

REST API & Retrofit

Материалы



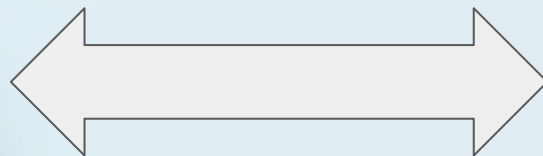
github.com/adamxrvn/hse-lyceum-android-course

REST - Пример

Приложение может общаться с сервером посредством REST API



REST API



Что такое REST?

REST - **Re**presentational **S**tate **T**ransfer («передача репрезентативного состояния») - это архитектурный подход взаимодействия сайтов и приложений с сервером

Преимущества REST:

- Простота/стандартизация
- Масштабируемость/отсут. состояний
- Производительность/кэширование

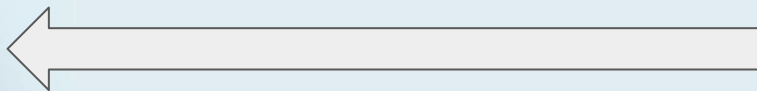
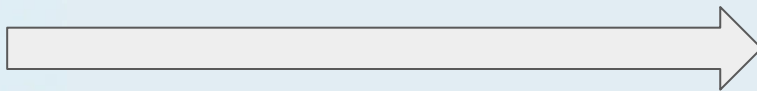
REST - Пример

Приложение

Endpoint:
<https://сендвичи.рф/api/menu>



Запрос



Ответ



Запросы

Функции работы с БД

Методы HTTP-запросов

Create → Post

Read → Get

Update → Put

Delete → Delete

Запросы

Из чего состоят запросы?

Content-Type, Authorization, и т.д.

Заголовки

GET/POST/PUT/DELETE/и т.д.

Метод

ADDRESS

Endpoint

Параметры/тело

Пример GET запроса

Запрос:

Заголовки

GET

Метод

<https://сендвичи.рф/api/menu>

Endpoint

Параметры/тело

Ответ:

```
[
  {
    "id": 0,
    "name": "Classic Italian",
    "price": 290,
    "icon": "https://img.com/ClassicItalian.png"
  },
  {
    "id": 1,
    "name": "Turkey Ranch & Swiss",
    "price": 400,
    "icon": "https://img.com/TurkeyRanch.png"
  },
  {
    "id": 2,
    "name": "Veggie Delight",
    "price": 52,
    "icon": "https://img.com/Veggie.png"
  }
]
```


Пример PUT запроса

Запрос:

Заголовки

PUT

Метод

<https://сэндвичи.рф/api/menu/1>

Endpoint

`{"name": "Chicken Sandwich", "price": 350}`

Параметры/тело

Ответ:

```
{  
  "id": 1,  
  "name": "Chicken Sandwich",  
  "price": 350,  
  "icon": "https://img.com/TurkeyRanch.png"  
}
```

Пример POST запроса

Запрос:

Заголовки

POST

Метод

<https://сэндвичи.рф/api/menu/>

Endpoint

```
{"name": "RESTful Sandwich", "price": 999,  
"icon": "https://img.com/rest.png"}
```

Параметры/тело

Ответ:

```
{  
  "id": 3,  
  "name": "RESTful Sandwich",  
  "price": 999,  
  "icon": "https://img.com/rest.png"  
}
```

Пример документации

(ссылка на гите)

FastAPI 0.1.0 OAS 3.1

[/openapi.json](#)

default



GET	/sandwiches/	Get Sandwiches	▼
POST	/sandwiches/	Add Sandwich	▼
PUT	/sandwiches/{sandwich_id}	Update Sandwich	▼
DELETE	/sandwiches/{sandwich_id}	Delete Sandwich	▼



Метод



Ресурс



Описание

Пример документации

Параметры запроса

Тело запроса

Варианты ответа сервера

PUT

/sandwiches/{sandwich_id} Update Sandwich

⌵

Parameters

Try it out

Name

Description

sandwich_id * required

integer

(path)

sandwich_id

Request body required

application/json

Example Value | Schema

```
{  "name": "string",  "price": 0,  "icon": "https://example.com/"}
```

Responses

Code

Description

Links

200

Successful Response

No links

Media type

application/json

Controls Accept header.

Example Value | Schema

```
{  "id": 0,  "name": "string",  "price": 0,  "icon": "https://example.com/"}
```

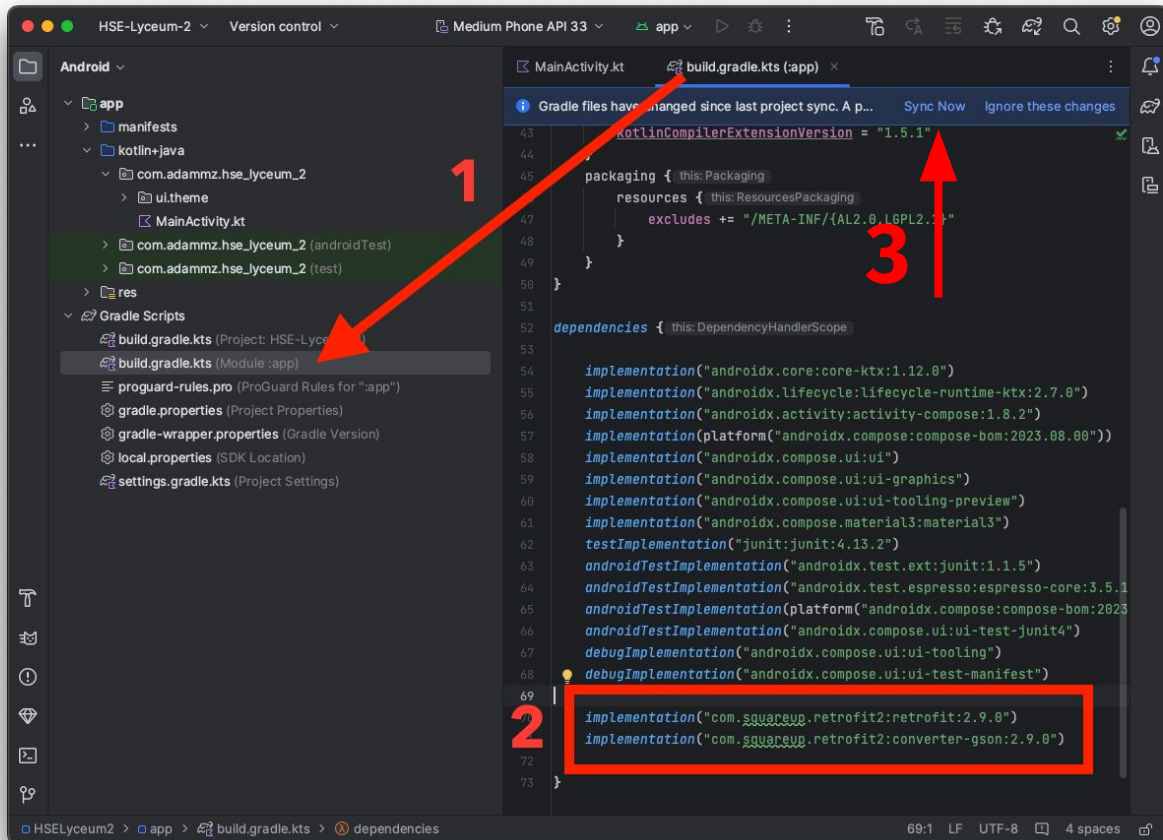
422

Validation Error

No links

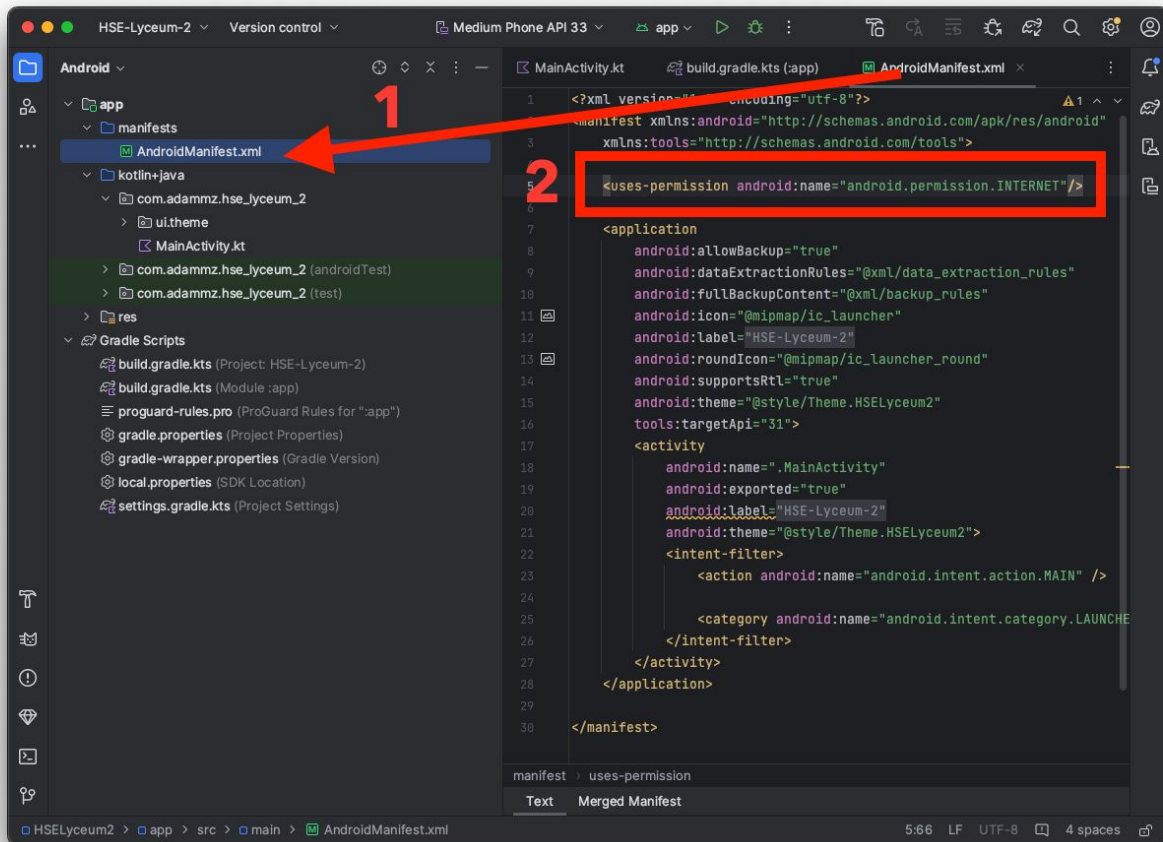
Media type

Перейдем к Android!



Для работы с REST API
нам нужно добавить
библиотеку Retrofit

Internet Permission



Добавляем разрешение
для интернета

Создание дата класса

Массив объектов

```
[  
  {  
    "id": 0,  
    "name": "Classic Italian",  
    "price": 290,  
    "icon": "https://img.com/ClassicItalian.png"  
  },  
  {  
    "id": 1,  
    "name": "Turkey Ranch & Swiss",  
    "price": 400,  
    "icon": "https://img.com/TurkeyRanch.png"  
  },  
  {  
    "id": 2,  
    "name": "Veggie Delight",  
    "price": 52,  
    "icon": "https://img.com/Veggie.png"  
  }  
]
```

Объект, в нашем случае сэндвич,
со следующими полями:

Id - Int

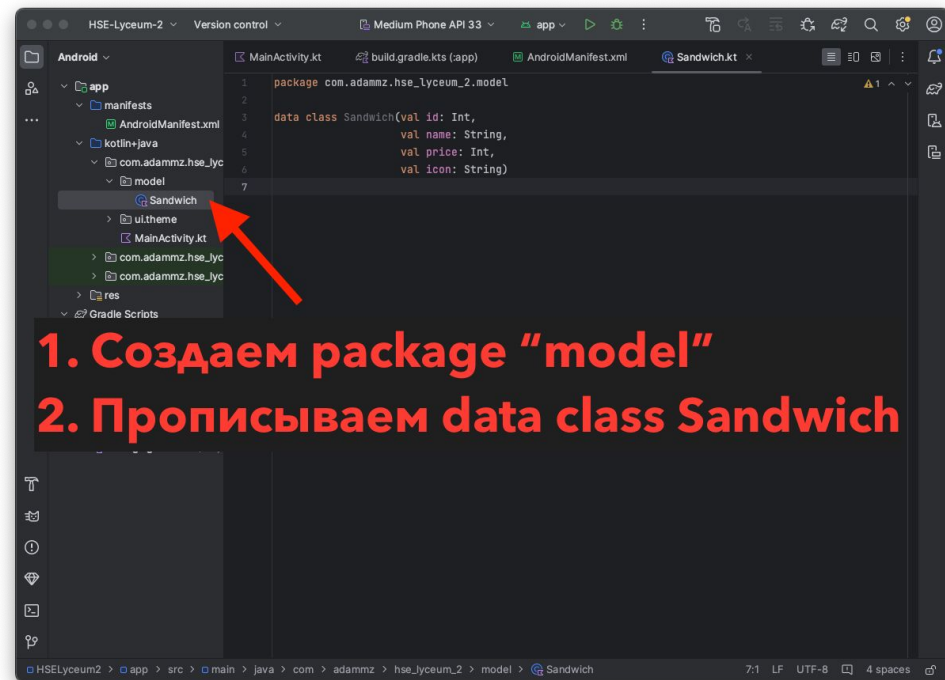
Name - String

Price - Int

Icon - String

Создание дата класса

```
[  
{  
  "id": 0,  
  "name": "Classic Italian",  
  "price": 290,  
  "icon": "https://img.com/ClassicItalian.png"  
},  
{  
  "id": 1,  
  "name": "Turkey Ranch & Swiss",  
  "price": 400,  
  "icon": "https://img.com/TurkeyRanch.png"  
},  
{  
  "id": 2,  
  "name": "Veggie Delight",  
  "price": 52,  
  "icon": "https://img.com/Veggie.png"  
}  
]
```

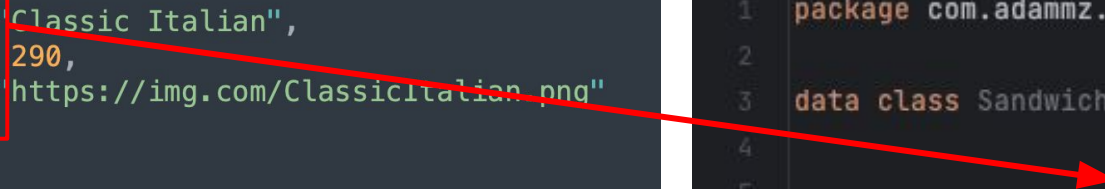


Создание дата класса

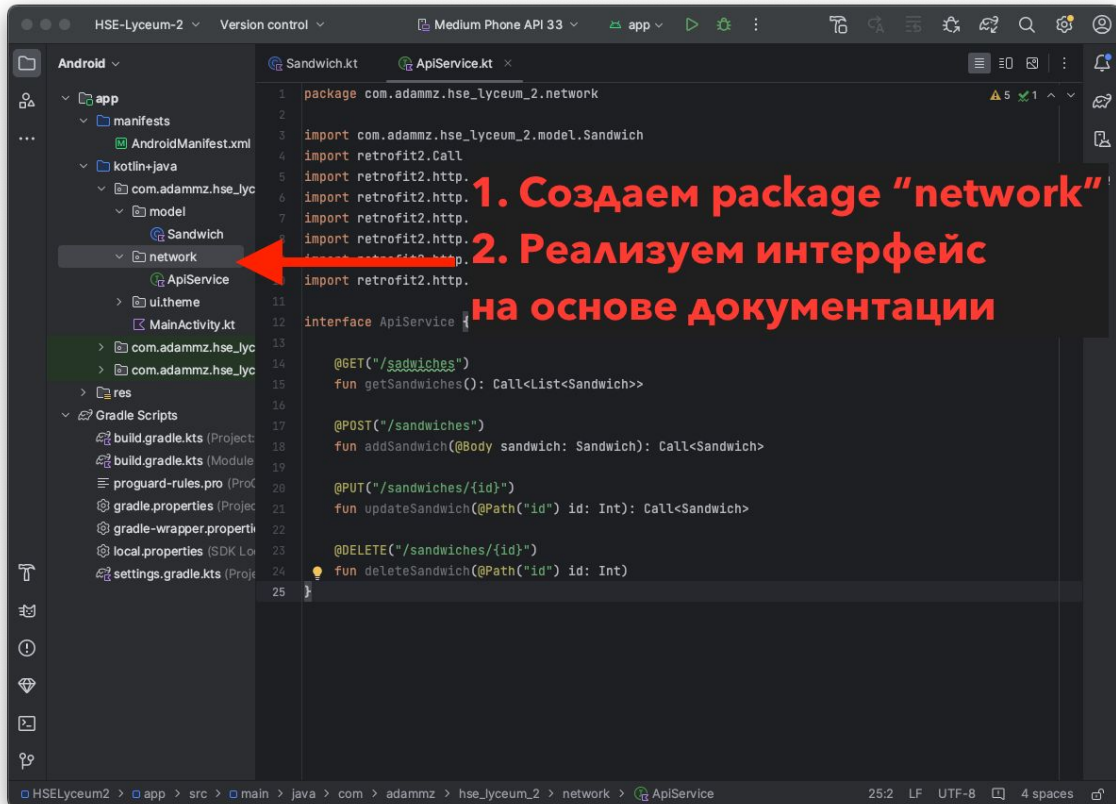
```
[  
  {  
    "id": 0,  
    "name": "Classic Italian",  
    "price": 290,  
    "icon": "https://img.com/ClassicItalian.png"  
  },  
  {  
    "id": 1,  
    "name": "Turkey Ranch & Swiss",  
    "price": 400,  
    "icon": "https://img.com/TurkeyRanch.png"  
  },  
  {  
    "id": 2,  
    "name": "Veggie Delight",  
    "price": 52,  
    "icon": "https://img.com/Veggie.png"  
  }  
]
```

Sandwich.kt ×

```
1 package com.adammz.hse_lyceum_2.model  
2  
3 data class Sandwich(val id: Int,  
4                     val name: String,  
5                     val price: Int,  
6                     val icon: String)  
7
```



Создаем интерфейс



Создаем интерфейс

```
interface ApiService {  
    Метод Путь к ресурсу  
    @GET("/sadwiches")  
    fun getSandwiches(): Call<List<Sandwich>>  
  
    @POST("/sandwiches") Ответ сервера  
    fun addSandwich(@Body sandwich: Sandwich): Call<Sandwich>  
  
    @PUT("/sandwiches/{id}")  
    fun updateSandwich(@Path("id") id: Int): Call<Sandwich>  
  
    @DELETE("/sandwiches/{id}")  
    fun deleteSandwich(@Path("id") id: Int)  
}
```

Создаем интерфейс

GET

/sandwiches/ Get Sandwiches



```
@GET("/sandwiches")  
fun getSandwiches(): Call<List<Sandwich>>
```

POST

/sandwiches/ Add Sandwich



```
@POST("/sandwiches")  
fun addSandwich(@Body sandwich: Sandwich): Call<Sandwich>
```

PUT

/sandwiches/{sandwich_id} Update Sandwich



```
@PUT("/sandwiches/{id}")  
fun updateSandwich(@Path("id") id: Int): Call<Sandwich>
```

DELETE

/sandwiches/{sandwich_id} Delete Sandwich



```
@DELETE("/sandwiches/{id}")  
fun deleteSandwich(@Path("id") id: Int)
```