# [**https://firebase.google.com/docs/auth/admin/manage-users?hl=ru**](https://firebase.google.com/docs/auth/admin/manage-users?hl=ru) (Документация)

## 1. **auth/create-user**

|  |  |
| --- | --- |
| Запрос | {    "data": {      "email": "user@example.com",      "password": "password123"    }  } |
| Метод | module.exports.createUser = async function (*request*, *reply*) {    const { data } = *request*.body    if (!data || !data.email || !data.password) {      throw new Error('Invalid data')    }    console.dir(Firebase.admin)    const firebaseAuth = Firebase.admin.auth()    try {      const userCredential = await firebaseAuth.createUser({        email: data.email,        password: data.password      })      const uid = userCredential.uid  *reply*.send({ data: `User ${uid} created` })    } catch (err) {      if (err.code === 'auth/email-already-exists') {        throw HttpError[409]('Email already exists')      } else {        console.log(err)        throw HttpError.InternalServerError('Server error.')      }    }  } |
| Результат | {      "statusCode": 409,      "error": "Conflict",      "message": "Email already exists"  } or  {      "data": "User ZiWmk9sPrkVnmc0AoD2GC52PxlF3 created"  } |

## 2. **auth/list-all-users**

|  |  |
| --- | --- |
| Запрос | * GET |
| Метод | module.exports.listAllUsers = async function (*request*, *reply*) {    const firestoreAuth = Firebase.admin.auth()    try {      const listUsersResult = await firestoreAuth.listUsers()      const users = listUsersResult.users.map((*userRecord*) => {        const { uid, email } = *userRecord*;        return { uid, email }      })  *reply*.send({ data: users })    } catch (err) {      console.log(err);      throw HttpError.InternalServerError('Server error.')    }  } |
| Результат | {      "data": [          {              "uid": "U4WTHKj6twPdPzBBYax97wX8HZ13",              "email": "user3@example3.com"          },          {              "uid": "sKmGDjaDqZU2NiRTVbSe4VcZG1x2",              "email": "user5@example5.com"          }      ]  } |

## 3. **auth/login-user (\*return jwt-token)**

“API также возвращает passwordSalt и passwordHash , хешированные серверной частью аутентификации Firebase для пользователей паролей, если учетная запись пользователя/службы, используемая для создания токена доступа OAuth запроса, имеет разрешение firebaseauth.configs.getHashConfig . В противном случае passwordHash и passwordSalt не будут установлены.”

gcloud iam service-accounts add-iam-policy-binding SERVICE\_ACCOUNT\_EMAIL --member=serviceAccount:**firebase-adminsdk-dk1j4@tm-shpf-srv-local-ksv.iam.gserviceaccount.com** --role=roles/firebaseauth.configs.getHashConfig

NOT\_FOUND: Unknown service account

gcloud iam service-accounts add-iam-policy-binding firebase-service-account@firebase-sa-management.iam.gserviceaccount.com --member=serviceAccount:**firebase-adminsdk-dk1j4@tm-shpf-srv-local-ksv.iam.gserviceaccount.com** --role=roles/firebaseauth.configs.getHashConfig

NOT\_FOUND: Service account projects/tm-shpf-srv-local-ksv/serviceAccounts/firebase-service-account@firebase-sa-management.iam.gserviceaccount.com does not exist.

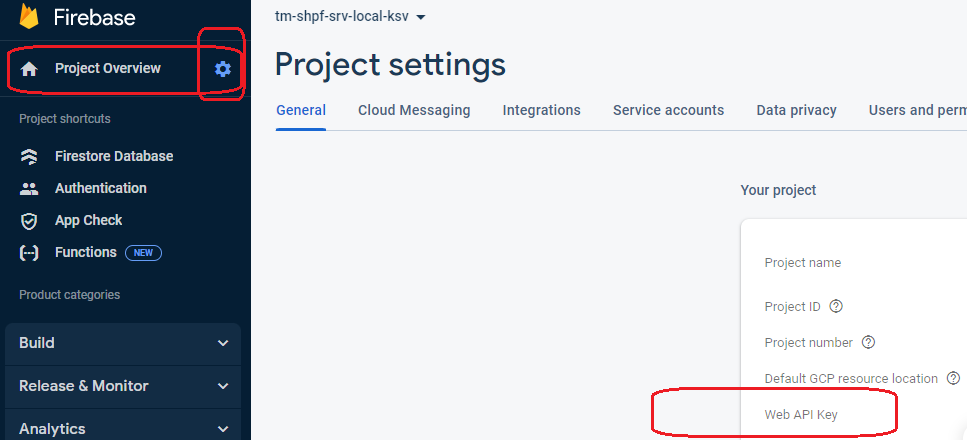
|  |  |
| --- | --- |
| Запрос | {    "data": {      "email": "user@example.com",      "password": "password123"    } } |
| Метод | module.exports.loginUser = async function (*request*, *reply*) {    const { data } = *request*.body    if (!data || !data.email || !data.password) {      throw new Error('Invalid data')    }    const firebaseAuth = Firebase.admin.auth()    try {      const user = await firebaseAuth.getUserByEmail(data.email)      const idToken = await firebaseAuth.createCustomToken(user.uid)  *reply*.send({ data: idToken })    } catch (err) {      if (err.code === 'auth/user-not-found' || err.code === 'auth/wrong-password') {        throw HttpError.Unauthorized('Invalid email or password')      } else {        console.log(err)        throw HttpError.InternalServerError('Server error.')      }    }  } |
| Результат | {      "data": “tokencvbcvbcvbcbvfgfgjdghjdghjdglj;ld;lj;ld;glj;dlg;j;ghjdghjdgh…”  } |
| decode | **https://token.dev/** |

## 4. **auth/verify-token**

## 4.1.. **auth/verify-token through jsonwebtoken node-rsa**

|  |  |
| --- | --- |
| Запрос | Header Authorization = ‘<token>’ |
| Метод | *//#region verify-token*  module.exports.verifyToken = async function (*request*, *reply*) {    const { authorization } = *request*.headers    if (!authorization) {      throw new Error('Invalid data')    }    try {      const firebaseAuth = Firebase.admin.auth()      const publicKey = new NodeRSA().importKey(serviceAccount.private\_key, 'pkcs8-private-pem').exportKey('pkcs8-public-pem')      const decodedToken = jwt.verify(authorization, publicKey)      const user = await firebaseAuth.getUser(decodedToken.uid)  *// const user = await firebaseAuth.verifyIdToken(idToken)*  *// reply.send({ data: user.uid, message: `Token is valid for ${user.email}` })*  *reply*.send({ data: decodedToken.uid, message: `Token is valid for ${decodedToken.uid}, ${user.email}` });    } catch (err) {      if (err.code === 'auth/user-not-found' || err.code === 'auth/wrong-password') {        throw HttpError.Unauthorized('Invalid email or password')      } else {        console.log(err)        throw HttpError.InternalServerError('Server error.')      }    }  }  *//#endregion verify-token* |
| Результат | {      "statusCode": 500,      "error": "Internal Server Error",      "message": "Server error."  } |
|  | **npm install jsonwebtoken node-rsa**  const jwt = require('jsonwebtoken');  const NodeRSA = require('node-rsa'); |
|  | const publicKey = new NodeRSA().importKey(serviceAccount.private\_key, 'pkcs8-private-pem').exportKey('pkcs8-public-pem');  const decodedToken = jwt.verify(idToken, publicKey); |
|  | https://stackoverflow.com/questions/44368643/verifying-firebase-custom-token-to-get-token-id-fails-when-using-jsonwebtoken |
|  | {      "data": "ZiWmk9sPrkVnmc0AoD2GC52PxlF3",      "message": "Token is valid for ZiWmk9sPrkVnmc0AoD2GC52PxlF3, user11@example11.com"  } |

## 4.2. **auth/verify-token through Firestore access-token**



|  |  |
| --- | --- |
| Запрос | {    "data": {      "email": "user11@example11.com",      "password": "password123 "    }  } |
| Метод login | module.exports.loginUser = async function (*request*, *reply*) {    const { data } = *request*.body    if (!data || !data.email || !data.password) {      throw new Error('Invalid data')    }    const { FIREBASE\_WEB\_API\_KEY } = process.env    const firebaseAuth = Firebase.admin.auth()    try {      const user = await firebaseAuth.getUserByEmail(data.email)      console.log(user.uid)      const token = await firebaseAuth.createCustomToken(user.uid)      const res = await axios({        url: `https://www.googleapis.com/identitytoolkit/v3/relyingparty/verifyCustomToken?key=${FIREBASE\_WEB\_API\_KEY}`,        method: 'post',        data: {          token,          returnSecureToken: true        },        json: true      })      const idToken = res.data.idToken  *reply*.send({ jwttoken: idToken })    } catch (err) {      if (err.code === 'auth/user-not-found' || err.code === 'auth/wrong-password') {        throw HttpError.Unauthorized('Invalid email or password')      } else {        console.log(err)        throw HttpError.InternalServerError('Server error.')      }    }  } |
| Результат | {      "jwttoken": "eyJhbGciQ"  } |
| Запрос | Header Authorization = ‘<token>’ |
| Method Verify | module.exports.verifyToken = async function (*request*, *reply*) {    const { authorization } = *request*.headers    if (!authorization) {      throw new Error('Invalid data')    }    try {      const firebaseAuth = Firebase.admin.auth()      const user = await firebaseAuth.verifyIdToken(authorization)  *reply*.send({ data: user.uid, message: `Token is valid for ${user.email}` })    } catch (err) {      if (err.code === 'auth/user-not-found' || err.code === 'auth/wrong-password') {        throw HttpError.Unauthorized('Invalid email or password')      } else {        console.log(err)        throw HttpError.InternalServerError('Server error.')      }    }  } |
| Результат | {      "data": "ZiWmk9sPrkVnmc0AoD2GC52PxlF3",      "message": "Token is valid for user11@example11.com"  } |

# Consclusions.

**Firebase Auth**:

1. **auth/create-user** **+**
2. **auth/list-all-users +**
3. **auth/login-user** with CreateCustomToken and send has problems such as:
   1. проблема с методами проверки пароля пользователя, а именно
      1. методы типа VerifyPassword или SignInWithemailand passwd не работают
      2. при получении пользователя поля, хранящие хеши пароля = undefined. В документации пишут “API также возвращает passwordSalt и passwordHash , хешированные серверной частью аутентификации Firebase для пользователей паролей, если учетная запись пользователя/службы, используемая для создания токена доступа OAuth запроса, имеет разрешение firebaseauth.configs.getHashConfig . В противном случае passwordHash и passwordSalt не будут установлены.” но установить firebaseauth.configs.getHashConfig через gcloud cli не удалось, интерактивно тоже не нашла его
   2. по самому токену тоже какая-то проблема возможно есть, т.к. декодинг на сайте iojwt показывает данные токена как JWT RS256 … , но пишет Invalid Signature
4. **verify-token** has problems such as:
   1. firebaseAuth.verifyIdToken(id) code: 'auth/argument-error', message: 'Decoding Firebase ID token failed. Make sure you passed the entire string JWT which represents an ID token. See https://firebase.google.com/docs/auth/admin/verify-id-tokens for details on how to retrieve an ID token.'