Описание функциональных характеристик программного обеспечения

Наименование ПО: Интернет эквайринг и платежный процессинг "Логика"
Версия: v2.3.3

1. Назначение ПО

Программное обеспечение представляет собой платёжную платформу (Payment Processing System), предназначенную для приёма и обработки электронных платежей от клиентов мерчантов с использованием банковских карт.

2. Основные функции

- **Приём платежей** через банковские карты (Мир).
- **Выплаты** совершаются по запросу мерчанта через веб-кабинет или API (создание заявки на вывод средств).
- **API-интерфейс** для интеграции с внешними системами мерчантов.
- **Веб-кабинет с дашбордом** для управления операциями, просмотра статистики и выгрузки отчётов.
- **3-D Secure** для защиты транзакций.
- **Соответствие PCI DSS** требованиям.
- **Антифрод-модуль** для анализа транзакций и предотвращения мошеннической активности.

3. Технические характеристики

- Архитектура: микросервисная, сервисы на **Python** и **Ruby**, веб-сервер **Nginx**.
- Поддерживаемые ОС: **Debian, Ubuntu, CentOS**.
- Система виртуализации: **KVM** (допустимо использование VMware ESXi).
- Хранилище данных: PostgreSQL, Redis, MongoDB.
- Масштабирование: поддержка горизонтального и вертикального масштабирования сервисов.
- Логирование и мониторинг: централизованная система журналирования событий и ошибок.

4. Безопасность

- Поддержка **3-D Secure**.
- Собственный **антифрод-модуль** для выявления подозрительных операций.
- Шифрование данных (TLS 1.2/1.3).
- Соответствие требованиям **PCI DSS** по хранению и обработке данных карт.

5. Ограничения

- ПО требует подключения к банкам-эквайерам для работы.
- Выплаты возможны только при наличии подтверждённой заявки мерчанта.

Getting Started

This repo includes all needed services.

To clone all repos at once just run:

```
git clone git@gitlab.com:reactive-pay-distribution/rpay-engine.git
```

Docker install

Install docker on your local machine: https://www.docker.com/community-edition#/download

Then start building your services:

```
docker-compose build
```

or

```
./build.sh
```

Once it finished we need to setup some basic data.

Setup all service:

```
./init.sh
```

Setup business service:

- `docker-compose run --rm business bash`
- `rake db:create`
- `rake db:migrate`
- `rake db:seed`
- `exit`

If something went wrong with development data, you can try to clear the database using the following rake task:

- `rake db:clear`

It clears all data but does not drop the database, so you can start over `rake db:migrate` again.

Setup core service:

- `docker-compose run --rm core bash`
- `rake db:create`
- `rake db:migrate`
- `rake db:seed`
- `rake reactivepay:setup_db_data`
- `rake reactivepay:add_demo_business_account`
- `rake reactivepay:add_wallet_business_account`
- `exit`

And again here, if something went wrong with development data, try again with `rake db:clear`.

Setup settings service:

- `docker-compose run --rm settings bash`
- `rake db:create`
- `rake db:migrate`
- `rake db:seed`
- `exit`

Setup guard service:

For guard service there's no any additional setup.

Get up and running your services:

- `docker-compose up`

or

- `./up.sh`

Metabase

- setup your local metabase at http://localhost:9000/setup
- to connect to your business and core dbs enter the following credentials
- name: business (or core)
- host: postgres
- port: 5432
- database name: reactivepay_business_development (or reactivepay_core_development)
- database username: postgres
- database password: leave empty
- Additional JDBC connection string options: leave empty
- Use an SSH-tunnel for database connections: false
- This is a large database, so let me choose when Metabase syncs and scans: false

To add another db just go to http://localhost:9000/admin/databases/create

Final checks

Check that services are reachable for each other.
To do that, enter into each container and try to reach other service via `RestClient`

please note:

- you should get up and running your services first to be able to reach multiple services from separate console

core:

```
→ docker-compose run --rm core bash
Starting rpay-engine_postgres_1 ... done
Starting rpay-engine_redis_1    ... done
root@b3a4703d4a1c:/services/core# rails c
Loading development environment (Rails 4.2.8)
[1] pry(main)> RestClient.get(Rails.application.config.settings_host_url)
=> <RestClient::Response 200 "{\"version\":...">
[2] pry(main)> RestClient.get(Rails.application.config.business_host_url)
=> <RestClient::Response 200 "{\"version\":...">
[3] pry(main)> redis_info = Sidekiq.redis { |conn| conn.info }
=> {"redis_version"=>"4.0.8",
...
```

business:

```
→ docker-compose run --rm business bash
Starting rpay-engine_postgres_1 ... done
Starting rpay-engine_redis_1    ... done
root@b837767f5d4e:/services/business# rails c
Loading development environment (Rails 4.2.8)
[1] pry(main)> RestClient.get(Rails.application.config.settings_host_url)
=> <RestClient::Response 200 "{\"version\":...">
[2] pry(main)> RestClient.get(Rails.application.config.core_host_url)
=> <RestClient::Response 200 "{\"success\":...">
[3] pry(main)> RestClient.get(Rails.application.config.antifraud_service_url)
=> <RestClient::Response 200 "{\"version\":...">
[4] pry(main)> redis_info = Sidekiq.redis { |conn| conn.info }
=> {"redis_version"=>"4.0.8",
...
```

How to perform default gateway callback for dev needs

Send **POST** request to http://localhost:4000/callback/default with the following data:

```
{
        "token": "58576d820b58552cddc3b03e548c8017"
}
```

Where `58576d820b58552cddc3b03e548c8017` is your payment token (you can grab it from url http://localhost:4000/checkout_results/58576d820b58552cddc3b03e548c8017 via checkout process)

How to remove all docker's local artefacts

```
./clean_all.sh
```

Описание методов эксплуатации

1. Авторизация пользователя

Для доступа к функционалу приложения менеджер (или иной уполномоченный пользователь) проходит процедуру аутентификации на странице входа.

Менеджер имеет доступ к административному функционалу, включая просмотр транзакций, управление мерчантами, создание отчётов и экспорт данных.

2. Раздел «Мерчанты»

Раздел предназначен для управления учётными записями мерчантов (торговых партнёров), подключённых к платёжной системе. Здесь менеджеры могут просматривать, редактировать и добавлять новых мерчантов, а также получать доступ к их учётным данным и настройкам.

Функциональные возможности:

1. Поиск и фильтрация:

o В верхней части интерфейса расположен поисковый блок, позволяющий выполнять поиск по ID, номеру телефона или адресу электронной почты.

o Также доступен фильтр по статусу — реальный или тестовый мерчант.

2. Таблица мерчантов:

В таблице отображаются основные сведения по каждому мерчанту:

o Дата создания — дата регистрации мерчанта в системе.

o ID — уникальный идентификатор в базе данных.

o Наименование компании — юридическое или торговое имя партнёра.

o Email — контактный адрес для уведомлений и связи.

o Субаккаунты — количество дочерних аккаунтов (если используется многоуровневая структура).

o Приватный ключ мерчанта — используется для аутентификации запросов в API.

o Реальный / Тестовый статус — определяет тип учётной записи (True — боевой, False — тестовый).

o Действия — панель управления для редактирования, удаления и входа в профиль мерчанта.

3. Действия над записями:

o Редактирование (иконка карандаша) — открывает форму изменения данных мерчанта.

o Авторизация от имени мерчанта (иконка стрелки) — выполняет вход в систему от имени выбранного мерчанта для проверки его интерфейса и прав доступа.

o Удаление (иконка корзины, если доступна) — удаляет учётную запись после подтверждения.

4. Добавление нового мерчанта:

В правом верхнем углу расположена кнопка «Добавить нового мерчанта». При нажатии открывается форма регистрации нового партнёра, где указываются реквизиты, контактные данные, валюты, ключи API и прочие параметры подключения.

3. Раздел «Платежи / Выплаты»

Назначение:

Раздел предназначен для мониторинга, анализа и управления всеми финансовыми операциями, проходящими через систему — как входящими платежами от клиентов, так и исходящими выплатами мерчантам.

Функциональные возможности:

1. Поиск и фильтрация:

o В верхней панели доступен поиск по токену, email, наименованию компании, псевдониму шлюза или ID мерчанта.

o Поддерживается фильтрация по типу операции (например, PayinRequest, PayoutRequest) и статусу (Created, Approved, Declined, Error и т.д.).

o Возможность выбора количества отображаемых записей на странице (по умолчанию — 20).

o Дополнительно доступна фильтрация по дате создания через встроенный календарь.

2. Таблица операций:

В таблице отображается список всех финансовых транзакций с ключевыми параметрами:

o Дата создания — время регистрации операции в системе.

o Имя компании — мерчант, от имени которого был инициирован платёж.

o Сумма — величина транзакции.

o Валюта — код валюты (RUB, EUR, USD и т.п.).

o Токен — уникальный идентификатор платежа, используемый для API-запросов и отслеживания статуса.

o Псевдоним шлюза — имя подключенного платёжного шлюза, через который проведена операция.

- Тип — классификация операции (например, входящий платёж — PayinRequest, исходящий — PayoutRequest).

- Статус — текущее состояние операции (Created, Approved, Declined, Error и др.), отображается цветовым индикатором для удобства мониторинга.

- Действия — доступ к дополнительной информации о транзакции (кнопка INFO) и контекстное меню с расширенными функциями.

3. Информационная карточка операции:

При нажатии на кнопку INFO открывается детальная карточка транзакции, содержащая:

- параметры запроса и ответа шлюза,

- временные метки всех этапов обработки,

- результат проверки антифрод-модуля,

- данные о мерчанте и используемом шлюзе.