THE INITIAL DESCRIPTION OF THE OBJECT OF INTELLECTUAL PROPERTY
SPACEFEX – Payment and Transactions Source Code

```
<?php

namespace SORQPAY\Modules\Banking\Repositories;

use Illuminate\Contracts\Pagination\LengthAwarePaginator;
use Illuminate\Support\Collection;
use Illuminate\Support\Facades\Request;
use SORQPAY\Modules\AemiDashboard\Http\Requests\WalletsRequest;
use SORQPAY\Modules\Banking\Exception\WalletNotFoundException;
use SORQPAY\Modules\Banking\Models\Wallet;
use SORQPAY\Modules\Banking\Property\PaymentVendor;
use SORQPAY\Modules\Banking\Property\WalletType;
use SORQPAY\Modules\Users\Models\UserType\UserInterface;
use SORQPAY\System\Repositories\Filters\AccountNumberFilter;
use SORQPAY\System\Repositories\Filters\DateRangeFilter;
use SORQPAY\System\Repositories\Filters\ExectFilter;
use SORQPAY\System\Repositories\Filters\FilterCompanyName;
use SORQPAY\System\Repositories\Filters\LastTransactionDateFilter;
use SORQPAY\System\Repositories\Filters\LikeFilter;
use Spatie\QueryBuilder\AllowedFilter;
use Spatie\QueryBuilder\QueryBuilder;
use SORQPAY\System\Repositories\Filters\ExectCustomerIdFilter;
use SORQPAY\Modules\Users\Models\Profile\BusinessProfile;
use Jenssegers\Mongodb\Eloquent\Builder;

/**
 * Class WalletRepository
 * @package SORQPAY\Modules\Banking\Repositories
 */
class WalletRepository extends AbstractBankingRepository
{
/**
 * @return string
 */
public function model(): string
{
        return Wallet::class;
}

/**
 * @return AllowedFilter[]
 */
private function getDefaultFilters(): array
{
        return [
                AllowedFilter::custom('_id', new ExectFilter()),
                AllowedFilter::custom('business_profile_id', new ExectFilter()),
                AllowedFilter::custom('currency', new ExectFilter()),
                AllowedFilter::custom('vendor', new ExectFilter()),
                AllowedFilter::custom('wallet_type', new ExectFilter()),
                AllowedFilter::custom('customer_id', new ExectCustomerIdFilter()),
                AllowedFilter::custom('name', new ExectFilter()),
                AllowedFilter::custom('crypto_address', new LikeFilter()),
                AllowedFilter::custom('network', new ExectFilter()),
                AllowedFilter::custom('interest_class', new ExectFilter()),
```

```
                AllowedFilter::custom('status', new LikeFilter(), 'status'),
                AllowedFilter::custom('term_start', new ExectFilter()),
                AllowedFilter::custom('term_end', new ExectFilter()),
                AllowedFilter::custom('iban', new ExectFilter()),
                AllowedFilter::custom('account_number', new AccountNumberFilter()),
                AllowedFilter::custom('coupon', new ExectFilter()),
                AllowedFilter::custom('wallet_name', new LikeFilter(), 'name'),
                AllowedFilter::custom('lockdown_period', new DateRangeFilter()),
                AllowedFilter::custom('user_id', new ExectFilter()),
                AllowedFilter::custom('company_name', new FilterCompanyName()),
        ];
}

/**
 * @param string $id
 *
 * @return Wallet
 */
public function getById(string $id): Wallet
{
        $wallet = $this->find($id);
        if (null === $wallet) {
                throw WalletNotFoundException::byId($id);
        }

        return $wallet;
}

/**
 * @param UserInterface $user
 * @return Collection
 */
public function getByUser(UserInterface $user): Collection
{
        return $this->where(['business_profile_id' => $user->getBusinessProfileId()])->get();
}

/**
 * @param UserInterface $user
 * @param string $wallet_id
 * @return Collection
 */
public function getByUserAndWalletId(UserInterface $user, string $wallet_id)
{
        return $this->where(
                [
                        '_id' => $wallet_id,
                        'business_profile_id' => $user->getBusinessProfileId()
                ]
        )->first();
}

/**
 * @param UserInterface $user
 * @param string $iban
 * @return Collection
 */
public function getByUserAndIban(UserInterface $user, string $iban)
{
        return $this->where(
                [
                        'iban' => $iban,
                        'business_profile_id' => $user->getBusinessProfileId()
```

```
                    ]
            )->first();
}

/**
 * @param WalletsRequest $request
 * @return LengthAwarePaginator
 */
public function getClientWalletsPaginated(WalletsRequest $request): LengthAwarePaginator
{
            $queryBuilder = QueryBuilder::for($this->model());

            // if case has incorrect id or manually deleted businessProfile
            // we will have an issue (Models\Wallet throw BusinessProfileNotFoundException::byId).
            // we need to check that and skip unknown/deleted businessProfileId.
            $businessProfiles = BusinessProfile::query()->get();
            $queryBuilder->whereIn('business_profile_id', $businessProfiles->pluck('_id'));
/*
            if (!empty($conditions['account_number'])) {
```