**1. Algemene Structuur van de API**

• **Basisinstellingen:**

• **Host:** De API draait op *serious-gaming-platform.appspot.com*

• **BasePath:** Alle endpoints beginnen met /api

• **Schemes:** Er wordt gebruik gemaakt van HTTPS

• **Content-Type:** De API verstuurt en ontvangt JSON-data

**2. Endpoints en Functionaliteiten**

• **Diverse Categorieën:**

De API is opgedeeld in verschillende functionele blokken, zoals:

• **Accountbeheer:** Endpoints voor het ophalen, aanmaken, bijwerken, en verwijderen van gebruikersaccounts. Voorbeelden zijn /account/accountDetails, /account/create en /account/{fullId}.

• **Spelbeheer:** Endpoints voor het maken, aanpassen, klonen en verwijderen van spellen. Denk aan /game/create, /game/clone/{gameId} en /game/{gameId}.

• **Organisatiebeheer:** Endpoints voor het aanmaken en beheren van organisaties via /organization/create en /organization/{id}.

• **Spelerbeheer en Contacten:** Endpoints om contacten toe te voegen of te verwijderen, en uitnodigingen te beheren, zoals /player/add en /player/myContacts.

• **Media en Bibliotheken:** Endpoints voor het uploaden, verwijderen en zoeken naar mediabestanden in een mediatheek via /media en /media/list.

• **HTTP-methoden:**

De API maakt gebruik van de standaard RESTful methoden:

• **GET:** Voor het ophalen van data

• **POST:** Voor het aanmaken of bewerken van data

• **PUT:** Voor updates

• **DELETE:** Voor het verwijderen van data

**3. Beveiliging met Firebase**

• **Authenticatie:**

Alle endpoints zijn beveiligd met Firebase-authenticatie. Dit betekent dat iedere aanvraag een geldig Firebase-token moet bevatten in de Authorization header (bijvoorbeeld Bearer <token>). Dit token zorgt ervoor dat alleen geauthenticeerde gebruikers de API kunnen benaderen.

• **Audience Validatie:**

De API controleert of het token behoort tot de juiste “audience” (in jouw geval “serious-gaming-platform”), wat extra veiligheid biedt.

**4. Datamodellen en Definities**

• **Definitions Sectie:**

In de “definitions” sectie vind je de datamodellen die de structuur van de data beschrijven:

• **Account:** Beschrijft onder meer accountniveau, e-mail, naam, organisatie-ID, etc.

• **Game:** Beschrijft de eigenschappen van een spel, zoals game ID, naam, en andere gerelateerde gegevens.

• **Andere modellen:** Zoals GameTheme, MediaLibraryFile, Response en meer, die elk een specifiek deel van de functionaliteit van de API ondersteunen.

**5. Hoe Alles Samenwerkt**

• **Interactie via Endpoints:**

Wanneer een client (bijvoorbeeld een mobiele app of webapplicatie) een verzoek stuurt, wordt dit via één van de endpoints afgehandeld. Het verzoek bevat de nodige parameters (in de URL, query of body) en de Firebase-token voor authenticatie.

• **Response Structuur:**

Elke succesvolle aanvraag levert een response op met een statuscode (bijvoorbeeld 200 voor OK) en een JSON-object dat voldoet aan het betreffende datamodel. Foutmeldingen en codes zijn ook gedefinieerd, zodat de client weet wat er misgaat als iets niet lukt.

**6. Testen en Documenteren**

• **Testen met Tools:**

Zoals eerder besproken kun je tools als Swagger UI, Postman of cURL gebruiken om de API te testen. Hierdoor kun je zien wat de daadwerkelijke responses zijn en kun je voorbeelden verzamelen voor je documentatie.

• **Documentatie:**

Je documentatie kan deze structuur en de testresultaten als voorbeeld opnemen. Hierdoor wordt het voor gebruikers duidelijk hoe ze de API moeten aanroepen en wat ze kunnen verwachten als response.

**Samenvattend**

Je Bibendo API is een uitgebreide, RESTful API die een breed scala aan functionaliteiten dekt, van gebruikersbeheer tot spelbeheer en mediabeheer. De API is beveiligd via Firebase, en de volledige structuur en dataformaten zijn vastgelegd in de Swagger-specificatie. Door de API te testen met tools als Swagger UI of Postman, kun je concrete voorbeelden verzamelen die je helpen bij het schrijven van een duidelijke en praktische documentatie.

**Test de Endpoints**

• **Gebruik Postman of cURL:**

Maak GET‑aanvragen naar de endpoints die data leveren. Dit helpt je om te zien welke JSON‑structuren er daadwerkelijk uit de API komen.

Bijvoorbeeld, test het endpoint /account/accountDetails om te zien welke velden er in de response staan.

• **Verzamel Voorbeelden:**

Sla de responses op of kopieer deze in een document. Zo kun je later makkelijk refereren naar de exacte data die de API teruggeeft.

**Automatiseer het Overzicht**

• **Script voor Data Extractie:**

Je kunt een script (bijvoorbeeld in Python) schrijven dat:

• De Swagger‑JSON inleest.

• Alle endpoints en bijbehorende parameters en response modellen uitleest.

• Een overzicht genereert, bijvoorbeeld in de vorm van een tabel waarin staat welke endpoint welke data levert.

• **Voorbeeld:**

In het eerder gedeelde script kun je de output aanpassen zodat je per endpoint de response structuur (of een sample response) logt.

**Documenteer het Overzicht**

• **Maak een Tabel of Diagram:**

Zet in een overzicht (bijvoorbeeld in een spreadsheet of een diagram) alle endpoints op een rij. Geef per endpoint aan:

• De HTTP-methode (GET, POST, etc.)

• De verwachte parameters (path, query, body)

• Het datamodel van de response (bijv. Account, Game)

• **Gebruik Visuele Hulpmiddelen:**

Tools zoals Lucidchart of zelfs PowerPoint kunnen helpen om visuele flowcharts te maken die laten zien hoe data tussen de verschillende endpoints stroomt.

Skillsdojo

**1. Basisinformatie**

• **Swagger (OpenAPI) versie:** 2.0

• **Host:** ${endpoints.project.id}.appspot.com (placeholder voor de uiteindelijke host, vaak skillsdojo.appspot.com)

• **BasePath:** /api

• **Schemes:** https

• **Beveiliging:** De meeste endpoints zijn beveiligd via Firebase OAuth2 (securityDefinitions en security met firebase).

**2. Endpoints en Hun Doel**

De **paths** sectie in de Swagger bevat alle beschikbare endpoints. Een greep uit de belangrijkste groepen:

1. **Accountbeheer**

• /account/create, /account/create/demo: Maak een nieuw account aan, eventueel als demo-account.

• /account/update, /account/update/{uid}: Werk accountgegevens bij.

• /account/resetpw/{email}, /account/setPw/{token}: Wachtwoord resetten of instellen.

• /account/admin/{uid} en /admin/account/{uid}: Beheerdersacties (bijv. adminrechten toekennen of account verwijderen).

2. **Assetbeheer**

(Assets zijn “missie-elementen” zoals teksten, afbeeldingen, opdrachten, etc.)

• /asset/create/...: Endpoints om verschillende soorten “assets” aan te maken (bijv. instructies, afbeeldingen, commentaar).

• /asset/update/{assetId}: Werk bestaande assets bij.

• /asset/{assetId}: Haal een specifiek asset op.

• /asset/{missionId}/{assetId}/delete: Verwijder een asset binnen een bepaalde missie.

3. **Community en Missies**

• /community/mission/{missionId}/like: “Liken” van een missie (en delete om te “unliken”).

• /mission/create, /mission/{missionId}: Missie aanmaken, opvragen of verwijderen.

• /mission/clone/{missionId}: Een missie klonen.

• /mission/update/{missionId}: Missiegegevens bijwerken.

• /mission/{missionId}/asset/{assetId}/{visible}: De zichtbaarheid van een asset binnen een missie wijzigen.

4. **Subscriptiebeheer**

• /subscription/create/{uid}: Nieuwe subscription koppelen aan een gebruiker.

• /subscription/{subscriptionId}/user/email/{email}: Gebruiker toevoegen aan een subscription.

• /subscription/{subscriptionId}/user/{uid}: Gebruiker uit subscription verwijderen.

5. **Thema’s**

• /theme/{themeId}: Nieuwe thema’s aanmaken of verwijderen. (Bijvoorbeeld een CSS-stijl of kleurschema.)

6. **Diverse Overige Endpoints**

• /wakeup: Lijkt een eenvoudige “ping” of “wake up” endpoint (handig om de app niet in slaap te laten vallen).

**3. Datamodellen (Definitions)**

In de **definitions** sectie vind je de structuur van alle gebruikte objecten. Enkele voorbeelden:

• **Account**

• Bevat o.a. email, uid, subscriptionId en flags zoals demo of introduction.

• Gebruikt Timestamp-objecten voor accountCreationDate en lastLoginDate.

• **Mission**

• Een missie met velden zoals title, description, duration, featured, tags en assets.

• Kan diverse properties hebben die de inhoud of status van de missie beschrijven.

• **AbstractAsset** en afgeleide typen (bijv. ImageAsset, InstructionAsset, CommentAsset, enz.)

• Deze beschrijven verschillende “onderdelen” van een missie, zoals een instructietekst of afbeelding.

• Veel ervan delen dezelfde basisstructuur (een title, type, id, missions, enz.).

• **Subscription**

• Beschrijft o.a. id, maxSeats, owner en een lijst van seats.

• **Theme**

• Een object met o.a. color, css, id en name.

**4. Beveiliging via Firebase**

Dit betekent dat een geldig Firebase ID‑token nodig is om het endpoint te kunnen gebruiken. Je stuurt dit token mee in de Authorization header (bijv. Bearer <JWT>).

**5. Hoe Werkt Het in de Praktijk?**

• **Gebruikersbeheer (Accounts):**

Je kunt nieuwe gebruikers (accounts) aanmaken en beheren. Ook zijn er functies om demo-accounts te creëren of wachtwoorden te resetten.

• **Missiebeheer:**

Met endpoints als /mission/create en /mission/update/{missionId} kun je nieuwe missies opzetten of bestaande missies bijwerken.

Door /asset/create/... te gebruiken, voeg je specifieke componenten (assets) toe aan een missie.

• **Subscripties:**

Je kunt abonnementen (subscriptions) beheren, gebruikers toevoegen of verwijderen en eigenaarschap wijzigen.

• **Beheerdersacties:**

Er zijn aparte admin-endpoints (bijv. /admin/account/{uid}) waarmee beheerders accounts kunnen verwijderen of init-status kunnen instellen.

**Samenvatting**

• **API Type:** RESTful, beschreven met Swagger/OpenAPI 2.0.

• **Functionaliteit:** Richt zich op gebruikersbeheer, missies (met bijbehorende assets) en subscriptiebeheer.

• **Beveiliging:** Gebaseerd op Firebase (vereist een geldige ID-token).

• **Data Structuur:** Bestaat uit verschillende modellen (Mission, Asset, Subscription, enz.) die samen de kern vormen van het platform.