Разбор пет проекта

Структура репозитория состоит из 2 папок и 2 файлов:

* client – клиентская часть проекта
* server – серверная часть проекта
* Readme – в нем расположена информация об развертывании проекта
* API – документ описывающий методы бэкенда, при разработке нового серверного метода, он сначала описывается в файле API и только после этого пишется код.

Note: Ответственность за ведение документа API на бэкенд разработчике.

Тимлид имеет право не принимать пул-реквест с новыми метода, при отсутствии его в документе.

Бэкенд

Для запуска бэкенда необходимо всё содержимое папки сервер скопировать в соответствующую директорию ospanel, либо можно настроить ospanel.

Бэкенд всегда возвращает ответы в виде json файла.

Бэкенд спроектирован в парадигме MVC –, по мнению Трусова А.С., самая адекватная система придуманная человечеством.

Архитектура MVC (model view controller) -где model – бизнес-логика приложения, view – это представление и controller – это связующее звено между моделями и представлением. Модели не знают о представлениях, а представления не знают о моделях. Представления должны быть тупыми, то есть не содержать вообще никакой бизнес-логики, не знают о представлении и работают только с данными.

Структура бэкенда:

1. api
   1. index.php
   2. application
      1. модули
      2. answer.php
      3. application.php

index.php – это единственная точка входа в бэкенд, в нем мы определяем:

* Формат возвращаемого ответа
* Список поддерживаемых методов

Также в нем разрешены кросс доменные запросы.

В index.php есть два header:

* Первый отвечает за формат ответа в виде файла json
* Второй

Подключение файлов в php осуществляется множеством способом, мы будем использовать require\_once.

Функция result определяет какие методы поддерживает бэкенд и каким методом из application каждый входящий запрос будет обработан.

Каждый новый метод сначала объявляется в index.php

Index.php формирует ответ с помощью команды echo, функции json\_encode и метода response из класса answer.

Весь бэкенд реализован на классах кроме index.php

Answer.php

В answer.php – объявлен статический класс, отвечающий за формирование ответа, также в нем расписаны все возможные коды ошибок. Все коды ошибок мы придумываем сами.

Коды ошибок задаются в виде ассоциативного массива, где ключ – это код ошибки (какое-то число), а значение – это текст ошибки.

Статический метод response в answer проверяет следующее:

* Если есть переданные данные, то это не false, и это данные которые содержат ассоциативный массив с ключом error, в противном случае возвращается успешный ответ
* Если никакие данные не переданы, то возвращается неизвестная ошибка (999), самая плохая ошибка и предполагается, что она никогда не будет вызвана.

В php вызов статических методов или обращение к статическим свойствам осуществляется через двойное двоеточие (::).

Статические методы не имеют доступа к контексту класса, поэтому для вызова статического метода в самом классе используется конструкция self:: … .

По концепции MVC класс answer является представлением (фактические не только он, но и частично файл index.php).

В index.php в функцию result передается аргументом суперглобальная переменная $\_GET.

Application.php

Application.php – это класс, который выполняет функцию контроллера.

В конструкторе этого класса объявляются экземпляры всех моделей (в нашем проекте это user и chat), также в конструкторе класса создается экземпляр для работы с базой данных и этот экземпляр передается в каждую модель.

Note: Передавать модели друг в друга – запрещено!

В классе application, для каждого запроса создается свой отдельный метод.

Лучше называть методы одинаково.

Каждый метод, как обработчик запроса, проверяет и валидирует входящие параметры запроса и в случае если они хорошие вызывает соответствующий метод соответствующей модели.

Если модели для работы необходимы данные из другой модели, то эти данные мы получаем в application.

Разберем для примера метод sendMessage, этот метод отвечает за отправку сообщения в чат, сначала в этом методе мы проверяем наличие параметров запроса, а именно параметров token и message, в случае если их нету, возвращается ошибка, потом из модуля user мы получаем пользователя по токену. В случае, если такого пользователя нет, то возвращается другая ошибка. Если пользователь есть, то вызывается метод sendMessage, который передает идентификатор пользователя и сообщение, которые мы передаем в чат.

После, в модели chat в методе sendMessage сначала добавляется новое сообщение в базу данных, вторым методом обновляется хэш сообщения и возвращается true.

Note: модели тоже умеют возвращать ошибки.

Авторизация

Авторизация осуществляется с помощью метода login. В этот метод мы передаем параметры запроса: login, hash и параметр rnd, где login – это логин пользователя, параметр rnd – это случайное целое число, а hash – это исходная функция, которая шифрует исходное сообщение, в нашем случае md5(md5(логин + пароль) + rnd). Безопасность обеспечивается тем, что в базе данных хранится хэш от логина плюс пароля, хранить пароль в явном виде – это кринж.

Процесс авторизации: по логину из базы данных мы получает пароль, добавляя к нему параметр rnd, мы берем от него хэш и получаем контрольную суммы которая сравнивается в переданным хэшем клиента, и если они равны, то мы создаем токен, для этого пользователя, он записывается в базу данных и вместе с информацией о пользователе возвращается в клиент.

Логаут происходит так, что по мы берем пользователя из базы данных, и если этот пользователь нашелся, то мы очищаем токен.

Субд надо открывать с помощью специальной внешнего инструмента в ospanel phpMyAdmin

Сохранённые данные между запросами не сохраняются. Единственный способ сохранить данные между запросами – это сохранить их в бд.

User.php

$userLogin = ‘vasya’;

$userPass = ‘…’;

md5($userLogin.’111’);

А вместо бд закомментить обращение к бд или возвращать объект.

Дезешка: на бэкенде сделать новую модель mathematics (painOfDomashka), в неё перенести методы про произоводные, сплайновые графики. И клиент должен поддерживать 4 запроса: логин, логаут, касательная и построение сплайна. И реализовать логин без бд.