**TECH CHALLANGE**

**JAVA**

**Description**

Build an API to:

* Allow Authentication;
* Query Products;
* Receive Orders;
* Cancel an Order;
* Get Order Status;
* Store data in a database of his/her choice;

**Technologies**

**Java** will be used as the back-end language.

**Maven** framework will be used to manage dependencies.

**Junit** framework will be used to handle unit tests.

**Spring** framework will be used to create the WebServices.

**MySQL** will be used as a database.

**Java.sql** will be used to manage connection and transactions between the Java and MySQL.

**Database sctructure**

Database name = “techChallengeDb”;

Database tables

* + Customer
    - To store users in which will be placing orders.
      * Id integer($int64)
      * email\* string
      * name\* string
      * address\* string
      * creation string($date-time)
      * password\* string
  + Order
    - To store orders (products as order items), user and status.
      * id integer($int64)
      * date string($date-time)
      * customerId integer($int64)
      * deliveryAddress\* string
      * contact\* string
      * storeId\* integer($int64)
      * orderItems\* [...]
      * total number($double)
      * status\* string
      * lastUpdate string($date-time)
  + OrderItem
    - id integer($int64)
    - orderId\* integer($int64)
    - productId\* integer($int64)
    - product Product{...}
    - price\* number($double)
    - quantity\* integer($int64)
    - total number($double)
  + Product
    - Food itens
      * productId integer($int64)
      * restaurantId integer($int64)
      * name string
      * description string
      * price number($double)
  + Restaurant (TBD)
    - restaurantId
    - name
    - description
    - Address
    - Phone
    - Latitude
    - Longitude
  + OrderStatus (TBD)
    - To map all possible order status.
      * Id
      * Name
      * Description
  + OrderReview (TBD)

**Backlog**

* Create a database using mysql
  + Create an customer table to store users to authenticate
  + Create an order table to store orders
  + Create a status table to map all possible order status
  + Create a restaurant table
  + Create a products table
  + Create a order review table (TBD)
* Create a java project
  + Create DTO objects to reflect database tables
    - AuthenticationDTO
    - CustomerDTO
    - OrderDTO
    - OrderItemDTO
    - OrderStatusDTO
    - RestaurantDTO
    - ProductDTO
    - OrderReviewDTO (TBD)
  + Create test layer
    - Create unit test to:
      * AutheticationDTO UserAuthentication(String email, String password)
      * List<RestaurantDTO> getRestaurants(float userLatitude, float userLongitude)
      * List<ProductDTO> getProductsFromRestaurant(int restaurantId)
      * OrderDTO createOrder(CustomerDTO user, List<ProductDTO> productList)
      * OrderStatusDTO cancelOrder(OrderDTO order)
      * OrderStatusDTO getOrderStatus(OrderDTO order)
  + Create Business layer
    - to authenticate users
    - to get restaurants from database
    - to get restaurant products from database
    - to create order
    - to cancel order
    - to get order status
  + Create web services
    - AuthenticationDTO UserAuthentication(String user name, String password)
    - List<RestaurantDTO> getRestaurants(float userLatitude, float userLongitude)
    - List<ProductDTO> getProductsFromRestaurant(int restaurantId)
    - OrderDTO createOrder(CustomerDTO user, List<ProductDTO> productList)
    - OrderStatusDTO cancelOrder(OrderDTO order)
    - OrderStatusDTO getOrderStatus(OrderDTO order)