Projeto

CampusPark – Smart Campus Parking Platform

Relatório que fundamenta as funcionalidades implementadas no projeto

|  |  |
| --- | --- |
| **Nome:** | Miguel |
| **Número:** |  |
| **Turno prático:** | PL3 |
| **Docente do turno prático:** | Marisa Maximiano |
|  |  |
| **Nome:** | Rúben Lopes |
| **Número:** |  |
| **Turno prático:** | PL3 |
| **Docente do turno prático:** | Marisa Maximiano |
|  |  |
| **Nome:** | Tiago Batista |
| **Número:** | 2161353 |
| **Turno prático:** | PL3 |
| **Docente do turno prático:** | Marisa Maximiano |
|  |  |
| **Nome:** | Tiago |
| **Número:** |  |
| **Turno prático:** | PL3 |
| **Docente do turno prático:** | Marisa Maximiano |

Índices

1 Introdução 3

2 Providers 3

3 Data Acquisition and Content Enricher (ParkDACE)   3

4 Parking Spot Transmission Unit (ParkTU) 3

5 Parking Spot Storage App (ParkSS) 3

6 Smart Park RESTful Web Service 3

7 Admin Dashboard (ParkDashboard) 3

8 Requisitos não implementados 3

9 Contribuição de cada elemento do grupo 4

10 Conclusão 4

*Anexo A* – Listagem dos Ficheiro XML utilizados 5

*Anexo B* – Outras informações 5

# Introdução

[Breve descrição do trabalho. Notas introdutórias.

Nas seções seguintes deve descrever todos os pormenores de implementação das várias aplicações/módulos.]

Para o projeto da cadeira de Integração de Sistemas foi pedido que criássemos um conjunto de serviços para fornecer dados de consumo para uma aplicação que deverá ajudar os alunos da ESTG a encontrar um lugar de estacionamento nos parques da escola. Este conjunto de serviços inclui:

**BOT-SpotSensors** (em SOAP) para imitar os sensores reais que iriam ser aplicados na vida real;

**Data Acquisition and Content Enricher App** ou **ParkDACE** que recebe informações limitadas e as completa com informação adicional e então redireciona a informação para a plataforma de comunicação;

**Parking Spot Storage App** (ParkSS) que de forma automática guarda a informação dos sensores e dos parques numa base de dados relacional assim como o histórico de alterações dos spots ao longo do dia;

**SmartPark** (RESTFul web service), serviço disponível para os utilizadores;

**Admin Dashboard** (ParkDashboard), usado pelo administrador para vigiar o estado dos lugares.

# Providers

[Descrição das funcionalidade e algoritmo adotado na implementação do serviço que simula o BOT que efetua o envio de dados fictícios sobre a ocupação de locais de estacionamento de um determinado parque.]

Para o serviço em SOAP que simula o BOT é indicado o número de lugares no código de um determinado parque para que seja enviada informação de todos os supostos sensores. É então gerado um spot de cada vez onde o estado é definido aleatoriamente assim como o estado da bateria, tendo o estado da bateria maior probabilidade de estar operacional. Para além do estado do spot e da bateria cada sensor envia também o id, nome e a hora/min/seg em que envia a informação. Assim é recebida uma lista de todos os spots de um parque.

# Data Acquisition and Content Enricher (ParkDACE)

[Nesta seção deve fazer uma explicação da abordagem adotada na implementação de todos as funcionalidades implementadas nesta aplicação; apresentar e explicar a representação final adotada para representação da mensagem referente ao local de estacionamento (parking spot) e clarificar como é que tecnologicamente é disponibilizada esta informação para o componente seguinte – *Parking Spot Transmission Unit*.]

Com o ParkDACE é recebida a informação de cada sensor e adicionada a restante informação adicional para que esta seja então exportada num formato XML de modo a dar reposta para o Parking Spot Transmission Unit.

Completar…

# Parking Spot Transmission Unit (ParkTU)

[Nesta seção deve fazer uma explicação da abordagem adotada na implementação deste componente de modo a clarificar como é que tecnologicamente é disponibilizado/encaminhadas as mensagens com informação referente à ocupação de um parque de estacionamento.]

Para o ParkTU optámos por juntar com o ParkDACE uma vez que toda a informação que o ParkTU iria necessitar já se encontra no ParkDACE. Assim dentro do ParkDACE “sai” toda a informação para um canal subscrito no [Mosquitto](https://mosquitto.org/) para que o ParkSS a possa receber.

Optámos também por escolher passar no canal as informações de referentes aos spots, para que possa depois ficar registado um histórico das mudanças de todos os spots.

# Parking Spot Storage App (ParkSS)

[Explicação da abordagem adotada na implementação de todos as funcionalidades existentes nesta aplicação, incluindo o esquema adotado para representar os dados na base de dados da aplicação e justificação do mesmo, bem como o mecanismo utilizado para otimizaram a recolha dos dados existente no **Parking Spot Transmission Unit**, etc. **Nota**: Não esquecer que têm que entregar a script SQL com o esquema da base de dados (ficheiro .sql)]

Para o ParkSS escolhemos que fique registado não só os parques e os spots mas também o histórico completo dos spots sempre que o spot sofra alguma alteração como a mudança de estado de ocupado para livre e vice-versa. Assim temos dados para as consultas funcionarem otimamente e com precisão.

# Smart Park RESTful Web Service

[Explicação de todos as funcionalidades implementadas ao nível da camada de serviço. Descrever a API do serviço, especificando as operações/recursos disponíveis.]

O SmartPark disponibiliza através de rotas o acesso aos dados para consulta. A API tem as rotas definidas de forma fácil de interpretar. A única confusão que pode existir é saber o controlador em que se encontra a rota que se quer usar mas para resolver esse problema disponibilizamos um [guia de rotas](Guia%20de%20Rotas.xlsx) com exemplos de uso e onde as encontrar.

# Admin Dashboard (ParkDashboard)

[Descrever as funcionalidades desenvolvidas na aplicação de monitorização do sistema para visualização do estado global do sistema por parte do administrador do sistema.]

O ParkDashboard optámos por um Windows Application From para fazer uso dos serviços do SmartPark. Para que as consultas possam ser facilmente feitas e que se obtenham respostas rápidas e precisas da forma mais intuitiva que nos foi possível implementar.

No Admin Dashboard encontram-se vários separadores que fazem pedidos ao servidor para obter os dados referentes à resposta pretendida. De modo a não sobrecarregar o servidor, após o primeiro pedido ao servidor, a resposta (sendo uma lista ou apenas um só produto(spot ou park) é armazenado localmente. Dessa forma sempre que precisarmos de apresentar novamente os dados referentes ao mesmo pedido já não se torna necessário fazer conexão com o servidor. Existe a possibilidade através de um botão (refresh) que torna a atualização dos dados possível. Faz com que a nossa aplicação tenha de fazer novamente pedidos à API, apresentando e armazenando então os possíveis novos dados presentes na nossa base de dados.

# Requisitos não implementados

[Lista de todos as funcionalidades não implementadas, ou implementadas apenas parcialmente para cada módulo].

# Contribuição de cada elemento do grupo

[Indicação clara das funcionalidades implementadas por cada membro do grupo em cada módulo].

# Conclusão

[Processo de desenvolvimento do trabalho, enfatizando eventuais dificuldades encontradas. Notas finais.]

Este trabalho foi interessante do ponto de vista de programador uma vez que pudemos tomar decisões relativas ao funcionamento interligado dos vários *providers.* Era necessário para o projeto que os *providers* conseguissem comunicar uns com os outros de uma forma quase que sequencial. Em que são recebidos os dados no ParkDACE, tratados com as informações adicionais e enviados para o ParkSS de forma a serem registados na base de dados. Uma vez registados, aplicações como uma aplicação *windows* ou uma página web seja no browser ou em mobile, conseguem aceder a esses dados fazendo pedidos ao SmartPark.

Tivemos algumas dificuldades mais na área da escolha de funcionamento, muitas vezes não sabendo bem que opções é que estavam disponíveis para tomar. Em geral pensamos que tenha sido um desenvolvimento bastante positivo com um resultado que estamos orgulhosos ainda que a linguagem não fosse um forte de nenhum dos membros do grupo o que provou um desafio interessante de ultrapassar.

#### *Anexo A* – Listagem dos Ficheiro XML utilizados

[Colocar o XSD e respetivo XML para o(s) diverso(s) ficheiro(s) de XML que foram criados no projeto.]

#### *Anexo B* – Outras informações

[*ConnectionStrings*, Logins e passwords necessárias para testar o trabalho (caso se justifique). Devem também indicar todos os ficheiros de configuração que é necessário alterar para executar o trabalho corretamente.

App Harbour