Техническое задание на доработку API

v2.0

Оглавление

Оглавление
Методы
- Статусы
- Общее для всех методов
  - Передача APP ID и UID устройства
- Ресурсы пользователя
  - Регистрация (существующий метод)
  - Изменение профиля (существующий метод)
  - Анкета пользователя (существующий метод)
  - Журнал вывода средств (новый метод)
  - Запрос на вывод средств (новая группа методов)
    - Получение информации о лимите (новый метод)
    - Вывод средств (новый метод)
- Ресурс “Задачи”
  - Список выполненных заданий (существующий метод)
  - Начать выполнение задачи (существующий метод)
- Push-сообщения
  - Проверка наличия заданий рядом с агентом (новый метод)
  - Проверка наличия новых сообщений (новый метод)
- Чат на стороне агента
  - Загрузка сообщений (новый метод)
  - Отправка сообщения (новый метод)
  - Проверка наличия новых сообщений на сервере (новый метод)
  - Статус чата на стороне сотрудника myTask и агента (новый метод)

Методы

В качестве исходного используется API v1.0 http://docs.mytask.apiary.io/

Статусы

Добавить новые статусы ответов сервера. Данный статус может вернуться для любого запроса от устройства агента:

Ответ	Описание
600	Пользователю доступен только чат
601	Пользователь переведен в режим “Заблокирован”

Общее для всех методов

Передача APP ID и UID устройства

Для всех запросов отправляемых на сервер требуется помимо token отправлять APP ID (ID приложения), а так же UID устройства (или иную связку уникальных идентификаторов приложения и устройства, доступных в операционной системы агента), переменные:

“appid”: “string” - APP ID
“uid”: ”string” - UID

Ответ сервера:

1. Любой статус, соответствующий запросу, кроме 600 или 601, означающий что пользователь, если был ранее заблокирован или ему был доступен только чат - разблокирован;
2. Статус 600 - пользователю доступен только чат;
3. Статус 601 - пользователь блокируется (пользователя разавторизовывает из приложения, не дает авторизоваться).

Ресурсы пользователя

Регистрация (существующий метод)

Добавить новую переменную:

1. “gender”:”string” - пол пользователя, возможные значения:
    - Мужской;
    - Женский.

Изменение профиля (существующий метод)

Добавить новую переменную:

1. “gender”:”string” - пол пользователя, возможные значения:
    - Мужской;
    - Женский.

Анкета пользователя (существующий метод)

Добавить две переменных для ответа сервера:

1. "balance": "float" - баланс пользователя;
2. “gender”:”string” - пол пользователя, возможные значения:
    - Мужской;
    - Женский.

Журнал вывода средств (новый метод)

Запрос истории вывода средств агентом.
Тип: GET /user/money/

Запрос:
Accept: application/json

Ответ:

```
200 (OK)
Content-Type: application/json
{
    "balance": "Float",
    {
        "itemId": ”String”,
        "sum": "Float",
        “createdAt": "unix time",
        "finishedAt": "unix time",
        “canceledAt”: “unix time”
    },
    {
        "itemId": ”String”,
        "sum": "Float",
        “createdAt": "unix time",
        "finishedAt": "unix time",
        “canceledAt”: “unix time”
    },
}
```

Переменные:

1. balance - баланс пользователя;
2. itemId - ID запроса выплаты;
3. sum - сумма выплаты;
4. createdAt - дата/время запроса на выплату, отправленного агентом;
5. finishedAt - дата/время отправки средств на счет пользователя;
6. canceledAt - дата/время отмены запроса на вывод средств.

Запрос на вывод средств (новая группа методов)

Новая группа методов. При отправке запроса на вывод средств, сначала отправляется запрос на текущий лимит вывода средств для агента.

Получение информации о лимите (новый метод)

Тип: GET /user/limit/

Запрос:
Accept: application/json

Ответ:

200 (OK)
Content-Type: application/json

```
{
    "limit": "Float",
}
```

Переменные:
limit - лимит на вывод средств для агента

Вывод средств (новый метод)

Тип: PUT /user/money/

Запрос:
Accept: application/json

```
{
    "sum": "Float",
}
```

Ответ:
204 (No Content)

Переменные:

1. sum - сумма выводимая на счет пользователя

Ресурс “Задачи”

Список выполненных заданий (существующий метод)

Для получения списка выполненных задач используется без изменений текущий метод: GET /tasks?filter={filter}&sort[{sortBy}]={sortType}&limit=14&offset=0=&distance=&longitude=&latitude=

Для параметра filter добавить новое возможное значение: completed (т.е. выполненные).

Итоговый список возможных значений для параметра filter:

1. available;
2. assigned;
3. completed.

Начать выполнение задачи (существующий метод)

При старте задачи отправлять на сервер текущее местоположение пользователя, в ответ сервер разрешает (статус 200) или запрещает старт выполнения задачи (статус 403).

Для отправки запроса используется существующий метод: PUT /tasks/{itemId}

Требуется добавить в метод два новых параметра: PUT /tasks/{itemId}?location=&longitude=&latitude, где: longitude и latitude - координаты пользователя.

Получить список задач (существующий метод)

При перемещении устройства агента на местности - срабатывает событие вызывающее существующий метод (подробнее см. ТЗ. на Доработки мобильного приложения): GET /tasks?filter={filter}&sort[{sortBy}]={sortType}&limit=14&offset=0=&distance=&longitude=&latitude=

В данном случае в методе передаются дополнительные параметры:

minSum - минимальная стоимость задания

typeTasks - фильтр по типам задач

Чат на стороне агента

Разработчикам предложить иные, более оптимальные варианты реализации (не через API) механизма обменам сообщениями.

Загрузка сообщений (новый метод)

При открытии чата, выполняется запрос списка предыдущих сообщений с сервера.

При прокрутке чата, после достижения конца списка, выполняется новый запрос с указанием смещения (параметр offset)/

Тип: GET /chat&limit=14&offset=0

Обязательные параметры:

1. limit - количество элементов в выдаче, по умолчанию 14
2. offset - смещение, по умолчанию 0

Запрос:

Accept: application/json

Ответ:

```
200 (OK)
Content-Type: application/json
{
    {
        "itemId": "String",
        "author": "String",
        "text": "String",
        "createdAt": "unix time",
        "files": {
            "fileName": "String",
            "URL": "String",
        }
    },
    {
        "itemId": "String",
        "author": "String",
        "text": "String",
        "createdAt": "unix time",
        "files": {
            "fileName": "String",
            "URL": "String",
        }
    }
}
```

Переменные:

1. itemId - ID сообщения;
2. author - автор сообщения;
3. title - текст сообщения;
4. createdAt - дата/время создания комментария;
5. fileName - имя файла с расширением;

6. URL - ссылка на файл для загрузки пользователем с сервера.

Отправка сообщения (новый метод)

В случае ответа сервера со статусом 204, сообщение выводится в окне истории чата, иначе выводится сообщение об ошибке.

Тип: PUT /chat

Запрос:

Accept: application/json

```
{
    "text": "String",
    "postData": "multipart form-data"
}
```

Ответ:

204 (No Content)

Переменные:

1. text - текст сообщения;
2. postData - файлы прикрепленные к сообщению;

Проверка наличия новых сообщений на сервере (новый метод)

В фоне, при наличии интернета, каждые 5 минут для закрытого чата (или свернутого приложения), и каждые 5 секунд для открытого чата, проверять наличие новых сообщений на сервере. Если на сервере найдены новые сообщения выполняется метод загрузки сообщений.

Тип: GET /chat

Запрос:

Accept: application/json

Ответ:

200 (OK)

Content-Type: application/json

```
{
    {
        "message": "bool",
    },
}
```

Статус чата на стороне сотрудника myTask и агента (новый метод)

Если у агента открыт чат, каждые 10 секунд выполняется запрос статуса чата на стороне сотрудника myTask. Таким образом на сервере по факту поступления запроса определяется, что чат у агента открыт.

Тип: GET /chat/status

Запрос:
Accept: application/json

Ответ:

200 (OK)
Content-Type: application/json

```
{
    {
        "status": "String",
    }
}
```

Переменные:

1. status - статус чата, возможные значения:
   - Открыт;
   - Закрыт.