**CityGuide - API**

1.0

*22.10.2014*

*Content*

[General server notes](#h.jh0ff1fu27lz)

[1. General](#h.vk5lfaiwxowp)

[2. Entities structure](#h.m8jhq42k8858)

[2.1 Category](#h.87j362u6x96c)

[АТРИБУТЫ](#h.ibl644iejr9z)

[2.2 Place](#h.2hr8fapc8bcn)

[АТРИБУТЫ](#h.ct72292be9yz)

[2.3 Comment](#h.dnf70gxpxlix)

[2.4 Action](#h.ehawf22nlsf2)

[2.5 Banner](#h.e8dxoynrd9ub)

[3. Synchronization](#h.hi7oknehbicz)

[3.1 Should client synchronize content?](#h.j9di0m1zn32x)

[3.2 Get content](#h.o7jqoagc36m4)

[4. Authorization](#h.jen2a1rb48a3)

[4.1 User Signup](#h.tx2w1m1dyljq)

[5. Actions](#h.squobqgckhdc)

[5.1 Get action by id](#h.lz2z8oaxtivv)

[6. Banners](#h.vg7n0w8n4tfr)

[6.1 Get list of banners](#h.5zn90ikgkcn5)

[7. Comments](#h.lvtt34v7etv9)

[7.1 Send comment](#h.mg86nskcbmeo)

[8. Favorites](#h.aewis5c1d0ug)

[8.1 Add to favorite](#h.req8z16x5kh6)

[8.2 Delete from favorite](#h.mrtsqdoqv9bn)

[8.3 Get favorites](#h.gi12i5pwzrws)

[8.4 Sync favorites to server](#h.o8y6yee3jjep)

[9. Remote notifications & Settings](#h.9xb6gd453n5l)

[9.1 Register push token](#h.tnc1hy4f7igo)

[9.2 Unregister push token](#h.x98jjmvyxsav)

[9.3 Get settings for remote notifications](#h.ynnye3seh3e)

[9.4 Remote notification parameters](#h.96z2zusvx792)

# 

# **General server notes**

У каждой категории могут быть или список под категорий, или список заведений.

Заведение может принадлежать сразу нескольким категориям/подкатегориям

Нужен механизм администрирования пользовательских отзывов и рейтинга. Нужно разграничить права доступа:

1. Админ
2. Контент менеджер - права на добавление/редактирование/удаление контента
3. Модератор отзывов - добавление/редактирование/удаление отзывов

Предосавлять аналитику

1) Евент на каждый экран для анализа поведения пользователей (количество ежедневной аудитории (DAU), гендерный состав, пути пользователей в приложении, насколько часто используются функции приложения, шаринг в соц. сети, достижения целей (регистрация, клик по баннеру, заходы в течении месяца, частота возврата аудитории (retention rate) => это все реализуется в связке iOS client + Flurry

2) Push-аналитика (количество отправленных сообщений, количество открытий приложения после push сообщения, ) => это реализуется на стороне сервера

3) Отчет об ошибках, когда, с какой ошибкой приложение завершило работу и т.д. => это будет отслеживать сервис Crashlytics

4) количество пользователей в неделю/месяц, состав аудитории, возраст/пол, количество просмотров подкатегори/заведения в неделю/месяц, количество кликов по банеру, количество просмотров акции, количество реакции на пуш- уведомленя, куда чаще заходят, куда кликают, как часто использую => это все реализуется в связке iOS client + Flurry

Язык только русский.

Обязательно предусмотреть комментарий заказчика:

Это наши планы на дальнейшее развитие приложения:

После запуска, мы планировали доработать следующее:

1) Запустить сайт - аналог моб. приложения. (брать данные должен с той же базы).

2) Добавить еще несколько экранов Афиша и Объявления (с возможностью загрузки фотографий пользователя на сервер)

3) Добавить возможность выбора другого города. (Я так понимаю новый город - новая база)

4) Добавить стандартную авторизацию при помощи имени пользователя и пароля. (для объявлений)

Нужно это учесть при проектировании серверной части + конечно маштабируемость и возможность выгрузки базы.

Учесть возможности Админки People&Times – тк она была показана заказчику как пример функционала. Доступ: armavir-portal.ru/admin Логин и пароль admin

# **1. General**

**Base URL:**

Production SERVER:<URL>

Development SERVER:**<**URL**>**,   
 Development SERVER - admin panel:<URL>

login: <login>

password:<password>

As part of base URL should be version of API. Example:

<http://cityguide.com/api/1.0/>

**Common headers:**

type=<ios|android>

# 

# 

# 2. Entities structure

## 2.1 Category

{

id: <int>,

parent\_id: <int>,

name: <string>,

photo: <url>,

sort: <int>,

filters: [

{

name: <string>,

type:<”array”|”bool”>

value: <array|null>

},

...

]

}

Категории сортируются по ключу который приходит с сервера. Если у нескольких категорий, одинаковый ключ - эти категории сортируются по алфавиту.

У каждой категории могут быть или список под категорий, или список заведений.

Фильтр категории формируется на основе списка ее атрибутов.

### АТРИБУТЫ

Каждая категория имеет набор атрибутов. Атрибуты - это набор полей и их значений, задаваемые пользователем (админом или контент менеджером).

Каждый атрибут характеризуется следующими полями:

* name (string) - название, текст который видят пользователи мобильных приложений и менеджеры в админке
* type (string) - тип атрибута. возможны значения:
  + url
  + string
  + array
  + bool
* value (bool|array|string) - хранит значения атрибута. Это может быть как булевский тип, так и массив или строка.
* require (bool) - определяет обязателен ли даннный атрибут для заполнения в заведении (нужен для типов: url, string, array).
* filterable (bool) - определяет будет ли этот атрибут отображаться в фильтре.

В категории создается и менеджируется массив ее атрибутов. А именно: для них задается поле name, type и value. Важно следующее: так как атрибуты это динамические характеристики заведений данной категории, то менеджер заполняет value в зависомости от значения в поле type. Если type = string|bool|url то поле value не заполняется (содержит NULL), если type = array, то поле value заполняется возможными значениями этого атрибута (позже из этого списка менеджер будет выбирать значения для конкретного заведения этой категории).

Далее атрибуты используются в фильтре мобильных приложений и в детальном экране заведения.

Например, категория Кафе может иметь следующие атрибуты:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **value** | **require** |
| Pinterest | url | null | false |
| Доставка | checkbox | null | - |
| Способы оплаты | list | Карточкой, Наличкой, Bitcoin, WebMoney | true |

Некое заведение Tanuki может иметь следующие атрибуты и их значения:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **type** | **value** | **require** |
| Pinterest | url | http://pinterest/1234566 | можно не заполнять |
| Доставка | checkbox | true | - |
| Способы оплаты | list | Наличкой, Bitcoin | обязательно нужно выбрать хотябы один элемент со списка |

Т.е. менеджер зайдя на детальный экран этого заведения в админке видит список атрибутов:

1. Pinterest - это текстовое поле для ввода URLа. Сюда вводит значение. Поле не обязательно для заполнения.
2. Доставка - это чекбокс. Отмечает его или оставляет пустым.
3. Способы оплаты - это поле в котором можно из списка выбрать несколько значений. Поле обязательно для заполнения.

Для реализации атрибутов на сервере рекомендую использовать NoSQL DB (например, MongoDB)

## 2.2 Place

{

id: <int>,

categories\_ids:[<array of categories IDs>],

name: <string>

description: <text>

photo: <url>,

photo\_big: <url>,

address: <string>,

latitude: <double>

longitude: <double>,

rate: <float>,

rate\_count: <int>,

phones:[<string>, <string>,....],

work\_time\_description: <string>,

work\_time\_start: <int>,

work\_time\_end: <int>,

web\_site: <url>,

promoted: <int>,

sort: <int>,

keys: [<string>, <string>, ...],

gallery: [

{

id: <int>,

photo: <url>,

photo\_big: <url>

},

....

],

properties:[

{

name: <string>,

type: <”url”|”string”|”array”|”bool”>,

value: <string|array\*|bool>

},

...

]

}

Заведения сортируются по ключу который приходит с сервера. Если у нескольких заведений, одинаковый ключ - эти заведения сортируются по алфавиту.

Заведения могут быть проплачены. Для обозначения этого использовать поле promoted, когда оно равно 0, значит заведение не проплачено, иначе - проплачено. Пока для проплаченного используем значение 1, но в дальнейшем может понадобится разделять на разные типы приплаченные заведения, тогда здесь будут значения 1,2,3… потому тип поля - integer.

Характеристики заведения формируются на основе заполненных атрибутов.

### 

### АТРИБУТЫ

В админке каждое заведение отображает список атрибутов из своей категории. Менеджер задает им конкретные значения.

если заведение будет лежать в разных категориях как учитывать фильтр

**\***Если поле type = array, то в поле value возвращается список **выбранных** значений (а не всех возможных)

## 

## 

## 2.3 Comment

{

id: <int>,

place\_id: <int>,

user\_id: <int>,

user\_name: <string>,

user\_photo: <url>,

comment: <text>,

rate: <int>,

create\_at: <timestamp>

},

Запросы должны возвращать только промодерированные коменты.

## 

## 

## 2.4 Action

{

id: <int>

place\_id: <int>,

name: <string>,

description: <text>,

text: <html>,

photo: <url>,

phoro\_big: <url>,

start\_date: <timestamp>,

end\_date: <timestamp>,

sort: <int>,

keys: [<string>, <string>, ...]

}

keys что означает

## 2.5 Banner

{

id: <int>,

photo: <url>,

transaction\_type: <place | action | site | none>,

endpoint: <url | place\_id | action\_id | null>

}

# 3. Synchronization

## 

## 3.1 Should client synchronize content?

REQUEST:

**GET:** /contents/is\_new

**PARAMS:**

update\_time=<timestamp>

RESPONSE:

{

code: 0,

data: {

has\_update: <0|1>

}

}

# 

# 

## 3.2 Get content

REQUEST:

**GET:** /contents

**PARAMS:**

update\_time=<timestamp>

Return zip file with JSON with next format:

{

categories: {

insert: [ <array of items to insert> ],

update: [ <array of items to update> ],

delete: [ <array of IDs to delete items > ],

},

places: {

insert: [ <array of items to insert> ],

update: [ <array of items to update> ],

delete: [ <array of IDs to delete items > ],

},

comments: [

insert: [ <array of items to insert> ],

update: [ <array of items to update> ],

delete: [ <array of IDs to delete items > ],

],

actions: {

insert: [ <array of items to insert> ],

update: [ <array of items to update> ],

delete: [ <array of IDs to delete items > ],

},

~~banners: [ <array of items to replace> ],~~

~~about\_app\_url: <string>~~

}

когда формируется файл? (на лету или генерится по требованию администратора или по крону)

# 4. Authorization

## 

## 4.1 User Signup

REQUEST

**POST:** /user/signup

**PARAMS:**

type=<fb|vk|tw>

name=<name>

social\_id=<string>

social\_id\_hash=<hash>

**FILES**:

avatar=<binary>

RESPONSE:

{

code:0,

data: {

access\_token: <token>

}

}

social\_id\_hash формируется как md5 от “city\_guide\_<social\_id>”. Это для защиты. Нужно проверять что бы переданный хеш и свормированный на сервере совпадали. + Если social\_id нет, то создавать пользователя, генерить и возвращать access\_token, а если пользователь уже есть, то пересоздать access\_token.

не понятен механизм авторизации

# 5. Actions

## 

## 5.1 Get action by id

REQUEST

**GET:** /actions/<action\_id>

RESPONSE:

{

code: 0,

data: { <action structure here> }

}

# 

# 

# 

# 6. Banners

## 

## 6.1 Get list of banners

REQUEST

**GET:** /banners

RESPONSE:

{

code: 0,

data: [ <array of banners structures> ]

}

# 7. Comments

## 

## 7.1 Send comment

REQUEST

**POST:** /comments/add

**HEADER**:

access\_token=<token>

**PARAMS:**

place\_id=<int>

comment=<comment>

rate=<int>

RESPONSE:

{

code: 0

}

# 

# 

# 8. Favorites

# 

## 8.1 Add to favorite

REQUEST

**POST:** /favorites

**HEADER**:

access\_token=<token>

**PARMAS:**

place\_id=<int>

type=”<place|category>”

RESPONSE:

{

code: 0

}

# 

# 

## 8.2 Delete from favorite

REQUEST

**DELETE:** /favorites

**HEADER**:

access\_token=<token>

**PARMAS:**

place\_id=<int>

type=”<place|category>”

RESPONSE:

{

code: 0

}

# 

# 

## 8.3 Get favorites

REQUEST:

**GET:** /favorites

**HEADER**:

access\_token=<token>

**PARMAS:**

type=<category|place>

RESPONSE if type=”place” :

{

code: 0

data: [ <array of places> ]

}

RESPONSE if type=”category” :

{

code: 0,

data: [ <array of categories> ]

}

# 

# 

# 

## 8.4 Sync favorites to server

REQUEST

**POST:** /favorites/sync

**HEADER**:

access\_token=<token>

**BODY:**

{

add:{

categories: [<array of ids to add to favorites>],

places: [<array of ids to add to favorites>]

}

delete:{

categories: [<array of ids to add to favorites>],

places: [<array of ids to add to favorites>]

}

}

RESPONSE:

{

code: 0,

data: {

categories: [ <array of ids in favorites> ],

places: [ <array of ids in favorites> ]

}

}

# 

**CLIENT WARNING:** On the client for vaforites create separate entity “CTFavorites” with fields:

id - id of entity in favorite

type - type of entity (place | category)

# 

# 9. Remote notifications & Settings

# 

## 9.1 Register push token

REQUEST

**POST:** /users/push

**HEADERS**:

access\_token=<token> (only if user authorized)

**PARMAS:**

push\_token=<token>

favorites=<0|1>

actions=<0|1>

RESPONSE:

{

code: 0,

data: {

favorites: <0|1>,

actions: <0|1>

}

}

**CLIENT WARNING:** On the client you should sychronize settings when user change value of some of settings parameter. To sync use this request.

**SERVER WARNING:**

* Any user can receive pushes: authorized and notauthorized.
* Logic of pushes descibed in UC27 in document “CityGuide UC”

## 

## 

## 9.2 Unregister push token

REQUEST

**DELETE:** /users/push

**HEADERS**:

access\_token=<token> (only if user authorized)

**PARMAS:**

push\_token=<token>

RESPONSE:

{

code:0

}

## 9.3 Get settings for remote notifications

REQUEST

**GET:** /users/push

**PARMAS:**

push\_token=<token>

RESPONSE:

{

code:0,

data: {

favorites: <0|1>,

actions: <0|1>

}

}

## 9.4 Remote notification parameters

body (standart field for message, i.e. aps->alert->body) - name of the action

place\_name - name of tha place (if action linked to place)

action\_url - relative url to action image