VERSION 1.0

AUGUST, 2023

PEMROGRAMAN MOBILE

APPWRITE, SERVERLESS BACKEND SERVICE – MODUL 5 MATERI

TIM PENYUSUN:

- DIDIH RIZKI CHANDRANEGARA, S.KOM., M.KOM.
- MUHAMMAD ZULFIQOR LILHAQ
- RIYAN PUTRA FIRJATULLAH

PRESENTED BY: LAB. INFORMATIKA UNIVERSITAS MUHAMMADIYAH MALANG

CAPAIAN PEMBELAJARAN PRAKTIKUM

1. Mahasiswa dapat memahami penggunaan Appwrite pada Framework Flutter.
2. Mahasiswa dapat memahami penggunaan Appwrite Service pada aplikasi Framework Flutter..
3. Mahasiswa dapat memahami penggunaan Serverless pada aplikasi Framework Flutter.

SUB CAPAIAN PEMBELAJARAN PRAKTIKUM

1. Mahasiswa dapat mengimplementasikan Appwrite Service pada aplikasi menggunakan Framework Flutter.
2. Mahasiswa dapat mengimplementasikan Serverless pada aplikasi menggunakan Framework Flutter.

KEBUTUHAN HARDWARE & SOFTWARE

1. PC/Laptop
2. IDE Android Studio/ Visual Studio Code
3. Flutter SDK: https://docs.flutter.dev/release/archive?tab=windows

MATERI POKOK

Appwrite

Adalah secure open source backend yang dapat menghandle tugas-tugas complex serta berulang untuk sebuah modern app appwrite.

Selain support REST, GraphQL, Realtime dan Oauth Appwrite juga memiliki banyak service yang dapat digunakan secara gratis :

1. Account
2. Users
3. Teams
4. Databases
5. Storage
6. Functions
7. Dan lain-lain

Untuk belajar terdapat dua cara dalam menggunakan appwrite, yang pertama dengan menginstallnya di local dan yang kedua dengan menggunakan cloud appwrite.

Instalasi

Local

1. Untuk dapat menginstall appwrite di local, OS harus sudah terdapat docker terlebih dahulu appwrite self-hosting docker install.

2. Untuk linux jalankan command berikut

```
docker run -it --rm \
   --volume /var/run/docker.sock:/var/run/docker.sock \
   --volume "$(pwd)"/appwrite:/usr/src/code/appwrite:rw \
   --entrypoint="install" \
   appwrite/appwrite:1.3.8
```

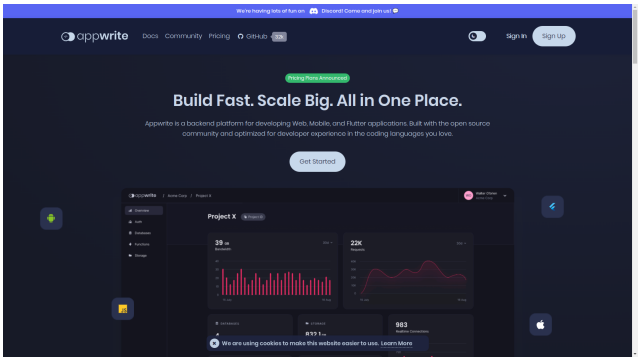
3. Untuk Windows jalankan pada cmd, command berikut

```
docker run -it --rm ^
   --volume //var/run/docker.sock:/var/run/docker.sock ^
   --volume "%cd%"/appwrite:/usr/src/code/appwrite:rw ^
   --entrypoint="install" ^
   appwrite/appwrite:1.3.8
```

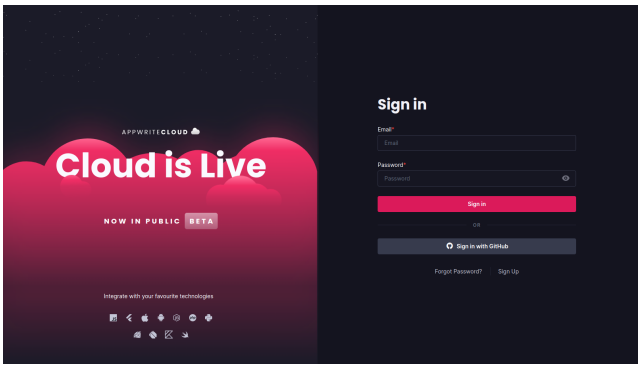
4. Setelah proses instalasi selesai
5. Appwrite Local sudah dapat dijalankan pada http://localhost:8080
   Bila terdapat error / tidak dapat diakses bisa jadi karena port 8080 sudah digunakan service lain.
   Solusi: tambahkan command custom port pada docker run
6. Lanjut ke Penggunaan Appwrite

Cloud

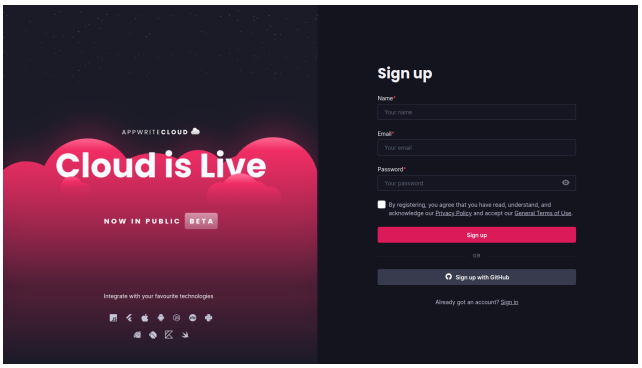
1. Buka situs appwrite.io dan click Get Started



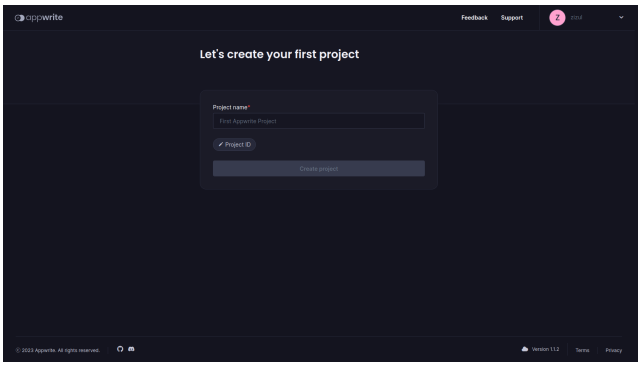
2. Sign In bila sudah punya account atau Sing Up bila belum.



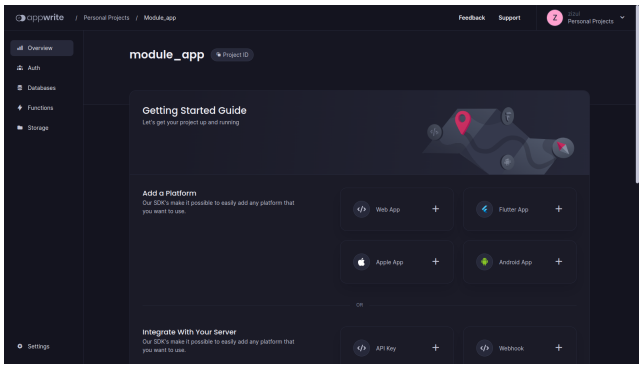
3. Isi sesuai arahan lalu Sign Up



4. Tambahkan sebuah project awal (formalitas)

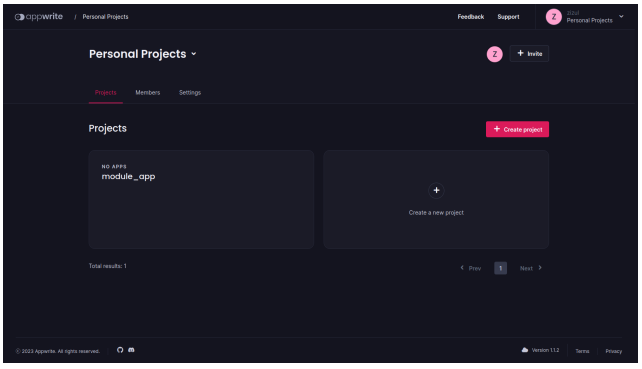


3. Appwrite Cloud siap digunakan [Penggunaan]

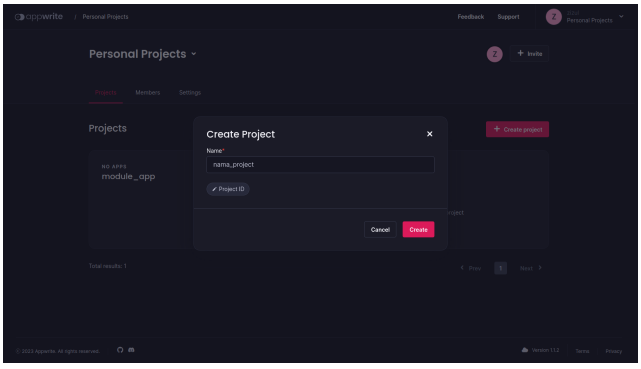


Penggunaan

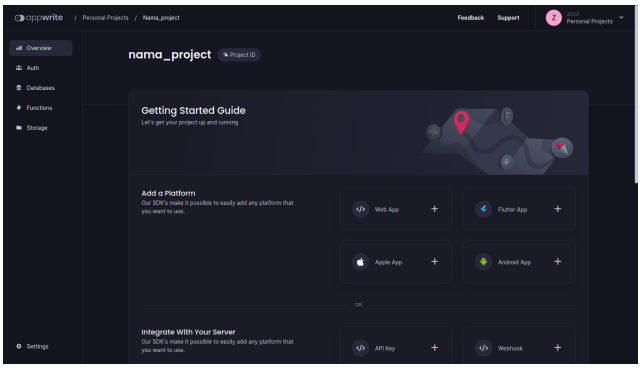
1. Buat atau gunakan sebuah project



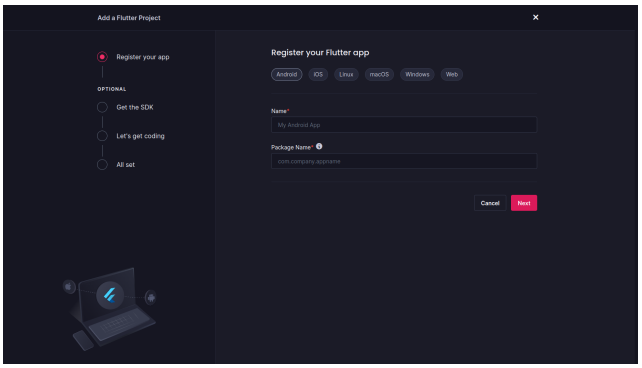
2. Sesuaikan dengan nama project



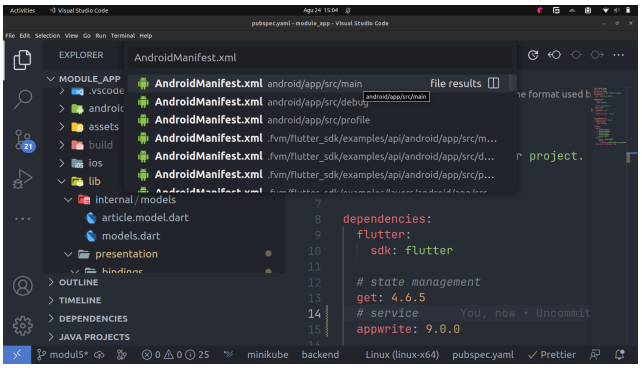
3. Click Flutter App



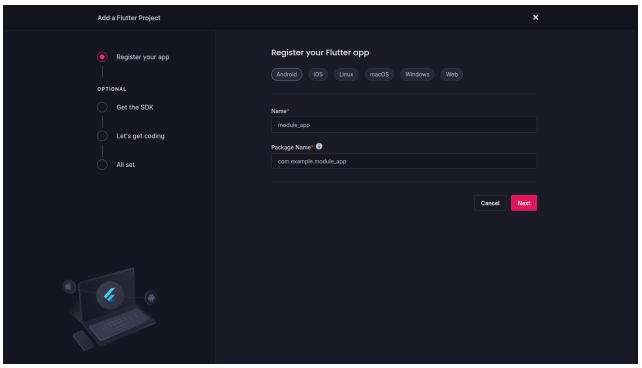
4. Masukan nama project

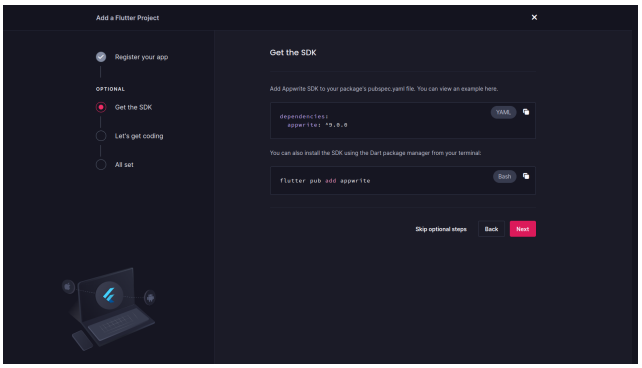


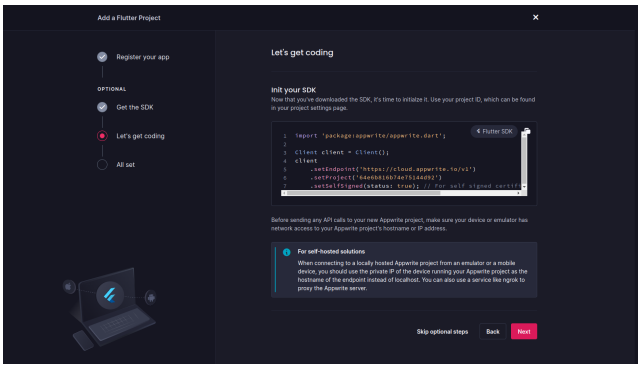
5. Temukan package name pada android/app/src/main/AndroidManifest.xml



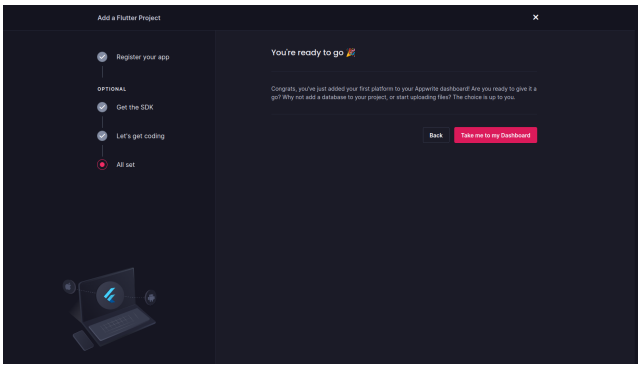
6. Isi dengan data yang sesuai

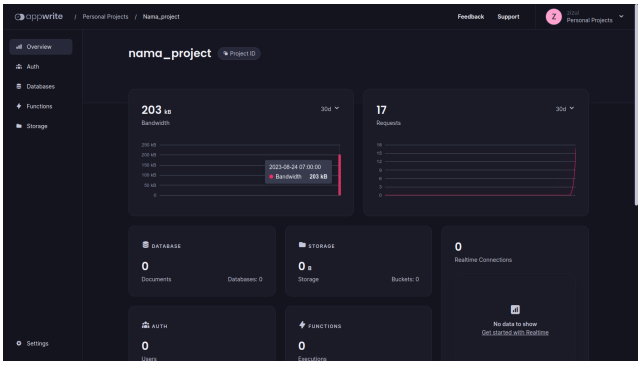






7. Selanjutnya service appwrite sudah dapat digunakan, beberapa penggunaannya akan dibahas pada modul ini





Client

Service yang pertama kali dipakai untuk membuat session pada aplikasi client agar dapat terhubung dengan sebuah project appwrite. Dengan begitu perizinan API dapat lebih secure untuk dipakai user [appwrite client](#).

1. Pada contoh ini penggunaan dibuat sesederhana mungkin

```
class ClientController extends GetxController {
  Client client = Client();

  @override
  void onInit() {
    super.onInit();

    // appwrite
    const endPoint = "ENDPOINT_APPWRITE";
    const projectID = "PROJECT_APPWRITE";
    client.setEndpoint(endPoint).setProject(projectID).setSelfSigned(status: true);
  }
}
```

2. Buka setting project Appwrite

3. Copy dan Paste Endpoint

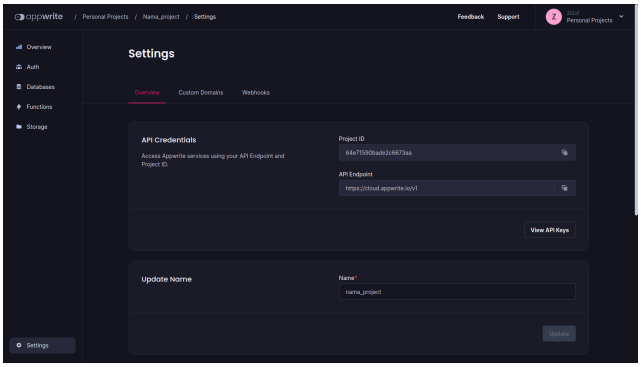
```
const endPoint = "https://cloud.appwrite.io/v1";
```

4. Khusus Local Appwrite gunakan

```
const endPoint = "http://10.0.2.2:8080/v1";
```

5. Copy dan Paste Project ID

```
const projectID = "64e71590bade2c6673aa";
```



Account

Service yang dipakai untuk authenticate dan mengatur sebuah user account. Account dapat dipakai untuk update informasi user, mengambil session user antar device yang berbeda, dan melihat log security user secara uptodate [appwrite account].

1. Pada contoh ini penggunaan dibuat sesederhana mungkin

```
class AccountController extends ClientController {
 Account? account;

 @override
 void onInit() {
  super.onInit();

  // appwrite
  account = Account(client);
 }

 Future createAccount(Map map) async {
  try {
   final result = await account!.create(
    userId: map['userId'],
    email: map['email'],
    password: map['password'],
    name: map['name'],
   );
   print("AccountController:: createAccount $result");
  } catch (error) {
   Get.defaultDialog(
    title: "Error Account",
    titlePadding: const EdgeInsets.only(top: 15, bottom: 5),
    titleStyle: Get.context?.theme.textTheme.titleLarge,
    content: Text(
     "$error",
     style: Get.context?.theme.textTheme.bodyMedium,
     textAlign: TextAlign.center,
    ),
    contentPadding: const EdgeInsets.only(top: 5, left: 15, right: 15),
   );
  }
 }

 Future createEmailSession(Map map) async {
```

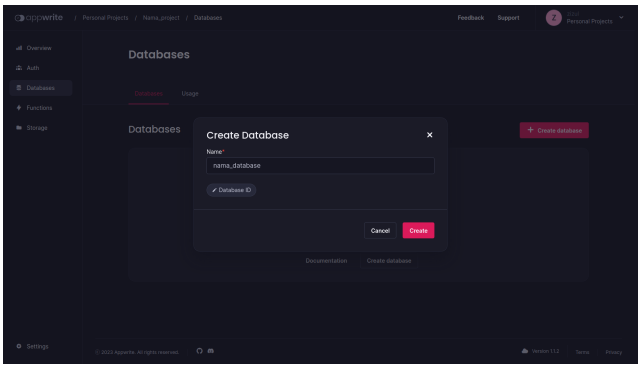
```
    try {
      final result = await account!.createEmailSession(
        email: map['email'],
        password: map['password'],
      );
      print("AccountController:: createEmailSession $result");
    } catch (error) {
      Get.defaultDialog(
        title: "Error Account",
        titlePadding: const EdgeInsets.only(top: 15, bottom: 5),
        titleStyle: Get.context?.theme.textTheme.titleLarge,
        content: Text(
          "$error",
          style: Get.context?.theme.textTheme.bodyMedium,
          textAlign: TextAlign.center,
        ),
        contentPadding: const EdgeInsets.only(top: 5, left: 15, right: 15),
      );
    }
  }
}
```

2. Silahkan baca dokumentasi untuk case account yang lain [appwrite account]

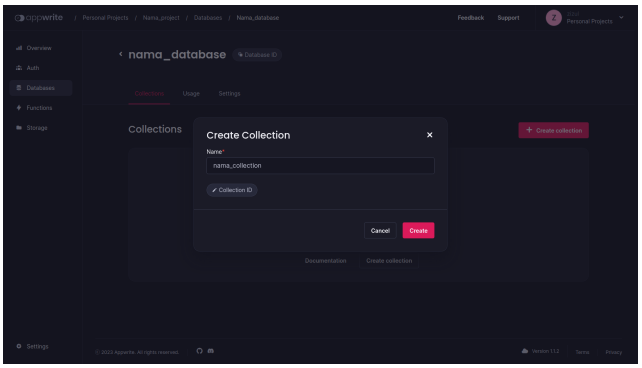
Databases

Service untuk membuat kumpulan dokumen terstruktur yang dapat diatur read dan write access permissionsnya secara advance [appwrite databases](#).

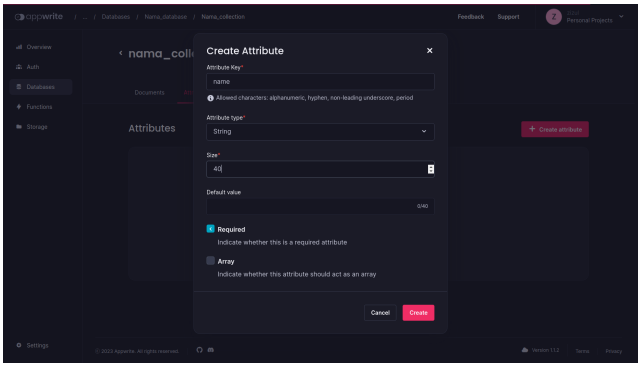
3. Buat atau gunakan sebuah database

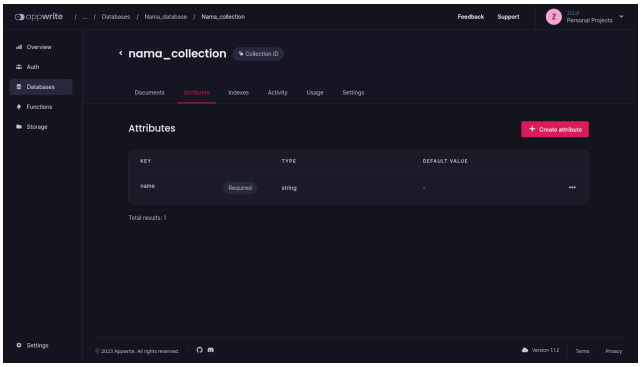


4. Buat atau gunakan collection

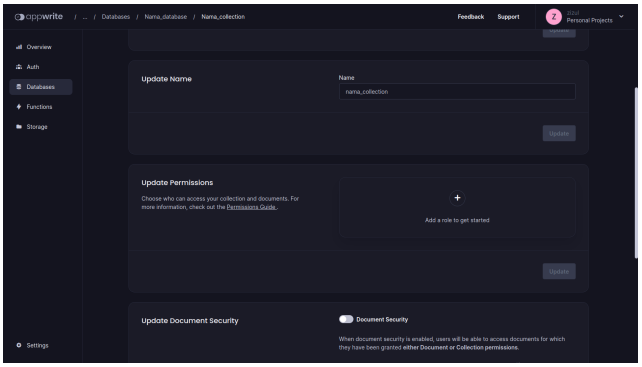


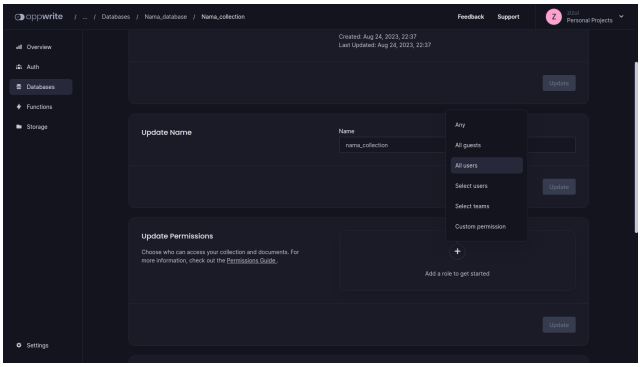
5. Buat query sesuai yang diinginkan

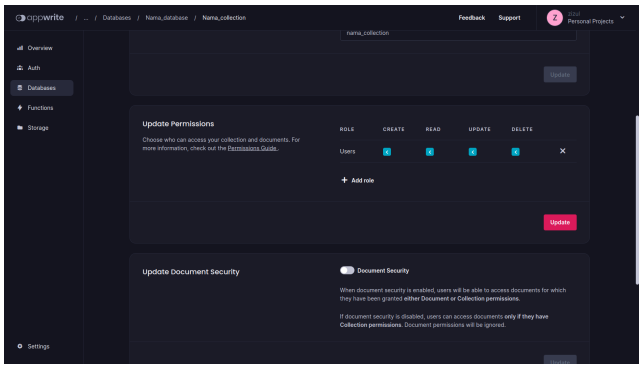




6. Update permission pada setting collection





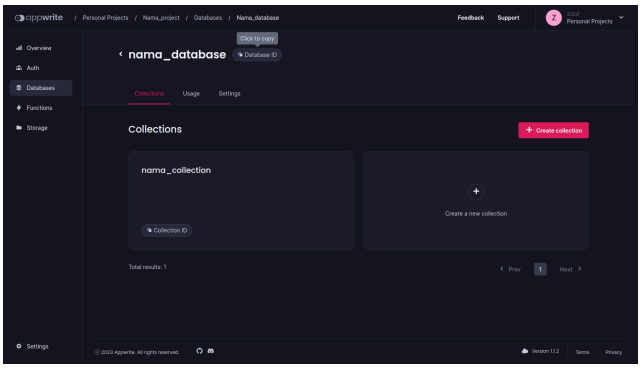


7. Pada contoh ini penggunaan dibuat sesederhana mungkin

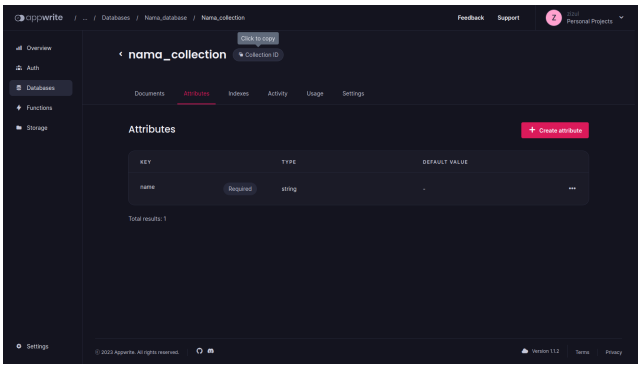
```
class DatabaseController extends ClientController {
 Databases? databases;

 @override
 void onInit() {
  super.onInit();

  // appwrite
  databases = Databases(client);
 }

 Future storeUserName(Map map) async {
  try {
   final result = await databases!.createDocument(
    databaseId: "YOUR_DATABASE_ID",
    documentId: ID.unique(),
    collectionId: "YOUR_COLLECTION_ID",
    data: map,
    permissions: [
     Permission.read(Role.user("USERID")),
     Permission.update(Role.user("USERID")),
     Permission.delete(Role.user("USERID")),
    ],
   );
   print("DatabaseController:: storeUserName $database");
  } catch (error) {
   Get.defaultDialog(
    title: "Error Database",
    titlePadding: const EdgeInsets.only(top: 15, bottom: 5),
    titleStyle: Get.context?.theme.textTheme.titleLarge,
    content: Text(
```

```
        "$error",
        style: Get.context?.theme.textTheme.bodyMedium,
        textAlign: TextAlign.center,
      ),
      contentPadding: const EdgeInsets.only(top: 5, left: 15, right: 15),
    );
  }
 }
}
```

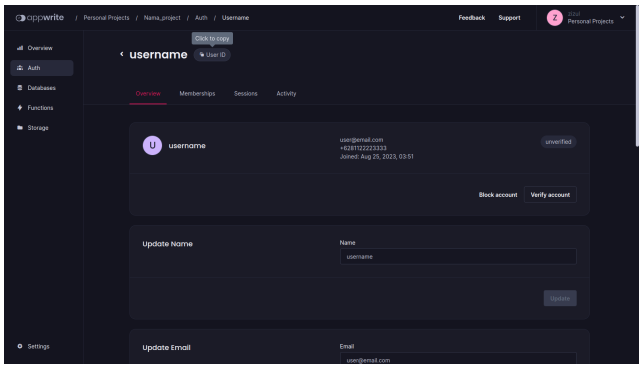
8. Copy Paste Database ID



9. Copy Paste Collection ID



10. Copy Paste User ID

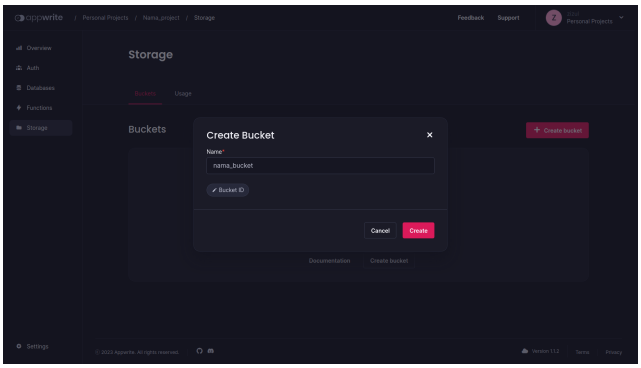


11. Silahkan baca dokumentasi untuk case database yang lain [appwrite databases]

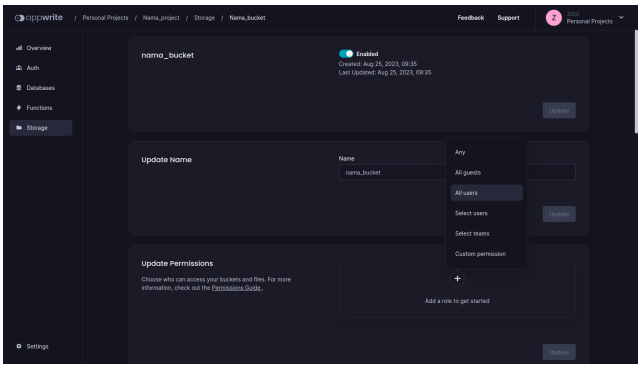
Storage

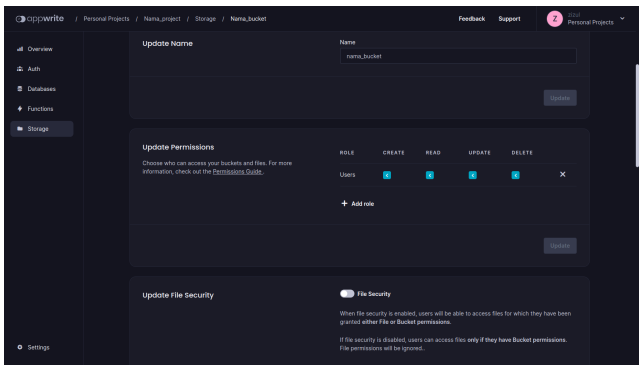
Service untuk mengatur files yang dipakai aplikasi seperti upload, melihat, mendownload, dan query images dan file-file lainnya. File akan diletakkan pada sebuah bucket (same as collection in databases) yang size datanya dapat diatur, dapat di encrypt, dapat di pasang scan anti virus dan lain-lain appwrite storage.

1. Buat atau gunakan sebuah bucket



2. Update permission pada setting bucket



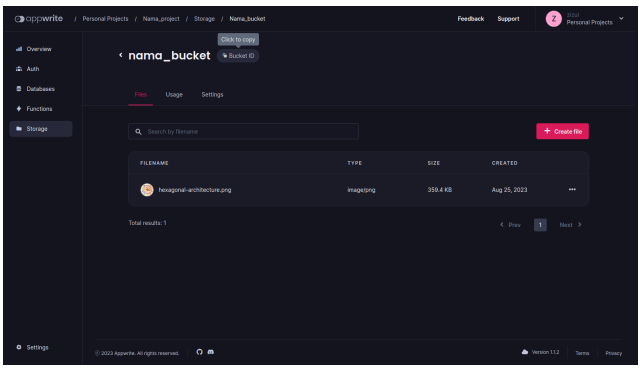


3. Pada contoh ini penggunaan dibuat sesederhana mungkin

```
class StorageController extends ClientController {
 Storage? storage;

 @override
 void onInit() {
  super.onInit();

  // appwrite
  storage = Storage(client);
 }

 Future storeImage() async {
  try {
   final result = await storage!.createFile(
    bucketId: '[BUCKET_ID]',
```

```
      fileId: ID.unique(),
      file: InputFile.fromPath(
        path: './path-to-files/image.jpg',
        filename: 'image.jpg',
      ),
    );
    print("StorageController:: storeImage $result");
  } catch (error) {
    Get.defaultDialog(
      title: "Error Storage",
      titlePadding: const EdgeInsets.only(top: 15, bottom: 5),
      titleStyle: Get.context?.theme.textTheme.titleLarge,
      content: Text(
        "$error",
        style: Get.context?.theme.textTheme.bodyMedium,
        textAlign: TextAlign.center,
      ),
      contentPadding: const EdgeInsets.only(top: 5, left: 15, right: 15),
    );
  }
 }
}
```

4. Copy Paste Bucket ID



5. Copy Paste URL path
   path: './path-to-files/image.jpg'

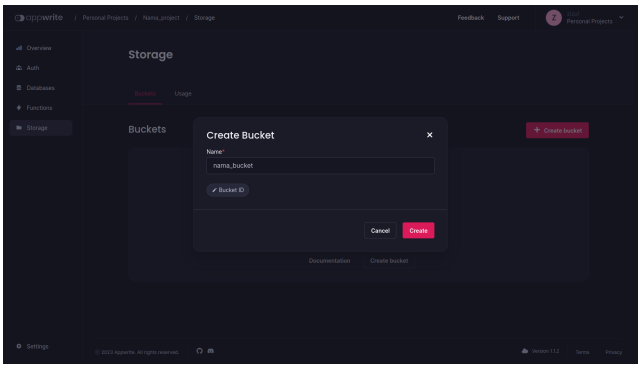
6. Input nama file untuk di storage
   filename: 'image.jpg'

7. Silahkan baca dokumentasi untuk case storage yang lain [appwrite storage]

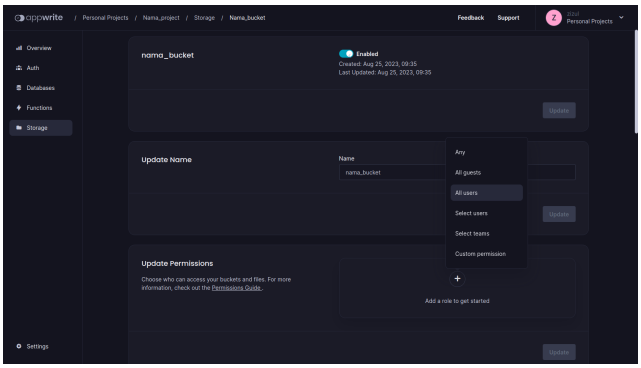
Realtime

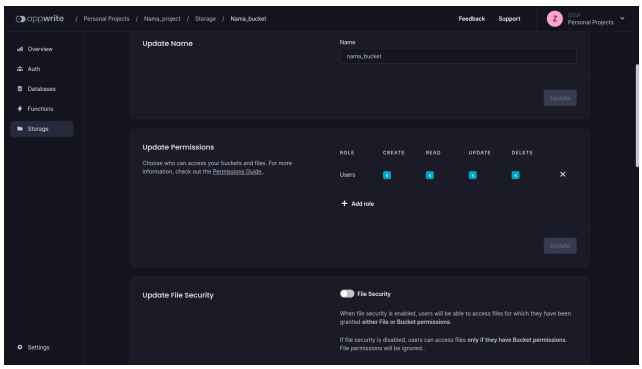
Service yang memantau sebuah request baru melalui HTTP, dengan metode subscription yang menerima setiap data baru bila terjadi perubahan. Setiap client yang mensubscribe akan update dalam milliseconds melalui koneksi WebSocket [appwrite realtime] [appwrite realtime-channel].

1. Buat atau gunakan sebuah bucket



2. Update permission pada setting bucket



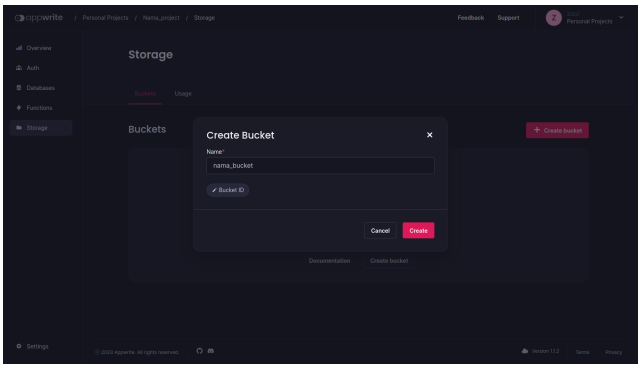


3. Pada contoh ini penggunaan dibuat sesederhana mungkin

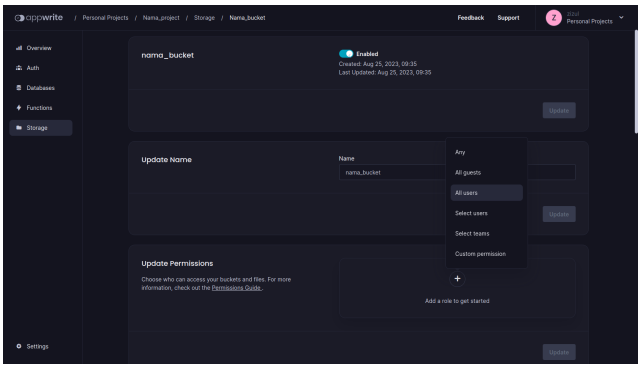
```
class RealtimeController extends ClientController {
 Realtime? realtime;

 @override
 void onInit() {
  super.onInit();

  // appwrite
  realtime = Realtime(client);
 }

 subsUserName() async {
  final subscription = realtime!.subscribe(['files']);

  subscription.stream.listen((response) {
```

```
    if (response.events.contains('buckets.*.files.*.create')) {
      print("RealtimeController:: subsUserName ${response.payload}");
      print("RealtimeController:: subsUserName ${response.events}");
    }
   });
  }
}
```

4. Buat atau gunakan sebuah bucket



5. Update permission pada setting bucket



6. List channel yang dapat subscribe [appwrite realtime-channel]

account	Semua event pada account (session create, name update...)
databases.[ID].collections.[ID].documents	Setiap create/update/delete events semua document di spesifik database dan collections
documents	Setiap create/update/delete events di semua document
databases.[ID].collections.[ID].documents.[ID]	Setiap update/delete events di sebuah document
files	Setiap create/update/delete events di semua file
buckets.[ID].files.[ID]	Setiap update/delete events di sebuah file
buckets.[ID].files	Setiap update/delete events untuk semua file di spesifik bucket
teams	Setiap create/update/delete events di semua team

teams.[ID]	Setiap update/delete events di spesifik team
memberships	Setiap create/update/delete events di semua membership
memberships.[ID]	Setiap update/delete events di spesifik membership
executions	Setiap update dari executions
executions.[ID]	Setiap update di sebuah execution
functions.[ID]	Setiap execution event di spesifik function

7. Silahkan baca dokumentasi untuk case realtime yang lain appwrite realtime

Link Github

Berikut merupakan link github yang akan digunakan pada modul pemrograman mobile:
Link Github

KEGIATAN PRAKTIKUM

A. Setup dan Instalasi Appwrite

1. Lakukan setup dan instalasi Appwrite ke project Anda.
2. Boleh menggunakan Local maupun Cloud

B. Latihan Melengkapi Tutorial Diatas

1. Lengkapi code-code dari penjelasan materi diatas.
2. Buatlah implementasi sesuai case aplikasi masing-masing.
3. Buatlah screen page untuk satu atau lebih penggunaan service.

RUBRIK PENILAIAN MODUL 4 MATERI

Bobot Penilaian Modul 4 Materi (20%)

Berhasil melakukan setup dan instalasi Appwrite	40
Berhasil menjalankan code	30
Memahami maksud dari code dengan baik	30
Total	100