Ubimon

- A versão deste commit é apenas uma prova de conceito, que implementa:
  - um radar, capaz de listar outros dispositivos próximo rodando Ubimon no *smartspace*, categorizando-os em jogador, estação de troca ou *gathering point*;
  - interação com estações, permitindo acessar a estação para depositar e retirar ubimons;
  - batalhas aleatórias no mundo, com os comandos de ataque, troca de ubimon, *auraball* (tentativa de captura) e fuga, incluindo as mensagens de vitória e derrota.
- A tétrade elementar do jogo está disponível no arquivo README.md, na raiz deste repositório.
- O jogo é composto por duas aplicações: um cliente, multiplataforma (testado em PC e Android), implementado em Unity + C#; e um servidor, implementado em java e que roda somente em PC, responsável por ser um centralizador de dados de dispositivos no *smartspace* e também serve como uma estação de troca.
- No lado do cliente, foram im plementados:
  - um *port* do *middleware* para .Net, que permite a comunicação do jogo Unity com outros dispositivos uOS, utilizando uP; além do *MulticastRadar*;
  - drivers:
    - *GlobalPositionDriver*, com os serviços:
      - *getPos*, serviço discreto que retorna uma tupla <*latitude, longitude, delta*>, latitude e longitude dados em graus, delta dado em metros;
      - serviços *registerListener* e *unregisterListener* assíncronos para o evento *POS_CHANGE*, que notifica mudanças de posição, enviando os mesmos valores que *getPos*;
    - *GoogleMapsDriver*, com os serviços:
      - *updatePos*, serviço discreto que informa ao driver a posição global atual do dispositivo, passando os parâmetros <*latitude,longitude*> em graus;
      - *render*, que agenda uma atualização da textura de mapa no jogo.
- No lado do servidor, foram implementados:
  - drivers:
    - *PositionRegistryDriver*, com os serviços:

    - *checkIn (clientName, latitude, longitude, delta, metadata)*, que inicializa um cliente no registro global de posições, e retorna um *clientId* para chamadas subsequentes ao driver; se um cliente passar mais que 5 min (tempo configurável) sem atualizar seus dados junto ao servidor, ele é removido do registro e precisa fazer um novo *checkIn*;
    - *update (clientId, latitude, longitude, delta, metadata)*, que atualiza os dados de um cliente no registro;
    - *checkOut (clientId)*, que remove um cliente do registro;
    - *listNeighbours (latitude, longitude, delta, range)*, que, dada uma posição global central e um raio de busca, lista todos os clientes registrados;
- serviços de aplicação (todos são relacionados à estação de troca):
    - *enter(playerId)*, que admite um jogador na estação, desde que já não esteja sendo usada por outro jogador, se o jogador já houver previamente utilizado a estação, seus ubimon armazenados são carregados e mostrado, senão um novo registro é criado para o jogador;
    - *leave()*, se houver algum jogador utilizando a estação, o retira e libera a estação para uso por outro jogador, esta operação é realizada automaticamente após 30 segundos de inatividade por parte do jogador;
    - *cursorToLeft*, *cursorToRight*, *cursorToUp* e *cursorToDown*, todos recebendo *playerId*, que, se possível, movem o cursor dentro da estação, para mudar a seleção atual;
    - *peek(playerId)*, que retorna o ubimon atualmente selecionado na estação, caso haja algum selecionado, no parâmetro de resposta *ubimon*;
    - *removeSelected(playerId)*, que remove da estação o ubimon atualmente selecionado;
    - *store(playerId, ubimon)*, que deposita o ubimon informado na estação, desde que haja espaço disponível.