|  |
| --- |
| Code Race- HACKATON  Food Order Challenge |
|  |
|  |
|  |
|  |
|  |

|  |
| --- |
| Food Order Challenge |
|  |

# Informações Gerais

Geral

|  |  |
| --- | --- |
| **Nome do Projeto:** | Food Order Challenge |
| **Nome Documento:** | Functional and Technical Requirements |
| **Referência:** |  |

Lista de Distribuição

|  |  |  |  |
| --- | --- | --- | --- |
| **Nome** | **Função** | **Cópia** | **Informação** |
| GEC – Digital |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Tabela 1 – Lista de Distribuição

Histórico de Alterações

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Data** | **Versão#** | **Autor** | **Descrição** | **Aprovação** |
| 19/11/2019 | 1.0 | Pedro B Costa | Criada Versão Inicial | Mike Marques |
|  |  |  |  |  |
|  |  |  |  |  |

Tabela 2 – Controlo de versões

Documentos de Referência

|  |  |  |
| --- | --- | --- |
| **Documento** | **Descrição** | **Autor** |
|  |  |  |
|  |  |  |
|  |  |  |

Tabela 3 – Documentos de Referência

|  |
| --- |
|  |
| Índice |
|  |

[1. Informações Gerais 2](#_Toc26364491)

[2. Objetivo 4](#_Toc26364492)

[3. Definições e Abreviaturas 5](#_Toc26364493)

[4. Arquitectura da Solução 6](#_Toc26364494)

[4.1 Backend 6](#_Toc26364495)

[4.2 Frontend 6](#_Toc26364496)

[4.3 MobileApp 7](#_Toc26364497)

[5. Requisitos Funcionais 8](#_Toc26364498)

[5.1 RF01 - Obrigatório 8](#_Toc26364499)

[5.2 RF02 - Obrigatório 8](#_Toc26364500)

[5.3 RF03 - Obrigatório 8](#_Toc26364501)

[5.4 RF04 - Obrigatório 8](#_Toc26364502)

[5.5 RF05 - Obrigatório 8](#_Toc26364503)

[5.6 RF06 - Obrigatório 8](#_Toc26364504)

[5.7 RF07 - Obrigatório 8](#_Toc26364505)

[5.8 RF08 - Obrigatório 8](#_Toc26364506)

[5.9 RF09 - Obrigatório 8](#_Toc26364507)

[5.10 RF10 - Obrigatório 9](#_Toc26364508)

[5.11 RF11 - Obrigatório 9](#_Toc26364509)

[5.12 RF12 – Obrigatório 9](#_Toc26364510)

[5.13 RF13 – Obrigatório 9](#_Toc26364511)

[5.14 RF14 - Obrigatório 9](#_Toc26364512)

[5.15 RF15 - Obrigatório 9](#_Toc26364513)

[5.16 RF16 - Opcional 9](#_Toc26364514)

[5.17 RF17- Opcional 9](#_Toc26364515)

[5.18 RF18- Opcional 9](#_Toc26364516)

[6. Requisitos Técnicos 10](#_Toc26364517)

[6.1 RT01 10](#_Toc26364518)

[6.2 RT02 10](#_Toc26364519)

[6.3 RT03 10](#_Toc26364520)

[6.4 RT04 10](#_Toc26364521)

[6.5 RT05 10](#_Toc26364522)

[6.6 RT06 10](#_Toc26364523)

[6.7 RT07 10](#_Toc26364524)

[6.8 RT08 10](#_Toc26364525)

[7. Factores de Avaliação 11](#_Toc26364526)

# Objetivo

O objectivo deste documento é estruturar os requisitos do Hackaton a realizar pelo departamento de Digital do GEC - Fundão, em data a confirmar.

Pretende-se detalhar os requisitos da solução a propor pelos concorrentes, especificando os requisitos técnicos e funcionais e detalhar os componentes a desenvolver.

O documento tem como função introduzir os factores diferenciadores das soluções que serão propostas.

Pretende-se que os candidatos sejam capazes de entregar uma solução end-to-end de uma WebApp infotainment, construída sobre uma framework Javascript da General Motors, que irá comunicar com uma aplicação Java, baseada em Spring Boot, que servirá de Backend e uma aplicação Mobile, também com comunicação ao Backend.

A arquitectura é a propor é assim composta por três componentes: **WebApp**, construída em JavaScript, **MobileApp**, construída sobre Android ou iOS e **Backend**, construído sobre Spring Boot, doravante denominados em conjunto como **Solução**.

A **Solução** tem como objectivo representar a encomenda de comida de restaurantes a partir de um carro. Cada carro terá uma distância aos restaurantes medida em KM’s.

Na **WebApp** será feita a encomenda, ou cancelamento, da comida do restaurante e a **MobileApp** terá como responsabilidade, entre outras, possibilitar o reporting das encomendas.

O Backend terá como responsabilidade gerir as encomendas, calcular a distância em KMs a partir das coordenadas GPS, concluir os pedidos e gerir toda a informação requerida pela **Solução**.

Exclui-se da **Solução** qualquer cálculo de distância real ou caminho, seja mais curto ou mais rápido, para o ponto onde se encontra o restaurante.

O **Backend** deverá servir para armazenar os restaurantes, os menus, os pedidos e gerir a autenticação no sistema.

O **Frontend** será a interface para os clientes.

A **MobileApp** será a interface para os restaurantes.

# Definições e Abreviaturas

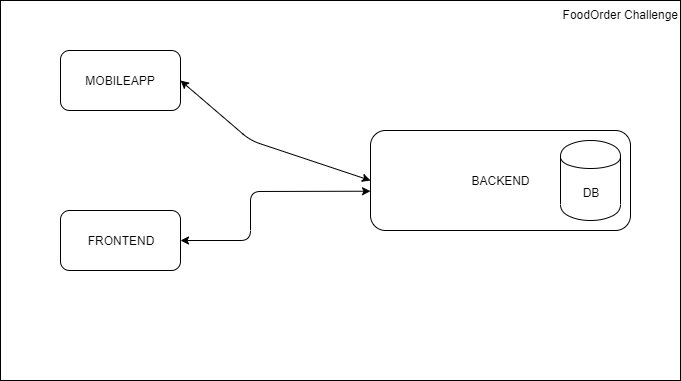
|  |  |
| --- | --- |
| **Definições/ Abreviaturas** | **Descrição** |
| Framework GM | Framework da General Motors que emula o comportamento de uma aplicação a correr dentro de uma viatura |
| GPS | Global Positioning System, descreve coordenadas – duas coordenadas podem ser medidas em distância em metros e/ou quilómetros |
| Android/iOS | Sistemas operativos onde correm as MobileApp |
| Spring Boot | Framework onde deve correr o Backend, tem um servidor aplicacional integrado |

Tabela 4 – Definições e Abreviaturas

# Arquitectura da Solução

O desafio passa por construir um **Backend**, que necessariamente terá de uma Base de dados, a comunicar com a **MobileApp** e o **Frontend**, baseado na framework da GM.

A **MobileApp** e o **Frontend** não devem comunicar entre si, apenas comunicam com o **Backend**.



## Backend

O **Backend** terá de providenciar acesso dos carros aos restaurantes, pelo que terá de conseguir traduzir duas coordenadas GPS em distância em metros e/ou quilómetros.

Será construído sobre o padrão MVC e utilizando Spring Boot. Um esqueleto desta solução será fornecida aos participantes.

Deverá correr no porto default 8080 e permitir a comunicação com os seus elementos por via de serviços REST, transportando JSON.

## Frontend

O **Frontend** será construído sobre Angular+ e usando a framework da GM, cuja instalação se providenciará.

## MobileApp

A **MobileApp** poderá correr sobre Android ou iOS e ser demonstrado num tablet, telemóvel ou ambos, a decidir pela equipa.

# Requisitos Funcionais

Listam-se de seguida os requisitos funcionais que devem guiar a solução.

## RF01 - Obrigatório

O sistema deverá permitir a gestão de carros que podem aceder aos restaurantes, de forma a autenticá-los sobre o sistema.

## RF02 - Obrigatório

O sistema terá de possuir uma lista de restaurantes, cada um determinado pelas posições GPS, nome do restaurante, morada do restaurante.

## RF03 - Obrigatório

O sistema deverá ter uma lista de produtos por restaurante, cada um descrito por um nome do item de menu, um preço e uma imagem descritiva – esta imagem pode ser apenas sugestiva.

## RF04 - Obrigatório

O sistema terá de permitir criar, atualizar, apagar e ler as ordens criadas aos restaurantes – todos as operações CRUD sobre os as ordens de menus devem ser desenvolvidas.

## RF05 - Obrigatório

O sistema deverá construir um histórico de ordens de menu para possibilitar reporting.

## RF06 - Obrigatório

A MobileApp e o Frontend devem autenticar-se perante o Backend, ainda que seja um mecanismo básico, com pouca segurança, ou até simulado.

## RF07 - Obrigatório

O Frontend deverá poder mostrar a lista de restaurantes, com uma distância em metros e/ou quilómetros.

## RF08 - Obrigatório

O Frontend deverá permitir a escolha de itens de menu, com quantificadores para cada, à semelhança de um carrinho de compras.

## RF09 - Obrigatório

O Frontend deverá permitir a confirmação da compra, antes de finalizar a comunicação com o restaurante.

## RF10 - Obrigatório

O Backend deverá permitir saber o estado da encomenda, o tempo de entrega/cozedura dos itens de menu comprados poderá ser imediato ou com um tempo pré-definido para cada.

## RF11 - Obrigatório

As encomendas devem ser unívocas, representadas por um identificar único.

## RF12 – Obrigatório

O Frontend deverá permitir a escolha de itens de menu, com quantificadores para cada, à semelhança de um carrinho de compras.

## RF13 – Obrigatório

O Frontend deverá permitir a escolha de itens de menu, com quantificadores para cada, à semelhança de um carrinho de compras.

## RF14 - Obrigatório

O Frontend deverá permitir cancelar uma encomenda.

## RF15 - Obrigatório

A MobileApp deve permitir alterar o estado de uma encomenda, pelo restaurante.

## RF16 - Opcional

O sistema pode permitir o uso de pontos por cliente.

## RF17- Opcional

O sistema poderá ser alvo de reporting (pedido mais efectuado, pontos por cliente, restaurante mais lucrativo, etc).

## RF18- Opcional

As interfaces de utilizador devem ser elegantes e responsivas.

# Requisitos Técnicos

Listam-se de seguida os requisitos técnicos que devem guiar a solução.

## RT01

A comunicação do sistema deve usar serviços REST, transportando JSON.

## RT02

O Backend deverá seguir o padrão MVC.

O Frontend e a MobileApp deverão ser modulares.

## RT03

O uso de padrões de desenho deve ser privilegiado, em vez de código esparguete.

## RT04

O código de todos os sistemas deve estar testado automaticamente.

## RT05

O código do sistema deverá seguir as melhores práticas de desenvolvimento (code quality and analysis).

## RT06

O sistema deve estar documentado, também com comentários ao código.

## RT07

O sistema deve possuir código com soluções elegantes, IDENTADAS e seguindo as naming conventions.

## RT08

O sistema deve ter persistência baseado num Sistema de Gestão de Base de Dados, onde devem ser armazenados todos os dados e operações.

# Factores de Avaliação

A solução será avaliada, em termos técnicos, com a média ponderada entre requisitos funcionais e técnicos implementados, numa pontuação de 0 a 5.

|  |  |  |
| --- | --- | --- |
| Requisitos | Factor | Descrição |
| RF01-RF15 | 50% | Requisitos obrigatórios |
| RF16-RF18 | 20% | Requisitos opcionais |
| RT01 | 10% | REST com JSON |
| RT02 | 10% | Backend segue MVC |
| RT08 | 10% | SGBD |
| RT04 | 5% | Unit Testing |
| RT03, RT05 | 5% | Code Quality (Simplicidade, Elegância, Design Patterns) |
| RT06 | 5% | Code documentation |
| RT07 | 5% | Identação, , Naming Conventions, Legibilidade |