مدیریت کیف پول

مدیریت کیف پول‌ها

(پس از گذشت سال‌ها از سوال آخر تکلیف قبل) کامیاب چند سالی‌ست که به عنوان مهندس نرم‌افزار (با Node.js) در یکی از شرکت‌های معروف در حال کار است. او مدتی‌ست که اندوخته‌های خود را وارد بازار ارزهای دیجیتال کرده و تا به حال درآمد خیلی خوبی داشته(!) اما به تازگی با مشکل جدیدی روبه‌رو شده. از آنجایی که کامیاب کیف پول‌های زیادی دارد، مدیریت آنها در یک جا و یک لحظه برای او کمی سخت شده و چون وقت خالی زیادی ندارد، از شما خواسته برای او APIای بنویسید که در مدیریت کیف پول به او کمک کند. در ازای کار شما، کامیاب مبلغ خوبی رمزارز به حسابتان ارسال می‌کند!

اطلاعات هر کیف پول در جدول زیر قابل مشاهده است:

Key	Type	Description	Example
`name`	`String`	Wallet name	`My Wallet 1`
`balance`	`Float`	Total wallet balance (in USD)	`521.24`
`coins`	`[Coin]`	List of coins in the wallet	`*Shown later*`
`last_updated`	`Time`	Latest update timestamp	`2020-12-05 23:51`

همچنین مشخصات موجودیت `Coin` در جدول زیر قابل مشاهده است:

Key	Type	Description	Example
`name`	`String`	Coin (cryptocurrency) name	`Bitcoin`
`symbol`	`String`	Abbreviation of the coin	`BTC`
`amount`	`Float`	Amount of that coin held in the waller	`2.1`
`rate`	`Float`	Exchange rate (to USD)	`25.12`

بخش اول - پیاده‌سازی چهار عملیات اصلی CRUD ویژه‌ی کیف پول‌ها:

در جدول زیر endpointهای چهار قسمت API را مشاهده می‌کنید و در ادامه برای هر کدام توضیحات بیشتری آورده شده است.

Endpoint	Description
`POST /wallets`	Create a new wallet
`GET /wallets`	Get all the wallets
`PUT /wallets/{wname}`	Edit an existing wallet by `wname`
`DELETE /wallets/{wname}`	Delete an existing wallet by `wname`

در این endpoint از API پس از دریافت نام کیف پول توسط سرور، اگر آن نام تکراری نباشد، یک کیف پول جدید ایجاد شده و نتیجه‌ی آن برمی‌گردد. در صورت وجود خطا نیز می‌بایست ارور مناسب برگردانده شود. **دقت کنید** که مقدار `last_updated` در جواب سرور همان زمان ایجاد کیف پول است.

نمونه‌ای از درخواست کاربر:

```
// POST /wallets
{
    "name": "Wallet3",
}
```

پاسخ سرور (در صورت نبود مشکل):

```
{
    "name": "Wallet3",
    "balance": 0.0,
    "coins": [],
    "last_updated": "2020-12-05 12:32",
    "code": 200,
    "message": "Wallet added successfully!"
}
```

۲. عملیات دریافت، خواندن یا Read:

در این endpoint از API نیازی به وارد کردن اطلاعات نیست و صرفاً با درخواست زدن، اطلاعات تمامی کیف پول‌های موجود برگردانده می‌شود.

درخواست به سرور:

```
// GET /wallets
```

پاسخ سرور پس از درخواست:

```
{
    "size": 5,
    "wallets": [
        {
            "name": "Wallet6",
            "balance": 24.10,
            "coins": [
                {
                    "name": "Bitcoin",
                    "symbol": "BTC",
                    "amount": 2.15,
                    "rate": 55.214
                },
                {...},
                {...}
            ],
            "last_updated": "2020-12-08 12:32"
        },
        {...},
        {...},
        {...},
        {...}
    ],
    "code": 200,
    "message": "All wallets received successfully!"
}
```

۳. عملیات ویرایش یا Update:

در این endpoint تنها می‌توان اسم یک کیف پول مورد نظر را عوض کرد که به راحتی با درخواست زیر قابل انجام است:

```
// PUT /wallets/{wname}
{
    "name": "Wallet3_new",
}
```

پاسخ سرور (در صورت نبود مشکل):

```
{
    "name": "Wallet3_new",
    "balance": 212.1,
    "coins": [
        {...},
        {...}
    ],
    "last_updated": "2020-12-05 12:32",
    "code": 200,
    "message": "Wallet name changed successfully!"
}
```

۴. عملیات حذف یا Delete:

در آخرین endpoint مربوط به کیف پول‌ها با استفاده از نام یک کیف پول می‌توان آن کیف پول و تمامی کوین‌های موجود در آن را حذف کرد (البته حذف کردن کیف پول در اینجا به معنی خروج از آن تلقی می‌شود).

```
// DELETE /wallets/{wname}
```

پاسخ سرور (در صورت نبود مشکل):

```
{
    "name": "Wallet3_new",
    "balance": 212.1,
    "coins": [
        {...},
```

```
        {...}
    ],
    "last_updated": "2020-12-05 12:32",
    "code": 200,
    "message": "Wallet deleted (logged out) successfully!"
}
```

بخش دوم - پیاده‌سازی چهار عملیات اصلی CRUD ویژه‌ی کوین‌ها:

در جدول زیر endpointهای چهار قسمت API بخش دوم را مشاهده می‌کنید. **دقت کنید** که تمامی عملیات CRUD مربوط به کوین‌ها حتماً باید مربوط به یک کیف پول خاص باشند.

Endpoint	Description
`POST /{wname}/coins`	Create a new coin for a wallet
`GET /{wname}`	Get a wallet's information
`PUT /{wname}/{symbol}`	Edit an existing coin by `symbol`
`DELETE /{wname}/{symbol}`	Delete an existing coin by `symbol`

۱. عملیات ایجاد یا Create:

در این endpoint از API پس از دریافت مشخصات کوین توسط سرور، اگر آن نام یا نماد تکراری نباشد، یک کوین جدید برای کیف پول مورد نظر ایجاد شده و نتیجه‌ی آن برمی‌گردد. در صورت وجود خطا نیز می‌بایست ارور مناسب برگردانده شود.

نمونه‌ای از درخواست کاربر:

```
// POST /{wname}/coins
{
    "name": "Go Coin",
    "symbol": "GOC",
    "amount": 2.1,
    "rate": 21.4
}
```

پاسخ سرور (در صورت نبود مشکل):

```
{
    "name": "Go Coin",
    "symbol": "GOC",
    "amount": 2.1,
    "rate": 21.4,
    "code": 200,
    "message": "Coin added successfully!"
}
```

۲. عملیات دریافت، خواندن یا Read:

در این endpoint از API نیازی به وارد کردن اطلاعات نیست و صرفاً با درخواست زدن، اطلاعات آن کیف پول به همراه تمامی کوین‌های آن برگردانده می‌شود.

درخواست به سرور:

```
// GET /{wname}
```

پاسخ سرور پس از درخواست:

```
{
    "name": "Wallet3_new",
    "balance": 212.1,
    "coins": [
        {
            "name": "Bitcoin",
            "symbol": "BTC",
            "amount": 2.15,
            "rate": 55,214.21
        },
        {...},
        {...}
    ],
    "last_updated": "2020-12-07 12:32",
    "code": 200,
    "message": "All coins received successfully!"
}
```

۳. عملیات ویرایش یا Update:

در این endpoint می‌توانید اطلاعات مربوط به یک کوین خاص از یک کیف پول را عوض کنید:

```
// PUT /{wname}/{symbol}
{
    "name": "Go Coin",
    "symbol": "GOC",
    "amount": 12.1,
    "rate": 21.4
}
```

پاسخ سرور (در صورت نبود مشکل):

```
{
    "name": "Go Coin",
    "symbol": "GOC",
    "amount": 12.1,
    "rate": 21.4,
    "code": 200,
    "message": "Coin updated successfully!"
}
```

۴. عملیات حذف یا Delete:

در آخرین endpoint مربوط به کوین‌ها با استفاده از نام یک کوین در یک کیف پول می‌توان آن کوین را حذف کرد (البته حذف کردن کوین در دنیای واقعی به معنی فروش کل آن کوین تلقی می‌شود). **دقت کنید** که یک راه برای بررسی درستی این عملیات این است که اطلاعات همان کیف پول را بخوانیم.

```
// DELETE /{wname}/{symbol}
```

پاسخ سرور (در صورت نبود مشکل):

```
{
    "name": "Go Coin",
    "symbol": "GOC",
    "amount": 12.1,
    "rate": 21.4,
```

```
    "code": 200,
    "message": "Coin deleted successfully!"
}
```

نکات

بسیار مهم: در صورتی که تغییراتی در `amount` کوین‌های موجود در یک کیف پول انجام شود (با اضافه کردن کوین یا تغییر ویژگی `amount` یک کوین خاص یا حذف یک کوین)، مقدار ویژگی `balance` آن کیف پول باید آپدیت شود. به عبارت دیگر مقدار `balance` یک کیف پول باید همیشه آپدیت باشد.

نحوه‌ی محاسبه‌ی `balance`: برای محاسبه‌ی `balance` یک کیف پول، صرفاً کافی‌ست که عملیات ضرب مقادیر `amount` و `rate` را برای همه‌ی کوین‌های درون کیف پول انجام داده و جمع تمامی این حاصل‌ضرب‌ها، مطلوب ماست.

- می‌توانید فیلد `name` را به هر صورتی که می‌خواهید نام‌گذاری کنید. مثال‌ها صرفاً برای درک بهتر آورده شده‌اند و نیازی به داشتن کارکترهای خاص یا فاصله در این فیلد نیست.
- آپدیت شدن کوین‌های درون هر کیف پول (اضافه کردن، ویرایش ویا حذف کوین)، مثل آپدیت کردن `name` کیف پول، مقدار `last_updated` درون کیف پول را آپدیت می‌کند.
- شما **می‌بایست** در صورت بروز ارورهای معروف **پیام و کد وضعیت** مناسب آن را ارسال کنید.

- تمامی فایل‌های مورد نیاز خود را در یک فایل زیپ به فرمت زیر ذخیره کرده و آپلود کنید. در صورت رعایت نکردن فرمت از نمره‌ی شما کسر خواهد شد. `HW5-[STUDENT ID]-[STUDENT NAME].zip`

امتیازی

در حالت عادی اطلاعاتی که شما نگهداری می‌کنید درون یک data structure ذخیره می‌شود و پس از اتمام برنامه از بین می‌رود (در صورتی که جایی آن را ذخیره نکنید). حال برای پایدار کردن داده‌تان، از یک دیتابیس دلخواه استفاده کنید. از دیتابیس‌های معروف می‌توان به `PostgreSQL` یا `MongoDB` اشاره کرد.