

**Detail Design Document**

**For Motorsports Claims APIs**



**Version: 1.0**

**Author: Tanushree Deshpande**

**Date: 26-May-2020**

Version: 1.0

Date: 31/12/2019

Author: Suraj Desai

Table of Contents

[Revision Log 1](#_Toc44600529)

[Introduction 2](#_Toc44600530)

[System Overview 2](#_Toc44600531)

[API Architecture 2](#_Toc44600532)

[API Requirements 3](#_Toc44600533)

[Login API: 3](#_Toc44600534)

[Claim Details API: 5](#_Toc44600535)

[Technologies 8](#_Toc44600536)

[Environments 8](#_Toc44600537)

[Deployment Strategy 8](#_Toc44600538)

[Security 8](#_Toc44600539)

[Logging Methods 8](#_Toc44600540)

[Monitoring 8](#_Toc44600541)

# Revision Log

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Author** | **Version** | **Changes** |
| 26-May-2020 | Tanushree Deshpande | 1.0 | Initial Draft |

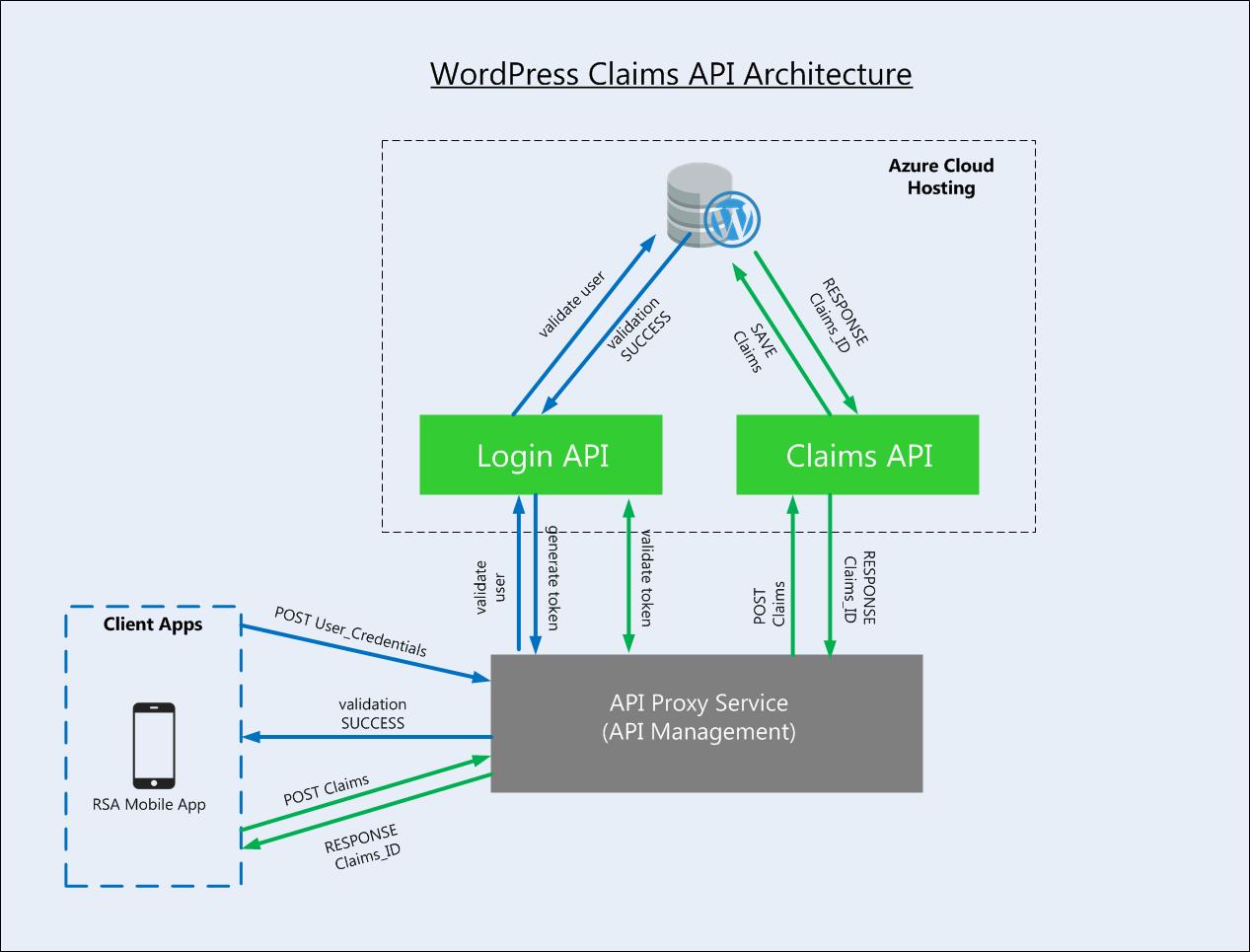
# Introduction

Motorsport Claims application will be a web responsive application that will allow technical advisors to raise claims and follow the status to closure before sending final claims settlement details to dealers. Motorsport Claims API is a web service that allows you to send Motorsport Claims data from your website or mobile app. The API gives access to all the supported functions and it uses ‘RESTful’ services over HTTP POST requests with parameters URL encoded into the request and its response is encoded in JSON.

# System Overview

The API allows you to build an application using the various API functions in any of the programming languages that allows you to generate Web requests. The API will be hosted in **Blogfac WordPress** instance. API URLs will be secured using existing 1-way SSL certificate.

## API Architecture



# 

# API Requirements

## Login API:

This API will be consumed when User wants to Login in application using Login Credentials. These Credentials are in encrypted format using ***RSA key cryptography*** then validated with WordPress Database, and ***if valid an Access Token will be generated for Authenticated User*** which then authorize user to proceed further.

#### **Sequence Diagram:**

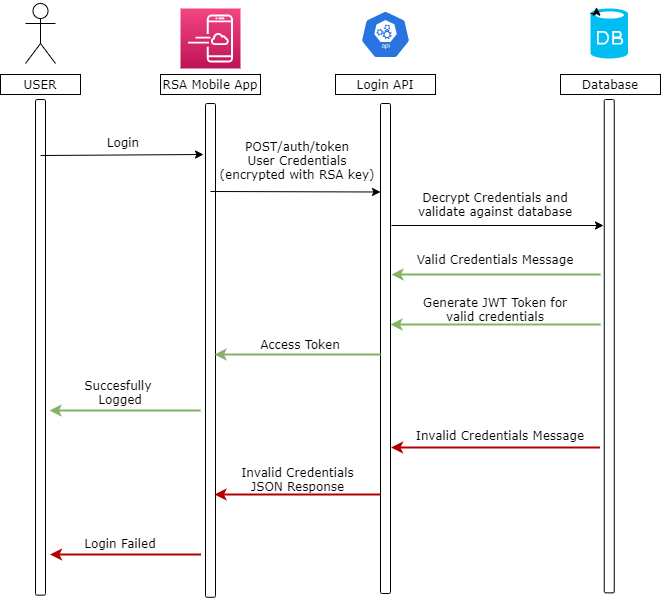


Fig. Sequence Diagram for Login API

#### **JWT Token Generation:**

#### 

POST/auth/token

LOGIN API

#### **Http Method:** POST

#### **Http Method:** http://blogfacapi.michelin.com/api/v1/auth/token

#### **Request Headers:**

Basic Authorization is to validate call is coming from authenticated API Gateway.

|  |  |
| --- | --- |
| **Parameters** | **Value** |
| Authorization | Basic b942e004-c4ec-48fd-9228-ae348501e87f |

#### **Request Body**:

#### {

#### "username":"MciMYLty2SVZrYO0fNQc4XdHRIT5ZvlWnxf1caIoT0LjZpUfFT216E7kjvyLb3FGXrDWJF5qr72uB4qrXaY95Y8/y8jnSlepKCuUTXAxWkBlaBuTjCHiJJeLNNCJFUt3/D1mh9ZTyvYWZX4cuZXyP+6Z3zSeZeSBPmdcm2ZSb9rZpWBocIGBZmwkSNMiMJwErDjAEYXdro6dAs7xKyAUvPM4T+Kjo5C8tIqXJkv7B+ld6FkZOFzQeL74WDKPjtOXvaiTmkqU926KR2uGo3LFFPU5O3ZBmO1fku7VpzFhACS+zT7gXqo/2WGHfGm7NdqI+wc32uE1jXF3BGBgzgL8lw==",

#### "password":"VKFl4psogBIOG6TKCpSDtG+VAvr8twqo+ayNV7SWyXzwLcWc1/WxsisK5PaLqToCSGGXuasYtcimEW0RRXER5pu6mxSDuaMzBuORxrpZ0afGjWuIXLkL3T8ZzuGvnkYvhR5XfTSisW87BKft0+N7N+/FEfiYqrOuBYCvEhz1Nw5wdG44tuxL+AhBECX7193LdVR9QZXyMafWXRvIZxTyqhpjb+YfOj/DP3eoPVSemyrQJnnPD7KcnLf1C7P/7G1tMg+093pePtYDr3OiD9iwoRQ7o6bysjIX0Bc7WQPA+Z6RUudzBiGzuZPeR2Vz/de6NT0w67KsnQFkIn1gUZPx8A=="

#### }

#### **Response Body**:

#### {

#### "access\_token": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.

#### "token\_type": "bearer",

#### "expires\_in": 3598

#### }

#### **Response Codes**:

|  |  |
| --- | --- |
| **Status Code** | **Description** |
| 200 | OK |
| 405 | Method Not Allowed |
| 404 | UserNotFound |
| 500 | Internal Server Error |

## Claim Details API:

* Claim Details API will be consumed by **CTE** user to **register Claim** details in database.
* When a claim is raised for the first time, irrespective of the user role, the claim status changes to ‘New Claim’.
* Every new claim created in the system will have a unique id that will follow the convention ‘MSPMMDDYYYYXXXXX’ where MSP = text identifier, MM = month of claim raised, DD = day of claim raised, YYYY = year of claim raised, XXXXX = incremental number.

#### **Sequence Diagram:**

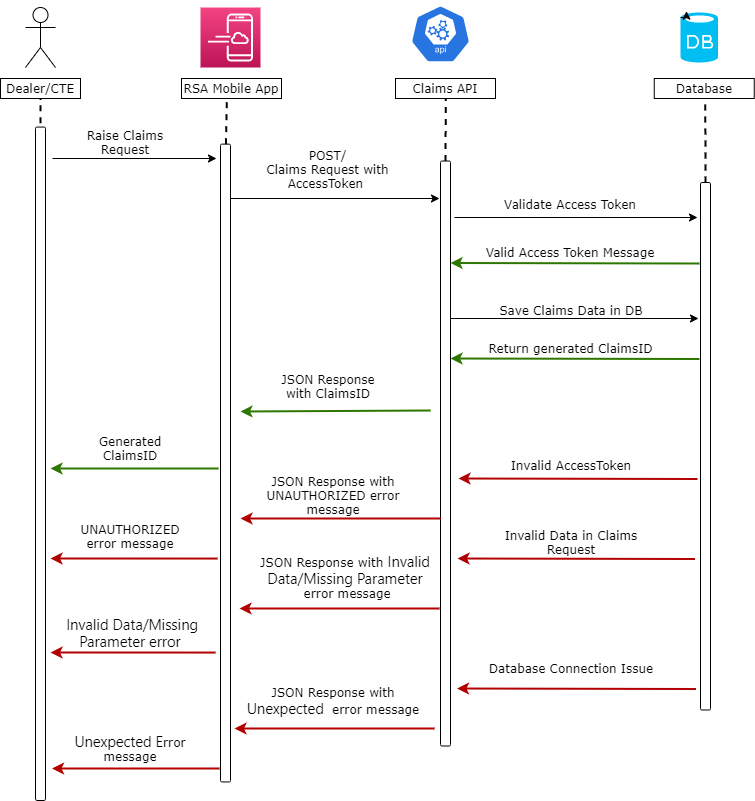
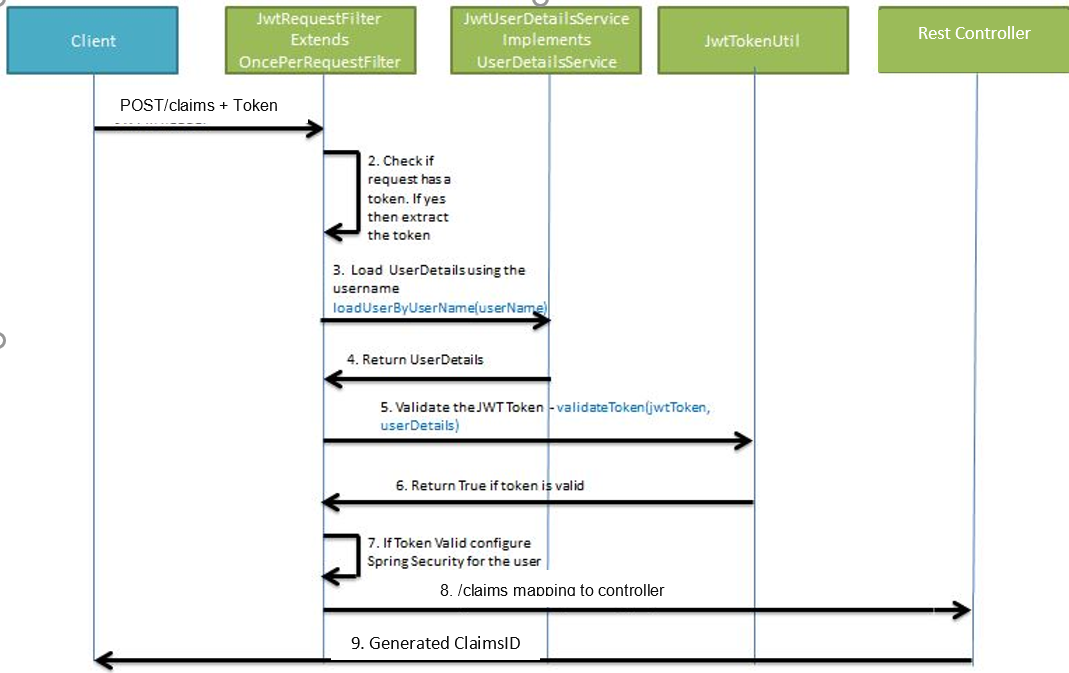


Fig. Claims Registration Sequence Diagram

#### **JWT Token Validation:**



9. Generated ClaimsID

Claims API

#### **Http Method:** POST

#### **Http Method:** http://blogfacapi.michelin.com/api/v1/claims

#### **Request Headers:**

Basic Authorization is added to validate request call coming from Gateway Layer, and Bearer Token is JWT token which is used to validate Authorized user.

|  |  |
| --- | --- |
| **Parameters** | **Value** |
| Authorization | **Basic** b942e004-c4ec-48fd-9228-ae348501e87f |
| Authorization | **Bearer** eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9. |

#### **Request Body:**

<claims.json>

#### **Response Body**:

#### {

#### "code": 201,

#### "message": "Created",

#### "result": {

"claimID": MSP10211164735985"

#### },

#### "error": {}

#### }

#### **Response Code**:

|  |  |
| --- | --- |
| **Status Code** | **Description** |
| 201 | Created |
| 405 | Method Not Allowed |
| 401 | Unauthorized |
| 403 | Forbidden |
| 500 | Internal Server Error |

Technologies

Restful APIs will be designed in Java language using **Spring Boot** framework.

Environments

QA, PreProd, Prod

## Deployment Strategy

*D*eployment will be done in **Azure** by using GIT lab CICD pipeline.

Security

1. *API level security is configured by JWT Token (****Spring Security****), it will be generated to authenticate the User and have expiry of certain period.*
2. *Credentials are passed using Asymmetric Encryption.*

*For more details Refer :* [*https://ssd.eff.org/en/module/deep-dive-end-end-encryption-how-do-public-key-encryption-systems-work*](https://ssd.eff.org/en/module/deep-dive-end-end-encryption-how-do-public-key-encryption-systems-work)

Logging Methods

Slf4j is utilized for logging the APIs at different level i.e. Debug, Info and Warning level.

Monitoring:

Azure monitoring dashboard will be utilized to manage the health status of APIs.