## 1. **db-requests/list-collections (**http://[::1]:8181/db-requests/list-collections)

|  |  |
| --- | --- |
| Запрос | - |
| Метод | module.exports.listCollections = async function (*request*, *reply*) {    try {      const collections = await Firebase.admin.firestore().listCollections()      const collectionNames = collections.map((*c*) => *c*.id)      return { collections: collectionNames }    } catch (err) {      if (err.code === 404 || err.message === 'Not active') {        throw new Error('Not Found or Not active')      }      console.error(err)      throw new Error('Server error.')    }  } |
| Результат | {      "collections": [          "clients",          "ip\_addresses",          "services",          "test\_111"      ]  } |

## 2. **db-requests/create-collection**

|  |  |
| --- | --- |
| Запрос | {      "collectionId": "test\_222"  } |
| Метод | module.exports.createCollection = async function (*request*, *reply*) {    try {      const collectionId = *request*.body.collectionId      const firestore = Firebase.admin.firestore()  *// Check if the collection already exists*      const collectionRef = firestore.collection(collectionId)      const collectionSnapshot = await collectionRef.get()      if (!collectionSnapshot.empty) {        return { message: `Collection '${collectionId}' already exists.` }      }  *// Create the collection*      await collectionRef.add({})      return { message: `Collection '${collectionId}' created successfully.` }    } catch (err) {      console.error(err)      throw new Error('Server error.')    }  } |
| Результат | {      "message": "Collection 'test\_222' created successfully."  } |

## 3. **db-requests/get-collection-with-name**

|  |  |
| --- | --- |
| Запрос | {      "collectionId": "test\_222"  } |
| Метод | module.exports.getCollectionWithName = async function (*request*, *reply*) {    const collectionName = *request*.headers?.collectionname || 'ip\_addresses'    const firestore = Firebase.admin.firestore().collection(collectionName)    try {      const collections = await firestore.listDocuments()      const collectionNames = collections.map((*c*) => *c*.id)      return { collections: collectionNames }    } catch (err) {      if (err.message === 'Response code 404 (Not Found)' || err.message === 'Not active') {        throw HttpError.NotFound('Not Found or Not active')      }      console.log(err)      throw HttpError.InternalServerError('Server error.')    }  } |
| Результат | {      "collections": [          "02P9ykErqwUC3UBFZ90V",          "03oDbhnzCqqzJ6iRAs2V",          "04nZgOn76ewmg53PXT0D",         "zytfaHJ7v8G6eYlBgyFf"      ]  } |

## 4. **db-requests/list-documents**

|  |  |
| --- | --- |
| Запрос | {"collectionId": "preOrders"} |
| Метод | module.exports.listDocumentsInCollection = async function (*request*, *reply*) {    try {      const collectionId = *request*.body.collectionId      const firestore = Firebase.admin.firestore()      const collectionRef = firestore.collection(collectionId)      const querySnapshot = await collectionRef.get()      const documents = querySnapshot.docs.map((*doc*) => *doc*.data())      return { documents }    } catch (err) {      if (err.message === 'Response code 404 (Not Found)' || err.message === 'Not active') {        throw HttpError.NotFound('Not Found or Not active')      }      console.log(err)      throw HttpError.InternalServerError('Server error.')    }  } |
| Результат | {      "documents": [          {              "owner": "nobody",              "preOrderId": 1253,              "description": "This is preOrder number 1253",              "createdAt": {                  "\_seconds": 1684325002,                  "\_nanoseconds": 52000000              }          }      ]  } |

## 5. **db-requests/create-document-in-collection**

|  |  |
| --- | --- |
| Запрос | {      "collectionId": "ip\_addresses",      "document": {          "ip\_address": "192.168.1.55",          "description": "nothing 55",          "owner": "Andre Pocahontas"      }  } |
| Метод | module.exports.createDocumentInCollection = async function (*request*, *reply*) {    const { collectionId, document } = *request*.body    const firestore = Firebase.admin.firestore()    try {      const collectionRef = firestore.collection(collectionId)      const documentRef = collectionRef.doc()      const timestamp = Firebase.admin.firestore.FieldValue.serverTimestamp()      const updatedDocument = { ...document, createdAt: timestamp, documentId: documentRef.id }      const existingDocument = await documentRef.get()      if (existingDocument.exists) {        throw new Error('Document with the same ID already exists.')      }      await documentRef.set(updatedDocument)      return { documentId: documentRef.id }    } catch (err) {      if (err.message === 'Response code 404 (Not Found)' || err.message === 'Not active') {        throw HttpError.NotFound('Not Found or Not active')      }      console.log(err)      throw HttpError.InternalServerError('Server error.')    }  } |
| Результат | {      "documentId": "3agFNgKrjPcwhI8sfeiC"  } |

## 6. **db-requests/update-document-in-collection**

|  |  |
| --- | --- |
| Запрос | {      "collectionId": "ip\_addresses",      "documentId": "5trp4EEANRJoPqRiIaoe",      "document": {          "ip\_address": "192.168.1.77",          "description": "nothing 77",          "new\_field": "nobody"      }  } |
| Метод | odule.exports.updateDocumentInCollection = async function (*request*, *reply*) {    const { collectionId, documentId, document } = *request*.body    const firestore = Firebase.admin.firestore()    const timestamp = Firebase.admin.firestore.FieldValue.serverTimestamp()    const updatedDocument = { ...document, updatedAt: timestamp }    try {      const collectionRef = firestore.collection(collectionId)      await collectionRef.doc(documentId).update(updatedDocument)      return { documentId }    } catch (err) {      if (err.message === 'Response code 404 (Not Found)' || err.message === 'Not active') {        throw HttpError.NotFound('Not Found or Not active')      }      console.log(err)      throw HttpError.InternalServerError('Server error.')    }  } |
| Результат | {      "documentId": "5trp4EEANRJoPqRiIaoe"  } |

## 7. **db-requests/get-document-with-id**

|  |  |
| --- | --- |
| Запрос | {      "collectionId": "ip\_addresses",      "documentId": "5trp4EEANRJoPqRiIaoe"  } |
| Метод | module.exports.getDocumentWithId = async function (*request*, *reply*) {    const { collectionId, documentId } = *request*.body    const firestore = Firebase.admin.firestore()    try {      const collectionRef = firestore.collection(collectionId)      const document = await collectionRef.doc(documentId).get()      return { document: document.data() }    } catch (err) {      if (err.message === 'Response code 404 (Not Found)' || err.message === 'Not active') {        throw HttpError.NotFound('Not Found or Not active')      }      console.log(err)      throw HttpError.InternalServerError('Server error.')    }  }  *//* |
| Результат | {      "document": {          "owner": "Andre Pocahontas",          "documentId": "5trp4EEANRJoPqRiIaoe",          "createdAt": {              "\_seconds": 1684159130,              "\_nanoseconds": 825000000          },          "new\_field": "nobody",          "description": "nothing 77",          "ip\_address": "192.168.1.77",          "updatedAt": {              "\_seconds": 1684219045,              "\_nanoseconds": 284000000          }      }  } |

## 8. **db-requests/get-documents-with-filter**

|  |  |
| --- | --- |
| Запрос | {      "collectionId": "ip\_addresses",      "documentId": "5trp4EEANRJoPqRiIaoe",      "filter": {          "field": "ip\_address",          "operator":"==",          "value": "192.168.1.77"      }  } |
| Метод | module.exports.getDocumentsWithFilter = async function (*request*, *reply*) {    const { collectionId, filter } = *request*.body    const firestore = Firebase.admin.firestore()    try {      const collectionRef = firestore.collection(collectionId)      const query = collectionRef.where(filter.field, filter.operator, filter.value)      const querySnapshot = await query.get()      const documents = querySnapshot.docs.map((*doc*) => *doc*.data())      return { documents }    } catch (err) {      if (err.message === 'Response code 404 (Not Found)' || err.message === 'Not active') {        throw HttpError.NotFound('Not Found or Not active')      }      console.log(err)      throw HttpError.InternalServerError('Server error.')    }  } |
| Результат | {      "documents": [          {              "owner": "Andre Pocahontas",              "documentId": "5trp4EEANRJoPqRiIaoe",              "createdAt": {                  "\_seconds": 1684159130,                  "\_nanoseconds": 825000000              },              "new\_field": "nobody",              "description": "nothing 77",              "ip\_address": "192.168.1.77",              "updatedAt": {                  "\_seconds": 1684350098,                  "\_nanoseconds": 14000000              }          }      ]  } |

## 9. **db-requests/delete-document-in-collection**

|  |  |
| --- | --- |
| Запрос | {      "collectionId": "ip\_addresses",      "documentId": "5trp4EEANRJoPqRiIaoe"  } |
| Метод | module.exports.deleteDocumentInCollection = async function (*request*, *reply*) {    const { collectionId, documentId } = *request*.body    const firestore = Firebase.admin.firestore()    try {      const collectionRef = firestore.collection(collectionId)      await collectionRef.doc(documentId).delete()      return { documentId }    } catch (err) {      if (err.message === 'Response code 404 (Not Found)' || err.message === 'Not active') {        throw HttpError.NotFound('Not Found or Not active')      }      console.log(err)      throw HttpError.InternalServerError('Server error.')    }  } |
| Результат | {      "documentId": "HBrb1xRd5LVkQU7SZNUe"  } |

## 11. **db-requests/generate-random-data-into-db**

|  |  |
| --- | --- |
| Запрос | {    "data": {      "collectionId": "preOrders",      "quantity": 0,      "document": {          "preOrderId": 1,          "description": "This is preOrder number ",          "owner": "nobody"      }    }  } |
| Метод | module.exports.generateRandomDataIntoDB = async function (*request*, *reply*) {    const { collectionId, document, quantity } = *request*.body.data    try {      const firestore = Firebase.admin.firestore()      const collectionRef = firestore.collection(collectionId)      const collectionSnapshot = await collectionRef.get()      if (!collectionSnapshot.empty) {        console.log(`Collection '${collectionId}' already exists.`)      } else {        await collectionRef.add({})        console.log(`Collection '${collectionId}' created.`)      }      const timestamp = Firebase.admin.firestore.FieldValue.serverTimestamp()      for (let i = 0; i < quantity; i++) {        const documentRef = collectionRef.doc()        const updatedDocument = {          ...document,          preOrderId: i,          description: `This is preOrder number ${i.toString()}`,          createdAt: timestamp,        }        const existingDocument = await documentRef.get()        if (existingDocument.exists) {          console.log('Document with the same ID already exists.')        } else {          await documentRef.set(updatedDocument)          console.log(`Document '${documentRef.id}' created.`)        }      }      return {        message: `Collection '${collectionId}' with ${quantity.toString()} documents created successfully.`,      }    } catch (err) {      console.error(err)      throw new Error('Server error.')    }  } |
| Результат | {      "message": "Collection 'preOrders' with 1999 documents created successfully."  } |