



# Institut Teknologi Telkom Purwokerto 2020/2021

## PEMROGRAMAN PERANGKAT BERGERAK

### Modul Praktikum Topik 14

- Pengenalan QuotesAPI
- CRUD REST API

### 14.1 Persiapan

1. Android Studio
2. Device Android/Emulator Android
3. Kabel Data
4. Buat module praktikum 14  
Nama activity **MainActivity (Bottom Navigation Activity)**
5. Aplikasi Postman <https://www.postman.com/downloads/>

### 14.2 Pengenalan QuotesAPI

1. Pada praktikum ini kita akan menggunakan REST API dari <https://quotes.informatika.app/>, terdapat beberapa endpoint yang dapat digunakan untuk pengujian atau pelatihan CRUD.

No	Method	Endpoint	Parameter
1	POST	<b>Register</b> <a href="https://quotes.informatika.app/auth/registration">https://quotes.informatika.app/auth/registration</a>	class_id registration_number name email password
2	POST	<b>Login</b> <a href="https://quotes.informatika.app/auth/login">https://quotes.informatika.app/auth/login</a>	registration_number password
3	POST	<b>Create Quote</b> <a href="https://quotes.informatika.app/api/v1/quotes">https://quotes.informatika.app/api/v1/quotes</a>	bearer token name description
4	GET	<b>My Quote</b> <a href="https://quotes.informatika.app/api/v1/myquotes">https://quotes.informatika.app/api/v1/myquotes</a>	bearer token
5	GET	<b>Class Quote</b> <a href="https://quotes.informatika.app/api/v1/class_quotes">https://quotes.informatika.app/api/v1/class_quotes</a>	bearer token
6	GET	<b>Global Quote</b> <a href="https://quotes.informatika.app/api/v1/quotes">https://quotes.informatika.app/api/v1/quotes</a>	bearer token
7	PUT	<b>Update Quote</b> <a href="https://quotes.informatika.app/api/v1/quotes/{quote_id}">https://quotes.informatika.app/api/v1/quotes/{quote_id}</a>	bearer token name description
8	DELETE	<b>Delete Quote</b> <a href="https://quotes.informatika.app/api/v1/quotes/{quote_id}">https://quotes.informatika.app/api/v1/quotes/{quote_id}</a>	bearer token

2. **Register**, uji coba dengan method POST sesuai dengan endpoint dan parameter. Isilah dengan data sesuai dengan pribadi masing-masing.

id	name	academic_year
1	S1IF-06-TI1	2021
2	S1IF-06-TI2	2021
3	S1IF-06-TI3	2021
4	S1IF-06-SC2	2021
5	S1IF-06-MM2	2021
6	S1IF-06-SC1	2021
7	S1IF-06-MM1	2021

POST <https://quotes.informatika.app/auth/registration> Send Save

Params Authorization Headers (9) **Body** Pre-request Script Tests Settings Cookies Code

☐ none ☒ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

KEY	VALUE	DESCRIPTION	*** Bulk Edit
<input checked="" type="checkbox"/> class_id	1		
<input checked="" type="checkbox"/> registration_number	123456		
<input checked="" type="checkbox"/> name	Novian Adi Prasetyo		
<input checked="" type="checkbox"/> email	novian@ittelkom-pwt.ac.id		
<input checked="" type="checkbox"/> password	123456		
Key	Value	Description	

### 3. Login, uji coba dengan method POST sesuai dengan endpoint dan parameter.

POST <https://quotes.informatika.app/auth/login> [Send](#)

Params Authorization Headers (9) **Body** Pre-request Script Tests Settings

☐ none ☐ form-data ☒ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> registration_number	123456	
<input checked="" type="checkbox"/> password	123456	
Key	Value	Description

Body Cookies Headers (14) Test Results [Status: 200 OK](#) [Time: 595 ms](#) [Size: 740 B](#) [Save](#)

Pretty Raw Preview Visualize **JSON** [Send](#)

```

1 {
2   "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ3ZWRhdGEiOiJlZ2ZlZDhhdG1vbi9udW1iZXIiOiJxMjM0NTYiLCJpYXQiOiJlZ2MTE3MjgwMzIsImV4cCI6MTYxMTgxNDQzMn0.WHpqeGARjeUVDsm5jGEG9Ovm1hq1LTxeKxX1h-X0k8M"
3 }
```

### 4. Create Quotes, uji coba dengan method POST sesuai dengan endpoint dan parameter.

POST <https://quotes.informatika.app/api/v1/quotes> [Send](#) [Save](#)

Params Authorization **Headers (10)** **Body** Pre-request Script Tests Settings [Cookies](#) [Code](#)

**TYPE**  
Bearer Token

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. [Learn more about variables](#)

Token

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ3ZWRhdGEiOiJlZ2ZlZDhhdG1vbi9udW1iZXIiOiJxMjM0NTYiLCJpYXQiOiJlZ2MTE3MjgwMzIsImV4cCI6MTYxMTgxNDQzMn0.WuqszJsqAaeG6QHv2oqLVH9A6dot5MPJMrOIhtyUE
```

POST <https://quotes.informatika.app/api/v1/quotes> [Send](#)

Params Authorization Headers (10) **Body** Pre-request Script Tests Settings

☐ none ☒ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> name	Kehidupan	
<input checked="" type="checkbox"/> description	Hidup tak semudah membalikkan telapak tangan, tetapi den...	
Key	Value	Description

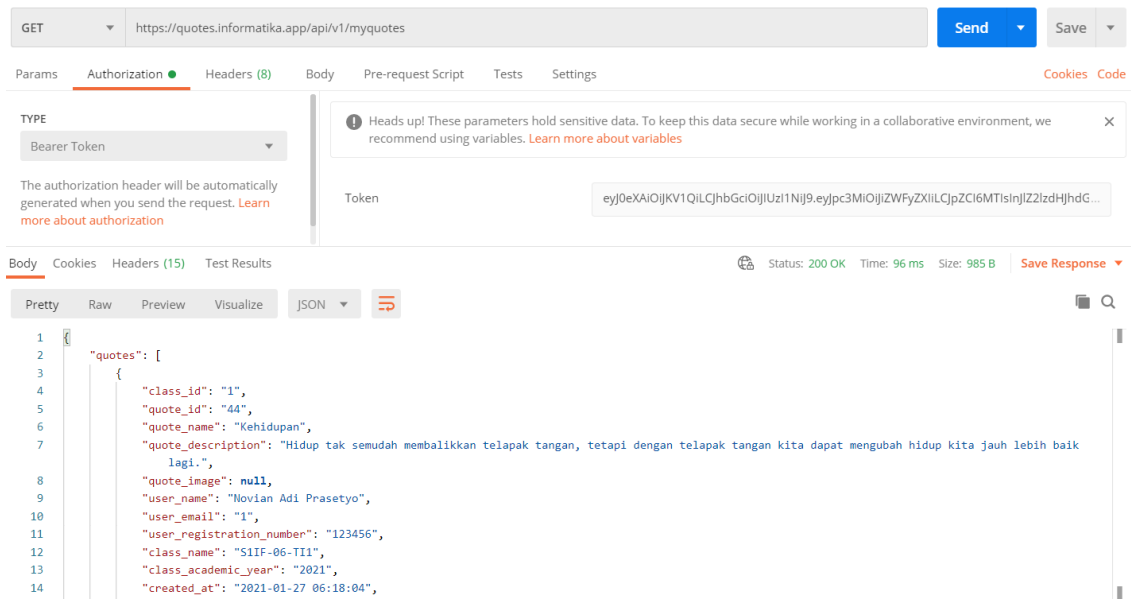
Body Cookies Headers (14) Test Results [Status: 200 OK](#) [Time: 218 ms](#) [Size: 563 B](#) [Save](#)

Pretty Raw Preview Visualize **JSON** [Send](#)

```

1 {
2   "message": "Inserted Successfully"
3 }
```

5. **My Quotes, Class Quotes & Global Quotes**, uji coba dengan method GET sesuai dengan endpoint dan parameter.



GET <https://quotes.informatika.app/api/v1/myquotes> Send Save

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies Code

TYPE: Bearer Token

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

Token: eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJZWlZlZldHJhdG...

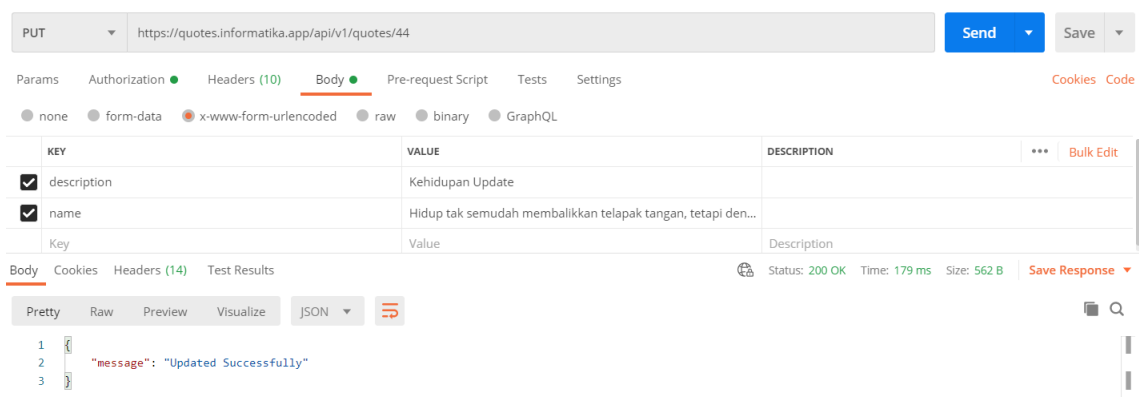
Status: 200 OK Time: 96 ms Size: 985 B Save Response

```

1 {
2   "quotes": [
3     {
4       "class_id": "1",
5       "quote_id": "44",
6       "quote_name": "Kehidupan",
7       "quote_description": "Hidup tak semudah membalikkan telapak tangan, tetapi dengan telapak tangan kita dapat mengubah hidup kita jauh lebih baik lagi.",
8       "quote_image": null,
9       "user_name": "Novian Adi Prasetyo",
10      "user_email": "1",
11      "user_registration_number": "123456",
12      "class_name": "SIIF-06-TI1",
13      "class_academic_year": "2021",
14      "created_at": "2021-01-27 06:18:04",

```

6. **Update Quotes**, uji coba dengan method PUT sesuai dengan endpoint dan parameter.



PUT <https://quotes.informatika.app/api/v1/quotes/44> Send Save

Params Authorization Headers (10) Body Pre-request Script Tests Settings Cookies Code

none form-data x-www-form-urlencoded raw binary GraphQL

KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/> description	Kehidupan Update			
<input checked="" type="checkbox"/> name	Hidup tak semudah membalikkan telapak tangan, tetapi den...			
Key	Value	Description		

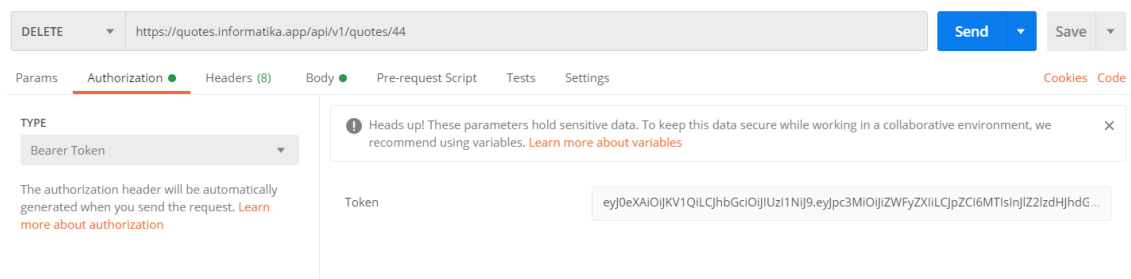
Status: 200 OK Time: 179 ms Size: 562 B Save Response

```

1 {
2   "message": "Updated Successfully"
3 }

```

7. **Delete Quotes**, uji coba dengan method DELETE sesuai dengan endpoint dan parameter.



DELETE <https://quotes.informatika.app/api/v1/quotes/44> Send Save

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies Code

TYPE: Bearer Token

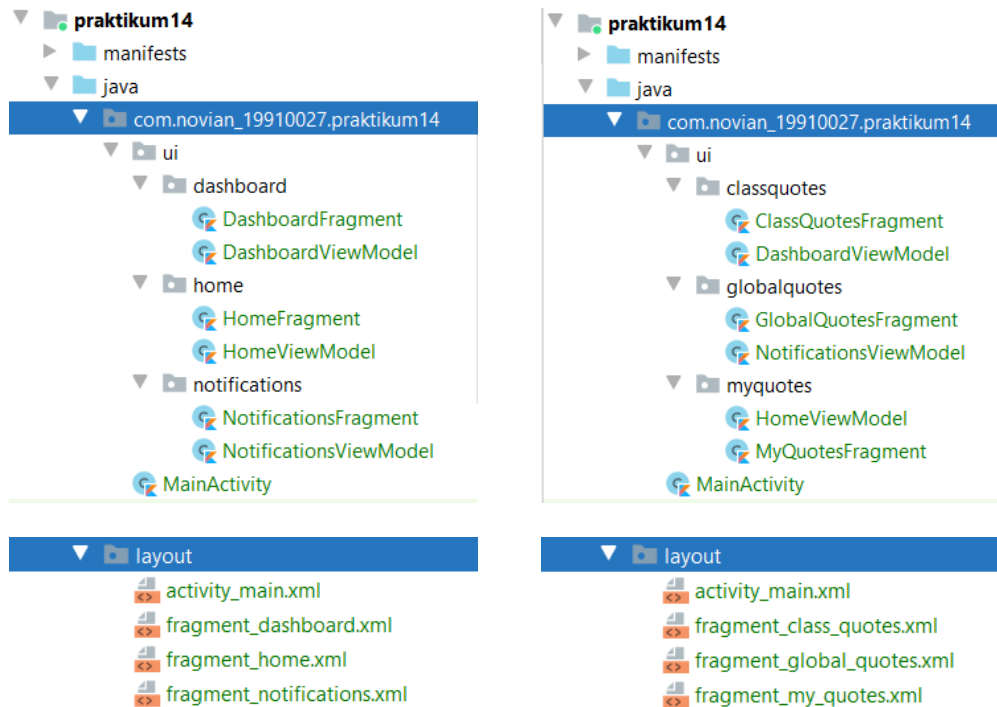
The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

Token: eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJZWlZlZldHJhdG...

Status: 200 OK

### 14.3 Persiapan UI

1. Didalam project kita saat ini terdapat MainActivity dan package ui yang didalamnya berisi 3 buah fragment. Untuk memulai kita akan sesuaikan terlebih dahulu nama masing-masing fragmentnya.
2. Lakukan rename dengan cara refactoring pada, sehingga hasilnya seperti dibawah ini.



3. Buka mobile\_navigation.xml sesuaikan kodenya menjadi seperti dibawah ini.

```

android:id="@+id/mobile_navigation"
app:startDestination="@+id/navigation_my_quotes">

<fragment
    android:id="@+id/navigation_my_quotes"
    android:name="com.novian_19910027.praktikum14.ui.myquotes.MyQuotesFragment"
    android:label="@string/title_my_quotes"
    tools:layout="@layout/fragment_my_quotes" />

<fragment
    android:id="@+id/navigation_class_quotes"

    android:name="com.novian_19910027.praktikum14.ui.classquotes.ClassQuotesFragment"
    android:label="@string/title_class_quotes"
    tools:layout="@layout/fragment_class_quotes" />

<fragment
    android:id="@+id/navigation_global_quotes"

    android:name="com.novian_19910027.praktikum14.ui.globalquotes.GlobalQuotesFragment"
    android:label="@string/title_global_quotes"
    tools:layout="@layout/fragment_global_quotes" />

```

4. Buka `bottom_nav_menu.xml` sesuaikan kodenya menjadi seperti dibawah ini.

```
<item
    android:id="@+id/navigation_my_quotes"
    android:icon="@drawable/ic_home_black_24dp"
    android:title="@string/title_my_quotes" />

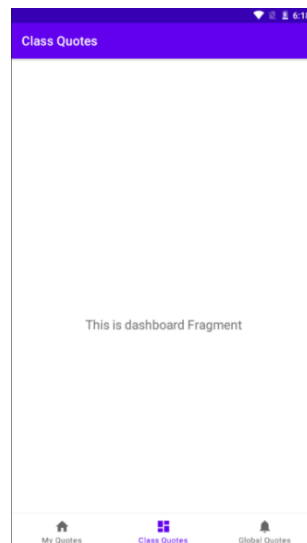
<item
    android:id="@+id/navigation_class_quotes"
    android:icon="@drawable/ic_dashboard_black_24dp"
    android:title="@string/title_class_quotes" />

<item
    android:id="@+id/navigation_global_quotes"
    android:icon="@drawable/ic_notifications_black_24dp"
    android:title="@string/title_global_quotes" />
```

5. Buka `MainActivity.kt` sesuaikan kodenya menjadi seperti dibawah ini.

```
val appBarConfiguration = AppBarConfiguration(
    setOf(
        R.id.navigation_my_quotes, R.id.navigation_class_quotes,
        R.id.navigation_global_quotes
    )
)
```

6. Jika dijalankan maka kita telah memiliki 3 halaman yang dibentuk menggunakan fragment.



7. Buat package baru dengan nama **login** didalam package **ui**, didalamnya buatlah activity baru dengan nama **LoginActivity**.
8. Halaman login kali ini kita desain seperti pada pertemuan 11. Tambahkan kode berikut [link](#) pada `activity_login.xml`.

## 14.4 Konfigurasi Token

1. Buat package baru dengan nama **model**, didalamnya buatlah class baru dengan nama Token. Tambahkan kode berikut ini.

```
@Parcelize
data class Token (
    var token: String? = null
): Parcelable
```

2. Buat class baru diluar package dengan nama **TokenPref**, pada praktikum ini kita akan mengimplementasikan shared preference untuk menyimpan token setelah login berhasil dilakukan. Tambahkan kode berikut ini pada **TokenPref**.

```
internal class TokenPref(context: Context) {
    companion object {
        private const val PREFS_NAME = "token_pref"
        private const val TOKEN = "token"
    }

    private val preferences = context.getSharedPreferences(PREFS_NAME,
Context.MODE_PRIVATE)

    fun setToken(value: Token) {
        val editor = preferences.edit()
        editor.putString(TOKEN, value.token)
        editor.apply()
    }

    fun getToken(): Token {
        val model = Token()
        model.token = preferences.getString(TOKEN, "")
        return model
    }

    fun removeToken() {
        val editor = preferences.edit()
        editor.clear()
        editor.apply()
    }
}
```

3. Buka MainActivity lalu tambahkan kode berikut agar pada aplikasi terdapat kondisi jika token belum tersimpan maka halaman yang dibuka adalah halaman login.

```
class MainActivity : AppCompatActivity() {
    private lateinit var tokenPref: TokenPref
    private lateinit var token: Token
    override fun onCreate(savedInstanceState: Bundle?) {

        ...
        ...

        navView.setupWithNavController(navController)
        tokenPref = TokenPref(this)
        token = tokenPref.getToken()
        if (TextUtils.isEmpty(token.token)) {
            val intent = Intent(this@MainActivity, LoginActivity::class.java)
            startActivity(intent)
            finish()
        }
    }
}
```

4. Buka manifest, tambahkan permission internet.

```
<uses-permission android:name="android.permission.INTERNET" />
```

## 14.5 Konfigurasi API Client

1. Buka gradle lalu tambahkan depedensi berikut ini.

```
implementation 'com.google.code.gson:gson:2.8.5'
implementation 'com.squareup.retrofit2:retrofit:2.4.0'
implementation 'com.squareup.retrofit2:converter-gson:2.4.0'
implementation 'com.squareup.okhttp3:okhttp:3.10.0'
implementation 'com.squareup.okhttp3:logging-interceptor:3.10.0'
implementation "org.jetbrains.anko:anko:0.10.8"
```

2. Buat class baru dengan nama **Login** didalam package **model**, lalu tambahkan kode berikut ini.

```
data class Login(
    @SerializedName("token")
    var token: String? = null,
    @SerializedName("message")
    var message: String? = null
)
```

Pada model kali ini memiliki perbedaan dari model sebelumnya yaitu terdapat anotasi **@SerializedName**, anotasi tersebut digunakan untuk memasukan **value** yang diterima dari JSON sesuai dengan **name** pada anotasinya.

3. Buat class baru dengan nama **Message** didalam package **model**, lalu tambahkan kode berikut ini.

```
data class Message(
    @SerializedName("message")
    var message: String? = null
)
```

4. Buat class baru dengan nama **Quote** didalam package **model**, lalu tambahkan kode berikut ini.

```
@Parcelize
data class Quote(
    @SerializedName("quote_id")
    var quote_id: String? = null,
    @SerializedName("user_name")
    var user_name: String? = null,
    @SerializedName("class_name")
    var class_name: String? = null,
    @SerializedName("quote_name")
    var quote_name: String? = null,
    @SerializedName("quote_description")
    var quote_description: String? = null,
    @SerializedName("created_at")
    var created_at: String? = null
) : Parcelable
```

5. Buat class baru dengan nama **QuoteResponse** didalam package **model**, lalu

```
data class QuoteResponse(
    val quotes: ArrayList<Quote>
)
```

tambahkan kode berikut ini.

6. Buat sebuah package baru dengan nama **interface**, lalu buat class baru didalamnya dengan nama **MainView**. Tambahkan kode berikut ini.

```
interface MainView {
    fun showMessage(message : String)
    fun resultQuote(data: ArrayList<Quote>)
    fun resultLogin(data: Login)
}
```



7. Buat class baru dengan nama **CoroutineContextProvider**, lalu tambahkan kode berikut ini.

```
open class CoroutineContextProvider {  
    open val main: CoroutineContext by lazy { Dispatchers.Main }  
}
```

Kode ini digunakan untuk menjalankan suatu proses agar berjalan secara asynchronous, sehingga tidak terlihat proses yang menunggu waktu lama di depan layar.

8. Buka gradle lalu tambahkan kode berikut ini untuk menambahkan url utama dari API.

```
defaultConfig {  
    applicationId "com.novian 19910027.praktikum14"  
    minSdkVersion 21  
    targetSdkVersion 30  
    versionCode 1  
    versionName "1.0"  
    buildConfigField "String", "BASE_URL", "\"https://quotes.informatika.app/\""  
    testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"  
}
```

9. Buat package baru dengan nama **api**, lalu buat class baru di dalamnya dengan nama **ApiServices**, tambahkan kode berikut ini.

```
interface ApiService {  
    @FormUrlEncoded  
    @POST("auth/login")  
    fun login(  
        @Field("registration_number") nim: String,  
        @Field("password") password: String  
    ): Call<Login>  
  
    @GET("api/v1/myquotes")  
    fun getMyQuotes(  
        @Header("Authorization") token:String?  
    ): Call<QuoteResponse>  
  
    @GET("api/v1/class_quotes")  
    fun getClassQuotes(  
        @Header("Authorization") token:String?  
    ): Call<QuoteResponse>  
  
    @GET("api/v1/quotes")  
    fun getAllQuotes(  
        @Header("Authorization") token:String?  
    ): Call<QuoteResponse>  
  
    @FormUrlEncoded  
    @POST("api/v1/quotes")  
    fun addQuote(  
        @Header("Authorization") token:String,  
        @Field("name") name: String,  
        @Field("description") description: String  
    ): Call<Message>  
  
    @FormUrlEncoded  
    @PUT("api/v1/quotes/{quote_id}")  
    fun updateQuote(  
        @Header("Authorization") token:String,  
        @Path("quote_id") quote_id: String,  
        @Field("name") title: String,  
        @Field("description") description: String  
    ): Call<Message>  
  
    @DELETE("api/v1/quotes/{quote id}")  
    fun deleteQuote(  
        @Header("Authorization") token:String,  
        @Path("quote id") quote id: String  
    ): Call<Message>  
}
```

Jika dilihat pada kode diatas, setiap fungsinya sesuai dengan endpoint yang ada di langkah 14.2.1. Fungsi login menggunakan endpoint **auth/login**, method **POST** dan parameter **registration\_number & password**. Begitu juga untuk semua fungsi di atas dibuat sesuai dengan endpoint yang telah disiapkan.

10. Buat class baru dengan nama ApiMain di dalam package api, lalu tambahkan kode berikut ini. Kode ini digunakan untuk melakukan request ke server.

```
class ApiMain : Application() {
    private val client = OkHttpClient().newBuilder()
        .addInterceptor(HttpLoggingInterceptor().apply {
            level = if (BuildConfig.DEBUG) HttpLoggingInterceptor.Level.BODY else
            HttpLoggingInterceptor.Level.NONE
        })
        .readTimeout(30, TimeUnit.SECONDS)
        .writeTimeout(30, TimeUnit.SECONDS)
        .build()

    private val retrofit = Retrofit.Builder()
        .baseUrl(BuildConfig.BASE_URL)
        .client(client)
        .addConverterFactory(GsonConverterFactory.create())
        .build()

    val services: ApiService = retrofit.create(ApiServices::class.java)
}
```

11. Buat class baru dengan nama MainPresenter di dalam package api, lalu tambahkan kode berikut ini.

```
class MainPresenter(private val view: MainView, private val context:
    CoroutineContextProvider = CoroutineContextProvider()) {

}
```

MainPresenter akan bekerja sebagai penghubung antara activity agar dapat mengakses ApiService, fungsi yang akan dipanggil adalah fungsi dari MainPresenter lalu selanjutnya MainPresenter akan mengarahkan ke ApiService sesuai yang dibutuhkan oleh activity.

12. Masih pada MainPresenter, tambahkan kode berikut ini untuk proses getMyQuotes.

```
fun getMyQuotes(token:String?) {
    GlobalScope.launch (context.main){
        try {
            ApiMain().services.getMyQuotes("Bearer "+token).enqueue(object :
                Callback<QuoteResponse> {
                    override fun onResponse(call: Call<QuoteResponse>, response:
                    Response<QuoteResponse>) {
                        if(response.code() == 200) {
                            response.body()?.quotes?.let {
                                view.resultQuote(it)
                            }
                        }
                    }
                    override fun onFailure(call: Call<QuoteResponse>, t: Throwable) {
                        view.showMessage("Koneksi Terputus")
                    }
                })
        } catch (e:Exception) {
        }
    }
}
```

Selanjutnya untuk getClassQuotes, getAllQuotes, addQuote, updateQuote, deleteQuote dan login dapat di ambil kodenya pada [tautan ini](#).

## 14.6 Login

1. Buka LoginActivity lalu tambahkan kode berikut ini.

```
class LoginActivity : AppCompatActivity(), View.OnClickListener, MainView {
    private lateinit var presenter: MainPresenter
    private lateinit var binding: ActivityLoginBinding
    private lateinit var tokenPref: TokenPref
    private lateinit var token: Token
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityLoginBinding.inflate(layoutInflater)
        setContentView(binding.root)
        binding.btnSign.setOnClickListener(this)
        presenter = MainPresenter(this, CoroutineContextProvider())
        tokenPref = TokenPref(this)
        token = tokenPref.getToken()
    }
}
```

Pada kode diatas kita lihat bahwa class melakukan implement berupa MainView, MainView sebelum sudah pernah kita buat pada package interface. Karena kita mengimplementasikan MainView maka kita perlu memanggil juga fungsi yang ada didalamnya yaitu showMessage, resultQuote dan resultLogin. Tidak lupa juga kita panggil onClick karena kita mengimplementasikan View.OnClickListener.

2. Selanjut proses login akan bekerja ketika tombol sign di tekan, mari kita tambahkan pada onclicknya.

```
override fun onClick(v: View) {
    when (v.id) {
        R.id.btnSign -> {
            presenter.login(
                binding.inputNim.text.toString(),
                binding.inputPassword.text.toString()
            )
        }
    }
}
```

Tombol ini akan mengarahkan untuk menjalankan fungsi login yang ada di MainPresenter dengan membawa nim dan password. Jika kita lihat di dalam fungsi login pada MainPresenter setelah mendapat respon, maka akan dijalankan fungsi resultLogin dan showMessage yang ada di LoginActivity.

3. Masih pada LoginActivity, tambahkan kode berikut ini.

```
override fun showMessage(message: String) {
    Toast.makeText(this, message, Toast.LENGTH_SHORT).show()
}

override fun resultQuote(data: ArrayList<Quote>) {
}

override fun resultLogin(data: Login) {
    if (!TextUtils.isEmpty(data.token)) {
        token.token = data.token
        tokenPref.setToken(token)
        val intent = Intent(this@LoginActivity, MainActivity::class.java)
        startActivity(intent)
        finish()
    }
}
```

Pada resultLogin kita lihat jika token tersedia, maka token akan disimpan pada shared preference. Sampai tahap ini proses login sudah berhasil dibuat.

4. Buat **Menu Resource File** baru dengan nama **menu\_form.xml**, lalu tambahkan kode berikut ini.

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">
    <item
        android:id="@+id/action_logout"
        android:icon="@drawable/ic_baseline_exit_to_app_24"
        android:title="@string/logout"
        app:showAsAction="always" />
    </menu>
```

5. Buka **MainActivity**, lalu tambahkan kode berikut ini.

```
override fun onCreateOptionsMenu(menu: Menu): Boolean {
    menuInflater.inflate(R.menu.topbar_menu, menu)
    return super.onCreateOptionsMenu(menu)
}
override fun onOptionsItemSelected(item: MenuItem): Boolean {
    when (item.itemId) {
        R.id.action_logout -> {
            tokenPref.removeToken()
            val intent = Intent(this@MainActivity, LoginActivity::class.java)
            startActivity(intent)
            finish()
        }
    }
    return super.onOptionsItemSelected(item)
}
```

Sampai tahap ini proses logout sudah selesai dibuat.

## 14.7 Read

1. Buat sebuah adapter dengan nama **QuoteAdapter**, isi dari adapter ini sama seperti dengan praktikum sebelumnya. Tambahkan kode berikut ini [link](#).  
Buat sebuah class dengan nama **helper**, isi dari class ini sama seperti dengan praktikum sebelumnya. Tambahkan kode berikut ini [link](#).
2. Buat sebuah layout dengan nama **item\_quote**, isi dari layout ini sama seperti dengan praktikum sebelumnya. Tambahkan kode berikut ini [link](#).
3. Buka **fragment\_my\_quotes.xml**, lalu tambahkan kode berikut ini [link](#). Isi dari layout ini sama seperti dengan praktikum sebelumnya.
4. Buka **fragment\_class\_quotes.xml**, lalu tambahkan kode berikut ini [link](#). Isi dari layout ini sama seperti dengan praktikum sebelumnya.
5. Buka **fragment\_global\_quotes.xml**, lalu tambahkan kode berikut ini [link](#). Isi dari layout ini sama seperti dengan praktikum sebelumnya.
6. Buka **MyQuotesFragment.kt**, lalu ubah kodenya seperti dibawah ini.

```
class MyQuotesFragment : Fragment(), MainView {

    private lateinit var presenter: MainPresenter
    private var quotes: MutableList<Quote> = mutableListOf()
    private lateinit var adapter: QuoteAdapter
    private lateinit var tokenPref: TokenPref
    private lateinit var token: Token
    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?, savedInstanceState:
        Bundle?
    ): View? = inflater.inflate(R.layout.fragment_my_quotes, container, false)
}
```

7. Masih pada **MyQuotesFragment.kt** tambahkan **onViewCreated**.

```

override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)
    val binding = FragmentMyQuotesBinding.bind(view)
    binding.recyclerviewMyQuotes.layoutManager = LinearLayoutManager(activity)
    tokenPref = TokenPref(requireActivity())
    token = tokenPref.getToken()
    adapter = QuoteAdapter(requireActivity())
    binding.recyclerviewMyQuotes.adapter = adapter
    presenter =
        MainPresenter(this, CoroutineContextProvider())
    progressbar.visibility = View.VISIBLE
    presenter.getMyQuotes(token.token)
    swipeRefreshLayout.onRefresh {
        progressbar.visibility = View.INVISIBLE
        presenter.getMyQuotes(token.token)
    }
}
override fun onResume() {
    super.onResume()
    presenter.getMyQuotes(token.token)
}

```

Pada kode diatas kita akan memanggil **presenter.getMyQuotes(token.token)** yang berfungsi untuk memanggil my quotes menggunakan token yang sudah tersimpan.

8. Masih pada **MyQuotesFragment.kt**, kita implementasikan member dari **MainView**.

```

override fun showMessage(message: String) {
    Toast.makeText(requireActivity(), message, Toast.LENGTH_SHORT).show()
}

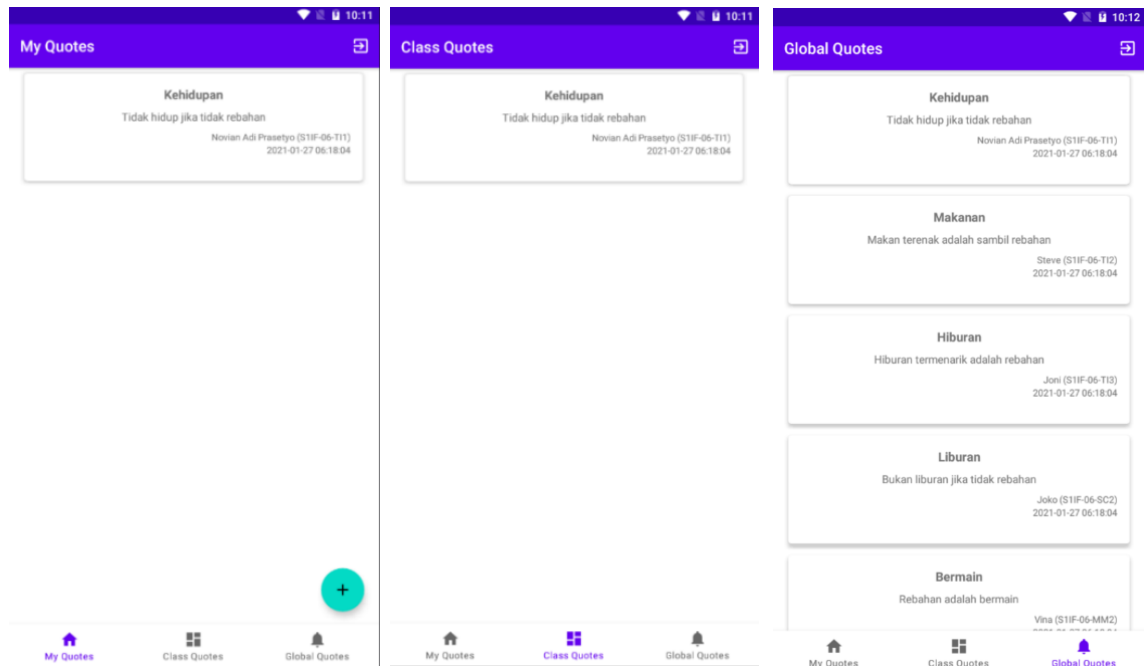
override fun resultQuote(data: ArrayList<Quote>) {
    quotes.clear()
    adapter.listQuotes = data
    quotes.addAll(data)
    adapter.notifyDataSetChanged()
    progressbar.visibility = View.INVISIBLE
    swipeRefreshLayout.isRefreshing = false
}

override fun resultLogin(data: Login) {
}

```

9. Ulangi langkah 6 sampai 8 pada **ClassQuotesFragment**, namun layout yang digunakan adalah **fragment\_class\_quotes**, binding yang digunakan adalah **FragmentClassQuotesBinding**, recyclerview yang digunakan adalah **recyclerviewClassQuotes**, presenter yang digunakan adalah **presenter.getClassQuotes**.
10. Ulangi langkah 6 sampai 8 pada **GlobalQuotesFragment**, namun layout yang digunakan adalah **fragment\_global\_quotes**, binding yang digunakan adalah **FragmentGlobalQuotesBinding**, recyclerview yang digunakan adalah **recyclerviewGlobalQuotes**, presenter yang digunakan adalah **presenter.getAllQuotes**.

- Sampai tahap ini aplikasi telah berhasil membaca data untuk my quote, class quote dan global quote.



## 14.8 Create

- Buat activity baru dengan nama **QuoteAddUpdateActivity**, activity ini sama seperti pada praktikum sebelumnya. Tambahkan kode berikut ini sebagai kode starter [link](#).
- Buka layout **activity\_quote\_add\_update**, layout ini sama seperti pada praktikum sebelumnya. Tambahkan kode berikut ini [link](#).
- Buka **MyQuoteFragment**, lalu tambahkan kode berikut ini agar tombol fab dapat berfungsi membuka activity add data.

```
presenter.getMyQuotes(token.token)
binding.fab.setOnClickListener {
    val intent = Intent(requireActivity(),
        QuoteAddUpdateActivity::class.java)
    startActivityForResult(intent, REQUEST_ADD)
}
swiperefresh.onRefresh {
    progressbar.visibility = View.INVISIBLE
}
```

- Buka **QuoteAddUpdateActivity**, tambahkan kode berikut ini untuk mengakses presenter dan mengakses presenter.addQuotes.

```
private lateinit var token: Token
private lateinit var presenter: MainPresenter
override fun onCreate(savedInstanceState: Bundle?) {
```

```
    binding = ActivityQuoteAddUpdateBinding.inflate(layoutInflater)
    setContentView(binding.root)
    presenter = MainPresenter(this, CoroutineContextProvider())
    tokenPref = TokenPref(this)
```

```

if (isEdit) {
} else {
    presenter.addQuote(
        token.token.toString(),
        binding.edtTitle.text.toString(),
        binding.edtDescription.text.toString()
    )
}
}

```

5. Tambahkan MainView dan impelentasikan, seperti kode dibawah ini.

```

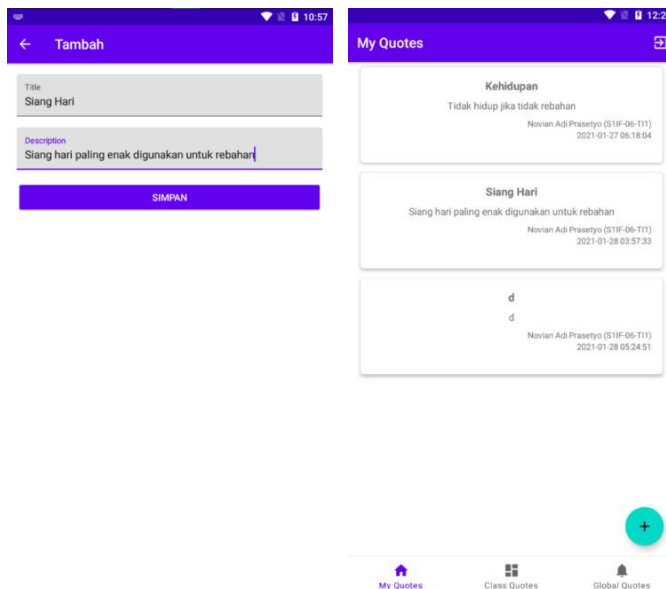
override fun showMessage(message: String) {
    Toast.makeText(this,message, Toast.LENGTH_SHORT).show()
    finish()
}

override fun resultQuote(data: ArrayList<Quote>) {
}

override fun resultLogin(data: Login) {
}

```

6. Sampai tahap ini proses tambah data sudah dapat di uji coba.



## 14.9 Update

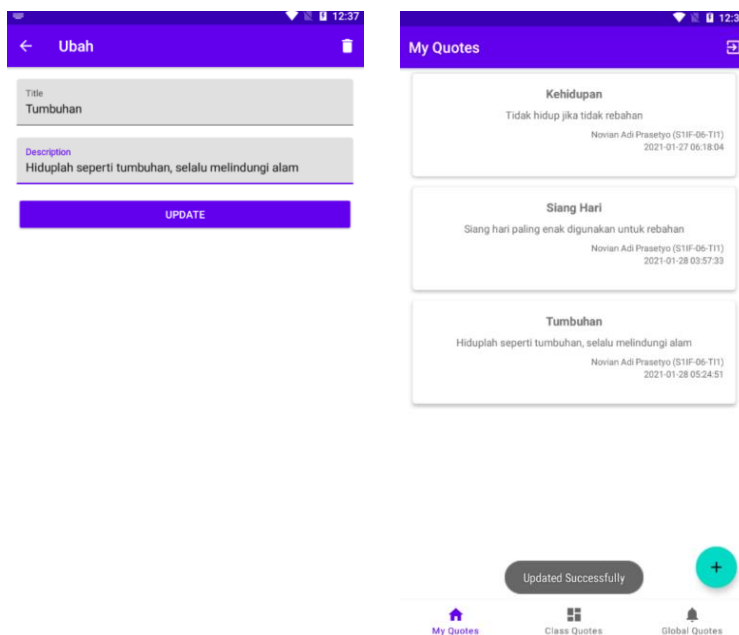
1. Buka **QuoteAdapter**, lalu tambahkan kode berikut ini.

```
binding.tvItemDescription.text = quote.quote_description
binding.cvItemQuote.setOnClickListener{
    val intent = Intent(activity,
QuoteAddUpdateActivity::class.java)
    intent.putExtra(EXTRA_POSITION, position)
    intent.putExtra(EXTRA_QUOTE, quote)
    activity.startActivityForResult(intent, REQUEST_UPDATE)
}
}
```

2. Buka **QuoteAddUpdateActivity**, tambahkan kode berikut ini untuk menambahkan fungsi update presenter.updateQuotes.

```
if (isEdit) {
    presenter.updateQuote(
        token.token.toString(),
        quote!!.quote_id.toString(),
        binding.edtTitle.text.toString(),
        binding.edtDescription.text.toString()
    )
} else {
```

3. Jika di uji coba hasilnya seperti dibawah ini.



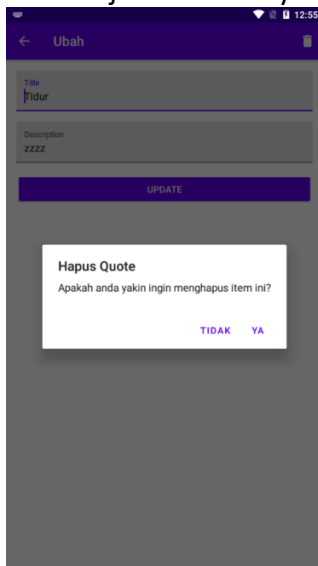


### 14.9 Update

1. Buka **QuoteAddUpdateActivity**, lalu tambahkan kode berikut ini untuk memberikan aksi kepada tombol delete.

```
.setPositiveButton("Ya") { _, _ ->
    if (isDialogClose) {
        finish()
    } else {
        presenter.deleteQuote(token.token.toString(), quote?.quote_id.toString())
    }
}
```

2. Jika di uji coba hasilnya seperti dibawah ini.



### 14.10 TUGAS

1. Selesaikan modul praktikum.
2. Pastikan tidak ada terjadi error pada aplikasi.
3. Build aplikasi menjadi .apk.
4. Upload file .apk ke lms, deadline dapat dilihat pada lms.
5. Kriteria Penilaian :
  - a. Melanjutkan project pada pertemuan kemarin, lakukan commit & push pada github (30)
  - b. Semua fitur pada module praktikum dapat dijalankan dengan baik (55)
  - c. Improvisasi (15)
  - d. Keterlambatan pengurangan nilai (20)