Kayaan – Supabase Integration Skeleton (Storage + Realtime)

> Spring Boot 3.x, Java 17. ใช้ **Auth เดิม (JWT)** และเสริม **Supabase Storage + Realtime** ผ่าน Backend เท่านั้น

---

0) Gradle/Maven Dependencies (ตัวอย่าง pom.xml ส่วนที่เกี่ยวข้อง)

```xml
<!-- Supabase ใช้ผ่าน HTTP (Storage) และ WebSocket (Realtime) -->
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-validation</artifactId>
</dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-websocket</artifactId>
</dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-security</artifactId>
</dependency>
<dependency>
  <groupId>com.fasterxml.jackson.core</groupId>
  <artifactId>jackson-databind</artifactId>
</dependency>

<!-- (ถ้าใช้ OkHttp หรือ Reactor Netty สำหรับเรียก Supabase) -->
<dependency>
  <groupId>com.squareup.okhttp3</groupId>
  <artifactId>okhttp</artifactId>
  <version>4.12.0</version>
</dependency>
<dependency>
  <groupId>io.projectreactor.netty</groupId>
  <artifactId>reactor-netty</artifactId>
  <version>1.1.20</version>
</dependency>
```

---

1) application.yml (เพิ่มค่า env สำหรับ Supabase)

```
kayaan:
  supabase:
    url: ${SUPABASE_URL:https://YOUR-PROJECT.supabase.co}
    serviceKey: ${SUPABASE_SERVICE_KEY}
    buckets:
      avatars: ${SUPABASE_BUCKET_AVATARS:avatars}
      ai: ${SUPABASE_BUCKET_AI:ai-outputs}
    realtime:
      enabled: true
      # channel prefix ตัวอย่าง
      groupChannelPrefix: groups.

# แนะนำให้ใช้ profile แยก dev/prod และเก็บ secret ไว้ใน env ไม่ commit
```

---

2) Core Interfaces

2.1 `StorageService.java`

```
package com.kayaan.shared.storage;

import java.time.Duration;

public interface StorageService {
    record SignedUrl(String url, String path, long expiresInSeconds) {}

    SignedUrl createSignedUploadUrl(String bucket, String path, Duration
ttl, String contentType);
    String getPublicUrl(String bucket, String path);
    void delete(String bucket, String path);
}
```

2.2 `RealtimeBus.java`

```
package com.kayaan.shared.realtime;

public interface RealtimeBus {
    void publish(String channel, String event, String jsonPayload);
    void subscribe(String channel, RealtimeListener listener);

    interface RealtimeListener {
        void onEvent(String channel, String event, String jsonPayload);
        default void onError(Throwable t) {}
    }
}
```

3) Supabase Adapters

3.1 `SupabaseStorageAdapter.java`

```java
package com.kayaan.infra.supabase;

import com.kayaan.shared.storage.StorageService;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.http.HttpHeaders;
import org.springframework.http.MediaType;
import org.springframework.stereotype.Component;

import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.time.Duration;
import java.util.Map;

@Component
public class SupabaseStorageAdapter implements StorageService {

    private final String supabaseUrl;
    private final String serviceKey;

    private final HttpClient http = HttpClient.newHttpClient();

    public SupabaseStorageAdapter(
            @Value("${kayaan.supabase.url}") String supabaseUrl,
            @Value("${kayaan.supabase.serviceKey}") String serviceKey
    ) {
        this.supabaseUrl = supabaseUrl;
        this.serviceKey = serviceKey;
    }

    @Override
    public SignedUrl createSignedUploadUrl(String bucket, String path,
Duration ttl, String contentType) {
        try {
            String endpoint = supabaseUrl + "/storage/v1/object/sign/" +
bucket + "/" + path;
            String body = "{\"expiresIn\":" + ttl.toSeconds() + ",
\"contentType\":\"" + contentType + "\"}";

            HttpRequest req = HttpRequest.newBuilder()
```

```
                    .uri(URI.create(endpoint))
                    .header("Authorization", "Bearer " + serviceKey)
                    .header("apikey", serviceKey)
                    .header(HttpHeaders.CONTENT_TYPE,
MediaType.APPLICATION_JSON_VALUE)
                    .POST(HttpRequest.BodyPublishers.ofString(body))
                    .build();

            HttpResponse<String> resp = http.send(req,
HttpResponse.BodyHandlers.ofString());
            if (resp.statusCode() >= 300) {
                throw new IllegalStateException("Supabase sign URL failed: "
+ resp.statusCode() + " " + resp.body());
            }
            // response: { "signedURL":"/object/sign/...", "path":"bucket/
path" }
            String signedUrl = supabaseUrl + "/storage/v1" +
extract(resp.body(), "signedURL");
            return new SignedUrl(signedUrl, path, ttl.toSeconds());
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }

    @Override
    public String getPublicUrl(String bucket, String path) {
        return supabaseUrl + "/storage/v1/object/public/" + bucket + "/" +
path;
    }

    @Override
    public void delete(String bucket, String path) {
        try {
            String endpoint = supabaseUrl + "/storage/v1/object/" + bucket +
"/" + path;
            HttpRequest req = HttpRequest.newBuilder()
                    .uri(URI.create(endpoint))
                    .header("Authorization", "Bearer " + serviceKey)
                    .header("apikey", serviceKey)
                    .DELETE()
                    .build();
            HttpResponse<String> resp = http.send(req,
HttpResponse.BodyHandlers.ofString());
            if (resp.statusCode() >= 300) {
                throw new IllegalStateException("Supabase delete failed: " +
resp.statusCode() + " " + resp.body());
            }
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }
    private static String extract(String json, String key) {
        // mini extractor; แนะนำให้ใช้ Jackson จริงจังในโปรดักชัน
        int i = json.indexOf("\"" + key + "\"");
        if (i < 0) return "";
        int colon = json.indexOf(':', i);
        int q1 = json.indexOf('"', colon + 1);
        int q2 = json.indexOf('"', q1 + 1);
        return json.substring(q1 + 1, q2);
    }
}
```

หมายเหตุ: เพื่อความเรียบร้อยจริง ควรใช้ `ObjectMapper` แปลง JSON แทนเมธอด `extract`

3.2 `SupabaseRealtimeAdapter.java` (โครง)

```
package com.kayaan.infra.supabase;

import com.kayaan.shared.realtime.RealtimeBus;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Component;

// โครงเริ่มต้น: เก็บ publish/subscribe ไว้ให้ backend เท่านั้น
// ในโปรดักชันอาจใช้ websocket client (เช่น Java-WebSocket) ต่อกับ Supabase Realtime
Server
@Component
public class SupabaseRealtimeAdapter implements RealtimeBus {
    private final String supabaseUrl;
    private final String serviceKey;

    public SupabaseRealtimeAdapter(
            @Value("${kayaan.supabase.url}") String supabaseUrl,
            @Value("${kayaan.supabase.serviceKey}") String serviceKey
    ) {
        this.supabaseUrl = supabaseUrl;
        this.serviceKey = serviceKey;
    }

    @Override
    public void publish(String channel, String event, String jsonPayload) {
        // ทางเลือก 1: ใช้ Realtime REST relay (ถ้ามี) หรือผ่าน DB NOTIFY (ถ้าใช้ Postgres)
        // ทางเลือก 2: ใช้ WebSocket client เชื่อมต่อ channel แล้วส่ง message (ต้องใช้ไลบรารี
เสริม)
        // ที่นี่ใส่เป็นโครงเพื่อเสียบ implementation จริงภายหลัง
        System.out.printf("[Realtime PUBLISH] channel=%s event=%s
payload=%s\n", channel, event, jsonPayload);
    }

    @Override
    public void subscribe(String channel, RealtimeListener listener) {
        // โครง subscribe ฝั่ง backend เท่านั้น
        System.out.printf("[Realtime SUBSCRIBE] channel=%s\n", channel);
    }
}
```

---

4) Avatar – Controller (ขอ Signed URL + บันทึก URL)

4.1 `AvatarController.java` (รุ่นใหม่)

```
package com.kayaan.user;

import com.kayaan.shared.storage.StorageService;
import jakarta.validation.constraints.NotBlank;
import jakarta.validation.constraints.NotNull;
import org.springframework.http.ResponseEntity;
import org.springframework.security.access.prepost.PreAuthorize;
import org.springframework.web.bind.annotation.*;

import java.time.Duration;
import java.util.Map;

@RestController
@RequestMapping("/api/users")
public class AvatarController {

    private final StorageService storage;
    private final UserService userService; // ของเดิมในโปรเจ็กต์
    private final String avatarBucket;

    public AvatarController(StorageService storage, UserService userService,

@org.springframework.beans.factory.annotation.Value("$
{kayaan.supabase.buckets.avatars}") String avatarBucket) {
        this.storage = storage;
        this.userService = userService;
        this.avatarBucket = avatarBucket;
    }

    public record UploadUrlReq(@NotBlank String fileName, @NotBlank String
contentType) {}

    @PostMapping("/{id}/avatar-upload-url")
    @PreAuthorize("#id == authentication.principal.id or hasRole('ADMIN')")
    public ResponseEntity<?> requestUploadUrl(@PathVariable Long id,
@RequestBody UploadUrlReq req) {
        String path = "users/" + id + "/" + System.currentTimeMillis() + "_"
+ req.fileName();
        var signed = storage.createSignedUploadUrl(avatarBucket, path,
Duration.ofMinutes(10), req.contentType());
        return ResponseEntity.ok(Map.of(
                "signedUrl", signed.url(),
                "path", signed.path(),
                "expiresIn", signed.expiresInSeconds()
        ));
    }

    public record SaveAvatarUrlReq(@NotBlank String path) {}

    @PutMapping("/{id}/avatar-url")
    @PreAuthorize("#id == authentication.principal.id or hasRole('ADMIN')")
    public ResponseEntity<?> saveAvatarUrl(@PathVariable Long id,
@RequestBody SaveAvatarUrlReq req) {
        String publicUrl = storage.getPublicUrl(avatarBucket,
req.path()); // ถ้าใช้ private ทั้งหมด ให้เสิร์ฟผ่าน backend แทน
        userService.updateAvatarUrl(id, publicUrl, req.path());
        return ResponseEntity.ok(Map.of("avatarUrl", publicUrl, "path",
req.path()));
    }
}
```

> ถ้าบัคเก็ตเป็น `private` ทั้งหมดจริง ๆ ไม่ควรให้ `publicUrl`; ให้เสิร์ฟรูปผ่าน proxy endpoint ที่ตรวจ JWT + สร้าง signed URL ชั่วคราวอีกชั้น

---

5) Study Group – WebSocket (ผ่าน Backend)

5.1 `WebSocketConfig.java`

```
package com.kayaan.realtime;

import org.springframework.context.annotation.Configuration;
import org.springframework.messaging.simp.config.MessageBrok
```