Estructuras de Datos

Proyecto Segunda Evaluación – PAO II 2021
"Tres en Raya" contra el computador

IMPORTANTE

Éste es un proyecto **grupal**. Todo grupo debe estar conformado por **TRES** estudiantes. Ningún grupo puede tener menos miembros y ningún estudiante puede entregar el proyecto de manera individual. Excepciones a estas reglas requieren una **autorización explícita** del profesor y solo serán consideradas por circunstancias de fuerza mayor.

Éste es un proyecto **grupal**. Por tanto, se espera que **todos** los miembros del grupo se involucren en el diseño e implementación de la solución que se solicita. La entrega del proyecto incluirá un reporte de auto y coevaluación, donde cada miembro del grupo calificará sus contribuciones al proyecto y las contribuciones de sus compañeros.

Si algún grupo experimenta el incumplimiento sistemático de alguno de sus miembros, esta situación debe ser reportada al profesor **TAN PRONTO COMO SE PRESENTE**. Cualquier estudiante cuyo incumplimiento sea comprobado, será retirado del grupo y **DEBERÁ IMPLEMENTAR EL PROYECTO DE MANERA INDIVIDUAL Y SOBRE EL 50% DEL PUNTAJE POSIBLE**, sin derecho a aplicar a puntos extra por funcionalidades opcionales implementadas. Si un grupo no reporta estos problemas al profesor **TAN PRONTO COMO SE PRESENTEN**, el grupo no podrá argumentar en la entrega del proyecto el incumplimiento de alguno de sus miembros. De hacerlo así, extemporáneamente, el grupo será penalizado con el 25% de la nota obtenida en su entrega y no tendrá derecho a aplicar a puntos extra por funcionalidades opcionales implementadas.

Considere las políticas de buena conducta académica que se explicaron en la primera clase del curso respecto al plagio y demás violaciones del código de Ética de la ESPOL. Los autores del proyecto deben ser **usted y sus compañeros de grupo**. Cualquier indicio de lo contrario, otorgará al proyecto una nota igual a cero y los estudiantes involucrados serán reportados a las unidades de la ESPOL correspondientes para el tratamiento pertinente.

Finalmente, **siga las instrucciones de este documento** para evitar inconvenientes. Si tiene dudas, consulte al profesor en lugar de asumir cosas que pueden ser incorrectas. Los ayudantes del curso podrían no ser la mejor fuente para despejar dudas en cuanto al proyecto.

Las instrucciones de este documento están diseñadas para que ni usted ni el profesor pierdan tiempo en cuanto a la entrega/revisión del proyecto.

SIGA LAS INSTRUCCIONES AL PIE DE LA LETRA.

EN CASO DE TENER DUDAS, CONSULTE AL PROFESOR OPORTUNAMENTE.

Introducción

"Tres en Raya" es un juego en el que dos jugadores (**O** y **X**), marcan los espacios de un tablero de 3×3 alternadamente. Cada jugador debe colocar su símbolo una vez por turno y no debe ser sobre una casilla ya jugada. El primer jugador que logra formar una fila, columna o diagonal con sus tres piezas, gana el juego.

Ejemplo de partida de tres en raya, ganada por **X**:

Ejemplo de una partida que termina en empate:

En este proyecto, usted implementará una aplicación gráfica de JavaFX que permitirá a una persona jugar el "Tres en Raya" **contra la computadora**. Como parte del diseño de su solución, usted deberá decidir cuáles de las estructuras de datos revisadas en el curso hasta ahora son las más apropiadas para implementar el juego que se solicita.

En combinación con otros elementos, algunas de estas estructuras se utilizarán para dotar a la computadora de "inteligencia" para que pueda jugar el "Tres en Raya".

Utilidad de un Tablero

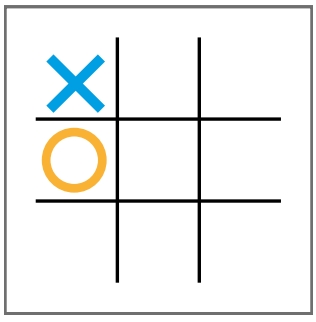
Al igual que los jugadores humanos, en un juego, la computadora debe decidir qué movimiento realizar cada vez que sea su turno. Cuando no es factible calcular de antemano todos los posibles movimientos que podrían derivar en una jugada ganadora, la computadora debe tomar una decisión óptima **en cada turno**. En general, dicha decisión considera el estado actual del juego y una función de utilidad **u** que indica lo cerca que se está de una jugada ganadora (o perdedora).

Dato un tablero `t`, la función de utilidad para el "Tres en Raya" se define como:

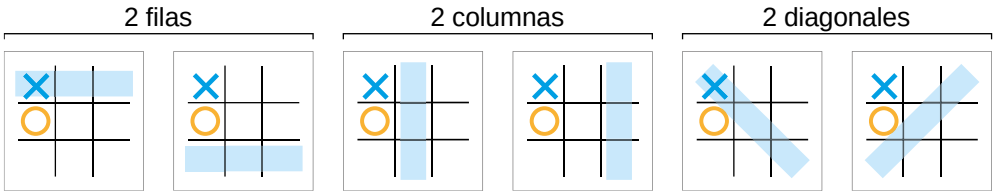
$$u_{jugador}(t) = P_{jugador} - P_{oponente}$$

donde P_j es el número total de filas, columnas y diagonales disponibles en el tablero `t` para el jugador `j`.

Considere el siguiente tablero:

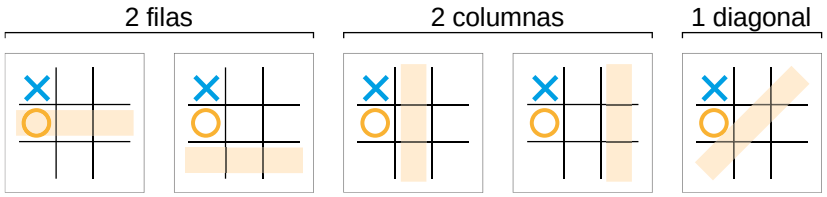


El cálculo de los valores de **P** para cada jugador se ilustra en las figuras mostradas a continuación:



P(X) = 2 + 2 + 2

P(X) = 6



P(O) = 2 + 2 + 1

P(O) = 5

Por tanto, la utilidad del tablero mostrado para el jugador que juega con el símbolo **X** sería:

$$\mathbf{u_X} = P_X - P_O$$
$$\mathbf{u_X} = 6 - 5$$
$$\mathbf{u_X} = 1$$

Decidiendo Jugadas

Utilizando la función de utilidad descrita en la sección anterior, su proyecto dotará a la computadora de una estrategia de decisión para jugar el "Tres en Raya". Dicha estrategia puede resumirse en "*Computadora, elije el mejor movimiento para ti misma, suponiendo que el humano escogerá el peor para ti*". Esta filosofía, aplicada ampliamente en el mundo de la inteligencia artificial y los juegos, se conoce con el nombre de *minimax*. En términos matemáticos, minimax consiste en **mini**mizar la pérdida **máx**ima esperada.

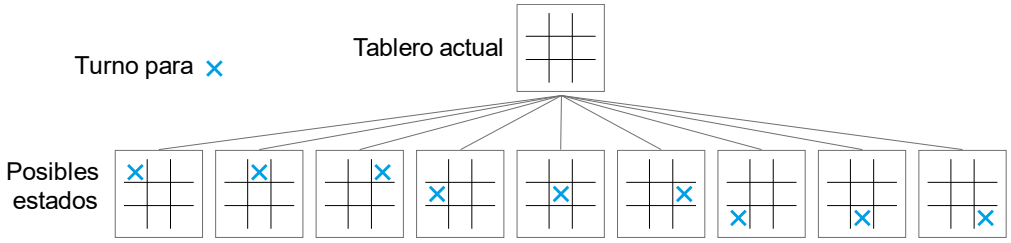
En el proyecto que usted implementará, cada vez que le toque jugar, la computadora aplicará la estrategia minimax para decidir qué movimiento realizar. Para el "Tres en Raya", el *algoritmo* de minimax consiste en:

1. Generar los posibles estados que un tablero podría tomar después de dos turnos (el propio y el del oponente).
2. Calcular la utilidad de cada uno de los tableros que podrían ser generados por el oponente.
3. Encontrar la utilidad mínima de cada familia de tableros que podría generar el oponente y asociar esa utilidad mínima al padre respectivo.
4. Elegir el tablero propio con la máxima de todas las utilidades mínimas y efectuar el movimiento que lleve a este tablero.

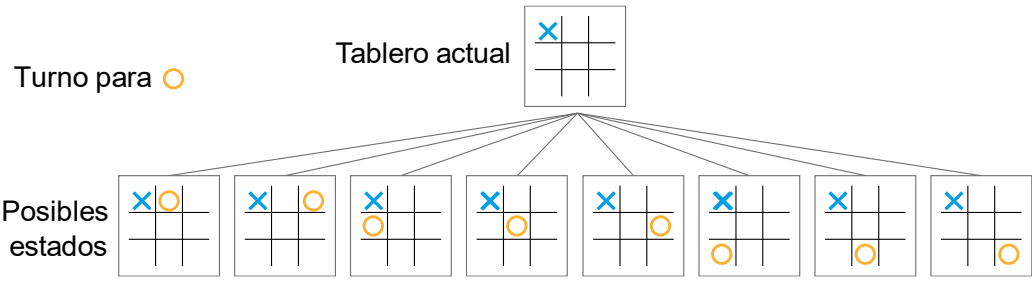
A continuación, se describe con más detalle cada uno de estos pasos.

Paso 1: Generar Posibles Estados

Generar los posibles estados de un tablero significa producir todos los tableros que podrían resultar a partir del tablero actual dado un turno específico. Por ejemplo, si el tablero actual está vacío y le toca el turno al jugador con el símbolo **X**, hay nueve posibles tableros que podrían generarse. Estos se muestran en la siguiente figura:

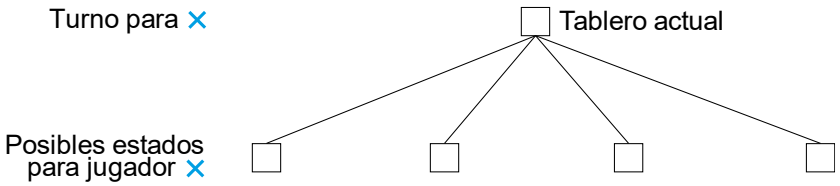


Si el turno es para el jugador con el símbolo **O**, y el tablero actual es el que se muestra a continuación, habría únicamente ocho posibles estados:

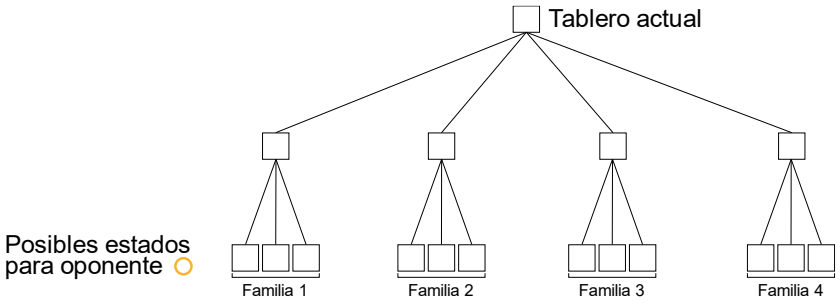


El algoritmo de minimax descrito indica que se deben generar los estados del tablero considerando **dos turnos**. Esto significa que se deben generar los posibles estados que resulten del turno actual y, por cada uno de estos, los que podrían resultar por el turno del oponente. Las siguientes figuras ilustran esto:

Primeros estados generados:



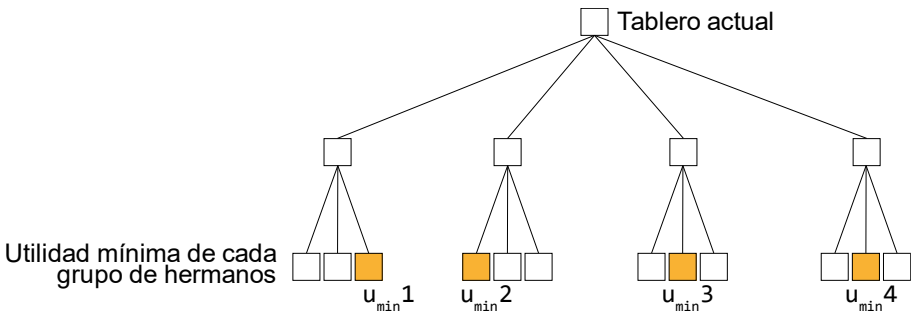
Segunda generación de estados:



Como se puede apreciar, el Paso 1 genera un árbol multicaminos. Es decir, un árbol en el que cada uno de sus nodos puede tener cero o más hijos.

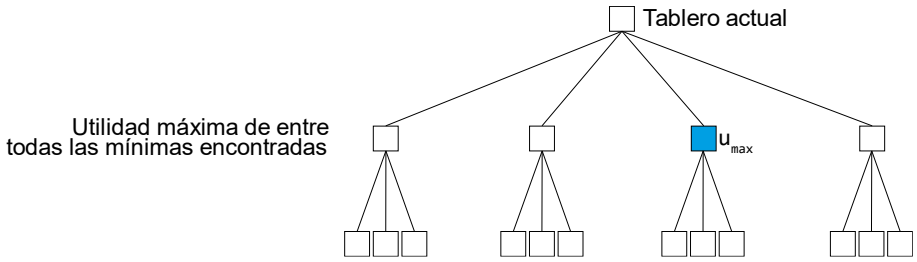
Pasos 2 y 3: Utilidades Mínimas en Tableros que Podría Generar el Oponente

El siguiente paso consiste en calcular las utilidades de cada uno de los tableros que podrían ser generados por el oponenente (las hojas del árbol mostrado en el paso anterior). Considerando estos valores, se encontrará posteriormente al tablero que garantice la **utilidad mínima** de cada familia (es decir, de cada grupo de hermanos). El valor mínimo de cada familia se ilustra en la figura de abajo mediante los tableros pintados en naranja, cada uno de los cuales tiene una utilidad u_{min}.



Paso 4: Utilidad Máxima en Tableros que Podría Generar el Jugador

Una vez que se conoce la utilidad mínima de cada familia de tableros, este valor se asocia al padre correspondiente. Finalmente, de todos los tableros que podrían ser generados por el jugador con el turno actual, se escoge aquel tablero que genera la máxima utilidad.

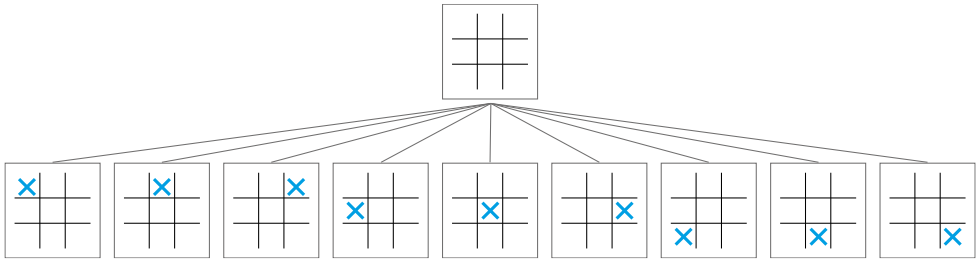


El tablero encontrado (mostrado en celeste en la imagen de arriba) indica el movimiento que debe realizar el jugador en su turno.

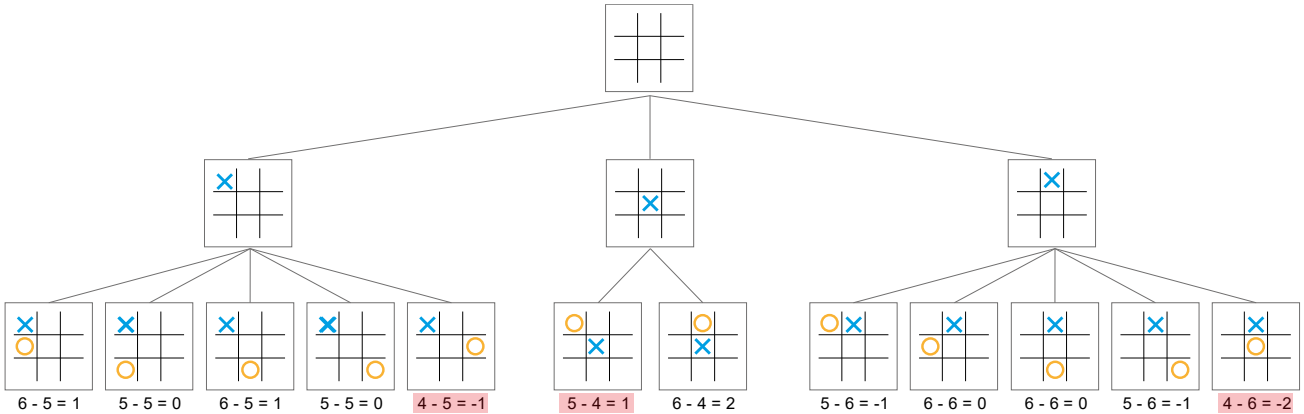
Ejemplo Ilustrativo

El algoritmo descrito en la sección anterior se ilustra aquí con un ejemplo concreto, que inicia con un tablero vacío. El turno inicial corresponde a la computadora, quien juega con el símbolo **X**. Para decidir en qué posición le conviene marcar su símbolo, la computadora aplicará el algoritmo minimax de acuerdo con lo descrito anteriormente.

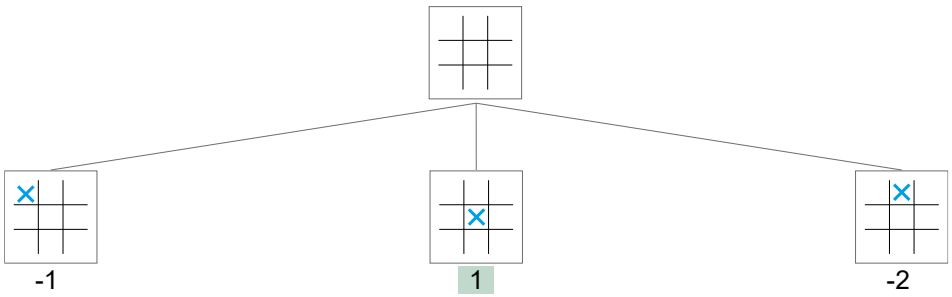
En primer lugar, se generan los posibles tableros que podría producir la jugada de la computadora:



Por **cada uno** de estos tableros, se generan aquellos que podrían resultar del movimiento hecho por el oponente (es decir, por el jugador humano). Un subconjunto de estos tableros se muestra en la siguiente figura. Note que por restricciones de espacio, no se muestran todos los posibles 72 tableros del segundo nivel ni los 9 del primero (que sí se muestran en la figura anterior).



La imagen anterior incluye también los valores de la función de utilidad de cada uno de los tableros del segundo nivel del árbol. Los valores resaltados con rojo indican el mínimo dentro de cada familia. Este valor mínimo es luego asociado a cada uno de los padres, como se muestra en la siguiente figura:



En este punto, la computadora debe escoger al tablero que garantiza la mayor utilidad. En el ejemplo dado, es el tablero con utilidad 1, que se encuentra en el centro de la figura anterior y cuya utilidad aparece resaltada con color verde.

Esto significa que, dado un tablero vacío, a la computadora le conviene colocar su símbolo en la celda central del tablero, ya que este movimiento maximiza su ganancia potencial y, al mismo tiempo, minimiza las posibles de perder.

Modelamiento del Problema y Requerimientosn Mínimos

Como probablemente habrá notado, el escenario descrito arriba puede ser modelado con árboles multicaminos y otros TDAs lineales. Dados los objetivos que se persiguen en este curso, usted y sus compañeros de grupo serán responsables de implementar los TDAs que consideren pertinentes para la solución de este problema. Usted puede utilizar colecciones lineales del Java Collection Framework (listas, pilas, colas, conjuntos, mapas), pero es responsable de implementar toda estructura no lineal que vaya a utilizar. Un requisito obligatorio de este proyecto es utilizar árboles. **Una solución que no utilice árboles invalidaría su proyecto y le otorgaría una nota automática de cero (0) a todos los miembros del grupo.**

Su juego debe consistir en una interfaz gráfica de JavaFX que permita al usuario jugar contra la computadora indicando inicialmente:

1. Los símbolos que cada jugador tendrá (es decir, quién será X y quién será O)
2. Quién inicia el juego (el humano o la computadora)

Estos parámetros deben ser indicados por el usuario utilizando controles gráficos. Usted debe decidir cómo su interfaz le permitirá al usuario efectuar estás operaciones. Asimismo, deberá decidir cómo el usuario humano indicará dónde colocar un símbolo en el tablero. Asegúrese de diseñar interacciones que sean fáciles y amigables.

Su programa debe indicar, al final, quién gana el juego o si existe un empate.

Funcionalidad Opcional

Las funcionalidades descritas arriba son obligatorias y constituyen los **REQUERIMIENTOS MÍNIMOS PARA QUE SU PROYECTO SEA ACEPTADO Y ADMITIDO A SER CALIFICADO**. No cumplir con los requerimientos mínimos otorga al grupo una nota automática de cero (0). Esto aplica, incluso, si su proyecto aplica funcionalidad extra.

Tenga en cuenta, sin embargo, que cumplir los requerimientos significa apuntar a una nota mínima. Ventajosamente, este proyecto es también una oportunidad para ser creativo. Así, usted y su grupo pueden implementar funcionalidad extra, que contribuya a una mejor versión del juego. A continuación, se muestra

una lista de algunas de funcionalidades opcionales que usted y sus compañeros de grupo podrían implementar para aplicar a puntos más allá de la nota mínima:

- Hacer recomendaciones al usuario humano sobre qué movimiento le conviene hacer.
- Mostrar los tableros intermedios generados cada vez que la computadora debe tomar una decisión junto con los valores de la función de utilidad.
- Mostrar el árbol general que se genera desde que el juego comienza hasta que éste termina.
- Permitir un modo de juego en el que la computadora juega contra sí misma.
- Permitir un modo de juego en el que dos humanos juegan uno contra otro.
- Guardar y reabrir partidas a medio jugar
- Etcétera

Usted y sus compañeros de grupo son libres de implementar cualquier otra funcionalidad extra que consideren pertinente para los objetivos del curso de Estructuras de Datos. Recuerde, esta es una oportunidad para ser creativos.

La reproducción de sonidos durante el juego NO SERÁ CONSIDERADO COMO FUNCIONALIDAD que merezca puntos extra. Esto no significa que su juego no puede incluir sonidos. Sin embargo, reproducir sonidos en Java es trivial y no aporta mucho al contenido del curso de Estructuras.

Entregables

El proyecto de Netbeans que implemente la interfaz gráfica final de su proyecto, con -al menos- las funcionalidades mínimas detalladas anteriormente, debe ser entregado hasta las 8:00 pm del día de nuestra penúltima sesión teórica del semestre. **La fecha específica se indica en la tarea del Aula Virtual, y depende del paralelo en que usted está registrado(a)**.

Usted y sus compañeros de grupo deberán presentar su proyecto al profesor en vivo, en una reunión que tendrá lugar en nuestra última sesión del semestre, el nuestro horario usual de clases. La reunión tendrá una duración de 10 minutos por grupo, será grabada, y se efectuará únicamente entre **TODOS** los miembros del grupo y el profesor. Su grupo deberá mostrar su proyecto en funcionamiento con los requisitos mínimos y cualquier funcionalidad adicional que hayan implementado. Durante la reunión, cada miembro del grupo deberá responder una pregunta sobre el proyecto. Finalmente, en la reunión recibirán feedback sobre su entrega. No se entregará feedback adicional escrito posteriormente.

Su entrega deberá incluir además un archivo **.docx** con:

- *Screenshots* de su interfaz,
- ULR del repositorio donde está alojado su código,
- Tabla de co-evaluación

Su proyecto de Netbeans y el archivo **.docx** deben ser entregados a través de SidWeb en un único archivo comprimido **.zip**.

Solo en caso de que el Aula Virtual presente problemas técnicos el día de la entrega, usted deberá alojar su archivo de respuesta (el archivo **.zip**) en una carpeta compartida de Google Drive. En este caso, usted deberá enviar un correo electrónico a la dirección gmendez@espol.edu.ec indicando el enlace de la carpeta compartida. Asegúrese de probar que su enlace funcione sin necesidad de permisos especiales. Asimismo, asegúrese de **no sobrescribir el contenido de la carpeta después de la hora máxima de entrega**. De otro modo, la hora de entrega de su respuesta reflejará cualquier sobreescritura en la metadata de la carpeta. **Utilice este método SI, Y SOLO SI, el Aula Virtual tiene problemas.**

TODOS los archivos detallados arriba (**.docx, .zip y su proyecto de NetBeans**) deben ser nombrados con el número de su grupo precedido del String "Grupo_". Por ejemplo: Grupo_03 o Grupo_12.

Recuerde lo que se explicó en la primera clase:

> Toda instrucción de este curso está diseñada para no hacer perder tiempo a nadie. Ni al profesor, ni al estudiante. Todo incumplimiento de una instrucción que incurra en una pérdida de tiempo para el profesor **SERÁ PENALIZADO CON PUNTOS** en la actividad correspondiente.

Como se indicó en la primera clase del curso, dada la modalidad en la que estamos llevando este semestre, el proyecto no será sustentado (solicitando cambios en vivo de su código). Mas bien, su entrega será revisada y calificada con una nota que será la misma para todos los miembros del grupo. La calificación de cada estudiante resultará de multiplicar esta nota por dos factores: (1) el de auto y coevaluación, y (2) un factor calculado de acuerdo con sus respuestas en la presentación del proyecto.

No se aceptarán entregas atrasadas o incompletas.

Instrucciones Finales

Recuerde: **Este es un proyecto grupal.**

Es **OBLIGATORIO** que considere las políticas de buena conducta académica que se explicaron en la primera clase del curso respecto al plagio académico y demás violaciones del código de Ética de la ESPOL. Los autores del proyecto deben ser usted y sus compañeros de grupo. Cualquier indicio de lo contrario, será reportado a las unidades correspondientes dentro de la universidad para el tratamiento pertinente.

Finalmente, **siga TODAS las instrucciones de este documento** para evitar inconvenientes. Si tiene dudas, consulte al profesor (en lugar de asumir cosas que puede ser incorrectas). Los ayudantes del curso podrían no ser la mejor fuente para despejar dudas en cuanto al proyecto. Pregunte siempre al PROFESOR, preferiblemente, en las sesiones de Zoom.

¡Éxitos a todos!

RECUERDE

Un proyecto que no implementa la funcionalidad mínima solicitada en este documento es calificado automáticamente con **CERO** y **NO PUEDE SER DEFENDIDO**. Esto aplica, incluso, si su proyecto implementa funcionalidad opcional.

Por tanto, si su proyecto no implementa los requerimientos mínimos descritos en este documento, **NO SE PRESENTE EN EL MEETING DE ZOOM DE DEFENSA DEL PROYECTO** esperando convencer al profesor que le califique un proyecto incompleto.