

# Introduction to Firebase

## | Session 4




# Session Overview:

## Training Approach and Interaction



### Session 4

Presentation of Training Materials  
Discussion



# Session 4 Objective

Participants understand to set up and configure a Firebase project, perform CRUD operations with Firebase Realtime Database, implement serverless functions with Firebase Cloud Functions, and manage file uploads using Firebase Storage in a Spring Boot application

---

# MATERI PELATIHAN

---

- **Introduction to Firebase cloud platform and its services.**
- **Setting up Firebase project and configuring real-time.**
- **Basic CRUD operations with Firebase Realtime Database using Spring Boot.**
- **Implementing serverless functions with Firebase Cloud Functions and triggers.**
- **Utilizing Firebase Storage for file uploads in a Spring Boot application.**
- **Configuring and managing file uploads with Firebase Storage in Spring Boot.**

# Firebase cloud platform

---

- Firebase adalah platform yang dikembangkan oleh Google yang menyediakan berbagai layanan berbasis cloud untuk membangun dan mengelola aplikasi mobile dan web.
- Firebase menawarkan fitur seperti database real-time, otentikasi, penyimpanan cloud, hosting, dan fungsi serverless, yang memudahkan pengembang untuk mengintegrasikan layanan backend tanpa perlu mengelola infrastruktur.

# Firebase Services

---

- **Firestore Database:** Menyediakan database NoSQL yang menyimpan data secara real-time.
- **Authentication:** Mempermudah pengelolaan otentikasi pengguna dengan berbagai metode seperti email, Google, Facebook, dan lainnya.
- **Cloud Firestore:** Database fleksibel dan scalable yang memungkinkan penyimpanan dan pengambilan data dalam aplikasi secara efisien.
- **Storage:** Menyediakan tempat penyimpanan untuk file seperti gambar, video, atau dokumen yang dapat diakses oleh aplikasi mobile atau web.

# Firebase Services

---

- **Firebase Hosting:** Menyediakan layanan hosting untuk aplikasi web statis dan dinamis dengan pengaturan yang mudah dan cepat.
- **Firebase Cloud Functions:** Memungkinkan pengembang untuk menjalankan kode di cloud tanpa perlu mengelola server, seperti memproses data atau merespons peristiwa tertentu.
- **Firebase Analytics:** Menyediakan pelaporan dan analitik terkait penggunaan aplikasi untuk memahami perilaku pengguna dan mengoptimalkan pengalaman aplikasi.
- **Firebase Cloud Messaging:** Layanan untuk mengirimkan pemberitahuan push ke perangkat pengguna baik di aplikasi mobile maupun web.



# Setting up Firebase Project

---

- Membuat firebase project:  
<https://console.firebase.google.com/>
- Menambahkan dependency firebase pada project Spring Boot
- Download secret key dari firebase project
- Menambahkan Firebase Admin SDK untuk project



# Realtime database

---

```
9
10 @Service
11 public class FirebaseDatabaseService {
12
13     private final DatabaseReference databaseReference;
14
15     public FirebaseDatabaseService() {
16         this.databaseReference = FirebaseDatabase.getInstance().getReference("users");
17     }
18
19     public CompletableFuture<Void> createUser(User user) {
20         return CompletableFuture.supplyAsync(() -> {
21             try {
22                 databaseReference.child(String.valueOf(user.getId())).setValueAsync(user).get();
23                 return null;
24             } catch (Exception e) {
25                 throw new RuntimeException(e);
26             }
27         });
28     }
29
30     public DatabaseReference getUser(String id) {
31         return databaseReference.child(id);
32     }
33
34     public CompletableFuture<Void> updateUser(User user) {
35         return CompletableFuture.supplyAsync(() -> {
36             try {
37                 databaseReference.child(String.valueOf(user.getId())).setValueAsync(user).get();
38                 return null;
39             } catch (Exception e) {
40                 throw new RuntimeException(e);
41             }
42         });
43     }
44
45     public CompletableFuture<Void> deleteUser(String id) {
46         return CompletableFuture.supplyAsync(() -> {
47             try {
48                 databaseReference.child(id).removeValueAsync().get();
49                 return null;
50             } catch (Exception e) {
51                 throw new RuntimeException(e);
52             }
53         });
54     }
55 }
```

Firestore Database Service merupakan class service untuk mengatur logic CRUD operations.

# Realtime database

Firestore User Controller merupakan class controller untuk menangani HTTP klien sesuai dengan operasi yang dilakukan beserta metodenya.

```
1  @RestController
2  @RequestMapping("/api/firebase/users")
3  public class FirebaseUserController {
4
5      @Autowired
6      private FirebaseDatabaseService firebaseDatabaseService;
7
8      @PostMapping("/create")
9      public ResponseEntity<String> createUser(@RequestBody User user) {
10         try {
11             firebaseDatabaseService.createUser(user).get();
12             return ResponseEntity.ok("User created successfully");
13         } catch (InterruptedException | ExecutionException e) {
14             return ResponseEntity.status(500).body("Failed to create user: " + e.getMessage());
15         }
16     }
17
18     @GetMapping("/get/{id}")
19     public CompletableFuture<ResponseEntity<User>> getUser(@PathVariable String id) {
20         CompletableFuture<ResponseEntity<User>> future = new CompletableFuture<>();
21         DatabaseReference userRef = firebaseDatabaseService.getUser(id);
22
23         userRef.addListenerForSingleValueEvent(new ValueEventListener() {
24             @Override
25             public void onDataChange(DataSnapshot dataSnapshot) {
26                 if (dataSnapshot.exists()) {
27                     User user = dataSnapshot.getValue(User.class);
28                     future.complete(ResponseEntity.ok(user));
29                 } else {
30                     future.complete(ResponseEntity.status(HttpStatus.NOT_FOUND).body(null));
31                 }
32             }
33
34             @Override
35             public void onCancelled(DatabaseError databaseError) {
36                 future.completeExceptionally(databaseError.toException());
37             }
38         });
39
40         return future;
41     }
42
43     @PutMapping("/update")
44     public ResponseEntity<String> updateUser(@RequestBody User user) {
45         try {
46             firebaseDatabaseService.updateUser(user).get();
47             return ResponseEntity.ok("User updated successfully");
48         } catch (InterruptedException | ExecutionException e) {
49             return ResponseEntity.status(500).body("Failed to update user: " + e.getMessage());
50         }
51     }
52
53     @DeleteMapping("/delete/{id}")
54     public ResponseEntity<String> deleteUser(@PathVariable String id) {
55         try {
56             firebaseDatabaseService.deleteUser(id).get();
57             return ResponseEntity.ok("User deleted successfully");
58         } catch (InterruptedException | ExecutionException e) {
59             return ResponseEntity.status(500).body("Failed to delete user: " + e.getMessage());
60         }
61     }
62 }
```

# Firebase Cloud Functions

---

- Firebase Cloud Functions adalah layanan yang memungkinkan pengembang untuk menjalankan kode di cloud tanpa perlu mengelola server.
- Fungsi-fungsi ini dijalankan sebagai respons terhadap peristiwa atau pemicu tertentu, seperti perubahan data di Firebase Realtime Database atau Firestore, otentikasi pengguna, atau permintaan HTTP.
- Layanan ini mendukung pemrograman serverless, sehingga pengembang dapat fokus pada logika aplikasi tanpa khawatir tentang infrastruktur dan pengelolaan server.
- Firebase Cloud Functions sangat berguna untuk memproses data secara otomatis, mengirim pemberitahuan, atau melakukan integrasi dengan layanan lain

# Firestore Triggers

---

- Firestore Trigger: Fungsi yang dipicu ketika ada perubahan (penambahan, pembaruan, penghapusan) di koleksi atau dokumen dalam Firestore.
- Realtime Database Trigger: Fungsi yang dipicu ketika ada perubahan data dalam Realtime Database.
- Authentication Trigger: Fungsi yang dipicu saat ada peristiwa terkait autentikasi pengguna, seperti pendaftaran pengguna baru atau perubahan status akun pengguna.
- Analytics Trigger: Fungsi yang dapat dijalankan berdasarkan peristiwa yang tercatat di Analytics, seperti saat pengguna menyelesaikan aksi tertentu di aplikasi.
- HTTP Trigger: Fungsi yang dipicu oleh permintaan HTTP, memungkinkan pengembang untuk membuat API atau mengintegrasikan dengan layanan eksternal.
- Storage Trigger: Fungsi yang dipicu saat ada perubahan pada file yang disimpan di Storage, seperti file yang diunggah atau dihapus.

# Firestore Triggers

---

- Firestore Trigger: Fungsi yang dipicu ketika ada perubahan (penambahan, pembaruan, penghapusan) di koleksi atau dokumen dalam Firestore.
- Realtime Database Trigger: Fungsi yang dipicu ketika ada perubahan data dalam Realtime Database.
- Authentication Trigger: Fungsi yang dipicu saat ada peristiwa terkait autentikasi pengguna, seperti pendaftaran pengguna baru atau perubahan status akun pengguna.
- Analytics Trigger: Fungsi yang dapat dijalankan berdasarkan peristiwa yang tercatat di Analytics, seperti saat pengguna menyelesaikan aksi tertentu di aplikasi.
- HTTP Trigger: Fungsi yang dipicu oleh permintaan HTTP, memungkinkan pengembang untuk membuat API atau mengintegrasikan dengan layanan eksternal.
- Storage Trigger: Fungsi yang dipicu saat ada perubahan pada file yang disimpan di Storage, seperti file yang diunggah atau dihapus.

# Firestore

---

- Firestore adalah layanan yang memungkinkan penyimpanan dan pengelolaan file seperti gambar, video, dan dokumen di cloud.
- Firestore dirancang untuk menangani file berukuran besar dan memberikan akses yang aman serta cepat untuk aplikasi mobile dan web.



# Firestore

---

- Firestore adalah layanan yang memungkinkan penyimpanan dan pengelolaan file seperti gambar, video, dan dokumen di cloud.
- Firestore dirancang untuk menangani file berukuran besar dan memberikan akses yang aman serta cepat untuk aplikasi mobile dan web.



# Implement Firebase Storage

---

```
1 private File convertToFile(MultipartFile multipartFile, String fileName) throws IOException {
2     File tempFile = new File(fileName);
3     try (FileOutputStream fos = new FileOutputStream(tempFile)) {
4         fos.write(multipartFile.getBytes());
5         fos.close();
6     }
7     return tempFile;
8 }
```

```
1 private String uploadFile(File file, String fileName) throws IOException {
2     BlobId blobId = BlobId.of("your-bucket-name", fileName); // Replace with your bucket name
3     BlobInfo blobInfo = BlobInfo.newBuilder(blobId).setContentType("media").build();
4     InputStream inputStream = ImageService.class.getClassLoader().getResourceAsStream("firebase-private-key.json"); // change the file name with your one
5     Credentials credentials = GoogleCredentials.fromStream(inputStream);
6     Storage storage = StorageOptions.newBuilder().setCredentials(credentials).build().getService();
7     storage.create(blobInfo, Files.readAllBytes(file.toPath()));
8
9     String DOWNLOAD_URL = "https://firebasestorage.googleapis.com/v0/b/<bucket-name>/o/%s?alt=media";
10    return String.format(DOWNLOAD_URL, URLEncoder.encode(fileName, StandardCharsets.UTF_8));
11 }
```

# Implement Firebase Storage

---


```
1 public String upload(MultipartFile multipartFile) {
2     try {
3         String fileName = multipartFile.getOriginalFilename(); // to get original file name
4         fileName = UUID.randomUUID().toString().concat(this.getExtension(fileName)); // to generated random string
5                                         // values for file name.
6         File file = this.convertToFile(multipartFile, fileName); // to convert multipartFile to File
7         String URL = this.uploadFile(file, fileName); // to get uploaded file link
8         file.delete();
9         return URL;
10    } catch (Exception e) {
11        e.printStackTrace();
12        return "Image couldn't upload, Something went wrong";
13    }
14 }
```

```
1 @RestController
2 @RequiredArgsConstructor
3 @RequestMapping("/api/upload")
4 public class ImageController {
5
6     private final ImageService imageService;
7
8     @PostMapping
9     public String upload(@RequestParam("file") MultipartFile multipartFile) {
10         return imageService.upload(multipartFile);
11     }
12 }
```

# REFERENCES

---

 Firebase. (n.d.). Firebase. <https://firebase.google.com/>

 Irosha, P. (2023, June 6). File uploading with Spring Boot & Firebase Cloud Storage. Medium. <https://medium.com/@poojithairosha/image-uploading-with-spring-boot-firebase-cloud-storage-e5ef2fbf942d>



# THANK YOU

Thank you for following session 4 to the end.  
See you at the next meeting