PHP Department
PHPTL@iSoftBet.com

iSOFTBET
SIMPLY PLAY

iSoftBet Remote PHP Test

version 1.0

Table of contents

Confidential note

Please note that this document and the information within it are confidential and are the sole property of **iSoftBet**.
This document has been provided to you for the purpose of testing your **PHP CODING** abilities may not be disclosed to any third party or used for any other purpose without the express written permission of **iSoftBet**. By continuing with the test you agree with the above statements.

Introduction

Create a API that handles request / responses for a made up BANK. This API should be able to handle the following calls and reply in json format:

- adding of a customer:
  - **Request:** name, cnp
  - **Response:** customerId
- getting a transaction:
  - **Request:** customerId, transactionId
  - **Response:** transactionId, amount, date
- getting transaction by filters:
  - **Request:** customerId, amount, date, offset, limit
  - **Response:** an array of transactions
- adding a transaction:
  - **Request:** customerId, amount
  - **Response:** transactionId, customerId, amount, date
- updating a transaction:
  - **Request:** transactionId, amount
  - **Response:** transactionId, customerId, amount, date
- deleting a transaction:
  - **Request:** trasactionId
  - **Response:** success/fail

Request example for getting a transaction: APP_URL/transaction/{customerId}/{transactionId}
Response example for getting a transaction:

```
{
        "transactionId": 100,
        "amount": 205.67,
        "date": "20.03.2015"
}
```

The Test

- You will receive points for each of the requirements you solve.
- A fully functioning platform must be sent in an archive via email
- You may want to describe the project installation process, parameters that your routes expect, default login or whatever you find useful.

API(75 points)

1. **Framework Setup**
   a. Start and setup an app with Symfony 3.3.

2. **Database handling**
   a. You are free to create a database structure that will fulfill the API requirements.
   b. **Bonus:** Use ORM system.

3. **Security**
   a. Create a login system for GUI.
   b. Create an auth system for API.

4. **Logging**
   a. Log every request and response. Feel free to log anything else you think may also be important.

5. **Caching**
   a. Cache all API request / responses. API should use cached response if same request is sent within 1 hour.

GUI(15 points)

1. **Views**
   a. Create a page where we can see transactions. Page should be accessible only for logged in users.

2. **Pagination**
   a. Based on the view created at point 1., implement a pagination. 5/10/50 transactions / page should be displayed.

3. **Filtering**
   a. Based on the same view, we should be able to filter transactions by:
      i. amount
      ii. date
      iii. **Bonus:** use a datepicker
      iv. customerId

All the points above should use the API to get the list of transactions.

CRON(10 points)

1. Create a script that stores the sum of all transactions from previous day.
2. Attach a file where you add the command you use to setup the cron job to run every 2 days at 23:47.

Application diagram