**Slides 1/2**

Olá, boa tarde a todos. Hoje, eu e os meus colegas viemos apresentar o trabalho prático proposto no âmbito desta UC.

Nesta apresentação iremos falar de vários tópicos realizados ao longo deste trabalho, que são:  
Definição do Sistema; Levantamento e Análise de Requisitos; Modelação Conceptual; Modelação Lógica; Implementação Física; Implementação do Sistema de Migração de Dados; Conclusão e Trabalho futuro.

**Slide 3**

**- Definição do Sistema**

Para entenderem melhor toda a implementação do projeto, é necessário fornecer em antemão, um contexto da empresa. A Hole in One é uma empresa de aluguer de carros de golfe fundada em 2004 por Orlando Esbelto, após um acidente que o afastou do golfe. Sediada em Vilamoura, começou no campo Old Course e expandiu-se para o Porto (City Golf) e Braga (Clube Golfe Braga). Atualmente, conta com três stands e seis funcionários (três gerentes e três secretárias).

Com o crescimento da Hole in One e o aumento da procura, o sistema em papel tornou-se ineficaz. Para modernizar a gestão e melhorar a eficiência, o Sr. Esbelto decidiu investir numa base de dados e por isso, foram definidos os seguintes objetivos: facilitar o registo e a gestão de clientes, funcionários, carros e alugueres; Organizar e estruturar a informação de forma eficiente; Apoiar a gestão de recursos humanos e financeiros e Melhorar o contacto e acompanhamento dos clientes.

**Slide 4**

Apesar disto, a implementação da BD tem de ser viável, isto é, o custo da sua implementação terá de trazer alguns lucros que cubram a despesa. Contudo, para a Hole in One, o investimento é viável e será recuperado em cerca de 3 anos, com previsão de aumento de 22% nos lucros devido ao crescimento do turismo e popularidade do golfe.

E em termos de Recursos, foram contabilizados os da Empresa ( Orlando Esbelto – Fundador principal interlocutor; José Calado – Gerente (Vilamoura); Rogério Samora – Gerente (Porto); Edgar Novo – Gerente (Braga) ). Estes foram escolhidos por serem os que apresentam maior responsabilidade e, por isso, uma melhor perceção das necessidades da empresa. Em relação aos recursos materiais, foram usados o MySQL Workbench para implementar o modelo lógico, o brModelo para criar o modelo conceptual e os computadores pessoais dos membros do grupo.

**Slide 5**

O plano de execução foi criado com o intuito de garantir a mais alta qualidade do relatório realizado, e por isso, foi seguido à risca, o que leva o grupo a sentir-se bastante confiante o seu resultado final.

**Slide 6**

**- Levantamento e Análise de Requisitos**

Passando agora para o Levantamento e Análise de Requisitos.

Para a obtenção de requisitos, foram usadas várias formas de levantamento, o que permitiu ao grupo obter informação de uma maneira mais ampla e, assim, garantir que a implementação da BD garante o normal funcionamento da empresa e tem em conta o seu crescimento futuro. Para isto, foram realizadas Reuniões por videoconferência com os fundadores e gerentes, o que permitiu identificar tarefas diárias que poderiam ser otimizadas na BD e foi realizada uma reunião pessoal com o Sr.Esbelto, o que permitiu recolher perceções sobre o impacto esperado da BD. Para além disso, foi realizado um questionário ao fundador que avaliou expectativas e necessidades específicas e uma análise documental que permitiu compreender o processo de aluguer e a informação existente sobre clientes.

**Slide 7**

Em relação à organização dos requisitos, estes foram divididos em 3 categorias:

Descrição – que retém informações sobre entidades, atributos e restrições;  
Manipulação – que são as operações possíveis na base de dados;

Controlo – que concedem permissões e acessos atribuídos aos utilizadores.

Como podem ver, termos aqui 1 exemplo de cada um destes. Estes requisitos foram usados no relatório e foram tidos em conta mais à frente na implementação da BD.

**Slide 8**

Depois da obtenção dos requisitos, foi necessário validá-los com os gerentes e o fundador, de maneira a confirmar e corrigir alguns destes. Para isso, foi definida uma reunião onde se identificou que era importante adicionar uma data de nascimento ao Cliente e os números de lugares que um Carro suporta. Depois disto, os requisitos foram validados e o grupo seguiu para a MODELAÇÃO CONCEPTUAL.

**Slide 9**

**- MODELAÇÃO CONCEPTUAL**

Nesta parte do projeto, foram identificadas as entidades, os relacionamentos e os atributos usando os requisitos definidos na fase anterior. ( FALAR DAS PRINTS ? )

**Slide 10**

De seguida, como podem ver, foi criado o modelo conceptual usando o brModelo, tendo em conta as tabelas mostradas no slide anterior.

**Slide 11**

**- Modelação Lógica**

Passando para a Modelação Lógica, o grupo usou o MySQL Workbench para derivar o Modelo Lógico a partir do Modelo Conceptual. Para fazer isto, o grupo usou as 3 primeiras fórmulas normais para garantir integridade, consistência e controlo de redundância. Com isso, surgiu o seguinte Modelo Lógico.

**Slide 14**

**- Álgebra Relacional**

Após toda a construção do modelo lógico e da normalização dos dados, o nosso grupo seguiu para uma validação do mesmo, implementando uma série de queries previamente enunciadas nos requisitos de manipulação. Estas foram traduzidas em expressões de Álgebra Relacional, como poderemos ver neste slide:

- RM4

- RM5

- RM6

**Slide 15**

**- Implementação Física**

Tendo o trabalho anterior em conta, passamos para a implementação física. Para isto, foi necessário desenvolver bastante código SQL que correspondesse ao trabalho prévio. Em geral, foi necessário criar a BD e as suas tabelas, de acordo com a ordem definida pela existência de chaves estrangeiras nas diferentes tabelas. Tal como em todas as outras tabelas, foi necessário definir o código de acordo com o modelo lógico, o que levou a que, por exemplo, a tabela Cliente fica-se, como podem ver, com os respetivos atributos e chave primária. Para as restantes tabelas, o procedimento foi o mesmo.

**Slide 16**

Para garantir a segurança e o controlo do acesso à base de dados, foram criados diferentes ROLES com permissões especificas, conforme os requisitos de controlo, tal como o Dono, Gerente e Secretário.

**Slide 17**

Para inserir dados nas 6 tabelas da base de dados, isto é, para fazer o povoamento da BD, foram utilizados 3 métodos diferentes: Ficheiros CSV; Ficheiros JSON e SQL (INSERT INTO).

**Slide 18**

De seguida, calculamos uma aproximação do tamanho que a BD teria para que fosse possível prever as necessidades de hardware e garantir a escalabilidade futura do sistema. Para isso, consultamos a documentação oficial do MySQL e obtivemos o tamanho que cada tipo de dados ocupa em bytes. Depois, sabendo o tamanho de um registo em cada tabela, usamos os dados com os quais a BD iria ser populada e calculamos uma estimativa do tamanho que a BD tomaria, depois de populada. Com isso, chegamos ao valor de 6.8kB.

**Slide 19**

**-View**

Tal como nos foi pedido, implementamos uma vista para a BD. Apesar de não nos ter sido necessária a implementação de uma vista nesta BD, a equipa achou que, ainda assim, deveria ser implementada uma pelo simples facto de exemplificação. Com isso chegamos à seguinte vista:

A vista “NumeroCarrosAlugadosPorFuncionariosTipo2” foi criada com o objetivo de representar um subconjunto de informação da BD, que inclui apenas números dos carros, cujos alugueres foram realizados por funcionários com ID\_Funcao = 2, ou seja, funcinários do tipo gerente.

**Slide 20**

Durante a análise dos requisitos e validação do modelo lógico, foram identificadas diversas

interrogações (através dos requisitos de manipulação) que os utilizadores do SBD poderão

realizar no seu dia a dia. Estas interrogações mostram necessidades operacionais e de gestão,

tais como:

• Consultar dados de clientes e funcionários;

• Verificar e aceder a informação sobre alugueres;

• Realizar a associação de diferentes tabelas;

- RM1

- RM2

- RM6

- RM7

**Slide 21**

**-Procedimentos**

Um procedimento é um bloco de código reutilizável que recebe determinados parâmetros e executa um conjunto de instruções para realizar uma tarefa específica, como por exemplo modificar dados, executar cálculos ou interagir com bases de dados.

Neste slide somos capazes de ver um exemplo de um procedimento implementado pelo nosso grupo de trabalho, sendo ele o procedimento “Adiciona\_a\_User\_e\_Telefone”, que através da passagem de vários argumentos, este, tal como o seu nome indica, insere um novo cliente na BD e, simultaneamente, regista o respetivo número de telefone, garantindo que ambas as operações são realizadas de forma segura e consistente através de uma transação (TRANSACTION), que assegura que este processo é atómico.

**Slide 22**

**-Trigger**

Um trigger é uma estrutura especial armazenada na base de dados, que é executado automaticamente em resposta a certos eventos, como:

* Inserção de dados (INSERT);
* Atualização de dados (UPDATE);
* Eliminação de dados (DELETE);

Neste slide, podemos ver o trigger definido pelo nosso grupo, onde o mesmo é responsável por, antes de uma inserção na tabela Aluguer, verificar se a data inserida na nova instância de Aluguer pertence a um intervalo de tempo ao qual o carro já esteja alugado.

Se essa situação ocorrer, o trigger gera um erro e não realiza a dada inserção.

**Slide 23**

**-Function**

Semelhante a um procedimento, uma função também é um bloco de código reutilizável que recebe determinados parâmetros e executa um conjunto de instruções para realizar uma tarefa específica, onde, diferente de um procedimento, esta retorna um valor.

Aqui somos capazes de observar uma das funções realizadas no decorrer do nosso trabalho, sendo ela a função “TotalAlugueresCliente”, onde a mesma recebe como argumento um INT que corresponde a um id de um cliente e retorna o número de alugueres realizados por esse cliente.

**Slide 24**

**-Index**

Os índices são estruturas de dados que aceleram a busca e acesso a dados numa

tabela. Na realização deste trabalho prático decidimos que os índices poderiam ser utilizados em algumas das relações que criamos no contexto da nossa BD, apesar de que, devido à dimensão do projeto, a diferença do tempo computacional da realização das operações que necessitamos foi, antes e depois da criação dos índices, praticamente, nula.

Neste slide, estão representados dois dos indexs criados na BD, onde o primeiro é feito sobre os atributos Data\_Inicio e Data\_Fim da tabela Aluguer e o segundo é feito sobre o atributo ID\_Funcionario também da tabela Aluguer, permitindo-nos acelerara buscas, filtragens e ordenações nesses mesmos atributos.

**Slide 25**

**-Implementação do sistema de migração de dados**

Nesta secção do trabalho, tivemos como objetivo integrar dados provenientes de três fontes distintas, sendo elas ficheiros CSV, JSON e através de instruções SQL diretas.

De modo a implementar este sistema, optamos pela criação de um programa escrito em python, onde o mesmo recolhe os dados das três fontes distintas, constroí dinamicamente instruções SQL de inserção de registos e envia-as para o servidor para serem executadas.

O programa é constituído por três funções principais, cada uma relativa a uma das

fontes utilizadas, além, também, da função main e de uma função auxiliar usada na fonte

JSON.

* Inserir\_dados\_CSV: Lê um ficheiro **CSV** linha a linha e insere os dados na tabela da **BD**,referente ao ficheiro lido, através de instruções **SQL** do tipo **INSERT INTO** criadas dinamicamente.

**Slide 26**

* Inserir\_dados\_JSON: Recebe como argumento informação obtida de um ficheiro **JSON** aberto antes da chamada da função e usa essa informação para inserir dados em uma tabela da **BD**, referente ao ficheiro **JSON** em questão.
* Carregar\_JSON: Abre um ficheiro **JSON** dado o seu ***path*** e retorna uma estrutura de dicionários ***python*** a ser usada em **"inserir\_dados\_json"**.

**Slide 27**

* Inserir\_dados\_relacionais: Função executa comandos **SQL** previamente definidos para inserir registos diretamente nas tabelas **Carro** e **Aluguer**.
* Main: Função que define as configurações da ligação **MySQL** e chama as funções mencionadas anteriormente.

**Slide 28**

**-Conclusão**

**Slide 29**

**-Fim**