**Маршрутизация.**

Каждый метод в любом случае возвращает ResponseEntity (объект HTTP ответа), который содержит статус. Например, 200 (ok), 201 (created) в случае успеха, или 404 (not found), 500 (internal error) в случае неудачи. При необходимости также возвращает body (Json, строку или число).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Метод | Путь | Принимает | Возвращает | Пояснение |
| GET | /client/{id || login} | Принимает логин или id клиента (в url) | JSON клиента (без id) | Ищет клиента в БД |
| GET | /client/{id || login}/name | Принимает логин или id клиента (в url) | Строку имени клиента | Ищет имя клиента в БД |
| POST | /client | JSON клиента | Id (типа Long) | Добавляет клиента в БД |
| GET | /client/{id || login}/purchases | Принимает логин или id клиента (в url) | JSON список покупок клиента | Каждая покупка записана в JSON формате (без id) |
| PUT | /purchase/{ id || login } | Принимает логин или id клиента (в url) + JSON предмета для покупки | JSON покупки (без id) | Если предмет с таким названием уже есть в БД, возьмёт его (при чём по цене, указанной в БД), иначе просто создаст новый предмет в БД. |
| GET | /purchase/{ id || login }  ?item=”…”&  cost=90&  description=”…” | Принимает логин или id клиента (в url) + название предмета (обязательно) + описание и цену (опционально) | JSON покупки (без id). | Если предмет с таким названием уже есть в БД, возьмёт его (при чём по цене, указанной в БД). Иначе, если указана цена, создаст новый предмет. Описание будет взято из параметра или сгенерировано автоматически |

**Хранение данных.**

Для хранения данных приложение использует hibernate и подключается к внешней MySQL DB. Каждая сущность использует свою таблицу, также присутствуют таблицы для обеспечения OneToMany и ManyToMany связей. Все SQL запросы hibernate генерирует и выполняет автоматически. Внутри приложения определены следующие сущности:

**Клиент** содержит в себе следующие поля:

- Логин (уникальный, начинается с буквы, также может содержать числа)

- Имя  
 - Пароль  
 - Баланс (количество денег на счету, или например, количество бонусных баллов. Зависит от потребностей)

**Покупка** содержит в себе следующие поля:  
 - Дата (автоматически записывается с помощью внутренней функции)

- Список предметов  
Подразумевается, что одна покупка может состоять из множества предметов (покупка – чек, предмет – позиция в чеке). Предметы хранятся в отдельной таблице в БД и имеют свои поля для описания сущности.

- Стоимость. Суммируется из стоимости предметов в покупке.

**Предмет** содержит в себе следующие поля:

- Имя  
 - Цена

- Описание

Также каждая сущность имеет уникальный для таблицы числовой id, который генерирует БД. В проекте каждая сущность имеет дублёра - класс ресурса. Эти ресурсы используются для передачи и принятия данных с фронта, они гарантируют возможность сериализовать сущность, и то, что переданный приложению JSON будет правильно десериализован. Также благодаря им, о сущностях знает только класс сервиса, который использует БД. При этом контроллеры, обрабатывающие HTTP запросы, работают только с ресурсами. Подобный формат хранения данных позволит в будущем быстро отмасштабировать систему под задачи более крупного проекта.

**Тестирование.**

В проекте содержится 17 функций для автоматического unit и интеграционного тестирования. Эти тесты покрывают только основные функции приложения, и их всегда можно расширить или дописать новые по образцу. Они помогут следить за работоспособностью приложения при его дальнейшем масштабировании. Также они используют собственную БД (h2, вместо MySQL). Что позволяет тестировать приложение, не поднимая рабочую БД.