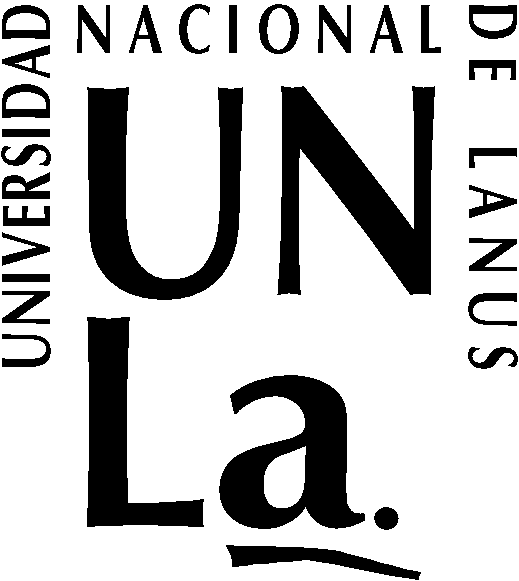
**Gamificando Algoritmos**

****

Martin Rios  
Ignacio Oliveto  
Luciano Otegui  
Tomas Pereyra

**Cátedra Proyecto de Software**

**Licenciatura en Sistemas**

**2018**

**Link Repositorio: https://github.com/Tomaspereyra/Gamificando-Algoritmos.g**

Gamificando Algoritmos

# *1.Documento de requisitos (entrevista con el cliente):*

En el siguiente documento se reafirmarán los requisitos propuestos y llevados a cabo por el cliente en la entrevista sobre el software a desarrollar. Comenzaremos con una descripción del producto para poder entender mejor el mismo y en que consiste, luego seguirán una serie de preguntas que se le hicieron al cliente donde se irán explicando la respuesta del mismo para dejar asentado que lo que se entendió sea lo correcto. Al final de este documento se pedirá la firma del cliente en donde aceptará todos los requisitos entendidos y en caso de querer una modificación o un requisito nuevo, el costo del software se verá aumentado.

*Entrevista:*

*Descripción del producto:*

*Es una plataforma para docentes y alumnos en la que se va a enseñar programación mediante la denominada "Programación en Bloques". La misma se desarrollaría a través de juegos (A definir), de los cuales los docentes tienen que poder crear las instancias que deseen, con la configuración que deseen. Es decir: Si nosotros hacemos un sistema de laberintos para resolver con Bloques, entonces el docente tiene que poder crear sus propios laberintos para darlos como clases.*

Rta: En base a la descripción del producto el cliente está de acuerdo con lo entendido.

*Preguntas respondidas por el cliente:*

*1. ¿Qué cantidad de juegos distintos es requerida? ¿Cuál es la libertad que tenemos para diseñarlos? (Desde total libertad hasta modelos bien definidos).*

*Rta: La cantidad de juegos puede ser de dos o tres, dependiendo los tiempos o dificultad que se de en el camino. Los juegos deben ser óptimos en el caso del laberinto pueden irse configurando o diseñando de menor a mayor dificultad en donde se vayan acciones mínimas como caminar por ejemplo y a medida que voy avanzando le voy agregando dificultad, o como segunda opción pueden venir niveles pre definidos. Los juegos distintos deben ser simples, pueden ser laberinto, plataformero, etc. La libertad de diseño es totalmente libre, debe ser algo simple y no se debe explayar tanto. Los juegos deben ser para ambos géneros.*

*2. ¿Es necesario que los juegos y sus instancias se registren en planes de estudio o clases?*

*Rta: El docente puede armar cursos con cierta cantidad de niveles, con descripciones, agruparlo por gustos, temas, etc. El docente puede elegir bloques predefinidos o que bloques permite.*

*3. ¿Qué más se necesita del software con respecto a la administración de alumnos por parte de los docentes? ¿Presentismo? ¿Clases? ¿Exámenes? ¿Cursos?*

*Rta: En este caso es mas orientado a solo los juegos, lo ideal sería que haya persistencia del usuario, dándole un progreso, que se puede saltear un nivel un curso, un tema, que sea más libre. Indicar si la solución esta bien o mal, mostrar una solución propuesta, no darle tanta importancia al tiempo de respuesta. Implementar un puntaje(adicional).*

*4. ¿Qué nivel de asistencia necesita el usuario en la resolución de los problemas? ¿Si el juego se pusiera muy complicado, deberíamos ofrecer ayuda o algo así?*

*Rta: Si, otorgarle una ayuda si el profesor lo desea o una pista. Si tarda tanto tiempo una sugerencia a través de un cartel. Otorgarle un hit al estudiante por tiempo de inactividad dependiendo del docente si quiere o no. Los estudiantes no necesitan estar dados de alta, cualquiera puede jugar no hay necesidad de estar en un curso o logueado.*

*5. ¿A qué público está orientado este desarrollo? ¿Alumnos de que edades? ¿Y con que conocimiento previo de programación? (Nulo, básico, intermedio, avanzado, otros)*

*Rta: El publico es para adultos jóvenes adolescentes, a partir de los 18 años aproximadamente, no hace falta conocimientos previos.*

*6. ¿Qué variedad de bloques es requerida? ¿Hay algún conjunto obligatorio?*

*Rta: Incluir los tradicionales, comandos simples, variables, procedimientos y funciones, iteración y condicional, alguna estructura como opcional. El profesor debe elegir que mostrar de todo el paquete por escenario y se puede limitar la cantidad de bloques en la que el usuario debe resolver el escenario.*

*7. ¿Es necesario que se traduzcan las soluciones a código fuente? En tal caso ¿A qué lenguajes debe poder exportarse la solución del problema? Tenemos para ofrecer: Python, Lúa, Dart, JavaScript y PHP. ¿Es necesario alguno más?*

*Rta: Si, estaría bueno persistir las soluciones, otro lenguaje podría ser XML.*

*Para cerrar dicho documento, constatamos que lo entendido anteriormente sobre los requisitos del producto, es lo charlado y acordado con el cliente. Con el fin de evitar futuros malentendidos o exigir cambios críticos a mitad de proyecto, perjudicando su desarrollo.*

# *2.Documento requisitos de interfaz:*

En el siguiente documento se reafirmarán los requisitos de interfaz en base a la primera iteración de la maqueta, en donde el cliente propuso modificaciones en cuanto a lo visto y algunos aspectos claves del mismo.

En la interfaz de inicio, se estableció que la opción de iniciar sesión (login) se encuentre en la pantalla principal y que no sea un redireccionamiento a otra página, en el mismo se propuso agregar un remember-me.   
En la parte derecha de arriba, se señaló un botón de “Acerca de” con algún tipo de información del software-equipo. Luego, bajo el botón de iniciar sesión, agregar un botón jugar y otro de información.  
En la interfaz del estudiante, se propuso quitar los cuadros y quitar la barra de desplazamiento, establecer algo más sencillo, también planteo agregar una barra de progreso por curso.  
También se deberán añadir opciones de continuar jugando, de volver a empezar y un menú despegable para mostrar opciones del usuario, si es estudiante( explorar, juegos, perfil y salir), si es docente( administrar cuenta, administrar cursos y salir).

***Requisitos funciones nuevos:***  
Se establecieron algunos requisitos funcionales más como:

-No hacer un seguimiento del estudiante.  
-Usar cookies para persistir usuarios no registrados.  
-Guardar y mostrar estadísticas de cuanta gente resuelve bien determinado curso y cuanta gente no.

# 3.Requisitos funcionales:

RF 1 El sistema debe ofrecer una serie de juegos interactivos cuyas soluciones se alcanzarán a través de una combinación de bloques algorítmicos por parte de los estudiantes.

RF 1.1 Debe haber 2 variedades de los mencionados juegos, pudiendo haber más   
 de 2 de forma opcional.

RF 1.2 Dichos juegos deben exigir el uso de los siguientes conceptos   
 algorítmicos: Estructuras de selección (Simples y Múltiples), Estructuras

de iteración, Variables, Funciones y Procedimientos

RF 1.3 Dichos juegos deben poder exportar sus soluciones a algún lenguaje de   
 programación, con el objetivo de que el estudiante asocie los bloques con   
 código fuente

RF 1.4 Los juegos deben ser inclusivos para ambos géneros.

RF 1.5 Los juegos no deben requerir conocimientos previos en programación ni algoritmia

RF 2 El sistema debe permitir generar nuevas instancias de cada uno de estos   
 juegos para que el docente las configure como desee, a partir de ahora   
 llamados **Escenarios**.   
 Las propiedades configurables de cada uno deben ser:

* Calidad y Cantidad de Bloques para la resolución
* Cantidad máxima absoluta de bloques a usar
* Sugerencia (Hint) (Opcional)
* Solución Propuesta por el docente (Opcional)

RF 2.1 Cada Escenario completado debe registrar e informar:

* Tiempo de resolución
* Solución Brindada
* Traducción de solución a un lenguaje de programación
* Puntaje (En base al tiempo de resolución)

RF 3 El sistema debe permitir la creación de **Cursos** por parte de los docentes,   
 los cuales serán una serie ordenada de **Escenarios** que definirán el flujo   
 de juego para los estudiantes.

RF 3.1 Cada curso debe tener una cantidad flexible de escenarios.

RF 3.2 El docente tiene que poder establecer si es menester que los escenarios   
 sean jugados en orden (Opcional)

RF 3.3 Cada estudiante debe poder suscribirse a dichos Cursos, lo cual lo   
 habilitará para jugar el mismo, y se persistirá su progreso dentro de su   
 perfil.

RF 3.4 No debe ser necesario asociarse como estudiante para jugar los cursos,   
 pero no se persistirá su progreso ni se realizará ningún seguimiento sobre   
 el mismo. Estos usuarios se llamarán **Visitantes**

RF 3.5 El sistema debe registrar datos estadísticos de cada Escenario del curso,   
 siendo estos:

* Intentos Totales
* Intentos Fallidos

RF 4 El sistema deberá persistir la información de:

* Docentes (Con sus cursos creados)
* Estudiantes (Con su progreso detallado en los cursos suscriptos)
* Juegos (Con sus reglas y bloques)
* Escenarios (Con su configuración y parámetros)
* Cursos (Con sus escenarios ordenados y configuración)

# *4.Requisitos de Interfaz:*

RI 1: La aplicación debe contar con un botón "Home" que llevara al usuario a la Landing Page. El mismo debe ser accesible desde todas las vistas de la aplicación.

RI 2: La aplicación debe contar con una sección en la parte superior derecha   
 dedicada a login del usuario, tanto docente como estudiante.  
 La misma debe verse desde todas las vistas de la aplicación, y debe   
 ofrecer un menú contextual para:

* Mi Cuenta
* Mis cursos (En curso para estudiantes / Creados para docentes)
* Jugar
* Acerca De / Contacto

RI 2.1 Para ingresar al sistema, se solicitarán:

* Nombre de Usuario
* Contraseña

RI 3: Interfaz Gráfica - Menú Cursos en progreso de Estudiante  
  
RI 3.1: Debe contar con una lista de los cursos en los que participa, ordenado en   
 forma vertical en la parte central de la pantalla.   
  
  
RI 3.1.1: En cada curso debe informarse:

* Tipo de Juego
* Nombre
* Progreso del curso (Escenarios completados / Escenarios totales)
* Puntaje promedio

Y a su vez debe haber botones para:

* Continuar desde el ultimo escenario jugado
* Comenzar desde el principio

RF 4: Interfaz Gráfica – Menú Cursos Creados (Docente)

RF 4.1 El menú debe tener una lista de los cursos creados, ordenado en forma   
 vertical en la parte central de la pantalla. Debe tener también un botón   
 para crear un nuevo curso.

RF 4.1.1 Al crear un nuevo curso, se debe solicitar el tipo de juego y el nombre. Luego se re direccionara

RF 4.1.2 Cada curso creado debe informar:

* Fecha de Creación
* Nombre (Editable)
* Tipo de Juego
* Cantidad de Escenarios

A su vez, debe contar con botones para:

* Eliminarlo
* Modificarlo
* Ver Estadísticas

RF 5: Interfaz Gráfica – Menú Editar Cursos.  
  
RF 5.1 Este menú deberá Informar:

* Cantidad de Escenarios
* Creador
* Fecha de Creación

A su vez, debe contar con botones para:

* Eliminar curso
* Editar Escenario (1 por Escenario)
* Eliminar Escenario (1 por Escenario)
* Volver a "Mis Cursos"

RF 6 Interfaz Gráfica – Editar Escenario

RF 6.1 Debe contar con los recursos necesarios para editar el escenario en   
 cuestión. Variara según el tipo de Juego. En el caso del laberinto, deberá   
 tener un editor de mapas del mismo, por ejemplo.

RF 6.2 El menú deberá contar con los siguientes datos comunes modificables:

* Tiempo "par" (Optimo esperado)
* Bloques permitidos
* Cantidad máxima absoluta de bloques
* Sugerencia o "Hint"
* Solución propuesta

RF 7 Interfaz Gráfica – Menú Administrar Cuenta

RF 7.1 Debe brindar campos editables para modificar:

* Nombre
* Apellido
* URL
* Email

RF 7.2 Debe brindar una opción para cambiar contraseña, compuesta de:

* Contraseña Actual
* Nueva Contraseña
* Confirmar Nueva Contraseña

RF 8 Interfaz Gráfica – Menú Jugar Escenario

RF 8.1 El mismo se mostrará en disposición, contando con los siguientes   
 elementos ordenados de izquierda a derecha:

* Interfaz interactiva Blockly
* Botonera con botones para "Ejecutar Algoritmo", "Reiniciar" y "Consejo"
* Interfaz gráfica del videojuego que corresponda al escenario

RF 8.2 Luego de ejecutar el algoritmo construido en la interfaz Blockly, Se debe  
dar un feedback al usuario de lo sucedido (Tanto haya sido correcto como errado) a través de un popup o una notificación de algún tipo

RF 8.3 Debe contar también con algún botón para volver al índice del curso en   
que se encuentra dicho escenario

# *5.Documento entorno de desarrollo:*

En el siguiente documento se establecerá el entorno de desarrollo, llevado a cabo por el equipo de desarrollo sobre el software a desarrollar. El documento simplemente tendrá el formato de una lista en donde se presentarán los distintos tipos de herramientas. Las mismas podrían modificarse o agregar nuevas en caso de ser necesario.

En el entorno de Recurso Humanos (RRHH), se dividirán los dos tipos de áreas entre los cuatro integrantes del equipo en donde dos se encargarán de Front-End y los otros dos de Back-End:

***Back-End:***  
-Ignacio Oliveto (controladores)  
-Tomas Pereyra(persistencia)

***Front-End:***   
-Martin Ríos(aplicaciones)  
-Luciano Otegui(vistas)

En cuanto al Hardware, se usarán cuatro computadoras de escritorio y dos netbooks/notebooks con una arquitectura de 64-bits. En cuanto al Software se utilizarán las siguientes herramientas:

-Sistema operativo Windows 7 y 10  
-Servidores Python, host Python.   
-Base de datos MySQL  
-Python 2.7  
-Phaser 3.12.0  
-MySQL server 8.0  
-MySQL Workbench  
-GitHub desktop (repositorio y versionado)  
-Materialize (Materialización y estilos)  
-HTML5  
-Google’s Blockly Version bc210e

# **6.Informe de Viabilidad:**

Este informe tiene como objetivo justificar la viabilidad del desarrollo proyecto “Gamificando Algoritmos” a llevar a cabo por el equipo 5.

* **Tiempo**: En primera medida se realizó la estimación del tiempo que llevaría desarrollar este proyecto. Para encontrar el mismo se utilizó el modelo COCOMO II y un promedio de tres (3) modelos empíricos. Ambos estiman alrededor de 7 meses de desarrollo. Dado que ambos modelos coinciden, llegamos a la conclusión de que podremos llegar a realizarlo en el plazo estipulado.
* **Tecnología**: Para llevar a cabo este proyecto, sería necesario implementar una serie de tecnologías mínimas, las cuales hemos logrado adoptar en las pruebas de viabilidad realizadas por el equipo. Estas tecnologías serian:

1. *Blockly*: Hemos realizado con éxito una interfaz entre Blockly y una aplicación de prueba, evidenciando nuestra capacidad de manejar dicha tecnología.
2. *Librerías Graficas (Para los juegos)*: Dos (2) de nuestros programadores cuentan con experiencia previa en Phaser (Un pequeño framework gratuito orientado a los videojuegos), por lo que implementar los videojuegos no será demasiado laborioso. Comprobamos lo anteriormente dicho al haber implementado un videojuego de prueba (Laberinto DEMO) para la interfaz con Blockly.

* **Conocimientos Técnicos:** El equipo tiene el Know-how necesario acerca de Análisis, Diseño y desarrollo de Sistemas Informáticos para atacar la problemática planteada por este proyecto, así como también dispone de las herramientas necesarias para desarrollar la solución requerida.

1. *Bases de Datos*: Necesitaremos bases de datos para la persistencia de los datos de usuario. Para realizarla, usaremos MySQL como motor de base de datos el cual es gratuito, y nuestro equipo cuenta con las metodologías y técnicas necesarias para operarlo y diseñar un esquema de datos relacional correspondiente al problema.
2. *Cliente-Servidor*: Al ser una aplicación web, se necesitará una arquitectura Cliente-Servidor que provea un punto de acceso para el cliente y la aplicación, así como centralizar los datos del sistema en un servidor dedicado. Nuestro equipo cuenta con un robusto conocimiento de desarrollo Backend, asegurando la persistencia y consistencia de los datos almacenados, así como también tenemos una fuerte base en Frontend, incluyendo tanto el desarrollo de aplicaciones en JavaScript como la estilización de sitios web.

* **Entorno de Desarrollo, Servidor y Aplicación:** Para poder sostener este proyecto, se necesitará un servidor que pueda atender las peticiones de los usuarios finales, así como una plataforma sobre la cual el usuario pueda manejar la aplicación.  
  Al ser una aplicación Web, la plataforma será el mismo navegador, que pueda encontrarse en cualquier PC o celular, por lo que no hay un costo de desarrollo de la misma. Así mismo, el servidor será provisto por la Universidad de Lanús, por lo que ambos entornos ya están suplidos. Con respecto a los entornos de desarrollo, usamos herramientas puramente gratuitas en nuestras PCs hogareñas, por lo que no habrá costo ni limitación alguna en las mismas.

Habiendo evaluado los puntos anteriores, sumados a las pruebas de tecnología realizadas por el equipo, concluimos que el proyecto 'Gamificando-Algoritmos' a desarrollar por el equipo 5 de la catedra de Proyecto de Software es **VIABLE**.

# **7.Estimacion de Proyecto: Gantt**



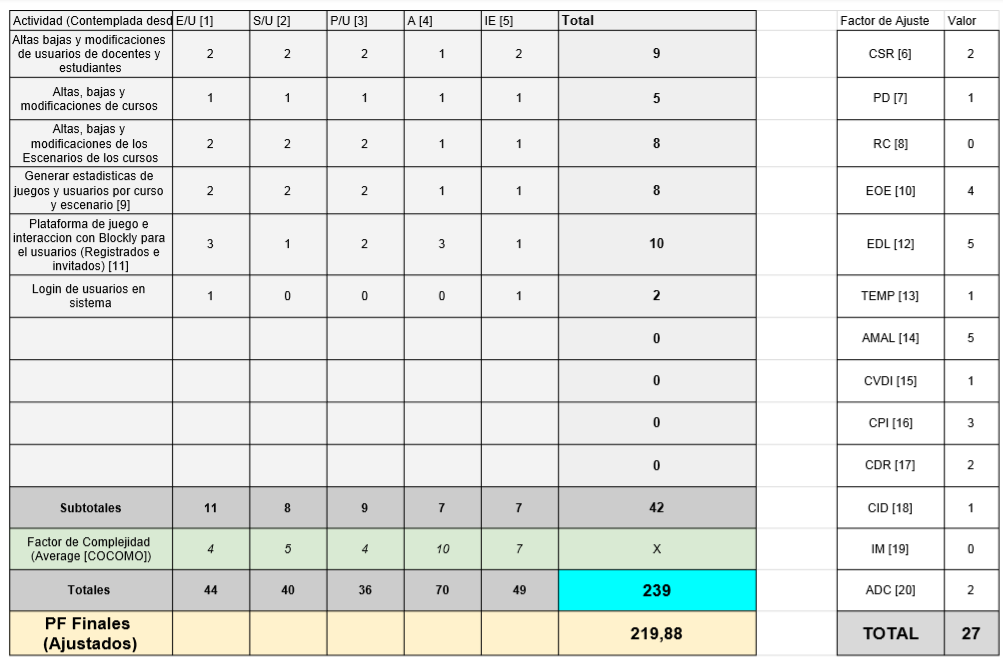
# **8.Estimacion de Proyecto: Pert**

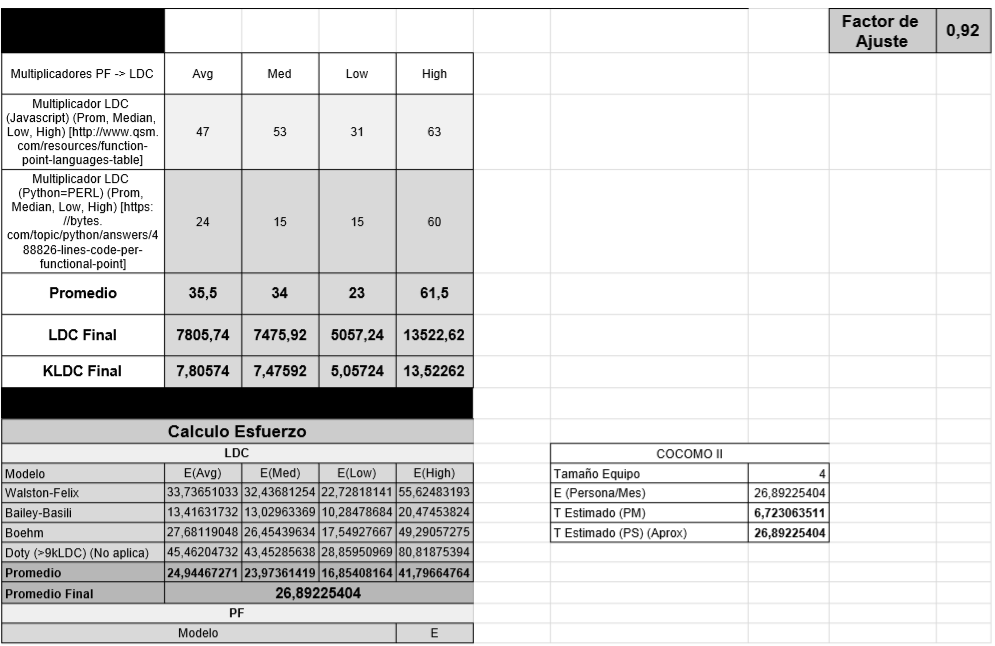


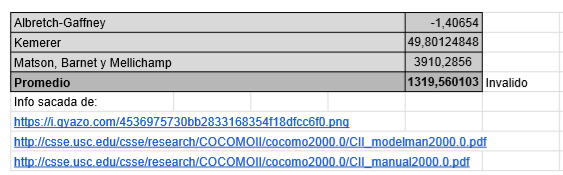




# **9.Estimacion de Proyecto: COCOMO**







# **10.Mapa de actividades:**

