Relatório do trabalho da disciplina de Integração de Sistemas de Informação

Trabalho Prático II

José Paulo Antunes - 11582

João Paiva - 14154

ESI

dezembro de 2020

|  |
| --- |
| Afirmo por minha honra que não recebi qualquer apoio não autorizado na realização deste trabalho prático. Afirmo igualmente que não copiei qualquer material de livro, artigo, documento web ou de qualquer outra fonte exceto onde a origem estiver expressamente citada. |

José Paulo Antunes - 11582

João Paiva - 14154

**Índice**

[Introdução 3](#_Toc61135418)

[Descrição do problema (primeira fase do relatório) 5](#_Toc61135419)

[Arquitetura a usar (primeira fase do relatório) 7](#_Toc61135420)

[Funcionalidades 9](#_Toc61135421)

[Estrutura/Funções 11](#_Toc61135422)

[Problemas 13](#_Toc61135423)

[Conclusão 15](#_Toc61135424)

**Lista de Tabelas**

[Tabela 1 ― Instruções de uso das funções implementadas 11](#_Toc61135414)

[Tabela 2 ― Contas já criadas (experimentar programa) 11](#_Toc61135415)

[Tabela 3 ― Classes utilizadas 12](#_Toc61135416)

# Introdução

Este relatório de projeto é feito no contexto do segundo trabalho prático da cadeira de Integração de Sistemas de Informação do segundo ano da Licenciatura em Engenharia de Sistemas Informáticos, e consiste na criação de uma aplicação de consulta e inscrição em eventos de cariz cultural, e na gestão de todo o sistema através dessa mesma aplicação.

Neste relatório pretendemos fazer uma breve descrição das funcionalidades presentes no projeto e as funcionalidades e metodologia do mesmo. Além disso, incluiremos algumas instruções de uso para facilitar o uso da aplicação, bem como uma explicação detalhada de cada funcionalidade implementada.

A aplicação está toda feita em C#, e a base de dados que escolhemos usar é do tipo Microsoft Access Database.

# Descrição do problema (primeira fase do relatório)

Para este trabalho prático o nosso grupo decidiu fazer uma espécie de “cartaz cultural”, com vários eventos e novidades, que serão implementados na página através dos métodos que estamos a aprender nas aulas.

Pretendemos também integrar outras funcionalidades, como por exemplo, uma previsão das condições meteorológicas para a data de cada evento, e outros serviços/informações que nos pareçam pertinentes à medida que fomos desenvolvendo em mais detalhe a aplicação.

# Arquitetura a usar (primeira fase do relatório)

Para este trabalho pretendemos usar a arquitetura SOA, Arquitetura Orientada a Serviços.

Vamos ter uma base de dados onde serão armazenados os diferentes dados, e o sistema funcionará através de um serviço de Web Central, que servirá para gerir não só os eventos como as pessoas que neles se inscreverem, e que vai conectar estes 2 serviços (eventos e clientes) para assegurar que não há sobrelotação ou outros problemas do género. As pessoas que estiverem a ver a lista de eventos não terão acesso diretamente à base dados que os contém, sendo que só poderão ver a lista através de um pedido efetuado pelo serviço central, para garantir a segurança de todos os dados.

# Funcionalidades

Vamos listar primeiro as funcionalidades implementadas e só depois demonstrar como estão implementadas, por isso todas as funções/métodos e outras soluções usadas para elaborarmos estas funcionalidades estarão detalhadas no capítulo seguinte.

A aplicação tem um sistema básico de login e registo, que está diretamente ligado à base de dados do projeto, mais concretamente à tabela “*Users*”, que contém não só a informação de login (*user* e *pass*) mas também informações pessoais de cada utilizador, que nem são necessárias de extrair.

Consoante o tipo de utilizador (tipo normal ou tipo administrador), o portal de Login reencaminha o utilizador para o passo seguinte.

Os utilizadores do tipo administrador têm acesso ao painel de administrador, que tem uma lista de todos os eventos atuais (e diz quais estão indisponíveis, ou cheios) e um botão junto a esta para eliminar o evento escolhido, uma lista de todos os utilizadores existentes e um botão junto a esta para eliminar o utilizador escolhido, e também a possibilidade de criar um novo evento.

Os utilizadores normais, por sua vez, são redirecionados para uma janela da aplicação que contém uma lista de todos os eventos (novamente, indica quais estão indisponíveis ou cheios).

Ao escolher um evento e confirmar, vão para a última janela da aplicação, que tem toda a informação sobre o estado da reserva desse utilizador nesse evento escolhido (se tem ou não reserva efetuada, se é possível efetuar, detalhes da previsão meteorológica [] quando é possível).

# Estrutura/Funções

Todos estas funcionalidades estão implementadas em várias soluções, as que lidam com informações mais sensíveis, por exemplo passwords, estão separadas por 2 níveis de conexão (aplicação comunica com *WS\_Central* que comunica com *WS\_Pessoa*), enquanto que as de uso mais imediato estão implementadas na própria solução da aplicação (por exemplo, as funções de converter *strings* de Data de um formato para outro).

A maior parte das funções são chamadas de forma automática sem ter que ser o utilizador a fornecer dados, mas para as que requerem *input*, elaboramos uma pequena tabela informativa de instruções de uso/restrições.

Algumas das funções implementadas são, por exemplo, Efetuar Reserva, Remover Reserva, Eliminar Pessoa, Listar Todos os Users, Listar Todos os Eventos, Listar Previsão IPMA (funciona para capitais de distrito, e até 4 dias depois do dia atual), Formatar Data, etc…

Tabela 1 ― Instruções de uso das funções implementadas

|  |  |  |  |
| --- | --- | --- | --- |
| Função | Input | Formato | Exemplo |
| CriarEvento | Data | String (dd/MM/yyyy HH:mm:ssh) | “31/01/2021 14:30:00h” |
| CriarEvento | Lotação M. | Int | 100 |

Tabela 2 ― Contas já criadas (experimentar programa)

|  |  |  |
| --- | --- | --- |
| Username | Pass | Tipo |
| admin | admin | administrador |
| user1, user2, user3, user4 | 1234 | normal |

Tabela 3 ― Classes utilizadas

|  |  |  |
| --- | --- | --- |
| Nome da Classe | Função | Atributos |
| PessoaS | Armazenar Pessoa | Int id – Número automático de identificação (único)  String username – Nome da conta (único)  String password – Password da conta  String nome – Nome do utilizador  Int idade – Idade do utilizador  String localidade – Localidade do utilizador  Int tipo – Tipo de conta  Int bi – Número B.I. (único) |
| EventoS | Armazenar Evento | Int id – Número automático de identificação (único)  String nome – Nome do evento (único)  String local – Local do evento  Int lotacao – Lotação atual do evento  Int capacidade – Capacidade máxima do evento |
| EventoS2 | Igual a EventoS mas em WS\_Pessoa | Mesmos que EventoS |
| Previsao | Armazenar Previsão metereológica importada do IPMA em .json | Int globalIdLocal – ID da cap. de distrito  Int dataUpdate – Data da atualização da previsão  Int precipitaProb – Probabilidade de precipitação  Int tMin – Temperatura mínima  Int tMax – Temperatura máxima  String predWindDir – Direção vento  Int classWindSpeed – Velocidade vento  Int idWeatherType – Letra-tipo das condições |
| Cidades | Armazenar a lista das capitais de distrito importada do IPMA em .json | Int globalIdLocal – ID da cap. de distrito |

# Problemas

Durante o processo de desenvolvimento do projeto deparamo-nos com vários problemas, mas conseguimos resolver todos através das aulas gravadas disponíveis no repositório da disciplina, das várias aulas de apoio ao trabalho lecionadas pelo professor e também recorremos à sebenta.

Um dos primeiros obstáculos que encontramos foi a necessidade de ter a classe EventoS presento no Web Service Pessoa, pois a função que verifica se um determinado utilizador tem uma reserva num determinado evento recebe como argumento um EventoS e uma PessoaS, portanto tivemos que criar um EventoS2 que é funcionalmente igual ao EventoS mas que está definido no outro WebService.

Também tivemos dificuldades que nos surpreenderam quanto à implementação do serviço exterior, pois a grande maioria dos exemplos que encontramos recorriam a Tasks e esperavam pelo resultado, o que não fazia sentido no contexto do nosso projeto, pelo que acabamos por fazer nós a nossa própria função com métodos assíncronos para ser independente do funcionamento do resto da aplicação.

Fomos obrigados a reformular várias vezes a maneira como estávamos a preencher as comboBoxes e a richTextBox do evento, pois queríamos armazenar a class completa mas mostrar na lista apenas um campo, por exemplo, o campo .Nome. Acabamos por guardar mesmo só o campo específico e usar funções auxiliares que vão buscar o evento/pessoa através do campo dado.

# Conclusão

Este trabalho prático deu-nos a oportunidade de aplicar os conceitos adquiridos nas aulas de ISI e de trabalhar num ambiente de cooperação para atingir os nossos objetivos.

De salientar que para completar este projeto, foram extremamente úteis tanto as aulas em que o professor explicou como se fazia cada parte do projeto como todos os materiais disponibilizados no e-learning, dado que continham imensas aplicações práticas dos conceitos que acabamos por utilizar na elaboração deste trabalho.

Na nossa opinião, a realização deste trabalho é extremamente importante para o nosso desenvolvimento enquanto engenheiros de sistemas informáticos, pois obriga-nos a familiarizar-nos com técnicas/conceitos que são vitais na nossa área, especialmente nos dias que correm.