.
14. Arreglo de posibles bugs.
15. Escritura del trabajo escrito.
16. Envío del trabajo escrito al director.
17. Devolución del trabajo escrito y correcciones.
18. Envío del trabajo escrito junto con sus correcciones al director.
19. Revisión final del trabajo escrito con el director.

La planificación en el tiempo fue estimada en 7 meses y se refleja en el diagrama de Gantt mostrado en la Figura 36.

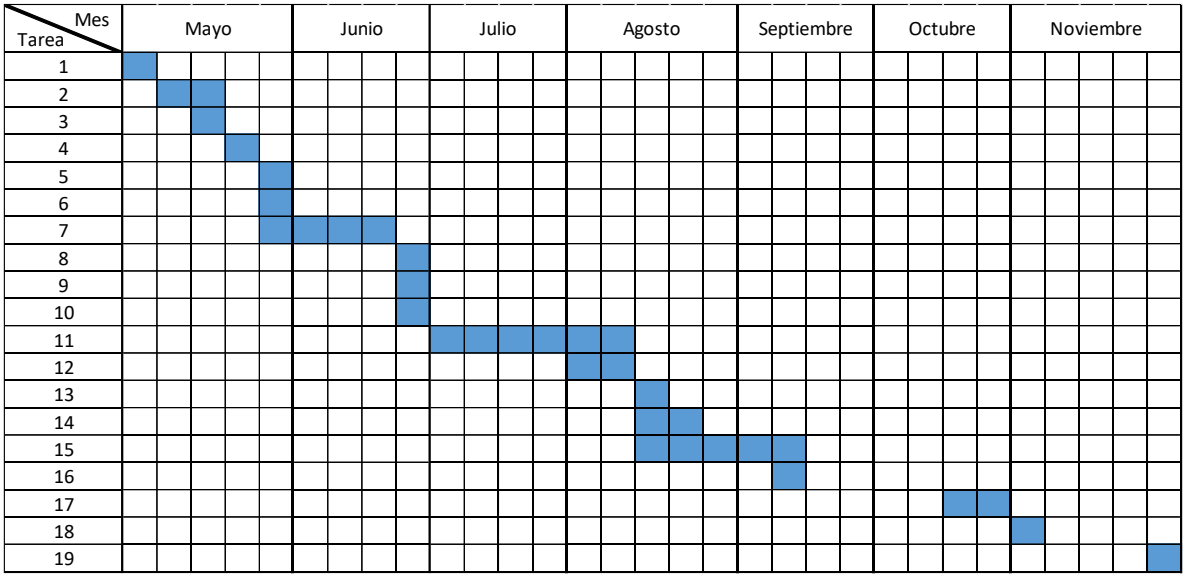


Figura 36: Diagrama de Gantt estimado del Proyecto.

4.5.1.1 Presupuesto del proyecto estimado

Para determinar el costo total estimado del proyecto, se definió para cada tarea el rol principal a cumplir junto con el precio en función de la hora, valor que se obtuvo desde la página del Consejo Profesional de Ciencias Informáticas de la Provincia de Buenos Aires para los valores de marzo 2022 (https://www.cpciba.org.ar/honorarios/page/). Finalmente, estos valores fueron convertidos al precio del dólar de marzo de 2022, el cual era de 200 pesos argentinos en ese momento.

Costo Tareas del Proyecto				
Número de Tarea	**Rol Principal**	**Horas [Hs]**	**Costo hora [$/Hs]**	**Costo Tarea [$]**
1	Analista Funcional	5	3,32	16,60
2	Analista Funcional	10	3,32	33,21
3	Analista Funcional	1	3,32	3,32
4	Diseñador de Videojuegos	4	6,01	24,05
5	Diseñador de Videojuegos	2	6,01	12,02
6	Diseñador de Videojuegos	1	6,01	6,01
7	Desarrollador de Videojuegos	120	6,75	810,56
8	Desarrollador de Videojuegos	1	6,75	6,75
9	Desarrollador de Videojuegos	1	6,75	6,75
10	Desarrollador de Videojuegos	1	6,75	6,75
11	Desarrollador de Videojuegos	150	6,75	1.013,21
12	Programador de Páginas Web	30	4,36	130,87
13	Desarrollador de Videojuegos	1	6,75	6,75
14	Desarrollador de Videojuegos	30	6,75	202,64
Total		357		2.279,51

Figura 37: Presupuesto del proyecto estimado.

En la Figura 37 podemos ver que la cantidad de horas necesarias para llevar a cabo el proyecto es de 357 y el costo total del mismo es de $2.279,51 dólares.

4.5.2 Plan de Trabajo ejecutado

A los efectos de evaluar el proyecto en su finalización, se fueron registrando las diferentes incidencias del proyecto, contabilizando en una planilla de cálculo el comienzo y fin de cada tarea.

Con estos registros, al finalizar el proyecto se realizó de nuevo el diagrama de Gantt, pero esta vez con los plazos de tiempo ejecutados, como se puede ver en la Figura 38.

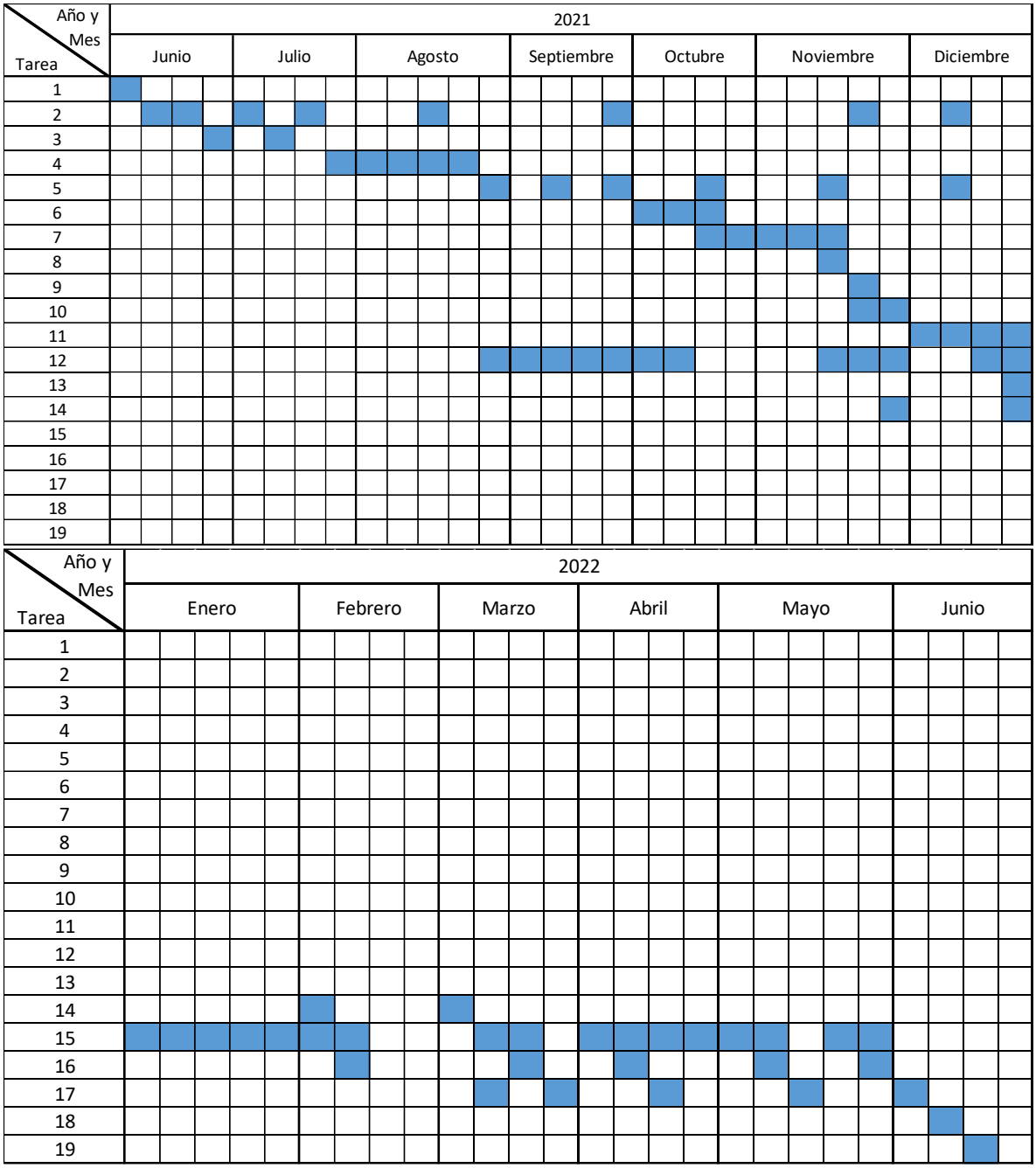


Figura 38: Diagrama de Gantt ejecutado del proyecto.

4.5.2.1 Presupuesto ejecutado

Costo Tareas del Proyecto				
Número de Tarea	**Rol Principal**	**Horas [Hs]**	**Costo hora [$/Hs]**	**Costo Tarea [$]**
1	Analista Funcional	5	3,32	16,60
2	Analista Funcional	20	3,32	66,41
3	Analista Funcional	2	3,32	6,64
4	Diseñador de Videojuegos	20	6,01	120,24
5	Diseñador de Videojuegos	2	6,01	12,02
6	Diseñador de Videojuegos	1	6,01	6,01
7	Desarrollador de Videojuegos	80	6,75	540,38
8	Desarrollador de Videojuegos	1	6,75	6,75
9	Desarrollador de Videojuegos	1	6,75	6,75
10	Desarrollador de Videojuegos	1	6,75	6,75
11	Desarrollador de Videojuegos	60	6,75	405,28
12	Programador de Páginas Web	170	4,36	741,57
13	Desarrollador de Videojuegos	1	6,75	6,75
14	Desarrollador de Videojuegos	30	6,75	202,64
Total		394		2.144,83

Figura 39: Presupuesto del proyecto ejecutado.

En la Figura 39 podemos ver que la cantidad de horas usadas para llevar a cabo el proyecto fue de 394 y que el costo del mismo fue de $2.144,83 dólares.

4.5.3 Comparativa entre lo estimado y lo ejecutado

Comparación Horas estimadas vs ejecutadas			
Número de Tarea	**Rol Principal**	**Horas estimadas [Hs]**	**Horas ejecutadas [Hs]**
1	Analista Funcional	5	5
2	Analista Funcional	10	20
3	Analista Funcional	1	2
4	Diseñador de Videojuegos	4	20
5	Diseñador de Videojuegos	2	2
6	Diseñador de Videojuegos	1	1
7	Desarrollador de Videojuegos	120	80
8	Desarrollador de Videojuegos	1	1
9	Desarrollador de Videojuegos	1	1
10	Desarrollador de Videojuegos	1	1
11	Desarrollador de Videojuegos	150	60
12	Programador de Páginas Web	30	170
13	Desarrollador de Videojuegos	1	1
14	Desarrollador de Videojuegos	30	30
Total		357	394

Figura 40: Comparación entre las horas estimadas y las ejecutadas del proyecto.

En la Figura 40 podemos observar que las horas ejecutadas totales fueron de aproximadamente un 10% más de tiempo que lo estimado.

Comparación Horas estimadas vs ejecutadas			
Número de Tarea	**Rol Principal**	**Costo Tarea estimado [$]**	**Costo Tarea ejecutado [$]**
1	Analista Funcional	16,60	16,60
2	Analista Funcional	33,21	66,41
3	Analista Funcional	3,32	6,64
4	Diseñador de Videojuegos	24,05	120,24
5	Diseñador de Videojuegos	12,02	12,02
6	Diseñador de Videojuegos	6,01	6,01
7	Desarrollador de Videojuegos	810,56	540,38
8	Desarrollador de Videojuegos	6,75	6,75
9	Desarrollador de Videojuegos	6,75	6,75
10	Desarrollador de Videojuegos	6,75	6,75
11	Desarrollador de Videojuegos	1.013,21	405,28
12	Programador de Páginas Web	130,87	741,57
13	Desarrollador de Videojuegos	6,75	6,75
14	Desarrollador de Videojuegos	202,64	202,64
Total		2.279,51	2.144,83

Figura 41: Comparación entre las costos estimados y ejecutados del proyecto.

Como podemos observar en la Figura 41, el costo total del proyecto ejecutado es aproximadamente un 6% menor que lo esperado. Si bien el proyecto tomó más horas de lo estimado, el costo menor del proyecto se debe sobre todo a que no se dedicaron tantas horas en la tarea de desarrollador de videojuegos, que es la que mayor costo hora tiene de la tabla, y la mayoría de este tiempo se utilizó en la programación de la página web, cuyo costo hora es el menor de todos los roles.

4.5.4 Desvíos

El cambio que mayor impacto tuvo en el proyecto fue el de agregarle una mayor funcionalidad a la página web, que inicialmente estaba pensada solamente para mostrar analíticas. Esto permitió en gran medida paralelizar el desarrollo de la interfaz, ya que, si bien se iba a necesitar el juego completamente desarrollado para las analíticas, el sistema de gestión de alumnos y docentes de la página web se fue haciendo de a poco en los tiempos muertos.

Como consecuencia de la variación previamente mencionada, se utilizó una cantidad de tiempo mayor en entrevistas para la definición de los requerimientos

específicos del proyecto y además para el diseño del videojuego, ya que debía adaptarse a la página web. También provocó que se le haya dedicado una mayor cantidad de tiempo a la programación de la página web y, para evitar sobrepasar excesivamente la cantidad de horas del proyecto, una menor cantidad de tiempo al desarrollo del videojuego.

Por ultimo podemos agregar que, si bien el autor utilizó en gran medida los lenguajes utilizados para la programación del proyecto, muchas horas fueron dedicadas para que el mismo se capacite, que en la estimación inicial no fueron tenidas en cuenta.

Capítulo 5: Conclusión

El objetivo general del proyecto es el de desarrollar y poner a disposición de los docentes una herramienta que contribuya en el proceso de aprendizaje de lectura y escritura de los alumnos que cursan la Unidad Pedagógica, y el cumplimiento del mismo puede medirse a través del análisis de varios objetivos específicos.

Mediante la especificación de los requerimientos a partir de entrevistas con los docentes, se determinaron cuestiones claves para el éxito del proyecto. Las más críticas son, que el videojuego debió ser desarrollado para dispositivos móviles con sistema operativo Android, que el mismo permita a los estudiantes trabajar con palabras relevantes a su entorno, principalmente su nombre y el de sus compañeros, y que, si bien debía contar con conexión a internet para traer la información, también debía permitir a los alumnos jugar de forma offline.

Con respecto al diseño de un videojuego que les resulte entretenido a los alumnos y que a su vez les permita fortalecer y comprobar los conocimientos adquiridos en clases se elaboró el esquema de un videojuego en el cual el jugador arroja un objeto que puede colisionar con una letra o una palabra hasta completar con el objetivo que un docente haya definido, lo que permite que los usuarios que interactúan con el videojuego reforzar sus conocimientos aprendidos en la Unidad Pedagógica y al mismo tiempo entretenerse.

En cuanto al desarrollo del videojuego, primero se definió que la tecnología en la cual fue codificado sería Godot, y luego se elaboró un prototipo que fue siendo refinado a lo largo del avance del proyecto y que concluyó en el videojuego del arquero, que cuenta con hasta cinco modos de juego y que los docentes pueden personalizar en función del conocimiento de sus alumnos.

Por último, para el desarrollo de una interfaz que el docente utiliza para obtener analíticas a partir del progreso de los alumnos, inicialmente se determinó que la mejor tecnología para esto era Vue y luego se llevó a cabo su codificación que concluyó en una página web desde la cual se pueden visualizar las analíticas de los alumnos y que además permite a los docentes la creación de ejercicios personalizados, con distintos modos de

juego y dificultades, para las distintas aulas a las que están asignados de una forma sencilla y rápida.

Si bien todos los objetivos específicos fueron satisfechos y podemos decir que el sistema funciona correctamente como un prototipo, el alcance del proyecto es solamente el de una escuela ya que el mismo no permite agregar instituciones. Por lo que, si otro colegio desease utilizar el software, el mismo debería instalar otra instancia del mismo, el cual tendría sus propios administradores, aulas, docentes y alumnos. Es por este motivo que podemos decir que el sistema posee problemas para escalar, ya que, si se quisiese utilizar el sistema en miles de instituciones, se debería tener un administrador para cada una de ellas.

Además, cabe mencionar que la etapa máxima hasta la que se pudo llegar fue hasta la entrega del proyecto a la referente funcional, y que, si bien fue testeado y validado por Alina, el mismo nunca llegó a ser utilizado en una escuela, por lo que, si bien para el autor de este trabajo el proyecto está terminado, es probable que, si a futuro se quisiese utilizar el software en un entorno real, surjan cambios o se encuentren errores en el sistema.

Para finalizar, el autor añade que la realización de este proyecto ayudó a desarrollar y reforzar múltiples competencias que son esperables en un ingeniero informático. Entre ellas podemos destacar:

- Trabajar con un cliente que presenta una problemática real.
- Estimar la duración de un proyecto de software.
- Elicitar, diseñar, desarrollar y testear el software en sí.
- Utilizar múltiples tecnologías y herramientas para lograr alcanzar el objetivo.
- Afrontar problemas no esperados al inicio del proyecto y ser capaz de resolverlos.

Anexo 1 – Encuesta a docentes y preguntas realizadas a Alina

1 Preguntas y respuestas de la encuesta docente

- **¿Realizan actividades relacionadas con juegos en el aula para enseñar a los alumnos? No necesariamente deben ser videojuegos. (Por ejemplo, trivias, juegos de mesa, Kahoot, etc.)**

Todas las respuestas fueron afirmativas, pero nadie ahondó sobre qué actividades realizan en el aula.

- **¿Saben si los alumnos jugaron alguna vez a algún videojuego? En caso de respuesta afirmativa, ¿A cuál jugaron?**

Entre los juegos más mencionados en las respuestas podemos encontrar el Minecraft, el Free Fire y el Among Us. Además, un docente añadió que en la escuela han utilizado juegos que vienen incluidos en las computadoras enviadas por la provincia.

- **¿Los alumnos tienen computadoras en sus casas? ¿Dirían que saben utilizarlas?**

La mayoría contestó que los alumnos no tienen computadoras en sus casas.

- **¿Los alumnos tienen celulares? ¿Dirían que saben utilizarlos?**

El consenso en esta pregunta fue que, si bien muchos alumnos no tienen celulares, saben utilizarlos debido a que sus padres les tienden a prestar el de ellos.

- **En caso de que los alumnos lleven celulares a la escuela ¿Permiten que los usen en clase?**

La mayoría de los alumnos no llevan celulares a la escuela.

- **En caso de que el alumno no tenga celular, ¿creen que los familiares estarían dispuestos a instalar un videojuego en su celular y a prestárselo al alumno?**

La mayoría contestó que sí.

- **Dado que el alumno puede no estar alfabetizado, ¿creen que un familiar estaría dispuesto a ayudarlo a iniciar el juego?**

  Todas las respuestas fueron afirmativas.

- **¿Qué aspecto del proceso de alfabetización le gustaría enseñar a través de un videojuego? Desarrolle.**

Entre las respuestas podemos encontrar la alfabetización, el nombre propio, la lectoescritura, la silábica y la escritura.

- **¿Cómo se imaginan el videojuego? Desarrolle.**

Uno de los docentes contestó que el videojuego podría ser como el Mario, en el que el avatar se mete en túneles y en cada uno de ellos hay una actividad, y además dio un ejemplo que consistía en que aparezca su nombre, algunas letras del alfabeto y que el alumno deba ir atrapando las letras que le sirven para completar su nombre.

Otro maestro respondió que se lo imaginaban con distintos niveles de complejidad a medida que el juego avanza, acompañado con música y otros sonidos que alienten a seguir con el juego.

Finalmente, un tercer docente contestó que el videojuego podría consistir en ir apretando en palabras relacionadas con un cuento que hayan leído.

El resto de las respuestas no fueron relevantes a la pregunta.

- **¿Cómo creen que debería darse el progreso en el videojuego? Por ejemplo, por niveles, por puntos, etc.**

  La gran mayoría contesto que el progreso debería darse por niveles.

- **¿Considerarían entregar premios o recompensas escolares a los alumnos que interactúen con el videojuego?**

En general los docentes respondieron que preferirían no hacerlo ya que quizás no todos los alumnos podrían jugar al juego.

2 Preguntas y respuestas de las entrevistas con Alina

Cabe destacar en este anexo que las siguientes preguntas se fueron dando a lo largo de múltiples entrevistas con Alina y no solamente en una reunión.

- **¿Para qué plataformas deberá estar destinado el videojuego? (PC, Celular, Web)**

"Como la mayoría de los alumnos no tienen computadoras en sus casas lo mejor sería que el juego sea para celulares, y si bien muchos tampoco tienen su propio celular, la idea es que los padres les presten el de ellos para jugar al videojuego. También sería importante que utilice pocos recursos así todos pueden instalarlo.

Además, el videojuego debe permitir ser jugado de forma offline ya que no todos los alumnos tienen internet en sus casas o móvil de datos."

- **¿Cómo te imaginas el videojuego?**

"Me imagino un videojuego 2D en el que tengas que ir apretando o coleccionando letras para completar tu nombre o una palabra específica relacionada al entorno del estudiante. La idea de la Unidad Pedagógica es que el alumno ya no aprende palabras como "mama" o "papa", si no que primero aprende su nombre, y luego aprende el de algún compañero que sea parecido relacionándolo. Por ejemplo, si tu nombre es "Mariano", luego de aprenderlo, se te podría enseñar un nombre como "Mariela" ya que ambos empiezan con "Ma".

Esto también se da cuando leemos un libro. La idea es que luego de leer un libro, que los niños aprendan las palabras presentes en el mismo. Por ejemplo, luego de leer el cuento de caperucita roja, se intenta que los estudiantes aprendan palabras presentes en el libro, como "rojo", "lobo", "abuela", etc.

También me gustaría agregar que el juego se pueda pausar o reiniciar el nivel en caso de ser necesario.

Y, por último, todas las letras deben ser legibles y estar en imprenta mayúscula, que es la tipografía con la que más se trabaja en el aula."

- **¿Cómo debe estar dado el progreso?**

"Creo que lo mejor sería que esté dado mediante niveles y que haya varios modos de juego ya que en un nivel estaría bueno que trabaje con su nombre y en otro nivel aprenda sobre las palabras que empiezan con "Ma" o las palabras de algún cuento.

Al iniciar un nivel, el juego deberá indicar al alumno lo que debe hacer, ya sea atrapar las letras de determinada palabra o directamente atrapar las palabras relacionadas a un tópico en especial.

Estaría bueno también que el alumno cuando se equivoque pierda puntos o una vida para evitar que alguien pueda completar el nivel de suerte.

Me gustaría que los docentes puedan elegir distintas dificultades a los ejercicios y que, por ejemplo, que en la dificultad más baja aparezcan menos cantidad de letras del diccionario

Además, estaría bueno que haya un temporizador y que el tiempo del mismo dependa de la dificultad que le asigne un docente al ejercicio.

Al finalizar un nivel se le debe mostrar al alumno el puntaje que realizó."

- **¿Qué modos de juego te gustaría que tenga?**

"Me gustaría tener un modo de juego en el que el alumno deba ir agarrando letra por letra hasta completar su nombre y que ese sea siempre el primer nivel para todas las aulas. Otro modo de juego sería uno en el que se debe agarrar siempre su nombre y que alrededor estén los nombres de sus compañeros o palabras que un docente pueda haber agregado.

Además, estaría bueno extender estos modos a palabras que el docente pueda elegir, por ejemplo, recolectar todas las letras de la palabra "Mariposa" o atrapá todas las palabras que empiezan con "Ma".

Por último, se podría agregar un modo en el que haya más de una palabra correcta, como, por ejemplo, atrapar todos los meses que tienen 31 días, y que el alumno le tenga que disparar a las palabras "Enero", "Marzo", etc."

- **¿Cómo debe estar dado el ingreso al juego? ¿Cuáles son los ejercicios a los que pueden acceder los alumnos?**

"La idea es que los docentes les den un código a los alumnos que usarán para poder ingresar al juego por primera vez, y que estudiantes ingresen este código, cuando abren el juego por primera vez, junto con su nombre y apellido. Como se debe poder trabajar con los nombres de los compañeros, es importante que un docente pueda determinar quiénes de los que ingresaron son alumnos y quienes no. También es importante que los docentes puedan modificar el código cuando el mismo lo requiera.

Con respecto a los ejercicios, yo pienso que lo mejor sería que cada aula tenga su conjunto de ejercicios, es decir, que un alumno de 1º no pueda acceder a los niveles del aula de 2º por ejemplo."

- **¿Cómo crees debe ser la página web?**

"Primero y principal debe ser gratuita ya que no creo que la escuela pueda pagarla por más barata que sea. Debe permitir a los docentes crear, editar y eliminar ejercicios del aula en la que están asignados. Además, debe permitir ver las analíticas de los alumnos que quieran.

Con respecto al sistema de usuarios estaría bueno que cada maestro tenga su propio usuario y que un administrador determine si son docentes o no, las aulas en las que enseña y además poder editar sus nombres o apellidos. Para ello se debería tener un formulario de registro de usuario y de inicio de sesión."

- **¿Qué información te gustaría obtener de los alumnos que juegan el videojuego?**

"Un gráfico de tortas que indique la cantidad de veces que se ganó el nivel en función de las veces que se jugó. Es importante saber fácilmente si el alumno puede hacer los ejercicios o si a estos hay que bajarles la dificultad.

También me gustaría tener un gráfico que muestre el puntaje a medida que va reintentando el nivel ya que lo lógico de alguien que aprende sería ir logrando mejores puntajes mientras siga intentando.

Además, estaría bueno tener gráficos que muestren cuantas sopapas acertaron con la letra o la palabra correcta o la incorrecta para saber si los alumnos están completando los niveles porque golpean todas las letras o palabras que aparecen o debido a que saben diferenciar los objetos del nivel."

Anexo 2 – Proceso de Selección de Tecnologías para el Desarrollo del Proyecto

Como existen varias herramientas en las que se puede desarrollar una página web, un videojuego y un servidor, se armaron tablas comparativas entre las tecnologías más relevantes para seleccionar las más convenientes para llevar a cabo el proyecto.

Para cada tecnología se seleccionaron las características más relevantes y a cada una de ellas se le asignó un valor del uno (1) al cinco (5) en función de que tan bien está implementada dicha funcionalidad. A su vez, a cada característica se la ponderó en función a la utilidad relativa del proyecto. Por último, se sumó la puntuación de cada característica para cada tecnología y se seleccionaron las que tenían mayor valor.

Los candidatos para el desarrollo de la página web fueron Vue (https://vuejs.org/), Angular (https://angular.io/) y React (https://reactjs.org/) debido a su participación en el mercado y el armado de las tablas se realizó utilizando la información comparativa de Pattakos [27], Abraham [28] y Daityari [29].

1 Tabla Comparativa: Página Web

Característica	Vue	Angular	React	Ponderación
Performance	5	4	5	1%
Curva de aprendizaje	4	3	5	5%
Documentacion	5	5	5	3%
Foros	5	3	5	3%
Conocimiento	3	1	1	60%
Tiempo de proyecto	5	4	5	10%
Librerias	5	5	5	10%
Seguridad	5	5	5	1%
Mantenibilidad	3	5	4	1%
Flexibilidad	5	4	5	6%
Total	3,73	2,27	2,59	100%

Figura 42: Tabla comparativa utilizada para determinar el lenguaje en el cual se desarrolló la página web del sistema.

Para el desarrollo del videojuego, los motores de que se consideraron para la tabla comparativa fueron Unity (https://unity.com/), Unreal (https://www.unrealengine.com/) y Godot (https://godotengine.org/) ya que son los motores más populares y que además

permiten exportar a Android y el desarrollo de videojuegos 2D. La tabla se armó en función a la información la comparación que definió Wilson [30] y Harpooner [31].

2 Tabla Comparativa: Videojuego

Característica	Unity	Unreal	Godot	Ponderación
Performance	4	5	4	1%
Curva de aprendizaje	5	3	5	1%
Documentación	5	5	5	1%
Foros	5	4	3	1%
Conocimiento	3	1	4	60%
Tiempo de proyecto	4	3	5	5%
IDE	5	5	5	1%
Exportar a Android	5	5	5	10%
Añadir sonido	5	5	5	1%
Creación de videojuegos 2D	4	3	5	3%
Utilización en pantalla táctil	5	5	5	3%
Creación de animaciones	5	5	5	1%
Creación de HUD	5	5	5	1%
Facilidad del lenguaje	5	3	5	1%
Soporte	5	5	4	1%
Facilidad de instalación	4	4	5	1%
Físicas y movimiento de objetos	5	5	5	3%
Es gratuito	5	5	5	5%
Total	3,66	2,33	4,32	100%

Figura 43: Tabla comparativa utilizada para determinar el lenguaje en el cual se desarrolló el videojuego del sistema.

Por último, para el servidor, las herramientas que fueron candidatas para el desarrollo fueron PHP (https://www.php.net/) con Laravel (https://laravel.com/), Java (https://www.java.com/es/) con Spring Boot (https://spring.io/) y Node.js (https://nodejs.org/en/) junto con Express (https://expressjs.com/), debido al conocimiento previo que se consideró tener con cada sistema. Para el completado de la tabla se tomó la información que definió Asadi [32].

3 Tabla Comparativa: Servidores

Característica	PHP & Laravel	Java & Spring Boot	Node.js & Express	Ponderación
Performance	3	4	5	1%
Curva de aprendizaje	5	3	4	5%
Documentación	5	4	5	3%
Foros	5	5	5	3%
Conocimiento	2	5	2	60%
Tiempo de proyecto	4	3	5	10%
Librerias	4	5	4	5%
Seguridad	4	5	4	1%
Mantenibilidad	5	5	3	1%
Flexibilidad	4	4	5	1%
Base de datos	5	5	5	5%
Debug	4	5	4	5%
Total	2,96	4,65	3,02	100%

Figura 44: Tabla comparativa utilizada para determinar el lenguaje en el cual se desarrolló el servidor del sistema.

De las tablas se puede apreciar que la tecnología con más puntuación para el desarrollo de la página web es Vue (Figura 42), la de mayor puntuación para el desarrollo del videojuego es Godot (Figura 43), y la de mayor puntuación para el desarrollo del servidor es Java & Spring Boot (Figura 44).

Anexo 3 – Proyecto PTF

1 Clases de Dominio

Dentro del paquete “com.ptf.entities” se pueden encontrar todas las entidades del proyecto, como se puede ver en la Figura 45.

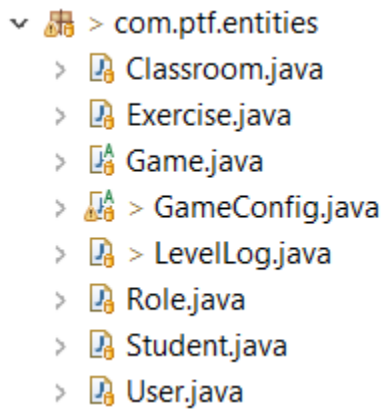


Figura 45: Paquete “com.ptf.entities” con las entidades del proyecto.

En Spring Boot, una entidad no es más que un objeto de Java que representa una tabla almacenada en una base de datos. Cada instancia de una entidad equivale a una fila en la tabla. Para que una clase sea una entidad, se le debe agregar la anotación @Entity.

Para emplear este tipo de anotaciones se debe utilizar un framework (que ya viene incluido en Spring Boot) conocido como Hibernate, cuya característica principal es el mapeo de clases Java a tablas de bases de datos y el mapeo de tipos de datos Java a tipos de datos SQL. Hibernate también proporciona funciones de consulta y recuperación de datos, genera llamadas SQL y libera al desarrollador del manejo manual y la conversión de objetos del conjunto de resultados, entre otras cosas.

Dentro de este paquete, existe la clase “User”, que es utilizada para almacenar los usuarios que pueden acceder a la página web y se codificó como se muestra en la Figura 46.

```java
@Entity
@Table(name="user", schema = "public")
public class User {
    @Id
    @Column(name="id")
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    private Long id;

    @Column(name="username", unique = true, nullable=false, length=100)
    private String username;

    @JsonProperty(access = Access.WRITE_ONLY)
    @Column(name="password", nullable=false, length=100)
    private String password;

    @ManyToOne(optional = false, cascade = CascadeType.MERGE)
    private Role role;

    @Column(name="first_name", nullable=false, length=100)
    private String firstName;

    @Column(name="last_name", nullable=false, length=100)
    private String lastName;

    @ManyToMany(cascade = CascadeType.MERGE)
    @JsonIgnoreProperties("teachers")
    @JoinTable(name = "user_classroom",
        joinColumns = @JoinColumn(name = "user_id", referencedColumnName = "id"),
        inverseJoinColumns = @JoinColumn(name = "classroom_id", referencedColumnName = "id"))
    private List<Classroom> classrooms;
```

Figura 46: Clase "User" utilizada para representar a los usuarios en el sistema.

Las variables “firstName” y “lastName” sirven para identificar al usuario que quiere ingresar al sitio, y “username” y “password”, son las credenciales de acceso a la página web. Cabe destacar que los constructores, los getters y los setters, si bien no están en la imagen, fueron codificados ya que son requeridos por Hibernate para el mapeo de clases a tablas.

Anotaciones Hibernate utilizadas en la clase “User”:

- @Table es utilizada para definir el nombre de la tabla y el esquema al que pertenece.
- @Id es utilizada para definir la clave primaria de una tabla.
- @Column es utilizado para definir condiciones de una columna de una tabla, como el nombre, si es única, si admite valores nulos, etc.
- @GeneratedValue es utilizada para configurar la forma en la que se incrementa la clave primaria. En este caso y para el resto de las entidades se utiliza una estrategia de clave primaria de tipo auto incremental.

- @JsonProperty.access(): especifica el acceso a una variable mediante la serialización y la deserialización. En nuestro caso, el acceso se codificó en solo lectura para que en caso de que el servidor responda un JSON armado a partir de una clase de User, no se muestre la propiedad password.
- @ManyToOne y @ManyToMany: nos permiten mapear una entidad con otra.
- @JoinTable es utilizada en las relaciones muchos a muchos para especificar el nombre de la tabla de la relación, sus columnas, etc.
- @JsonIgnoreProperties es utilizado para evitar que exista recursividad en las respuestas del servidor.

Por lo que la tabla “user” generada por esta entidad queda como está ilustrada en la Figura 47.

Name	Type	Length	Decimals	Allow Null
id	bigint	20	0	☐
first_name	varchar	100	0	☐
last_name	varchar	100	0	☐
password	varchar	100	0	☐
username	varchar	100	0	☐
role_id	bigint	20	0	☐

Figura 47: Tabla "user" de la base de datos utilizada para almacenar los usuarios.

Y la tabla “user_classroom”, que es utilizada para asignar profesores a las aulas y viceversa se puede ver en la Figura 48.

Name	Type	Length	Decimals	Allow Null
user_id	bigint	20	0	☐
classroom_id	bigint	20	0	☐

Figura 48: Tabla "user_classroom" de la base de datos utilizada para relacionar a los usuarios con un rol.

La clase “Role”, que tiene relación con la clase “User”, es utilizada para asignar un rol a los usuarios que utilizan la página, que puede ser igual a “Guest” (invitado), “Teacher” (docente) o “Admin” (administrador). La explicación de las funciones de cada rol se encuentra en la sección 4.2

La codificación de la clase “Role” se puede ver en la Figura 49.

```java
@Entity
@Table(name="role", schema = "public")
public class Role {
    @Id
    @Column(name="id")
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    private Long id;

    @Column(name="name", nullable=false, length=100)
    private String name;
```

Figura 49: Clase "Role" utilizada para representar a los roles que existen en el sistema.

Y esto generó la tabla “role”, mostrada en la Figura 50, y las filas en la Figura 51.

Name	Type	Length	Decimals	Allow Null	
id	bigint	20	0	☐	1
name	varchar	100	0	☐	

Figura 50: Tabla "role" de la base de datos utilizada para almacenar los roles del sistema.

id	name
1	Admin
2	Teacher
3	Guest

Figura 51: Contenido de la tabla “Role” de la base de datos.

La codificación de la clase “Classroom” está expresada en la Figura 52.

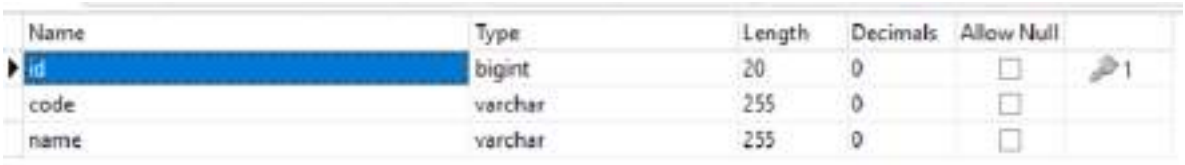
```java
@Entity
@Table(name="classroom", schema = "public")
public class Classroom {
    @Id
    @Column(name="id")
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    private Long id;

    @Column(name="name", unique = true, nullable=false)
    private String name;

    @Size(min=4, max=4)
    @Column(name="code", unique = true, nullable=false)
    private String code;

    @ManyToMany(cascade = CascadeType.MERGE, mappedBy="classrooms")
    @JsonIgnoreProperties("classrooms")
    private List<User> teachers;

    @OneToMany(cascade = CascadeType.MERGE, mappedBy="classroom")
    @JsonIgnoreProperties("classroom")
    private List<Student> students;

    @OneToMany(cascade = CascadeType.MERGE, mappedBy="classroom")
    @JsonIgnoreProperties("classroom")
    private List<Exercise> exercises;
```

Figura 52: Clase "Classroom" utilizada para representar a las aulas en el sistema.

Esta clase es utilizada para representar las aulas del colegio. La variable “name” es utilizada para distinguir el aula y la variable “code” es un identificador único que los estudiantes deberán ingresar en el videojuego la primera vez que acceden para que puedan ser asociados con un aula. Dicho código solo puede tener una longitud igual a cuatro (4) caracteres alfanuméricos (gracias a la anotación @Size) y puede ser modificado en todo momento por un administrador o un docente.

Además de la relación muchos a muchos con la tabla de usuarios, la clase “Classroom”, posee una lista de estudiantes y de ejercicios/niveles asociados al aula. Esto está dado mediante las relaciones con las clases “Student” y la clase “Exercise”, las cuales son de uno a muchos ya que un aula puede tener asignada muchos ejercicios y alumnos, pero un estudiante o un nivel solo pueden estar asociados a un aula.

Finalmente, esta entidad genera la tabla “classroom” que se puede ver en la Figura 53, Y dentro de ella se encuentra la siguiente información que se muestra en la Figura 54 (recordar que solo la columna “code” es modificable).



Name	Type	Length	Decimals	Allow Null	
id	bigint	20	0	☐	1
code	varchar	255	0	☐	
name	varchar	255	0	☐	

Figura 53: Tabla "classroom" de la base de datos utilizada para almacenar las aulas.

id	code	name
1	1A1A	1A
2	1B1B	1B
3	1C1C	1C
4	1D1D	1D
5	2A2A	2A
6	2B2B	2B
7	2C2C	2C
8	2D2D	2D

Figura 54: Contenido de la tabla “Classroom” de la base de datos.

La entidad “Student” es usada para almacenar a los usuarios que se registran en el sistema mediante el videojuego y está mostrada en la Figura 55.

```java
@Entity
@Table(name="student", schema = "public")
public class Student {
    @Id
    @Column(name="id")
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    private Long id;

    @Column(name="first_name", nullable=false, length=100)
    private String firstName;

    @Column(name="last_name", nullable=false, length=100)
    private String lastName;

    private Boolean isStudent;

    @ManyToOne
    @JoinColumn(name="classroom_id")
    @JsonIgnoreProperties("students")
    private Classroom classroom;
```

Figura 55: Clase "Student" utilizada para representar a los estudiantes en el sistema.

Las variables “firstName” y “lastName” son utilizadas para identificar al usuario. La variable “isStudent” es empleada para permitir que el usuario pueda acceder al juego.

Finalmente, la tabla “student” quedó como se muestra en la Figura 56:

Name	Type	Length	Decimals	Allow Null	
id	bigint	20	0	☐	1
first_name	varchar	100	0	☐	
is_student	bit	1	0	☐	
last_name	varchar	100	0	☐	
classroom_id	bigint	20	0	☐	

Figura 56: Tabla "student" de la base de datos utilizada para almacenar los estudiantes.

La entidad “LevelLog” es utilizada para registrar eventos que ocurren cuando un estudiante termina de jugar un nivel y que luego son procesados en la página para mostrar las analíticas. Se codificó como se muestra en la Figura 57:

```java
@Entity
@Table(name="levelLog", schema = "public")
public class LevelLog {
    @Id
    @Column(name="id")
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    private Long id;

    @ManyToOne(cascade = CascadeType.MERGE)
    @JsonIgnoreProperties("classroom")
    private Exercise exercise;

    @ManyToOne(cascade = CascadeType.MERGE)
    @JsonIgnoreProperties("classroom")
    private Student student;

    private int score;

    private int correctPlungers;

    private int incorrectPlungers;
```

Figura 57: Clase "LevelLog" utilizada para representar los eventos del videojuego en el sistema.

Las variables “correctPlungers” e “incorrectPlungers” son utilizadas para armar el gráfico de barras para la analítica sopapas correctas e incorrectas en función del intento. La variable “score” es utilizada para la construcción del otro gráfico de barras para la analítica puntuación final en función del intento.

Cada evento registra el puntaje final del estudiante, las sopapas que disparó y acertaron una letra correcta, y las que disparó y colisionaron con una letra incorrecta.

Además, cada uno de estos eventos está asociado a un jugador y a un ejercicio, y es por este motivo que las relaciones son de uno a muchos, ya que un estudiante o un nivel pueden tener asignados muchos eventos, pero un evento solo está asociado con un jugador y un ejercicio.

Finalmente, en la Figura 58 se puede ver la tabla generada por la entidad “LevelLog”.

Name	Type	Length	Decimals	Allow Null	
id	bigint	20	0	☐	1
correct_plungers	int	11	0	☐	
incorrect_plungers	int	11	0	☐	
score	int	11	0	☐	
exercise_id	bigint	20	0	☐	
student_id	bigint	20	0	☐	

Figura 58: Tabla "levelLog" de la base de datos utilizada para almacenar los eventos.

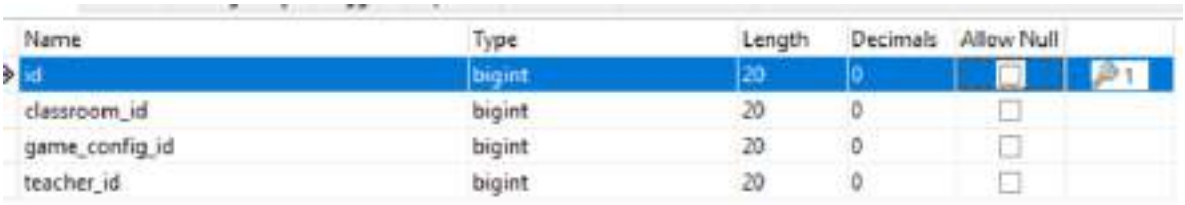
La entidad “Exercise” es utilizada para representar los niveles que los estudiantes tendrán y es mostrada en la Figura 59.

```
@Entity
@Table(name="exercise", schema = "public")
public class Exercise {
    @Id
    @Column(name="id")
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    private Long id;

    @ManyToOne
    @JoinColumn(name="classroom_id")
    @JsonIgnoreProperties("exercises")
    private Classroom classroom;

    @OneToOne(cascade = CascadeType.MERGE)
    @JsonIgnoreProperties("classrooms")
    private User teacher;

    @OneToOne(cascade = CascadeType.MERGE)
    private GameConfig gameConfig;

    @OneToMany(cascade = CascadeType.MERGE, mappedBy="exercise")
    @OrderBy("id ASC")
    @JsonIgnoreProperties("exercise")
    private List<LevelLog> levelLogs;
```

Figura 59: Clase "Exercise" utilizada para representar a los ejercicios/niveles en el sistema.

La anotación “@OrderBy” es utilizada para definir el orden en el que serán devueltos los eventos del nivel cuando se haga una petición que devuelva una entidad de tipo “Exercise”.

Cada ejercicio estará asociado con un aula, un docente y una configuración de juego. Además, los ejercicios poseen una lista de eventos. Finalmente, la tabla “exercise” generada se puede ver en la Figura 60.



Name	Type	Length	Decimals	Allow Null
id	bigint	20	0	☐
classroom_id	bigint	20	0	☐
game_config_id	bigint	20	0	☐
teacher_id	bigint	20	0	☐

Figura 60: Tabla "exercise" de la base de datos utilizada para almacenar los ejercicios/niveles.

La entidad “GameConfig” es una clase abstracta utilizada para asociar una configuración de juego a un ejercicio y su codificación se puede ver en la Figura 61.

```
@Entity
@Table(name="gameConfig", schema = "public")
@JsonTypeInfo( use = JsonTypeInfo.Id.NAME, include = JsonTypeInfo.As.PROPERTY, property = "type")
@JsonSubTypes({
  @Type(value = NameLetters.class, name = "Letras del Nombre"),
  @Type(value = WholeName.class, name = "Nombre Completo"),
  @Type(value = WordLetters.class, name = "Letras de Palabra"),
  @Type(value = WholeWord.class, name = "Palabra Completa"),
  @Type(value = MultipleWords.class, name = "Multiples Palabras")})
@DiscriminatorColumn(name = "gameModeClass")
public abstract class GameConfig {
    @Id
    @Column(name="id")
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    private Long id;

    @OneToOne(cascade = CascadeType.MERGE)
    private Game game;

    @Column(name="gameMode", length=100)
    private String gameMode;

    @Column(name="description")
    private String description;

    @Column(name="difficulty")
    private String difficulty;
```

Figura 61: Clase "GameConfig" utilizada para representar las configuraciones de los niveles en el sistema.

Cada configuración de juego tendrá una variable “difficulty”, que es utilizada para asignar una determinada dificultad al nivel, una variable “description”, que es empleada para describir lo que hay que hacer en el nivel, y una variable “gameMode”, que es usada para definir el modo de juego, y una variable de tipo “Game”, que sirve para indicar que juego utiliza la configuración.

En Spring Boot no todas las entidades crean tablas en la base de datos. Cuando una clase hereda de otra, existen distintas instrucciones que permiten definir como se almacenará la información. La estrategia por defecto en Spring Boot y que se utilizó en la clase de “GameConfig” debido a que es la más fácil y rápida de implementar es una conocida como “Single Table”, es decir, tabla única.

En esta metodología, las entidades heredadas de “GameConfig” no crean sus propias tablas y por lo tanto almacenan su información en su clase padre. Además, las variables adicionales de las clases hijas son agregadas en la tabla “game_config” como columnas.

Las anotaciones @JsonTypeInfo, @JsonSubTypes son utilizadas para definir las posibles clases que heredan de la clase “GameConfig”, es decir, los distintos modos de juego. La anotación @DiscriminatorColumn es utilizada para agregar una columna llamada discriminador que le permite a Java identificar a que clase hija pertenece.

Finalmente, la tabla “game_config” creada por esta entidad se muestra en la Figura 62.

Name	Type	Length	Decimals	Allow Null
game_mode_class	varchar	31	0	☐
id	bigint	20	0	☐ 1
description	varchar	255	0	☐
difficulty	varchar	255	0	☐
game_mode	varchar	100	0	☐
counter	int	11	0	☐
use_classmates	bit	1	0	☐
word	varchar	255	0	☐
game_id	bigint	20	0	☐

Figura 62: Tabla " game_config " de la base de datos utilizada para almacenar las configuraciones de los niveles.

Las columnas “use_classmates”, “counter” y “word” son variables de las clases que heredan de “GameConfig”.

Por último, la entidad abstracta “Game” solo posee una variable “name” que es utilizada para identificar los distintos videojuegos que el proyecto soporta, y está representada en la Figura 63.

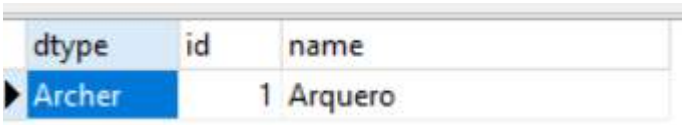
```java
@Entity
@Table(name="game", schema = "public")
public abstract class Game {
    @Id
    @Column(name="id")
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    private Long id;

    @Column(name="name", unique = true, nullable=false, length=100)
    private String name;
```

Figura 63: Clase "Game" utilizada para representar a los juegos en el sistema.

Esta clase fue incluida para permitir la escalabilidad del sistema y actualmente solo posee una entrada “Arquero”, que es la única actividad que se puede jugar. Al igual que la clase abstracta “GameConfig”, contiene una columna que le permite a Java identificar a que clase hija pertenece llamada “dtype”. La tabla “game” se puede ver en la Figura 64 y su contenido en la Figura 65.

Name	Type	Length	Decimals	Allow Null
dtype	varchar	31	0	☐
id	bigint	20	0	☐ 1
name	varchar	100	0	☐

Figura 64: Tabla "game" de la base de datos utilizada para almacenar los juegos.



dtype	id	name
Archer	1	Arquero

Figura 65: Contenido de la tabla “Game” de la base de datos.

Dentro del paquete “com.ptf.entities.games” solo se encuentra la clase “Archer” que representa el único videojuego desarrollado, y esto se puede ver en la Figura 66.

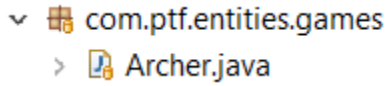


Figura 66: Paquete “com.ptf.entities.games” con los juegos del proyecto.

Esta clase extiende de “Game” y solo posee un constructor, como se muestra en la Figura 67.

```java
@Entity
public class Archer extends Game {

    public Archer() {
        super("Arquero");
    }
}
```

Figura 67: Clase "Archer" utilizada para representar al juego del arquero en el sistema.

El paquete “com.ptf.entities.games.configs.archerconfigs” contiene las clases que heredan de “GameConfig”, como se indica en la Figura 68.

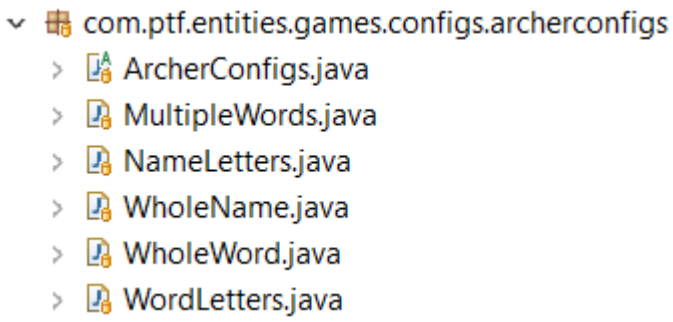


Figura 68: Paquete “com.ptf.entities.games.configs.archerconfigs” con las configuraciones de juego del juego del arquero.

La clase abstracta “ArcherConfigs”, que se puede ver en la Figura 69, extiende de “GameConfig”, y el resto de las clases del paquete heredan “ArcherConfigs” y son utilizadas para representar los distintos modos de juego.

```java
@Entity
public abstract class ArcherConfigs extends GameConfig {

    public ArcherConfigs() {
        super(null, null, null, null);
    }

    public ArcherConfigs(Game game, String gameMode, String description, String difficulty) {
        super(game, gameMode, description, difficulty);
    }

    public ArcherConfigs(Long id, Game game, String gameMode, String description, String difficulty) {
        super(id, game, gameMode, description, difficulty);
    }
}
```

Figura 69: Clase "ArcherConfigs" utilizada para representar a las configuraciones del videojuego del arquero en el sistema.

La clase “NameLetters” es usada cuando el modo de juego es el de “Letras del Nombre” y está mostrada en la Figura 70. Esta entidad extiende de “ArcherConfigs” y no posee variables adicionales.

```
@Entity
@JsonTypeName("Letras del Nombre")
public class NameLetters extends ArcherConfigs {

    public NameLetters() {
        super();
    }
```

Figura 70: Clase "NameLetters" utilizada para representar el modo de juego “Letras del Nombre” en el sistema.

La anotación @JsonTypeName es utilizada para nombrar a la clase de Java.

La clase “WholeName” es usada para el modo de juego “Nombre Completo” y es mostrada en la Figura 71.

```
@Entity
@JsonTypeName("Nombre Completo")
public class WholeName extends ArcherConfigs {
    @Column(name="counter")
    private int counter;

    @Column(name="useClassmates")
    private Boolean useClassmates;

    @Column(name="incorrectWords")
    @ElementCollection
    private List<String> incorrectWords;
```

Figura 71: Clase "NameLetters" utilizada para representar el modo de juego “Nombre Completo” en el sistema.

La variable “counter” se emplea para definir cuantas veces se deberá atrapar la palabra deseada. “useClassmates” es utilizada para determinar si en el modo de juego aparecen los nombres de los estudiantes que pertenecen al aula en la que existe este ejercicio. “incorrectWords” es usada para que los docentes puedan agregar palabras incorrectas al ejercicio. La anotación @ElementCollection utilizada en esta variable crea la

tabla “whole_name_incorrect_words” en la base de datos para almacenar las palabras incorrectas. Esta tabla se puede ver en la Figura 72.

Name	Type	Length	Decimals	Allow Null
multiple_words_id	bigint	20	0	☐
correct_words	varchar	255	0	☐

Figura 72: Tabla " whole_name_incorrect_words" de la base de datos utilizada para almacenar las palabras incorrectas en el modo de juego “Nombre Completo”.

La clase “WordLetters” es usada cuando el modo de juego es el de “Letras de Palabra” y la misma está mostrada en la Figura 73.

```
@Entity
@JsonTypeName("Letras de Palabra")
public class WordLetters extends ArcherConfigs{

    @Column(name="word")
    private String word;
```

Figura 73: Clase "WordLetters" utilizada para representar el modo de juego “Letras de Palabra” en el sistema.

La variable “word” es usada para almacenar la palabra que utiliza el nivel.

La clase “WholeWord” es utilizada para el modo de juego “Palabra Completa” y se puede ver en la Figura 74.

```java
@Entity
@JsonTypeName("Palabra Completa")
public class WholeWord extends ArcherConfigs {
    @Column(name="counter")
    private int counter;

    @Column(name="useClassmates")
    private Boolean useClassmates;

    @Column(name="word")
    private String word;

    @Column(name="incorrectWords")
    @ElementCollection
    private List<String> incorrectWords;
```

Figura 74: Clase "WordLetters" utilizada para representar el modo de juego “Letras de Palabra” en el sistema.

Y la tabla “whole_word_incorrect_words” utilizada para almacenar la lista “incorrectWords” y se puede ver en la Figura 75.

Name	Type	Length	Decimals	Allow Null
whole_word_id	bigint	20	0	☐
incorrect_words	varchar	255	0	☐

Figura 75: Tabla " whole_word_incorrect_words" de la base de datos utilizada para almacenar las palabras incorrectas en el modo de juego “Palabra Completa”.

Por último, el modo de juego “Multiples Palabras” está representado en la entidad “MultipleWords”, y su codificación se muestra en la Figura 76.

```java
@Entity
@JsonTypeName("Multiples Palabras")
public class MultipleWords extends ArcherConfigs {
    @Column(name="counter")
    private int counter;

    @Column(name="useClassmates")
    private Boolean useClassmates;

    @Column(name="correctWords")
    @ElementCollection
    private List<String> correctWords;

    @Column(name="incorrectWords")
    @ElementCollection
    private List<String> incorrectWords;
```

Figura 76: Clase "MultipleWords" utilizada para representar el modo de juego "Multiples Palabras" en el sistema.

Las variables de esta clase generan dos tablas, "multiple_words_correct_words", la cual es utilizada para almacenar las palabras correctas y se muestra en la Figura 77 y la tabla "multiple_words_incorrect_words", la cual es utilizada para almacenar las palabras incorrectas, se puede ver en la Figura 78.

Name	Type	Length	Decimals	Allow Null
multiple_words_id	bigint	20	0	☐
correct_words	varchar	255	0	☐

Figura 77: Tabla " multiple_words_correct_words" de la base de datos utilizada para almacenar las palabras correctas en el modo de juego "Multiples Palabras".

Name	Type	Length	Decimals	Allow Null
multiple_words_id	bigint	20	0	☐
incorrect_words	varchar	255	0	☐

Figura 78: Tabla " multiple_words_incorrect_words" de la base de datos utilizada para almacenar las palabras incorrectas en el modo de juego "Multiples Palabras".

Finalmente, en la Figura 79 se puede observar el diagrama del modelo final de la base de datos generada por Hibernate.

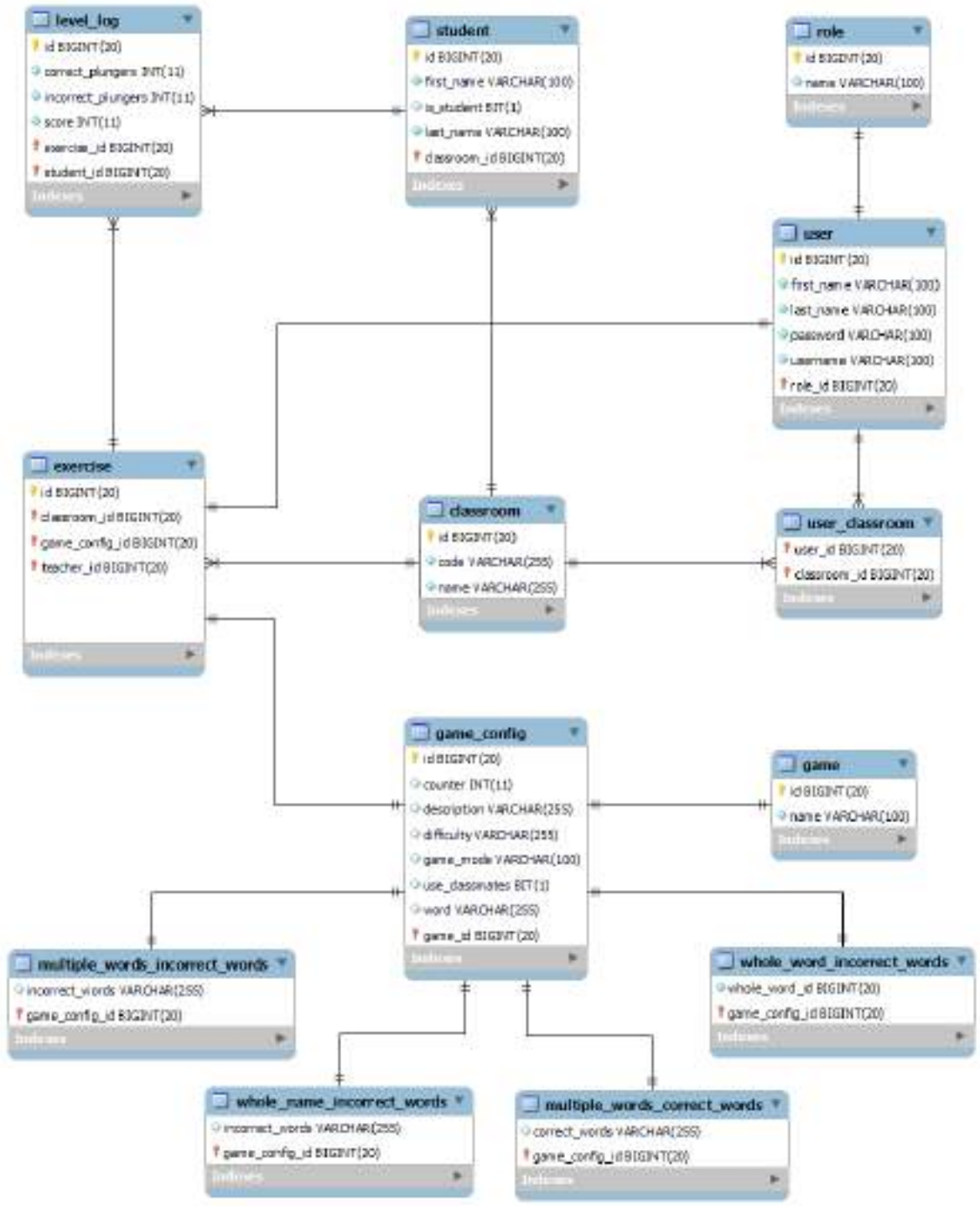


Figura 79: Diagrama de la base de datos del sistema.

2 Construcción de la base de datos inicial

Cada vez que se inicia el servidor hay tablas que se autocompletan con la información necesaria para que la aplicación funcione correctamente. Por ejemplo, la tabla

“role” contiene filas que deben existir en todo momento y no son modificables. Estas filas se pueden ver en la Figura 51.

Para esto, se creó la clase “DatabaseBuilder” que se encarga de inicializar las tablas con sus respectivos datos. Para ello, implementa la interfaz “CommandLineRunner” que contiene el método “run()”. Al iniciar el servidor, Spring Boot llama automáticamente a dicho método.

Luego, para inicializar la tabla “role” con los roles se agregó el código que se puede ver en la Figura 80.

```
60
61      @Autowired
62      RoleService roleService;
63
64      @Override
65      public void run(String... args) throws Exception {
66          Role admin = new Role(Long.valueOf(1), "Admin");
67          roleService.createRole(admin);
68
69          Role teacher = new Role(Long.valueOf(2), "Teacher");
70          roleService.createRole(teacher);
71
72          Role guest = new Role(Long.valueOf(3), "Guest");
73          roleService.createRole(guest);
```

Figura 80: Código utilizado para agregar los roles en la tabla “role”.

Una vez instanciado el rol de administrador en la variable “admin”, se ejecuta el código de la línea 67 de la Figura 80 para agregarlo en la base de datos. Para ello se llama al método createRole() del servicio “roleService”, pero esta variable no es instanciable ya que “RoleService” es una interfaz, como se puede observar en la Figura 81.

```
@Service
public interface RoleService {
    ResponseEntity<Role> createRole(Role role);
}
```

Figura 81: Interfaz “RoleService”.

Por lo que para usar el método “createRole()”, se utiliza la anotación “@Autowired” para la variable “roleService”, la cual mediante la inyección de dependencias brinda una instancia de “RoleService” que nos permite utilizar el método, pero para ello se debe definir el método en una clase, y esto es realizado en “RoleImpl”, como se muestra en la Figura 82.

```
@Component
public class RoleImpl implements RoleService {

    @Autowired
    private RoleDAO roleDAO;

    @Override
    public ResponseEntity<Role> createRole(Role role) {
        Role newRole = roleDAO.save(role);
        return ResponseEntity.ok(newRole);
    }
}
```

Figura 82: Clase “RoleImpl” utilizada para definir el comportamiento de los métodos de la interfaz “RoleService”.

Esta clase también utiliza la inyección de dependencias y utiliza una instancia de la interfaz “RoleDAO”, la cual extiende de la interfaz “JpaRepository”, como se puede observar en la Figura 83.

```
public interface RoleDAO extends JpaRepository<Role, Long>{
    Optional<Role> findByName(@Param("name") String name);
}
```

Figura 83: Interfaz “RoleDAO” en la que se definen métodos que deberán ser implementados por “RoleImpl”.

Esta última interfaz es la que brinda múltiples métodos que permiten hacer altas, bajas y modificaciones en la base de datos. Para este caso, se utiliza la función “save()”, que es utilizada para almacenar una función en la base de datos.

Si bien la interfaz “JpaRepository” posee varios métodos para trabajar con la base de datos, estos son muy básicos. Es por este motivo que las consultas más complejas fueron escritas a mano. Por ejemplo, en el caso de la interfaz “ClassroomDAO”, la función

findClassroomsByUserId() que es usada para recuperar todas las aulas a las que pertenece un usuario, fue codificada como se muestra en la Figura 84.

```
public interface ClassroomDAO extends JpaRepository<Classroom, Long>{
    @Query("SELECT c FROM Classroom c WHERE c.code = ?1")
    Optional<Classroom> findByCode(String code);

    @Query("SELECT c FROM Classroom c WHERE c.name = ?1")
    Optional<Classroom> findByName(String name);

    @Query(value = "SELECT c.id, c.code, c.name "
            + "FROM classroom c "
            + "INNER JOIN user_classroom uc ON c.id = uc.classroom_id "
            + "WHERE uc.user_id = ?1 ORDER BY c.id", nativeQuery = true)
    List<Classroom> findClassroomsByUserId(Long userId);
}
```

Figura 84: Interfaz "ClassroomDAO" en la que se definen métodos que deberán ser implementados por "ClassroomImpl".

3 Servicios REST

Dentro del paquete "com.ptf.rest" se pueden encontrar las clases que proporcionan los servicios REST. La clase "UserREST" es utilizada para brindar los servicios asociados con la tabla "user" y la entidad "User", como se ve en la Figura 85.

```
@RestController
@RequestMapping("/user")
public class UserREST {
    private static final Logger logger = LoggerFactory.getLogger(UserREST.class);

    @Autowired
    private UserService userService;

    @RequestMapping
    public ResponseEntity<List<User>> getUsers(){
        logger.debug("Fetching all users.");
        return userService.getUsers();
    }

    @RequestMapping(value="id/{userId}")
    public ResponseEntity<User> getUserById(@PathVariable("userId") Long userId){
        logger.debug("Fetch user with id: " + userId);
        return userService.getUserById(userId);
    }
```

Figura 85: Clase "UserREST" utilizada para brindar servicios relacionados con los usuarios del servidor.

La anotación “@RestController” es utilizada para indicar que la clase funciona como un servicio REST. “@RequestMapping” se utiliza para indicar la ruta de los servicios. Para el caso de “UserREST”, todos sus servicios se encuentran dentro de la ruta “/user”.

En caso de que se quiera solicitar la lista de todos los usuarios, hay que dirigirse a “www.direcciónDeLaPágina/user”. En caso de que se quiera solicitar solo un usuario en función de su id, habrá que dirigirse a “www.direcciónDeLaPágina/user/x”, siendo “x” un valor entero mayor a 0.

Estos servicios también utilizan la inyección de dependencias para poder cumplir con su funcionamiento y, además, emplean una instancia de la clase “Logger” que permite ir mostrando por pantalla las operaciones que va realizando el servidor.

4 Seguridad en el Servidor

Como indican Morgan et al. [33], una de las características del protocolo REST es que es un protocolo sin estados, cada petición HTTP contiene toda la información necesaria para ejecutarla, lo que permite que el servidor no recuerde ningún estado previo, y por ende no sepa si el cliente que solicitó un servicio ha sido autenticado o no.

Para evitar que cualquier usuario pueda utilizar los servicios que utiliza un docente o un administrador, se implementó un sistema de seguridad en el servidor utilizando JWT, que funciona de la siguiente forma:

Cuando desde la página un usuario inicia sesión, se envía una petición HTTP al servidor de “login”. El servidor procesa la información y si los datos son válidos, se crea un token de acceso que luego es enviado al cliente, y que el mismo luego deberá usar para acceder a las rutas protegidas del servidor. Este token es único para cada usuario.

La próxima vez que el cliente haga una petición a una ruta protegida, el servidor validará el token, y en caso de que el mismo no haya sido modificado, comprobará que el rol del usuario permite acceder a la dirección, realizará la solicitud y luego responderá con la información que el cliente solicitó. Este comportamiento se puede ver en la Figura 86.

Figura 86: Flujo entre un cliente y un servidor que implementa el protocolo REST. Adaptado de Castillo Rodríguez [34].

Por este motivo, toda petición al servidor que realice un usuario deberá tener en su cabecera el token de acceso que recibió luego de autenticarse.

Como en el sistema hay algunos servicios que solo los usuarios con rol de administrador o de docente pueden acceder, se utilizó un filtro en Spring Boot que permite asignar niveles de seguridad a dichos servicios. Para ello, previamente se construyó una tabla, ilustrada en la Figura 87, que permite ver para cada servicio, el rol requerido (columnas “Role”), y, además, si lo solicita la página o el videojuego (columnas “System”).

Service		System		Role			
		FrontEnd	Game	Admin	Teacher	Guest	All
/analytic	createLevelLog		X				X
/classroom	getClassrooms	X			X		
	updateClassroomCode	X		X	X		
	getClassroomsOfTeacherByUsername	X			X		
/exercise	getExerciseById	X			X		
	createExercise	X			X		
	deleteExerciseById	X			X		
	updateExercise	X			X		
	getExercisesByClassroomId		X		X		X
	getExercisesByStudentId	X			X		
/game	getGames	X			X		
/student	getStudentById	X	X				X
	createStudent		X				X
	deleteStudentById	X			X		
	updateStudent	X			X		
/user	getUsers	X		X			
	getUserById	X		X			
	deleteUserById	X		X			
	updateUser	X		X			
	getUserByUsername	X					X
	createUser	X					X
/login	login	X					X
/util	accessToken	X					X

Figura 87: Tabla utilizada para definir el nivel de seguridad de los distintos servicios REST.

En la Figura 88 se puede ver parte del código que permite agregar filtros a los distintos servicios.

```
http.authorizeRequests().antMatchers(HttpMethod.GET, "/student/id/**").permitAll();
http.authorizeRequests().antMatchers("/student/createStudent/**").permitAll();
http.authorizeRequests().antMatchers("/user/username/**").permitAll();
http.authorizeRequests().antMatchers(HttpMethod.POST, "/user/**").permitAll();
http.authorizeRequests().antMatchers("/login/**").permitAll();
http.authorizeRequests().antMatchers("/util/**").permitAll();

http.authorizeRequests().antMatchers("/classroom/**").hasAnyAuthority("Teacher", "Admin");
http.authorizeRequests().antMatchers("/exercise/**").hasAnyAuthority("Teacher", "Admin");
http.authorizeRequests().antMatchers("/game/**").hasAnyAuthority("Teacher", "Admin");
```

Figura 88: Ejemplos de filtros agregados para permitir o denegar el acceso.

Bibliografía

1. Claudia, A. (3 de abril de 2016). UP – Unidad Pedagógica. *Educación Primaria*. https://blogedprimaria.blogspot.com/2016/04/up.html
2. Merriam-Webster, diccionario. (15 de abril de 2022). "Video game". https://www.merriam-webster.com/dictionary/video%20game
3. Zimmerman, E. (7 de julio de 2004). Narrative, Interactivity, Play and Games: Four Naughty Concepts in Need of Discipline.
4. Hunicke, R., LeBlanc, M., Zubek, R. (2004). MDA: A Formal Approach to Game Design and Game Research.
5. Scolari, C.A. (ed., 2013). Homo Videoludens 2.0. De Pacman a la gamificación. Col·lecció Transmedia XXI. Laboratori de Mitjans Interactius. Barcelona: Universitat de Barcelona.
6. Hiebaum, J. (11 de agosto de 2014). Componentes básicos de un videojuego. *Manual del Game Designer.* https://manualdelgamedesigner.blogspot.com/2014/08/componentes-basicos-de-un-videojuego.html
7. Personaje (30 de mayo de 2022). En *Wikijuegos*. https://videojuegos.fandom.com/es/wiki/Personaje
8. Significados, diccionario (30 de mayo de 2022). "Efecto". https://www.significados.com/efecto/
9. Järvinen, A. (2008). Games without Frontiers: Theories and Methods for Game Studies and Design. Tampere University Press.
10. Plass, J., Frye, J., Homer, B., Perlin, K. (2011). Learning Mechanics and Assessment Mechanics for Games for Learning.
11. Zimmerman, E., Salen, K. (2003). Rules of Play: Game Design Fundamentals. Cambridge, MA: MIT Press.
12. Michael, D., Chen, S. (2006). Serious Games: Games That Educate, Train and Inform.

13. Marcano, B. (2008). Juegos serios y entrenamiento en la sociedad digital. Revista Electrónica Teoría de la Educación: Educación y Cultura en la Sociedad de la Información. Vol. 9, nº 3. Universidad de Salamanca.
14. Spinelli, A., Massa S., Revisión Sistematica: Elicitación de Requerimientos Educativos en Serious Games, Actas del XXV Congreso Argentina de Ciencias de la Computación (CACIC 2019), Rio Cuarto, Cordoba, Argentina del 14 al 18 de Octubre del 2019. Universidad de Rio Cuarto.
15. Arnab, S., Lim, T., Carvalho, M., Bellotti, F., Freitas, S., Louchart, S., Suttie, N., Berta, R., De Gloria, A. (2014). Mapping learning and game mechanics for serious games analysis: Mapping learning and game mechanics. British Journal of Educational Technology.
16. Fournier, H., Kop, R., Hanan, S. (2011). The Value of Learning Analytics to Networked Learning on a Personal Learning Environment.
17. Pérez, D. (s.f.). Analíticas del Aprendizaje en la Educación Virtual. https://ude.edu.uy/analiticas-del-aprendizaje-en-la-educacion-virtual/
18. Baalsrud Hauge, J., Berta, R., Fiucci, G., Fernández-Manjón, B., Padron-Napoles, C. Westra, W., Nadolski, R. (septiembre de 2014). Implications of Learning Analytics for Serious Game Design.
19. Freire, M., Serrano-Laguna, A., Manero, B., Martínez-Ortiz, I., Moreno- Ger, P., Fernández-Manjón, B. (2016). Game Learning Analytics: Learning Analytics for Serious Games.
20. Massa, S. M., Evans, F., Spinelli, A., Zapirain, E., Rico, C., Morcela, A., Kühn, F. Modelos y herramientas para el proceso de desarrollo de Serious Games.
21. Alonso-Fernandez, C., Calvo-Morata, A., Freire, M., Martinez-Ortiz, I., Fernandez-Manjon, B. (abril de 2017). Systematizing game learning analytics for serious games.
22. Kühn, F., Massa, S. (2018). Analíticas de Aprendizaje en Serious Games: una revisión sistemática de la literatura, en actas del 4 congreso bienal de la IEEE ARGENCON 2018, San Miguel de Tucumán, Tucumán, Argentina, 6,7,8

de junio del 2018. Universidad Nacional de Tucumán, Universidad Tecnológica Nacional y la IEEE Argentina.

23. ComparaSoftware (13 de mayo de 2021). Qué es el modelo incremental. *comparasoftware*. https://blog.comparasoftware.com/que-es-el-modelo-incremental/
24. Ortiz, M. (septiembre de 2012). Modelo Incremental. *isw-udistrital*. http://isw-udistrital.blogspot.com/2012/09/ingenieria-de-software-i.html
25. Intelisoft (2016). Metodo Incremental. *intelisoft*. https://sites.google.com/site/intelisoft2016/metodo-incremental
26. Transferencia de Estado Representacional. (5 de enero de 2022). En *Wikipedia*. https://es.wikipedia.org/w/index.php?title=Transferencia_de_Estado_Representacional&oldid=140748018
27. Pattakos, Aris (4 de enero de 2022). Angular vs React vs Vue 2022. *aThemes*. https://athemes.com/guides/angular-vs-react-vs-vue/
28. Abraham, A. (29 de octubre de 2019). Angular vs React vs Vue: ¿Cuál es la mejor opción? *SomosWigou*. https://medium.com/somoswigou/angular-vs-react-vs-vue-cu%C3%A1l-es-la-mejor-opci%C3%B3n-941a207951c7
29. Daityari, S. (27 de diciembre de 2021). Angular vs React vs Vue: Which Framework to Choose. *codeinwp*. https://www.codeinwp.com/blog/angular-vs-vue-vs-react/
30. Wilson, L. (1ro de noviembre de 2019). Godot, Unity, Unreal Engine, CryEngine? Which Game Engine Should I Choose? *@thelukaswils*. https://medium.com/@thelukaswils/godot-unity-unreal-engine-cryengine-which-game-engine-should-i-choose-553f8ff7999f
31. Harpooner, B. (9 de septiembre de 2018). Making Cyberglads 1: choosing a game engine. *cyberglads*. https://cyberglads.com/making-cyberglads-1-choosing-a-game-engine.html

32. Asadi, S. (18 de enero de 2019). Express.js vs Java Spring vs PHP Laravel. *@lvlr.xaus*. https://medium.com/@lvlr.xaus/express-js-vs-java-spring-vs-php-laravel-fe74a68828b3
33. Morgan, C., Hightower, K., Dekena, G. [Udacity] (6 de junio de 2016). *How does JWT work* [Video]. Youtube. https://www.youtube.com/watch?v=K6pwjJ5h0Gg
34. Castillo Rodríguez, S. (8 de enero de 2019). Autenticación de APIs basada en tokens con Spring y JWT. *Softtek*. https://blog.softtek.com/es/autenticando-apis-con-spring-y-jwt