

# El ataque de la mantícora

Cómo programador la nación principal se pone en contacto contigo ya que se aproxima el ataque de una nación vecina y se requiere el diseño de una interfaz de tipo MVC desde la que se pueda llevar el control del ataque, así como el avance de las tropas, teniendo las siguientes necesidades

- Selección de 2 personajes
  - Los personajes deberán de ser obtenidos desde la siguiente API <https://rickandmortyapi.com/api> - "characters"
  - Se deberá tener una vista paginada del listado de personajes con filtro por nombre
  - Vista detalle del personaje en dónde se encuentre su foto y sus datos personales (nombre, especie, tipo y género)
- Cada defensor deberá tener su inventario
- Cuando se seleccione a un personaje se le deberán asignar 100 oros para que pueda equiparse en la tienda
- Tienda para adquirir diferentes artículos con diferente valor de ataque
  - Gran cañón - 80 oros - x 50 alcance
  - Metralleta - 60 oros - x 40 alcance
  - Mosquete - 50 oros - x 30 alcance
  - Pistola - 30 oros - x 20 alcance
  - Granada - 10 oros - x 10 alcance
- La información de la nación atacante deberá de ser obtenida desde la misma api pero en la sección "locations", se deberá obtener una sola de las opciones disponibles de manera aleatoria cada que comience una partida nueva

Una vez que los defensores hayan sido configurados y equipados el ataque de la nave enemiga (mantícora) comenzará el asedio a la ciudad teniendo las siguientes consideraciones:

- Establecer el estado inicial del juego teniendo la vida de la Mantícora en 10 puntos y la vida de la ciudad en 15 puntos
- Cuando se inicie el juego
  - Mostrar la información de la nación atacante (ubicación obtenida) mostrando los datos (nombre, tipo, dimensión, cantidad de habitantes)
  - Mostrar la información de personaje seleccionado (nombre, especie, tipo, género)
- Por cada tiro la posición de la Mantícora cambiará en un rango entre 10 y 50 aleatoriamente en múltiplos de 10 y no se deberá repetir con la posición anterior
- Cada defensor deberá seleccionar un arma de su inventario para realizar su ataque y disparar
- Cada defensor tendrá un máximo de 5 disparos
- En caso de que el valor de alcance del arma sea menor a la posición de la mantícora, la Mantícora recibirá un daño de 0 puntos y la ciudad de 5 puntos

- En caso de que el valor de alcance del arma sea mayor a la posición de la Mánticora, la Mánticora recibirá un daño de 2 puntos y la ciudad 1 punto
- En caso de que el valor de alcance del arma sea igual a la posición de la Mánticora, la Mánticora recibirá un daño de 5 puntos y la ciudad 0 puntos
- Por cada tiro el sistema deberá de mostrar un estatus de la ciudad y la Mánticora con la siguiente información

---

STATUS: Ronda: 1 Ciudad: 9/10 Mánticora: 15/15 Disparo: 5/5

La Mánticora está a una distancia de [distancia], el ataque con [arma] cubre una distancia de [distancia del arma] y fue un tiro (Corto - Directo - Largo)

La ciudad recibió [puntos de daño]

La Mánticora recibió [puntos de daño]

---

- El juego acaba cuando la ciudad llegué a 0 o bien la vida de la Mánticora llegué a 0
- Si después de todos los tiros del defensor 1 y 2 la vida de la Mánticora no llega a 0 entonces el juego termina con la ciudad destruida
- Al término del juego incluir un resumen con la información

Número de rondas: [#] Ciudad: [vida] Mánticora: [vida]

Daño [nombre defensor 1]: [daño] a la Mánticora

Daño [nombre defensor 2]: [daño] a la Mánticora

La Mánticora fue derribada ó La ciudad fue destruida (dependiendo del caso)

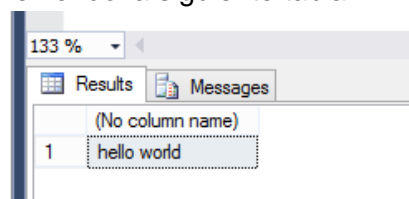
Prácticas deseables:

- Persistir la información de manera local
  - SQLite
  - Cache system
  - File system
- Hacer uso de Entity core
- Hacer uso de POO y SOLID
- Hacer uso de TDD o BDD
- Patrón de diseño

## SQL

Conteo de caracteres:

Teniendo la siguiente tabla:



(No column name)	
1	hello world

Escribe un query para determinar cuántas veces se repite cada letra, teniendo un resultado como el siguiente

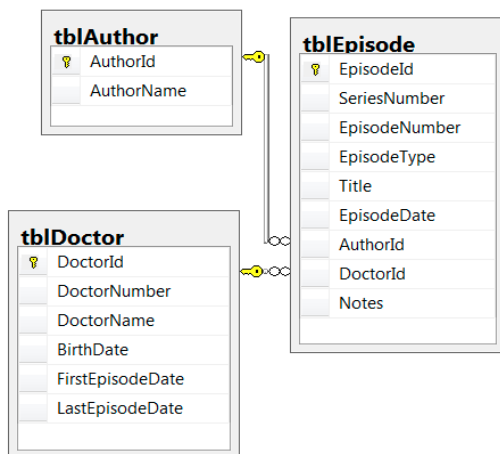
133 %

Results Messages

	letter	CountLetters
1	d	1
2	e	1
3	h	1
4	l	3
5	o	2
6	r	1
7	w	1

## Agregación

Teniendo la siguiente estructura



Escribir un query el cual nos regrese el siguiente resultado:

AuthorName	DoctorName	Episodes
Russell T. Davies	David Tennant	21
Steven Moffat	Matt Smith	20
Russell T. Davies	Christopher Eccleston	9
Steven Moffat	Peter Capaldi	6

## Queries jerárquicos

Teniendo la siguiente tabla:

Results Messages

	employee_id	First_Name	Last_Name	manager_id
1	1	Alvin	Smith	NULL
2	2	Jose	Jones	1
3	3	Amado	Taylor	1
4	4	Stuart	Williams	1
5	5	Demarcus	Brown	2
6	6	Mark	Davies	2
7	7	Merlin	Evans	2
8	8	Elroy	Wilson	7
9	9	Charles	Thomas	7
10	10	Rudolph	Roberts	7

Realizar un query para que dependiendo del número de empleado solicitado genere un resultado mostrando todo su árbol jerárquico como el siguiente

	employee_id	first_name	manager_id
1	10	Rudolph	7
2	7	Merlin	2
3	2	Jose	1
4	1	Alvin	NULL

## Reporte de faltas

Teniendo la siguiente información

	employee_id	absence_type_id	num_of_hours	absence_date
1	5	2	9	2017-01-02
2	5	2	9	2017-01-08
3	5	3	6	2017-01-05
4	6	2	6	2017-01-02
5	6	3	6	2017-01-05
6	6	8	9	2017-01-08
7	7	2	2	2017-12-07
8	7	8	5	2017-12-13
9	7	8	1	2017-12-08
10	7	8	9	2017-12-07
11	8	2	3	2017-09-14
12	8	3	2	2017-09-15
13	8	8	9	2017-09-18

Se requiere un query para que se pueda observar la cantidad de faltas y tipo de falta que se tiene por empleado teniendo el siguiente resultado

	EMPLOYEE_ID	NUM_OF_ABSENCE_TYPE_2	NUM_OF_HOURS	NUM_OF_ABSENCE_TYPE_3	NUM_OF_HOURS	NUM_OF_ABSENCE_TYPE_8	NUM_OF_HOURS
1	5	2	18	1	6	0	NULL
2	6	1	6	1	6	1	9
3	7	1	2	0	NULL	3	15
4	8	1	3	1	2	1	9

- Realizar los queries para la generación de las tablas ejemplo (complementar la estructura si así lo consideras)
- Realizar queries de solución
- Deseable que al menos una solución sea una vista, procedimiento almacenado o función. (incluir el query para su generación)

## Entrega

Para la entrega del ejercicio se requiere y se tomará en cuenta lo siguiente

- Generar un repositorio público en GitHub o GitLab
- Dentro de este se deberá poder observar los diferentes commits de los cambios realizados en sus respectivas ramas.
- Dividir la solución en dos carpetas
  - src: Dónde se alojará el código .Net

- db: Dónde se alojarán los archivos SQL
- Generar el GitFlow para el proyecto y ordenar los hitos por features con sus respectivos merges a la rama principal, manejo de commits y pull requests (**Deseable**).
- Todo el código y la solución de BD serán tomadas desde la rama principal (master/main) así que en esta rama el código deberá de poderse compilar y ejecutar.
- Arquitectura
- Buenas prácticas
- Documentación
- Tiempo de resolución de máximo 12 días, subir el progreso que se tenga en caso de no concluir el ejercicio para su validación.