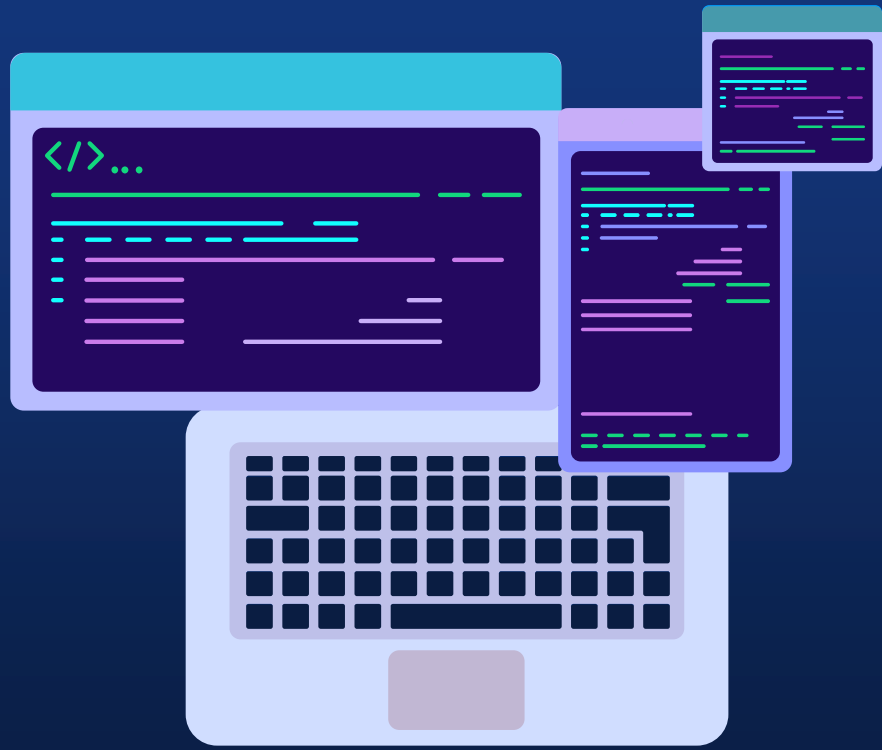
TypeScript Back-end



Mentor: MSc. Einar Rocha

CONTENT

01
Background

TypeScript, ECMAScript, Node.js..., Editor

02
TypeScript

Types, Functions, Classes, Asynchronous...

03
Storage/Persistence

Relational ORM, Typeorm

04
Software Design

OOP Pillars, SOLID

05
Clean Code

Hundreds of good practices

06
Rest Services

Best Practices

Final work

- NO UI
- No third party library for snake logic
- C4 model (level 1, level 2, level 3)
- Code
- “Real” app
- Unit test 60% coverage
- Demonstrate that all the requirements are covered in a short video
- code + C4 model + video in a Github repository
- 50% daily progress, 50% final presentation
- Postman validation

Snake API

En general se desea implementar el back-end para un juego de Snake

Requerimientos:

- Manejar una Arquitectura onion.
- El juego debe exponer un tablero de “n” por “n”, configurable.
- El juego inicia en una posición aleatoria x, y. (No esta permitido usar una librería de generación de números aleatorios ejemplo: Math).
- La víbora se mueve arriba, abajo, izquierda y derecha.
- No se muere al salir del tablero, continua en la parte opuesta del tablero.
- Cuando come comida “crece” e inmediatamente existe nueva comida.
- Se debe tener un status: “Ready to start”, “Playing”, “Ended”
- El juego debe poder reiniciarse.

Snake API

- Pierde cuando choca con alguna parte de una víbora
- El tamaño inicial de la víbora es 1.
- Timer-Based-Movement. Deberá dispararse un evento cada cierto tiempo simulando la animación y lograr que todo se mueva automáticamente.
- El intervalo de tiempo debe ser configurable.
- Solo existe un movimiento entre cada periodo de tiempo.
- Agregar un score y una lista de score mas altos.
- Agregar múltiples jugadores.

Snake API

- Cumplir SOLID
- Cumplir clean code
- Cumplir Rest API Best Practices
- Usar alguna base de datos