IFSP – Instituto Federal de São Paulo (Catanduva)
Tecnólogo em Análise e Desenvolvimento de Sistemas
LD1 – Lab. de Desenvolvimento I
Trabalho I

Prof. Flavio Souza - flavio.souza@ifsp.edu.br

INSTRUÇÕES

1. O Trabalho Prático deve ser feito em grupo com 3 componentes.
2. O programa (código) deve ser enviado para o professor por e-mail (flavio.souza@ifsp.edu.br) até a data limite conforme a programação de entrega parciais do trabalho.
3. Entregas parciais em datas a posteriori da data limite não serão considerados, e será atribuído a nota ZERO ao grupo naquela parcial.
4. O programa deve ser feito exclusivamente em C# Core.
5. Não são permitidos o uso de bibliotecas e/ou frameworks.
6. Para os códigos que não compilarem e/ou executarem receberão ZERO como nota.
7. Cada grupo terá um tema que delimita as funcionalidades e preocupações da API a ser desenvolvida.

TEMAS

Ranking de Futebol (Giovani, Guilherme e Yan)
Simule, por meio de uma API, a Eleição "The best" da FIFA. A eleição simulada deve ocorrer atrás de votos divididos em quatro grupos de eleitores (Técnicos, Capitães, Jornalistas e Publico Geral), considere 10 votantes para cada grupo. Cada votante tem direito a escolher os três melhores atletas, entre 10, de cada categoria. Ao final da votação ordena-se os atletas por quantidade de votos recebidos, para cada grupo votante, e atribui-se ao atleta 5 pontos para o primeiro, 3 pontos para o segundo colocado e 1 ponto para o terceiro. Os três primeiros, melhores votados são os atletas que obtiverem a maior quantidade de pontos entre os 4 grupos votantes.
A simulação deve conter os endpoints:

- **cadastroJogador:** esse endpoint possibilitará o cadastro de jogador (Id, Nome, Idade, Clube, Nacionalidade e Quantidade Votos). Sendo o Id gerado pelo simulador e a quantidade de votos inicialmente com ZERO, limitando em uma lista de 10 jogadores;
- **atualizarJogador:** nesse endpoint será possível atualizar o Nome, Idade, Clube e Nacionalidade de um jogador por Id;
- **excluirJogador:** a exclusão do jogador deve ocorrer por esse endpoint, via Id, apenas se o jogador não ter recebido nenhum voto;
- **listaCandidatos:** listar todos os 10 jogados candidatos ao prêmio, apresentando nome, idade, clube e nacionalidade;
- **votar:** através deste serviço, um votante, envia o seu voto, informando qual o grupo pertence, seu código de identificação e o voto (com os 3 jogadores escolhidos);
- **votosPorGrupo:** a considerar os votos, esse endpoint deve retornar os 3 atletas mais votados pelo grupo informando quantos votos foram recebidos;
- **theBest:** Contabilizando os mais votados de cada grupo, esse endpoint retorna os 3 atletas que receberam mais pontos entre os quatro grupos.

Sistema logístico (João Lucas e Renan)
Simule por meio de uma API um Sistema Logístico Portuário. A simulação ocorrer pelo carregamento de containers ao navio. Um container é composto por Código, Ponto de

Descarregamento (A, B, C e D como pontos de descarregamento) e peso. Para manter a estabilidade do navio, os containers devem ser distribuídos sob três pilhas regidas pela ordem decrescente de Descarregamento sem exceder o limite máximo de carga do navio.
A simulação deve conter endpoints:

- **cadastroNavio:** esse endpoint deve criar um navio, que tem um código de registro (numérico), lista de descarregamento (A, B, C e D) e limite de carga (em Tonelada).
- **alterarNavio:** a alteração de um navio ocorre por um código de registro, podendo alterar a lista de descarregamento e o limite de caga.
- **filaEmbarque:** essa função adiciona, em uma fila, um container recendo apenas o Ponto de Descarregamento, código do container e o peso do container.
- **alfandega:** pelo código do container esse endpoint possibilita a correção da carga do container, apenas se o mesmo não estiver embarcado.
- **confisco:** neste endpoint um container pode ser removido da fila de embarque pelo código do container, apenas se o mesmo não estiver embarcado.
- **carregamento:** a considerar a sequência de containers na fila e os navios disponíveis, o carregamento do container ao(s) navio(s) dever ser simulado distribuídos nas pilhas conforme o critério de carregamento.
- **descarregamento:** conforme a chegada do navio aos pontos, os containers do respectivo ponto devem ser descarregados. No descarregamento deve-se verificar a fila de embarque provocando uma reorganização na disposição dos containers.

Drive Thru (Isaias, Pedro)
Implemente uma API que simule o funcionamento de um estabelecimento de Drive Thru, na qual os pedidos efetuados no Drive Thru concorrem com os pedidos do Balcão e Delivery. Um pedido é composto por Senha e origemPedido (DriveThru, Balcao, Delivery). A realização dos pedidos deve seguir uma política de priorização. A cada 2 pedidos do Drive Thru (no máximo), um pedido do Balcão deve ser atendido. A cada 3 pedidos do Drive Thru (no máximo) ou 2 pedidos de Balcão (no máximo) 1 pedido do Delivery deve ser atendido.
A simulação deve conter as funções:

- 1º **ealizarPedido:** na qual é informado a origem do pedido e a senha deve ser gerada sequencialmente pelo simulador. O pedido tem um status, neste momento o status do pedido é AGUARDANDO.
- 2º **alterarPedido:** esse endpoint possibilita a alteração de um pedido, por meio da senha, neste caso o pedido deve ir para o final da lista de pedidos e não deverá influenciar na política de priorização. Alterar o status do pedido para ALTERADO.
- 3º **fazerPedido:** faz o papel da cozinha, sendo assim um pedido da fila deve ser feito ficando disponível para entrega. A cozinha tem um limite de atendimento em 3 pedidos. Alterar o status do pedido para FAZENDO.
- 4º **entragaPedido:** a cada pedido pronto, este deve ser entregue ao cliente, a considerar a origem. Sendo assim, os de DriveThru e Balcão são entregues imediatamente. Já o delivery, deve-se acumular 3 pedidos para efetuar a entrega. O pedido passa ao status de PRONTO.
- 5º **retirarPedido:** pela senha do pedido, o cliente pode ser rapaz de retirar o pedido caso este este com o status de PRONTO.

Concluído
Post (origem)
Put (senha)
Concluído
Patch
Concluído
Get
Emergência
Delete (senha)
Concluído

Sistema de Agendamento (André, Enzo e João Simões)
Esse sistema simula o controle de agenda médica. Um consultório pode ter 1 ou N médicos, cada médico tem um CRM e uma agenda. Todos os médicos realizam atendimento apenas nos dias úteis e com uma restrição de 5 consultas por dia ou 4 consultas e 1 cirurgia.
A simulação deve conter as funções:

- **adicionarMedico:** por este endpoint deve ser capaz de adicionar a clínica um novo médico, cada médico tem um código (CRM) e uma agenda.

- **removerMedico:** nesse endpoint é possível remove um médico da clínica considerando que o médico tenha a agenda vazia para a próxima semana.
- **listaMedicos:** esse endpoint deve listar todos os médicos da clínica.
- **agendar:** Essa função tem por propósito adicionar um compromisso, consulta (1) ou cirurgia (2), na agenda do médico por CRM. Cada agendamento de compromisso médico, é composto por ID (gerado e controlado sequencialmente pelo simulador), data do agendamento. No entanto, para agendar um compromisso, a função recebe o CRM do médico, o tipo de compromisso (consulta ou cirurgia) e o dia desejado do compromisso. (agendamentos no passado não são aceitáveis).
- **alterarAgendamento:** com base no ID do agendamento, este endpoint permite a alteração de data de um atendimento.
- **cancelarAgendamento:** Para cancelar um compromisso é necessário informar a data e o CRM do médico, ficando disponível para um novo atendimento. O cancelamento só é permitido se for um atendimento do tipo cirurgia, para as consultas deve ter um intervalo de 3 dias úteis.
- **otimizarAgenda:** Essa função deve otimizar os compromissos da agenda de um médico. A otimização reagendará os compromissos, antecipando os compromissos conforme a política, preenchendo os horários vagos.

Processo de Matrícula (Gabriel Gomes, Gustavo e João Amaral)
Implemente uma API que simule o contexto de uma secretária escolar. Durante o período de matricula os alunos (RA, nome, série e nota) são organizados em turmas. Com início das aulas o professor solicitar o diário de classe da Turma e ao decorrer lança as notas dos respectivos alunos. Por fim, ao final do curso, a secretaria disponibiliza um relatório de Aprovados e Reprovados.
A simulação deve conter as funções:

- **realizarMatricula:** a matrícula de um aluno será representada por esse endpoint que recebe o Nome e série do aluno. O endpoint deve gerar o RA do aluno (que é definido pelo código da série seguido do número de matriculados na série; Ex. RA=0218, o aluno matriculado está na segunda série [02] sendo decimo oitavo matriculado na turma).
- **alterarMatricula:** esse endpoint permite alterar a matrícula de um aluno por um RA, a alteração pode ocorrer no Nome e na série (apenas quando a Nota for nula). Na alteração de série um novo RA deve ser gerado.
- **cancelarMatricula:** o aluno pode fazer o cancelamento matricula, pelo RA, apenas se o referido aluno estiver sem nota.
- **criarDiario:** informando a séria, a função deve criar uma lista de alunos matriculados para a séria em ordem alfabética.
- **lancaNota:** com base no RA do aluno, a nota do respectivo aluno deve ser atualizada.
- **aprovadosResprovados:** essa função deve ser um relatório que informa, em ordem alfabética e separadamente, quais os alunos aprovados (nota >= 6.0) e reprovados (nota < 6.0)

Escalonamento de Processos (Vitor, Paulo e Maxuel)
Implemente uma API que simule o escalonamento de processos de um sistema operacional. A simulação é composta por um processo, contendo os atributos ID, Número de Ciclos e Status (á inciar, em espera e em execução). A gestão da política do Escalonador deve ocorrer uma fila (limitado em 30 processos). Sendo uma fila, o Escalonador deve operar sobre FIFO. A simulação deve conter as funções:

- **adicionarProcesso:** recebendo apenas o número de ciclos, sendo o ID definido pelo simulador (sendo este único). O processo deve ser adicionado no final da fila. A função deve informar caso esteja cheia.
- **atualizarProcesso:** pelo ID do processo, o endpoint irá atualizar, para mais, a quantidade de ciclos.

- **encerrarProcesso:** esse endpoint permite a finalização do processo pelo ID, independentemente da quantidade de ciclos, exceto se o processo estiver no status de "em execução".
- **executarProcesso:** O processo que está na primeira posição da fila deve ser "processado", passando para o status de "em execução" e simbolizado pelo decremento do número de ciclos do processo. Após a execução, o processo deve ir para o final da fila caso tenha ciclos á executar.
- **estadoProcesso:** com base no ID do processo, esse endpoint possibilitará a verificação do status do processo (ID, número de clicos e status de execução)
- **estadoEscalonador:** A função deve informar o estado do escalonador, quantos processos tem na fila e o ID que está em execução.

Bingo (João Querino, Mateus Colombo e Rafael)
Simule, por meio de uma API, um sistema de aposta. Cada apostador tem ao menos uma cartela com 9 únicos números (entre 1 e 100). Ao iniciar o bingo, um número é sorteado (entre 1 e 100). A cada sorteio deve-se verificar o acerto do número na cartela de cada apostador (desconsidere o ato "comer barriga"). No final de cada rodada, após o sorteio e a verificação, deve-se apresentar uma lista em ordem de maior número de acertos entre os apostadores. O primeiro apostador que completar a cartela vence o Bingo, este deve ser informando como ganhador.
A simulação deve conter as funções:

- **adicionarApostador:** neste endpoint um novo apostador deve ser gerado, com o ID (gerenciado pelo simulador) e a lista, inicialmente vazia, de cartelas.
- **removerApostador:** com base no ID do apostador, esse endpoint exclui todas as cartelas e o próprio jogador
- **adicionarCartela:** pelo ID do apostador o endpoint deve gerar uma nova cartela com 9 números únicos (com valores aleatórios e não repetidos na cartela).
- **rankingDeAcertos:** o endpoint deve contabilizar e construir o ranking com de acertos de entre os apostadores.
- **probabilidade:** esse endpoint deve calcular a probabilidade de vitória para cada um dos apostadores, levando em consideração o percentual de acertos e os números ainda não sorteados.
- **sorteioNumero:** esse endpoint sorteia um número (entre 1 e 100, ainda não sorteado). A cada sorteio verifica-se os acertos de cada cartela dos apostadores.