

**Technical documentation**

**WALLET SERVICE**

Version 1.0.0

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 14/10/2020 | 1.0.0 | Application deployed for usage by customers | Tru Blue |

**Table of Contents**

1. Introduction
   1. Purpose of this document
   2. Intended Audience
   3. Scope
   4. Definitions and acronyms
      1. Definitions
      2. Acronyms and abbreviations
2. General overview
   1. Technologies used
   2. General functioning
   3. Services
   4. Error handling
3. Technical requirement
   1. Client requirement
   2. Server requirement
4. Architecture
   1. High Level Architecture
5. **Introduction**
   1. **Purpose of this document**

The purpose of this document is to give technical information about Wallet Service design and implementation.

* 1. **Intended Audience**

The intended audiences are:

* Supervisors, to analyze the design and implementation of **Wallet**.
* Project Development team members.
* Future developers to refactor or use some ideas of project.
  1. **Scope**

This document will describe the design and the architecture of the project.

* 1. **Definition and acronyms**
     1. Acronyms and abbreviations

|  |  |
| --- | --- |
| **Acronym or abbreviation** | **Definitions** |
| **API** | Application Programming Interface |
| **Azure DevOps** | Version control technology |

1. **General Overview**
   1. **Technologies use**

Wallet is a web but mobile responsive application that was built with Angular on the Client side and C# (.NET Framework) for the Web APIs and MSSQL for the database. Azure DevOps was used for Version Control and the source codes resides on on-prem Bitbucket.

* 1. **General functioning**

The major function of the service is for users to access banking services using an e-wallet.

* 1. **Services**

Wallet **Live API** is hosted at Digital Live API Server at 10.20.3.39

Wallet **Live DB** Server is hosted at 10.20.7.212

**Table 1: Summary of infrastructure for Wallet**

|  |  |  |  |
| --- | --- | --- | --- |
| S/N | IP Address | Service | Status |
| 1 | 10.20.3.33 | Digital Live API Server | Live |
| 2 | 10.10.7.212 | EWallet DB Server | Test |

**Table 2: Summary of Services Architecture for Wallet**

|  |  |  |
| --- | --- | --- |
| S/N | URL API Endpoint | Services |
| 1 | https://dtptest.fidelitybank.ng/eWalletAPI/swagger/index.html | Wallet URL Endpoint |
| 2 | http://192.168.41.124/Coligo | Coligo |
| 3 | http://196.13.161.229:5004 | Validation API URL |
| 4 | https://fidelitypaygate.fidelitybank.ng/cipg/MerchantServices | Paygate Merchant Base URL |
| 5 | http://10.10.5.90/finpad | Finpad Base URL |
| 6 | <http://10.10.5.45/finpad> | Finpad Base URL 2 |
| 7 | https://10.10.2.35/BankApiV2/customerservices/v1/customers/bvn/GetSingle | Bank API URL for BVN Validation |
| 8 | https://onesignal.com/api/v1/notifications | One Signal API URL |

* 1. **Error handling**

Errors are handled majorly from the API level however all the exceptions are caught on the frontend and users are duly notified.

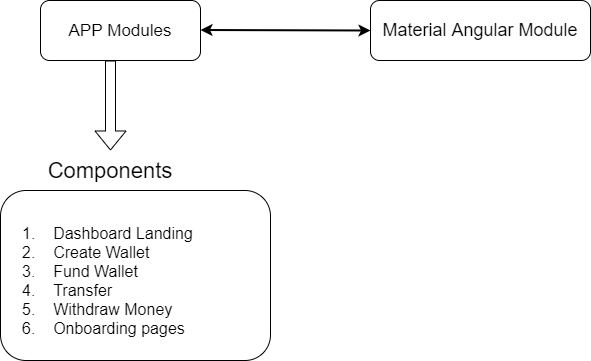
**Table 3: Error returned to users on the Wallet Service**

|  |  |  |
| --- | --- | --- |
| S/N | Error Code | Message to Customer |
| 1 | EE00 | Request successful |
| 2 | EE02 | BadRequest: Request not successful |
| 3 | EE99 | Error occurred while processing your request |
| 4 | EE401 | Invalid Authorization Header |
| 5 | EE401 | Invalid Username or Password |

1. **Technical Requirement**
   1. **Client-Side Architecture**

The application was built with the Angular framework on the client side. It is composed majorly of Modules, Components and Services.

**Figure 1: Client-side architecture - Wallet**



3.1.2 Services

**a. Account Creation Services**

This is used for calling endpoints relating account details and Name Enquiries

1. GetFidelityAccountDetails(GET)

<https://dtptest.fidelitybank.ng/eWalletAPI/api/Accounts/v1/getFidelityAccountDetails>

1. NameEnquiryForInterbankTransfer (POST)

<https://dtptest.fidelitybank.ng/eWalletAPI/api/Accounts/v1/nameEnquiryForInterbankTransfer>

1. GetAllBanksAndBankCodes (GET)

<https://dtptest.fidelitybank.ng/eWalletAPI/api/Accounts/v1/getAllBanksAndBankCodes>

**b. AutoTransactions**

i. FundWalletFromMultipleAccounts (POST)

<https://dtptest.fidelitybank.ng/eWalletAPI/api/AutoTransactions/v1/fundwalletfrommultipleaccounts>

ii. CashOutWalletToFidelityAccount (POST)

<https://dtptest.fidelitybank.ng/eWalletAPI/api/AutoTransactions/v1/cashoutwallettofidelityaccount>

**c.**  **CashOut**

i. CashOutWalletToFidelityAccount (POST)

[**https://dtptest.fidelitybank.ng/eWalletAPI/api/CashOut/v1/cashoutwallettofidelityaccount**](https://dtptest.fidelitybank.ng/eWalletAPI/api/CashOut/v1/cashoutwallettofidelityaccount)

ii. CashOutWalletToNonFidelityAccount (POST):

[**https://dtptest.fidelitybank.ng/eWalletAPI/api/AutoTransactions/v1/cashoutwallettononfidelityaccount**](https://dtptest.fidelitybank.ng/eWalletAPI/api/AutoTransactions/v1/cashoutwallettononfidelityaccount)

**d. FundWallet Service**

i. VerifyTransactionStatus (POST):

<https://dtptest.fidelitybank.ng/eWalletAPI/api/FundWallet/v1/verifytransactionstatus>

ii. FundWalletFromFidelityAccount (POST):

<https://dtptest.fidelitybank.ng/eWalletAPI/api/FundWallet/v1/fundwalletfromfidelityaccout>

**f. Onboarding Service**

i. Register (POST):

<https://dtptest.fidelitybank.ng/eWalletAPI/api/Onboarding/v1/register>

ii. VerifyPhoneNumber (POST):

<https://dtptest.fidelitybank.ng/eWalletAPI/api/Onboarding/v1/verifyphonenumber>

iii. OnboardChannel (POST):

<https://dtptest.fidelitybank.ng/eWalletAPI/api/Onboarding/v1/onboardchannel>

1. VerifyPhoneNumberAndPIN (POST):

<https://dtptest.fidelitybank.ng/eWalletAPI/api/Onboarding/v1/verifyphonenumberandpin>

1. UpdateWalletProfile (POST):

<https://dtptest.fidelitybank.ng/eWalletAPI/api/Onboarding/v1/updatewalletprofile>

1. UpdateTransactionPIN (POST):

<https://dtptest.fidelitybank.ng/eWalletAPI/api/Onboarding/v1/updatetransactionpin>

1. ResetPIN (POST):

<https://dtptest.fidelitybank.ng/eWalletAPI/api/Onboarding/v1/resetpin>

1. ConfirmResetPIN (POST):

<https://dtptest.fidelitybank.ng/eWalletAPI/api/Onboarding/v1/confirmresetpin>

**g. Wallet Service**

i. CheckBalance (POST):

<https://dtptest.fidelitybank.ng/eWalletAPI/api/Wallet/v1/checkbalance>

ii. FundWalletToWallet (POST):

<https://dtptest.fidelitybank.ng/eWalletAPI/api/Wallet/v1/fundwallettowallet>

iii. GetBeneficiary (POST):

<https://dtptest.fidelitybank.ng/eWalletAPI/api/Wallet/v1/getbeneficiary>

iv. SaveBeneficiary (POST):

<https://dtptest.fidelitybank.ng/eWalletAPI/api/Wallet/v1/savebeneficiary>

v. DeleteBeneficiary (POST):

<https://dtptest.fidelitybank.ng/eWalletAPI/api/Wallet/v1/deletebeneficiary>

vi. GetAllTransactions (POST):

<https://dtptest.fidelitybank.ng/eWalletAPI/api/Wallet/v1/getalltransactions>

vii. SearchTransactionsByDate (POST):

<https://dtptest.fidelitybank.ng/eWalletAPI/api/Wallet/v1/searchtransactionsbydate>

1. SearchTransactionsByNarration (POST):

<https://dtptest.fidelitybank.ng/eWalletAPI/api/Wallet/v1/searchtransactionsbynarration>

* 1. **Server requirement**

The following technologies are used for the Wallet:

* MSSQL

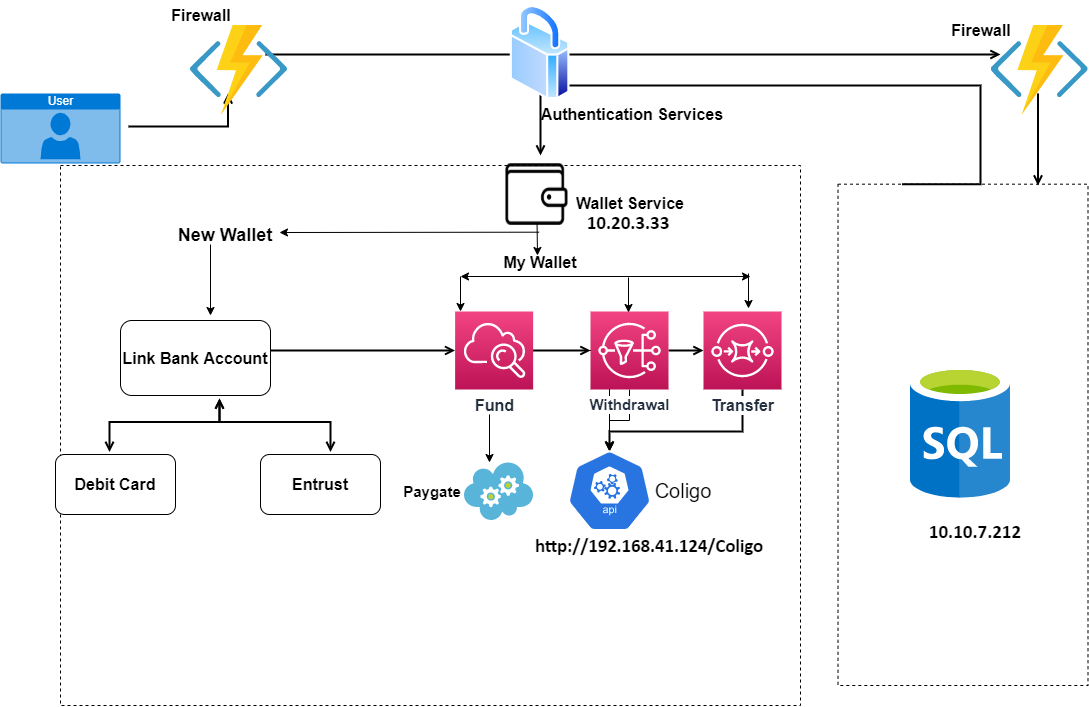
SQL was configured with SQL server 2017 layered on a Windows Server 2019.

* IIS 8 (Worker instances of 2)

**4. High Level Architecture**

**Figure 2: Service Architecture for Wallet**

The summary of the service architecture for Wallet can be seen on the figure below.



THANK YOU