# TRABALHO PRÁTICO
# AVALIAÇÃO CONTÍNUA

| | |
|---|---|
| **Unidade:** | Programação II |
| **Ano Letivo:** | 2022/2023 |
| **Curso:** | Engenharia Informática |
| **Ano:** | 2º **Semestre:** 1º |
| **Data-limite submissão:** | 06 de janeiro de 2023 |

O presente trabalho prático tem como objetivo desenvolver as capacidades dos estudantes em programação de software, no paradigma orientado a objetos, assim como permitir a avaliação da componente prática da disciplina Programação II do curso de Engenharia Informática. Leia atentamente todo o enunciado e contacte os docentes da disciplina em caso de dúvidas.

- Cada **grupo de trabalho** deve ter **exatamente 2 alunos, pertencentes à mesma turma.**
- Cada trabalho prático deverá seguir os requisitos de um dos temas abaixo descritos (A, B ou C), assim como os requisitos gerais. A afetação dos grupos aos temas será realizada pelo docente das práticas após definição dos grupos. A afetação será disponibilizada no Moodle da disciplina.

## REGRAS

- Para o desenvolvimento dos trabalhos propostos, apenas poderá ser usada a linguagem de programação Java, segundo o paradigma de orientação a objetos.
- O executável resultante deste projeto deverá manter a interoperabilidade característica da linguagem Java, isto é, deverá ser suportado nos sistemas operativos mais utilizados e que suportam Java, nomeadamente Linux, Mac e Windows.
- Os pormenores de implementação que se devem à interpretação dos enunciados por parte dos grupos de alunos deverão ser claramente descritos no relatório com detalhe e justificação das opções tomadas.
- A implementação de funcionalidades extra não presentes no enunciado será valorizada, desde que estas funcionalidades não modifiquem os requisitos obrigatórios e não reduzam a dificuldade do trabalho. As funcionalidades extra

implementadas deverão ser documentadas no relatório.

- A apresentação de relatórios e/ou implementações **não originais e que constituam plágio**, conduzindo à imediata atribuição de **nota zero** no trabalho de grupo e a **eventuais processos disciplinares**.

## AVALIAÇÃO E ENTREGA

- O trabalho prático faz parte da avaliação da Componente Prática da disciplina de **Programação II (correspondendo a 50% da nota final).**
- A nota é atribuída individualmente aos elementos do grupo segundo a apresentação, visualização e discussão dos elementos entregues e as impressões obtidas pelos docentes acerca do aluno durante o decorrer das aulas de acompanhamento.
- Para aprovação à disciplina, a nota da Componente Prática deverá ter a classificação mínima obrigatória de **8.0 valores**.
- O trabalho prático é obrigatório em qualquer época de avaliação.
- O trabalho prático será defendido em uma sessão própria, em horário previamente definido e anunciado no Moodle.
- As discussões orais deste trabalho serão realizadas no dia indicado no calendário e no horário previamente definido para o efeito. Os grupos de trabalho devem reservar o dia e hora marcados, na sua agenda. Os trabalhadores-estudantes devem, ao abrigo do respetivo estatuto, solicitar a folha justificativa de exame para a empresa.
- Os alunos não aprovados no trabalho prático, durante a época da avaliação contínua (época de frequência) deverão apresentar e defender um trabalho com novo enunciado a solicitar ao docente, sendo que este deverá ser entregue e defendido durante a época de exame de RECURSO do primeiro semestre, em data e hora a anunciar oportunamente.
- O trabalho prático deverá ser submetido através do Moodle seguindo as instruções lá indicadas. A entrega deverá conter os seguintes elementos:
    - Código fonte em Java;
    - Manual de utilização (até 10 páginas). Alternativamente poderá ser entregue um vídeo com duração máxima de 5 minutos com instruções de utilização e

demonstração de funcionalidades;

- Relatório escrito (até 10 páginas)

Nota: não serão contabilizadas as páginas do relatório ou manual (se escrito) que excedem os limites estabelecidos. O relatório deverá utilizar tamanho de letra 12, com espaçamento 1,5. Para a contagem do número de páginas não serão consideradas capas, páginas de título, índices, glossários e páginas em branco.

- Relativamente ao código entregue, serão valorizados os seguintes aspetos:
  - Execução, eficiência e robustez;
  - Legibilidade do código, utilização das convenções do Java em termos de nomes, etc.;
  - Comentários (bónus se forem utilizados comentários em JavaDoc);
  - Qualidade geral do código e facilidade de manutenção;
  - Cobertura com testes unitários (opcional)

# Tema A – Sistema de rastreio de tempo em tarefas

## Descrição do Problema

Este projeto consiste no desenvolvimento de um sistema de rastreio de tarefas para os *freelancers* poderem anotar o trabalho realizado diariamente, de forma a ajudar na cobrança do valor adequado pelos serviços.

## Funcionalidades a incluir

1. Um utilizador pode criar uma conta. Para aceder ao sistema, o utilizador deverá efetuar login.
2. O utilizador pode editar os dados da sua conta, nomeadamente o seu nome, dados de autenticação e número habitual de horas de trabalho diário.
3. O utilizador poderá criar um projeto para agrupar tarefas. Um projeto tem um nome,nome do cliente e preço por hora. As informações sobre o projeto poderão ser alteradas. Os projetos podem ser removidos, tendo o utilizador a opção de apagar todas as tarefas associadas ou simplesmente desassociar as tarefas.
4. Um utilizador pode iniciar uma tarefa indicando uma curta descrição e data/hora de início. Alternativamente, se não for indicada data e hora, considera-se que a tarefa inicia no momento atual. As tarefas podem ter um projeto associado e um preço hora (por defeito, o preço hora desse projeto).
5. Um utilizador pode terminar uma tarefa em determinada data e hora. Se não for indicada uma data e hora, considera-se o momento atual.
6. Um utilizador pode remover tarefas em curso ou finalizadas.
7. O utilizador pode listar todas as tarefas em curso, obtendo informação do tempo total despendido até ao momento.
8. O utilizador pode listar todas as tarefas finalizadas entre duas datas.
9. Um utilizador pode convidar outro utilizador para participar num projeto. O utilizador que é convidado para um projeto deverá aceitar o convite para poder introduzir tarefas nesse projeto.
10. O utilizador criador do projeto pode remover utilizadores convidados dos seus projetos.
11. Deverá ser possível visualizar (imprimir) um relatório pessoal para determinado mês,que deverá incluir lista de tarefas realizadas em cada dia (com indicação do projeto),número de horas total em cada dia e número de horas totais no mês. Caso alguma das tarefas tenha um preço hora associado, deverá ser indicado o preço total por diae preço total do mês. As tarefas em curso não deverão ser incluídas no relatório. Os dias em que o utilizador excedeu o número habitual de horas diárias devem ser sinalizados no relatório.
12. Deverá ser possível imprimir um relatório mensal semelhante ao descrito na linha acima, mas ao nível do projeto e ao nível do cliente, indicando adicionalmente todos os utilizadores que executaram as tarefas a cada dia.

# Tema B – Plataforma de gestão de talentos IT

## Descrição do Problema

Este projeto consiste na implementação de um sistema de gestão de currículos de profissionais IT, nomeadamente *developers*, *ux specialists*, *product managers* e *project managers.*

## Funcionalidades a incluir

1. Um utilizador pode criar uma conta. Para aceder ao sistema, o utilizador deverá efetuar login.
2. Um utilizador pode criar e editar *skills (React, C++, TimeManagement, etc.)*, indicando um nome e selecionado a área profissional (*developer, ux, ...*). Uma *skill* só pode ser apagada se não estiver atualmente associada a nenhum profissional. As*skills* são partilhadas entre todos os utilizadores do sistema.
3. Um utilizador pode criar um perfil de um talento indicando o nome, país, e-mail, e preço por hora. Os perfis podem ser públicos ou privados, isto é, visíveis ou não para todos os utilizadores do sistema.
4. Um utilizador pode criar um cliente na plataforma ao qual pode apresentar determinado talento.
5. Cada perfil de talento deverá ter várias *skills*, indicando o número de anos de experiência para cada *skill*.
6. A cada perfil poderá ser adicionado detalhe de uma determinada experiência, indicando um título, nome da empresa, ano de começo e ano de término (opcional,caso o talento ainda trabalhe nessa empresa). Considere que não pode haversobreposição de experiências no mesmo ano.
7. Permitir pesquisar talentos por uma combinação de *skills*. Em caso de interface na linha de comandos, as *skills* deverão ser introduzidas num único comando separadas por virgula (ex.: “Java,Spring,JavaScript”, permitiria pesquisar por todos os talentos que tem as *skills* “Java”, “Spring” e “Javascript”. Os resultados desta pesquisa deverão ser ordenados por nome de talento.
8. Um utilizador pode registar propostas de trabalho para os seus clientes. A proposta de trabalho deverá ter um nome, categoria de talento, *skills* necessárias, número mínimo de anos de experiência por *skill* e indicação do número total de horas e descrição do trabalho. As propostas de trabalho podem ser atualizadas ouremovidas.
9. É possível listar, para uma proposta de trabalho, todos os talentos existentes nosistema que são elegíveis para essa proposta, ordenados por valor total.
10. Permitir obter um relatório com preço médio mensal (considere um mês como 176horas) por categoria de talento e por país.
11. Permitir obter um relatório com preço médio mensal (considere um mês como 176horas) por *skill*.

# Tema C – Gestão de um stand de automóveis

## Descrição do Problema

Pretende-se elaborar um programa que permita fazer a gestão de veículos disponíveis, de vendas e de clientes de um stand de automóveis.

## Funcionalidades a incluir

1. Deve ser possível a criação de novos clientes. Este registo deve ser feito pelo próprio cliente. Assuma que todos os clientes são registados no programa, não sendo possível realizar vendas a um cliente não registado;
2. O cliente deve conseguir aceder ao seu perfil, onde pode ver a sua informação e onde a pode modificar;
3. O cliente deve conseguir aceder à listagem de veículos, e deve conseguir ver a informação de um dado veículo;
4. O cliente deve conseguir reservar um veículo para compra. Na reserva, o cliente deve conseguir marcar uma data na qual se irá deslocar ao stand para efetuar a compra do veículo. Esta reserva altera o estado do veículo para reservado. Quando um veículo está neste estado, não deve ser possível outro cliente reservar o veículo;
5. O cliente deve conseguir aceder a um histórico de compras e reservas;
6. O dono do stand deve conseguir adicionar, modificar ou desativar veículos do stand. Veículos desativados não aparecem na listagem de veículos disponíveis para compra, mas continuam a aparecer no histórico de compras e reservas do cliente;
7. O dono do stand deve ter acesso a uma listagem onde consegue visualizar todos os veículos que estão registados no stand (incluindo desativados);
8. O dono do stand deve conseguir validar ou cancelar uma reserva. Se a reserva for cancelada, o veículo passa a estar novamente disponível para ser reservado por um cliente;
9. O dono do stand deve conseguir registar uma venda de um veículo. Quando o veículo é vendido, este passa a deixar de ser listado nos veículos disponíveis para compra.
    a. Se a venda for de uma reserva criada por um cliente, essa reserva passa para um estado concluída;
    b. Se a venda for de um veículo que não tenha sido reservado, o dono deve conseguir associar a venda ao cliente;
10. O dono do stand deve conseguir aceder a uma listagem onde verifica os próximos clientes que se irão deslocar ao stand (isto é, uma lista ordenada pela data indicada na reserva de um veículo);
11. O dono do stand deve conseguir aceder à informação de um dado cliente.
12. Cada tipo de utilizador identificado apenas deve ter acesso às funcionalidades que lhe dizem respeito.

## REQUISITOS GERAIS (PARA TODOS OS TEMAS)

1. Criar 3 níveis de autenticação no sistema: *User* (pode aceder aos seus registos ou a registos consigo partilhados); *UserManager* (pode criar utilizadores, editar permissões na aplicação e visualizar todos os registos); *Admin* (pode realizar qualquer ação no sistema; criar/editar utilizadores (de todos os tipos), alterar permissões e visualizar todos os registos disponíveis no sistema). Quando a aplicação inicia pela primeira vez (ainda sem utilizadores), deverá automaticamente criar um único utilizador do tipo *Admin* com um nome de utilizador e password conhecidos.
2. Toda a gestão de entidades deve ser executada em coleções (List, Map, Set), em memória. No início e no fim da execução, os dados da aplicação poderão ser persistidos em ficheiro(s). Poderá utilizar ficheiros de texto (exemplo: formato CSV) ou em formato binário através da serialização de classes (ver interface *Serializable*).
3. A aplicação deverá ter uma interface (com menus) que funciona através da linha de comandos. A aplicação deverá receber como argumento o(s) ficheiro(s) com os dados anteriormente gravados ou o nome de um novo ficheiro.
4. Os dados introduzidos na aplicação devem der validados.

**Bónus:**

1. Utilização de comentários no formato JavaDoc nos atributos e métodos das classes.
2. Criação de histórico de ações ou comandos realizados no sistema, indicando o nome da ação realizada, o utilizador que a realizou e data/hora. Cada utilizador pode consultar o seu histórico, sendo que os Admins podem consultar todo o histórico de utilização da aplicação.
3. As passwords deverão ser encriptadas (exemplo: ver classes *SecureRandom* e *MessageDigest*)
4. Implementação de interface gráfica (substitui a implementação de uma interface baseada em consola).
5. Utilização de ferramentas de versionamento de código, como o GIT.