**Спецификация Infodoctor WebAPI Ver. 1.0**

Содержание

[**Работа с юзерами**](#_gjdgxs) **2**

[**Работа с клиниками**](#_30j0zll) **3**

[**Работа c api городов**](#_3znysh7) **4**

[**Работа с отзывами на клиники**](#_2et92p0) **6**

[**Категории врачей**](#_3dy6vkm) **8**

[**Специализации врачей**](#_1t3h5sf) **9**

[**Статьи**](#_8xexruvdehqo) **14**

[**Поиск**](#_33kdt72sm1jt) **17**

# Работа с юзерами

Регистрация:

URL: /api/account/register,

METHOD: POST,

HEADER:

Content-Type : application/json; charset=utf-8,

BODY:

JSON объекта

var data = {

Email: ,

Password: ,

UserName: ,

Phone:

};

*Поле "Phone" является необязательным, остальные обязательны!*

Ответ:

HttpResponse

Пример:

|  |  |
| --- | --- |
| Запрос | {  "Email":"test1@test.by",  "Password":"1234Aa",  "UserName":"TestName1",  "Phone":"+123456789"  } |
| Ответ если удачно | Status 200 OK |
| Ответ отсутствует обязательное поле | Status 400 Bad Request  {  "Message": "The request is invalid.",  "ModelState": {  "model.Email": [  "Требуется поле Email."  ]  }  } |
| Ответ если такой Email или UserName уже имеются в БД | Status 400 Bad Request  {  "Message": "The request is invalid.",  "ModelState": {  "": [  "Name test1@test.by is already taken.",  "Email 'test1@test.by' is already taken."  ]  }  } |

Авторизация:

URL: /token,

METHOD: POST,

HEADER:

Content-Type : application/x-www-form-urlencoded,

BODY:

grant\_type: "password",

username,

password

Ответ:

{

"access\_token": "token",

"token\_type": "bearer",

"expires\_in": 1209599,

"userName": "user",

".issued": "Wed, 22 Feb 2017 16:54:22 GMT",

".expires": "Wed, 08 Mar 2017 16:54:22 GMT"

}

# Работа с клиниками

URL: /api/clinic/{id} – общий вид

где {id} – необязательный параметр

Если id указан, то возвращается клиника с этим id, если не указан – то массив всех клиник

Пример:

|  |  |
| --- | --- |
| запрос клиники 3: /api/clinic/3  метод GET | Status 200 OK  {  "Id": 3,  "Name": "Медицинский центр Кравира",  "Email": "kravira@kravira.by",  "ClinicAddress": [  {  "Street": "пр-т. Победителей 75,",  "City": "Минск",  "ClinicPhones": [  {  "Desc": "",  "Phone": "+375 17 211 28 61"  },  {  "Desc": "",  "Phone": "+375 29 611 28 61"  }  ]  },  {  "Street": "ул.Скрипникова 11Б,",  "City": "Минск",  "ClinicPhones": [  {  "Desc": "",  "Phone": "+375 17 314 94 94"  },  {  "Desc": "",  "Phone": "+375 29 664 44 44"  }  ]  },  {  "Street": "ул.Захарова 50Д",  "City": "Минск",  "ClinicPhones": [  {  "Desc": "",  "Phone": "+375 17 253 33 33"  },  {  "Desc": "",  "Phone": "+375 29 103 43 43"  }  ]  }  ],  "ClinicSpecialization": [  {“Id”: 1, “Name”:"Гинекология"},  {“Id”: 2, “Name”:"Дерматология", *и т.д.*  ],  "RatePrice": 3,  "RateQuality": 4,  "RatePoliteness": 5,  "RateAverage" : 4,  "Favorite": “false”} |
| запрос клиники, которой нет в БД :  /api/clinic/666  метод GET | Status 500 Internal Server Error  {  "Message": "An error has occurred.",  "ExceptionMessage": "Clinic not found",  "ExceptionType": "System.ApplicationException",  "StackTrace": " в Infodoctor.BL.Services.ClinicService.GetClinicById(Int32 id) …  … и еще много букв  } |

Общий вид объекта clinic

"Id"– id клиники в базе данных

"Image"– путь к картинке клиники

"Name" – название клиники

"Email" – адрес эл. Почты

"ClinicSpecialization" – массив специализаций клиники

"ClinicAddress" – массив адресов филиалов клиник. Если клиника без филиалов, то в массиве будет 1 адрес

"City" – название города, в котором находится клиника

"Street" – улица и дом клиники

"ClinicPhones" – массив телефонов, соответствующий каждому филиалу клиники

"Desc": описание номера, например, "Детское отделение",

"Phone": собственно, номер

"RatePrice " – рейтинг по стоимости услуг

"RateQuality " – рейтинг по качеству услуг

"RatePoliteness " – рейтинг вежливости персонала

"RateAverage " – общий средний рейтинг

"Favorite" – приоритетная ли клиника (спонсируемая)

Пагинация клиник

URL: /api/clinic/page/{perPage}/{numPage}

где perPage и numPage значения типа int

Body:

{

Clinics[] – массив объектов типа clinic (см. выше)

Page – номер страницы (совпадает с numPage в запросе)

PageSize – – номер страницы (совпадает с perPage в запросе)

TotalCount – общее количество объектов типа clinic в БД

TotalPages – общее количество страниц при условии PageSize

}

Пример:

|  |  |
| --- | --- |
| /api/clinic/page/2/2  метод GET | Status 200 OK  {  "Clinics": [],  "TotalCount": 5,  "Page": 2,  "PageSize": 2,  "TotalPages": 3  } |
| /api/clinic/page/10/666  метод GET  Запрос страницы с номером больше, чем страниц может существовать | Status 500 Internal Server Error  {  "Message": "An error has occurred.",  "ExceptionMessage": "Page not found",  "ExceptionType": "System.ApplicationException",  "StackTrace": " в …  … и еще много букв  } |
| /api/clinic/page/-10/0  метод GET  Значения переменных perPage или numPage < 1 | Status 500 Internal Server Error  {  "Message": "An error has occurred.",  "ExceptionMessage": "Incorrect request parameter",  "ExceptionType": "System.ApplicationException",  "StackTrace": " в …  … и еще много букв  } |
| /api/clinic/page/+-=/xyz  метод GET  Переменные perPage или numPage не типа int | Status 404 Not Found |

# Работа c api городов

Получение всего списка Городов:

METHOD: GET,

URL: /api/cities,

HEADER: none,

BODY: none,

Ответ:

Json List

[

{

"Id": 1,

"Name": "Минск"

},

{

"Id": 2,

"Name": "Брест"

},

...

]

Получение всего списка городов имеющих клиники:

METHOD: GET,

URL: /api/cities/clinicused,

HEADER: none,

BODY: none,

Ответ:

Json List

[

{

"Id": 1,

"Name": "Минск"

},

{

"Id": 2,

"Name": "Брест"

},

...

]

Получение города по численному id:

URL: /api/cities,

PARAM: id(int),

METHOD: GET,

HEADER: none,

BODY: none,

Ответ:

Json объект

{

"Id": 1,

"Name": "Минск"

}

Добавление нового города:

METHOD: POST,

URL: /api/cities,

PARAM: none,

HEADER:

Authorization : Bearer token,

Content-Type : application/x-www-form-urlencoded,

BODY: название нового элемента(string),

Ответ: none

Изменение существующего города:

METHOD: PUT,

URL: /api/cities,

PARAM: id(int),

HEADER:

Authorization : Bearer token,

Content-Type : application/x-www-form-urlencoded,

BODY: новое значение для существующего элемента(string),

Ответ: none

Удаление существующего города:

METHOD: DELETE,

URL: /api/cities,

PARAM: id(int),

HEADER:

Authorization : Bearer token,

BODY: none,

Ответ: none

# Работа с отзывами на клиники

Получение отзыва по Id:

URL: api/clinicreview/{Id}

Метод GET

Где: Id – Id отзыва в БД, тип int

Body:

{

ClinicId – Id клиники, на которую пишется отзыв

Id - Id отзыва в БД

UserId" – Id автора в БД

UserName" – Имя автора в БД

Text" - текст отзыва

RatePrice" – оценка стоимости услуг

RateQuality" - оценка качества услуг

RatePoliteness – оценка вежливости персонала

}

Требуется авторизация пользователя

Получение всех отзывов:

URL: /api/clinicreview

Метод GET

Body:

[] – массив объектов типа clinicreview (см. выше)

Требуется авторизация пользователя с ролью "admin" или "moder"

Пагинация отзывов по Id клиники:

URL: /api/ClinicReview/page/{clinicId}/{perPage}/{numPage}

Метод GET

Body:

{

ClinicReviews[] – массив объектов типа clinicreview (см. выше)

Page – номер страницы (совпадает с numPage в запросе)

PageSize – номер страницы (совпадает с perPage в запросе)

TotalCount – общее количество объектов типа clinicreview в БД

TotalPages – общее количество страниц при условии PageSize

}

Авторизация не требуется

Добавление:

URL: /api/clinicreview

Метод PUT

Body:

{

ClinicId – Id клиники, на которую пишется отзыв

Text - текст отзыва

RatePrice – оценка стоимости услуг, тип int

RateQuality - оценка качества услуг, тип int

RatePoliteness – оценка вежливости персонала, тип int

}

*Все поля обязательны*

Требуется авторизация пользователя

Удаление отзыва:

URL: /api/clinicreview/{Id}

Где: Id – Id удаляемого отзыва, тип int

Метод DELETE

Body:

пусто

Требуется авторизация пользователя с ролью "admin" или "moder"

Пример:

|  |  |
| --- | --- |
| /api/clinicreview/page/1/2/3  Метод GET | Status 200 OK  {  "ClinicReviews": [],  "TotalCount": 6,  "Page": 3,  "PageSize": 2,  "TotalPages": 3  } |
| /api/clinicreview/page/1/2/666  метод GET  Запрос страницы с номером больше, чем страниц может существовать | Status 500 Internal Server Error  {  "Message": "An error has occurred",  "ExceptionMessage": "Page not found",  "ExceptionType": "System.ApplicationException",  "StackTrace": " в …  … и еще много букв  } |

|  |  |
| --- | --- |
| /api/clinicreview /page/-10/0/0  метод GET  Значения переменных clinicId, perPage или numPage < 1 | Status 500 Internal Server Error  {  "Message": "An error has occurred.",  "ExceptionMessage": "Incorrect request parameter",  "ExceptionType": "System.ApplicationException",  "StackTrace": " в …  … и еще много букв  } |

|  |  |
| --- | --- |
| /api/clinicreview  Метод POST  Body:  {  "ClinicId": 1,  "Text": "Test",  "RatePrice": 3,  "RateQuality": 4,  "RatePoliteness": 3  } | Status 204 No Content |
| /api/clinicreview  Метод POST  Body:  {  "ClinicId": 1,  "Text": "",  "RatePrice": 0,  "RateQuality": 4,  "RatePoliteness": 3  }  Некоторые поля некорректны | Status 500 Internal Server Error  {  "Message": "An error has occurred.",  "ExceptionMessage": "Incorrect data, some required fields are null or empty",  "ExceptionType": "System.ApplicationException",  "StackTrace": " в …  … и еще много букв  } |
| /api/clinicreview  Метод POST  Body:  {  "ClinicId": 1,  "Text": "Test",  "RatePrice": 3,  "RateQuality": 4,  "RatePoliteness": 3  }  Пользователь не авторизован | Status 401 Unauthorized  Body:  {  "Message": "Authorization has been denied for this request."  } |
|  |  |
| /api/clinicreview/5  Метод DELETE | Status 204 No Content |
| /api/clinicreview/666  Метод DELETE  Если отзыва c Id =666 нет в БД | Status 500 Internal Server Error  {  "Message": "An error has occurred.",  "ExceptionMessage": " Review not found",  "ExceptionType": "System.ApplicationException",  "StackTrace": " в Infodoctor.BL.Services.ClinicService.GetClinicById(Int32 id) …  … и еще много букв  } |
| Ответ отсутствует обязательное поле | Status 400 Bad Request  {  "Message": "The request is invalid.",  "ModelState": {  "model.Email": [  "Требуется поле Email."  ]  }  } |

# Категории врачей

Получение всего списка категории врачей:

METHOD: GET,

URL: /api/doctorcategories,

HEADER: none,

BODY: none,

Ответ:

Json List

[

{

"Id": 1,

"Name": "Без категории",

"Doctors": [2]

},

...

]

Получение списка категорий врачей по численому id:

URL: /api/doctorcategories,

PARAM: id(int),

METHOD: GET,

HEADER: none,

BODY: none,

Ответ:

Json объект

{

"Id": 1,

"Name": "Без категории",

"Doctors": [2]

}

# Специализации врачей

Получение всего списка специализаций врачей:

METHOD: GET,

URL: /api/doctorspecialisations,

HEADER: none,

BODY: none,

Ответ:

Json List

[

{

"Id": 1,

"Name": "Аллерголог",

~~"Doctors": [ 1 ],~~

~~"ClinicSpecializationId": 1~~

},

...

]

Получение специализации враче по численому id:

URL: /api/doctorspecialisations,

PARAM: id(int),

METHOD: GET,

HEADER: none,

BODY: none,

Ответ:

Json объект

{

"Id": 1,

"Name": "Аллерголог",

~~"Doctors": [ 1 ],~~

~~"ClinicSpecializationId": 1~~

}

Получение всего списка врачей:

METHOD: GET,

URL: /api/doctor,

HEADER: none,

BODY: none,

Ответ:

Json List

[

"Id": 1,

"Name": "Степанов Иван Степанович",

"Email": "giperivan2@gmail.com",

"Experience": 20,

"Manipulation": "Может что-то отлично.",

"Address": {

"Street": "пр-т. Независимости 58",

"City": "Минск",

"ClinicPhones": [

{

"Desc": "",

"Phone": "+375 29 110 12 12"

}

]

},

"Specialization": {“Id”: 1, “Name”:"Аллерголог"},

"Category": "Первая категория",

"ClinicsId": [ 1,2 ],

...

]

Пагинация врачей

METHOD: GET,

URL: /api/doctor/page/2/2

где /api/doctor/page/элементов на страницу/номер страницы,

HEADER: none,

BODY: none,

Ответ:

Json объект

{

"Doctors":

[

{

"Id": 3,

"Name": "Степанов Степан Степанович",

"Email": "infosuperstepa1999@gmail.com",

"Experience": 14,

"Manipulation": "Может что-то хорошо.",

"Address": {

"Street": "ул.Сурганова 47Б",

"City": "Минск",

"ClinicPhones": [

{

"Desc": "",

"Phone": "159"

},

{

"Desc": "",

"Phone": "(017) 296 62 72"

}

]

},

"Specialization": {“Id”: 33, “Name”:"Массажист"},

"Category": "Вторая категории",

"ClinicsId": [

1

]

},

{

"Id": 4,

"Name": "Иванов Степан Степанович",

"Email": "tainiidoctor2@gmail.com",

"Experience": 14,

"Manipulation": "Может что-то хорошо.",

"Address": {

"Street": "пр-т. Независимости 58",

"City": "Минск",

"ClinicPhones": [

{

"Desc": "",

"Phone": "+375 29 110 12 12"

}

]

},

"Specialization": {“Id”: 55, “Name”:"Онколог"},

"Category": "Кандидат в доктора медицинских наук",

"ClinicsId": [

2

]

}

],

"TotalCount": 5,

"Page": 2,

"PageSize": 2,

"TotalPages": 3

}

Работа с отзывами на врачей

Получение всего списка отзывов на врачей:

METHOD: GET,

URL: /api/doctorreview,

HEADER: none,

BODY: none,

Ответ:

Json List

[

{

"DoctorId": 1,

"Id": 1,

"UserId": "7d374085-71e4-4819-8d09-91cfc8239463",

"UserName": "user0",

"Text": "orem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.",

"PublishTime": "2016-11-16T02:12:57.377",

"RatePrice": 4,

"RateQuality": 3,

"RatePoliteness": 3

},

...

]

Получение отзыва на врача по численному id:

URL: /api/doctorreview,

PARAM: id(int),

METHOD: GET,

HEADER: none,

BODY: none,

Ответ:

Json объект

{

"DoctorId": 1,

"Id": 1,

"UserId": "7d374085-71e4-4819-8d09-91cfc8239463",

"UserName": "user0",

"Text": "orem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.",

"PublishTime": "2016-11-16T02:12:57.377",

"RatePrice": 4,

"RateQuality": 3,

"RatePoliteness": 3

}

Добавление нового города:

METHOD: POST,

URL: /api/doctorreview,

PARAM: none,

HEADER:

Authorization : Bearer token,

Content-Type : application/x-www-form-urlencoded,

BODY:

JSON

{

"DoctorId": 3,

"Text": "Test",

"RatePrice": 3,

"RateQuality": 4,

"RatePoliteness": 3

}

Ответ: none

Удаление существующего города:

METHOD: DELETE,

URL: /api/doctorreview,

PARAM: id(int),

HEADER:

Authorization : Bearer token,

BODY: none,

Ответ: none

# 

# 

# Статьи

Получение всего списка статей:

METHOD: GET,

URL: /api/articles,

HEADER: none,

BODY: none,

Ответ:

Json List

[

{

"Id": 1,

"Title": "test1",

"Content": "<article><h1>Test1</h1><p><img src=\"\"/></p></article>",

"PublishDate": "2017-03-15T22:00:52.61",

"Author": "admin"

},

...

]

Получение статьи по численному id:

METHOD: GET,

URL: /api/articles,

PARAM: id(int),

HEADER: none,

BODY: none,

Ответ:

Json объект

{

"Id": 1,

"Title": "test1",

"Content": "<article><h1>Test1</h1><p><img src=\"\"/></p></article>",

"PublishDate": "2017-03-15T22:00:52.61",

"Author": "admin"

}

METHOD: GET,

URL: /api/articles/page/2/2 где /api/articles/page/элементов на страницу/номер страницы,

HEADER: none,

BODY: none,

Ответ:

Json объект

{

"Articles":

[

{

"Id": 1,

"Title": "test1",

"Content": "<article><h1>Test1</h1><p><img src=\"\"/></p></article>",

"PublishDate": "2017-03-15T22:00:52.61",

"Author": "admin"

},

...

],

"TotalCount": 1,

"Page": 1,

"PageSize": 5,

"TotalPages": 1

}

Загрузка изображения для новости:

METHOD: POST,

URL: /api/articles/postimage,

PARAM: none,

HEADER:

Authorization : Bearer token(доступ есть только у администратора и модератора),

Content-Type : form-dara,

BODY: картинка типа "file",

Ответ:

string

Если в ответе пришла пустая строка, то изображение не загрузилось

Если изображение загружено приходит полная ссылка на это изображение

Добавление новой статьи:

METHOD: POST,

URL: /api/articles,

PARAM: none,

HEADER:

Authorization : Bearer token(доступ есть только у администратора и модератора),

Content-Type : application/json; charset=utf-8,

BODY:

Json объект

{

"title":"test1",

"content": "<article><h1>Test1</h1><p><img src=\"\"/></p></article>"

},

Ответ: none

Изменение существующей статьи:

METHOD: PUT,

URL: /api/articles,

PARAM: id(int),

HEADER:

Authorization : Bearer token(доступ есть только у администратора и модератора),

Content-Type : application/json; charset=utf-8,

BODY:

Json объект

{

"title":"test2",

"content": "<article><h1>Test2</h1><p><img src=\"\"/></p></article>"

},

Ответ: none

Удаление существующей статьи:

METHOD: DELETE,

URL: /api/articles,

PARAM: id(int),

HEADER:

Authorization : Bearer token(доступ есть только у администратора и модератора),

BODY: none,

Ответ: none

# Поиск

Быстрый поиск (для автодополнения):

URL: /api/search/fastsearch,

METHOD: POST,

HEADER:

Content-Type : application/json; charset=utf-8,

BODY:

JSON объекта

var data = {

Text: "",

TypeId: []

};

В поле TypeId сервер принимает массив чисел, указывающий на категории для поиска.

1 - поиск по клиникам(названия, специализации);

2 - поиск по врачам(имена, специализации, манипуляции);

Пример:

{

"Text":"аллер",

"TypeId":[1,2]

}

Ответ:

["",""] (list строк)

Поиск по клиникам

поиск по клиникам c пагинацией

URL: "api/clinic/search/{perPage:int}/{numPage:int}"

Метод POST

HEADER:

Content-Type: application/json

BODY:

JSON объекта:

SearchWord:,

CityId: ,

SpecializationIds: ,

SortBy: ,

Descending:

где:

SearchWord - string с поисковым запросом (если строка пустая, то выдаются все врачи в соответствии с нижеуказанными условиями поиска)

CityId - Id города (если 0, то город не учитыватся, будет поиск по всем города)

SpecializationIds – массив [] Id специализации (массив пустой, то специализации не учитываются)

SortBy - string c видом сортировки, доступные варианты "alphabet", "rate" и "price"

Descending - string сортировка по убыванию "true" или "false"

Возвращается такой же массив объектов, как и для Пагинация клиник

Если в результате фильтрации не найдено ни одной подходящей клиники, то возвращается null

Реакция на ошибки в параметрах пагинации такая же, как и в п. Пагинация клиник

Пример:

|  |  |
| --- | --- |
| /api/clinic/search/10/1  метод POST  {  SearchWord:"мЕД",  CityId: 0,  SpecializationIds: [21,22],  SortBy: "price",  Descending: "false"  } | Status 200 OK  {  "Clinics": [],  "TotalCount": 5,  "Page": 2,  "PageSize": 2,  "TotalPages": 3  } |

Поиск по врачам

поиск по врачам c пагинацией

URL: "api/doctor/search/{perPage:int}/{numPage:int}"

Метод POST

в запросе должен присутствовать Json объект c полями:

SearchWord:,

CityId: ,

SpecializationId: ,

SortBy: ,

Descending:

где:

SearchWord - string с поисковым запросом (если строка пустая, то выдаются все врачи в соответствии с нижеуказанными условиями поиска)

CityId - Id города (если 0, то город не учитыватся, будет поиск по всем города)

SpecializationId - Id специализации (если 0, то специализации не учитываются)

SortBy - string c видом сортировки, доступные варианты "alphabet", "rate" и "prof"

Descending - string сортировка по убыванию "true" или "false"

Пример:

|  |  |
| --- | --- |
| /api/clinic/search/10/1  метод POST  {  SearchWord:"Иван",  CityId: 0,  SpecializationId: 0,  SortBy: "alphabet",  Descending: "false"  } | Status 200 OK  {  "Doctors": [],  "ReviewsIds": null,  "RateProfessionalism": 4.166666666666667,  "RateWaitingTime": 4.166666666666667,  "RatePoliteness": 3.8333333333333335,  "RateAverage": 4.0555555555555562  }  ],  "TotalCount": 3,  "Page": 1,  "PageSize": 10,  "TotalPages": 1  } |

Возвращается такой же массив объектов, как и для Пагинация врачей

Если в результате фильтрации не найдено ни одного подходящего доктора, то возвращается null

Реакция на ошибки в параметрах пагинации такие же как и в п. Пагинация клиник